UNI-T®

Instruments.uni-trend.com

Programming Manual

UTG2000X Series Function/Arbitrary Waveform Generator

V1.1

2024.07.03

Warranty and Statement

Copyright

Brandmark

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

File Number

20240703

Software Version

V1.15.0019

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

Statement

- UNI-T products are protected by patents (including obtained and pending) in China and other countries and regions.
- UNI-T reserves the right to change specifications and prices.
- The information provided in this manual supersedes all previous publications.
- The information provided in this manual is subject to change without notice.
- **UNI-T** shall not be liable for any errors that may be contained in this manual. For any incidental or consequential damages arising out of the use or the information and deductive functions provided in this manual.
- No part of this manual shall be photocopied, reproduced or adapted without the prior written permission of **UNI-T**.

Product Certification

UNI-T has certified that the product conforms to China national product standard and industry product standard as well as ISO9001:2015 standard and ISO14001:2015 standard. UNI-T will go further to certificate product to meet the standard of other member of the international standards organization.

Contact Us

If you have any question or problem, please can contact **UNI-T**.

E-mail address：infosh@uni-trend.com.cn

Website：http://www.uni-trend.com

SCPI

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

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

Instruction Format

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

Symbol Description

The following four symbols are not part of the SCPI command. They cannot be sent with the command, but they are commonly used for supplementary specification.

- **Braces { }**

  The braces usually contains multiple optional parameters. It should select one parameter when sending a command.

  For example, :DISPlay: GRID: MODE {FULL | GRID | CROSS | NONE}

- **Vertical Bar |**

  The vertical bar is used to separate multiple parameters, it should select o one parameter when sending a command.

  For example, :DISPlay:GRID:MODE { FULL | GRID | CROSS | NONE}

- **Square Brackets [ ]**

  The contents in square brackets（command keywords）can be omissible. If the parameter is ignored, then this parameter will be the default value. For example, for the command :MEASure:NDUTy? [<source>], [<source>] represents the current channel.

- **Triangular Brackets < >**

  The parameter in triangular brackets must be replace with a valid value.

  For example, send the command :DISPlay:GRID:BRIGhtness 30 in the format of DISPlay:GRID:BRIGhtness <count>

Parameter Description

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

- **Boolean**

  The parameter can take value as "ON" (1) or "OFF" (0).

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

- **Integer**

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

  Note: At this point, do not set the parameter to decimal format, otherwise, an error will occur.

  For example, <count> in the command of :DISPlay:GRID:BRIGhtness <count>, which can take any valid integer value of 0~100.

- **Real**

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

  For example, for CH1, <offset> in the command of CHANnel1:OFFSet <offset>, which take value as real type.

- **Discrete**

  The parameter can only take some specified numbers or characters.

  For example, the parameter in the command :DISPlay:GRID:MODE { FULL | GRID | CROSS | NONE} cam only be FULL, GRID, CROSS, NONE.

- **ASCII String**

  String parameter contain all ASCII string sets. Strings must begin and end with paired quotes; it can use single or double quotation marks. The quotation and delimiter can also be part of a string by typing it twice and not adding any characters.

  For example, IP: SYST:COMM:LAN:IPAD "192.168.1.10"

Abbreviation Rule

All commands are case sensitivity. If command writes in abbreviation format, all capital letters in the command should input completely.

Data Return

Data return is divided into single data and batch data. The single data return is the corresponding parameter type, in which the real return type is express in scientific notation. The part before e retains three figure behind the decimal point, and the e part retains three figure; the batch return must be obey IEEE 488.2# string data format, '#'+ the length of character bits [fixed to one character] + ASCII valid value+ valid data+ end mark ['\n']

For example, #3123xxxxxxxxxxxxxxxxxxx\n represents 123 strings batch data return format, '3' represents "123" occupies three character bits.

SCPI

IEEE488.2 Common Command

*IDN?

- **Command Format**

  *IDN?

- **Functional Description**

  Query manufacture name, product model, product serial number and software version.

- **Return Format**

  Manufacture name, product model, product serial number and software version separated by dot mark.

  Note: The returned model should be same as the nameplate.

- **For Example**

  UNI-T Technologies, UTG2000X, 000000001, 00.00.01

*RST

- **Command Format**

  *RST

- **Functional Description**

  Restore factory settings and clear the entire error message, send and receive queue buffers.

SYSTem Command

The command is used for the basic operation of the waveform generator, including full keyboard lock and system data setting.

:SYSTem:LOCK

- **Command Format**

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

  :SYSTem:LOCK?

- **Functional Description**

  Lock/unlock the full keyboard and touch input.

- **Return Format**

  The query returns the lock state of full keyboard. 0 indicates that the full keyboard is unlock. 1

indicates that the full keyboard is locked.

- **For Example**

:SYSTem:LOCK ON	The full keyboard is locked.
:SYSTem:LOCK OFF	The full keyboard is unlock.
:SYSTem:LOCK?	Indicating that the full keyboard is locked.

:SYSTem:CONFigure

- **Command Format**

:SYSTem:CONFigure <file>

:SYSTem:CONFigure?

- **Functional Description**

Read/write the configuration file. Send the instruction at first, and then send the configuration file data to the signal source.

<file> represents the configuration file.

- **Return Format**

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

- **For Example**

:SYSTem:CONFigure	Write the configuration file data into the signal source and load it.
:SYSTem:CONFigure?	The query returns the binary data of the current configuration file data.

:SYSTem:PHASe:MODe

- **Command Format**

:SYSTem:PHASe:MODe {INDependent | SYNChronization}

:SYSTem:PHASe:MODe?

- **Functional Description**

Set the phase mode between the interchannels. If the phase mode is sync, then the start phase of the two channels keeps synchronized, otherwise, the phase is independent.

- **Return Format**

The query returns the phase mode between the interchannels.

- **For Example**

:SYSTem:PHASe:MODe INDependent	Set the phase mode to INDependent.
:SYSTem:PHASe:MODe?	The query returns INDependent.

:SYSTem:LANGuage

- **Command Format**

:SYSTem:LANGuage {ENGLish|CHINese}

:SYSTem:LANGuage?

- **Functional Description**

Set the system language.

- **Return Format**

The query returns the system language.

- **For 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?

- **Functional Description**

Control the switch of beep.

- **Return Format**

The query returns the state of beep switch.

- **For Example**

:SYSTem:BEEP ON	Turn on the beep.
:SYSTem:BEEP?	The query returns 1.

:SYSTem:NUMBer:FORMat

- **Command Format**

:SYSTem:NUMBer:FORMat {COMMa|SPACe|NONe}

:SYSTem:NUMBer:FORMat?

- **Functional Description**

Set the separator of system number format.

- **Return Format**

The query returns the separator of system number format.

- **For Example**

:SYSTem:NUMBer:FORMat NONe	Set the system number format to NONe.
:SYSTem:NUMBer:FORMat?	The query returns NONe.

:SYSTem:PICTure:FORMat

- **Command Format**

  :SYSTem:PICTure:FORMat { BMP | JPEG | PNG}

  :SYSTem:PICTure:FORMat?

- **Functional Description**

  Set the data format for acquiring image and local image.

- **Return Format**

  The query returns the image format { BMP | JPEG | PNG}.

- **For Example**

:SYSTem:PICTure:FORMat PNG	Set the image format as PNG.
:SYSTem:PICTure:FORMat?	The query returns PNG.

:SYSTem:BRIGhtness

- **Command Format**

  :SYSTem:BRIGhtness { 10|30|50|70|90|100}

  :SYSTem:BRIGhtness?

- **Functional Description**

  Control the backlight brightness level of the system.

- **Return Format**

  The query returns the backlight brightness level of the system.

- **For Example**

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

:SYSTem:SLEEP:TIMe

- **Command Format**

  :SYSTem:SLEEP:TIMe { CLOSe | 5MIN | 15MIN | 30MIN | 60MIN}

  :SYSTem:SLEEP:TIMe?

- **Fuonctional Description**

  Control the sleep time of system, the unit is minute.

- **Return Format**

  The query returns the sleep time.

- **For Example**

:SYSTem:SLEEP:TIMe 5 MIN	Set the sleep time to 5MIN.
:SYSTem:SLEEP:TIMe?	The query returns 5MIN.

:SYSTem:ECLK:STATus?

- **Command Format**

  :SYSTem:ECLK:STATus?

- **Functional Description**

  Query the state of external clock source.

- **Return Format**

  The query returns the state of external clock source. The query returns 0, indicating that the external clock source is invalid. The query returns 1, indicating that the external clock source is valid.

- **For Exampley**

:SYSTem:ECLK:STATus?	The query returns 1, indicating that the external clock source is valid.

:SYSTem:CYMometer

- **Command Format**

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

  :SYSTem:CYMometer?

- **Functional Description**

  Control the switch state of system's cymometer.

- **Return Format**

  The query returns the switch state of system's cymometer. The query returns 0 (OFF), indicating that the system's cymometer is turned off. The query returns 1 (ON), indicating that the system's cymometer is turned on.

- **For Example**

:SYSTem:CYMometer ON	Turn on system's cymometer.
:SYSTem:CYMometer?	The query returns 1.

:SYSTem:CYMometer:STATistics

- **Command Format**

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

- **Functional Description**

  Turn on/off the statistics function of the cymometer.

- **Return Format**

  The query returns the statistics state of the cymometer. The query returns 0 (OFF), indicating

that the statistics function of the cymometer is turned off. The query returns 1 (ON), indicating that the statistics function of the cymometer is turned on.

- **For Example**

:SYSTem:CYMometer:STATistics ON Turn on the statistics state of the cymometer.

:SYSTem:CYMometer:STATistics? The query returns 1.

:SYSTem:CYMometer:STATistics:TYPe

- **Command Format**

:SYSTem:CYMometer:STATistics:TYPe {FREQuency|PERiod|DUTY|PWIDTh|NWIDTh}

- **Functional Description**

Set the statistics type for cymometer.

- **Return Format**

The query returns the statistics type of the cymometer.

- **For Example**

:SYSTem:CYMometer:STATistics:TYPe FREQuency Set the statistics type to FREQuency.

:SYSTem:CYMometer:STATistics:TYPe? The query returns FREQuency.

:SYSTem:CYMometer:STATistics:COUNt?

- **Command Format**

:SYSTem:CYMometer:STATistics:COUNt?

- **Functional Description**

Acquire the current statistics count of the cymometer.

- **Return Format**

The query returns the current statistics count of the cymometer.

- **For Example**

:SYSTem:CYMometer:STATistics:COUNt? The query returns 40.

:SYSTem:CYMometer:STATistics:CURRent?

- **Command Format**

:SYSTem:CYMometer:STATistics:CURRent?

- **Functional Description**

Acquire the measured value of the cymometer for the current mode.

- **Return Format**

The query returns the measured value of the cymometer in scientific notation for the current mode.

- **For Example**

  :SYSTem:CYMometer:STATistics:CURRent? The query returns 1e+6.

:SYSTem:CYMometer:STATistics:MAX?

- **Command Format**

  :SYSTem:CYMometer:STATistics:MAX?

- **Functional Description**

  Acquire the maximum measured value of the cymometer for the current mode.

- **Return Format**

  The query returns the maximum measured value of the cymometer in scientific notation for the current mode.

- **For Example**

  :SYSTem:CYMometer:STATistics:MAX? The query returns 1e+7.

:SYSTem:CYMometer:STATistics:MIN?

- **Command Format**

  :SYSTem:CYMometer:STATistics:MIN?

- **Functional Description**

  Acquire the minimum measured value of the cymometer for the current mode.

- **Return Format**

  The query returns the minimum measured value of the cymometer in scientific notation for the current mode.

- **For Example**

  :SYSTem:CYMometer:STATistics:MIN? The query returns 1e-2.

:SYSTem:CYMometer:STATistics:MEAN?

- **Command Format**

  :SYSTem:CYMometer:STATistics:MEAN?

- **Functional Description**

  Acquire the averaged value of the cymometer for the current mode.

- **Return Format**

  The query returns the averaged value of the cymometer in scientific notation for the current mode.

- **For Example**

  :SYSTem:CYMometer:STATistics:MEAN? The query returns 1e+5.

:SYSTem:CYMometer:FREQuency?

- **Command Format**

  :SYSTem:CYMometer:FREQuency?

- **Functional Description**

  Acquire the current frequency of the cymometer.

- **Return Format**

  The query returns the current frequency value of the cymometer in scientific notation. The unit is Hz.

- **For Example**

  :SYSTem:CYMometer:FREQuency? The query returns 2e+3.

:SYSTem:CYMometer:PERiod?

- **Command Format**

  :SYSTem:CYMometer:PERiod?

- **Functional Description**

  Acquire the current period of the cymometer.

- **Return Format**

  The query returns the current period of the cymometer in scientific notation. The unit is s.

- **For Example**

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

:SYSTem:CYMometer:DUTY?

- **Command Format**

  :SYSTem:CYMometer:DUTY?

- **Functional Description**

  Acquire the current duty ratio of the cymometer.

- **Return Format**

  The query returns the current duty ratio of the cymometer in scientific notation. The unit is %.

- **For Example**

  :SYSTem:CYMometer:DUTY? The query returns 20，indicating that the duty ratio is 20 %.

:SYSTem:CYMometer:PWIDTh?

- **Command Format**

  :SYSTem:CYMometer:PWIDTh?

- **Functional Description**

  Acquire the current positive width pulse of the cymometer.

- **Return Format**

  The query returns the current positive width pulse of the cymometer. The unit is ms.

- **For Example**

  :SYSTem:CYMometer:PWIDTh? The query returns 1e-3，indicating that the duty ratio is 1 ms.

:SYSTem:CYMometer:NWIDTh?

- **Command Format**

  :SYSTem:CYMometer:NWIDTh?

- **Functional Description**

  Acquire the current negative width pulse of the cymometer.

- **Return Format**

  The query returns the current negative width pulse of the cymometer. The unit is ms.

- **For Example**

  :SYSTem:CYMometer:NWIDTh? The query returns 1e-3，indicating that the duty ratio is 1 ms.

:SYSTem:COMMunicate:LAN:APPLy

- **Command Format**

  :SYSTem:COMMunicate:LAN:APPLy

- **Functional Description**

  Set the network to take effect immediately.

:SYSTem:COMMunicate:LAN:GATEway

- **Command Format**

  :SYSTem:COMMunicate:LAN:GATEway <gateway>

  :SYSTem:COMMunicate:LAN:GATEway?

- **Functional Description**

  Set the default gateway. <gateway> belongs to ASCII string，the format is xxx.xxx.xxx.xxx.

- **Return Format**

  The query returns the default gateway.

- **For Example**

  :SYST:COMM:LAN:GATE "192.168.1.1" Set the default gateway to 192.168.1.1

:SYST:COMM:LAN:GATE? The query returns 1.92.168.1.1

:SYSTem:COMMunicate:LAN:SMASK

- **Command Format**

  :SYSTem:COMMunicate:LAN:SMASK <submask>

  :SYSTem:COMMunicate:LAN:SMASK?

- **Functional Description**

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

- **Return Format**

  The query returns the subnet mask.

- **For 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?

- **Functional Description**

  Set IP address. <ip> belongs to ASCII string, the format is xxx.xxx.xxx.xxx.

- **Return Format**

  The query returns IP address.

- **For Example**

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

  :SYST:COMM:LAN:IPAD? The query returns 1.92.168.1.10

:SYSTem:COMMunicate:LAN:DHCP

- **Command Format**

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

  :SYSTem:COMMunicate:LAN:DHCP?

- **Functional Description**

  Switch the DHCP to Auto IP or Manual IP.

- **Return Format**

  The query returns the DHCP. The query returns 0 (OFF), which indicates Manual IP. The query returns 1 (ON), which indicates Auto IP.

- **For Example**

  :SYST:COMM:LAN:DHCP ON Turn on DHCP.

  :SYST:COMM:LAN:DHCP? The query returns 1.

:SYSTem:COMMunicate:LAN:MAC?

- **Command Format**

  :SYSTem:COMMunicate:LAN:MAC?

- **Return Format**

  The query returns the physical address (MAC).

- **For Example**

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

CHANnel Command

This command is used to set the channel function.

:CHANnel<n>:MODE

- **Command Format**

  :CHANnel<n>:MODE {CONTinue | MODulation| SWEep| BURSt }

  :CHANnel<n>:MODE?

- **Functional Description**

  Set the mode for the specified channel signal, which can be set to CONTinue, MODulation, SWEep or BURSt.

  <n>: Channel number，n takes the value as 1 and 2.

- **Return Format**

  The query returns the mode of the specified channel signal.

- **For Example**

  :CHANnel1:MODE MODulation Set the mode of CH1 signal to MODulation.

  :CHANnel1:MODE? The query returns MODulation.

:CHANnel<n>:OUTPut

- **Command Format**

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

  :CHANnel<n>:OUTPut?

- **Functional Description**

Turn on/off the output of the specified channel.

<n>: Channel number，n takes the value as 1 and 2.

- **Return Format**

The query returns the output state of the specified channel. The query returns 0 (OFF), indicating that the output of the specified channel is turned off. The query returns 1 (ON), indicating that the output of the specified channel is turned on.

- **For Example**

:CHANnel1:OUTPut ON	Turn on CH1 output.
:CHANnel1:OUTPut?	The query returns 1.

:CHANnel<n>:INVersion

- **Command Format**

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

:CHANnel<n>:INVersion?

- **Functional Description**

Turn on/off the reversed output of the specified channel.

<n>: Channel number，n takes the value as 1 and 2.

- **Return Format**

The query returns the reversed output of the specified channel. The query returns 0 (OFF), indicating that the reversed output of the specified channel is turned off. The query returns 1 (ON), indicating that the reversed output of the specified channel is turned on.

- **For Example**

:CHANnel1:INVersion ON	Turn on the reversed output of CH1.
:CHANnel1:INVersion?	The query returns 1.

:CHANnel<n>:OUTPut:SYNC

- **Command Format**

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

:CHANnel<n>:OUTPut:SYNC?

- **Functional Description**

Set the sync output state of the specified channel.

Note：This device has only one sync output terminal, so it can only turn on the sync output of one channel.

<n>: Channel number，n takes the value as 1 and 2.

- **Return Format**

The query returns the sync output state of the specified channel. The query returns 0 (OFF), indicating that the sync output of the specified channel is turned off. The query returns 1 (ON), indicating that the sync output of the specified channel is turned on.

- **For Example**

:CHANnel1:OUTPut:SYNC ON	Turn on the sync output of CH1.
:CHANnel1:OUTPut:SYNC?	The query returns 1.

:CHANnel<n>:LIMit:ENABle

- **Command Format**

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

:CHANnel<n>:LIMit:ENABle?

- **Functional Description**

Turn on/off the amplitude limiting of the specified channel.

<n>: Channel number，n takes the value as 1 and 2.

- **Return Format**

The query returns the amplitude limiting state of the specified channel.

- **For Example**

:CHANnel1:LIMit:ENABle ON	Turn on the amplitude limiting of CH1.
:CHANnel1:LIMit:ENABle?	The query returns 1.

:CHANnel<n>:LIMit:LOWer

- **Command Format**

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

:CHANnel<n>:LIMit:LOWer?

- **Functional Description**

Set the lower limit of the amplitude limiting for the specified channel.

<voltage> represents the voltage, the unit is determined by the current channel.

<n>: Channel number，n takes the value as 1 and 2.

- **Return Format**

The query returns the lower limit of the amplitude limiting for the specified channel in scientific notation.

- **For Example**

:CHANnel1:LIMit:LOWer 2	Set the lower limit of the amplitude limiting for CH1 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?

- **Functional Description**

  Set the upper limit of the amplitude limiting for the specified channel.

  <voltage> represents the voltage, the unit is determined by the current channel.

  <n>: Channel number, n takes the value as 1 and 2.

- **Return Format**

  The query returns the upper limit of the amplitude limiting for the specified channel in scientific notation.

- **For Example**

  :CHANnel1:LIMit:UPPer 2 Set the lower limit of the amplitude limiting for CH1 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?

- **Functional Description**

  Set the unit of the specified channel amplitude output.

  <n>: Channel number, n takes the value as 1 and 2.

- **Return Format**

  The query returns the unit of the specified channel amplitude output.

- **For Example**

  :CHANnel1:AMPLitude:UNIT VPP Set the unit of CH1 amplitude output to VPP.

  :CHANnel1:AMPLitude:UNIT? The query returns VPP.

:CHANnel<n>:LOAD

- **Command Format**

  :CHANnel<n>:LOAD <resistance>

  :CHANnel<n>:LOAD?

- **Functional Description**

  Set the load of the specified channel output.

  <resistance> represents load resistance, the unit is Ω.

<n>: Channel number，n takes the value as 1 and 2.

Note：The resistance range is 1~1000000, 1000000 is corresponding to the high resistance.

- **Return Format**

The query returns the load of the specified channel output in scientific notation.

- **For Example**

:CHANnel1:LOAD 50　　Set CH1 output load to 50 Ω.

:CHANnel1:LOAD?　　The query returns 5e+1.

:CHANnel<n>:COPY

- **Command Format**

:CHANnel<n>:COPY

- **Functional Description**

Execute the channel copy function, this command does not support query.

<n>: Channel number，n takes the value as 1 and 2.

- **Return Format**

No return value.

- **For Example**

:CHANnel1:COPY　　Copy CH1 data to CH2.

:CHANnel<n>:PNCode

- **Command Format**

:CHANnel<n>:PNCode <code>

:CHANnel<n>:PNCode?

- **Functional Description**

Set PN code of the specified channel, this command is only valid for the waveform with PN code.

<code>：PN code,

{PN3|PN5|PN7|PN9|PN11|PN13|PN15|PN17|PN19|PN21|PN23|PN25|PN27|PN29|PN31|PN33}

<n>: Channel number，n takes the value as 1 and 2.

- **Return Format**

The query returns PN code of the specified channel.

- **For Example**

:CHANnel1:PNCode PN9　　Set PN code of the specified channel to PN9.

:CHANnel1:PNCode?　　The query returns PN9.

:CHANnel<n>:TRIGger:SOURce

- **Command Format**

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

  :CHANnel<n>:TRIGger:SOURce?

- **Functional Description**

  Set the trigger source for the specified channel, this command is only valid for the sweep and burst function.

  <n>: Channel number，n takes the value as 1 and 2.

- **Return Format**

  The query returns the trigger source of the specified channel.

- **For Example**

  :CHANnel1:TRIGger:SOURce INTernal　Set the trigger source of CH1 to INTernal.

  :CHANnel1:TRIGger:SOURce?　The query returns INTernal.

:CHANnel<n>:TRIGger:OUTPut

- **Command Format**

  :CHANnel<n>:TRIGger:OUTPut {CLOSe|OPEn}

  :CHANnel<n>:TRIGger:OUTPut?

- **Functional Description**

  Set the trigger output mode of the specified channel, this command is only valid for the sweep and burst function.

  <n>: Channel number，n takes the value as 1 and 2.

- **Return Format**

  The query returns the trigger output mode of the specified channel.

- **For Example**

  :CHANnel1:TRIGger:OUTPut OPEn　Set the trigger output mode of CH1 to OPEn.

  :CHANnel1:TRIGger:OUTPut?　The query returns OPEn.

:CHANnel<n>:MERge

- **Command Format**

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

  :CHANnel<n>:MERge?

- **Functional Description**

  Turn on/off the specified channel coupling output. If CH1 and CH2 is coupled, it can only output form CH1.

<n>: Channel number，n takes the value as 1 and 2.

- **Return Format**

The query returns the state of the specified channel coupling output.

- **For Example**

:CHANnel1:MERge ON	CH1 and CH2 is coupled and the signal is output form CH1.
:CHANnel1:MERge?	The query returns 1.

:CHANnel:COUPle<m>:ONOFf

- **Command Format**

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

:CHANnel:COUPle<m>:ONOFf?

- **Functional Description**

Turn on/off the specified channel coupling.

<m>: Channel number, m takes the value as 1.

The query return 1, indicating that CH1 is coupled with CH2.

- **Return Format**

The query returns the state of the specified channel coupling. The query returns 0 (OFF), indicating that the specified channel coupling is turned off. The query returns 1 (ON), indicating that the specified channel coupling is turned on.

- **For Example**

:CHANnel:COUPle1:ONOFf ON	Turn on the specified channel coupling.
:CHANnel:COUPle1:ONOFf?	The query returns 1.

:CHANnel:COUPle<m>:TYPe

- **Command Format**

:CHANnel:COUPle<m>:TYPe {PARAm|TRACk}

:CHANnel:COUPle<m>:TYPe?

- **Functional Description**

Set the coupling type of the specified channel, this command is only valid when the channel coupling is turned on.

<m>: Channel number, m takes the value as 1.

The query returns 1, indicating that CH1 is coupled with CH2.

- **Return Format**

The query returns the coupling type of the specified channel.

- **For Example**

:CHANnel:COUPle1:TYPe PARAm	Set the coupling type to PARAm.
:CHANnel:COUPle1:TYPe?	The query returns PARAm.

:CHANnel:COUPle<m>:FREQuency

- **Command Format**

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

  :CHANnel:COUPle<m>:FREQuency?

- **Functional Description**

  Set the frequency coupling of the channel. The channel coupling has only one type, CH1 coupling CH2. This command is only valid when the parameter follow is turned on.

  <m>: Channel number, m takes the value as 1.

  The query returns 1, indicating that CH1 is coupled with CH2.

- **Return Format**

  The query returns the switch state of channel coupling. The query returns 0 (OFF), indicating that the channel coupling is turned off. The query returns 1 (ON), indicating that the channel coupling is turned on.

- **For Example**

:CHANnel:COUPle1:FREQuency ON	Turn on CH1 coupling CH2.
: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?

- **Functional Description**

  Set the ratio of the channel frequency coupling. The channel coupling has only one type, CH1 coupling CH2. This command is only valid when the frequency follow is turned on.

  <scale>: The ratio of the frequency coupling.

  <m>: Channel number, m takes the value as 1.

  The query returns 1, indicating that CH1 is coupled with CH2.

- **Return Format**

  The query returns the ratio of the channel frequency coupling in scientific notation.

- **For Example**

:CHANnel:COUPle1:FREQuency:SCALe 0.1	Set CH2:CH1 coupling ratio 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?

- **Functional Description**

  Set the deviation of the channel frequency coupling. The channel coupling has only one type, CH1 coupling CH2. This command is only valid when the frequency follow is turned on.

  <scale>: The deviation of frequency coupling, the unit is Hz.

  <m>: Channel number, m takes the value as 1.

  The query returns 1, indicating that CH1 is coupled with CH2.

- **Return Format**

  The query returns the deviation of the channel frequency coupling in scientific notation.

- **For Example**

  :CHANnel:COUPle1:FREQuency:DEV 100 Set CH2:CH1 coupling ratio deviation 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?

- **Functional Description**

  Turn on/off the channel phase coupling. The channel coupling has only one type, CH1 coupling CH2. This command is only valid when the parameter follow is turned on.

  <m>: Channel number, m takes the value as 1.

  The query returns 1, indicating that CH1 is coupled with CH2.

- **Return Format**

  The query returns the state of channel coupling. The query returns 0 (OFF), indicating that the channel phase coupling is turned off. The query returns 1 (ON), indicating that the channel phase coupling is turned on.

- **For Example**

  :CHANnel:COUPle1:PHASe ON Turn on CH1 coupling CH2.

  :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?

- **Functional Description**

Set the ratio of channel phase coupling. The channel coupling has only one type, CH1 coupling CH2. This command is only valid when the phase follow is turned on.

<scale>: The ratio of channel phase coupling.

<m>: Channel number, m takes the value as 2.

The query returns 1, indicating that CH1 is coupled with CH2.

- **Return Format**

The query returns the ratio of channel phase coupling in scientific notation.

- **For Example**

:CHANnel:COUPle1:PHASe:SCALe 0.1 Set CH2:CH1 coupling ratio 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?

- **Functional Description**

Set the deviation of channel phase coupling. The channel coupling has only one type, CH1 coupling CH2. This command is only valid when the phase follow is turned on.

<scale>: The deviation of phase coupling, the unit is°.

<m>: Channel number, m takes the value as 1.

The query returns 1, indicating that CH1 is coupled with CH2.

- **Return Format**

The query returns the deviation of channel phase coupling in scientific notation.

- **For Example**

:CHANnel:COUPle1:PHASe:DEV 100 Set CH2:CH1 coupling ratio deviation 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?

- **Functional Description**

Set the channel amplitude coupling. The channel coupling has only one type, CH1 coupling CH2. This command is only valid when the parameter follow is turned on.

<m>: Channel number, m takes the value as 1.

The query returns 1, indicating that CH1 is coupled with CH2.

- **Return Format**

The query returns the state of the channel amplitude coupling. The query returns 0 (OFF), indicating that the channel amplitude coupling is turned off. The query returns 1 (ON), indicating that the channel amplitude coupling is turned on.

- **For Example**

:CHANnel:COUPle1:AMPLitude ON	Turn on CH1 coupling CH2.
: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?

- **Functional Description**

Set the ratio of channel amplitude coupling. The channel coupling has only one type, CH1 coupling CH2. This command is only valid when the amplitude follow is turned on.

<scale>: The ratio of coupling amplitude.

<m>: Channel number, m takes the value as 1.

The query returns 1, indicating that CH1 is coupled with CH2.

- **Return Format**

The query returns the ratio of channel amplitude coupling in scientific notation.

- **For Example**

:CHANnel:COUPle1:AMPLitude:SCALe 0.1	Set CH2:CH1 coupling ratio 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?

- **Functional Description**

Set the deviation of channel amplitude coupling. The channel coupling has only one type, CH1 coupling CH2. This command is only valid when the amplitude follow is turned on.

<scale>: The deviation of channel amplitude coupling, the unit is Vpp.

<m>: Channel number, m takes the value as 1.

The query returns 1, indicating that CH1 is coupled with CH2.

- **Return Format**

The query returns the deviation of channel amplitude coupling in scientific notation.

- **For Example**

:CHANnel:COUPle1:AMPLitude:DEV 1 Set CH2:CH1 coupling ratio deviation to 1 Vpp.

:CHANnel:COUPle1:AMPLitude:DEV? The query returns 1e+2.

:CHANnel:COUPle<m>:TRACk:PHASe:DEV

- **Command Format**

:CHANnel:COUPle<m>:TRACk:PHASe:DEV <dev >

:CHANnel:COUPle<m>:TRACk:PHASe:DEV?

- **Functional Description**

Set the phase deviation under the channel follow. The channel coupling has only one type, CH1 coupling CH2. This command is only valid when the channel follow is turned on.

<scale>: Phase deviation, the unit is °.

<m>: Channel number, m takes the value as 1.

The query returns 1, indicating that CH1 is coupled with CH2.

- **Return Format**

The query returns the phase deviation under the channel follow in scientific notation.

- **For Example**

:CHANnel:COUPle1:TRACk:PHASe:DEV 100 Set CH2:CH1 coupling ratio deviation 100°.

:CHANnel:COUPle1:TRACk:PHASe:DEV? The query returns 1e+2.

:CHANnel<n>:SELect

- **Command Format**

:CHANnel<n>:SELect

:CHANnel<n>:SELect?

- **Functional Description**

Select the channel.

<n>: {1|2}, which represents {CH1|CH2} respectively.

- **Return Format**

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

- **For Example**

:CHAN1:SELect Select CH1.

:CHAN1:SELect? The query returns 1, indicating that the channel is selected.

Continuous

:CHANnel<n>:BASE:WAVe

- **Command Format**

  :CHANnel<n>:BASE:WAVe { SINe | SQUare | PULSe | RAMP | ARB | NOISe | DC | HARMonic | PRBS|EXP|DPULse }

  :CHANnel<n>:BASE:WAVe?

- **Functional Description**

  Set the basic wave for the specified channel, which is sine wave, square wave, pulse wave, arbitrary wave, noise wave, DC wave, harmonic wave, PRBS, expression and double pulse.

  <n>: Channel number, n takes the value as 1 and 2.

- **Return Format**

  The query returns the basic wave of the specified channel.

- **For Example**

  :CHANnel1:BASE:WAVe SINe Set the basic wave of CH1 as SINe.

  :CHANnel1:BASE:WAVe? The query returns SINe.

:CHANnel<n>:BASE:FREQuency

- **Command Format**

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

  :CHANnel<n>:BASE:FREQuency?

- **Functional Description**

  Set the output frequency of the specified channel.

  <freq> represents the frequency, the unit is Hz. (1e-6Hz ~ Maximum frequency of the current waveform).

  <n>: Channel number, n takes the value as 1 and 2.

- **Return Format**

  The query returns the output frequency of the specified channel in scientific notation.

- **For Example**

  :CHANnel1:BASE:FREQuency 2000 Set the output frequency of CH1 to 2 kHz.

  :CHANnel1:BASE:FREQuency? The query returns 2e+3.

:CHANnel<n>:BASE:PERiod

- **Command Format**

  :CHANnel<n>:BASE:PERiod { <period>}

  :CHANnel<n>:BASE:PERiod?

- **Functional Description**

  Set the output period of the specified channel.

  <period> represents the period, the unit is S.

  If it is sine wave, the range is (the current maximum time ~ 1e3s)

  <n>: Channel number, n takes the value as 1 and 2.

- **Return Format**

  The query returns the output period of the specified channel in scientific notation.

- **For Example**

  :CHANnel1:BASE:PERiod 0.002 Set the output period of CH1 to 2 ms.

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

:CHANnel<n>:BASE:PHASe

- **Command Format**

  :CHANnel<n>:BASE:PHASe { <phase>}

  :CHANnel<n>:BASE:PHAse?

- **Functional Description**

  Set the output phase of the specified channel.

  <phase> represents the frequency, the unit is ° and the range is -360~360.

  <n>: Channel number, n takes the value as 1 and 2.

- **Return Format**

  The query returns the output phase of the specified channel.

- **For Example**

  :CHANnel1:BASE:PHAse 20 Set the output phase of CH1 is 20° .

  :CHANnel1:BASE:PHAse? The query returns 20.

:CHANnel<n>:BASE:AMPLitude

- **Command Format**

  :CHANnel<n>:BASE:AMPLitude { <amp>}

  :CHANnel<n>:BASE:AMPLitude?

- **Functional Description**

  Set the output amplitude of the specified channel.

<amp> represents the voltage，the unit is determined by the current channel. The range is 1mVpp ~ the maximum output under the current load.

If the current unit is VPP，the maximum under the current load = the current load *20/(50+the current load).

<n>: Channel number，n takes the value as 1 and 2.

- **Return Format**

The query returns the output amplitude of the specified channel in scientific notation.

- **For Example**

:CHANnel1:BASE:AMPLitude 2	Set the output amplitude of CH1 as 2 V.
:CHANnel1:BASE:AMPLitude?	The query returns 2e+0.

:CHANnel<n>:BASE:OFFSet

- **Command Format**

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

:CHANnel<n>:BASE:OFFSet?

- **Functional Description**

Set the DC offset output of the specified channel.

<voltage> represents the voltage，the unit is V. The range is 0~± the maximum DC under the current load.

The maximum DC under the current load = the current load *10/(50+ the current load) − the minimum of the current AC/2;

the minimum of AC is 2 mVpp, DC mode takes value as 0;

<n>: Channel number，n takes the value as 1 and 2.

- **Return Format**

The query returns the DC offset output of the specified channel in scientific notation.

- **For Example**

:CHANnel1:BASE:OFFSet 2	Set the DC offset output of CH1 as 2 V.
:CHANnel1:BASE:OFFSet?	The query returns 2e+0.

:CHANnel<n>:BASE:HIGH

- **Command Format**

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

:CHANnel<n>:BASE:HIGH?

- **Functional Description**

Set the high value of the specified channel signal output.

<voltage> represents the voltage，the unit is determined by the current channel.

<n>: Channel number，n takes the value as 1 and 2.

- **Return Format**

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

- **For Example**

:CHANnel1:BASE:HIGH 2　　Set the high value of CH1 signal output as 2 V.

:CHANnel1:BASE:HIGH?　　The query returns 2e+0.

:CHANnel<n>:BASE:LOW

- **Command Format**

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

:CHANnel<n>:BASE:LOW?

- **Functional Description**

Set the low value of the specified channel signal output.

<voltage> represents the voltage，the unit is determined by the current channel.

<n>: Channel number，n takes the value as 1 and 2.

- **Return Format**

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

- **For Example**

:CHANnel1:BASE:LOW 2　　Set the low value of CH1 signal output as 2 V.

:CHANnel1:BASE:LOW?　　The query returns 2e+0.

:CHANnel<n>:BASE:DUTY

- **Command Format**

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

:CHANnel<n>:BASE:DUTY?

- **Functional Description**

Set the duty ratio of the specified channel signal output.

<duty> represents the duty ratio，the unit is % and the range is 0~100.

<n>: Channel number，n takes the value as 1 and 2.

- **Return Format**

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

- **For Example**

:CHANnel1:BASE:DUTY 20　　Set the duty ratio of CH1 signal output as 20 %.

:CHANnel1:BASE:DUTY?　　The query returns 20.

:CHANnel<n>:BASE:ARB

- **Command Format**

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

  :CHANnel<n>:BASE:ARB?

- **Functional Description**

  Load the arbitrary waveform data of a file under the fundamental arbitrary waveform source to the specified channel.

  <n>: Channel number，n takes the value as 1 and 2.

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

  <filename>：The name of arbitrary wave file.

- **For Example**

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

:CHANnel<n>:BASE:ARB:SAMPling

- **Command Format**

  :CHANnel<n>:BASE:ARB:SAMPling { <sampling>}

  :CHANnel<n>:BASE:ARB:SAMPling?

- **Functional Description**

  Set the sampling rate for the arbitrary wave of the specified channel.

  Note：This command is only valid for the point by point mode of the arbitrary wave.

  <n>: Channel number，n takes the value as 1 and 2.

  { <sampling>} represents the sampling rate，the unit is Sa/s and the range is 1e-6~625e+6Sa/s.

- **For Example**

:CHANnel1:BASE:ARB:SAMPling 1000	Set the sampling rate of CH1 arbitrary wave as 1000Sa/s.
:CHANnel1:BASE:ARB:SAMPling?	The query returns 1e+3.

:CHANnel<n>:RAMP:SYMMetry

- **Command Format**

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

  :CHANnel<n>:RAMP:SYMMetry?

- **Functional Description**

  Set the symmetry for the ramp signal output of the specified channel.

  <symmetry> represents the symmetry, the unit is % and the range is 0~100.

  <n>: Channel number，n takes the value as 1 and 2.

- **Return Format**

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

- **For Example**

  :CHANnel1:RAMP:SYMMetry 20 Set the symmetry of CH1 ramp signal as 20 %.

  :CHANnel1:RAMP:SYMMetry? The query returns 20.

:CHANnel<n>:PULSe:RISe

- **Command Format**

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

  :CHANnel<n>:PULSe:RISe?

- **Functional Description**

  Set the pulse width of rising edge for the pulse wave of the specified channel.

  <width> represents the pulse width, the unit is S.

  <n>: Channel number, n takes the value as 1 and 2.

- **Return Format**

  The query returns the pulse width of rising edge for the pulse wave in the specified channel in scientific notation.

- **For Example**

  :CHANnel1:PULSe:RISe 0.002 Set the pulse width of rising edge for CH1 signal to 2 ms .

  :CHANnel1:PULSe:RISe? The query returns 2e-3.

:CHANnel<n>:PULSe:FALL

- **Command Format**

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

  :CHANnel<n>:PULSe:FALL?

- **Functional Description**

  Set the pulse width of falling edge for the pulse wave of the specified channel.

  <width> represents the pulse width, the unit is S.

  <n>: Channel number, n takes the value as 1 and 2.

- **Return Format**

  The query returns the pulse width of falling edge for the pulse wave in the specified channel in scientific notation.

- **For Example**

  :CHANnel1:PULSe:FALL 0.002 Set the pulse width of falling edge for CH1 signal to 2 ms .

  :CHANnel1:PULSe:FALL? The query returns 2e-3.

:CHANnel<n>:PRBS:EDGEtime

- **Command Format**

  :CHANnel<n>:PRBS:EDGEtime <time>

  :CHANnel<n>:PRBS:EDGEtime ?

- **Functional Description**

  Set the edge time for the specified pseudo-random wave, this command is only valid for the pseudo-random wave.

  <time> represents the edge time，the unit is s.

  <n>: Channel number，n takes the value as 1 and 2.

- **Return Format**

  The query returns the edge time of the specified pseudo-random wave in scientific notation.

- **For Example**

  :CHANnel1:PRBS:EDGEtime 1　　Set the edge time of CH1 pseudo-random wave as 1s.

  :CHANnel1:PRBS:EDGEtime?　　The query returns 1e+0.

:CHANnel<n>:PRBS:BITRatio

- **Command Format**

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

  :CHANnel<n>:PRBS:BITRatio?

- **Functional Description**

  Set the bitrate for the specified pseudo-random, this command is only valid for the pseudo-random wave.

  <ratio> represents the bitrate, the unit is bps.

  <n>: Channel number，n takes the value as 1 and 2.

- **Return Format**

  The query returns the bitrate of the specified pseudo-random in scientific notation

- **For Example**

  :CHANnel1:PRBS:BITRatio 1000000　　Set the bitrate of CH1 as 1 Mbps.

  :CHANnel1:PRBS:BITRatio?　　The query returns 1e+6.

:CHANnel<n>:NOISe:BANDwith

- **Command Format**

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

  :CHANnel<n>:NOISe:BANDwith?

- **Functional Description**

Set the bandwidth for the noise signal of the specified channel.

<width> represents the bandwidth, the unit is Hz.

<n>: Channel number, n takes the value as 1 and 2.

- **Return Format**

The query returns the bandwidth of the noise signal in the specified channel in scientific notation.

- **For Example**

:CHANnel1:NOISe:BANDwith 2MHz Set the bandwidth of CH1 noise signal as 2 MHz.

: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?

- **Functional Description**

Set the harmonic type of the specified channel.

<n>: Channel number, n takes the value as 1 and 2.

- **Return Format**

The query returns the harmonic type of the specified channel.

- **For Example**

:CHANnel1:HARMonic:TYPe ODD Set the harmonic type of CH1 as 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?

- **Functional Description**

Set the maximum harmonic order of the specified channel.

<order>: Harmonic number, the range is 2~16.

<n>: Channel number, n takes the value as 1 and 2.

- **Return Format**

The query returns the maximum harmonic order of the specified channel, it is integer data.

- **For Example**

:CHANnel1:HARMonic:TOTal:ORDer 2 Set the maximum harmonic order of CH1 as 2.

: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?

- **Functional Description**

  Set the user-defined harmonic for the specified channel.

  < order >: User-defined harmonic, #H represents hexadecimal notation, X0111 1111 1111 1111 represents the harmonic switch.

  <n>: Channel number, n takes the value as 1 and 2.

- **Return Format**

  The query returns the user-defined harmonic of the specified channel, it is integer data.

- **For Example**

  :CHANnel1:HARMonic:USER:TYPe #H7FFF Set the user-defined harmonic for CH1.

  :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?

- **Functional Description**

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

  < amp >: Amplitude value, the unit is Vpp.

  <n>: Channel number, n takes the value as 1 and 2.

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

- **Return Format**

  The query returns the amplitude value of the harmonic order in the specified channel in scientific notation.

- **For Example**

  :CHANnel1:HARM:ORDER2:AMPL 0.02 Set the amplitude value of the second harmonic as 20 mVpp in CH1.

  :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?

- **Functional Description**

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

  <phase>: Phase value, the unit is °.

  <n>: Channel number, n takes the value as 1 and 2.

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

- **Return Format**

  The query returns the phase value of the harmonic order in the specified channel in scientific notation.

- **For Example**

:CHANnel1:HARM:ORDer2:PHASe 20	Set the phase value of the second harmonic as 20° in CH1.
: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?

- **Functional Description**

  Set the output mode for the arbitrary wave of the specified channel, which is DDS and point by point.

  <n>: Channel number, n takes the value as 1 and 2.

- **Return Format**

  The query returns the output mode of the arbitrary wave in the specified channel.

- **For Example**

:CHANnel1:ARB:MODe DDS	Set the output mode of CH1 arbitrary wave as DDS.
:CHANnel1:ARB:MODe?	The query returns DDS.

:CHANnel<n>:ARB:FILTer

- **Command Format**

  :CHANnel<n>:ARB:FILTer {ZEROHOLD | LINE }

  :CHANnel<n>:ARB:FILTer?

- **Functional Description**

  Set the interpolation method for the arbitrary wave output of the specified channel, which is zero-order hold and linear interpolation.

  <n>: Channel number，n takes the value as 1 and 2.

- **Return Format**

  The query returns the interpolation method of the arbitrary wave output in the specified channel.

- **For Example**

  :CHANnel1:ARB:FILTer LINE　　Output the arbitrary wave of CH1 by LINE method.

  :CHANnel1:ARB:FILTer?　　The query returns LINE.

:CHANnel<n>: EXP:EXPStart

- **Command Format**

  :CHANnel<n>:EXP:EXPStart { < start value>}

  :CHANnel<n>:EXP:EXPStart?

- **Functional Description**

  Set the start value（minimum）for the expression signal output of the specified channel.

  < start value> represents the start value（minimum），which input with radian.

  <n>: Channel number，n takes the value as 1 and 2.

- **Return Format**

  The query returns the start value（minimum）of the expression signal output in the specified channel in scientific notation.

- **For Example**

  :CHANnel1:EXP:EXPStart 0.1　　Set the start value of CH1 expression signal as 0.1.

  :CHANnel1:EXP:EXPStart?　　The query returns 1e-01.

:CHANnel<n>: EXP:EXPEnd

- **Command Format**

  :CHANnel<n>:EXP:EXPEnd { < end value>}

  :CHANnel<n>:EXP:EXPEnd?

- **Functional Description**

  Set the end value（maximum）for the expression signal output of the specified channel.

  < end value> represents the end value（maximum），which input with radian.

  <n>: Channel number，n takes the value as 1 and 2.

- **Return Format**

The query returns the end value（maximum）of the expression signal output in the specified channel in scientific notation.

- **For Example**

:CHANnel1:EXP:EXPEnd 3.1415926	Set the end value（maximum）of CH1 expression signal as 3.1415926.
:CHANnel1:EXP:EXPEnd?	The query returns 3.141593e+00.

:CHANnel<n>: EXP:EXPStr

- **Command Format**

:CHANnel<n>:EXP:EXPStr { < expstr>}

:CHANnel<n>:EXP:EXPStr?

- **Functional Description**

Set the expression formula for the expression signal of the specified channel. The independent variable can only be written as x, and all the letters in the formula should be written in lower case.

< expstr > represents the expression formula，which input with character string.

Supports function：sin(x), cos(x), tan(x), sinc(x), abs(x), lg(x), ln(x), sqrt(x), acos(x), asin(x), atan(x), sinh(x), tanh(x), ceil(x), cosh(x), exp(x), fabs(x), floor(x).

The operator cannot be omissible between the numeric variable. For example, 3*x cannot be wirte written 3x. The expression cannot contain the space.

<n>: Channel number，n takes the value as 1 and 2.

- **Return Format**

The query returns the expression formula of the expression signal in the specified channel in character formula.

- **For Example**

:CHANnel1:EXP:EXPStr "sin(x)"	Set the expression formula of CH1 expression signal is sin(x).
:CHANnel1:EXP:EXPStr?	The query returns sin(x).

:CHANnel<n>:DPULse:DELay

- **Command Format**

:CHANnel<n>:DPULse:DELay <delay>

:CHANnel<n>:DPULse:DELay?

- **Functional Description**

Set the DoublePulse delay for the specified channel.

< delay >: Pulse delay value, with the unit "s".

<n>: Channel number, n can take values of 1 or 2.

➢ **Return Format**

The query returns the pulse delay of the specified channel in scientific notation.

➢ **For Example**

:CHANnel1:DPULse:DELay 0.02 Set the DoublePulse delay in Channel 1 to 20 ms.

:CHANnel1:DPULse:DELay? The query returns 2e-2.

:CHANnel<n>:DPULse:TOTal:ORDer

➢ **Command Format**

:CHANnel<n>:DPULse:TOTal:ORDer <order>

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

➢ **Functional Description**

Set the maximum pulse of the DoublePulse for the specified channel.

< order >: Maximum pulse of the DoublePulse, ranging from 2-30.

<n>: Channel number, n can take values of 1 or 2.

➢ **Return Format**

The query returns the maximum pulse of the DoublePulse for the specified channel as an integer.

➢ **For Example**

:CHANnel1:DPULse:TOTal:ORDer 5 Set the maximum pulse of the DoublePulse for Channel 1 to 5.

:CHANnel1:DPULse:TOTal:ORDer? The query returns 5.

:CHANnel<n>:DPULse:ORDer

➢ **Command Format**

:CHANnel<n>:DPULse:ORDer <order>

:CHANnel<n>:DPULse:ORDer?

➢ **Functional Description**

Set the current DoublePulse order for the specified channel.

< order >: DoublePulse order, ranging from 0-29.

<n>: Channel number, n can take values of 1 or 2.

➢ **Return Format**

The query returns the current DoublePulse order of the specified channel as an integer.

➢ **For Example**

:CHANnel1:DPULse:ORDer 1 Set the current DoublePulse order for Channel 1 to 1.

:CHANnel1:DPULse:ORDer? The query returns 1.

:CHANnel<n>:DPULse:ORDer<m>:PULSewidth

➢ **Command Format**

:CHANnel<n>:DPULse:ORDer<m>:PULSewidth <width>

:CHANnel<n>:DPULse:ORDer<m>:PULSewidth?

➢ **Functional Description**

Set the pulse width of the specified DoublePulse order under the specified channel.

< width >: Pulse width, with the unit "s".

<n>: Channel number, n can take values of 1 or 2.

<m>: DoublePulse order, m can take values from 0-29.

➢ **Return Format**

The query returns the pulse width of the specified DoublePulse order under the specified channel in scientific notation.

➢ **For Example**

:CHANnel1:DPULse:ORDer2:PULSewidth 0.02 Set the pulse width of the third pulse under Channel 1 to 20 ms.

:CHANnel1:DPULse:ORDer2:PULSewidth? The query returns 2e-2.

:CHANnel<n>:DPULse:ORDer<m>:GAP

➢ **Command Format**

:CHANnel<n>:DPULse:ORDer<m>:GAP <gap>

:CHANnel<n>:DPULse:ORDer<m>:GAP?

➢ **Functional Description**

Set the pulse interval of the specified DoublePulse order under the specified channel.

< gap >: Pulse interval, with the unit "s".

<n>: Channel number, n can take values of 1 or 2.

<m>: DoublePulse order, m can take values from 0-29.

➢ **Return Format**

The query returns the pulse interval of the specified DoublePulse order under the specified channel in scientific notation.

➢ **For Example**

:CHANnel1:DPULse:ORDer2:GAP 0.02 Set the pulse interval of the third pulse under

Channel 1 to 20ms.

:CHANnel1:DPULse:ORDer2:GAP? The query returns 2e-2.

:CHANnel<n>:DPULse:ORDer<m>:RISe

- **Command Format**

:CHANnel<n>:DPULse:ORDer<m>:RISe <rise>

:CHANnel<n>:DPULse:ORDer<m>:RISe?

- **Functional Description**

Set the rising edge of the specified DoublePulse order under the specified channel.

< rise >: Rising edge of the DoublePulse, with the unit "s".

<n>: Channel number, n can take values of 1 or 2.

<m>: DoublePulse order, m can take values from 0-29.

- **Return Format**

The query returns the rising edge of the specified DoublePulse order under the specified channel in scientific notation.

- **For Example**

:CHANnel1:DPULse:ORDer2:RISe 0.02 Set the rising edge of the third pulse under Channel 1 to 20 ms.

:CHANnel1:DPULse:ORDer2:RISe? The query returns 2e-2.

:CHANnel<n>:DPULse:ORDer<m>:FALl

- **Command Format**

:CHANnel<n>:DPULse:ORDer<m>:FALl <fall>

:CHANnel<n>:DPULse:ORDer<m>:FALl?

- **Functional Description**

Set the falling edge of the specified DoublePulse order under the specified channel.

< fall >: Falling edge of the DoublePulse, with the unit "s".

<n>: Channel number, n can take values of 1 or 2.

<m>: DoublePulse order, m can take values from 0-29.

- **Return Format**

The query returns the falling edge of the specified DoublePulse order under the specified channel in scientific notation.

- **For Example**

:CHANnel1:DPULse:ORDer2:FALl 0.02 Set the falling edge of the third pulse under Channel 1 to 20 ms.

:CHANnel1:DPULse:ORDer2:FALl? The query returns 2e-2.

Modulation

:CHANnel<n>:MODulate:TYPe

- **Command Format**

  :CHANnel<n>:MODulate:TYPe <type>

  :CHANnel<n>:MODulate:TYPe?

- **Functional Description**

  Set the signal modulation type for the specified channel.

  <type>: {AM|DSBAM|QAM|ASK|FM|FSK|ThreeFSK|FourFSK|PM|PSK|BPSK|QPSK|OSK|PWM|SUM} represents amplitude modulation, double-sideband amplitude modulation, quadrature modulation, amplitude shift keying, frequency modulation, frequency shift keying, three frequency shift keying, four frequency shift keying, phase modulation, binary phase shift keying, quadrature phase shift keying, oscillation keying, pulse width modulation and SUM modulation.

  <n>: Channel number, n takes the value as 1 and 2.

- **Return Format**

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

- **For Example**

  :CHANnel1:MODulate:TYPe AM Set the modulation type of CH1 signal as 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?

- **Functional Description**

  Set the modulation wave for the specified channel, which is sine wave, square wave, upper triangular, low triangular, arbitrary wave and noise.

  <n>: Channel number, n takes the value as 1 and 2.

- **Return Format**

  The query returns the modulation wave of the specified channel.

- **For Example**

  :CHANnel1:MODulate:WAVe SINe Set the modulation wave of CH1 signal as SINe.

  :CHANnel1:MODulate:WAVe? The query returns SINe.

:CHANnel<n>:MODulate:SOURce

- **Command Format**

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

  :CHANnel<n>:MODulate:SOURce?

- **Functional Description**

  Set the modulation source for the specified channel, which is internal and external.

  <n>: Channel number，n takes the value as 1 and 2.

- **Return Format**

  The query returns the modulation source of the specified channel

- **For Example**

  :CHANnel1:MODulate:SOURce INTernal　　Set the modulation source of CH1 as INTernal.

  :CHANnel1:MODulate:SOURce?　　The query returns INTernal.

:CHANnel<n>:MODulate:FREQuency

- **Command Format**

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

  :CHANnel<n>:MODulate:FREQuency?

- **Functional Description**

  Set the modulation frequency of the specified channel signal.

  <freq> represents the frequency，the unit is Hz.

  <n>: Channel number，n takes the value as 1 and 2.

- **Return Format**

  The query returns the modulation frequency of the specified channel signal in scientific notation.

- **For Example**

  :CHANnel1:MODulate:FREQuency 2000　　Set the modulation frequency of CH1 as 2 KHz.

  :CHANnel1:MODulate:FREQuency?　　The query returns 2e+3.

:CHANnel<n>:MODulate:IQMap

- **Command Format**

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

  :CHANnel<n>:MODulate: IQMap?

- **Functional Description**

  Set the IQ type for the specified QAM, which is

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

< IQ TYPE > represents IO map type.

<n>: Channel number，n takes the value as 1 and 2.

- **Return Format**

The query returns the IQ type of the specified channel.

- **For Example**

:CHANnel1:MODulate:IQMap QAM32 Set the IQ type of CH1 as QAM32.

:CHANnel1:MODulate:IQMap? The query returns QAM32.

:CHANnel<n>:MODulate:ARB

- **Command Format**

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

:CHANnel<n>:MODulate:ARB?

- **Functional Description**

Load an arbitrary wave data in modulation arbitrary wave source to the specified channel.

Note: If the source is external, the external storage device should create an ArbWave catalogue in the root directory to save the arbitrary wave.

<n>: Channel number，n takes the value as 1 and 2.

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

<filename>：The name of arbitrary wave file.

- **For Example**

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

:CHANnel<n>:MODulate:DEPTh

- **Command Format**

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

:CHANnel<n>:MODulate:DEPTh?

- **Functional Description**

Set the modulation depth for the specified channel.

<depth> represents the modulation depth，the unit is % and the range is 0% ~ 100%. The modulation depth of AM is 0% ~ 120%.

<n>: Channel number，n takes the value as 1 and 2.

- **Return Format**

The query returns the modulation depth of the specified channel.

- **For Example**

:CHANnel1:MODulate:DEPTh 50 Set the modulation depth of CH1 as 50%.

:CHANnel1:MODulate:DEPTh? The query returns 50.

:CHANnel<n>:MODulate:BITRatio

- **Command Format**

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

:CHANnel<n>:MODulate:BITRatio?

- **Functional Description**

Set the bitrate for the specified channel, this command is only valid for the waveform with ratio function.

<ratio> represents the bitrate，the unit is Hz.

<n>: Channel number，n takes the value as 1 and 2.

- **Return Format**

The query returns the bitrate of the specified channel in scientific notation.

- **For Example**

:CHANnel1:MODulate:BITRatio 100000 Set the bitrate of CH1 as 100 KHz.

:CHANnel1:MODulate:BITRatio? The query returns 1e+5.

:CHANnel<n>:MODulate:RATio

- **Command Format**

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

:CHANnel<n>:MODulate:RATio?

- **Functional Description**

Set the modulation bitrate for the specified channel, this command is only valid for the waveform with ratio function.

<ratio> represents the bitrate，the unit is Hz.

<n>: Channel number，n takes the value as 1 and 2.

- **Return Format**

The query returns the modulation bitrate of the specified channel in scientific notation.

- **For Example**

:CHANnel1:MODulate:RATio 100 Set the modulation bitrate of CH2 as 100 Hz.

:CHANnel1:MODulate:RATio? The query returns 1e+2.

:CHANnel<n>:FM:FREQuency:DEV

- **Command Format**

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

:CHANnel<n>:FM:FREQuency:DEV?

- **Functional Description**

Set the frequency deviation for the specified channel.

<freq> represents the frequency deviation，the unit is Hz and the range is 0Hz ~ the current fundamental frequency.

<n>: Channel number，n takes the value as 1 and 2.

- **Return Format**

The query returns the frequency deviation of the specified channel in scientific notation.

- **For Example**

:CHANnel1:FM:FREQuency:DEV 2000	Set the frequency deviation of CH1 as 2 KHz.
: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?

- **Functional Description**

Set the phase deviation for the specified channel.

< phase > represents the phase deviation，the unit is ° and the range is 0~360.

<n>: Channel number，n takes the value as 1 and 2.

- **Return Format**

The query returns the phase deviation of the specified channel.

- **For Example**

:CHANnel1:PM:PHASe:DEV 30	Set the phase deviation of CH1 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?

- **Functional Description**

Set the duty ratio deviation of the specified channel in pulse width modulation.

< duty > represents the duty ratio deviation，the unit is % and the range is 0~100.

<n>: Channel number，n takes the value as 1 and 2.

- **Return Format**

The query returns the duty ratio deviation of the specified channel in pulse width modulation in scientific notation.

- **For Example**

:CHANnel1:PWM:DUTY:DEV 10 Set the duty ratio deviation of CH1 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>?

- **Functional Description**

Set the hopping frequency output for frequency shift keying modulation in the specified channel. This command is only valid if the modulation mode has already been assigned.

< freq > represents the frequency, the unit is Hz.

<n>: Channel number, n takes the value as 1 and 2.

<m>: Frequency number, it takes the value as 1 when 2FSK 1; it takes the value as 1 ,2 when 3FSK; it takes the value as 1, 2, 3 when 4FSK.

- **Return Format**

The query returns the hopping frequency output of the specified channel in scientific notation.

- **For Example**

:CHANnel1:FSK:FREQ1 2000 Set the hopping frequency of CH1 to 2 KHz.

: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>?

- **Functional Description**

Set the phase output for phase shift keying modulation in the specified channel. This command is only valid if the modulation mode has already been assigned.

< phase> represents the phase, the unit is °and the range is -360~360.

<n>: Channel number, n takes the value as 1 and 2.

<m>: Phase number, it takes the value as 1 when PSK; it takes the value as 1, 2 when BPSK; it takes the value as 1, 2, 3, 4 when QPSK.

- **Return Format**

The query returns the phase of phase shift keying modulation in the specified channel in scientific notation.

- **For Example**

:CHANnel1:PSK:PHAS1 90	Set the phase output of CH1 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?

- **Functional Description**

Set the oscillation time for oscillation keying modulation in the specified channel.

<time> represents the oscillation time，the unit is S.

<n>: Channel number，n takes the value as 1 and 2.

- **Return Format**

The query returns the oscillation time of oscillation keying modulation in the specified channel in scientific notation.

- **For Example**

:CHANnel1:OSK:TIMe 0.002	Set the oscillation time of CH1 oscillation keying modulation to 2 ms.
:CHANnel1:OSK:TIMe?	The query returns 2e-3.

:CHANnel<n>:TRIGger:OUTEdge

- **Command Format**

:CHANnel<n>:TRIGger:OUTEdge { RISe|FALL}

:CHANnel<n>:TRIGger:OUTEdge?

- **Functional Description**

Set the output trigger edge for the specified channel. This command is only valid for the sweep frequency and burst function and only available when the sweep frequency and burst function are enabled.

<n>: Channel number，n takes the value as 1 and 2.

- **Return Format**

The query returns the output trigger edge of the specified channel.

- **For Example**

:CHANnel1:TRIGger:OUTEdge RISe	Set the output mode for CH1 rising edge.
:CHANnel1:TRIGger:OUTEdge?	The query returns RISe.

The sweep frequency

:CHANnel<n>:SWEep:TYPe

- **Command Format**

  :CHANnel<n>:SWEep:TYPe { LINe|LOG|STEP}

  :CHANnel<n>:SWEep:TYPe?

- **Functional Description**

  Set the sweep frequency mode for the specified channel, which is linear, logarithm and step.

  <n>: Channel number，n takes the value as 1 and 2.

- **Return Format**

  The query returns the sweep frequency mode of the specified channel.

- **For Example**

:CHANnel1:SWEep:TYPe LINe	Set the sweep frequency mode of CH1 to LINe.
: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?

- **Functional Description**

  Set the start frequency for the sweep frequency in the specified channel.

  < freq > represents the frequency，the unit is Hz.

  <n>: Channel number，n takes the value as 1 and 2.

- **Return Format**

  The query returns the start frequency of the sweep frequency in the specified channelin scientific notation.

- **For Example**

:CHANnel1:SWE:FREQ:STAR 2000	Set the start frequency of CH1 the sweep frequency to 2 KHz.
: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?

- **Functional Description**

  Set the cut-off frequency for the sweep frequency in the specified channel.

  < freq > represents the frequency, the unit is Hz.

  <n>: Channel number, n takes the value as 1 and 2.

- **Return Format**

  The query returns the cut-off frequency of the sweep frequency in the specified channel in scientific notation.

- **For Example**

:CHANnel1:SWE:FREQ:STOP 2000	Set the cut-off frequency of CH1 the sweep frequency to 2 KHz.
:CHANnel1:SWE:FREQ:STOP?	The query returns 2e+3.

:CHANnel<n>:SWEep:TIMe

- **Command Format**

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

  :CHANnel<n>:SWEEP:TIMe?

- **Functional Description**

  Set the sweep time for the sweep frequency in the specified channel.

  <time> represents the sweep time, the unit is S and the range is 1ms ~ 500s.

  <n>: Channel number, n takes the value as 1 and 2.

- **Return Format**

  The query returns the sweep time of the sweep frequency in the specified channel in scientific notation.

- **For Example**

:CHANnel1:SWEEP:TIMe 2	Set the sweep time of CH1 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?

- **Functional Description**

  Set the hold time for the sweep frequency in the specified channel. This command is only valid for the step sweep mode.

  <time> represents the sweep time，the unit is S.

  <n>: Channel number，n takes the value as 1 and 2.

- **Return Format**

  The query returns the hold time of the sweep frequency in the specified channel in scientific notation.

- **For Example**

:CHANnel1:SWEEP:HOLD 2	Set the holt time of CH1 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?

- **Functional Description**

  Set the total step for the sweep frequency in the specified channel. This command is only valid for the step sweep mode.

  < steps >: Step

  <n>: Channel number，n takes the value as 1 and 2.

- **Return Format**

  The query returns the total step of the sweep frequency in the specified channel. It is integer data.

- **For Example**

:CHANnel1:SWEEP:STEPs 10	Set the step of the specified channel to 1
:CHANnel1:SWEEP:STEPs?	The query returns 10.

:CHANnel<n>:SWEep:TRIGger

- **Command Format**

  :CHANnel<n>:SWEep:TRIGger

- **Functional Description**

  Set the sweep output trigger for the specified channel. This parameter is only valid for manual trigger mode.

- **For Example**

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

Burst

:CHANnel<n>:BURSt:TYPe

- **Command Format**

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

  :CHANnel<n>:BURSt:TYPe?

- **Functional Description**

  Set the burst type for the specified channel，which is N cycle, gate and infinite.

  <n>: Channel number，n takes the value as 1 and 2.

- **Return Format**

  The query returns the burst type of the specified channel

- **For Example**

:CHANnel1:BURSt:TYPe NCYC	Set the burst type of CH1 to NCYC.
:CHANnel1:BURSt:TYPe?	The query returns 2e+0.

:CHANnel<n>:BURSt:PERiod

- **Command Format**

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

  :CHANnel<n>:BURSt:PERiod?

- **Functional Description**

  Set the period for the burst of the specified channel.

  < period > represents time，the unit is S.

  <n>: Channel number，n takes the value as 1 and 2.

- **Return Format**

  The query returns the period of the specified channel in scientific notation.

- **For Example**

:CHANnel1:BURSt:PERiod 0.005	Set the period of CH1 burst to 5 ms.
:CHANnel1:BURSt:PERiod?	The query returns 5e-3.

:CHANnel<n>:BURSt:PHASe

- **Command Format**

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

  :CHANnel<n>:BURSt:PHASe?

- **Functional Description**

  Set the start phase for the burst of the specified channel.

  < phase > represents the phase, the unit is ° and the range is 0 ~ 360.

  <n>: Channel number, n takes the value as 1 and 2.

- **Return Format**

  The query returns the start phase of the burst in the specified channel in scientific notation.

- **For Example**

  :CHANnel1:BURSt:PHASe 18 Set the start phase of CH1 burst 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?

- **Functional Description**

  Set the cycle numbers for the burst of the specified channel.

  < cycles > represents the cycle numbers, it is integer data.

  <n>: Channel number, n takes the value as 1 and 2.

- **Return Format**

  The query returns the cycle numbers of the burst in the specified channel.

- **For Example**

  :CHANnel1:BURSt:CYCLes 2 Set the cycle numbers of the burst in the 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?

- **Functional Description**

  Set the polarity for the burst of the specified channel, which is positive and negative.

  <n>: Channel number, n takes the value as 1 and 2.

- **Return Format**

  The query returns the polarity of the burst in the specified channel.

- **For Example**

:CHANnel1:BURSt:GATe:POLarity POSitive Set the polarity for CH1 burst to positive.

:CHANnel1:BURSt:GATe:POLarity? The query returns POSitive.

:CHANnel<n>:BURSt:TRIGger

- **Command Format**

  :CHANnel<n>:BURSt:TRIGger

- **Functional Description**

  Set the burst output trigger, this parameter is only valid for the manual trigger mode.

- **For Example**

  :CHANnel1:BURSt:TRIGger Trigger burst signal to output once.

WARB Command

This command is used to write arbitrary wave file, including basic arbitrary wave and arbitrary modulation wave.

:WARB<n>:MODulate

- **Command Format**

  :WARB<n>:MODulate <arb file>

- **Functional Description**

  Write the arbitrary modulation wave. Send this command at first and then send the arbitrary wave file to the signal source.

  <arb file> represents the name of arbitrary wave file，it only supports the file format ".bsv".

- **For Example**

  :WARB1:MODulate "test.bsv" Write the arbitrary modulation wave file for CH1.

:WARB<n>:CARRier

- **Command Format**

  :WARB<n>:CARRier <arb file>

- **Functional Description**

  Write the basic arbitrary wave. Send this command at first and then send the arbitrary wave file to the signal source.

  <arb file> represents the name of arbitrary wave file，it only supports the file format ".bsv".

- **For Example**

  :WARB1:CARRier "test.bsv" Write the basic arbitrary wave file for CH1.

DISPlay Command

This command is used to display the relevant information of the signal source.

:DISPlay:DATA?

- **Command Format**

  :DISPlay:DATA?

- **Functional Description**

  Query the picture data of the current screen. The query returns the picture data in BMP by default. The format of picture data is determined by the command :SYSTem:PICTure:FORMat.

- **Return Format**

  The query returns the picture data, the return data is conform to IEEE 488.2 # binary system.

- **For Example**

:DISPlay:DATA?	The query returns the picture data. Data format: #800012345+picture data

Explanation of Programming

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

Programming Preparation

The programming preparation is only applicable for using Visual Studio and Lab VIEW development tools to programming under Windows operating system.

Firstly, the user need to confirm that whether NI –VISA library is installed (it can be download from the website

https://www.ni.com/en-ca/support/downloads/drivers/download.ni-visa.html).

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

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

VISA Programming Example

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

VC++Example

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

1. Open the visual studio software to create a new VC++ win32 console project.
2. Set the project environment that can adjust NI-VISA library, which is static library and dynamic library.

a) Static library

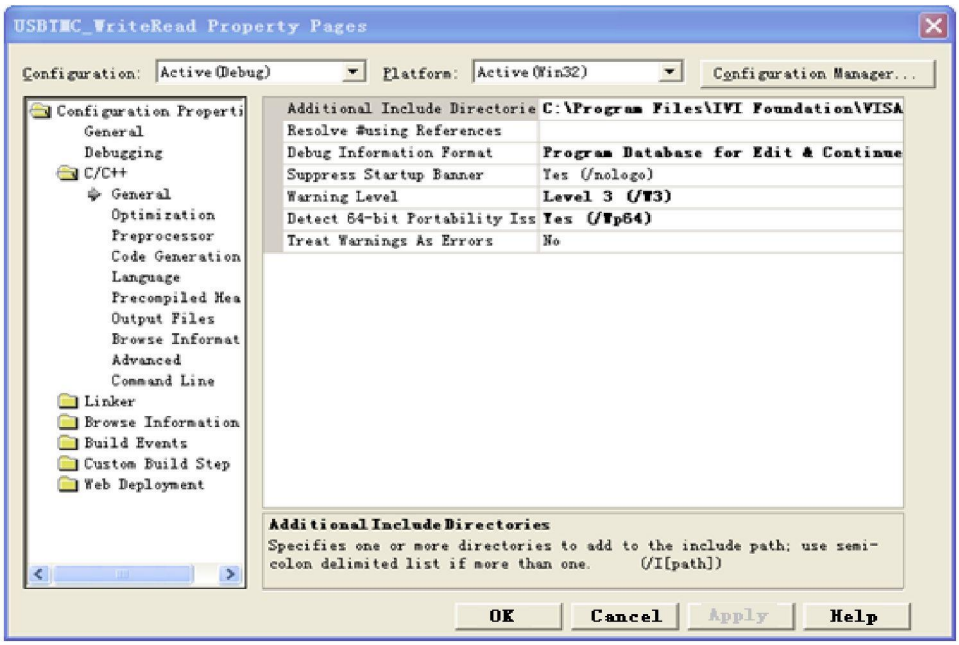
Find file visa.h, visatype.h and visa32.lib in NI-VISA installment path and copy them to the root path of VC++ project and add it to the project. Add two lines of code into file projectname.cpp as follows.

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

b) Dynamic library

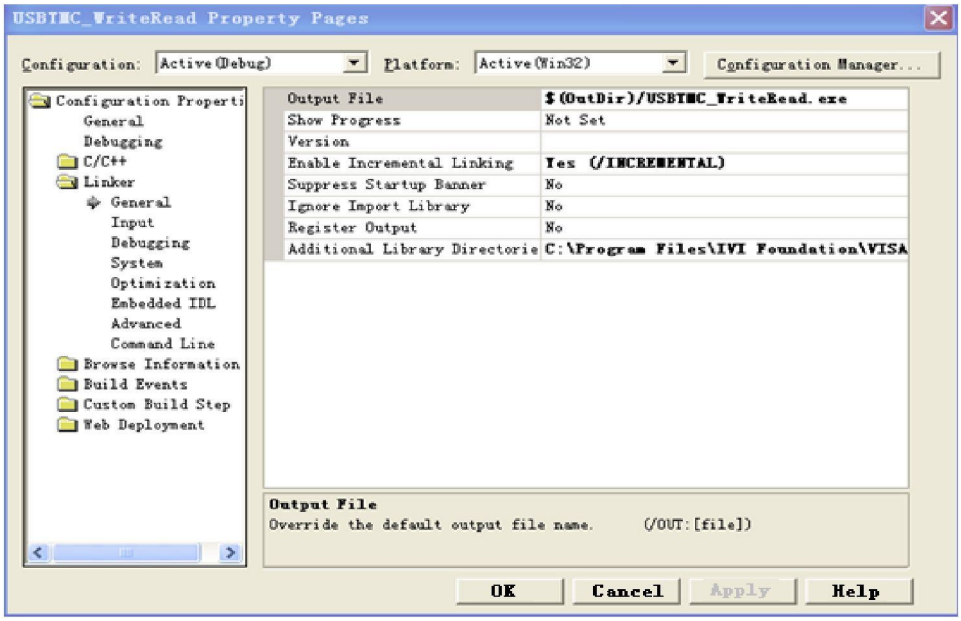
Press “project>>properties”, select"c/c++---General" in attribute dialog on the leftside, set the value of "Additional Include Directories" as the installment path of NI-VIS

(For example, C:\ProgramFiles\IVI Foundation\VISA\WinNT\include), as shown in the following figure.

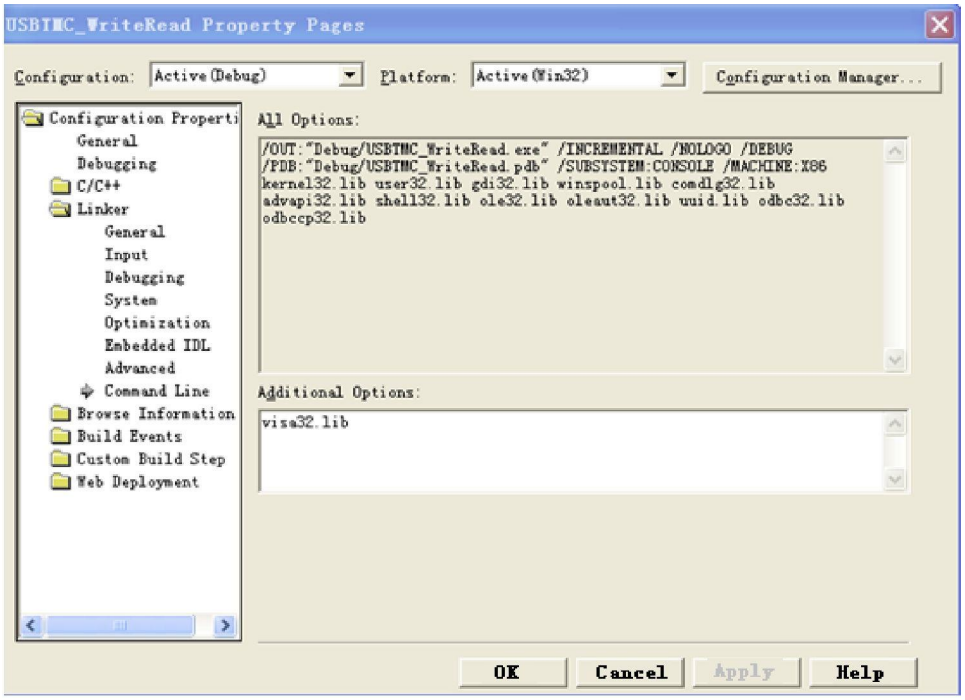


Select "Linker-General" in attribute dialog on the left side, set the value of "Additional Library Directories" as the installment path of NI-VIS

(For example, C:\Program Files\IVI Foundation\VISA\WinNT\include), as shown in the following figure.



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



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

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

1. Source code

a) USBTMC Example

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

```
        return status;
}
/*Find all the USB TMC VISA resources in our system and store the number of resources in the system in
numInstrs.*/
status = viFindRsrc(defaultRM, "USB?*INSTR", &findList, &numInstrs, instrResourceString);
if (status<VI_SUCCESS)
{
        printf("An error occurred while finding resources. \nPress Enter to continue.");
        fflush(stdin);
        getchar();
        viClose(defaultRM);
        return status;
}
/** Now we will open VISA sessions to all USB TMC instruments.
*       We must use the handle from viOpenDefaultRM and we must
*       also use a string that indicates which instrument to open. This
*       is called the instrument descriptor. The format for this string
*       can be found in the function panel by right clicking on the
*       descriptor parameter. After opening a session to the
*       device, we will get a handle to the instrument which we
*       will use in later VISA functions. The AccessMode and Timeout
*       parameters in this function are reserved for future
*       functionality. These two parameters are given the value VI_NULL. */
for (i = 0; i < int(numInstrs); i++)
{
        if (i > 0)
        {
                viFindNext(findList, instrResourceString);
        }
        status = viOpen(defaultRM, instrResourceString, VI_NULL, VI_NULL, &instr);
        if (status < VI_SUCCESS)
        {
                printf("Cannot open a session to the device %d. \n", i + 1);
                continue;
        }
        /** At this point we now have a session open to the USB TMC instrument.
        *We will now use the viPrintf function to send the device the string "*IDN?\n",
        *asking for the device's identification. */
        char * cmmand = "*IDN?\n";
        status = viPrintf(instr, cmmand);
        if (status < VI_SUCCESS)
        {
                printf("Error writing to the device %d. \n", i + 1);
                status = viClose(instr);
                continue;
        }
```

```
        /** Now we will attempt to read back a response from the device to
        *the identification query that was sent. We will use the viScanf
        *function to acquire the data.
        *After the data has been read the response is displayed. */
        status = viScanf(instr, "%t", buffer);
        if (status < VI_SUCCESS)
        {
            printf("Error reading a response from the device %d. \n", i + 1);
        }
        else
        {
            printf("\nDevice %d: %s\n", i + 1, buffer);
        }
        status = viClose(instr);
    }
    /*Now we will close the session to the instrument using viClose. This operation frees all
    system resources.*/
    status = viClose(defaultRM);
    printf("Press Enter to exit.");
    fflush(stdin);
    getchar();
    return 0;
}

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

b) TCP/IP Example

```
int tcp_ip_test(char *pIP)
{
    char outputBuffer[VI_FIND_BUFLEN];
    ViSession defaultRM, instr;
    ViStatus status;
    /* First we will need to open the default resource manager. */
    status = viOpenDefaultRM(&defaultRM);
    if (status < VI_SUCCESS)
    {
        printf("Could not open a session to the VISA Resource Manager!\n");
    }
    /* Now we will open a session via TCP/IP device */
    char head[256] = "TCPIP0::";
    char tail[] = "::inst0::INSTR";
    strcat(head, pIP);
```

```
    strcat(head, tail);
    status = viOpen(defaultRM, head, VI_LOAD_CONFIG, VI_NULL, &instr);
    if (status < VI_SUCCESS)
    {
        printf("An error occurred opening the session\n");
        viClose(defaultRM);
    }
    status = viPrintf(instr, "*idn?\n");
    status = viScanf(instr, "%t", outputBuffer);
    if (status < VI_SUCCESS)
    {
        printf("viRead failed with error code: %x \n", status);
        viClose(defaultRM);
    }
    else
    {
        printf("\nMesseage read from device: %*s\n", 0, outputBuffer);
    }
    status = viClose(instr);
    status = viClose(defaultRM);
    printf("Press Enter to exit.");
    fflush(stdin);
    getchar();
    return 0;
}

int _tmain(int argc, _TCHAR* argv[])
{
    printf("Please input IP address:");
    char ip[256];
    fflush(stdin);
    gets(ip);
    tcp_ip_test(ip);
    return 0;
}
```

C#Example

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

1. Open the visual studio software to create a new C# console project.
2. 2. Add C# quote Ivi.Visa.dll and NationalInstruments.Visa.dll of VISA.

3. Source code

a) USBTMC Example

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

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

b) TCP/IP Example

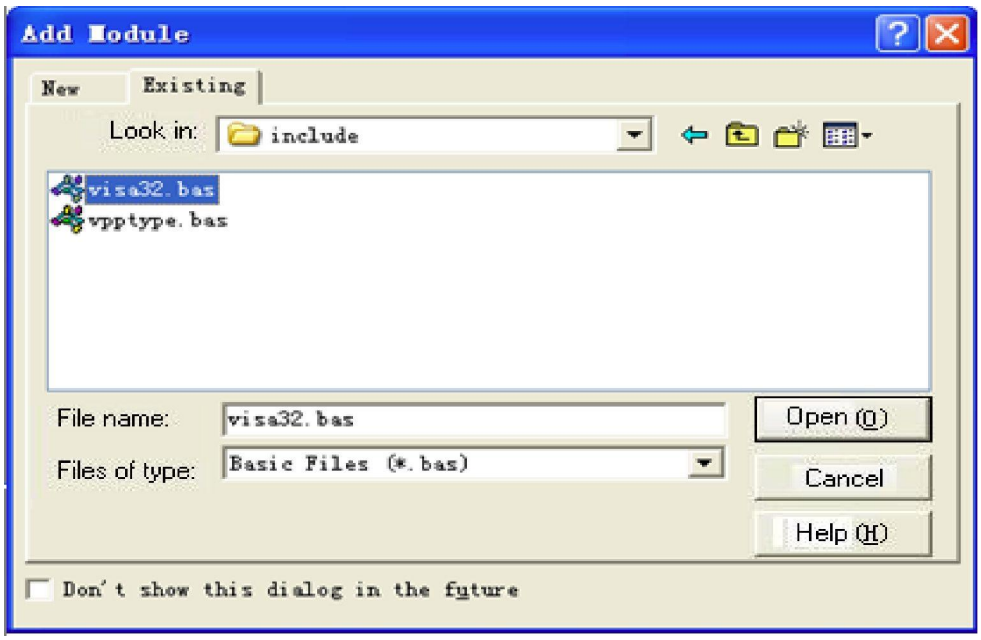
```
class Program
 {
    void tcp_ip_test(string ip)
    {
        using (var rmSession = new ResourceManager())
        {
            try
            {
                var resource = string.Format("TCPIP0::{0}::inst0::INSTR", ip);
                var mbSession = (MessageBasedSession)rmSession.Open(resource);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

```
                System.Console.WriteLine(mbSession.RawIO.ReadString());
            }
            catch (Exception ex)
            {
                System.Console.WriteLine(ex.Message);
            }
        }
    }

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

VB Example

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

1. Open the visual basic software and create a new standard application program project.
2. Set the project environment that can adjust NI-VISA library, press Existing tab of Project>>Add Existing Item, find the visa32.bas file in the "include" folder under the NI-VISA installation path



and add it, as shown in the following figure.

3. Source code

a) USBTMC Example

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

Const MAX_CNT = 200
Dim defaultRM AsLong
Dim instrsesn AsLong
Dim numlnstrs 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 numlnstrs.
status = viFindRsrc(defaultRM, "USB?*INSTR", findList, numlnstrs, instrResourceString)
If (status < VI_SUCCESS) Then
    resultTxt.Text = "An error occurred while finding resources."
    viClose(defaultRM)
    usbtmc_test = status
ExitFunction
EndIf

' Now we will open VISA sessions to all USB TMC instruments.
' We must use the handle from viOpenDefaultRM and we must
```

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

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

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

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

b) TCP/IP Example

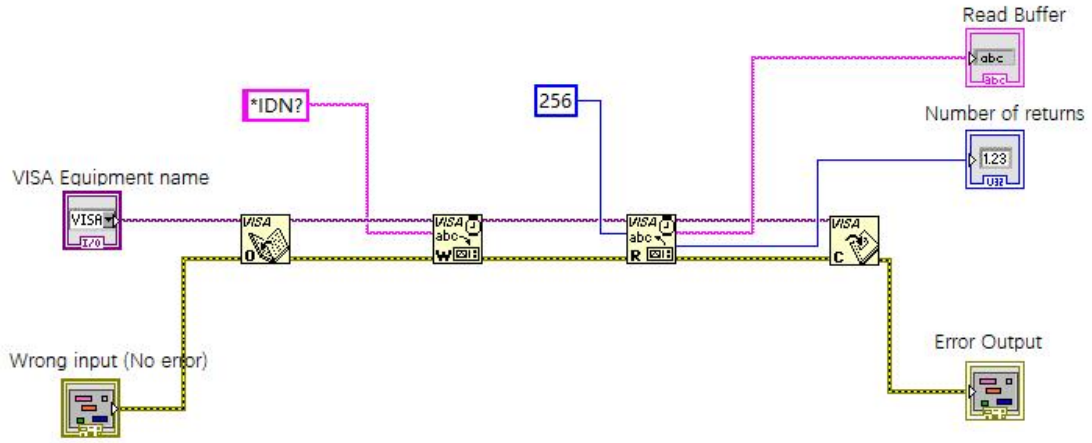
```
PrivateFunction tcp_ip_test(ByVal ip AsString) AsLong
Dim outputBuffer AsString * VI_FIND_BUFLEN
Dim defaultRM AsLong
Dim instrsesn AsLong
Dim status AsLong
Dim count AsLong

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

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

LabVIEW Example

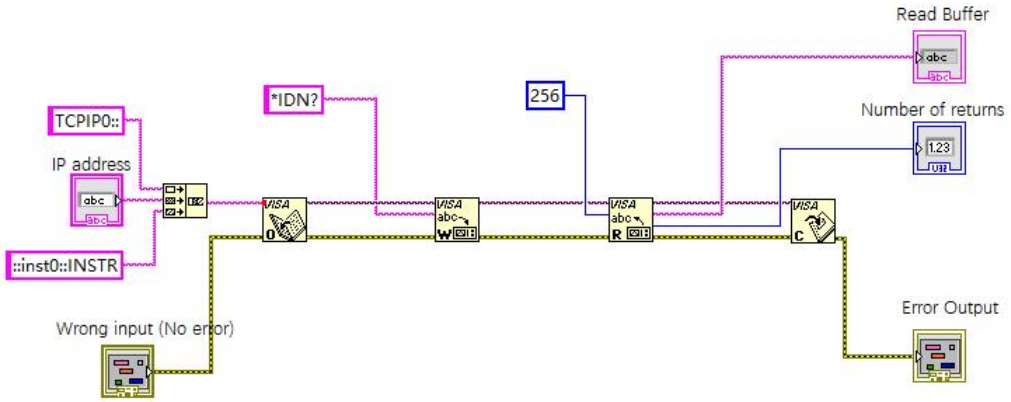
- Environment: Window system, LabVIEW.
- Description: Access the instrument via USBTMC and TCP/IP, and send "*IDN?" command on NI-VISA to query the device information.

➢ Steps

1. Open the Lab VIEW software to create a VI file.
2. Add the control, click the front panel to select and add the VISA source name, error input, error output, and part of the indicator from the control column.
3. Open the diagram, click the VISA source name, and then select and add these functions VISA Write, VISA Read, VISA Open and VISA Close on the VISA menu.
4. The VI opens a VISA session for a USBTMC device and writes the *IDN? command to the device and reads back the response value. When all communication is complete, the VI closes the VISA session as shown in the following figure.



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



MATLAB Example

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

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

a) USBTMC Example

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

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

%Open the VISA object created
fopen(vu);

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

%Request the data

outputbuffer = fscanf(vu);
```

```
disp(outputbuffer);

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

b) TCP/IP Example

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

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

%Open the VISA object created

fopen(vt);

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

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

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

end
```

Python Example

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

1. Install the Python at first, then turn on the Python to create an empty file test.py.
2. Use the command pip install PyVISA to install PyVISA. If it cannot be installed, please refer to this link (https://pyvisa.readthedocs.io/en/latest/).

3. Source code

a) USBTMC Example

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

b) TCP/TP Example

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

Programming Application Example

Configure Sine Wave

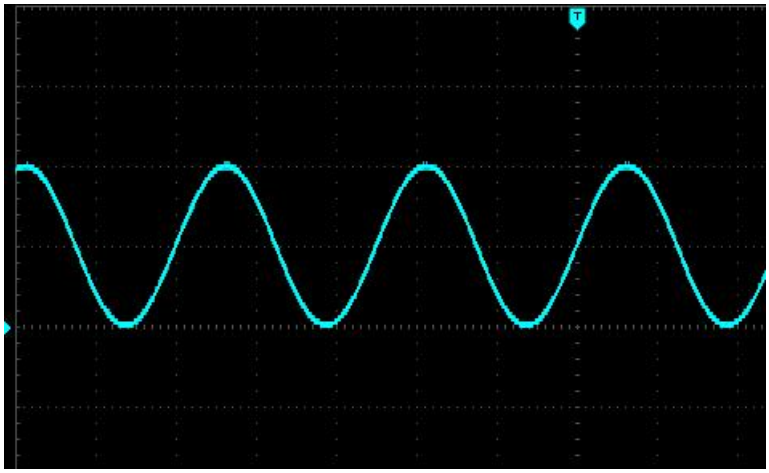
This section is to introduce how to configure the sine wave function.

Explanation

A sine wave has an amplitude, an offset, and a phase relative to a synchronous pulse. It can use high voltage value and low voltage value to set its amplitude and deviation.

Example

The following wave can set by SCPI, high level and low level can replace the command :CHANnel1:BASE:AMPLitude and :CHANnel1:BASE:OFFSet



The following command can generate the sine wave as shown above.

: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

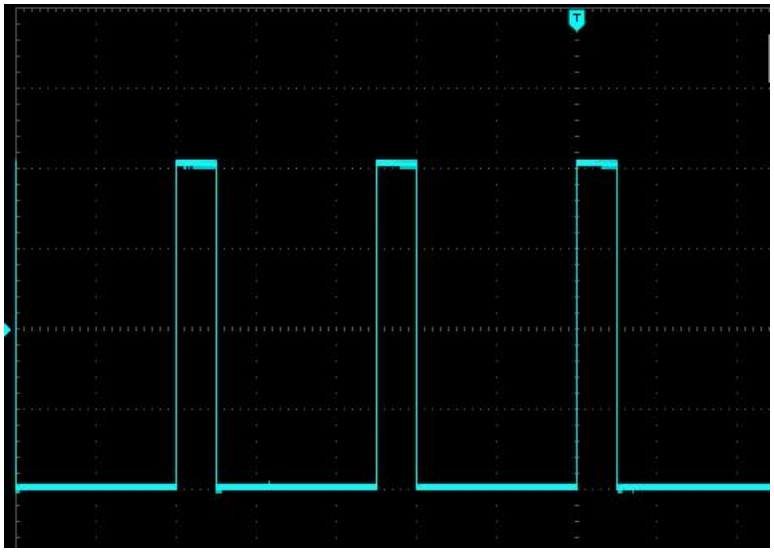
Configuring Square Wave

Explanation

A square wave has an amplitude, an offset, and a phase relative to a synchronous pulse. In addition, the square wave has duty ratio and period. It can use high voltage and low voltage to set its amplitude and deviation.

Example

The following wave can set by SCPI.



The following command can generate the square wave as shown above.

: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

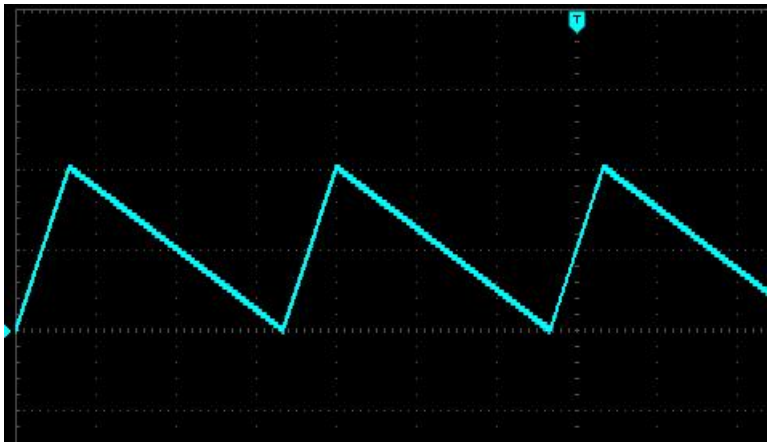
Configuring Sawtooth Wave

Explanation

A sawtooth wave has an amplitude, an offset, and a phase relative to a synchronous pulse. It also can create triangle wave and the symmetry of other similar waves. It can use high voltage and low voltage to set its amplitude and deviation.

Example

The following wave can set by SCPI, high level and low level can replace the command :CHANnel1:BASE:AMPLitude and :CHANnel1:BASE:OFFSet



The following command can generate the sawtooth wave as shown above.

: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

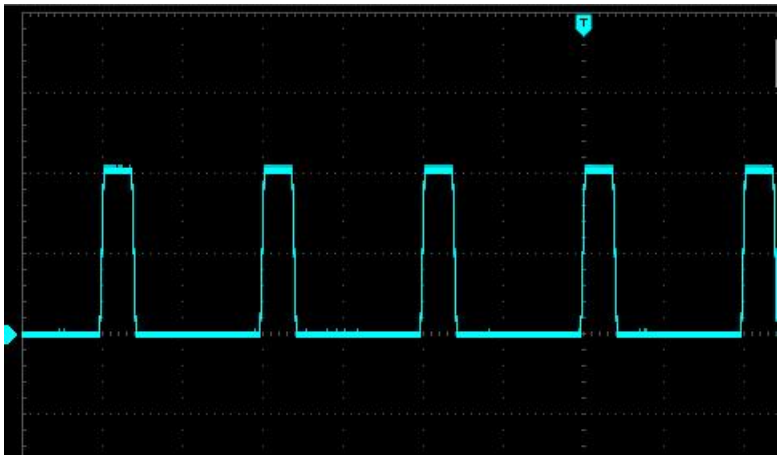
Configuring Pulse Wave

Explanation

A pulse wave has an amplitude, an offset, and a phase relative to a synchronous pulse. It can also add the edge slope and duty ratio (or pulse width). It can use high voltage and low voltage to set its amplitude and deviation.

Example

The following wave can set by SCPI, high level and low level can replace the command :CHANnel1:BASE:AMPLitude and :CHANnel1:BASE:OFFSet



The following command can generate the pulse wave as shown above.

: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

Configuring Arbitrary Wave

This section is to introduce how to configure an arbitrary wave.

Explanation

A harmonic wave has a frequency, amplitude, offset and phase. It can also add mode and wave file.

Example

The following command can load and modify built-in arbitrary wave.

:CHANnel1:MODe CONTinue

:CHANnel1:BASE:WAVe ARB

:CHANnel1:ARB:MODe DDS

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

:CHANnel1:BASE:FREQuency 200000

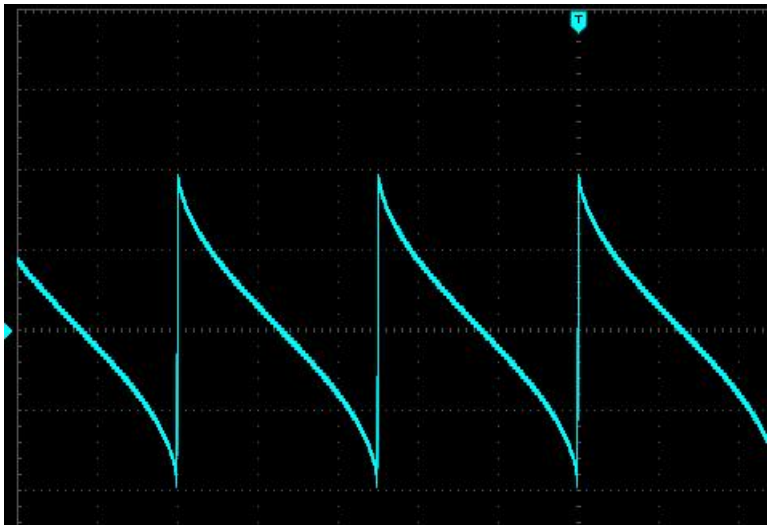
:CHANnel1:BASE:AMPLitude 2

:CHANnel1:BASE:OFFSet 0

:CHANnel1:BASE:PHAse 90

:CHANnel1:OUTPut ON

The wave generate by the above command as shown in the following figure.



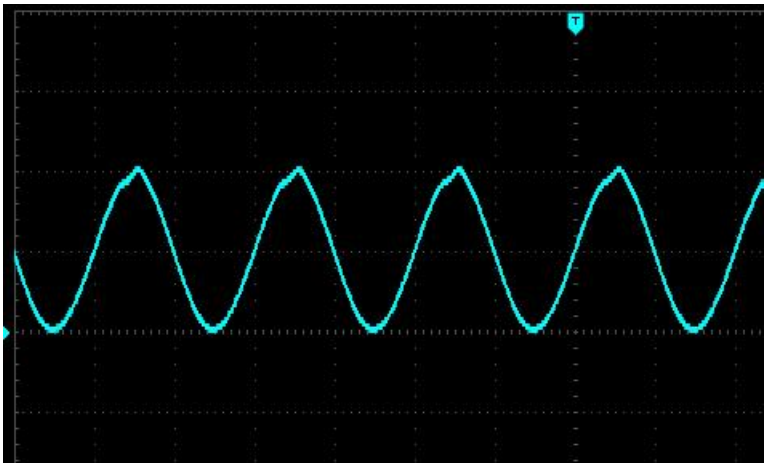
Configure Harmonic Wave

Explanation

A harmonic wave has an amplitude, offset and phase. It can also add total harmonic order, harmonic amplitude and harmonic phase. It can use high voltage and low voltage to set its amplitude and deviation.

Example

The following wave can set by SCPI, high level and low level can replace the command :CHANnel1:BASE:AMPLitude and :CHANnel1:BASE:OFFSet



The following command can generate the harmonic wave as shown above.

: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

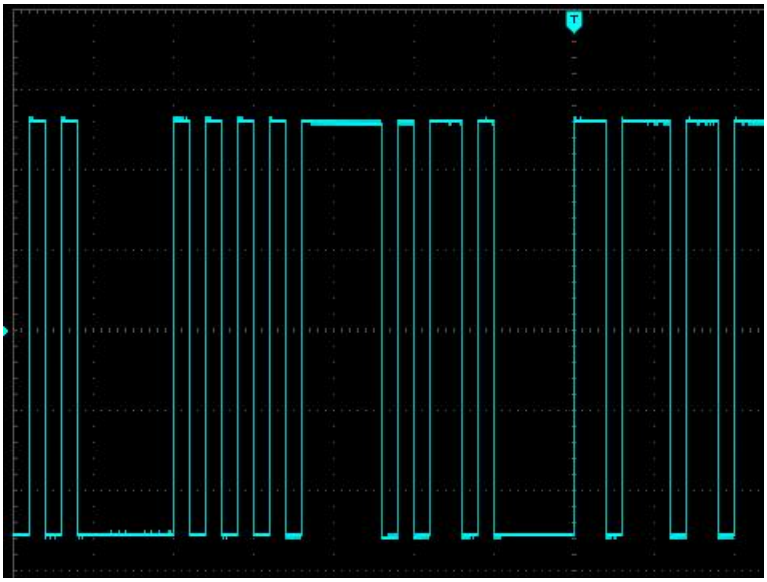
Configure Pseudo-random Wave

Explanation

A pseudo-random wave has a bitrate, phase, edge and symbol. It can use high voltage and low voltage to set its amplitude and deviation.

Example

The following wave can set by SCPI, high level and low level can replace the command:CHANnel1:BASE:AMPLitude and :CHANnel1:BASE:OFFSet



The following command can generate the harmonic wave as shown above.

: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	Functional Description	LED
F1	Select the first menu option at the current menu	
F2	Select the second menu option at the current menu	
F3	Select the third menu option at the current menu	
F4	Select the fourth menu option at the current menu	
F5	Select the fifth menu option at the current menu	
F6	Select the sixth menu option at the current menu	
MODE	Output mode	√
WAVE	Wave type	
UTILity	System	
TRIGger	Trigger	√
CH1	Channel 1	√
CH2	Channel 2	√
LEFT	Arrow key LEFT	
RIGHT	Arrow key RIGHT	
UP	Arrow key UP	
DOWN	Arrow key DOWN	
OK	Enter key	
NUM0	Digit key 0	
NUM1	Digit key 1	
NUM2	Digit key 2	
NUM3	Digit key 3	
NUM4	Digit key 4	
NUM5	Digit key 5	
NUM6	Digit key 6	
NUM7	Digit key 7	
NUM8	Digit key 8	
NUM9	Digit key 9	
DOT	Digit key decimal point	
SYMBOL	Digit key symbol	