UNI-T®

Instruments.uni-trend.com

Programming Manual

UTG9000T Series Function/Arbitrary Waveform Generator

REV 00

2023.09.21

Warranty and Statement

Copyright

Trademark

File Number

20230614

Software Version

V1.13.0020
Please follow the UNI-T website to get the latest manual or contact UNI-T if any software update needed or functions added.

Statement

Certification

UNI-T certifies that products are up to the Chinese national product standards, industry product standards, ISO9001:2008 standards and ISO14001:2004 standards, and further certifies that products are up to the relevant standards of other international standard organizations.

Contact Us

Contact UNI-T if you have any problems or requirements on the manual or about the device operation.
E-mail: infosh@uni-trend.com.cn
Website: http://www.uni-trend.com

SCPI Introduction

SCPI (Standard Commands for Programmable Instruments) is a standard instrument programming language based on the existing IEEE 488.1 and IEEE 488.2 standards, following the floating point arithmetic rules of IEEE754 standard, 7-bit coding symbols of information change in ISO646 (equivalent to ASCII programming), and etc. In this section, we describe the format, notations, Parameters, abbreviation rules of the SCPI command.

Commands Format

The SCPI command in a tree hierarchy includes several subsystems, each consisting of a root keyword and one or more hierarchical keywords. The command line usually starts with a colon ":". Keywords are separated by a colon ":", followed by optional the parameter settings. The command keyword and the first the parameter are separated by a space. The command string must end with a < Newline > (<NL>) character. Add the question mark "? " to the end of command line, which usually means to query this function.

Notations

The following four notations are not the part of SCPI command, not sent with the command, but generally used to help illustrate the parameters in the command.

- Brace { }

  It usually contains multiple optional Parameters, one of which must be selected when the command is sent.
  For example, :DISPlay:GRID:MODE {FULL | GRID | CROSS | NONE} command.

- Vertical Bar |

  It is used to separate multiple The parameter options, one of which must be selected when the command is sent.
  For example, :DISPlay:GRID:MODE {FULL | GRID | CROSS | NONE} command.

- Square Bracket [ ]

  The content in square bracket (command keywords) is omitted. If The parameter is omitted, it will be set as the default value by the instrument.
  For example, in the :MEASure:NDUTy? [<source>] command, [<source>] means the current channel.

- Angle Bracket < >

  The parameter in triangular bracket must be replaced by a valid value.
  For example, DISPlay: GRID: BRIGhtness <count> command is sent as the format of DISPlay: GRID: BRIGhtness 30.

Parameters

The parameters contained in the commands in this manual can be divided into five types: Boolean, Integer, Real, Discrete, and ASCII Strings.

- Boolean Type

The parameter value is "ON" (1) or "OFF" (0).

For example, :SYSTem: LOCK {{1|ON}|{0|OFF}}

- **Integer Type**

  The parameter can be any integer value within a valid range unless otherwise noted. Note: Now, please do not set the parameter to decimal format to avoid abnormity.
  For example, The parameter < count > in the command of :DISPlay:GRID:BRIGhtness <count> can be any integer value within range of 0~100.

- **Real Type**

  The parameter can be any value within a valid range unless otherwise noted.
  For example, the value of argument<offset> in CHANnel1: OFFSet <offset>, CH1 command is real.

- **Discrete Type**

  The parameter can only be specified value or character.
  For example, The parameter in :DISPlay:GRID:MODE {FULL|GRID|CROSS|NONE} command can only be FULL, GRID,CROSS, NONE.

- **ASCII Strings**

  The string parameter can virtually contain all ASCII character sets. Strings must begin and end with paired quotes; you can use single or double quotes. The quotation separator can also be used as part of a string by typing it twice without adding any characters, such as the IP setting: SYST: COMM: LAN: IPAD "192.168.1.10".

Abbreviation Rules

All commands can identify the capital letter and small letter. You must type all capital letters in the command format when you want to abbreviate.

Data Return

Data Return is divided into single data return and batch data return, and single data returns the corresponding The parameter types, while the real type returns data in scientific notation, the part before e retaining three digits after the decimal, and the e part retains three digits. Batch data must be returned the string data in the IEEE 488.2 # format: '#' + Byte bit width[1 Byte] + ASCII value in valid data + valid data + last symbol [' \n '], e.g. # 3123XXxxxxxxXXXXXXxxx \n represents the return format of valid batch data with 123 bytes, here '3' means that the '123' occupies 3 character positions.

SCPI Details

IEEE488.2 Common Commands

*IDN?

- Command Format
  *IDN?
- Function
  Query the manufacturer name, product model, product serial number and software versioning.
- Return Format
  The query returns the manufacturer name, product model, product serial number and the software versioning separated by dot.
  Note: The returning model should be same as the nameplate.
- Example
  UNI-T Technologies, UTG5000, 000000001, 00.00.01

*RST

- Command Format
  *RST
- Function
  Restore factory settings, clear all error messages and send/receive queue buffers

SYSTem Command

It is used for the most basic operation of signal source, mainly including the operation of full keyboard lock and the system data setting.

:SYSTem:LOCK

- Command Format
  :SYSTem:LOCK {{1|ON}|{0|OFF}}
  :SYSTem:LOCK?
- Function
  Lock or unlock the full keyboard keys and touch input.
- Return Format
  The query returns the full keyboard status, and 0 in UNLOCK, 1 in LOCK.
- Example

:SYSTem:LOCK ON	Full keyboard LOCK
:SYSTem:LOCK OFF	Full keyboard UNLOCK
:SYSTem:LOCK?	Query returns 1, LOCK

:SYSTem:CONFigure

- Command Format
  :SYSTem:CONFigure <file>

:SYSTem:CONFigure?

- Function

  Read and write the configuration file, sending commands first, and send configuration file data to the signal source.

  <file> means the configuration file.

- Return Format

  The query returns the current configuration file data of signal source.

- Example

  :SYSTem:CONFigure

  Write the configuration file data into signal source and load it.

  :SYSTem:CONFigure?

  The query returns a binary stream of current configuration file data of signal source.

:SYSTem:PHASe:MODe

- Command Format

  :SYSTem:PHASe:MODe {INDependent | SYNChronization}

  :SYSTem:PHASe:MODe?

- Function

  Control the phase mode between channels. The starting phases of two channels are synchronized if it is synchronous, otherwise independent.

- Return Format

  The query returns the phase mode between channels.

- Example

  :SYSTem:PHASe:MODe INDependent

  Set the independent phase mode between channels.

  :SYSTem:PHASe:MODe?

  The query returns INDependent.

:SYSTem:LANGuage

- Command Format

  :SYSTem:LANGuage {ENGLish|CHINese}

  :SYSTem:LANGuage?

- Function

  Control the system language.

- Return Format

  The query returns system languages.

- Example

  :SYSTem:LANGuage ENGLish

  Set the system language to English.

  :SYSTem:LANGuage?

  The query returns ENGLish.

:SYSTem:BEEP

- Command Format

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

  :SYSTem:BEEP?

- Function
  Control the buzzer of system.
- Return Format
  The query returns the buzzer ON/OFF status.
- Example
  :SYSTem:BEEP ON Turn buzzer on
  :SYSTem:BEEP? The query returns 1

:SYSTem:NUMBer:FORMat

- Command Format
  :SYSTem:NUMBer:FORMat {COMMa|SPACe|NONe}
  :SYSTem:NUMBer:FORMat?
- Function
  Control the separator of system number format.
- Return Format
  The query returns the separator of system number format.
- Example
  :SYSTem:NUMBer:FORMat NONe
  Set the system number format to be none
  :SYSTem:NUMBer:FORMat?
  The query returns NONe

:SYSTem:PICTure:FORMat

- Command Format
  :SYSTem:PICTure:FORMat {BMP|JPEG|PNG}
  :SYSTem:PICTure:FORMat?
- Function
  Set the data format of images capturing and storage.
- Return Format
  The query returns the images format {BMP|JPEG|PNG}。
- Example
  :SYSTem:PICTure:FORMat PNG
  Set the image format to PNG.
  :SYSTem:PICTure:FORMat?
  The query returns PNG.

:SYSTem:BRIGhtness

- Command Format
  :SYSTem:BRIGhtness {30|40|50|60|70|80|90|100}
  :SYSTem:BRIGhtness?
- Function
  Control the brightness of system backlight.
- Return Format
  The query returns the brightness of system backlight.
- Example
  :SYSTem:BRIGhtness 30

Set the brightness of system backlight to 30%
:SYSTem:BRIGhtness?
The query returns 30

:SYSTem:SLEEP:TIMe

- Command Format
  :SYSTem:SLEEP:TIMe { CLOSe | 5MIN | 15MIN | 30MIN | 60MIN}
  :SYSTem:SLEEP:TIMe?
- Function:
  Control the sleep time of system, in "Minute" unit.
- Return Format
  The query returns the sleep time.
- Example
  :SYSTem:SLEEP:TIMe 5 MIN
  Set the system to auto-sleep in 5 minutes
  :SYSTem:SLEEP:TIMe?
  The query returns 5MIN

:SYSTem:ECLK:STATus?

- Command Format
  :SYSTem:ECLK:STATus?
- Function
  Query the clock status of external system.
- Return Format
  The query returns the clock of external system, 0 in invalid, 1 in valid.
- Example
  :SYSTem:ECLK:STATus?
  The query returns 1, the external clock is valid.

:SYSTem:CLKSource

- Command Format
  :SYSTem:CLKSource {INTernal|EXTernal }
  :SYSTem:CLKSource?
- Function
  Set the clock source of system, INTernal means the internal clock source while EXTernal means the external one.
- Return Format
  The query returns the clock source of system.
- Example
  :SYSTem:CLKSource INTernal
  Set the clock source of system to be internal.
  :SYSTem:CLKSource?
  The query returns INTernal.

:SYSTem:CLKOut

- Command Format
  :SYSTem:CLKOut {{1 | ON} | {0 | OFF}}

:SYSTem:CLKOut?

- Function

  Control the system clock ON/OFF.

- Return Format

  The query returns the system clock ON/OFF, 0 in OFF, 1 in ON. Example

  :SYSTem:CLKOut ON System clock ON

  :SYSTem:CLKOut? The query returns 1

:SYSTem:CYMometer

- Command Format

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

  :SYSTem:CYMometer?

- Function

  Control the system frequency meter ON/OFF.

- Return Format

  The query returns the system frequency meter ON/OFF, 0 in OFF, 1 in ON.

- Example

  :SYSTem:CYMometer ON System frequency meter ON

  :SYSTem:CYMometer? The query returns 1

:SYSTem:CYMometer:TRIGger:COUPling

- Command Format

  :SYSTem:CYMometer:TRIGger:COUPling {DC|AC}

  :SYSTem:CYMometer:TRIGger:COUPling?

- Function

  Set the trigger coupling mode of frequency meter. DC (Direct Current) means the AC and DC components of input signal can be passed. AC (Alternating Current) means the DC components of input signal will be blocked.

- Return Format

  The query returns {DC|AC}.

- Example

  :SYSTem:CYMometer:TRIGger:COUPling DC

  Set the coupling mode of frequency meter to DC mode

  :SYSTem:CYMometer:TRIGger:COUPling?

  The query returns DC

:SYSTem:CYMometer:TRIGger:HF

- Command Format

  :SYSTem:CYMometer:TRIGger:HF {{1|ON}|{0|OFF}}

  :SYSTem:CYMometer:TRIGger:HF?

- Function

  Set the high frequency trigger-control ON/OFF of frequency meter.

- Return Format

  The query returns the high frequency trigger-control ON/OFF of frequency meter, 0 in OFF, 1 in ON.

- Example

:SYSTem:CYMometer:TRIGger:HF ON
High frequency trigger-control of frequency meter is ON
:SYSTem:CYMometer:TRIGger:HF?
The query returns 1

:SYSTem:CYMometer:TRIGger:LEVel

- Command Format
  :SYSTem:CYMometer:TRIGger:LEVel<level>
  :SYSTem:CYMometer:TRIGger:LEVel?
- Function
  Set the trigger level value of frequency meter, with range of 0~2.5V.
- Return Format
  The query returns the setting value of <level>, in "V" unit.
- Example
  :SYST:CYM:TRIG:LEV 2
  Set the trigger level to 2V
  :SYST:CYM:TRIG:LEV?
  The query returns 2.000e000

:SYSTem:CYMometer:TRIGger:SENSitivity

- Command Format
  :SYSTem:CYMometer:TRIGger:SENSitivity <sensitivity>
  :SYSTem:CYMometer:TRIGger:SENSitivity?
- Function
  Set the trigger sensitivity of frequency meter, with range of 0~100.
- Return Format
  The query returns the setting value of <sensitivity>, in "%" unit.
- Example
  :SYST:CYM:TRIG:SENS 50
  Set the trigger sensitivity of frequency meter to 50%
  :SYST:CYM:TRIG:SENS?
  The query returns 5.000e001.

:SYSTem:CYMometer:FREQuency?

- Command Format
  :SYSTem:CYMometer:FREQuency?
- Function
  Get the currently measured frequency of frequency meter.
- Return Format
  The query returns the currently measured frequency of frequency meter, in "Hz" unit, using scientific notation to return the data.
- Example
  :SYSTem:CYMometer:FREQuency?
  The query returns 2e+3

:SYSTem:CYMometer:PERiod?

- Command Format

:SYSTem:CYMometer:PERiod?

- Function

  Get the currently measured period of frequency meter.

- Return Format

  The query returns the currently measured period of frequency meter, in "s" unit, using scientific notation to return the data.

- Example

  :SYSTem:CYMometer:PERiod? The query returns 2e-3

:SYSTem:CYMometer:DUTY?

- Command Format

  :SYSTem:CYMometer:DUTY?

- Function

  Get the currently measured duty ratio of frequency meter.

- Return Format

  The query returns currently measured duty ratio of frequency meter, in "%" unit.

- Example

  :SYSTem:CYMometer:DUTY?

  The query returns 20, meaning the duty ratio 20%

:SYSTem:CYMometer:PWIDTh?

- Command Format

  :SYSTem:CYMometer:PWIDTh?

- Function

  Get the currently measured positive pulse width of frequency meter.

- Return Format

  The query returns the currently measured positive pulse width of frequency meter, in "s" unit.

- Example

  :SYSTem:CYMometer:PWIDTh?

  The query returns 1e-3, meaning the duty ratio 1ms

:SYSTem:CYMometer:NWIDTh?

- Command Format

  :SYSTem:CYMometer:NWIDTh?

- Function

  Get the currently measured negative pulse width of frequency meter.

- Return Format

  The query returns the currently measured negative pulse width of frequency meter, and in "s" unit.

- Example

  :SYSTem:CYMometer:NWIDTh?

  The query returns 1e-3, meaning the duty ratio 1ms

:SYSTem:COMMunicate:LAN:APPLy

- Command Format

  :SYSTem:COMMunicate:LAN:APPLy

- Function

  Take the current network parameter setting into effect immediately.

:SYSTem:COMMunicate:LAN:GATEway

- Command Format
  :SYSTem:COMMunicate:LAN:GATEway <gateway>
  :SYSTem:COMMunicate:LAN:GATEway?
- Function
  Set the default gateway. <gateway> is The parameter of ASCII character string, with xxx.xxx.xxx.xxx format.
- Return Format
  The query returns the default gateway.
- Example
  :SYST:COMM:LAN:GATE "192.168.1.1"
  Set the default gateway to 192.168.1.1
  :SYST:COMM:LAN:GATE?
  The query returns 192.168.1.1

:SYSTem:COMMunicate:LAN:SMASK

- Command Format
  :SYSTem:COMMunicate:LAN:SMASK <submask>
  :SYSTem:COMMunicate:LAN:SMASK?
- Function
  Set the subnet mask. <submask> is The parameter of ASCII character string, with xxx.xxx.xxx.xxx format.
- Return Format
  The query returns the subnet mask.
- Example
  :SYST:COMM:LAN:SMASK "255.255.255.0"
  Set the subnet mask to 255.255.255.0
  :SYST:COMM:LAN:SMASK?
  The query returns 255.255.255.0

:SYSTem:COMMunicate:LAN:IPADdress

- Command Format
  :SYSTem:COMMunicate:LAN:IPADdress <ip>
  :SYSTem:COMMunicate:LAN:IPADdress?
- Function
  Set the IP address. <ip> is The parameter of ASCII character string, with xxx.xxx.xxx.xxx format.
- Return Format
  The query returns the IP address.
- Example
  :SYST:COMM:LAN:IPAD "192.168.1.10"
  Set the IP address to 192.168.1.10
  :SYST:COMM:LAN:IPAD?
  The query returns 192.168.1.10

:SYSTem:COMMunicate:LAN:DHCP

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

:SYSTem:COMMunicate:LAN:DHCP?

- Function
  Switch the configuration mode (Auto IP and Manual IP).
- Return Format
  The query returns the dynamic configuration mode, 0 in Manual IP, 1 in Auto IP.
- Example
  :SYST:COMM:LAN:DHCP ON
  IP dynamic configuration is ON
  :SYST:COMM:LAN:DHCP?
  The query returns 1

:SYSTem:COMMunicate:LAN:MAC?

- Command Format
  :SYSTem:COMMunicate:LAN:MAC?
- Return Format
  The query returns the MAC address.
- Example
  :SYST:COMM:LAN:MAC?
  The query returns 00-2A-A0-AA-E0-56

CHANnel Command

Set the related functions of signal source channels.

:CHANnel<n>:MODE

- Command Format
  :CHANnel<n>:MODE {CONTinue | MODulation| SWEep| BURSt }
  :CHANnel<n>:MODE?
- Function
  Set the specified channel signal mode, with modes of CONTinue, MODulation, SWEep, and BURSt.
  <n>: Channel No., n value 1, 2, 3, 4.
- Return Format
  The query returns the specified channel signal mode.
- Example
  :CHANnel1:MODE MODulation
  Set the signal mode of channel 1 to Modulation
  :CHANnel1:MODE?
  The query returns MODulation

:CHANnel<n>:OUTPut

- Command Format
  :CHANnel<n>:OUTPut {{1|ON}|{0|OFF}}
  :CHANnel<n>:OUTPut?
- Function
  Set the specified channel output ON/OFF.
  <n>: Channel No., n value 1, 2, 3, 4.

- Return Format

  The query returns the specified channel output, 0 in OFF, 1 in ON.

- Example

  :CHANnel1:OUTPut ON Set the channel 1 output ON

  :CHANnel1:OUTPut? The query returns 1

:CHANnel<n>:INVersion

- Command Format

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

  :CHANnel<n>:INVersion?

- Function

  Set the specified channel reverse ON/OFF.

  <n>: Channel No., n value 1, 2, 3, 4.

- Return Format

  The query returns the specified channel reverse, 0 in OFF, 1 in ON.

- Example

  :CHANnel1:INVersion ON

  Set the reverse output of channel 1 ON

  :CHANnel1:INVersion?

  The query returns 1

:CHANnel<n>:OUTPut:SYNC

- Command Format

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

  :CHANnel<n>:OUTPut:SYNC?

- Function

  Set the sync output of channel.

  Note: Only one sync output interface in the device, and can only open the sync output of one channel.

  <n>: Channel No., n value 1, 2, 3, 4.

- Return Format

  The query returns the sync output of specified channel, 0 in OFF, 1 in ON.

- Example

  :CHANnel1:OUTPut:SYNC ON

  Set the sync output of channel 1 ON

  :CHANnel1:OUTPut:SYNC?

  The query returns 1.

:CHANnel<n>:OUTPut:SYNC:INVersion

- Command Format

  :CHANnel<n>:OUTPut:SYNC:INVersion {{1|ON}|{0|OFF}}

  :CHANnel<n>:OUTPut:SYNC:INVersion?

- Function

  Set the sync output phase reversal of channel.

  Note: Only one sync output interface in the device, and can only open the sync output of one channel.

  <n>: Channel No., n value 1, 2, 3, 4.

- Return Format

The query returns the reverse status of specified channel, 0 in OFF, 1 in ON.

- Example

:CHANnel1:OUTPut:SYNC:INVersion ON

Set the sync output phase reversal of channel 1 ON.

:CHANnel1:OUTPut:SYNC:INVersion?

The query returns 1.

:CHANnel<n>:LIMit:ENABle

- Command Format

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

:CHANnel<n>:LIMit:ENABle?

- Function

Set the amplitude limiting ON/OFF of specified channel.

<n>: Channel No., n value 1, 2, 3, 4.

- Return Format

The query returns the amplitude limiting status of specified channel.

- Example

:CHANnel1:LIMit:ENABle ON

Set the amplitude limiting of channel 1 ON.

:CHANnel1:LIMit:ENABle?

The query returns 1.

:CHANnel<n>:LIMit:LOWer

- Command Format

:CHANnel<n>:LIMit:LOWer {<voltage>}

:CHANnel<n>:LIMit:LOWer?

- Function

Set the lower amplitude limit of specified channel.

<voltage> means the voltage, and its unit is the specified unit of current channel.

<n>: Channel No., n value 1, 2, 3, 4.

- Return Format

The query returns the lower amplitude limit of specified channel, using scientific notation to return.

- Example

:CHANnel1:LIMit:LOWer 2

Set the lower amplitude limit of channel 1 to 2V

:CHANnel1:LIMit:LOWer?

The query returns 2e+0

:CHANnel<n>:LIMit:UPPer

- Command Format

:CHANnel<n>:LIMit:UPPer {<voltage>}

:CHANnel<n>:LIMit:UPPer?

- Function

Set the upper amplitude limit of specified channel.

<voltage> means voltage, and its unit is the specified unit of current channel.

<n>: Channel No., n value 1, 2, 3, 4.

- Return Format
  The query returns the upper amplitude limit of specified channel, using scientific notation to return.
- Example
  :CHANnel1:LIMit:UPPer 2
  Set the upper amplitude limit of channel 1 to 2V
  :CHANnel1:LIMit:UPPer?
  The query returns 2e+0

:CHANnel<n>:AMPLitude:UNIT

- Command Format
  :CHANnel<n>:AMPLitude:UNIT {VPP | VRMS | DBM}
  :CHANnel<n>:AMPLitude:UNIT?
- Function
  Set the unit of output amplitude in specified channel.
  <n>: Channel No., n value 1, 2, 3, 4.
- Return Format
  The query returns the unit of output amplitude in specified channel.
- Example
  :CHANnel1:AMPLitude:UNIT VPP
  Set the unit of output amplitude in channel 1 to VPP
  :CHANnel1:AMPLitude:UNIT?
  The query returns VPP

:CHANnel<n>:LOAD

- Command Format
  :CHANnel<n>:LOAD <resistance>
  :CHANnel<n>:LOAD?
- Function
  Set the output load of specified channel.
  <resistance> means the load resistance, in "Ω" unit.
  <n>: Channel No., n value 1, 2, 3, 4.
  Note: The resistance value should be within the range of 1~10000, and the 10000 is for high resistance.
- Return Format
  The query returns the load resistance of specified channel, using scientific notation to return.
- Example
  :CHANnel1:LOAD 50
  Set the output load of channel 1 to 50Ω
  :CHANnel1:LOAD?
  The query returns 50e+0

:CHANnel<n>:PNCode

- Command Format
  :CHANnel<n>:PNCode <code>
  :CHANnel<n>:PNCode?
- Function
  Set the PN code of specified channel, and the command is only effective for the wave with PN code function.

<code> means PN code, as following showed,
{PN3|PN5|PN7|PN9|PN11|PN13|PN15|PN17|PN21|PN23|PN25|PN27|PN29|PN31|PN33}
<n>: Channel No., n value 1, 2, 3, 4.

- Return Format

The query returns the PN code of specified channel.

- Example

:CHANnel1:PNCode PN9
Set the PN code of channel 1 to PN9
:CHANnel1:PNCode?
The query returns PN9

:CHANnel<n>:TRIGger:SOURce

- Command Format

:CHANnel<n>:TRIGger:SOURce {INTernal|EXTRise|EXTFall|MANual}
:CHANnel<n>:TRIGger:SOURce?

- Function

Set the trigger source of specified channel, and the command is only effective for the frequency sweep and trigger.
<n>: Channel No., n value 1, 2, 3, 4.

- Return Format

The query returns the trigger source of specified channel.

- Example

:CHANnel1:TRIGger:SOURce INTernal
Set the trigger source of channel 1 to be internal
:CHANnel1:TRIGger:SOURce?
The query returns INTernal

:CHANnel<n>:TRIGger:OUTPut

- Command Format

:CHANnel<n>:TRIGger:OUTPut {CLOSe|RISe|FALL}
:CHANnel<n>:TRIGger:OUTPut?

- Function

Set the trigger output mode of specified channel, and the command is only effective for the frequency sweep and trigger.
<n>: Channel No., n value 1, 2, 3, 4.

- Return Format

The query returns the trigger output mode of specified channel.

- Example

:CHANnel1:TRIGger:OUTPut RISe
Set the trigger output of channel 1 to rising edge
:CHANnel1:TRIGger:OUTPut?
The query returns RISe

:CHANnel<n>:NS

- Command Format

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

- Function

  Set the noise superimposition of specified channel ON/OFF, and the command is only effective for the wave with noise superimposition function.

  <n>: Channel No., n value 1, 2, 3, 4.
- Return Format

  The query returns the output status of specified channel, 0 in OFF, 1 in ON.
- Example

  :CHANnel1:NS ON

  Set the noise superimposition of channel 1 ON

  :CHANnel1:NS?

  The query returns 1

:CHANnel<n>:NS:SNR:UNIT

- Command Format

  :CHANnel<n>:NS:SNR:UNIT {DBM | RATIO }

  :CHANnel<n>:NS:SNR:UNIT?
- Function

  Set the noise superimposition SNR unit of channel, and the command is only effective for the wave with noise superimposition function.

  <n>: Channel No., n value 1, 2, 3, 4.
- Return Format

  The query returns the noise superimposition SNR unit of specified channel.
- Example

  :CHANnel1:NS:SNR:UNIT DBM

  Set the noise superimposition SNR unit of channel 1 to dB

  :CHANnel1:NS:SNR:UNIT?

  The query returns DBM

:CHANnel<n>:NS:SNR

- Command Format

  :CHANnel<n>:NS:SNR <value>

  :CHANnel<n>:NS:SNR?
- Function

  Set the noise superimposition SNR value, and the command is only effective for the wave with noise superimposition function.

  <n>: Channel No., n value 1, 2, 3, 4.
- Return Format

  The query returns the noise superimposition SNR value of channel, using scientific notation to return.
- Example

  :CHANnel1:NS:SNR 3

  Set the noise superimposition SNR of channel 1 to 3dB

  :CHANnel1:NS:SNR?

  The query returns 3e+0

:CHANnel<n>:MERge

- Command Format

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

:CHANnel<n>:MERge?

- Function

Set the merge output ON/OFF of specified channel.

Merge channel 1 & 2, and only output from channel 1 and channel 2.

Merge channel 3 & 4 and only output from channel 3 and channel 4.

<n>: Channel No., n value 1, 2, 3, 4.

- Return Format

The query returns the merge output ON/OFF status of specified channel.

- Example

:CHANnel1:MERge ON

Merge the signal of channel 1 & 2, and the signal can only be output from channel 1

:CHANnel1:MERge? The query returns 1

:CHANnel:COUPle<m>:FREQuency

- Command Format

:CHANnel:COUPle<m>:FREQuency {{1|ON}|{0|OFF}}

:CHANnel:COUPle<m>:FREQuency?

- Function

Set the channel frequency coupling ON/OFF, only two types of Channel Coupling 1 & 2 and Channel Coupling 3 & 4 existed.

<m>: Channel No., m value 1, 2.

1 means the Channel Coupling 1 & 2, 2 means the Channel Coupling 3 & 4.

- Return Format

The query returns the channel coupling ON/OFF status, 0 in OFF, 1 in ON.

- Example

:CHANnel:COUPle1:FREQuency ON

Channel coupling 1 & 2 is ON.

:CHANnel:COUPle1:FREQuency? The query returns 1

:CHANnel:COUPle<m>:FREQuency:SCALe

- Command Format

:CHANnel:COUPle<m>:FREQuency:SCALe <scale>

:CHANnel:COUPle<m>:FREQuency:SCALe?

- Function

Set the channel coupling frequency ratio, only two types of Channel Coupling 1 & 2 and Channel Coupling 3 & 4 existed.

<scale >: Coupling frequency ratio.

<m>: Channel No., m value 1, 2.

1 means the Channel Coupling 1 & 2, 2 means the Channel Coupling 3 & 4.

- Return Format

The query returns the channel coupling frequency ratio and the scientific notation.

- Example

:CHANnel:COUPle1:FREQuency:SCALe 0.1

Set the coupling ratio of channel 2 and 1 to 0.1

:CHANnel:COUPle1:FREQuency:SCALe?

The query returns 1e-1

:CHANnel:COUPle<m>:FREQuency:DEV

- **Command Format**

  :CHANnel:COUPle<m>:FREQuency:DEV <dev >

  :CHANnel:COUPle<m>:FREQuency:DEV?

- **Function**

  Set the channel coupling frequency deviation, only two types of Channel Coupling 1 & 2 and Channel Coupling 3 & 4 existed.

  <scale >: Coupling frequency deviation, in "Hz" unit.

  <m>: Channel No., m value 1, 2.

  1 means the Channel Coupling 1 & 2, 2 means the Channel Coupling 3 & 4.

- **Return Format**

  The query returns the channel coupling frequency deviation and the scientific notation.

- **Example**

  :CHANnel:COUPle1:FREQuency:DEV 100

  Set the coupling deviation between channel 2 and 1 to 100Hz

  :CHANnel:COUPle1:FREQuency:DEV?

  The query returns 1e+2

:CHANnel:COUPle<m>:PHASe

- **Command Format**

  :CHANnel:COUPle<m>:PHASe {{1|ON}|{0|OFF}}

  :CHANnel:COUPle<m>:PHASe?

- **Function**

  Set the channel phase coupling ON/OFF, only two types of Channel Coupling 1 & 2 and Channel Coupling 3 & 4 existed.

  <m>: Channel No., m value 1, 2.

  1 means the Channel Coupling 1 & 2, 2 means the Channel Coupling 3 & 4.

- **Return Format**

  The query returns the channel coupling ON/OFF status, 0 in OFF, 1 in ON.

- **Example**

  :CHANnel:COUPle1:PHASe ON

  Channel coupling 1 & 2 is ON

  :CHANnel:COUPle1:PHASe? The query returns 1

:CHANnel:COUPle<m>:PHASe:SCALe

- **Command Format**

  :CHANnel:COUPle<m>:PHASe:SCALe <scale>

  :CHANnel:COUPle<m>:PHASe:SCALe?

- **Function**

  Set the channel coupling phase ratio, only two types of Channel Coupling 1 & 2 and Channel Coupling 3 & 4 existed.

  <scale >: Coupling phase ratio

  <m>: Channel No., m value 1, 2.

  1 means the Channel Coupling 1 & 2, 2 means the Channel Coupling 3 & 4.

- **Return Format**

  The query returns the channel coupling phase ratio, and the scientific notation.

- **Example**

:CHANnel:COUPle1:PHASe:SCALe 0.1
Set the coupling ratio of channel 2 and 1 to 0.1
:CHANnel:COUPle1:PHASe:SCALe? The query returns 1e-1

:CHANnel:COUPle<m>:PHASe:DEV

- Command Format
  :CHANnel:COUPle<m>:PHASe:DEV <dev >
  :CHANnel:COUPle<m>:PHASe:DEV?
- Function
  Set the channel coupling phase deviation, only two types of Channel Coupling 1 & 2 and Channel Coupling 3 & 4 existed.
  <scale >: Coupling phase deviation, in " ° " unit.
  <m>: Channel No., m value 1, 2.
  1 means the Channel Coupling 1 & 2, 2 means the Channel Coupling 3 & 4.
- Return Format
  The query returns the channel coupling phase deviation, and the scientific notation.
- Example
  :CHANnel:COUPle1:PHASe:DEV 100
  Set the coupling deviation of channel 2 and 1 to 100°
  :CHANnel:COUPle1:PHASe:DEV?
  The query returns 1e+2

:CHANnel:COUPle<m>:AMPLitude

- Command Format
  :CHANnel:COUPle<m>:AMPLitude {{1|ON}|{0|OFF}}
  :CHANnel:COUPle<m>:AMPLitude?
- Function
  Set the channel amplitude coupling ON/OFF, only two types of Channel Coupling 1 & 2 and Channel Coupling 3 & 4 existed.
  <m>: Channel No., m value 1, 2.
  1 means the Channel Coupling 1 & 2, 2 means the Channel Coupling 3 & 4.
- Return Format
  The query returns the channel amplitude coupling ON/OFF status, 0 in OFF, 1 in ON.
- Example
  :CHANnel:COUPle1:AMPLitude ON
  Coupling amplitude of Channel 1 & 2 is ON
  :CHANnel:COUPle1:AMPLitude?
  The query returns 1

:CHANnel:COUPle<m>:AMPLitude:SCALe

- Command Format
  :CHANnel:COUPle<m>:AMPLitude:SCALe <scale>
  :CHANnel:COUPle<m>:AMPLitude:SCALe?
- Function
  Set the channel coupling amplitude ratio, only two types of Channel Coupling 1 & 2 and Channel Coupling 3 & 4 existed.
  <scale >: Coupling amplitude ratio.

<m>: Channel No., m value 1, 2.

1 means the Channel Coupling 1 & 2, 2 means the Channel Coupling 3 & 4.

- Return Format

The query returns the channel coupling amplitude ratio, and the scientific notation.

- Example

:CHANnel:COUPle1:AMPLitude:SCALe 0.1

Set the coupling ratio of channel 2 and 1 to 0.1

:CHANnel:COUPle1:AMPLitude:SCALe?

The query returns 1e-1

:CHANnel:COUPle<m>:AMPLitude:DEV

- Command Format

:CHANnel:COUPle<m>:AMPLitude:DEV <dev >

:CHANnel:COUPle<m>:AMPLitude:DEV?

- Function

Set the channel coupling amplitude deviation, only two types of Channel Coupling 1 & 2 and Channel Coupling 3 & 4 existed.

<scale >: Coupling amplitude deviation, in "Vpp" unit.

<m>: Channel No., m value 1, 2.

1 means the Channel Coupling 1 & 2, 2 means the Channel Coupling 3 & 4.

- Return Format

The query returns the channel coupling amplitude deviation, and the scientific notation.

- Example

:CHANnel:COUPle1:AMPLitude:DEV 1

Set the coupling deviation of channel 2 and 1 to 1Vpp

:CHANnel:COUPle1:AMPLitude:DEV?

The query returns 1e+2

:CHANnel<n>:SELect

- Command Format

:CHANnel<n>:SELect

:CHANnel<n>:SELect?

- Function

Select the channel.

<n>: {1|2|3|4}, respectively means {CH1|CH2|CH3|CH4}.

- Return Format

The query returns 1 or 0, which is ON or OFF.

- Example

:CHAN1:SELect

Select the channel 1

:CHAN1:SELect?

The query returns 1, meaning the channel selected

Continuity

:CHANnel<n>:BASE:WAVe

- Command Format
  :CHANnel<n>:BASE:WAVe { SINe | SQUare | PULSe | RAMP | ARB | NOISe | DC | HARMonic | PRBS }
  :CHANnel<n>:BASE:WAVe?
- Function
  Set the fundamental wave types of specified channel, including the Sine Wave, Square Wave, Pulse Wave, Ramp Wave, Arbitrary Wave, Noise, DC, Harmonic Wave, and Pseudo-Random Binary Sequence.
  <n>: Channel No, n value 1, 2, 3, 4.
- Return Format
  The query returns the fundamental wave types of specified channel.
- Example
  :CHANnel1:BASE:WAVe SINe
  Set the channel 1 to sine wave
  :CHANnel1:BASE:WAVe?
  The query returns SINe

:CHANnel<n>:BASE:FREQuency

- Command Format
  :CHANnel<n>:BASE:FREQuency {<freq>}
  :CHANnel<n>:BASE:FREQuency?
- Function
  Set the output frequency of specified channel.
  <freq> means the frequency value, in "Hz" unit. (1e-6s ~ current max. frequency of wave)
  <n>: Channel No, n value 1, 2, 3, 4.
- Return Format
  The query returns the output frequency of specified channel, using scientific notation to return.
- Example
  :CHANnel1:BASE:FREQuency 2000
  Set the output frequency of channel 1 to 2KHz
  :CHANnel1:BASE:FREQuency?
  The query returns 2e+3

:CHANnel<n>:BASE:PERiod

- Command Format
  :CHANnel<n>:BASE:PERiod {<period>}
  :CHANnel<n>:BASE:PERiod?
- Function
  Set the output period of specified channel.
  <period> means period, in "s" unit.
  If sine wave: max. ~ 1e3s
  <n>: Channel No, n value 1, 2, 3, 4.
- Return Format
  The query returns the upper amplitude limit of specified channel, using scientific notation to return.
- Example
  :CHANnel1:BASE:PERiod 0.002
  Set the output period of channel 1 to 2ms

:CHANnel1:BASE:PERiod?
The query returns 2e-3

:CHANnel<n>:BASE:PHASe

- Command Format
  :CHANnel<n>:BASE:PHASe {<phase>}
  :CHANnel<n>:BASE:PHAse?
- Function
  Set the output phase of specified channel.
  <phase> means the phase, in "°" unit, range of -360~360.
  <n>: Channel No, n value 1, 2, 3, 4.
- Return Format
  The query returns the output phase of specified channel.
- Example
  :CHANnel1:BASE:PHAse 20
  Set the output phase of channel 1 to 20°
  :CHANnel1:BASE:PHAse?
  The query returns 20

:CHANnel<n>:BASE:AMPLitude

- Command Format:
  :CHANnel<n>:BASE:AMPLitude {<amp>}
  :CHANnel<n>:BASE:AMPLitude?
- Function
  Set the output amplitude of specified channel.
  <amp> means voltage, unit is the specified one of current channel. 1mVpp ~ is the max.output in the current load condition.
  When the current unit is VPP, the current max.load=current load*20/ (50+current load)
  <n>: Channel No, n value 1, 2, 3, 4.
- Return Format
  The query returns the output amplitude of specified channel, using scientific notation to return.
- Example
  :CHANnel1:BASE:AMPLitude 2
  Set the output amplitude of channel 1 to 2V
  :CHANnel1:BASE:AMPLitude?
  The query returns 2e+0

:CHANnel<n>:BASE:OFFSet

- Command Format
  :CHANnel<n>:BASE:OFFSet {<voltage>}
  :CHANnel<n>:BASE:OFFSet?
- Function
  Set the DC output offset of specified channel.
  <voltage> means voltage, in "V" unit, range of 0~±max. DC of current load.
  Max.DC of current load= current load*10/ (50+ current load)-current min.AC/2.
  Min.AC is 2mVpp, value 0 in DC mode.
  <n>: Channel No, n value 1, 2, 3, 4.

- Return Format

  The query returns the DC output offset of specified channel, using scientific notation to return.

- Example

  :CHANnel1:BASE:OFFSet 2

  Set the DC output offset of channel 1 to 2V

  :CHANnel1:BASE:OFFSet?

  The query returns 2e+0

:CHANnel<n>:BASE:HIGH

- Command Format

  :CHANnel<n>:BASE:HIGH { <voltage>}

  :CHANnel<n>:BASE:HIGH?

- Function

  Set the high signal output value of specified channel.

  <voltage> means voltage and its unit is the specified unit of current channel.

  <n>: Channel No, n value 1, 2, 3, 4.

- Return Format

  The query returns the high signal output value of specified channel, using scientific notation to return.

- Example

  :CHANnel1:BASE:HIGH 2

  Set the high signal output value of channel 1 to 2V

  :CHANnel1:BASE:HIGH?

  The query returns 2e+0

:CHANnel<n>:BASE:LOW

- Command Format

  :CHANnel<n>:BASE:LOW { <voltage>}

  :CHANnel<n>:BASE:LOW?

- Function

  Set the low signal output value of specified channel.

  <voltage> means voltage and its unit is the specified unit of current channel.

  <n>: Channel No., n value 1, 2, 3, 4.

- Return Format

  The query returns the low signal output value of specified channel, using scientific notation to return.

- Example

  :CHANnel1:BASE:LOW 2

  Set the low signal output value of channel 1 to 2V

  :CHANnel1:BASE:LOW?

  The query returns 2e+0

:CHANnel<n>:BASE:DUTY

- Command Format

  :CHANnel<n>:BASE:DUTY { <duty>}

  :CHANnel<n>:BASE:DUTY?

- Function

  Set the duty ratio of signal output in specified channel.

  <duty> means the duty ratio, in "%" unit, range of 0~100.

<n>: Channel No., n value 1, 2, 3, 4.

- Return Format

  The query returns the duty ratio of signal output in specified channel.

- Example

  :CHANnel1:BASE:DUTY 20

  Set the signal output duty ratio of channel 1 to 20%

  :CHANnel1:BASE:DUTY?

  The query returns 20

:CHANnel<n>:BASE:ARB

- Command Format

  :CHANnel<n>:BASE:ARB <source>,<filename>

  :CHANnel<n>:BASE:ARB?

- Function

  Set the specified channel to load arbitrary wave data of a file in the condition of any fundamental wave source.

  <n>: Channel No., n value 1, 2, 3, 4.

  <source>: {INTernal|EXTernal|USER}, internal, external, and user.

  <filename>: Arbitrary wave filename.

- Example

  :CHANnel1:BASE:ARB INTernal, "test.bsv"

:CHANnel<n>:RAMP:SYMMetry

- Command Format

  :CHANnel<n>:RAMP:SYMMetry { < symmetry >}

  :CHANnel<n>:RAMP:SYMMetry?

- Function

  Set the signal output symmetry of ramp wave in specified channel.

  < symmetry > means symmetry, in "%" unit, range of 0~100.

  <n>: Channel No., n value 1, 2, 3, 4.

- Return Format

  The query returns the signal output symmetry of ramp wave in specified channel.

- Example

  :CHANnel1:RAMP:SYMMetry 20

  Set the ramp wave signal symmetry of channel 1 to 20%

  :CHANnel1:RAMP:SYMMetry?

  The query returns 20

:CHANnel<n>:PULSe:RISe

- Command Format

  :CHANnel<n>:PULSe:RISe {<width>}

  :CHANnel<n>:PULSe:RISe?

- Function

  Set the rising edge pulse width of signal pulse wave in specified channel.

  <width> means the pulse width, in "s" unit.

  <n>: Channel No., n value 1, 2, 3, 4.

- Return Format

The query returns the rising edge pulse width of signal pulse wave in specified channel, using scientific notation to return.

- Example

:CHANnel1:PULSe:RISe 0.002
Set the rising edge pulse width of channel 1 signal to 2ms
:CHANnel1:PULSe:RISe?
The query returns 2e-3

:CHANnel<n>:PULSe:FALL

- Command Format

:CHANnel<n>:PULSe:FALL {<width>}
:CHANnel<n>:PULSe:FALL?

- Function

Set the falling edge pulse width of signal pulse wave in specified channel.
<width> means the pulse width, in "s" unit.
<n>: Channel No., n value 1, 2, 3, 4.

- Return Format

The query returns the falling edge pulse width of signal pulse wave in specified channel, using scientific notation to return.

- Example

:CHANnel1:PULSe:FALL 0.002
Set the falling edge pulse width of channel 1 signal to 2ms.
:CHANnel1:PULSe:FALL?
The query returns 2e-3.

:CHANnel<n>:PRBS:BITRatio

- Command Format

:CHANnel<n>:PRBS:BITRatio <ratio>
:CHANnel<n>:PRBS:BITRatio?

- Function

Set the specified pseudo-random bit ratio, and the command is only effective for the wave with bit ratio function.
< ratio > means bit ratio, in "bps" unit.
<n>: Channel No., n value 1, 2, 3, 4.

- Return Format

The query returns the specified pseudo-random bit ratio, using scientific notation to return.

- Example

:CHANnel1:PRBS:BITRatio 1000000
Set the bit ratio of channel 1 to 100Kbps
:CHANnel1:PRBS:BITRatio?
The query returns 1e+6

:CHANnel<n>:NOISe:BANDwith

- Command Format

:CHANnel<n>:NOISe:BANDwith {<width>}
:CHANnel<n>:NOISe:BANDwith?

- Function

Set the noise signal bandwidth of specified channel.
<width> means the bandwidth, in "Hz" unit.
<n>: Channel No., n value 1, 2, 3, 4.

- Return Format

The query returns the noise signal bandwidth of specified channel, using scientific notation to return.

- Example

:CHANnel1:NOISe:BANDwith 2MHz
Set the noise signal bandwidth of channel 1 to 2MHz
:CHANnel1:NOISe:BANDwith?
The query returns 2e+6

:CHANnel<n>:HARMonic:TYPe?

- Command Format

:CHANnel<n>:HARMonic:TYPe {ODD|EVEN|ALL|USER}
:CHANnel<n>:HARMonic:TYPe?

- Function

Set the harmonic type of specified channel.
<n>: Channel No., n value 1, 2, 3, 4.

- Return Format

The query returns the harmonic type of specified channel.

- Example

:CHANnel1:HARMonic:TYPe ODD
Set the harmonic wave of channel 1 to be odd
:CHANnel1:HARMonic:TYPe?
The query returns ODD

:CHANnel<n>:HARMonic:TOTal:ORDer?

- Command Format

:CHANnel<n>:HARMonic:TOTal:ORDer <order>
:CHANnel<n>:HARMonic:TOTal:ORDer?

- Function

Set the max.harmonic orders of specified channel.
< order >: harmonic order, range of 2~16.
<n>: Channel No., n value 1, 2, 3, 4.

- Return Format

The query returns the max.harmonic order of specified channel, and the integer data.

- Example

:CHANnel1:HARMonic:TOTal:ORDer 2
Set the max.harmonic of channel 1 to 2 orders
:CHANnel1:HARMonic:TOTal:ORDer?
The query returns 2

:CHANnel<n>:HARMonic:USER:TYPe?

- Command Format

:CHANnel<n>:HARMonic:USER:TYPe #H<order>
:CHANnel<n>:HARMonic:USER:TYPe?

- Function

Set the user harmonic type of specified channel.

< order >: User harmonic type, #H is a hexadecimal number. X0111 1111 1111 1111 bit respectively means the harmonic switch.

<n>: Channel No., n value 1, 2, 3, 4.

- Return Format

The query returns the user harmonic type of specified channel, and the integer data.

- Example

:CHANnel1:HARMonic:USER:TYPe #H7FFF

Set the user harmonic type of channel 1

:CHANnel1:HARMonic:USER:TYPe?

The query returns 32767

:CHANnel<n>:HARMonic:ORDer<m>:AMPLitude?

- Command Format

:CHANnel<n>:HARMonic:ORDer<m>:AMPLitude <amp>

:CHANnel<n>:HARMonic:ORDer<m>:AMPLitude?

- Function

Set the amplitude value of specified harmonic order in specified channel.

< amp >: Amplitude value, in "Vpp" unit.

<n>: Channel No., n value 1, 2, 3, 4.

<m>: Harmonic order, m value 2~16.

- Return Format

The query returns the amplitude value of specified harmonic order in specified channel, using scientific notation to return.

- Example

:CHANnel1:HARM:ORDER2:AMPL 0.02

Set the amplitude of 2 harmonic orders in channel 1 to 20mVpp

:CHANnel1:HARM:ORDER2:AMPL?

The query returns 2e-2

:CHANnel<n>:HARMonic:ORDer<m>:PHASe?

- Command Format

:CHANnel<n>:HARMonic:ORDer<m>:PHASe <phase>

:CHANnel<n>:HARMonic:ORDer<m>:PHASe?

- Function

Set the phase value of specified harmonic order in specified channel.

<phase>: Phase value, in "°" unit.

<n>: Channel No., n value 1, 2, 3, 4.

<m>: Harmonic order, m value 2~16.

- Return Format

The query returns the phase value of specified harmonic order in specified channel, using scientific notation to return.

- Example

:CHANnel1:HARM:ORDer2:PHASe 20

Set the phase value of 2 harmonic orders in channel 1 to 20°

:CHANnel1:HARM:ORDer2:PHASe?

The query returns 2e+1

:CHANnel<n>:ARB:MODe

- Command Format
  :CHANnel<n>:ARB:MODe {DDS | POINTS }
  :CHANnel<n>:ARB:MODe?
- Function
  Set the arbitrary wave output mode of specified channel, with two modes of DDS and Point by Point.
  <n>: Channel No., n value 1, 2.
- Return Format
  The query returns the arbitrary wave of specified channel.
- Example
  :CHANnel1:ARB:MODe DDS
  Set the arbitrary wave mode of channel 1 to DDS
  :CHANnel1:ARB:MODe?
  The query returns DDS

:CHANnel<n>:ARB:FILTer

- Command Format
  :CHANnel<n>:ARB:FILTer {ZEROHOLD | LINE }
  :CHANnel<n>:ARB:FILTer?
- Function
  Set the output interpolation of arbitrary wave in specified channel, with two modes of Zero-Order Preservation and Linear Interpolation.
  <n>: Channel No., n value 1, 2.
- Return Format
  The query returns the output interpolation of arbitrary wave in specified channel.
- Example
  :CHANnel1:ARB:FILTer LINE
  Set the arbitrary wave mode of channel 1 to linear interpolation
  :CHANnel1:ARB:FILTer?
  The query returns LINE.

Modulation

:CHANnel<n>:MODulate:TYPe

- Command Format
  :CHANnel<n>:MODulate:TYPe <type>
  :CHANnel<n>:MODulate:TYPe?
- Function
  Set the modulation mode of specified channel signal.
  <type>: {AM|BAM|QAM|ASK|FM|FSK|ThreeFSK|FourFSK|PM|PSK|BPSK|QPSK|OSK|PWM|SUM}
  Amplitude Modulation, Double Side Amplitude Modulation, Quadrature Amplitude Modulation, Amplitude Shift Keying, Frequency Modulation, Frequency Shift Keying, Three Frequency Shift Keying, Four Frequency Shift Keying, Phase Modulation, Phase Shift Keying, Bi-Phase Shift Keying, Quad -phase Shift Keying, Oscillation Shift Keying, Pulse Width Modulation, Sum Modulation.
  <n>: Channel No., n value 1, 2, 3, 4.
- Return Format

The query returns the modulation mode of specified channel signal.

- Example

:CHANnel1:MODulate:TYPe AM

Set the signal mode of channel 1 to AM

:CHANnel1:MODulate:TYPe?

The query returns AM.

:CHANnel<n>:MODulate:WAVe

- Command Format

:CHANnel<n>:MODulate:WAVe { SINe|SQUare|UPRamp|DNRamp|ARB|NOISe }

:CHANnel<n>:MODulate:WAVe?

- Function

Set the modulated wave mode of specified channel signal, with modes of Sine Wave, Square Wave, Upper Ramp, Lower Ramp, Arbitrary Wave, and Noise.

<n>: Channel No., n value 1, 2, 3, 4.

- Return Format

The query returns the modulated wave mode of specified channel signal.

- Example

:CHANnel1:MODulate:WAVe SINe

Set the modulated wave of signal in channel 1 to Sine Wave

:CHANnel1:MODulate:WAVe?

The query returns SINe

:CHANnel<n>:MODulate:SOURce

- Command Format

:CHANnel<n>:MODulate:SOURce { INTernal|EXTernal }

:CHANnel<n>:MODulate:SOURce?

- Function

Set the modulated source of specified channel, Internal and External.

<n>: Channel No., n value 1, 2, 3, 4.

- Return Format

The query returns the modulated source of specified channel.

- Example

:CHANnel1:MODulate:SOURce INTernal

Set the modulated source of channel 1 to be internal

:CHANnel1:MODulate:SOURce?

The query returns the INTernal

:CHANnel<n>:MODulate:FREQuency

- Command Format

:CHANnel<n>:MODulate:FREQuency {<freq>}

:CHANnel<n>:MODulate:FREQuency?

- Function

Set the modulated frequency of specified channel signal.

<freq> means frequency, in "Hz" unit.

<n>: Channel No., n value 1, 2, 3, 4.

- Return Format

The query returns the modulated frequency of specified channel signal, and the sampling scientific

notation.

- Example

  :CHANnel1:MODulate:FREQuency 2000

  Set the modulated frequency of signal in channel 1 to 2KHz

  :CHANnel1:MODulate:FREQuency?

  The query returns 2e+3

:CHANnel<n>:MODulate:IQMap

- Command Format

  :CHANnel<n>:MODulate: IQMap {<IQ TYPE>}

  :CHANnel<n>:MODulate: IQMap?

- Function

  Set the IQ type of specified QAM to:

  QAM4, QAM8, QAM16, QAM32, QAM64, QAM128, QAM256.

  < IQ TYPE > means the IO Mapping type.

  <n>: Channel No., n value 1, 2, 3, 4.

- Return Format

  The query returns the IQ type of specified channel.

- Example

  :CHANnel1:MODulate:IQMap QAM32

  Set the modulated IQ mapping type of channel 1 to QAM32

  :CHANnel1:MODulate:IQMap?

  The query returns QAM32.

:CHANnel<n>:MODulate:ARB

- Command Format

  :CHANnel<n>:MODulate:ARB <source>,<filename>

  :CHANnel<n>:MODulate:ARB?

- Function

  Set the specified channel to load and modulate arbitrary wave data of a file in the condition of any wave source.

  <n>: Channel No., n value 1, 2, 3, 4.

  <source>: {INTernal|EXTernal|USER}, Internal, External, and User.

  <filename>: Arbitrary wave filename.

- Example

  :CHANnel1:MODulate:ARB INTernal, "test.bsv"

:CHANnel<n>:MODulate:DEPTh

- Command Format

  :CHANnel<n>:MODulate:DEPTh { <depth>}

  :CHANnel<n>:MODulate:DEPTh?

- Function

  Set the modulation depth of specified channel.

  <depth> means the modulation depth, in "%" unit, range of 0% ~ 100%, and AM modulation depth is 0% ~ 120%.

  <n>: Channel No., n value 1, 2, 3, 4.

- Return Format

  The query returns the modulation depth of specified channel.

- Example

  :CHANnel1:MODulate:DEPTh 50

  Set the modulation depth of channel 1 to 50%

  :CHANnel1:MODulate:DEPTh?

  The query returns 50

:CHANnel<n>:MODulate:BITRatio

- Command Format

  :CHANnel<n>:MODulate:BITRatio <ratio>

  :CHANnel<n>:MODulate:BITRatio?

- Function

  Set the bit ratio of specified channel, and the command is only effective for the wave with bit ratio function.

  < ratio > means bit ratio, in "bps" unit.

  <n>: Channel No., n value 1, 2, 3, 4.

- Return Format

  The query returns the bit ratio of specified channel, using scientific notation to return.

- Example

  :CHANnel1:MODulate:BITRatio 100000

  Set the bit ratio of channel 1 to 100Kbps

  :CHANnel1:MODulate:BITRatio?

  The query returns 1e+6

:CHANnel<n>:MODulate:RATio

- Command Format

  :CHANnel<n>:MODulate:RATio <ratio>

  :CHANnel<n>:MODulate:RATio?

- Function

  Set the modulated ratio of specified channel, and the command is only effective for the modulated type with ratio function.

  < ratio > means ratio, in "Hz" unit.

  <n>: Channel No., n value 1, 2, 3, 4.

- Return Format

  The query returns the modulated ratio of specified channel, using scientific notation to return.

- Example

  :CHANnel1:MODulate:RATio 100

  Set the ratio of channel 1 to 100Hz

  :CHANnel1:MODulate:RATio?

  The query returns 1e+2

:CHANnel<n>:OSK:TRIGger:SOURce

- Command Format

  :CHANnel<n>:OSK:TRIGger:SOURce { INTernal|EXTernal }

  :CHANnel<n>:OSK:TRIGger:SOURce?

- Function

Set the trigger source of Oscillation Shift Keying in specified channel, Internal and External.
<n>: Channel No., n value 1, 2, 3, 4.

- Return Format

The query returns the trigger source of Oscillation Shift Keying in specified channel.

- Example

:CHANnel1:OSK:TRIGger:SOURce INTernal
Set the trigger source of Oscillation Shift Keying in channel 1 to be internal
:CHANnel1:OSK:TRIGger:SOURce?
The query returns INTernal

:CHANnel<n>:FM:FREQuency:DEV

- Command Format

:CHANnel<n>:FM:FREQuency:DEV { <freq>}
:CHANnel<n>:FM:FREQuency:DEV?

- Function

Set the frequency deviation of specified channel.
<freq> means the frequency deviation, in "Hz" unit. 0Hz ~ is the current fundamental frequency.
<n>: Channel No., n value 1, 2, 3, 4.

- Return Format

The query returns the frequency deviation of specified channel, using scientific notation to return the data.

- Example

:CHANnel1:FM:FREQuency:DEV 2000
Set the frequency deviation of channel 1 to 2KHz
:CHANnel1:FM:FREQuency:DEV?
The query returns 2e+3

:CHANnel<n>:PM:PHASe:DEV

- Command Format

:CHANnel<n>:PM:PHASe:DEV { <phase>}
:CHANnel<n>:PM:PHASe:DEV?

- Function

Set the phase deviation of specified channel.
< phase > means phase deviation, in " ° " unit, range of 0~360.
<n>: Channel No., n value 1, 2, 3, 4.

- Return Format

The query returns the phase deviation of specified channel.

- Example

:CHANnel1:PM:PHASe:DEV 30
Set the phase deviation of channel 1 to 30°
:CHANnel1:PM:PHASe:DEV?
The query returns 30

:CHANnel<n>:PWM:DUTY:DEV

- Command Format

:CHANnel<n>:PWM:DUTY:DEV { <duty>}
:CHANnel<n>:PWM:DUTY:DEV?

- Function

  Set the pulse width deviation with the condition of pulse width modulation in specified channel.

  < duty > means pulse width deviation, in "%" unit, range of 0~100.

  <n>: Channel No., n value 1, 2, 3, 4.

- Return Format

  The query returns the pulse width deviation with the condition of pulse width modulation in specified channel, using scientific notation to return the data.

- Example

  :CHANnel1:PWM:DUTY:DEV 10

  Set the pulse width deviation of channel 1 to 10%

  :CHANnel1:PWM:DUTY:DEV?

  The query returns 1e+1

:CHANnel<n>:FSK:FREQuency<m>

- Command Format

  :CHANnel<n>:FSK:FREQuency<m> { <freq>}

  :CHANnel<n>:FSK:FREQuency<m>?

- Function

  Set the hopping frequency of Multiple Frequency Shift Keying in specified channel, and the command is only effective when the modulated way is specified in advance.

  < freq > means frequency, in "Hz" unit.

  <n>: Channel No., n value 1, 2, 3, 4.

  <m>: Frequency No., value 1 in 2FSK, value 1 & 2 in 3FSK, value 1 & 2 & 3 in 4FSK.

- Return Format

  The query returns the hopping frequency of specified channel, using scientific notation to return the data.

- Example

  :CHANnel1:FSK:FREQ1 2000

  Set the hopping frequency of channel 1 to 2KHz

  :CHANnel1:FSK:FREQ1?

  The query returns 2e+3

:CHANnel<n>:PSK:PHASe<m>

- Command Format

  :CHANnel<n>:PSK:PHASe<m> { < phase >}

  :CHANnel<n>:PSK:PHASe<m>?

- Function

  Set the phase of Multiple Phase Shift Keying in specified channel, and the command is only effective when the modulated way is specified in advance.

  < phase> means phase, in "°" unit, range of 0~360.

  <n>: Channel No., n value 1, 2, 3, 4.

  <m>: Phase No., value 1 in PSK, value 1 & 2 in BPSK, value 1 & 2 & 3 & 4 in QPSK.

- Return Format

  The query returns the phase of Phase Shift Keying in specified channel, using scientific notation to return the data.

- Example

  :CHANnel1:PSK:PHAS1 90

Set the phase of channel 1 to 90°.
:CHANnel1:PSK:PHAS1?
The query returns 9e+1

:CHANnel<n>:OSK:TIMe

- **Command Format**
  :CHANnel<n>:OSK:TIMe { <time>}
  :CHANnel<n>:OSK:TIMe?
- **Function**
  Set the oscillation time of Oscillation Shift Keying in the modulation mode of specified channel.
  < time > means oscillation time, in "s" unit.
  <n>: Channel No., n value 1, 2, 3, 4.
- **Return Format**
  The query returns the oscillation time of Oscillation Shift Keying in the modulation mode of specified channel, using scientific notation to return the data.
- **Example**
  :CHANnel1:OSK:TIMe 2ms
  Set the oscillation time of Oscillation Shift Keying in channel 1 to 2ms
  :CHANnel1:OSK:TIMe?
  The query returns 2e-3

Frequency Sweep

:CHANnel<n>:SWEep:TYPe

- **Command Format:**
  :CHANnel<n>:SWEep:TYPe { LINe|LOG|STEP|LIST}
  :CHANnel<n>:SWEep:TYPe?
- **Function**
  Set the sweep type of specified channel, with types of Linear Sweep, Log Sweep, Step Sweep, and List Sweep.
  <n>: Channel No., n value 1, 2, 3, 4.
- **Return Format**
  The query returns sweep type of specified channel.
- **Example**
  :CHANnel1:SWEep:TYPe LINe
  Set the sweep type of channel 1 to linear sweep
  :CHANnel1:SWEep:TYPe?
  The query returns LINe

:CHANnel<n>:SWEep:FREQuency:STARt

- **Command Format**
  :CHANnel<n>:SWEep:FREQuency:STARt <freq>
  :CHANnel<n>:SWEep:FREQuency:STARt?
- **Function**
  Set the start frequency of sweep in specified channel.
  < freq > means frequency, in "Hz" unit.
  <n>: Channel No., n value 1, 2, 3, 4.

- Return Format

  The query returns the start frequency of sweep in specified channel, using scientific notation to return the data.

- Example

  :CHANnel1:SWE:FREQ:STAR 2000

  Set the start frequency of sweep in channel 1 to 2KHz

  :CHANnel1:SWE:FREQ:STAR?

  The query returns 2e+3

:CHANnel<n>:SWEep:FREQuency:STOP

- Command Format

  :CHANnel<n>:SWEep:FREQuency:STOP <freq>

  :CHANnel<n>:SWEep:FREQuency:STOP?

- Function

  Set the stop frequency of sweep in specified channel.

  < freq > means frequency, in "Hz" unit.

  <n>: Channel No., n value 1, 2, 3, 4.

- Return Format

  The query returns the stop frequency of sweep in specified channel, using scientific notation to return the data.

- Example

  :CHANnel1:SWE:FREQ:STOP 2000

  Set the stop frequency of sweep in channel 1 to 2KHz

  :CHANnel1:SWE:FREQ:STOP?

  The query returns 2e+3

:CHANnel<n>:SWEep:TIMe

- Command Format

  :CHANnel<n>:SWEEP:TIMe <time>

  :CHANnel<n>:SWEEP:TIMe?

- Function

  Set the sweep time of specified channel.

  < time > means time, in "s" unit, range of 1ms ~ 500s.

  <n>: Channel No., n value 1, 2, 3, 4.

- Return Format

  The query returns the sweep time of specified channel, using scientific notation to return the data.

- Example

  :CHANnel1:SWEEP:TIMe 2

  Set the sweep time of channel 1 to 2S

  :CHANnel1:SWEEP:TIMe?

  The query returns 2e+0

:CHANnel<n>:SWEep:HOLD

- Command Format

  :CHANnel<n>:SWEEP:HOLD <time>

  :CHANnel<n>:SWEEP:HOLD?

- Function

Set the hold time of sweep in specified channel, and the command is only effective when in the Step Sweep and List Sweep mode.
< time > means time, in "s" unit.
<n>: Channel No., n value 1, 2, 3, 4.

- Return Format

The query returns the hold time of sweep in specified channel, using scientific notation to return the data.

- Example

:CHANnel1:SWEEP:HOLD 2
Set the hold time of sweep in channel 1 to 2S
:CHANnel1:SWEEP:HOLD?
The query returns 2e+0

:CHANnel<n>:SWEep:STEPs

- Command Format

:CHANnel<n>:SWEEP:STEPs <steps>
:CHANnel<n>:SWEEP:STEPs?

- Function

Set the total steps in Step Sweep of specified channel, and the command is only effective when in the Step Sweep mode.

< steps >: Steps

<n>: Channel No., n value 1, 2, 3, 4.

- Return Format

The query returns the sweep steps of specified channel, and the integer data.

- Example

:CHANnel1:SWEEP:STEPs 10
Set the steps of specified channel to 10 steps
:CHANnel1:SWEEP:STEPs?
The query returns 10

:CHANnel<n>:SWEep:TRIGger

- Command Format

:CHANnel<n>:SWEep:TRIGger

- Function

Trigger the sweep output of specified channel, and the parameter is only effective when in the manual trigger mode.

- Example

:CHANnel1:SWEep:TRIGger
Trigger the sweep output once.

Burst

:CHANnel<n>:BURSt:TYPe

- Command Format

:CHANnel<n>:BURSt:TYPe {NCYC|GATe|INFinit}
:CHANnel<n>:BURSt:TYPe?

- Function

Set the burst type of specified channel, with types of N Cycle, Gated, and Infinite.
<n>: Channel No., n value 1, 2, 3, 4.

- Return Format

The query returns the burst type of specified channel.

- Example

:CHANnel1:BURSt:TYPe NCYC
Set the burst of channel 1 to N Cycle
:CHANnel1:BURSt:TYPe?
The query returns 2e+0

:CHANnel<n>:BURSt:PERiod

- Command Format

:CHANnel<n>:BURSt:PERiod <period >
:CHANnel<n>:BURSt:PERiod?

- Function

Set the burst period of specified channel.
< period > means time, in "s" unit.
<n>: Channel No., n value 1, 2, 3, 4.

- Return Format

The query returns the burst period of specified channel, using scientific notation to return the data.

- Example

:CHANnel1:BURSt:PERiod 5ms
Set the burst period of channel 1 to 5ms
:CHANnel1:BURSt:PERiod?
The query returns 5e-3

:CHANnel<n>:BURSt:PHASe

- Command Format

:CHANnel<n>:BURSt:PHASe <phase>
:CHANnel<n>:BURSt:PHASe?

- Function

Set the burst phase of specified channel.
< phase > means phase, in "°" unit, range of 0 ~ 360.
<n>: Channel No., n value 1, 2, 3, 4.

- Return Format

The query returns the burst phase of specified channel, using scientific notation to return the data.

- Example

:CHANnel1:BURSt:PHASe 18
Set the burst phase of channel 1 to 18°
:CHANnel1:BURSt:PHASe?
The query returns 1.8e+1

:CHANnel<n>:BURSt:CYCLes

- Command Format

:CHANnel<n>:BURSt:CYCLes <cycles>
:CHANnel<n>:BURSt:CYCLes?

- Function

Set the burst cycles of specified channel.
< cycles > means cycles, integer data.
<n>: Channel No., n value 1, 2, 3, 4.

- Return Format
  The query returns the burst cycles of specified channel.
- Example
  :CHANnel1:BURSt:CYCLes 2
  Set the burst cycles of specified channel to 2
  :CHANnel1:BURSt:CYCLes?
  The query returns 2

:CHANnel<n>:BURSt:GATe:POLarity

- Command Format
  :CHANnel<n>:BURSt:GATe:POLarity {POSitive|NEGative}
  :CHANnel<n>:BURSt:GATe:POLarity?
- Function
  Set the gated burst polarity in the specified channel, Positive Polarity and Negative Polarity.
  <n>: Channel No., n value 1, 2, 3, 4.
- Return Format
  The query returns the gated burst polarity in specified channel.
- Example
  :CHANnel1:BURSt:GATe:POLarity POSitive
  Set the gated burst polarity of channel 1 to be positive
  :CHANnel1:BURSt:GATe:POLarity?
  The query returns POSitive

:CHANnel<n>:BURSt:TRIGger

- Command Format
  :CHANnel<n>:BURSt:TRIGger
- Function
  Trigger the burst output of specified channel, and the parameter is only effective when in the manual trigger mode.
- Example
  :CHANnel1:BURSt:TRIGger
  Trigger the burst output once

WARB Commands

It is used to write file commands of arbitrary wave, including the fundamental arbitrary wave and modulated arbitrary wave writing configuration.

:WARB<n>:MODulate

- Command Format
  :WARB<n>:MODulate <arb file>
- Function
  Write the modulated arbitrary wave. Send the command firstly, and send the arbitrary wave file data to the signal source.

<arb file> means the arbitrary wave filename, only bsv file supported.

- Example

  :WARB1:MODulate "test.bsv"

  Write the modulated arbitrary wave of channel 1

:WARB<n>:CARRier

- Command Format

  :WARB<n>:CARRier <arb file>

- Function

  Write the fundamental arbitrary wave. Send the command firstly, and send the arbitrary wave file data to the signal source.

  <arb file> means the arbitrary wave filename, only bsv file supported.

- Example

  :WARB1:CARRier "test.bsv"

  Write the fundamental arbitrary wave file of channel 1

:WFREQLIST<n>

- Command Format

  :WFREQLIST<n> <arb file>

- Function

  Write the frequency list file. Send the command firstly, and send the frequency list file data to the signal source.

  <arb file> means the frequency list filename, only csv file supported.

- Example:

  :WFREQLIST1"freq.csv"

  Write the frequency list file of channel 1

DIGital Commands

It is used to output digital communication signal, such as the UART, SPI, I2C, etc.

:DIGital

- Command Format

  :DIGital {{1|ON}|{0|OFF}}

  :DIGital?

- Function

  Set the digital communication signal function ON/OFF in specified channel.

- Return Format

  The query returns the digital communication signal function ON/OFF, 0 in OFF, 1 in ON.

- Example

  :DIGital ON Digital communication signal ON

  :DIGital? The query returns 1

:DIGital:TYPe

- Command Format

  :DIGital:TYPe {UART|IIC|SPI }

:DIGital:TYPe?

- Function

  Set the digital communication signal type of specified channel, with types of UART, IIC, and SPI.

- Return Format

  The query returns the digital communication signal type of specified channel.

- Example

  :DIGital:TYPe UART

  Set the communication signal of channel to UART.

  :DIGital:TYPe?

  The query returns UART.

:DIGital:AMPLitude

- Command Format

  :DIGital:AMPLitude <amp>

  :DIGital:AMPLitude?

- Function

  Set the digital communication signal amplitude of specified channel.

  <amp>: Amplitude, in "Vpp" unit.

- Return Format

  The query returns the digital communication signal amplitude of specified channel, using scientific notation to return.

- Example

  :DIGital:AMPLitude 3

  Set the communication signal of channel to 3Vpp

  :DIGital:AMPLitude? The query returns 3e+0

:DIGital:FORMat

- Command Format

  :DIGital:FORMat { HEX|CHAR }

  :DIGital:FORMat?

- Function

  Set the data format of digital communication signal in specified channel, with Hexadecimal Data and ASCII Data.

- Return Format

  The query returns the data format of digital communication signal in specified channel.

- Example:

  :DIGital:FORMat HEX

  Set the data format of digital communication signal to hexadecimal

  :DIGital:FORMat? The query returns HEX

:DIGital:AS

- Command Format

  :DIGital:AS {{1|ON}|{0|OFF}}

  :DIGital:AS?

- Function

  Set the sending way of digital communication signal in specified channel, OFF in Manual Sending, ON in Auto Sending.

- Return Format

  The query returns if the digital communication signal of specified channel sent in auto way, 1 in ON, 0 in OFF.

- Example

  :DIGital:AS ON

  Set the digital communication signal to auto sending

  :DIGital:AS? The query returns 1

:DIGital:AS:INTerval

- Command Format

  :DIGital:AS:INTerval <time>

  :DIGital:AS:INTerval?

- Function

  Set the auto digital communication signal interval of specified channel.

  <time>：Interval, in "s" unit.

- Return Format

  The query returns auto digital communication signal interval of specified channel, using scientific notation to return.

- Example

  :DIGital:AS:INTerval 10ms

  Set the auto digital communication signal interval to 10ms

  :DIGital:AS:INTerval?

  The query returns 1e-2

:DIGital:TRIGger

- Command Format

  :DIGital:TRIGger

- Function

  Trigger the digital communication signal sending of specified channel, effective in the manual way.

- Example

  :DIGital:TRIGger

  Trigger the digital signal sending once

:DIGital:DATA

- Command Format

  :DIGital:DATA <data>

  :DIGital:DATA?

- Function

  Setting the auto digital communication signal data needs to continuously operate twice. Send the command firstly, followed by the binary byte stream data, and its data format is related to the :DIGital:FORMat command.

  <data>：Binary byte stream data

- Return Format

  The query returns the digital communication signal data of specified channel, and the binary byte stream data.

- Example

:DIGital:DATA
Send the command to signal source firstly, then send the communication signal data
:DIGital:DATA?
The query returns the binary byte stream data

UART

:DIGital:UART:BAUDrate

- Command Format
  :DIGital:UART:BAUDrate <baudrate>
  :DIGital:UART:BAUDrate?
- Function
  Set the baudrate of UART digital communication signal in specified channel.
  <baudrate>：Baudrate, in "bps" unit, integer data
- Return Format
  The query returns the baudrate of UART digital communication signal in specified channel, and the integer data.
- Example
  :DIGital:UART:BAUDrate 115200
  Set the baudrate of UART communication signal to 115200
  :DIGital:UART:BAUDrate?
  The query returns 115200

:DIGital:UART:DATa

- Command Format
  :DIGital:UART:DATa <bit >
  :DIGital:UART:DATa?
- Function
  Set the data bit of UART digital communication signal in specified channel.
  <bit >：Data bit, integer data, range of 4~8.
- Return Format
  The query returns the data bit of UART digital communication signal in specified channel, and integer data.
- Example
  :DIGital:UART:DATa 4
  Set the data bit of UART communication signal to 4
  :DIGital:UART:DATa?
  The query returns 4.

:DIGital:UART:STOP

- Command Format
  :DIGital:UART:STOP <bit >
  :DIGital:UART:STOP?
- Function
  Set the stop bit of UART digital communication signal in specified channel.

<bit >: Stop bit, integer data, range of 1~2

- Return Format

  The query returns the stop bit of UART digital communication signal in specified channel, and integer data.

- Example

  :DIGital:UART:STOP 1

  Set the stop bit of UART communication signal to 1

  :DIGital:UART:STOP?

  The query returns 1

:DIGital:UART:PARity

- Command Format

  :DIGital:UART:PARity {NONE|EVEN|ODD}

  :DIGital:UART:PARity?

- Function

  Set the parity bit of UART digital communication signal in specified channel, with None, Even, and Odd Parity.

- Return Format

  The query returns the parity bit of UART digital communication signal in specified channel, and integer data.

- Example

  :DIGital:UART:PARity NONE

  Set the parity bit of UART communication signal to NONE

  :DIGital:UART:PARity? The query returns NONE

IIC

:DIGital:IIC:CLOCk

- Command Format

  :DIGital:IIC:CLOCk <freq>

  :DIGital:IIC:CLOCk?

- Function

  Set the clock of IIC digital communication signal in specified channel.

  <freq>: Clock, in "Hz" unit.

- Return Format

  The query returns the clock of IIC digital communication signal in specified channel, using scientific notation to return.

- Example

  :DIGital:IIC:CLOCk 1000

  Set the clock of IIC communication signal to 1KHz

  :DIGital:IIC:CLOCk?

  The query returns 1e+3

:DIGital:IIC:ADDRess

- Command Format

  :DIGital:IIC:ADDRess <address>

:DIGital:IIC:ADDRess?

- Function

  Set the address of IIC digital communication signal in specified channel.

  <address>：Address, integer data.

- Return Format

  The query returns the address of IIC digital communication signal in specified channel.

- Example

  :DIGital:IIC:ADDRess 3

  Set the address of IIC communication signal to 3

  :DIGital:IIC:ADDRess?

  The query returns 3

SPI

:DIGital:SPI:CLOCk

- Command Format

  :DIGital:SPI:CLOCk <freq>

  :DIGital:SPI:CLOCk?

- Function

  Set the clock of SPI digital communication signal in specified channel.

  <freq>：Clock, in "Hz" unit.

- Return Format

  The query returns the clock of SPI digital communication signal in specified channel, using scientific notation to return.

- Example

  :DIGital:SPI:CLOCk 1000

  Set the clock of SPI communication signal to 1KHz

  :DIGital:SPI:CLOCk?

  The query returns 1e+

DISPlay Commands

It is used in signal source display information.

:DISPlay:DATA?

- Command Format

  :DISPlay:DATA?

- Function

  Query the image data of current screen, returning the image data with BMP format by default, and the image data format depends on the :SYSTem:PICTure:FORMat command.

- Return Format

  The query returns the image data, and the returning data matches with the binary data of IEEE 488.2 # format.

- Example

  :DISPlay:DATA? The query returns the image data

  Data format: #800012345+ image data

Programming Instructions

Describe some troubles and solutions during the programming operation, and please follow instructions if any trouble met.

Programming Preparation

The programming preparation is only for the Visual Studio and LabVIEW programming in the Windows operating system.

Make sure that your computer has been installed VISA library of NI (Download in https://www.ni.com/en-ca/support/downloads/drivers/download.ni-visa.html), and the default installation path in this page: C:\Program Files\IVI Foundation\VISA.

Establishing communication with PC, please use USB cable to connect the USB DEVICE interface of device with the USB interface of PC, or use LAN cable to connect the LAN interface of device with the LAN interface of PC.

VISA Programming Example

Given some programming Example, this page can help you understand how to use VISA, operate the device as per the commands details, and develop more applications.

VC++ Example

- Condition: Windows System & Visual Studio.
- Description: Access the device through the USBTMC and TCP/IP, and send the "*IDN?" command in NI-VISA to check the device information.
- Steps

1. Open the Visual Studio software, and newly create a VC++ win32 console project.
2. Set the project condition for calling NI-VISA library, with Static Library and Dynamic Library.

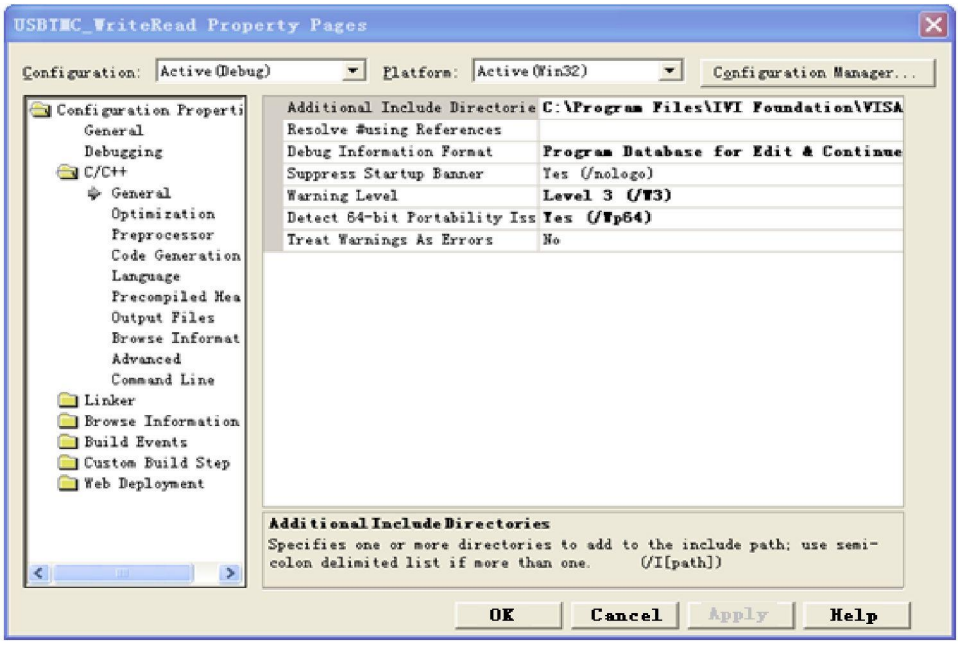
a) Static Library

Copy the visa.h, visatype.h, visa32.lib files in NI-VISA installation path to the root path of VC++ project and add them to the project. Add the following codes to the projectname.cpp file:

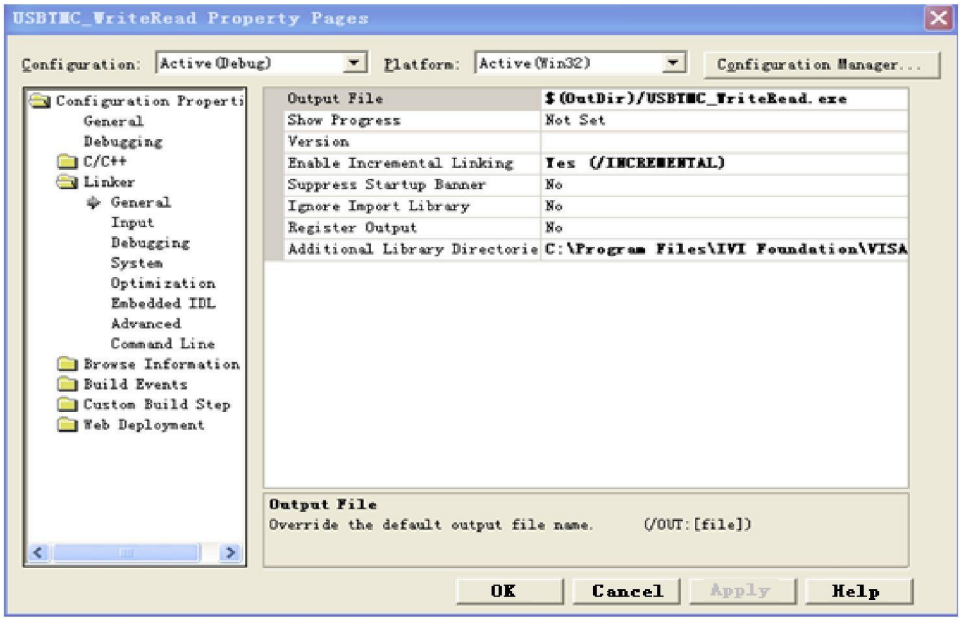
```
#include "visa.h"
#pragma comment(lib,"visa32.lib")
```

b) Dynamic Library

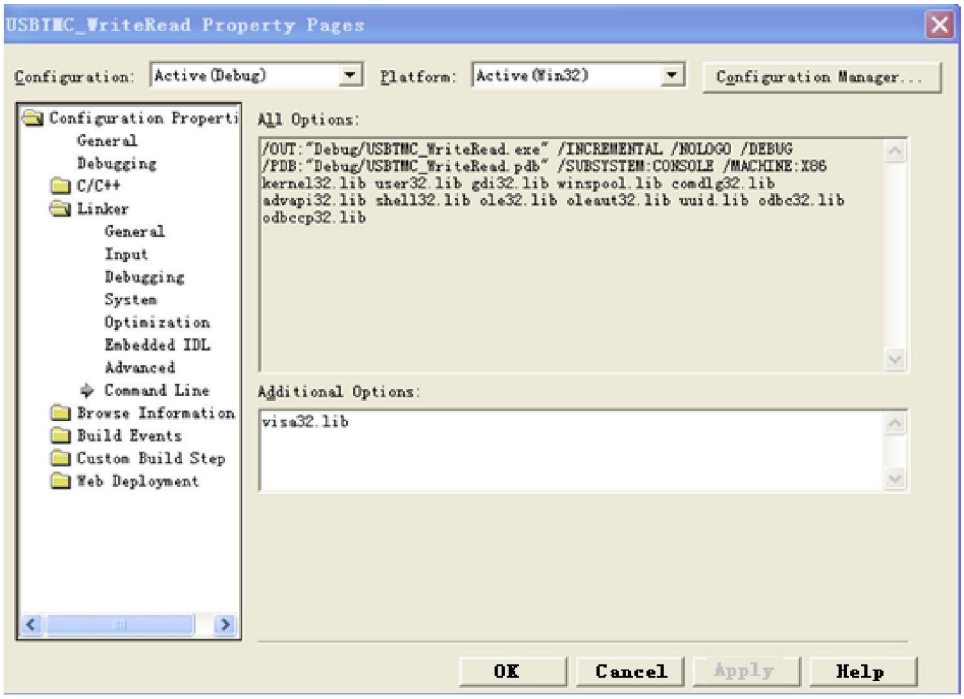
Click the "project>>properties", select the "c/c++---General" in the left side of properties dialog, and set the "Additional Include Directories" content to the installation path of NI-VISA,(For example, C:\ProgramFiles\IVI Foundation\VISA\WinNT\include), as followings showed.



Select the "Linker-General" in the left side of properties dialog, and set the "Additional Library Directories" content to the installation path of NI-VISA, (For example, C:\Program Files\IVI Foundation\VISA\WinNT\include), as followings showed.



Select the "Linker-Command Line" in the left side of properties dialog, and set the "Additional" content to the visa32.lib, as followings showed.



Add the visa.h file to the projectname.cpp.

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

1. Source Code

a) USBTMC Example

```
int usbtmc_test()
{ /** This code demonstrates sending synchronous read & write commands
    * to an USB Test & Measurement Class (USBTMC) instrument using NI-VISA
    * The example writes the "*IDN?\n" string to all the USBTMC
    * devices connected to the system and attempts to read back
    * results using the write and read functions.
```

```
* Open Resource Manager
* Open VISA Session to an Instrument
* Write the Identification Query Using viPrintf
* Try to Read a Response With viScanf
* Close the VISA Session*/
ViSession defaultRM;
ViSession instr;
ViUInt32 numInstrs;
ViFindList findList;
ViStatus status;
char instrResourceString[VI_FIND_BUFLEN];
unsigned char buffer[100];
int i;
status = viOpenDefaultRM(&defaultRM);
if (status < VI_SUCCESS)
{
    printf("Could not open a session to the VISA Resource Manager!\n");
    return status;
}
/*Find all the USB TMC VISA resources in our system and store the number of resources in the
system in numInstrs.*/
status = viFindRsrc(defaultRM, "USB?*INSTR", &findList, &numInstrs, instrResourceString);
if (status<VI_SUCCESS)
{
    printf("An error occurred while finding resources. \nPress Enter to continue.");
    fflush(stdin);
    getchar();
    viClose(defaultRM);
    return status;
}
/** Now we will open VISA sessions to all USB TMC instruments.
*     We must use the handle from viOpenDefaultRM and we must
*     also use a string that indicates which instrument to open. This
*     is called the instrument descriptor. The format for this string
*     can be found in the function panel by right clicking on the
*     descriptor parameter. After opening a session to the
*     device, we will get a handle to the instrument which we
*     will use in later VISA functions. The AccessMode and Timeout
*     parameters in this function are reserved for future
*     functionality. These two parameters are given the value VI_NULL. */
for (i = 0; i < int(numInstrs); i++)
{
    if (i > 0)
    {
        viFindNext(findList, instrResourceString);
    }
    status = viOpen(defaultRM, instrResourceString, VI_NULL, VI_NULL, &instr);
    if (status < VI_SUCCESS)
```

```
            {
                printf("Cannot open a session to the device %d. \n", i + 1);
                continue;
            }
            /** At this point we now have a session open to the USB TMC instrument.
            *We will now use the viPrintf function to send the device the string "*IDN?\n",
            *asking for the device's identification. */
            char * cmmand = "*IDN?\n";
            status = viPrintf(instr, cmmand);
            if (status < VI_SUCCESS)
            {
                printf("Error writing to the device %d. \n", i + 1);
                status = viClose(instr);
                continue;
            }
            /** Now we will attempt to read back a response from the device to
            *the identification query that was sent. We will use the viScanf
            *function to acquire the data.
            *After the data has been read the response is displayed. */
            status = viScanf(instr, "%t", buffer);
            if (status < VI_SUCCESS)
            {
                printf("Error reading a response from the device %d. \n", i + 1);
            }
            else
            {
                printf("\nDevice %d: %s\n", i + 1, buffer);
            }
            status = viClose(instr);
        }
        /*Now we will close the session to the instrument using viClose. This operation frees all
        system resources.*/
        status = viClose(defaultRM);
        printf("Press Enter to exit.");
        fflush(stdin);
        getchar();
        return 0;
    }

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

b) TCP/IP Example

```
    int tcp_ip_test(char *pIP)
    {
        char outputBuffer[VI_FIND_BUFLEN];
```

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

C# Example

- Condition: Windows System & Visual Studio.
- Description: Access the device through the USBTMC and TCP/IP, and send "*IDN?" command in NI-VISA to check the device information.
- Steps

1. Open the Visual Studio software, and newly create a C# console project.
2. Add the C# reference of VISA, Ivi.Visa.dll and NationalInstruments.Visa.dll.
3. Source Code

a) USBTMC Example

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

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

b) TCP/IP Example

```
class Program
  {
    void tcp_ip_test(string ip)
    {
        using (var rmSession = new ResourceManager())
        {
            try
```

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

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

VB Example

- Condition: Windows System & Microsoft Visual Basic 6.0.
- Description: Access the device through the USBTMC and TCP/IP, and send "*IDN?" command in the NI-VISA to check the device information.
- Steps

1. Open the Visual Basic software, and newly create a standard application.
2. Set the project condition for calling NI-VISA library: Click the Existing tab of Project>>Add Existing Item, find the visa32.bas file in the "include" folder of NI-VISA installation path and add it, as followings showed.

3. Source Code

a) USBTMC Example

```
PrivateFunction usbtmc_test() AsLong
```

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

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

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

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

' Now we will open VISA sessions to all USB TMC instruments.
' We must use the handle from viOpenDefaultRM and we must
' also use a string that indicates which instrument to open.    This
' is called the instrument descriptor.    The format for this string
' can be found in the function panel by right clicking on the
' descriptor parameter. After opening a session to the
```

```
    ' device, we will get a handle to the instrument which we
    ' will use in later VISA functions.    The AccessMode and Timeout
    ' parameters in this function are reserved for future
    ' functionality.    These two parameters are given the value VI_NULL.
    For i = 0 To numInstrs
    If (i > 0) Then
        status = viFindNext(findList, instrResourceString)
    EndIf
        status = viOpen(defaultRM, instrResourceString, VI_NULL, VI_NULL, instrsesn)
    If (status < VI_SUCCESS) Then
        resultTxt.Text = "Cannot open a session to the device " + CStr(i + 1)
    GoTo NextFind
    EndIf

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

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

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

b) TCP/IP Example

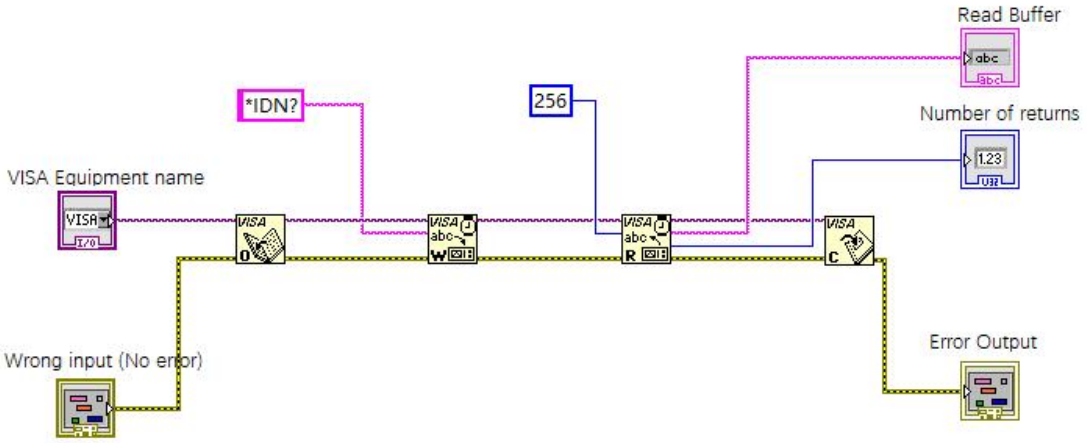
```
    PrivateFunction tcp_ip_test(ByVal ip AsString) AsLong
    Dim outputBuffer AsString * VI_FIND_BUFLEN
    Dim defaultRM AsLong
    Dim instrsesn AsLong
```

```
Dim status AsLong
Dim count AsLong

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

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

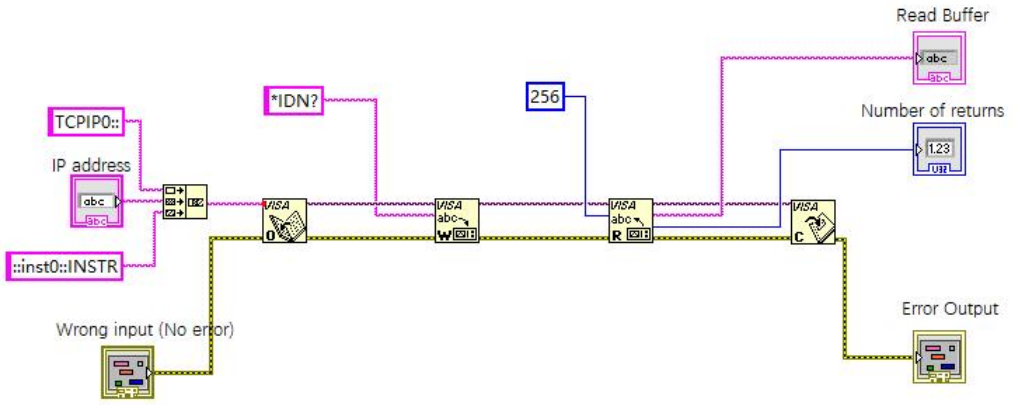
LabVIEW Example

- Condition: Windows System & LabVIEW.
- Description: Access the device through the USBTMC and TCP/IP, and send "*IDN?" command in the NI-VISA to check the device information.
- Steps

1. Open the LabVIEW software, and newly create a VI file.
2. Add the control, right-click the last interface, select and add the VISA resource name, input error, output error and some indicators.
3. Open the frame interface, right-click the VISA resource name, select and add the following functions in the VISA interface of popup menu: VISA Write, VISA Read, VISA Open, and VISA Close.
4. The VISA session of USBTMC device is opened by VI, writing "*IDN?" command to the device and reads back the response value. The VISA session will be closed by VI when the communication is finished, as followings showed:



5. Communicating with device through the TCP/IP is similar to USBTMC, set the VISA writing and reading to be synchronous I/O, and the LabVIEW to be asynchronous I/O by default. Right-click the node, and select the "Synchronous I/O Mode>>Synchronous" in shortcut menu to write or read data synchronously, as followings showed.



MATLAB Example

- Condition: Windows System & MATLAB.
- Description: Access the device through the USBTMC and TCP/IP, and send "*IDN?" command in NI-VISA to check the device information.
- Steps

1. Open the MATLAB software, and click the File>>New>>Script in the Matlab interface to create a new M file.
2. Source Code:

a) USBTMC Example

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

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

%Open the VISA object created
fopen(vu);

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

%Request the data

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

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

b) TCP/IP Example

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

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

%Open the VISA object created

fopen(vt);

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

%Request the data
outputbuffer = fscanf(vt);
disp(outputbuffer);
%Close the VISA object
fclose(vt);
delete(vt);
clear vt;
end
```

Python Example

- Condition: Windows System & Python3.8 & PyVISA 1.11.0.
- Description: Access the device through the USBTMC and TCP/IP, and send "*IDN?" command in the NI-VISA to check the device information.
- Steps

1. Firstly install the Python, then open its script compiling software, and create a new test.py file.
2. Install PyVISA through the pip install PyVISA command, please refer to https://pyvisa.readthedocs.io/en/latest/ if any questions.
3. Source Code

a) USBTMC Example

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

b) TCP/TP Example

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

Programming Applications

Sine Wave Configuration

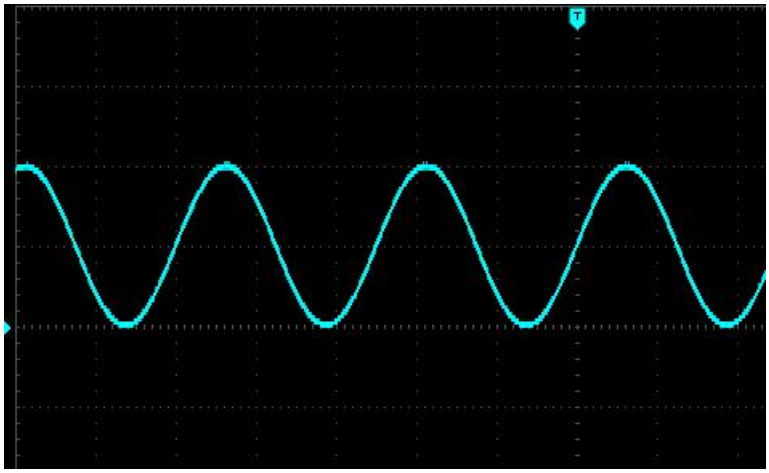
In this page, we will introduce how to configure the sine function.

Description

The sine wave has amplitude, offset, and phase relative to the synchronization pulse, and its amplitude and offset can be set by high and low voltage.

Example

The following waves can be set by the SCPI command series, and the high level and low level can be used to replace :CHANnel1:BASE:AMPLitude and :CHANnel1:BASE:OFFSet.



The above sine wave can be generated by following commands.

:CHANnel1:MODe CONTinue
:CHANnel1:BASE:WAVe SINe
:CHANnel1:BASE:FREQuency 2000
:CHANnel1:BASE:HIGH 2
:CHANnel1:BASE:LOW 0
:CHANnel1:BASE:PHAse 20
:CHANnel1:OUTPut ON

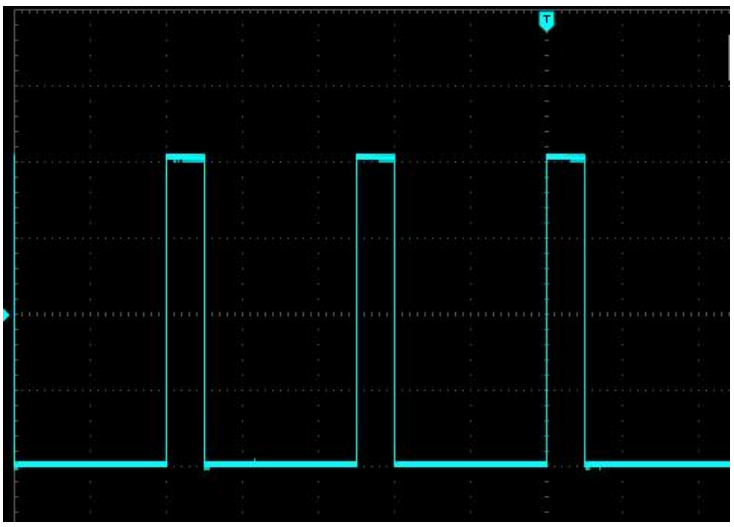
Square Wave Configuration

Description

The square wave has amplitude, offset, and phase relative to the synchronization pulse, and also with duty ratio and cycle. Its amplitude and offset can be set by high and low voltage.

Example

The following waves can be set by the SCPI command series.



The above square wave can be generated by following commands.

:CHANnel1:MODe CONTinue
:CHANnel1:BASE:WAVe SQUare
:CHANnel1:BASE:FREQuency 40000
:CHANnel1:BASE:AMPLitude 2
:CHANnel1:BASE:OFFSet 0
:CHANnel1:BASE:PHAse 90
:CHANnel1:BASE:DUTY 20
:CHANnel1:OUTPut ON

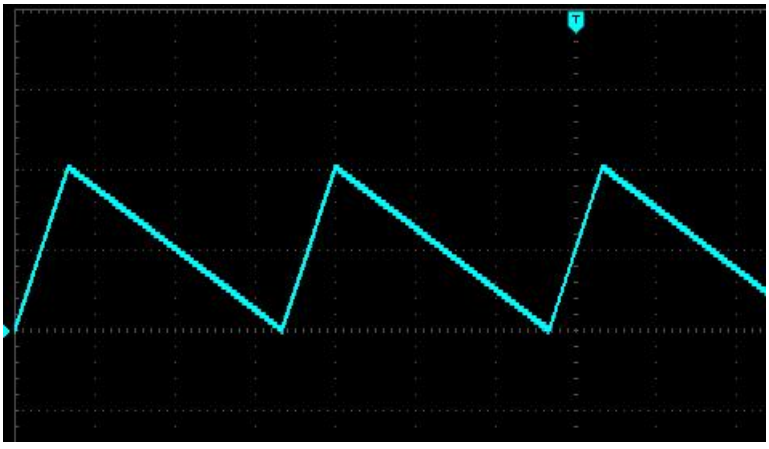
Sawtooth Wave Configuration

Description

The sawtooth wave has amplitude, offset, and phase relative to the synchronization pulse. Its symmetry can be used to create ramp wave and other waves. Its amplitude and offset can be set by high and low voltage.

Example

The following waves can be set by the SCPI command series, and the high level and low level can be used to replace :CHANnel1:BASE:AMPLitude and :CHANnel1:BASE:OFFSet.



The above sawtooth wave can be generated by following commands.

:CHANnel1:MODe CONTinue
:CHANnel1:BASE:WAVe RAMP
:CHANnel1:BASE:FREQuency 30000
:CHANnel1:BASE:HIGH 2
:CHANnel1:BASE:LOW 0
:CHANnel1:BASE:PHAse 90
:CHANnel1:RAMP:SYMMetry 20
:CHANnel1:OUTPut ON

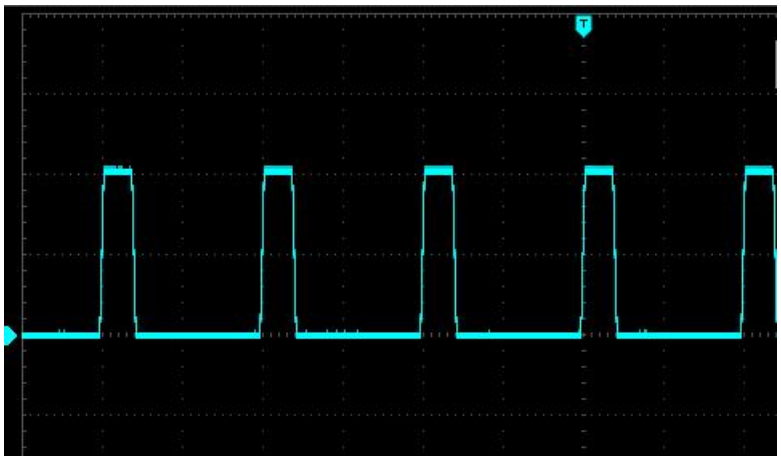
Pulse Wave Configuration

Description

The pulse wave has amplitude, offset, and phase relative to the synchronization pulse. It also adds edge slope and duty ratio (or pulse width). Its amplitude and offset can be set by high and low voltage.

Example

The following waves can be set by the SCPI command series, and the high level and low level can be used to replace :CHANnel1:BASE:AMPLitude and :CHANnel1:BASE:OFFSet.



The above pulse wave can be generated by following commands.

:CHANnel1:MODe CONTinue
:CHANnel1:BASE:WAVe PULSe
:CHANnel1:BASE:FREQuency 100000
:CHANnel1:BASE:HIGH 2
:CHANnel1:BASE:LOW 0
:CHANnel1:BASE:PHAse 270
:CHANnel1:BASE:DUTY 20
:CHANnel1:PULSe:RISe 0.0000002
:CHANnel1:PULSe:FALL 0.0000002
:CHANnel1:OUTPut ON

Arbitrary Wave Configuration

In this page, we will introduce how to configure arbitrary wave.

Description

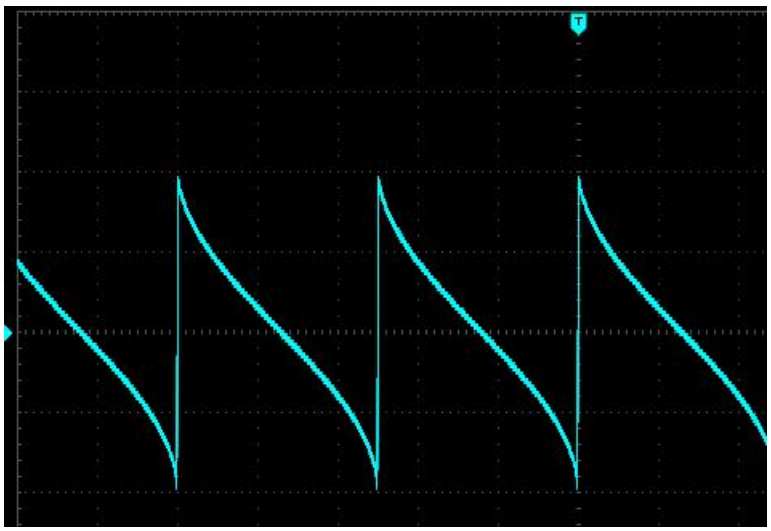
The arbitrary wave has frequency, amplitude, offset and phase, and also adds modes and wave files.

Example

The built-in arbitrary wave can be loaded and modified by the following codes.

```
:CHANnel1:MODe CONTinue
:CHANnel1:BASE:WAVe ARB
:CHANnel1:ARB:MODe DDS
:CHANnel1:BASE:ARB INTernal,"ACos.bsv"
:CHANnel1:BASE:FREQuency 200000
:CHANnel1:BASE:AMPLitude 2
:CHANnel1:BASE:OFFSet 0
:CHANnel1:BASE:PHAse 90
:CHANnel1:OUTPut ON
```

The waves generated by the commands are as followings.



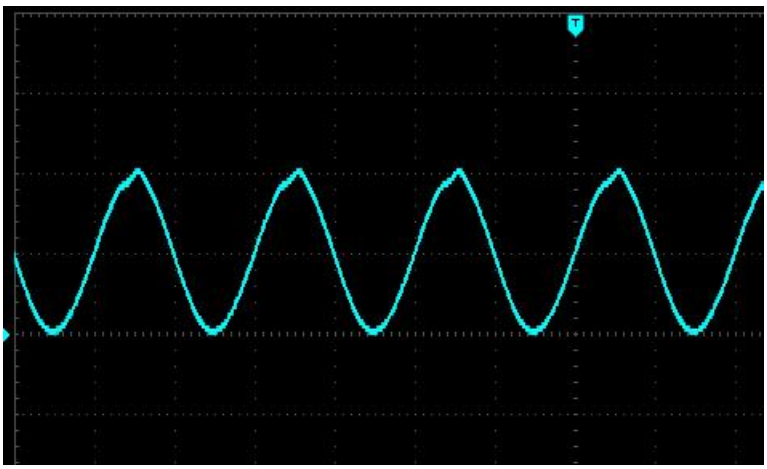
Harmonic Wave Configuration

Description

The harmonic wave has amplitude, offset, and phase.
It also adds total harmonic orders, harmonic amplitude and harmonic phase. Its amplitude and offset can be set by high and low voltage.

Example

The following waves can be set by the SCPI command series, and the high level and low level can be used to replace :CHANnel1:BASE:AMPLitude and :CHANnel1:BASE:OFFSet.



The above harmonic wave can be generated by following commands.

:CHANnel1:MODe CONTinue
:CHANnel1:BASE:WAVe HARMonic
:CHANnel1:BASE:FREQuency 1000
:CHANnel1:BASE:HIGH 1
:CHANnel1:BASE:LOW 0
:CHANnel1:BASE:PHAse 90
:CHANnel1:HARMonic:TOTal:ORDer 10
:CHANnel1:HARMonic:TYPe ALL
:CHANnel1:HARM:ORDER2:AMPL 0.02
:CHANnel1:HARM:ORDer2:PHASe 20
:CHANnel1:HARM:ORDER3:AMPL 0.01
:CHANnel1:HARM:ORDer3:PHASe 30
:CHANnel1:OUTPut ON

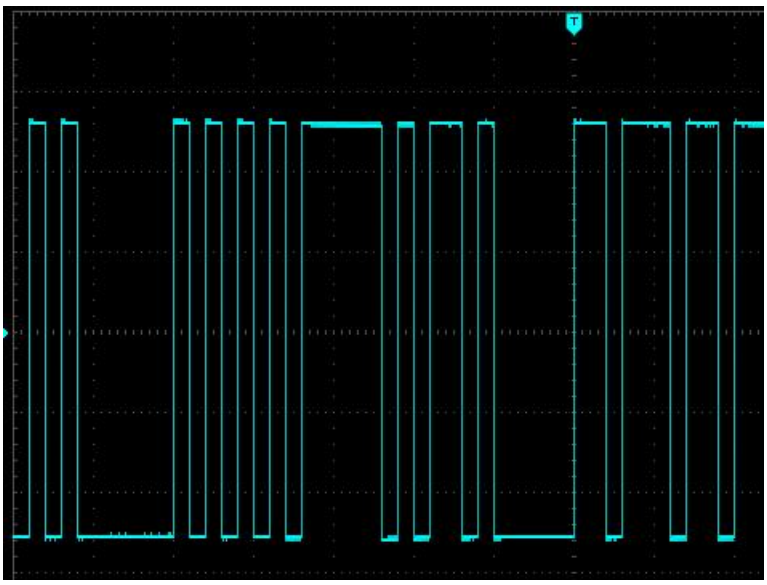
Pseudo-Random Wave Configuration

Description

The pseudo-random wave has bit ratio, offset, edge time, code element and etc. Its amplitude and offset can be set by high and low voltage.

Example

The following waves can be set by the SCPI command series, and the high level and low level can be used to replace :CHANnel1:BASE:AMPLitude and :CHANnel1:BASE:OFFSet.



The above pseudo-random wave can be generated by following commands.

:CHANnel1:MODe CONTinue
:CHANnel1:BASE:WAVe PRBS
:CHANnel1:PRBS:BITRatio 1000000
:CHANnel1:BASE:HIGH 1
:CHANnel1:BASE:LOW 0
:CHANnel1:PNCode PN9
:CHANnel1:OUTPut ON

Appendix 1: <key> List

Key	Function	LED Light
COUNtinue	Continue（CW）	√
MOD	Modulate	√
SWEep	Sweep	√
BURSt	Burst	√
SINe	Sine Wave	√
SQUare	Square Wave	√
RAMP	Ramp Wave	√
PULSe	Pulse Wave	√
ARB	Arbitrary Wave	√
HARMonic	Harmonic Wave	√
NOISe	Noise	√
DC	Direct Current	√
PRBS	Pseudo-Random Binary Sequence	√
WARB	Write Arbitrary Wave	√
CH1	CHANNEL 1	√
CH2	CHANNEL 2	√
CH3	CHANNEL 3	√
CH4	CHANNEL 4	√
UTILity	System	
RIGHT	RIGHT	
LEFT	LEFT	
OK	OK	
UP	UP	
DOWN	DOWN	
NUM0	NUMBER 0	
NUM1	NUMBER 1	
NUM2	NUMBER 2	
NUM3	NUMBER 3	
NUM4	NUMBER 4	
NUM5	NUMBER 5	
NUM6	NUMBER 6	
NUM7	NUMBER 7	
NUM8	NUMBER 8	
NUM9	NUMBER 9	
DOT	Number Dot	
SYMBOL	Number Symbol	
BACKspace	Backspace	