UNI-T®

Instruments.uni-trend.com

Programming manual

UTG2000B Series programmable signal source

Warranties and representations

copyright

Trademark information

UNI-Tis a registered trademark of Uni-Lead China Technology Co., Ltd.

Document number

20230619

Software version

V2.01.0010
Software upgrades may change or add product features, please pay attentionUNI-TWebsite to obtain the latest version of the manual or contactUNI-TUpgrade software.

statement

- The company's products are protected by patents (including patents obtained and pending applications) in China and other countries and regions.
- The company reserves the right to change specifications and prices.
- The information provided in this manual supersedes all previously published material.
- The information provided in this manual is subject to change without prior notice.
- We are not responsible for any errors that may be contained in this manual, or for any incidental or consequential losses caused by the information provided and the functions performed in this manual and the use of this manual.UNI-TNo responsibility is assumed.
- withoutUNI-TNo part of this manual may be photocopied, reproduced or adapted without prior written permission.

certified product

UNI-TCertify that this product complies with China's national product standards and industry product standards, ISO9001:2008 standards and ISO14001:2004 standards, and further certifies that this product complies with relevant standards of members of other international standards organizations.

contact us

If you have any questions or needs while using this product or this manual, you may contactUNI-Tconnect:
E-mail:
URL:

Introduction to SCPI commands

SCPI (Standard Commands for Programmable Instruments, Standard Command Set for Programmable Instruments) is based on the existing standards IEEE 488.1 and IEEE 488.2, and follows the floating point arithmetic rules in the IEEE754 standard and the ISO646 information exchange 7-bit encoding symbol (equivalent to ASCII programming) and other standardized instrument programming languages. This section introduces the format, symbols, parameters and abbreviation rules of SCPI commands.

Command format

The SCPI command is a tree-like hierarchical structure, including multiple subsystems. Each subsystem consists of a root keyword and one or several hierarchical keywords.**The command line usually starts with a colon ":"; the keywords are separated by a colon ":", and the keywords are followed by optional parameter settings. The command keyword and the first parameter are separated by a space. The command string must end with a <newline> (<NL>) character。 Adding a question mark "?" after the command line usually indicates querying this function.**

Symbol Description

The following four symbols are not part of the SCPI command and are not sent with the command, but are usually used to assist in explaining the parameters in the command.

- **big parantheses { }**
  Curly braces usually contain multiple optional parameters, one of which must be selected when sending the command. Such as: DISPlay:GRID:MODE { FULL | GRID | CROSS | NONE} command.

- **vertical line |**
  Vertical bars are used to separate multiple parameter options, one of which must be selected when sending a command.
  Such as: DISPlay:GRID:MODE { FULL | GRID | CROSS | NONE} command.

- **Square brackets [ ]**
  The content in square brackets (command keywords) can be omitted. If a parameter is omitted, the instrument sets the parameter to its default value. For example: For the :MEASure:NDUTy? [<source>] command, [<source>] represents the current channel.

- **triangle brackets < >**
  Parameters enclosed in triangle brackets must be replaced with a valid value. For example:
  DISPlay:GRID:BRIGhtness Send DISPlay:GRID:BRIGhtness in the form 30 <count> command.

Parameter Description

The parameters contained in the commands introduced in this manual can be divided into the following five types: Boolean, integer, real, discrete, and ASCII string.

- boolean
  The parameter value is "ON" (1) or "OFF" (0). For example: :SYSTem:LOCK {{1 | ON} | {0 | OFF}}.

- integer
  Unless otherwise noted, parameters can take on any integer value within the valid range. Note: At this time, please do not set the parameters to decimal format, otherwise an exception will occur. For example: The parameter <count> in the :DISPlay:GRID:BRIGhtness <count> command can be any integer in the range of 0 to 100.

- real type
  Unless otherwise stated, parameters can take on any value within the valid range.
  For example: For CH1, the value of parameter <offset> in the CHANnel1:OFFSet <offset> command is a real type.

- Discrete
  Parameters can only take several specified values or characters. For example: :DISPlay:GRID:MODE { FULL | GRID | CROSS | NONE} The parameters of the command can only be FULL, GRID, CROSS, and NONE.

- ASCII string
  String parameters can contain virtually all ASCII character sets. The string must start and end with matching quotes; you can use single or double quotes. The quote delimiter can also be part of a string, just type it twice without adding any characters in between, for example set IP: SYST:COMM:LAN:IPAD "192.168.1.10".

Abbreviation rules

All commands are case-aware and can be all uppercase or lowercase. If you want to abbreviate, you must enter all capital letters in the command format.

Data return

Data return is divided into single data and batch data return. Single data returns the corresponding parameter type, where real type returns are expressed in scientific notation., the first part of e retains three digits of data after the decimal point, and the part of e retains three digits of data.;Batch data returns string data that must comply with IEEE 488.2 # format, its format: '#'+ The number of characters occupied by the length [fixed to one character] + ASCII value of valid data length + valid data + terminator ['\n'], such as #3123xxxxxxxxxxxxxxxxxxx\n represents a valid batch data return format of 123 bytes, where '3' means "123" occupies 3 characters.

Detailed explanation of SCPI commands

IEEE488.2 common commands

*IDN?

- **Command format:**
  *IDN?
- **Function description:**
  Used to query the manufacturer name, product model, product serial number and software version number.
- **Return format:**
  Manufacturer name, Product number, Product Serial Number, Software version numbers separated by dots.
  Notice: The returned model number must be consistent with the nameplate information.
- **Example:**
  UNI-T Technologies, UTG2000, 000000001, 00.00.01

*RST

- **Command format:**
  *RST
- **Function description:**
  Used to restore factory settings and clear all error messages and send and receive queue buffers.

SYSTem command

Used to perform the most basic operations on the signal source, mainly including full keyboard lock and system setting data operations.

:SYSTem:LOCK

- **Command format:**
  :SYSTem:LOCK {{1|ON}|{0|OFF}}
  :SYSTem:LOCK?
- **Function description:**
  Used to lock or unlock the full keyboard keys and touch input.
- **Return format:**
  The query returns the full keyboard lock status, 0 means unlocked, 1 means locked.
- **Example:**

:SYSTem:LOCK ON	Full keyboard lock
:SYSTem:LOCK OFF	Full keyboard unlock
:SYSTem:LOCK?	The query returns 1, indicating locking

:SYSTem:CONFigure

- **Command format:**
  :SYSTem:CONFigure <file>
  :SYSTem:CONFigure?
- **Function description:**
  Used to read and write configuration files, first send this command, 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.
- **Example:**

:SYSTem:CONFigure	Write configuration file data to the signal source and let it load

:SYSTem:CONFigure?	The query returns the current configuration file data binary stream of the signal source.

:SYSTem:LANGuage

- **Command format:**
  :SYSTem:LANGuage {ENGLish|SIMChinese|TRAChinese}
  :SYSTem:LANGuage?
- **Function description:**
  Controls the system display language.
- **Return format:**
  The query returns the system display language.
- **Example:**

:SYSTem:LANGuage ENGLish	Set English as the system display language
:SYSTem:LANGuage?	Query returns ENGLish

:SYSTem:CLKSource

- **Command format:**
  :SYSTem:CLKSource {INTernal|EXTernal }
  :SYSTem:CLKSource?
- **Function description:**
  Controls selection of system clock source
- **Return format:**
  The query returns the system clock source.
- **Example:**

:SYSTem:CLKSource INTernal	Set the system clock source to the internal clock source
:SYSTem:CLKSource?	The query returns INTernal

:SYSTem:CLKOut

- **Command format:**
  :SYSTem:CLKOut { {1 | ON} | {0 | OFF} }
  :SYSTem:CLKOut?
- **Function description:**
  Controls the selection of system clock output on and off
- **Return format:**
  The query returns the status of the system clock output.
- **Example:**

:SYSTem:CLKOut ON	Set the system clock output to open
:SYSTem:CLKOut?	The query returns 1

:SYSTem:BEEP

- **Command format:**
  :SYSTem:BEEP {{1 | ON} | {0 | OFF}}
  :SYSTem:BEEP?
- **Function description:**
  Control system buzzer switch
- **Return format:**
  The query returns the status of the buzzer switch.
- **Example:**

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

:SYSTem:NUMBer:FORMat

- **Command format:**

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

- Function description:
  Delimiter that controls system number format
- Return format:
  The query returns the delimiter for the system number format.
- Example:

:SYSTem:NUMBer:FORMat NONe	Set systemless number format
:SYSTem:NUMBer:FORMat?	The query returns NONe

:SYSTem:BRIGhtness

- Command format:
  :SYSTem:BRIGhtness { 10|30|50|70|90|100}
  :SYSTem:BRIGhtness?
- Function description:
  Control system backlight brightness level
- Return format:
  Query returns the system backlight brightness level
- Example:

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

:SYSTem:CYMometer

- Command format:
  :SYSTem:CYMometer {{1 | ON} | {0 | OFF}}
  :SYSTem:CYMometer?
- Function description:
  Control system frequency meter start and stop status.
- Return format:
  The query returns the start and stop status of the system frequency meter. 0 means stop and 1 means start.
- Example:

:SYSTem:CYMometer ON	Start system frequency meter

:SYSTem:CYMometer:FREQuency?

- Command format:
  :SYSTem:CYMometer:FREQuency?
- Function description:
  Gets the frequency currently measured by the frequency meter.
- Return format:
  The query returns the currently measured frequency of the frequency meter, in Hz, and uses scientific notation to return the data.
- Example:

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

:SYSTem:CYMometer:PERiod?

- Command format:
  :SYSTem:CYMometer:PERiod?
- Function description:
  Gets the currently measured period of the frequency meter.
- Return format:
  The query returns the current measured period of the frequency meter, in S, and uses scientific notation to return the data.

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

:SYSTem:CYMometer:DUTY?

- Command format:
  :SYSTem:CYMometer:DUTY?
- Function description:
  Gets the currently measured duty cycle of the frequency meter.
- Return format:
  The query returns the currently measured duty cycle of the frequency meter, returning the data in scientific notation.
- Example:
  :SYSTem:CYMometer:DUTY? The query returns 2e+1, which means the duty cycle is 20%.

:SYSTem:CYMometer:PWIDTh?

- Command format:
  :SYSTem:CYMometer:PWIDTh?
- Function description:
  Gets the currently measured positive pulse width of the frequency meter.
- Return format:
  The query returns the current measured positive pulse width of the frequency meter, in s.
- Example:
  :SYSTem:CYMometer:PWIDTh? The query returns 1e-3, which means the duty cycle is 1 millisecond.

:SYSTem:CYMometer:NWIDTh?

- Command format:
  :SYSTem:CYMometer:NWIDTh?
- Function description:
  Gets the currently measured negative pulse width of the frequency meter.
- Return format:
  The query returns the currently measured negative pulse width of the frequency meter, in s.
- Example:
  :SYSTem:CYMometer:NWIDTh? The query returns 1e-3, which means the duty cycle is 1 millisecond.

CHANnel command

Used to set signal source channel related functions.

:CHANnel<n>:MODE

- Command format:
  :CHANnel<n>:MODE {CONTinue | MODulation| SWEep| BURSt }
  :CHANnel<n>:MODE?
- Function description:
  Set the specified channel signal mode, which are CONTinue, MODulation, SWEep, and BURSt.
  <n>: Channel number, n takes the value 1 or 2.
- Return format:
  The query returns the signal mode of the specified channel.
- Example:
  :CHANnel1:MODE MODulation Set channel 1 signal modulation output

:CHANnel1:MODE? Query returns MODulation

:CHANnel<n>:OUTPut

- **Command format:**
  :CHANnel<n>:OUTPut {{1 | ON} | {0 | OFF}}
  :CHANnel<n>:OUTPut?
- **Function description:**
  Set to turn on or off the output of the specified channel.
  <n>: Channel number, n takes the value 1 or 2.
- **Return format:**
  The query returns the output status of the specified channel, 0 means closed, 1 means open.
- **Example:**
  :CHANnel1:OUTPut ON — Set to turn on channel 1 output
  :CHANnel1:OUTPut? — The query returns 1

:CHANnel<n>:INVersion

- **Command format:**
  :CHANnel<n>:INVersion {{1 | ON} | {0 | OFF}}
  :CHANnel<n>:INVersion?
- **Function description:**
  Sets the reverse direction of the specified channel to be turned on or off.
  <n>: Channel number, n takes the value 1 or 2.
- **Return format:**
  The query returns the reverse status of the specified channel, 0 means closed, 1 means open.
- **Example:**
  :CHANnel1:INVersion ON — Set to turn on channel 1 reverse output
  :CHANnel1:INVersion? — The query returns 1

:CHANnel<n>:OUTPut:SYNC

- **Command format:**
  :CHANnel<n>:OUTPut:SYNC {{1 | ON} | {0 | OFF}}
  :CHANnel<n>:OUTPut:SYNC?
- **Function description:**
  Set the channel synchronization output status.
  Note: The device has only one synchronization output interface, and only one channel's synchronization output can be turned on at the same time.
  <n>: Channel number, n takes the value 1 or 2.
- **Return format:**
  The query returns the synchronization output status of the specified channel. 0 means closed, 1 means channel 1 is open, and 2 means channel 2 is open.
- **Example:**
  :CHANnel1:OUTPut:SYNC 1 — Set to turn on channel 1 sync output
  :CHANnel1:OUTPut:SYNC? — The query returns 1

:CHANnel<n>:AMPLitude:UNIT

- **Command format:**
  :CHANnel<n>:AMPLitude:UNIT {VPP | VRMS | DBM}
  :CHANnel<n>:AMPLitude:UNIT?
- **Function description:**
  Set the output amplitude unit of the specified channel.
  <n>: Channel number, n takes the value 1 or 2.
- **Return format:**

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

- Example:

:CHANnel1:AMPLitude:UNIT VPP	Set the output amplitude unit of channel 1 to VPP
:CHANnel1:AMPLitude:UNIT?	Query returns VPP

:CHANnel<n>:LOAD

- Command format:
  :CHANnel<n>:LOAD <resistance>
  :CHANnel<n>:LOAD?
- Function description:
  Set the specified channel output load.
  <resistance> represents the load resistance value in Ω
  <n>: Channel number, n takes the value 1 or 2.
  Note: The resistance value range is 1~ 1000000, where 1000000 corresponds to high resistance.
- Return format:
  The query returns the load resistance value of the specified channel, using scientific notation.
- Example:

:CHANnel1:LOAD 50	Set channel 1 output load to 50Ω
:CHANnel1:LOAD?	The query returns 5e+1

:CHANnel<n>:ADD

- Command format:
  :CHANnel<n>:ADD {{1|ON}|{0|OFF}}
  :CHANnel<n>:ADD?
- Function description:
  Sets channel overlay to turn on or off for the specified channel.
  <n>: Channel number, n takes the value 1 or 2.
- Return format:
  The query returns the overlay status of the specified channel, 0 means closed, 1 means open.
- Example:

:CHANnel1:ADD ON	Settings turn on channel 1 and channel 2 overlay through the channel 1 output
:CHANnel1:ADD?	The query returns 1

:CHANnel<n>:COPY

- Command format:
  :CHANnel<n>:COPY
- Function description:
  Execute channel copy operation. This instruction does not support query operation.
  <n>: Channel number, n takes the value 1 or 2.
- Return format:
  No return value.
- Example:

:CHANnel1:COPY	Data from channel 1 is copied to channel 2

:CHANnel<n>:LIMit:ENABle

- Command format:
  :CHANnel<n>:LIMit:ENABle {{1 | ON} | {0 | OFF}}
  :CHANnel<n>:LIMit:ENABle?
- Function description:
  Set the specified channel limiter switch.
  <n>: Channel number, n takes the value 1 or 2.
- Return format:

The query returns the limiting status of the specified channel.

- Example:

:CHANnel1:LIMit:ENABle ON	Set up channel 1 limiter
:CHANnel1:LIMit:ENABle?	The query returns 1

:CHANnel<n>:LIMit:LOWer

- Command format:
  :CHANnel<n>:LIMit:LOWer {<voltage>}
  :CHANnel<n>:LIMit:LOWer?
- Function description:
  Set the specified channel limiting lower limit value.
  <voltage> represents voltage, the unit is specified by the current channel.
  <n>: Channel number, n takes the value 1 or 2.
- Return format:
  The query returns the limiting lower limit value of the specified channel, using scientific notation.
- Example:

:CHANnel1:LIMit:LOWer 2	Set channel 1 limiting lower limit to 2V
:CHANnel1:LIMit:LOWer?	The query returns 2e+0

:CHANnel<n>:LIMit:UPPer

- Command format:
  :CHANnel<n>:LIMit:UPPer {<voltage>}
  :CHANnel<n>:LIMit:UPPer?
- Function description:
  Set the limiter limit value of the specified channel.
  <voltage> represents voltage, the unit is specified by the current channel.
  <n>: Channel number, n takes the value 1 or 2.
- Return format:
  The query returns the limiting upper limit value of the specified channel, using scientific notation.
- Example:

:CHANnel1:LIMit:UPPer 2	Set the upper limit of channel 1 limiter to 2V
:CHANnel1:LIMit:UPPer?	The query returns 2e+0

:CHANnel:COUPle<m>:ONOFf

- Command format:
  :CHANnel:COUPle<m>:ONOFf {{1|ON}|{0|OFF}}
  :CHANnel:COUPle<m>:ONOFf?
- Function description:
  Sets channel coupling on or off.
  <m>: Channel number, m takes the value 1.
  1 means one and two channel coupling
- Return format:
  The query returns the status of the specified channel coupling, 0 means closed, 1 means open.
- Example:

:CHANnel:COUPle1:ONOff ON	Set up 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?

- Function description:
  Sets the specified channel coupling type. This command is only valid when the channel coupling switch is turned on.
  <m>: Channel number, m takes the value 1.
  1 means one and two channel coupling
- Return format:
  The query returns the channel coupling type.
- Example:

:CHANnel:COUPle1:TYPe PARAm	Set channel coupling to parameter coupling type
:CHANnel:COUPle1:TYPe?	Query returns PARAm

:CHANnel:COUPle<m>:FREQuency

- Command format:
  :CHANnel:COUPle<m>:FREQuency {{0 | OFF}}
  :CHANnel:COUPle<m>:FREQuency?
- Function description:
  Set the channel frequency coupling to off, and the channel only has channel 1 and 2 coupling. This command is only valid when the channel coupling switch is turned on.
  <m>: Channel number, m takes the value 1.
  1 means one and two channel coupling
- Return format:
  The query returns the channel frequency coupling status, 0 means closed, 1 means other states.
- Example:

:CHANnel:COUPle1:FREQuency OFF	Turn off channel 1 and 2 frequency coupling
:CHANnel:COUPle1:FREQuency?	Return 0

:CHANnel:COUPle<m>:FREQuency:SCALe

- Command format:
  :CHANnel:COUPle<m>:FREQuency:SCALe <scale>
  :CHANnel:COUPle<m>:FREQuency:SCALe?
- Function description:
  Set the channel coupling frequency ratio. The channel only has channel 1 and 2 coupling. This command is only valid when the channel coupling switch is turned on.
  <scale>: Coupling frequency scale.
  <m>: Channel number, m takes the value 1.
  1 means one and two channel coupling
- Return format:
  The query returns the channel coupling frequency ratio in scientific notation.
- Example:

:CHANnel:COUPle1:FREQuency:SCALe 0.1	Set the coupling ratio of channel 2 to channel 1 to 0.1
:CHANnel:COUPle1:FREQuency:SCALe?	Return 1e-1

:CHANnel:COUPle<m>:FREQuency:DEV

- Command format:
  :CHANnel:COUPle<m>:FREQuency:DEV <dev >
  :CHANnel:COUPle<m>:FREQuency:DEV?
- Function description:
  Set channel coupling frequencydeviation, the channel only has channel 1 and channel 2 coupling. This command is only valid when the channel coupling switch is turned on.
  <scale>: Coupling frequency deviation, unit Hz.
  <m>: Channel number, m takes the value 1.
  1 means one and two channel coupling
- Return format:

Query returns channel coupling frequencydeviation, returns scientific notation.

- Example:

:CHANnel:COUPle1:FREQuency:DEV 100	Set 2-channel to 1-channel couplingdeviationto 100Hz
:CHANnel:COUPle1:FREQuency:DEV?	Return 1e+2

:CHANnel:COUPle<m>:PHASe

- Command format:
  :CHANnel:COUPle<m>:PHASe {{0 | OFF}}
  :CHANnel:COUPle<m>:PHASe?
- Function description:
  Set the channel phase coupling to off, and the channel only has channel 1 and 2 coupling. This command is only valid when the channel coupling switch is turned on.
  <m>: Channel number, m takes the value 1.
  1 means one and two channel coupling
- Return format:
  The query returns the channel coupling status, 0 means closed, 1 means other states.
- Example:

:CHANnel:COUPle1:PHASe OFF	Turn off phase coupling of channels 1 and 2
:CHANnel:COUPle1:PHASe?	Return 0

:CHANnel:COUPle<m>:PHASe:SCALe

- Command format:
  :CHANnel:COUPle<m>:PHASe:SCALe <scale>
  :CHANnel:COUPle<m>:PHASe:SCALe?
- Function description:
  Set the channel coupling phase ratio. The channel only has channel 1 and 2 coupling. This command is only valid when the channel coupling switch is turned on.
  <scale>: couplingPhaseProportion.
  <m>: Channel number, m takes the value 1.
  1 means one and two channel coupling
- Return format:
  The query returns the channel coupling phase ratio in scientific notation.
- Example:

:CHANnel:COUPle1:PHASe:SCALe 0.1	Set the coupling ratio of channel 2 to channel 1 to 0.1
:CHANnel:COUPle1:PHASe:SCALe?	Return 1e-1

:CHANnel:COUPle<m>:PHASe:DEV

- Command format:
  :CHANnel:COUPle<m>:PHASe:DEV <dev >
  :CHANnel:COUPle<m>:PHASe:DEV?
- Function description:
  Set channel coupling phasedeviation, the channel only has channel 1 and channel 2 coupling. This command is only valid when the channel coupling switch is turned on.
  <scale>: couplingPhaseDeviation, unit °.
  <m>: Channel number, m takes the value 1.
  1 means one and two channel coupling
- Return format:
  Query returns channel coupling phasedeviation, returns scientific notation.
- Example:

:CHANnel:COUPle1:PHASe:DEV 100	Set 2-channel to 1-channel couplingdeviationis 100°
:CHANnel:COUPle1:PHASe:DEV?	Return 1e+2

:CHANnel:COUPle<m>:AMPLitude

- **Command format:**
  :CHANnel:COUPle<m>:AMPLitude {{0 | OFF}}
  :CHANnel:COUPle<m>:AMPLitude?
- **Function description:**
  Set the channel amplitude coupling to off, and the channel only has channel 1 and 2 coupling. This command is only valid when the channel coupling switch is turned on.
  <m>: Channel number, m takes the value 1.
  1 means one and two channel coupling
- **Return format:**
  The query returns the channel coupling amplitude status, 0 means closed, 1 means other states.
- **Example:**

:CHANnel:COUPle1:AMPLitude OFF	Turn off channel 1 and 2 amplitude coupling
:CHANnel:COUPle1:AMPLitude?	Return 0

:CHANnel:COUPle<m>:AMPLitude:SCALe

- **Command format:**
  :CHANnel:COUPle<m>:AMPLitude:SCALe <scale>
  :CHANnel:COUPle<m>:AMPLitude:SCALe?
- **Function description:**
  Set the channel coupling amplitude ratio. The channel only has channel 1 and 2 coupling. This command is only valid when the channel coupling switch is turned on.
  <scale>: couplingAmplitudeProportion.
  <m>: Channel number, m takes the value 1.
  1 means one and two channel coupling
- **Return format:**
  The query returns the channel coupling amplitude ratio and returns scientific notation.
- **Example:**

:CHANnel:COUPle1:AMPLitude:SCALe 0.1	Set the coupling ratio of channel 2 to channel 1 to 0.1
:CHANnel:COUPle1:AMPLitude:SCALe?	Return 1e-1

:CHANnel:COUPle<m>:AMPLitude:DEV

- **Command format:**
  :CHANnel:COUPle<m>:AMPLitude:DEV <dev >
  :CHANnel:COUPle<m>:AMPLitude:DEV?
- **Function description:**
  Set channel coupling amplitudedeviation, the channel only has channel 1 and channel 2 coupling. This command is only valid when the channel coupling switch is turned on.
  <scale>: couplingAmplitudeDeviation, unit Vpp.
  <m>: Channel number, m takes the value 1.
  1 means one and two channel coupling
- **Return format:**
  Query returns channel coupling amplitudedeviation, returns scientific notation.
- **Example:**

:CHANnel:COUPle1:AMPLitude:DEV 1	Set 2-channel to 1-channel couplingdeviationto 1Vpp
:CHANnel:COUPle1:AMPLitude:DEV?	Return 1e+2

:CHANnel<n>:PSK:PNCode

- **Command format:**
  :CHANnel<n>:PSK:PNCode <code>
  :CHANnel<n>:PSK:PNCode?

➢ Function description:
Set the PN code of the specified channel. This command is valid for the waveforms of the PN code function under biphase shift keying modulation, quadrature phase shift keying modulation and quadrature modulation.
<code>: Indicates PN code, as shown below:
{ PN7|PN9|PN11|PN15|PN17|PN21|PN23|PN25}
<n>: Channel number, n takes the value 1 or 2.

➢ Return format:
The query returns the PN code under bi-phase shift keying modulation, quadrature phase-shift keying modulation and quadrature modulation of the specified channel.

➢ Example:

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

:CHANnel<n>:YOU:PNCode

➢ Command format:
:CHANnel<n>:QAM:PNCode <code>
:CHANnel<n>:QAM:PNCode?

➢ Function description:
Set the PN code of the specified channel. This command is valid for the waveforms of the PN code function under biphase shift keying modulation, quadrature phase shift keying modulation and quadrature modulation.
<code>: Indicates PN code, as shown below:
{PN7|PN9|PN11|PN15|PN17|PN21|PN23|PN25}
<n>: Channel number, n takes the value 1 or 2.

➢ Return format:
The query returns the PN code of the specified channel.

➢ Example:

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

:CHANnel<n>:TRIGger:SOURce

➢ Command format:
:CHANnel<n>:TRIGger:SOURce {INTernal|EXTernal| MANual}
:CHANnel<n>:TRIGger:SOURce?

➢ Function description:
Set the trigger source of the specified channel. This command is only valid for frequency sweep and burst functions.
<n>: Channel number, n takes the value 1 or 2.

➢ Return format:
The query returns the trigger source of the specified channel.

➢ Example:

:CHANnel1:TRIGger:SOURce INTernal	Set channel 1 internal trigger source
:CHANnel1:TRIGger:SOURce?	The query returns INTernal

:CHANnel<n>:TRIGger:OUTPut

➢ Command format:
:CHANnel<n>:TRIGger:OUTPut {{1 | ON} | {0 | OFF}}
:CHANnel<n>:TRIGger:OUTPut?

➢ Function description:
Set the trigger output status of the specified channel. This command is only valid for frequency sweep and burst functions.
<n>: Channel number, n takes the value 1 or 2.

- **Return format:**
  The query returns the trigger output status of the specified channel.
- **Example:**
  :CHANnel1:TRIGger:OUTPut ON Set channel 1 trigger output to open
  :CHANnel1:TRIGger:OUTPut? The query returns 1

:CHANnel<n>:TRIGger:OUTEdge

- **Command format:**
  :CHANnel<n>:TRIGger:OUTEdge { RISe|FALL}
  :CHANnel<n>:TRIGger:OUTEdge?
- **Function description:**
  Set the trigger output edge of the specified channel. This command is only valid for frequency sweep and burst functions.
  <n>: Channel number, n takes the value 1 or 2.
- **Return format:**
  The query returns the trigger output edge of the specified channel.
- **Example:**
  :CHANnel1:TRIGger:OUTEdge RISe Set channel 1 rising edge trigger output mode
  :CHANnel1:TRIGger:OUTEdge? The query returns RISe

Continuous

:CHANnel<n>:BASe:WAVe

- **Command format:**
  :CHANnel<n>:BASe:WAVe { SINe | SQUare | PULSe | RAMP | ARB | NOISe | DC | HARMonic |EXP }
  :CHANnel<n>:BASe:WAVe?
- **Function description:**
  Set the fundamental wave type of the specified channel. They are sine wave, square wave, pulse wave, triangle wave, arbitrary wave, noise, DC, harmonic and expression waveforms.
  <n>: Channel number, n takes the value 1 or 2.
- **Return format:**
  The query returns the fundamental wave type of the specified channel.
- **Example:**
  :CHANnel1:BASe:WAVe SINe Set the basic type of channel 1 to sine wave
  :CHANnel1:BASe:WAVe? The query returns SINe

:CHANnel<n>:BASe:FREQuency

- **Command format:**
  :CHANnel<n>:BASe:FREQuency {<freq>}
  :CHANnel<n>:BASe:FREQuency?
- **Function description:**
  Set the output frequency of the specified channel, in Hz when used.
  <freq> represents the frequency value in Hz. (1e-6s ~ the maximum frequency allowed by the current waveform)
  <n>: Channel number, n takes the value 1 or 2.
- **Return format:**
  The query returns the output frequency of the specified channel, using scientific notation. The unit of the return value is Hz.
- **Example:**
  :CHANnel1:BASe:FREQuency 2000 Set channel 1 output frequency to 2KHz
  :CHANnel1:BASe:FREQuency? The query returns 2e+3

:CHANnel<n>:BASe:PERiod

- **Command format:**
  :CHANnel<n>:BASe:PERiod { <period>}
  :CHANnel<n>:BASe:PERiod?
- **Function description:**
  Set the output period of the specified channel.
  <period> represents period, unit S.
  If it is a sine wave: the range is (16.7ns ~ 1000000s)
  <n>: Channel number, n takes the value 1 or 2.
- **Return format:**
  The query returns the limiting upper limit value of the specified channel, using scientific notation, and the return value is in S as the unit.
- **Example:**

:CHANnel1:BASe:PERiod 0.002	Set channel 1 output period to 2ms
:CHANnel1:BASe:PERiod?	The query returns 2e-3

:CHANnel<n>:BASe:PHASe

- **Command format:**
  :CHANnel<n>:BASe:PHASe { <phase>}
  :CHANnel<n>:BASe:PHASe?
- **Function description:**
  Set the output phase of the specified channel.
  <phase> represents phase, unit °, range -360~+360.
  <n>: Channel number, n takes the value 1 or 2.
- **Return format:**
  The query returns the output phase of the specified channel, using scientific notation.
- **Example:**

:CHANnel1:BASe:PHASe 20	Set channel 1 output phase to 20°
:CHANnel1:BASe:PHASe?	The query returns 2e+01

:CHANnel<n>:BASe:AMPLitude

- **Command format:**
  :CHANnel<n>:BASe:AMPLitude { <amp>}
  :CHANnel<n>:BASe:AMPLitude?
- **Function description:**
  Set the output amplitude of the specified channel.
  <amp> represents voltage, the unit is specified by the current channel. 1mVpp ~ maximum output value under current load.
  If the current unit is VPP, the maximum value under current load = current load * 20 / (50 + current load)
  <n>: Channel number, n takes the value 1 or 2.
- **Return format:**
  The query returns the output amplitude of the specified channel, using scientific notation.
- **Example:**

:CHANnel1:BASe:AMPLitude 2	Set channel 1 output amplitude to 2V
:CHANnel1:BASe:AMPLitude?	The query returns 2e+0

- **Notice:**
  When setting the amplitude in units of dBm and Vrms, there will be a certain loss of accuracy in the return value of scpi when the value is a negative number. For example: CHAN1: BASe:AMPL -0.45dBm returns: -4.500003e-01

:CHANnel<n>:BASe:OFFSet

- **Command format:**

:CHANnel<n>:BASe:OFFSet { <voltage>}
:CHANnel<n>:BASe:OFFSet?

- Function description:
  Set the output DC offset of the specified channel.
  <voltage> represents voltage, unit V. The range is: 0~±maximum DC under current load.
  Maximum DC under current load = current load*10/(50+current load) - current AC minimum value/2;
  The minimum AC value is 2mVpp, and the DC mode is 0;
  <n>: Channel number, n takes the value 1 or 2.
- Return format:
  The query returns the output DC offset of the specified channel, using scientific notation.
- Example:

:CHANnel1:BASe:OFFSet 2	Set channel 1 output DC offset to 2V
:CHANnel1:BASe:OFFSet?	The query returns 2e+0

:CHANnel<n>:BASe:HIGH

- Command format:
  :CHANnel<n>:BASe:HIGH { <voltage>}
  :CHANnel<n>:BASe:HIGH?
- Function description:
  Set the specified channel signal output high value.
  <voltage> represents voltage, the unit is specified by the current channel.
  <n>: Channel number, n takes the value 1 or 2.
- Return format:
  The query returns the high value of the specified channel signal output, using scientific notation.
- Example:

:CHANnel1:BASe:HIGH 2	Set the high value of channel 1 signal output to 2V
:CHANnel1:BASe:HIGH?	The query returns 2e+0

:CHANnel<n>:BASe:LOW

- Command format:
  :CHANnel<n>:BASe:LOW { <voltage>}
  :CHANnel<n>:BASe:LOW?
- Function description:
  Set the specified channel signal output low value.
  <voltage> represents voltage, the unit is specified by the current channel.
  <n>: Channel number, n takes the value 1 or 2.
- Return format:
  The query returns the low value of the specified channel signal output, using scientific notation.
- Example:

:CHANnel1:BASe:LOW 2	Set the low value of channel 1 signal output to 2V
:CHANnel1:BASe:LOW?	The query returns 2e+0

:CHANnel<n>:BASe:PWIDth

- Command format:
  :CHANnel<n>:BASe:PWIDth { <pulsewidth>}
  :CHANnel<n>:BASe:PWIDth?
- Function description:
  Set the pulse width of the specified channel signal output.
  <pulsewidth> represents the pulse width, in seconds (s).
  <n>: Channel number, n takes the value 1 or 2.
- Return format:
  The query returns the pulse width of the specified channel signal, using scientific notation.
- Example:

:CHANnel1:BASe:PWIDth 0.0004	Set the pulse width of channel 1 signal output to 400us
:CHANnel1:BASe:PWIDth?	The query returns 4.000000e-04

:CHANnel<n>:BASe:DUTY

- **Command format:**
  :CHANnel<n>:BASe:DUTY { <duty>}
  :CHANnel<n>:BASe:DUTY?
- **Function description:**
  Set the specified channel signal output duty cycle.
  <duty> indicates duty cycle, unit %, range 0~100.
  <n>: Channel number, n takes the value 1 or 2.
- **Return format:**
  The query returns the signal output duty cycle of the specified channel and returns the data in scientific notation.
- **Example:**

:CHANnel1:BASe:DUTY 20	Set channel 1 signal output duty cycle to 20%
:CHANnel1:BASe:DUTY?	The query returns 2e+1

:CHANnel<n>:BASe:ARB

- **Command format:**
  :CHANnel<n>:BASe:ARB <source>,<filename>
  :CHANnel<n>:BASe:ARB?
- **Function description:**
  Set the specified channel to load the arbitrary waveform data of a certain file under the fundamental arbitrary wave source.
  <n>: Channel number, n takes the value 1 or 2.
  <source>: {LOCal|INTernal|EXTernal}, respectively local, internal and external.
  <filename>: arbitrary waveform file name.
- **Example:**
  :CHANnel1:BASe:ARB LOCal, "test.bsv"

:CHANnel<n>:RAMP:SYMMetry

- **Command format:**
  :CHANnel<n>:RAMP:SYMMetry { < symmetry >}
  :CHANnel<n>:RAMP:SYMMetry?
- **Function description:**
  Set the symmetry of the ramp signal output of the specified channel.
  <symmetry> represents the degree of symmetry, unit %, range 0~100.
  <n>: Channel number, n takes the value 1 or 2.
- **Return format:**
  The query returns the symmetry of the ramp signal output of the specified channel.
- **Example:**

:CHANnel1:RAMP:SYMMetry 20	Set the channel 1 ramp signal symmetry to 20%
:CHANnel1:RAMP:SYMMetry?	The query returns 20

:CHANnel<n>:PULSe:RISe

- **Command format:**
  :CHANnel<n>:PULSe:RISe {<width>}
  :CHANnel<n>:PULSe:RISe?
- **Function description:**
  Set the rising edge pulse width of the specified channel signal pulse wave.
  <width> represents pulse width, unit S.

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

➢ Return format:

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

➢ Example:

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

:CHANnel<n>:PULSe:FALL

➢ Command format:

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

➢ Function description:

Set the falling edge pulse width of the specified channel signal pulse wave.
<width> represents pulse width, unit S.
<n>: Channel number, n takes the value 1 or 2.

➢ Return format:

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

➢ Example:

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

:CHANnel<n>:HARMonic:TYPe?

➢ Command format:

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

➢ Function description:

Set the harmonic type of the specified channel.
<n>: Channel number, n takes the value 1 or 2.

➢ Return format:

The query returns the harmonic type of the specified channel.

➢ Example:

:CHANnel1:HARMonic:TYPe ODD	Set the harmonic type of channel 1 to odd harmonics
:CHANnel1:HARMonic:TYPe?	Query returns ODD

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

➢ Command format:

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

➢ Function description:

Set the maximum harmonic order of the specified channel.
< order >: Harmonic order, range 2~16.
<n>: Channel number, n takes the value 1 or 2.

➢ Return format:

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

➢ Example:

:CHANnel1:HARMonic:TOTal:ORDer 2	Set the maximum harmonic 2nd of channel 1
:CHANnel1:HARMonic:TOTal:ORDer?	The query returns 2

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

- **Command format:**
  :CHANnel<n>:HARMonic:USER:TYPe #H<order>
  :CHANnel<n>:HARMonic:USER:TYPe?
- **Function description:**
  Set the custom harmonic type of the specified channel.
  < order >: Custom harmonic type, #H represents hexadecimal number. X0111 1111 1111 1111 bits represent harmonic switches respectively.
  <n>: Channel number, n takes the value 1 or 2.
- **Return format:**
  The query returns the custom harmonic type of the specified channel and returns integer data.
- **Example:**

:CHANnel1:HARMonic:USER:TYPe #H7FFF	Set channel 1 custom harmonic type
:CHANnel1:HARMonic:USER:TYPe?	The query returns 32767

:CHANnel<n>:HARMonic:ORDer?

- **Command format:**
  :CHANnel<n>:HARMonic:ORDer <order>
  :CHANnel<n>:HARMonic:ORDer?
- **Function description:**
  Set the total number of harmonics of the specified channel.
  < order >: Total number of harmonics, range 2~16.
  <n>: Channel number, n takes the value 1 or 2.
- **Return format:**
  The query returns the total number of harmonics of the specified channel and returns integer data.
- **Example:**

:CHANnel1:HARMonic:ORDer 2	Set the total number of harmonics of channel 1 to 2
:CHANnel1:HARMonic:ORDer?	The query returns 2

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

- **Command format:**
  :CHANnel<n>:HARMonic:ORDer<m>:AMPLitude <amp>
  :CHANnel<n>:HARMonic:ORDer<m>:AMPLitude?
- **Function description:**
  Set the amplitude value of the specified harmonic order in the specified channel.
  < amp >: Amplitude value, unit Vpp.
  <n>: Channel number, n takes the value 1 or 2.
  <m>: Harmonic order, m ranges from 2 to 16.
- **Return format:**
  The query returns the amplitude value of the specified harmonic order in the specified channel, using scientific notation.
- **Example:**

:CHANnel1:HARM:ORDER2:AMPL 0.02	Set the amplitude value of the second harmonic of channel 1 to 20mVpp
:CHANnel1:HARM:ORDER2:AMPL?	The query returns 2e-2

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

- **Command format:**
  :CHANnel<n>:HARMonic:ORDer<m>:PHASe <phase>
  :CHANnel<n>:HARMonic:ORDer<m>:PHASe?
- **Function description:**
  Set the phase value of the specified harmonic order in the specified channel.

<phase>: Phase value, unit °.
<n>: Channel number, n takes the value 1 or 2.
<m>: Harmonic order, m ranges from 2 to 16.

➢ Return format:

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

➢ Example:

:CHANnel1:HARM:ORDer2:PHASe 20 Set the phase value of the second harmonic of channel 1 to 20°
:CHANnel1:HARM:ORDer2:PHASe? The query returns 2e+1

:CHANnel<n>: EXP:EXPStart

➢ Command format:

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

➢ Function description:

Sets the starting value (minimum value) of signal output for the specified channel expression.
<start value> represents the starting value (minimum value), entered in radians.
<n>: Channel number, n takes the value 1 or 2.

➢ Return format:

The query returns the starting value of the specified channel expression signal output, returned in scientific notation.

➢ Example:

:CHANnel1:EXP:EXPStart 0.1 Set the channel 1 expression signal start value to 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?

➢ Function description:

Sets the end value (maximum value) of the specified channel expression signal output.
<end value> represents the end value, entered in radians.
<n>: Channel number, n takes the value 1 or 2.

➢ Return format:

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

➢ Example:

:CHANnel1:EXP:EXPEnd 3.1415926 Set the end value (maximum value) of the channel 1 expression signal to 3.1415926
:CHANnel1:EXP:EXPEnd? The query returns 3.1415926e+00

:CHANnel<n>: EXP:EXPStr

➢ Command format:

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

➢ Function description:

Sets the expression for the specified channel expression signal. Only x can be written as the independent variable, and all letters in the expression are lowercase.
<expstr> represents the expression, input in string form.
Supported functions are: 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)。
Operators cannot be omitted between numeric variables. For example, 3*x cannot be written as 3x. There cannot be spaces in the expression.

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

- Return format:
  The query returns the expression of the specified channel expression signal, returned in string form.
- Example:
  :CHANnel1:EXP:EXPStr "sin(x)" sets the expression of the channel 1 expression signal to sin(x)
  :CHANnel1:EXP:EXPStr? The query returns sin(x)

:CHANnel<n>:ARB:MODe

- Command format:
  :CHANnel<n>:ARB:MODe {DDS | POINTS }
  :CHANnel<n>:ARB:MODe?
- Function description:
  Set the arbitrary waveform output mode of the specified channel to DDS and point-by-point modes respectively. DDS corresponds to [Playback Mode] off, POINTS corresponds to [Playback Mode] on.
  <n>: Channel number, n takes the value 1 or 2.
- Return format:
  The query returns the arbitrary waveform mode of the specified channel ([Playback Mode] on or off).
- Example:
  :CHANnel1:ARB:MODe DDS Set channel 1 arbitrary waveform mode DDS output mode
  :CHANnel1:ARB:MODe? Query returns DDS

Modulation

:CHANnel<n>:MODulate:TYPe

- Command format:
  :CHANnel<n>:MODulate:TYPe <type>
  :CHANnel<n>:MODulate:TYPe?
- Function description:
  Set the signal modulation type of the specified channel.
  <type>: { AM |FM|PM|ASK|FSK|PSK|BPSK|QPSK|OSK|QAM|PWM |SUM|DSBAM }
  AM, frequency modulation, phase modulation, amplitude shift keying, frequency shift keying, phase shift keying, biphase shift keying, quadrature phase shift keying, oscillation keying, quadrature modulation, pulse width modulation, summation modulation, bilateral With AM. <n>: Channel number, n takes the value 1 or 2.
- Return format:
  The query returns the signal modulation type of the specified channel.
- Example:
  :CHANnel1:MODulate:TYPe AM Set channel 1 signal AM modulation
  :CHANnel1:MODulate:TYPe? Query returns AM

:CHANnel<n>:MODulate:WAVe

- Command format:
  :CHANnel<n>:MODulate:WAVe { SINe|SQUare|UPRamp|DNRamp|ARB|NOISe }
  :CHANnel<n>:MODulate:WAVe?
- Function description:
  Set the specified channel signal modulation wave type, which are sine wave, square wave, upper triangle, lower triangle, arbitrary wave, and noise.
  <n>: Channel number, n takes the value 1 or 2.
- Return format:
  The query returns the modulation wave type of the specified channel signal.
- Example:
  :CHANnel1:MODulate:WAVe SINe Set channel 1 signal modulation wave type to sine wave
  :CHANnel1:MODulate:WAVe? The query returns SINe

:CHANnel<n>:MODulate:SOURce

- **Command format:**
  :CHANnel<n>:MODulate:SOURce { INTernal|EXTernal }
  :CHANnel<n>:MODulate:SOURce?
- **Function description:**
  Set the specified channel modulation source, internal and external.
  <n>: Channel number, n takes the value 1 or 2.
- **Return format:**
  The query returns the modulation source of the specified channel.
- **Example:**

:CHANnel1:MODulate:SOURce INTernal	Set channel 1 modulation source to internal
:CHANnel1:MODulate:SOURce?	The query returns INTernal

:CHANnel<n>:MODulate:FREQuency

- **Command format:**
  :CHANnel<n>:MODulate:FREQuency {<freq>}
  :CHANnel<n>:MODulate:FREQuency?
- **Function description:**
  Set the modulation frequency of the specified channel signal.
  <freq> represents frequency in Hz.
  <n>: Channel number, n takes the value 1 or 2.
- **Return format:**
  The query returns the modulation frequency of the specified channel signal and returns the sampling scientific notation representation.
- **Example:**

:CHANnel1:MODulate:FREQuency 2000	Set channel 1 signal modulation frequency to 2KHz
:CHANnel1:MODulate:FREQuency?	The query returns 2e+3

:CHANnel<n>:MODulate:IQMap

- **Command format:**
  :CHANnel<n>:MODulate: IQMap {<IQ TYPE>}
  :CHANnel<n>:MODulate: IQMap?
- **Function description:**
  Setting the IQ type of the specified QAM can be:
  QAM4，QAM8，QAM16，QAM32，QAM64，QAM128，QAM256。
  <IQ TYPE> indicates the IO mapping type.
  <n>: Channel number, n takes the value 1 or 2.
- **Return format:**
  The query returns the IQ type of the specified channel
- **Example:**

:CHANnel1:MODulate:IQMap QAM32	Set channel 1 modulation IQ mapping to QAM32
:CHANnel1:MODulate:IQMap?	The query returns QAM32

:CHANnel<n>:MODulate:ARB

- **Command format:**
  :CHANnel<n>:MODulate:ARB <source>,<filename>
  :CHANnel<n>:MODulate:ARB?
- **Function description:**
  Set the specified channel to load arbitrary waveform data from a certain file under the modulated arbitrary wave source.
  <n>: Channel number, n takes the value 1 or 2.

<source>: {LOCal|INTernal|EXTernal}, respectively local, internal and external.
<filename>: arbitrary waveform file name.

- Example:
  :CHANnel1:MODulate:ARB LOCal, "test.bsv"

:CHANnel<n>:MODulate:DEPTh

- Command format:
  :CHANnel<n>:MODulate:DEPTh { <depth>}
  :CHANnel<n>:MODulate:DEPTh?
- Function description:
  Sets the modulation depth of the specified channel.
  <depth> represents the modulation depth, unit %. 0% ~ 100%, where AM modulation depth is 0% ~ 120%
  <n>: Channel number, n takes the value 1 or 2.
- Return format:
  The query returns the modulation depth of the specified channel, using scientific notation.
- Example:

:CHANnel1:MODulate:DEPTh 50	Set channel one modulation depth to 50%
:CHANnel1:MODulate:DEPTh?	The query returns 5e+1

:CHANnel<n>:MODulate:RATio

- Command format:
  :CHANnel<n>:MODulate:Ratio <system>
  :CHANnel<n>:MODulate:RATio?
- Function description:
  Set the specified channel modulation rate value. This command is only valid for modulation types with rate function.
  <ratio> represents the rate, unit is Hz
  <n>: Channel number, n takes the value 1 or 2.
- Return format:
  The query returns the modulation rate value of the specified channel, using scientific notation.
- Example:

:CHANnel1:MODulate:RATio 100	Set channel 1 bit rate to 100Hz
: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?
- Function description:
  Set the frequency deviation of the specified channel.
  <freq> represents the frequency offset in Hz. 0Hz ~ current fundamental frequency
  <n>: Channel number, n takes the value 1 or 2.
- Return format:
  The query returns the frequency offset of the specified channel, using scientific notation to return data.
- Example:

:CHANnel1:FM:FREQuency:DEV 2000	Set channel one frequency offset to 2KHz
:CHANnel1:FM:FREQuency:DEV?	The query returns 2e+3

:CHANnel<n>:PM:PHASe:DEV

- Command format:
  :CHANnel<n>:PM:PHASe:DEV { <phase>}

:CHANnel<n>:PM:PHASe:DEV?

- Function description:
  Set the output phase deviation of the specified channel.
  <phase> represents phase offset, unit °, range 0~360.
  <n>: Channel number, n takes the value 1 or 2.
- Return format:
  The query returns the output phase offset of the specified channel, using scientific notation.
- Example:

:CHANnel1:PM:PHASe:DEV 30	Set channel one phase offset to 30°
:CHANnel1:PM:PHASe:DEV?	The query returns 3e+01

:CHANnel<n>:PWM:DUTY

- Command format:
  :CHANnel<n>:PWM:DUTY { <duty>}
  :CHANnel<n>:PWM:DUTY?
- Function description:
  Set the duty cycle of the specified channel output under pulse width modulation.
  <duty> represents the duty cycle, unit %, range 0~100.
  <n>: Channel number, n takes the value 1 or 2.
- Return format:
  The query returns the pulse width deviation under pulse width modulation of the specified channel, and returns the data in scientific notation.
- Example:

:CHANnel1:PWM:DUTY 10	Set channel one duty cycle to 10%
:CHANnel1:PWM:DUTY?	The query returns 1e+1

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

- Command format:
  :CHANnel<n>:FSK:FREQuency { <freq>}
  :CHANnel<n>:FSK:FREQuency?
- Function description:
  Set the specified channel output multi-frequency shift keying jump frequency. The modulation method must be specified in advance for this command to take effect.
  <freq> represents frequency in Hz.
  <n>: Channel number, n takes the value 1 or 2.
- Return format:
  The query returns the output hopping frequency of the specified channel and returns the data in scientific notation.
- Example:

:CHANnel1:FSK:FREQ 2000	Set channel one output frequency hopping frequency to 2KHz
:CHANnel1:FSK:FREQ?	The query returns 2e+3

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

- Command format:
  :CHANnel<n>:PSK:PHASe<m> { < phase >}
  :CHANnel<n>:PSK:PHASe<m>?
- Function description:
  Set the specified channel output multi-phase shift keying phase value. The modulation method must be specified in advance for this command to take effect.
  <phase> represents phase, unit °, range -360~+360.
  <n>: Channel number, n takes the value 1, 2, 3, 4.
  <m>: Phase number, PSK value is 1; BPSK value is 1, 2; QPSK value is 1, 2, 3, 4;
- Return format:

The query returns the phase shift keying phase value of the specified channel, and the data is returned in scientific notation.

- Example:

:CHANnel1:PSK:PHAS1 90	Set channel one output phase to 90°
:CHANnel1:PSK:PHAS1?	The query returns 9e+1

:CHANnel<n>:OSK:TIMe

- Command format:
  :CHANnel<n>:OSK:TIMe { <time>}
  :CHANnel<n>:OSK:TIMe?
- Function description:
  Set the oscillation time of oscillation keying in the specified channel modulation mode.
  <time> represents the oscillation time, unit S.
  <n>: Channel number, n takes the value 1 or 2.
- Return format:
  The query returns the oscillation time of oscillation keying in the specified channel modulation mode, and returns the data in scientific notation.
- Example:

:CHANnel1:OSK:TIMe 2ms	Set the oscillation time of channel 1 oscillation keying to 2ms
:CHANnel1:OSK:TIMe?	The query returns 2e-3

Sweep

:CHANnel<n>:SWEep:TYPe

- Command format:
  :CHANnel<n>:SWEep:TYPe { LINe|LOG }
  :CHANnel<n>:SWEep:TYPe?
- Function description:
  Set the frequency sweep mode of the specified channel, which is linear sweep or logarithmic sweep.
  <n>: Channel number, n takes the value 1 or 2.
- Return format:
  The query returns the frequency sweep mode of the specified channel.
- Example:

:CHANnel1:SWEep:TYPe LINe	Set channel one linear sweep mode
:CHANnel1:SWEep:TYPe?	Query returns LINE

:CHANnel<n>:SWEep:FREQuency:STARt

- Command format:
  :CHANnel<n>:SWEep:FREQuency:STARt <freq>
  :CHANnel<n>:SWEep:FREQuency:STARt?
- Function description:
  Set the starting frequency of the specified channel sweep.
  <freq> represents frequency in Hz.
  <n>: Channel number, n takes the value 1 or 2.
- Return format:
  The query returns the starting frequency of the specified channel sweep, and returns the data in scientific notation.
- Example:

:CHANnel1:SWE:FREQ:STAR 2000	Set the starting frequency of channel one output sweep to 2KHz
:CHANnel1:SWE:FREQ:STAR?	The query returns 2e+3

:CHANnel<n>:SWEep:FREQuency:STOP

- **Command format:**
  :CHANnel<n>:SWEep:FREQuency:STOP <freq>
  :CHANnel<n>:SWEep:FREQuency:STOP?
- **Function description:**
  Sets the cutoff frequency of the specified channel sweep.
  <freq> represents frequency in Hz.
  <n>: Channel number, n takes the value 1 or 2.
- **Return format:**
  The query returns the cutoff frequency of the specified channel output sweep, and returns the data in scientific notation.
- **Example:**

:CHANnel1:SWE:FREQ:STOP 2000	Set the cutoff frequency of channel one output sweep to 2KHz
:CHANnel1:SWE:FREQ:STOP?	The query returns 2e+3

:CHANnel<n>:SWEep:TIMe

- **Command format:**
  :CHANnel<n>:SWEEP:TIMe <time>
  :CHANnel<n>:SWEEP:TIMe?
- **Function description:**
  Set the sweep time for the specified channel frequency sweep.
  <time> represents time, unit S. The range is: 1ms ~ 500s
  <n>: Channel number, n takes the value 1 or 2.
- **Return format:**
  The query returns the scanning time of the specified channel during frequency sweep, and returns the data in scientific notation.
- **Example:**

:CHANnel1:SWEEP:TIMe 2	Set the scanning time of channel 1 to 2S.
:CHANnel1:SWEEP:TIMe?	The query returns 2e+0

:CHANnel<n>:SWEep:TRIGger

- **Command format:**
  :CHANnel<n>:SWEep:TRIGger
- **Function description:**
  Trigger the frequency sweep output of the specified channel. This parameter is only valid when the trigger mode is set to manual trigger.
- **Example:**

:CHANnel1:SWEep:TRIGger	Trigger a frequency sweep signal output

burst

:CHANnel<n>:BURSt:TYPe

- **Command format:**
  :CHANnel<n>:BURSt:TYPe {NCYC|GATe|INFinit}
  :CHANnel<n>:BURSt:TYPe?
- **Function description:**
  Set the burst type of the specified channel, which is N period, gated, and infinite.
  <n>: Channel number, n takes the value 1 or 2.
- **Return format:**
  The query returns the burst type of the specified channel.
- **Example:**

:CHANnel1:BURSt:TYPe NCYC	Set channel one channel N period burst
:CHANnel1:BURSt:TYPe?	Query returns NCYC

:CHANnel<n>: BURSt:SOURce

- Command format:
  :CHANnel<n>:BURSt:SOURce {INTernal| EXTernal| MANual}
  :CHANnel<n>:BURSt:SOURce?
- Function description:
  Set the burst source of the specified channel. This command is only valid for frequency sweep and burst functions.
  <n>: Channel number, n takes the value 1 or 2.
- Return format:
  The query returns the burst source of the specified channel.
- Example:

:CHANnel1:BURSt:SOURce INTernal	Set channel 1 internal trigger source
:CHANnel1:BURSt SOURce?	The query returns INTernal

:CHANnel<n>:BURSt:TIRgOut

- Command format:
  :CHANnel<n>:BURSt TIRgOut {{1 | ON} | {0 | OFF}}
  :CHANnel<n>:BURSt:TIRgOut?
- Function description:
  Set the burst output status of the specified channel. This command is only valid for frequency sweep and burst functions.
  <n>: Channel number, n takes the value 1 or 2.
- Return format:
  The query returns the burst output status of the specified channel.
- Example:

:CHANnel1:BURSt:TIRgOut ON	Set channel 1 rising edge trigger output mode
:CHANnel1:BURSt:TIRgOut?	The query returns 1

:CHANnel<n>:BURSt:PERiod

- Command format:
  :CHANnel<n>:BURSt:PERiod <period >
  :CHANnel<n>:BURSt:PERiod?
- Function description:
  Set the burst period of the specified channel.
  <period> represents time, unit S.
  <n>: Channel number, n takes the value 1 or 2.
- Return format:
  The query returns the burst period of the specified channel and returns the data in scientific notation.
- Example:

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

:CHANnel<n>:BURSt:PHASe

- Command format:
  :CHANnel<n>:BURSt:PHASe <phase>
  :CHANnel<n>:BURSt:PHASe?
- Function description:
  Set the burst phase of the specified channel.

<phase> represents phase, unit °. The range is: 0 ~ 360
<n>: Channel number, n takes the value 1 or 2.

- **Return format:**
  The query returns the burst phase of the specified channel and returns the data in scientific notation.
- **Example:**

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

:CHANnel<n>:BURSt:CYCLes

- **Command format:**
  :CHANnel<n>:BURSt:CYCLes <cycles>
  :CHANnel<n>:BURSt:CYCLes?
- **Function description:**
  Set the number of burst cycles for the specified channel.
  <cycles> represents the number of cycles, integer data.
  <n>: Channel number, n takes the value 1 or 2.
- **Return format:**
  The query returns the number of burst cycles of the specified channel.
- **Example:**

:CHANnel1:BURSt:CYCLes 2	Set the number of burst cycles of the specified channel to 2
:CHANnel1:BURSt:CYCLes?	The query returns 2

:CHANnel<n>:BURSt:TIRgedge

- **Command format:**
  :CHANnel<n>:BURSt:TIRgedge { RISe|FALL}
  :CHANnel<n>:BURSt:TIRgedge?
- **Function description:**
  Set the trigger edge of the burst mode of the specified channel. This command is only valid for frequency sweep and burst functions.
  <n>: Channel number, n takes the value 1 or 2.
- **Return format:**
  The query returns the trigger output mode of the specified channel.
- **Example:**

:CHANnel1:BURSt:TIRgedge RISe	Set channel 1 rising edge trigger output mode
:CHANnel1:BURSt:TIRgedge?	The query returns RISe

:CHANnel<n>:BURSt:GATe:POLarity

- **Command format:**
  :CHANnel<n>:BURSt:GATe:POLarity {POSitive|NEGative}
  :CHANnel<n>:BURSt:GATe:POLarity?
- **Function description:**
  Set the specified channel gate burst polarity, which are positive polarity and negative polarity respectively.
  <n>: Channel number, n takes the value 1 or 2.
- **Return format:**
  The query returns the specified channel gate burst polarity.
- **Example:**

:CHANnel1:BURSt:GATe:POLarity POSitive	Set channel one gate burst polarity to positive polarity
:CHANnel1:BURSt:GATe:POLarity?	Query returns POSitive

:CHANnel<n>:BURSt:TRIGger

- **Command format:**

:CHANnel<n>:BURSt:TRIGger

- Function description:
  Trigger the specified burst output. This parameter is only valid when the trigger mode is set to manual trigger.
- Example:
  :CHANnel1:BURSt:TRIGger Trigger a burst signal output

:CHANnel<n>:BURSt:DELAy

- Command format:
  :CHANnel<n>:BURSt:DELAy <delay>
  :CHANnel<n>:BURSt:DELAy?
- Function description:
  Set the burst delay of the specified channel in seconds.
  <delay> represents the delay seconds.
  <n>: Channel number, n takes the value 1 or 2.
- Return format:
  The query returns the burst delay of the specified channel burst, in scientific notation, in seconds.
- Example:
  :CHANnel1:BURSt:DELAy 0.001 Set the burst delay number of the specified channel to 1ms
  :CHANnel1:BURSt:DELAy? The query returns 1e-03

WARB command

Instructions for writing arbitrary waveform files, including basic arbitrary waveform and modulated arbitrary waveform writing configurations.

:WARB<n>:MODulate

- Command format:
  :WARB<n>:MODulate <arb file>
- Function description:
  Used to write modulated arbitrary waveforms. Send this command first, and then send the arbitrary waveform file data to the signal source.
  <arb file> represents the name of the arbitrary waveform file, and only supports bsv file format.
- Example:
  :WARB1:MODulate "test.bsv" Write channel one modulated arbitrary waveform file

:WARB<n>:CARRier

- Command format:
  :WARB<n>:CARRier <arb file>
- Function description:
  Used to write the fundamental arbitrary waveform. Send this command first, and then send the arbitrary waveform file data to the signal source.
  <arb file> represents the name of the arbitrary waveform file, and only supports bsv file format.
- Example:
  :WARB1:CARRier "test.bsv" Write channel one fundamental waveform arbitrary waveform file

DISPlay command

Used to display relevant information of the signal source.

:DISPlay:DATA?

- **Command format:**
  :DISPlay:DATA?
- **Function description:**
  Used to query the image data of the current device screen.
- **Return format:**
  Query returnsimage data,The returned data conforms to the IEEE 488.2 # format of binary data.
- **Example:**

:DISPlay:DATA?	Query returns image data Data format: #800012345+bitmap 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

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

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

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

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

VISA Programming Example

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

VC++ Example

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

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

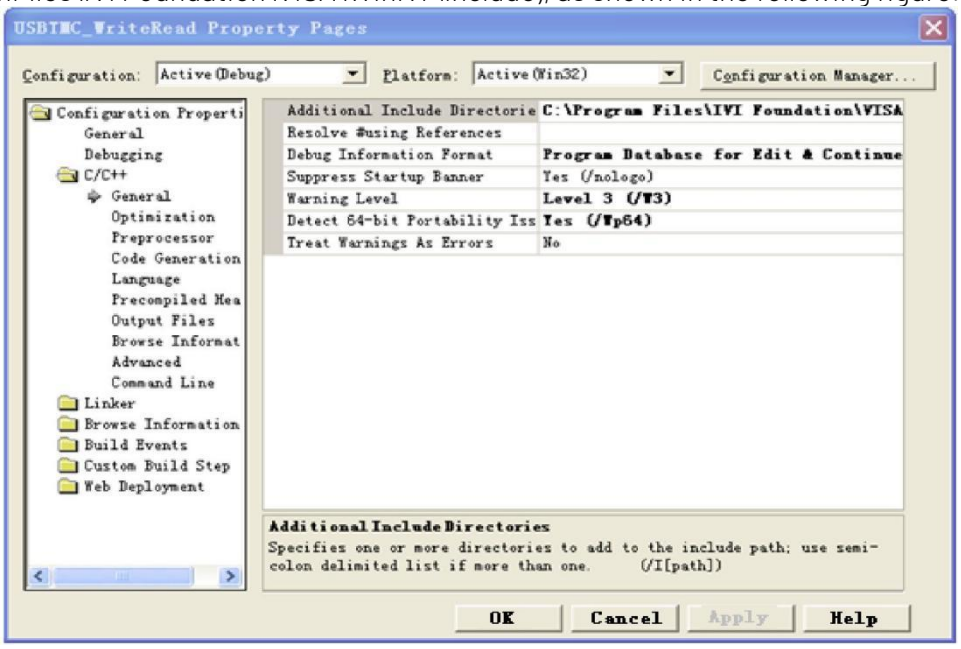
a) Static library:

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

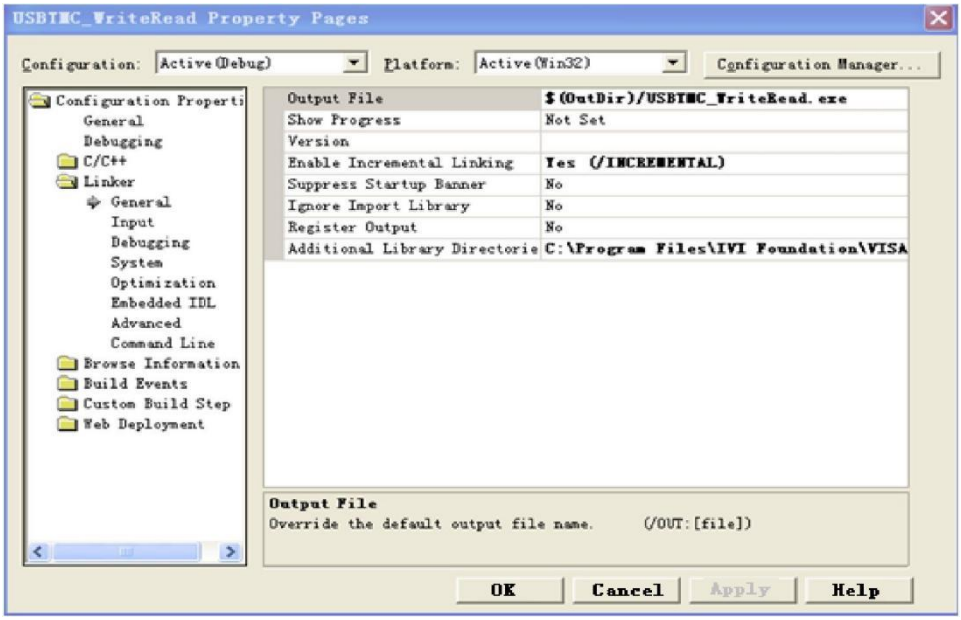
```
#include "visa.h"
#pragma comment(lib,"visa32.lib")
```

b) Dynamic library:

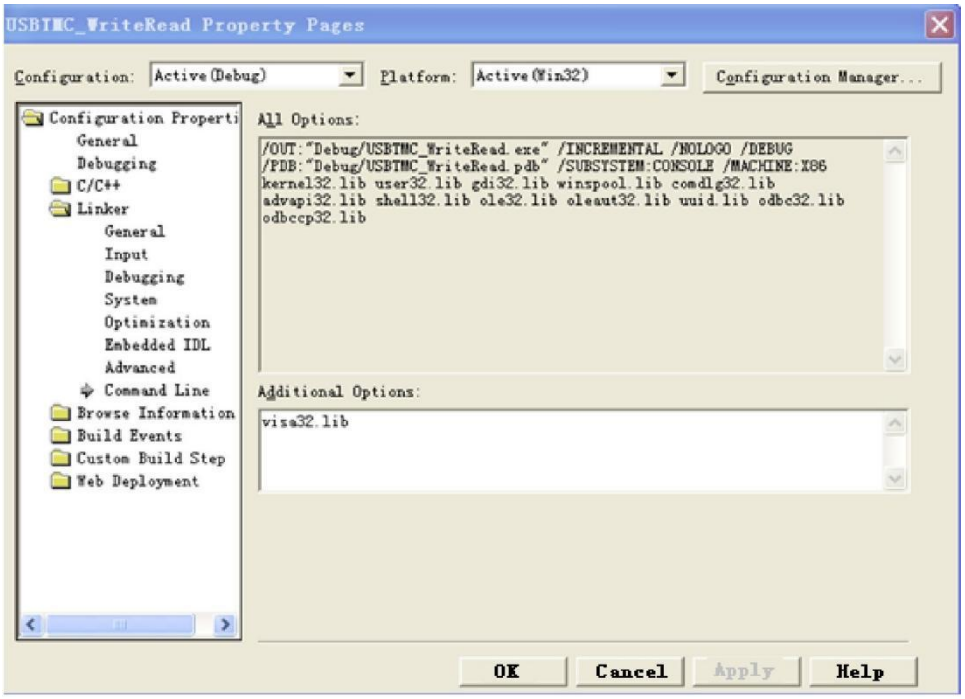
Press "project>>properties", select "c/c++---General" in attribute dialog on the left side, set the value of "Additional Include Directories" as the installment path of NI-VIS (such as C:\ProgramFiles\IVI Foundation\VISA\WinNT\include), as shown in the following figure.



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



Select "Linker-General" in attribute dialog on the left side, set the value of "Additional Library Directories" to visa32.lib, as shown in the following figure.



Add file visa.h in projectname.cpp file,

#include <visa.h>

1. Source code

a) USBTMC Example

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

```
* Try to Read a Response With viScanf
* Close the VISA Session*/
ViSession defaultRM;
ViSession instr;
ViUInt32 numInstrs;
ViFindList findList;
ViStatus status;
char instrResourceString[VI_FIND_BUFLEN];
unsigned char buffer[100];
int i;
status = viOpenDefaultRM(&defaultRM);
if (status < VI_SUCCESS)
{
        printf("Could not open a session to the VISA Resource Manager!\n");
        return status;
}
  /*Find all the USB TMC VISA resources in our system and store the number of resources in the system in
  numInstrs.*/
status = viFindRsrc(defaultRM, "USB?*INSTR", &findList, &numInstrs, instrResourceString);
if (status<VI_SUCCESS)
{
        printf("An error occurred while finding resources. \nPress Enter to continue.");
        fflush(stdin);
        getchar();
        viClose(defaultRM);
        return status;
}
/** Now we will open VISA sessions to all USB TMC instruments.
*       We must use the handle from viOpenDefaultRM and we must
*       also use a string that indicates which instrument to open. This
*       is called the instrument descriptor. The format for this string
*       can be found in the function panel by right clicking on the
*       descriptor parameter. After opening a session to the
*       device, we will get a handle to the instrument which we
*       will use in later VISA functions. The AccessMode and Timeout
*       parameters in this function are reserved for future
*       functionality. These two parameters are given the value VI_NULL. */
for (i = 0; i < int(numInstrs); i++)
{
        if (i > 0)
        {
                viFindNext(findList, instrResourceString);
        }
        status = viOpen(defaultRM, instrResourceString, VI_NULL, VI_NULL, &instr);
        if (status < VI_SUCCESS)
        {
                printf("Cannot open a session to the device %d. \n", i + 1);
                continue;
        }
        /** At this point we now have a session open to the USB TMC instrument.
        *We will now use the viPrintf function to send the device the string "*IDN?\n",
        *asking for the device's identification. */
        char * cmmand = "*IDN?\n";
        status = viPrintf(instr, cmmand);
        if (status < VI_SUCCESS)
        {
                printf("Error writing to the device %d. \n", i + 1);
                status = viClose(instr);
                continue;
        }
        /** Now we will attempt to read back a response from the device to
        *the identification query that was sent. We will use the viScanf
        *function to acquire the data.
        *After the data has been read the response is displayed. */
        status = viScanf(instr, "%t", buffer);
        if (status < VI_SUCCESS)
        {
```

```
                printf("Error reading a response from the device %d. \n", i + 1);
        }
        else
        {
                printf("\nDevice %d: %s\n", i + 1, buffer);
        }
        status = viClose(instr);
    }
    /*Now we will close the session to the instrument using viClose. This operation frees all
      system resources.*/
    status = viClose(defaultRM);
    printf("Press Enter to exit.");
    fflush(stdin);
    getchar();
    return 0;
}

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

b) TCP/IP Example

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

int _tmain(int argc, _TCHAR* argv[ ])
{
```

```
    printf("Please input IP address:");
    char ip[256];
    fflush(stdin);
    gets(ip);
    tcp_ip_test(ip);
    return 0;
```

C# Example

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

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

a) USBTMC Example

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

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

b) TCP/IP Example

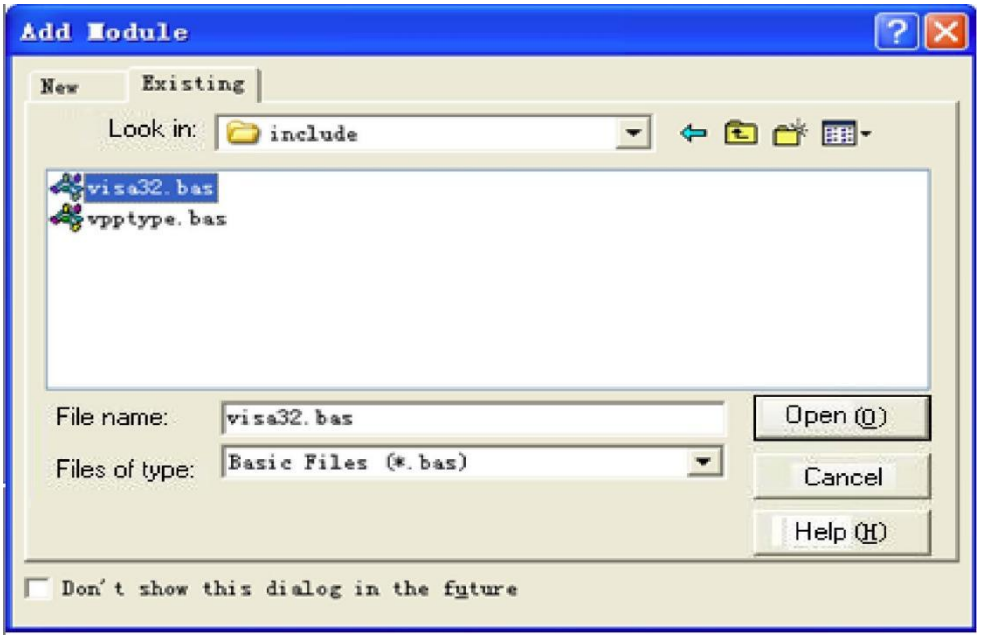
```
class Program
{
  void tcp_ip_test(string ip)
  {
    using (var rmSession = new ResourceManager())
    {
      try
      {
        var resource = string.Format("TCPIP0::{0}::inst0::INSTR", ip);
        var mbSession = (MessageBasedSession)rmSession.Open(resource);
        mbSession.RawIO.Write("*IDN?\n");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

```
            System.Console.WriteLine(ex.Message);
        }
    }
}

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

VB Example

- Environment: Window 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 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, in file "include" of NI-VISA installment path to find file visa32.bas and add this file, as



shown in the following figure.

3. Source code

a) USBTMC Example

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

Const MAX_CNT = 200
Dim defaultRM AsLong
Dim instrsesn AsLong
Dim numInstrs AsLong
Dim findList AsLong
```

```
Dim retCount AsLong
Dim status AsLong
Dim instrResourceString AsString *VI_FIND_BUFLEN
Dim Buffer AsString * MAX_CNT
Dim i AsInteger

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

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

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

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

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

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

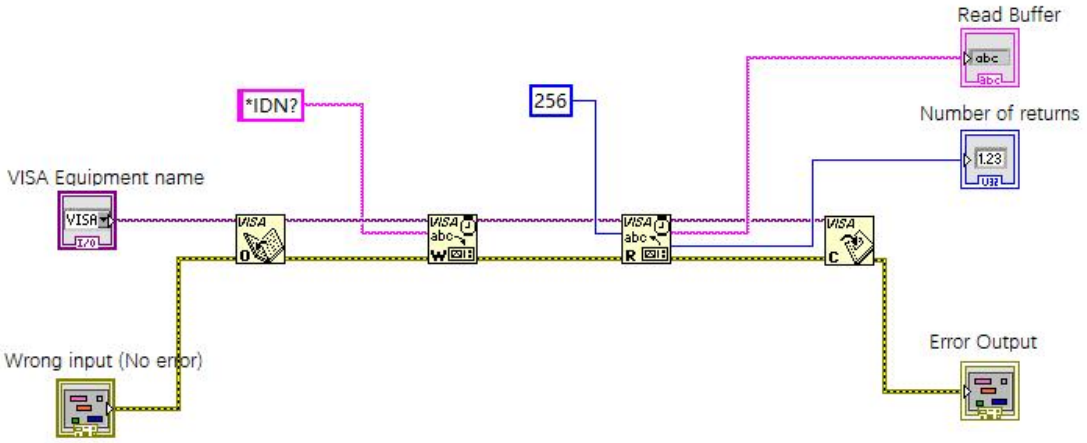
b) TCP/IP Example

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

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

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

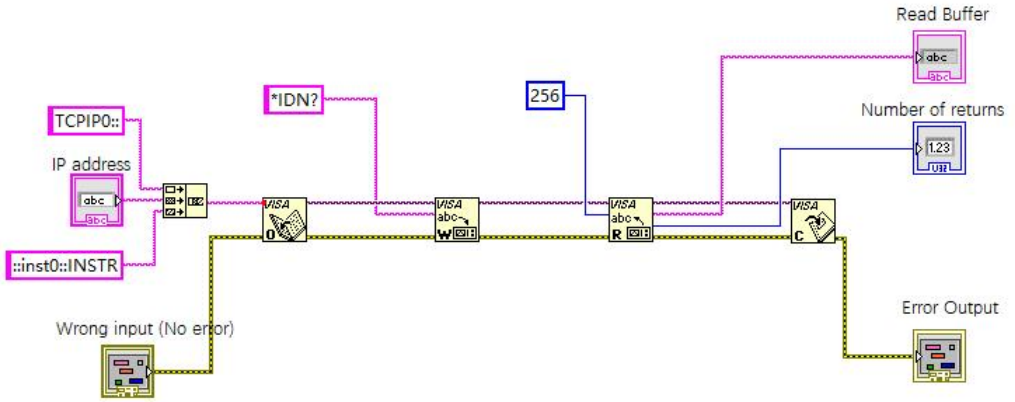
LabVIEW Example

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

1. Open LabVIEW software and create a VI file.
2. Add control, press the front panel interface, select and add VISA resource name, error input, error output and partial indetifier on control flow diagram.
3. Open diagram, press VISA resource name and then select and add function VISA Write, VISA Read, VISA Open and VISA Close on pop-out menu.
4. VI open a VISA session of USBTMC device and wrote *IDN? command and read back the response value. When all communication is complete, VI will close the VISA session. 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 LabVIEW to asynchronous IO by default. Right click on the node and select "Synchronous I/O Mode>>Synchronous" from shortcut menu to enable synchronous writing or reading of data, as shown in the following figure.



MATLAB Example

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

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

a) USBTMC Example

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

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

%Open the VISA object created
fopen(vu);

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

%Request the data

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

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

end
```

b) TCP/IP Example

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

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

%Open the VISA object created

fopen(vt);

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

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

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

end
```

Python Example

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

1. Install python, and then turn on Python script compiling software, create an empty test.py file.
2. Use pip install PyVISA instruction to install PyVISA, if it cannot install, please refer to this link (https://pyvisa.readthedocs.io/en/latest/).
3. Source code

a) USBTMC Example

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

b) TCP/TP Example

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

Programming Application Example

Configuring Sine Wave

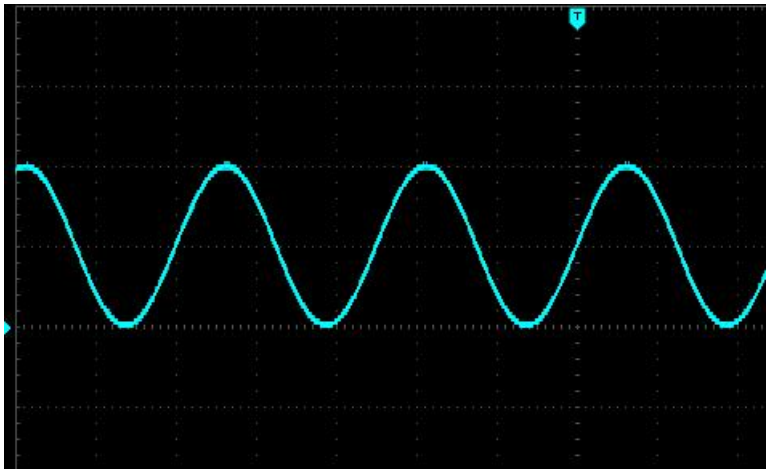
This section is to introduce how to configure the sine wave function.

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 command, high voltage value and low voltage value can replace :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

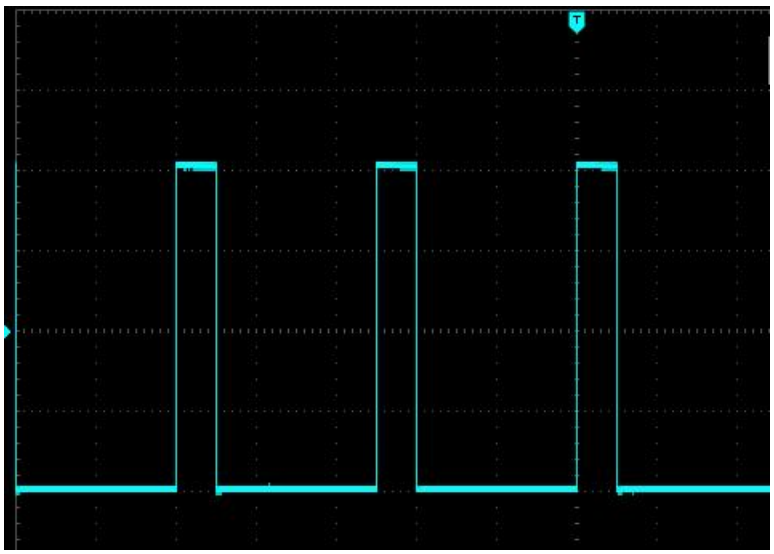
This section is to introduce how to configure the square wave.

Explanation

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

Example

The following wave can set by SCPI command.



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

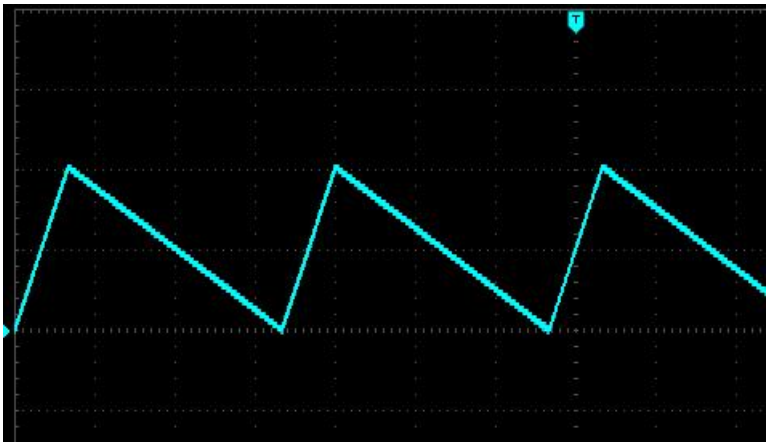
This section is to introduce how to configure the sawtooth wave.

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 value and low voltage value to set its amplitude and deviation.

Example

The following wave can set by SCPI command, high voltage value and low voltage value can replace : 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 Impulse Wave

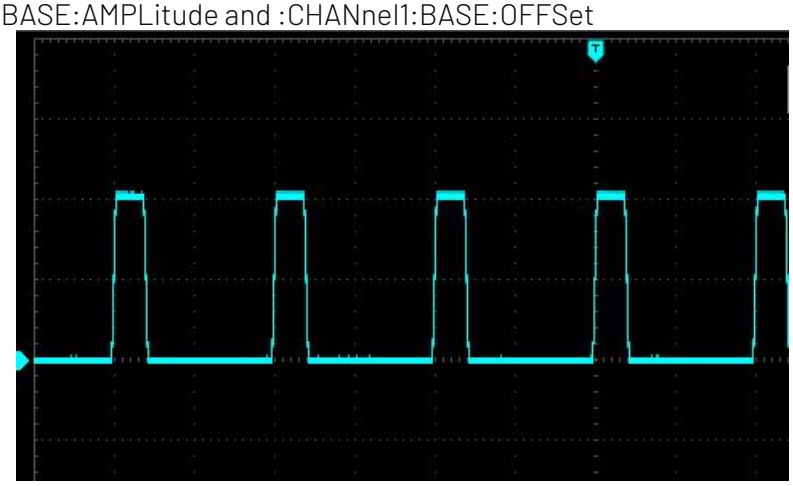
This section is to introduce how to configure the impulse wave.

Explanation

An impulse wave has an amplitude, an offset, and a phase relative to a synchronous pulse. It also adds slope of edge and duty cycle (or pulse width). It can use high voltage value and low voltage value to set its amplitude and deviation.

Example

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



The following command can generate the impulse 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 the arbitrary wave.

Explanation

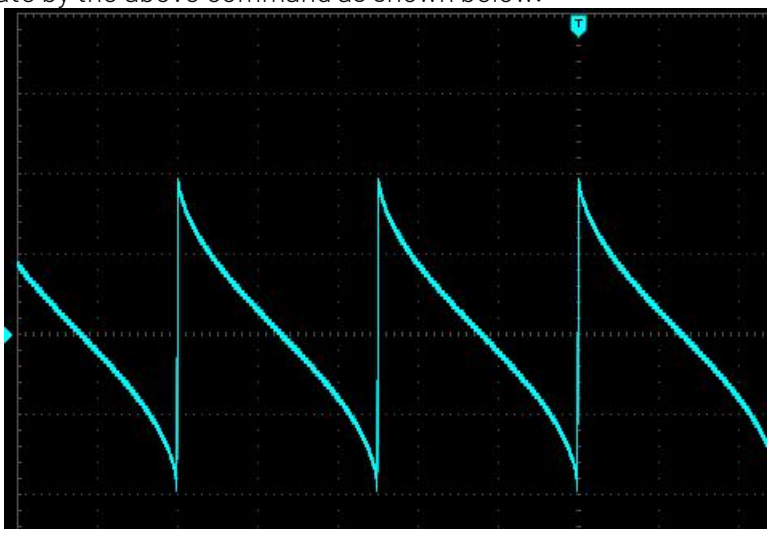
A harmonic wave has frequency, amplitude, offset and phase. It also adds mode and wave file.

Example

The following code 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 below.



Appendix 1: <key> list

key command keyword	Function description	LED light
CH1	Channel one button	√
CH2	Channel 2 button	√
RIGHT	Arrow key right	
LEFT	Arrow key left	
OK	Enter	
SYMBOL	numeric key symbols	
NUM0	Numeric key 0	
NUMBER1	Numeric key 1	
NUM2	Numpad 2	
NUM3	Numpad 3	
NUM4	Numpad 4	
NUM5	Numpad 5	
NUMBER 6	Numpad 6	
NUM7	Numpad 7	
NUM8	Numpad 8	
NUM9	Numpad 9	
DOT	Numeric key decimal point	
F1	Horizontal function key 1	
F2	Horizontal function key 2	
F3	Horizontal function key 3	
F4	Horizontal function keys 4	
F5	Horizontal function keys 5	
F6	Horizontal function keys 6	
MENU	Menu button	
VF1	Vertical function key 1	
VF2	Vertical function key 2	
VF3	Vertical function key 3	
VF4	Vertical function key 4	
UTILity	system	
TRIGGER	Trigger key	√
UP	Knob rotates clockwise	
DOWN	Knob rotates counterclockwise	