USB Dual Channel Waveform Generator

Programming Manual

Part Number: 72-3555

Newark.com/exclusive-brands
Farnell.com/exclusive-brands
Element14.com/exclusive-brands

Page <1>   25/06/25   V1.0

Introduction to SCPI

SCPI (Standard Commands for Programmable Instruments) is a standardized programming language formed based on current IEEE 488.1 and IEEE 488.2 standards and following multiple standards such as the rule ofv floating point arithmetic in IEEE754 standard, 7-digit numbering symbols (equivalent to ASCII programming) of information exchange in ISO646, and more. The SCPI command format, symbols, parameters and abbreviations are introduced in this section.

Command format

SCPI command (a tree structure) includes multiple sub-systems, with each constructed by a root key and one or several hierarchic keys. Command line starts with colon ":" typically; keys are separated by colon ":", key is followed by optional parameter setting. Command key and first parameter are separated by space.

The character string of command must ends with a <Line Feed> (<NL>) character. Adding a question mark "?" behind a command line means function query generally.

Symbols

The four symbols below are not the contents of SCPI command, which are not sent along with command, but used as an auxiliary instruction of the command parameters.

Curly Brackets { }

Curly brackets include multiple optional parameters typically, one of which must be selected when sending commands.
Example: DISPlay:GRID:MODE { FULL | GRID | CROSS | NONE} command.

Vertical Bar |

Vertical line is used to separate multiple parameter options, one of which must be selected when sending commands.
Example: DISPlay:GRID:MODE { FULL | GRID | CROSS | NONE} command.

Square Brackets [ ]

The content (command key) included in square brackets is negligible. If the parameter is neglected, it will be set as default value by the instrument.
Example: For MEASure:NDUTy? [<source>] command, [<source>] represents the current channel.

Angle Brackets < >

The parameter in angle brackets must be substituted by a valid value.
Example: Send DISPlay:GRID:BRIGhtness   <count> in the form of DISPlay:GRID:BRIGhtness 3

Parameters

There are 5 types of parameters contained in the command introduced by this Manual, including Boolean, integer, float, discrete, and ASCII character string.

Boolean

Take the parameter value as "ON" (1) or "OFF" (0). Example: :SYSTem:LOCK {{1 | ON} | {0 | OFF}}.

Integer

Unless otherwise specified, any integer within the valid value can be taken as the parameter. Note: Do not set the parameter as decimal format, otherwise, abnormality can occur. Example: Any integer in the range of 0 to 100 can be taken as the parameter < count > in :DISPlay:GRID:BRIGhtness <count> command.

Float

Unless otherwise specified, any value in the valid scope can be taken as the parameter.
Example: For CH1, the value taken as the parameter <offset> in CHANnel1:OFFSet <offset> command is float.

Discrete

Only several specified values or characters can be taken as the parameters. Example: The parameters of :DISPlay:GRID:MODE { FULL | GRID | CROSS | NONE} command can be taken as FULL, GRID, CROSS and NONE only.

Newark.com/exclusive-brands
Farnell.com/exclusive-brands
Element14.com/exclusive-brands

Page <2>   25/06/25   V1.0

ASCII Character String

The parameter of character string include all ASCII character sets actually. The character string must starts or ends with paired quotation marks; single quotes or double quotes can be used. The separator of quotation marks can also be used as a part of character string. Type twice and make sure no any character is added in the middle. Taking IP setting as example: SYST:COMM:LAN:IPAD "192.168.1.10".

Abbreviations

All commands can identify capital and small letters. Capital or small letters in whole can be used. If abbreviation is to be used, then all capital letters in the command format must be typed.

Data Return

Data return is categorized into single data return and mass data return. Single data returns the corresponding parameter type, and float return is represented by scientific notation. For the part before e, the data of three digit places behind decimal are retained; while for e, data of three digit places are retained. The mass data return must meet the data of character string of IEEE 488.2 # format, and the format is:'#'+ Number of character positions occupied by length[fixed as a character] + ASCII value of valid data length + Valid data + End mark['\n']. For example, #3123xxxxxxxxxxxxxxxxxxx\n represents a format of valid mass data return with 123 bytes, and the digit "3" in it means that "123" occupies 3 character positions.

Details about SCPI

IEEE488.2 Generic Command

*IDN?
» Command format:
  *IDN?
» Functional description:
  Used to query manufacturer name, product model, serial number, and software version number.
» Return format:
  Manufacturer name, product model, serial number, and software version number separated by period.
  Note: The model number returned shall be consistent with the information of nameplate.
» Example:
  72-3555
*RST
» Command format:
  *RST
» Functional description:
  Used to restore factory setting and clear all error information and send the buffer of the receiving array.

SYSTem Command

Used to perform most basic operations on the signal source, primarily including full keyboard locking and system data setting.
:SYSTem:LOCK
» Command format:
  :SYSTem:LOCK {{1 | ON} | {0 | OFF}}
  :SYSTem:LOCK?
» Functional description:
  Used to lock or unlock full keys of keyboard and touch input.
» Return format:
  Query returning the lock state of full keyboard. 0 denotes unlocked; 1 denotes locked.

Newark.com/exclusive-brands
Farnell.com/exclusive-brands
Element14.com/exclusive-brands

™

Page <3>   25/06/25   V1.0

» Example:
  :SYSTem:LOCK ON   Full keyboard locked
  :SYSTem:LOCK OFF   Full keyboard unlocked
  :SYSTem:LOCK?   Query returning 1, denoting locked

:SYSTem:CONFigure
» Command format:
  :SYSTem:CONFigure <file>
  :SYSTem:CONFigure?
» Functional description:
  Used to read and write the configuration file. Send the command first and then the configuration file data to signal source.
  <File> represents the configuration file.
» Return format:
  Query returning the current configuration file data of signal source.
» Example:*
  :SYSTem:CONFigure   Write configuration file data to signal source and load it
  :SYSTem:CONFigure?   Query returning the binary stream of current configuration file data of signal source

:SYSTem:LANGuage
» Command format:
  :SYSTem:LANGuage {ENGLish|CHINese|TRAChinese |KOREan}
  :SYSTem:LANGuage?
» Functional description:
  Control system language.
» Return format:
  Query returning system language.
» Example:
  :SYSTem:LANGuage ENGLish   Set English as system language
  :SYSTem:LANGuage?   Query returning ENGLish

:SYSTem:CLKSource
» Command format:
  :SYSTem:CLKSource { INTernal|EXTernal }
  :SYSTem:CLKSource?
» Functional description:
  Control the selection of clock source of system.
» Return format:
  Query returning clock source of system.
» Example:
  :SYSTem:CLKSource INTernal   Set the clock source of system as internal clock source
  :SYSTem:CLKSource?   Query returning INTernal

:SYSTem:CLKOut
» Command format:
  :SYSTem:CLKOut { {1 | ON} | {0 | OFF} }
  :SYSTem:CLKOut?
» Functional description:
  Control the selection of turning on or off the output of system clock.
» Return format:
  Query returning the state of the output of system clock.

Newark.com/exclusive-brands
Farnell.com/exclusive-brands
Element14.com/exclusive-brands

™

Page <4>   25/06/25   V1.0

» Example:
    :SYSTem:CLKOut ON   Set the output of system clock to be turned on
    :SYSTem:CLKOut?   Query returning 1

:SYSTem:BEEP
» Command format:
    :SYSTem:BEEP {{1 | ON} | {0 | OFF}}
    :SYSTem:BEEP?
» Functional description:
    Control the switch of the buzzer of system.
» Return format:
    Query returning the state of the switch of buzzer.
» Example:
    :SYSTem:BEEP ON   Turn on buzzer
    :SYSTem:BEEP?   Query returning 1

:SYSTem:NUMBer:FORMat
» Command format:
    :SYSTem:NUMBer:FORMat {COMMa|SPACe|NONe}
    :SYSTem:NUMBER:FORMat?
» Function description:
    Control the separator of the format of system number.
» Return format
    Query returning the separator of the format of system number.
» Example:
    :SYSTem:NUMBer:FORMat NONe Set the format of system number to None
    :SYSTem:NUMBer:FORMat?   Query returning NONe

:SYSTem:BRIGhtness
» Command format:
    :SYSTem:BRIGhtness { 10|30|50|70|90|100}
    :SYSTem:BRIGhtness?
» Function description:
    Control the backlight brightness of system.
» Return format:
    Query returning the backlight brightness of system.
» Example:
    :SYSTem:BRIGhtness 30   Set the backlight brightness of system as 30%
    :SYSTem:BRIGhtness?   Query returning 30

:SYSTem:CYMometer:FREQuency?
» Command format:
    :SYSTem:CYMometer:FREQuency?
» Function description:
    Acquire the current frequency measured by cymometer.
» Return format:
    Query returning the acquisition of the current frequency (unit: Hz) measured by cymometer. Returning data by scientific notation.
» Example:
    :SYSTem:CYMometer:FREQuency?   Query returning 2e+3

Newark.com/exclusive-brands
Farnell.com/exclusive-brands
Element14.com/exclusive-brands

Page <5>

25/06/25   V1.0

:SYSTem:CYMometer:PERiod?
- » Command format:
  :SYSTem:CYMometer:PERiod?
- » Function description:
  Acquire the current period measured by cymometer.
- » Return format:
  Query returning the acquisition of the current period (unit: s) measured by cymometer. Returning data by scientific notation.
- » Example:
  :SYSTem:CYMometer:PERiod? Query returning 2e-3

:SYSTem:CYMometer:DUTY?
- » Command format:
  :SYSTem:CYMometer:DUTY?
- » Functional description:
  Acquire the current duty cycle measured by cymometer.
- » Return format: Query returning the acquisition of the current duty cycle measured by cymometer. Returning data by scientific notation.
- » Example:
  :SYSTem:CYMometer:DUTY? Query returning 2e+1, representing the duty cycle is 20%

CHANnel Command

Used to set related functions of the channels of signal source.

:CHANnel<n>:MODE
- » Command format:
  :CHANnel<n>:MODE {CONTinue | MODulation| SWEep| BURSt }
  :CHANnel<n>:MODE?
- » Functional description:
  Set the modes of designated channel signal to CONTinue, MODulation, SWEep, and BURSt respectively.
  <n>: Channel number. The value of n is taken as 1 and 2.
- » Return format:
  Query returning the mode of designated channel signal.
- » Example:
  :CHANnel1:MODE MODulation Set the signal of channel 1 to modulation output
  :CHANnel1:MODE? Query returning MODulation

:CHANnel<n>:OUTPut
- » Command format:
  :CHANnel<n>:OUTPut {{1 | ON} | {0 | OFF}}
  :CHANnel<n>:OUTPut?
- » Functional description:
  Set turning on/off the output of designated channel.
  <n>: Channel number. The value of n is taken as 1 and 2.
- » Return format:
  Query returning the output state of designated channel. 0 denotes OFF; 1 denotes ON.
- » Example:
  :CHANnel1:OUTPut ON Set turning on the output of channel 1
  :CHANnel1:OUTPut? Query returning 1

Newark.com/exclusive-brands
Farnell.com/exclusive-brands
Element14.com/exclusive-brands

Page <6>   25/06/25 V1.0

:CHANnel<n>:INVersion

» Command format:
:CHANnel<n>:INVersion {{1 | ON} | {0 | OFF}}
:CHANnel<n>:INVersion?

» Functional description:
Set turning on/off the inversion of designated channel.
<n>: Channel number. The value of n is taken as 1 and 2.

» Return format:
Query returning the inversion state of designated channel. 0 denotes OFF; 1 denotes ON.

» Example:
:CHANnel1:INVersion ON Set turning on the inversion output of channel 1
:CHANnel1:INVersion? Query returning 1

:CHANnel<n>:OUTPut:SYNC

» Command format:
:CHANnel<n>:OUTPut:SYNC {{1 | SYNC_CH1} | {2 | SYNC_CH2}|{0 | SYNC_OFF}}
:CHANnel<n>:OUTPut:SYNC?

» Functional description:
Set the channel to synchronous output state.
Note: There is only one synchronous output port with the equipment, and synchronous output can be turned on for one channel only.
<n>: Channel number. The value of n is taken as 1 and 2.

» Return format:
Query returning the synchronous output state of designated channel. 0 denotes OFF; 1 represents channel 1 is turned on; 2 represents channel 2 is turned on.

» Example:
:CHANnel1:OUTPut:SYNC SYNC_CH1 Set turning on synchronous output of channel 1
:CHANnel1:OUTPut:SYNC? Query returning 1

:CHANnel<n>:LIMit:ENABle

» Command format:
:CHANnel<n>:LIMit:ENABle {{1 | ON} | {0 | OFF}}
:CHANnel<n>:LIMit:ENABle?

» Functional description:
Set the amplitude limiting switch of designated channel.
<n>: Channel number. The value of n is taken as 1 and 2.

» Return format:
Query returning the amplitude limiting state of designated channel.

» Example:
:CHANnel1:LIMit:ENABle ON   Set turning on the amplitude limit of channel 1
:CHANnel1:LIMit:ENABle?   Query returning 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 limit of designated channel.
<voltage> represents voltage, and the unit is the designated unit of current channel.
<n>: Channel number. The value of n is taken as 1 and 2.

Newark.com/exclusive-brands
Farnell.com/exclusive-brands
Element14.com/exclusive-brands

™
TENMA

Page <7>   25/06/25   V1.0

» Return format:
  Query returning the lower limit of the amplitude limit of designated channel. Returning data by scientific notation.

» Example:
  :CHANnel1:LIMit:LOWer 2   Set the lower limit of the amplitude limit of channel 1 to 2V
  :CHANnel1:LIMit:LOWer?   Query returning 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 limit of designated channel.
  <voltage> represents voltage, and the unit is the designated unit of current channel.
  <n>: Channel number. The value of n is taken as 1 and 2.

» Return format:
  Query returning the upper limit of the amplitude limit of designated channel. Returning data by scientific notation.

» Example:
  :CHANnel1:LIMit:UPPer 2   Set the upper limit of the amplitude limit of channel 1 to 2V
  :CHANnel1:LIMit:UPPer?   Query returning 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 output amplitude of designated channel.
  <n>: Channel number. The value of n is taken as 1 and 2.

» Return format:
  Query returning the unit of the output amplitude of designated channel.

» Example:
  :CHANnel1:AMPLitude:UNIT VPP   Set the unit of the output amplitude of channel 1 as 1
  :CHANnel1:AMPLitude:UNIT?   Query returning VPP

:CHANnel<n>:LOAD

» Command format:
  :CHANnel<n>:LOAD <resistance>
  :CHANnel<n>:LOAD?

» Functional description:
  Set the output load of designated channel.
  <resistance> represents the load resistance (unit: Ω)
  <n>: Channel number. The value of n is taken as 1 and 2.
  Note: The resistance range is 1~1000000, and 1000000 corresponds to high resistance.

» Return format:
  Query returning the load resistance of designated channel. Returning data by scientific notation.

» Example:
  :CHANnel1:LOAD 50   Set the output load of channel 1 to 50Ω
  :CHANnel1:LOAD?   Query returning 5e+1

Newark.com/exclusive-brands
Farnell.com/exclusive-brands
Element14.com/exclusive-brands

TENMA™

Page <8>   25/06/25 V1.0

:CHANnel<n>:TRIGger:SOURce
» Command format:
  :CHANnel<n>:TRIGger:SOURce {INTernal| EXTernal| MANual}
  :CHANnel<n>:TRIGger:SOURce?
» Functional description:
  Set the trigger source of designated channel. The command is valid for sweep frequency and burst function only.
  <n>: Channel number. The value of n is taken as 1 and 2.
» Return format:
  Query returning the trigger source of designated channel.
» Example:
  :CHANnel1:TRIGger:SOURce INTernal Set the internal trigger source of channel 1
  :CHANnel1:TRIGger:SOURce?   Query returning INTernal

:CHANnel<n>:TRIGger:OUTPut
» Command format:
  :CHANnel<n>:TRIGger:OUTPut {{1 | ON} | {0 | OFF}}
  :CHANnel<n>:TRIGger:OUTPut?
» Functional description:
  Set the trigger output state of designated channel. The command is valid for sweep frequency and burst function only.
  <n>: Channel number. The value of n is taken as 1 and 2.
» Return format:
  Query returning the trigger output state of designated channel.
» Example:
  :CHANnel1:TRIGger:OUTPut ON Set turning on the trigger output of channel 1
  :CHANnel1:TRIGger:OUTPut?   Query returning 1

:CHANnel<n>:TRIGger:OUTEdge
» Command format:
  :CHANnel<n>:TRIGger:OUTEdge { RISe|FALL}
  :CHANnel<n>:TRIGger:OUTEdge?
» Functional description:
  Set designated channel to trigger output edge. The command is valid for sweep frequency and burst function only.
  <n>: Channel number. The value of n is taken as 1 and 2.
» Return format:
  Query returning the triggering of output edge of designated channel.
» Example:
  :CHANnel1:TRIGger:OUTEdge RISe Set the rising edge of channel 1 to trigger the output mode
  :CHANnel1:TRIGger:OUTEdge?   Query returning RISe

Continuation
:CHANnel<n>:BASE:WAVe
» Command format:
  :CHANnel<n>:BASE:WAVe { SINe | SQUare | PULSe | RAMP | ARB | NOISe | DC }
  :CHANnel<n>:BASE:WAVe?
» Functional description:
  Set the types of fundamental wave of designated channel. The types include sine wave, square wave, pulse wave, ramp wave, arbitrary wave, noise, and direct current.
  <n>: Channel number. The value of n is taken as 1 and 2.
» Return format:
  Query returning the type of fundamental wave of designated channel.

25/06/25 V1.0

» Example:
    :CHANnel1:BASE:WAVe SINe   Set the basic type of channel 1 as sine wave
    :CHANnel1:BASE:WAVe?   Query returning SINe

:CHANnel<n>:BASE:FREQuency
» Command format:
    :CHANnel<n>:BASE:FREQuency {<freq>}
    :CHANnel<n>:BASE:FREQuency?
» Functional description:
    Set the output frequency (unit: Hz) of designated channel.
    <freq> represents frequency value, and the unit is Hz. (1e-6s ~ Maximum frequency that the current waveform allows)
    <n>: Channel number. The value of n is taken as 1 and 2.
» Return format:
    Query returning the output frequency of designated channel. Returning data by scientific notation.
    The unit of the return value is Hz.
» Example:
    :CHANnel1:BASE:FREQuency 2000   Set the output frequency of channel 1 as 2kHz
    :CHANnel1:BASE:FREQuency?   Query returning 2e+

:CHANnel<n>:BASE:PERiod
» Command format:
    :CHANnel<n>:BASE:PERiod { <period>}
    :CHANnel<n>:BASE:PERiod?
» Functional description:
    Set the output period of designated channel.
    <period> represents the period (unit: S)
    For sine wave, the range is (4ns ~ 1000s)
    <n>: Channel number. The value of n is taken as 1 and 2.
» Return format:

» Example:
    Query returning the upper limit of the amplitude limit of designated channel. Returning data by scientific notation. The unit of the return value is S.
    :CHANnel1:BASE:PERiod 0.002   Set the output period of channel 1 to 2ms
    :CHANnel1:BASE:PERiod?   Query returning 2e-3

:CHANnel<n>:BASE:PHASe
» Command format:
    :CHANnel<n>:BASE:PHASe { <phase>}
    :CHANnel<n>:BASE:PHASe?
» Functional description:
    Set the output phase of designated channel.
    <phase> represents phase (unit: °), with range at -360~360.
    <n>: Channel number. The value of n is taken as 1 and 2.
» Return format:
Query returning the output phase of designated channel.
» Example:7
    :CHANnel1:BASE:PHASe 20   Set the output phase of channel 1 as 20°
    :CHANnel1:BASE:PHASe?   Query returning 2e+01

Newark.com/exclusive-brands
Farnell.com/exclusive-brands
Element14.com/exclusive-brands

Page <10>   25/06/25   V1.0

:CHANnel<n>:BASE:AMPLitude

» Command format:
  :CHANnel<n>:BASE:AMPLitude { <amp>}
  :CHANnel<n>:BASE:AMPLitude?

» Functional description:
  Set the output amplitude of designated channel.
  <amp> represents voltage (the unit is the designated unit of current channel). 1mVpp ~ maximum value under current load.
  If the current unit is VPP, then, the maximum value under current load = current load * 20/(50 + current load)
  <n>: Channel number. The value of n is taken as 1 and 2.

» Return format:
  Query returning the output amplitude of designated channel. Returning data by scientific notation.

» Example:
  :CHANnel1:BASE:AMPLitude 2   Set the output amplitude of channel 1 as 2V
  :CHANnel1:BASE:AMPLitude?   Query returning 2e+0

» Note:
  If the value is negative when setting amplitude in units of dBm and Vrms, then certain losses of precision can occur at the return value of scpi.
  For example: CHAN1:BASE:AMPL -0.45dBm Return: -4.500003e-01

:CHANnel<n>:BASE:OFFSet

» Command format:
  :CHANnel<n>:BASE:OFFSet { <voltage>}
  :CHANnel<n>:BASE:OFFSet?

» Functional description:
  Set the output DC offset of designated channel.
  <Voltage> represent 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/2;
  AC minimum is 2mVpp. 0 is taken as the value in DC mode;
  <n>: Channel number. The value of n is taken as 1 and 2.

» Return format:
  Query returning the output DC offset of designated channel. Returning data by scientific notation.

» Example:
  :CHANnel1:BASE:OFFSet 2   Set the output DC offset of channel 1 as 2V
  :CHANnel1:BASE:OFFSet?   Query returning 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 signal output of designated channel.
  <Voltage> represents voltage. The unit is the designated unit of current channel.
  <n>: Channel number. The value of n is taken as 1 and 2.

» Return format:
  Query returning the high value of the signal output of designated channel. Returning data by scientific notation.

» Example:
  :CHANnel1:BASE:HIGH 2   Set the high value of the signal output of channel 1 to 2V
  :CHANnel1:BASE:HIGH?   Query returning 2e+0

Newark.com/exclusive-brands
Farnell.com/exclusive-brands
Element14.com/exclusive-brands

™

Page <11>   25/06/25   V1.0

:CHANnel<n>:BASE:LOW

» Command format:
  :CHANnel<n>:BASE:LOW { <voltage>}
  :CHANnel<n>:BASE:LOW?

» Functional description:
  Set the low value of the signal output of designated channel.
  <Voltage> represents voltage. The unit is the designated unit of current channel.
  <n>: Channel number. The value of n is taken as 1 and 2.

» Return format:
  Query returning the low value of the signal output of designated channel. Returning data by scientific notation.

» Example:
  :CHANnel1:BASE:LOW 2   Set the low value of the signal output of channel 1 to 2V
  :CHANnel1:BASE:LOW?   Query returning 2e+0

:CHANnel<n>:BASE:PWIDth

» Command format:
  :CHANnel<n>:BASE:PWIDth { <pulsewidth>}
  :CHANnel<n>:BASE:PWIDth?

» Functional description:
  Set the pulse width of the signal output of designated channel.
  < pulsewidth > represents pulse width (unit: s).
  <n>: Channel number. The value of n is taken as 1 and 2.

» Return format:
  Query returning the pulse width of the signal of designated channel. Returning data by scientific notation.

» Example:
  :CHANnel1:BASE:PWIDth 0.0004   Set the pulse width of the signal output of channel 1 to 400us
  :CHANnel1:BASE:PWIDth?   Query returning 4.000000e-04

:CHANnel<n>:BASE:DUTY

» Command format:
  :CHANnel<n>:BASE:DUTY { <duty>}
  :CHANnel<n>:BASE:DUTY?

» Functional description:
  Set the duty cycle of the signal output of designated channel.
  <Duty> represents duty cycle (unit: %). The range is 0~100.
  <n>: Channel number. The value of n is taken as 1 and 2.

» Return format:
  Query returning the duty cycle of the signal output of designated channel. Returning data by scientific notation.

» Example:
  :CHANnel1:BASE:DUTY 20   Set the duty cycle of the signal output of channel 1 to 20%
  :CHANnel1:BASE:DUTY?   Query returning 2e+1

:CHANnel<n>:BASE:ARB

» Command format:
  CHANnel<n>:BASE:ARB <source>,<filename>
  CHANnel<n>:BASE:ARB?

» Functional description:
  Set the designated channel to load arbitrary wave data of a certain file under the arbitrary wave source of fundamental wave.
  <n>: Channel number. The value of n is taken as 1 and 2.
  <Source>: {INTernal|EXTernal|USER}, there are three types including internal, external, and self-defined.
  <Filename>: The file name of arbitrary waveform.

Newark.com/exclusive-brands
Farnell.com/exclusive-brands
Element14.com/exclusive-brands

™

Page <12>   25/06/25   V1.0

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

:CHANnel<n>:RAMP:SYMMetry
» Command format:
  :CHANnel<n>:RAMP:SYMMetry { < symmetry >}
  :CHANnel<n>:RAMP:SYMMetry?
» Functional description:
  Set the symmetry of the ramp signal output of designated channel.
  < Symmetry > represents symmetry (unit: %). The range is 0~100.
  <n>: Channel number. The value of n is taken as 1 and 2.
» Return format:
  Query returning the symmetry of the ramp signal output of designated channel.
» Example:
  :CHANnel1:RAMP:SYMMetry 20   Set the symmetry of the ramp signal of channel 1 to 20%
  :CHANnel1:RAMP:SYMMetry?   Query returning 20

:CHANnel<n>:PULSe:RISe
» Command format:
  :CHANnel<n>:PULSe:RISe {<width>}
  :CHANnel<n>:PULSe:RISe?
» Functional description:
  Set the pulse width of the rising edge of signal pulse of designated channel.
  <Width> represents pulse width (unit: S).
  <n>: Channel number. The value of n is taken as 1 and 2.
» Return format:
  Query returning the pulse width of the rising edge of signal pulse of designated channel. Returning data by scientific notation.
» Example:
  :CHANnel1:PULSe:RISe 0.002   Set the pulse width of the rising edge of signal of channel 1 to 2ms
» :CHANnel1:PULSe:RISe?   Query returning 2e-3

:CHANnel<n>:PULSe:FALL
» Command format:
  :CHANnel<n>:PULSe:FALL {<width>}
  :CHANnel<n>:PULSe:FALL?
» Functional description:
  Set the pulse width of the falling edge of signal pulse of designated channel.7
  <Width> represents pulse width (unit: S).
  <n>: Channel number. The value of n is taken as 1 and 2.
» Return format:
  Query returning the pulse width of the falling edge of signal pulse of designated channel. Returning data by scientific notation.
» Example
  :CHANnel1:PULSe:FALL 0.002   Set the pulse width of the falling edge of signal of channel 1 to 2ms
  :CHANnel1:PULSe:FALL?   Query returning 2e-3

Modulation
:CHANnel<n>:MODulate:TYPe
» Command format:
  CHANnel<n>:MODulate:TYPe <type>
  CHANnel<n>:MODulate:TYPe?

Newark.com/exclusive-brands
Farnell.com/exclusive-brands
Element14.com/exclusive-brands

™

Page <13>   25/06/25   V1.0

» Functional description:
Set the modulation type of the signal of designated channel
<Type>: { AM|FM|PM|ASK|FSK|PSK| PWM }
Amplitude modulation, frequency modulation, phase modulation, amplitude-shift keying, frequency-shift keying, phase-shift keying, and pulse width modulation.
<n>: Channel number. The value of n is taken as 1 and 2.

» Return format:
Query returning the modulation type of the signal of designated channel.

» Example:
CHANnel1:MODulate:TYPe AM   Set the signal of channel 1 as AM
CHANnel1:MODulate:TYPe?   Query returning 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 types of modulation wave of the signal of designated channel. The types include sine wave, square wave, upper triangular, lower triangular, arbitrary wave, and noise.
<n>: Channel number. The value of n is taken as 1 and 2.

» Return format:
Query returning the type of modulation wave of the signal of designated channel.

» Example:
:CHANnel1:MODulate:WAVe SINe   Set the type of modulation wave of the signal of channel 1 to sine wave.
:CHANnel1:MODulate:WAVe?   Query returning SINe

:CHANnel<n>:MODulate:SOURce

» Command format:
:CHANnel<n>:MODulate:SOURce { INTernal|EXTernal }
:CHANnel<n>:MODulate:SOURce?

» Functional description:
Set the modulation sources of designated channel. There are two types including internal and external.
<n>: Channel number. The value of n is taken as 1 and 2.

» Return format:
Query returning the modulation source of designated channel.

» Example:
:CHANnel1:MODulate:SOURce INTernal   Set the modulation source of channel 1 as internal.
:CHANnel1:MODulate:SOURce?   Query returning INTernal

:CHANnel<n>:MODulate:FREQuency

» Command format:
:CHANnel<n>:MODulate:FREQuency {<freq>}
:CHANnel<n>:MODulate:FREQuency?

» Functional description:
Set the modulation frequency of the signal of designated channel.
<freq> represents frequency (unit: Hz).
<n>: Channel number. The value of n is taken as 1 and 2.

» Return format:
Query returning the modulation frequency of the signal of designated channel. Returning sampling by scientific notation.

Newark.com/exclusive-brands
Farnell.com/exclusive-brands
Element14.com/exclusive-brands

![TENMA™]

Page <14>   25/06/25   V1.0

» Example:
    :CHANnel1:MODulate:FREQuency 2000   Set the modulation frequency of the signal of channel 1 to 2KHz
    :CHANnel1:MODulate:FREQuency?   Query returning 2e+3

:CHANnel<n>:MODulate:DEPTh
» Command format:
    :CHANnel<n>:MODulate:DEPTh { <depth>}
    :CHANnel<n>:MODulate:DEPTh?
» Functional description:
    Set the modulation depth of designated channel.
    <depth> represents modulation depth (unit: %). 0% ~ 100%, and the depth of AM is 0% ~ 120%.
    <n>: Channel number. The value of n is taken as 1 and 2.
» Return format:
    Query returning the modulation depth of designated channel.
» Example:
    :CHANnel1:MODulate:DEPTh 50   Set the modulation depth of channel 1 to 50%
    :CHANnel1:MODulate:DEPTh?   Query returning 5e+1

:CHANnel<n>:MODulate:RATio
» Command format:
    :CHANnel<n>:MODulate:RATio <ratio>
    :CHANnel<n>:MODulate:RATio?
» Functional description:
    Set the modulation ratio of designated channel. The command is only valid for the modulation type with ratio function.
    < ratio > represents ratio (unit: Hz).
    <n>: Channel number. The value of n is taken as 1 and 2.
» Return format:
    Query returning the modulation ratio of designated channel. Returning data by scientific notation.
» Example:
    :CHANnel1:MODulate:RATio 100   Set the ratio of channel 1 to 100Hz
    :CHANnel1:MODulate:RATio?   Query returning 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 of designated channel.
    <freq> represents frequency deviation (unit: Hz). 0Hz ~ Current fundamental frequency
    <n>: Channel number. The value of n is taken as 1 and 2.
» Return format:
    Query returning the frequency deviation of designated channel. Returning data by scientific notation.
» Example:
    :CHANnel1:FM:FREQuency:DEV 2000   Set the frequency offset of channel 1 to 2KHz
    :CHANnel1:FM:FREQuency:DEV?   Query returning 2e+3

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

Newark.com/exclusive-brands
Farnell.com/exclusive-brands
Element14.com/exclusive-brands

Page <15>   25/06/25 V1.0

» Functional description:
  Set the output phase deviation of designated channel.
  < phase > represents phase deviation (unit:°), with range at 0~360.
  <n>: Channel number. The value of n is taken as 1 and 2.
» Return format:
  Query returning the output phase offset of designated channel.
» Example:
  :CHANnel1:PM:PHASe:DEV 30 Set the phase offset of channel 1 to 30°
  :CHANnel1:PM:PHASe:DEV?   Query returning 3e+01

:CHANnel<n>:PWM:DUTY
» Command format:
  :CHANnel<n>:PWM:DUTY { <duty>}
  :CHANnel<n>:PWM:DUTY?
» Functional description:
  Set the designated channel to output the duty cycle under pulse width modulation.
  < duty > represents duty cycle (unit: %), with range at 0~100.
  <n>: Channel number. The value of n is taken as 1 and 2.
» Return format:
  Query returning the pulse width deviation under pulse width modulation of designated channel. Returning data by scientific notation.
» Example:
  :CHANnel1:PWM:DUTY 10   Set the duty cycle of channel 1 to 10%
  :CHANnel1:PWM:DUTY?   Query returning 1e+1

:CHANnel<n>:FSK:FREQuency<m>
» Command format:
  :CHANnel<n>:FSK:FREQuency{ <freq>}
  :CHANnel<n>:FSK:FREQuency?
» Functional description:
  Set the designated channel to output the frequency hopping of MFSK. The command is valid only when the modulation mode is designated in advance.
  < freq > represents frequency (unit: Hz).
  <n>: Channel number. The value of n is taken as 1 and 2.
» Return format:
  Query returning the output of frequency hopping of designated channel. Returning data by scientific notation.
» Example:
  :CHANnel1:FSK:FREQ 2000   Set channel 1 to output 2KHz frequency hopping
  :CHANnel1:FSK:FREQ?   Query returning 2e+3

:CHANnel<n>:PSK:PHASe<m>
» Command format:
  :CHANnel<n>:PSK:PHASe<m> { < phase >}
  :CHANnel<n>:PSK:PHASe<m>?
» Functional description:
  Set the designated channel to output the phase value of multiple-phase shift keying. The command is valid only when the modulation mode is designated in advance.
  < phase> represents phase (unit:°), with range at -360°~+360°.
  <n>: Channel number. The value of n is taken as 1, 2, 3, and 4.
  <m>: Phase No.. The value of PSK is taken as 1.

Newark.com/exclusive-brands
Farnell.com/exclusive-brands
Element14.com/exclusive-brands

Page <16>   25/06/25   V1.0

» Return format:
  Query returning the phase value of PSK of designated channel. Returning data by scientific notation.
» Example:
  :CHANnel1:PSK:PHAS1 90   Set the output phase of channel 1 to 90°
  :CHANnel1:PSK:PHAS1?   Query returning 9e+1

Sweep

:CHANnel<n>:SWEep:TYPe
» Command format:
  :CHANnel<n>:SWEep:TYPe { LINe|LOG }
  :CHANnel<n>:SWEep:TYPe?
» Functional description:
  Set the sweep modes of designated channel. The modes include linear sweep and logarithmic sweep.
  <n>: Channel number. The value of n is taken as 1 and 2.
» Return format:
  Query returning the sweep mode of designated channel.
» Example:
  :CHANnel1:SWEep:TYPe LINe   Set channel 1 to linear sweep mode
  :CHANnel1:SWEep:TYPe?   Query returning 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 of the sweep of designated channel.
  < freq > represents frequency (unit: Hz).
  <n>: Channel number. The value of n is taken as 1 and 2.
» Return format:
  Query returning the start frequency of the sweep of designated channel. Returning data by scientific notation.
» Example:
  :CHANnel1:SWE:FREQ:STAR 2000   Set the start frequency of the output sweep of channel 1 to 2KHz
  :CHANnel1:SWE:FREQ:STAR?   Query returning 2e+3

:CHANnel<n>:SWEep:FREQuency:STOP
» Command format:
  :CHANnel<n>:SWEep:FREQuency:STOP <freq>
  :CHANnel<n>:SWEep:FREQuency:STOP?
» Functional description:
  Set the stop frequency of the sweep of designated channel.
  < freq > represents frequency (unit: Hz).
  <n>: Channel number. The value of n is taken as 1 and 2.
» Return format:
  Query returning the stop frequency of the output sweep of designated channel. Returning data by scientific notation.
» Example:
  :CHANnel1:SWE:FREQ:STOP 2000   Set the stop frequency of the output sweep of channel 1 to 2KHz
» :CHANnel1:SWE:FREQ:STOP?   Query returning 2e+3
»

Newark.com/exclusive-brands
Farnell.com/exclusive-brands
Element14.com/exclusive-brands

™

Page <17>   25/06/25   V1.0

:CHANnel<n>:SWEep:TIMe
» Command format:
  :CHANnel<n>:SWEEP:TIMe <time>
  :CHANnel<n>:SWEEP:TIMe?
» Functional description:
  Set the sweep time of designated channel when sweeping frequency.
  < time > represents time (unit: S), with range at 1ms ~ 500s.
  <n>: Channel number. The value of n is taken as 1 and 2.
» Return format:
  Query returning the sweep time of designated channel when sweeping frequency sweep. Returning data by scientific notation.
» Example:
  :CHANnel1:SWEEP:TIMe 2   Set the sweep time of channel 1 when sweeping frequency to 2S
  :CHANnel1:SWEEP:TIMe?   Query returning 2e+0

:CHANnel<n>:SWEep:TRIGger
» Command format:
  :CHANnel<n>:SWEep:TRIGger
» Functional description:
  Trigger the sweep output of designated channel. The parameter is valid only when set as manual trigger in triggering mode.
» Example:
  :CHANnel1:SWEep:TRIGger   Trigger the output of sweep signal once

Burst
:CHANnel<n>:BURSt:TYPe
» Command format:
  :CHANnel<n>:BURSt:TYPe {NCYC|GATe|INFinit}
  :CHANnel<n>:BURSt:TYPe?
» Functional description:
  Set the burst types of designated channel. The types include N cycle, gate, and infinite.
  <n>: Channel number. The value of n is taken as 1 and 2.
» Return format:
  Query returning the burst type of designated channel.
» Example:
  :CHANnel1:BURSt:TYPe NCYC   Set the burst type of channel 1 as N cycle*
  :CHANnel1:BURSt:TYPe?   Query returning 2e+0

:CHANnel<n>: BURSt:SOURce
» Command format:
  CHANnel<n>:BURSt:SOURce {INTernal| EXTernal| MANual}
  CHANnel<n>:BURSt:SOURce?
» Functional description:
  Set the burst source of designated source. The command is valid for sweep and burst function only.
  <n>: Channel number. The value of n is taken as 1 and 2.
» Return format:
  Query returning the burst source of designated channel.
» Example:
  CHANnel1:BURSt:SOURce INTernal   Set channel 1 as internal burst source
  CHANnel1:BURSt SOURce?   Query returning INTernal

Newark.com/exclusive-brands
Farnell.com/exclusive-brands
Element14.com/exclusive-brands

™

Page <18>   25/06/25   V1.0

:CHANnel<n>:BURSt:TIRgOut
- » Command format:
  :CHANnel<n>:BURSt TIRgOut {{1 | ON} | {0 | OFF}}
  :CHANnel<n>:BURSt:TIRgOut?
- » Functional description:
  Set the burst output state of designated channel. The command is valid for sweep and burst function only.
  <n>: Channel number. The value of n is taken as 1 and 2.
- » Return format:
  Query returning the burst output state of designated channel.
- » Example:
  :CHANnel1:BURSt:TIRgOut ON   Set the burst output mode of rising edge of channel 1
- » :CHANnel1:BURSt:TIRgOut?   Query returning 1

:CHANnel<n>:BURSt:PERiod
- » Command format:
  :CHANnel<n>:BURSt:PERiod <period >
  :CHANnel<n>:BURSt:PERiod?
- » Functional description:
  Set the burst period of designated channel.
  < Period > represents time (unit: S).
  <n>: Channel number. The value of n is taken as 1 and 2.
- » Return format:
  Query returning the burst period of designated channel. Returning data by scientific notation.
- » Example:
  :CHANnel1:BURSt:PERiod 5ms   Set the burst period of channel 1 to 5ms
  :CHANnel1:BURSt:PERiod?   Query returning 5e-3

:CHANnel<n>:BURSt:PHASe
- » Command format:
  :CHANnel<n>:BURSt:PHASe <phase>
  :CHANnel<n>:BURSt:PHASe?
- » Functional description:
  Set the burst phase of designated channel.
  < phase > represents phase (unit:°), with range at 0 ~ 360.
  <n>: Channel number. The value of n is taken as 1 and 2.
- » Return format:
  Query returning the burst phase of designated channel. Returning data by scientific notation.
- » Example:
  :CHANnel1:BURSt:PHASe 18   Set the burst phase of channel 1 to 18°
  :CHANnel1:BURSt:PHASe?   Query returning 1.8e+1

:CHANnel<n>:BURSt:CYCLes
- » Command format:
  :CHANnel<n>:BURSt:CYCLes <cycles>
  :CHANnel<n>:BURSt:CYCLes?
- » Functional description:
  Set the burst cycle of designated channel.
  < cycles > represents cycles. An integer data.
  <n>: Channel number. The value of n is taken as 1 and 2.

Newark.com/exclusive-brands
Farnell.com/exclusive-brands
Element14.com/exclusive-brands

Page <19>   25/06/25   V1.0

» Return format:
   Query returning the burst cycle of designated channel.
» Example:
   :CHANnel1:BURSt:CYCLes 2   Set the burst cycle of designated channel to 2
   :CHANnel1:BURSt:CYCLes?   Query returning 2

:CHANnel<n>:BURSt:TIRgedge
» Command format:
   :CHANnel<n>:BURSt:TIRgedge { RISe|FALL}
   :CHANnel<n>:BURSt:TIRgedge?
» Functional description:
   Set the trigger edge of designated channel in burst mode. The command is valid for sweep and burst function only.
   <n>: Channel number. The value of n is taken as 1 and 2.
» Return format:
   Query returning the trigger output mode of designated channel.
» Example:
   :CHANnel1:BURSt:TIRgedge RISe   Set the trigger output mode of rising edge of channel 1
   :CHANnel1:BURSt:TIRgedge?   Query returning RISe

:CHANnel<n>:BURSt:GATe:POLarity
» Command format:
   :CHANnel<n>:BURSt:GATe:POLarity {POSitive|NEGative}
   :CHANnel<n>:BURSt:GATe:POLarity?
» Functional description:
   Set the burst polarity of the gate of designated channel. The polarities include positive and negative.
   <n>: Channel number. The value of n is taken as 1 and 2.
» Return format:
   Query returning the burst polarity of the gate of designated channel.
» Example:
   :CHANnel1:BURSt:GATe:POLarity POSitive   Set the burst polarity of the gate of channel 1 to positive
   :CHANnel1:BURSt:GATe:POLarity?   Query returning POSitive

:CHANnel<n>:BURSt:TRIGger
» Command format:
   :CHANnel<n>:BURSt:TRIGger
» Functional description:
   Trigger the burst output of designated channel. The parameter is valid only when set as manual trigger in triggering mode.
» Example:
   :CHANnel1:BURSt:TRIGger   Trigger the output of burst signal once

WARB command

Used to write file command of arbitrary waveform, including writing configuration for basic arbitrary waveform and modulated arbitrary waveform.

:WARB<n>:MODulate
» Command format:
   :WARB<n>:MODulate <arb file>
» Functional description:
   Used to write modulated arbitrary waveform. Send the command and then send the file data of arbitrary waveform to signal source.

Newark.com/exclusive-brands
Farnell.com/exclusive-brands
Element14.com/exclusive-brands

™

Page <20>   25/06/25   V1.0

<arb file> represents the file name of arbitrary waveform. Supporting bsv file format only.

» Example:
  :WARB1:MODulate "test.bsv"   Write the file of modulated arbitrary waveform of channel 1

:WARB<n>:CARRier

» Command format:
  :WARB<n>:CARRier <arb file>

» Functional description:
  Used to write fundamental arbitrary waveform. Send the command and then send the file data of arbitrary waveform to signal source.
  <arb file> represents the file name of arbitrary waveform. Supporting bsv file format only.

» Example:
  :WARB1:CARRier "test.bsv"   Write the file of fundamental arbitrary waveform of channel 1

DISPlay command

Used for signal source to display related information.

:DISPlay:DATA?

» Command format:
  :DISPlay:DATA?

» Functional description:
  Used to query the current image data on a device screen.

» Return format:
  Query the returned image data. The returned data meets the binary data in IEEE 488.2 # format.

» Example:
  :DISPlay:DATA?   Query returning image data
                        Data format: #800012345+ bitmap data

Programming Instructions

This section describes some problems and solutions that occur during programming. If problems below occur, please handle with that according to the corresponding instructions.

Programming preparation

The programming preparation only applies to those programming that using developing tools "Visual Studio" and "LabVIEW" in Windows operating system.

Please confirm that the VISA library of NI has been installed to your computer (available at https://www.ni.com/en-ca/support/downloads/drivers/download.ni-visa.html),
the default installation path is C:\Program Files\IVI Foundation\VISA.

To establish communication with PC via the USB or LAN port of the instrument, please use USB cable to connect the USB DEVICE port at rear panel of the instrument to the USB port of PC, or use LAN data cable to the LAN port at rear panel of the instrument to the LAN port of PC.

Newark.com/exclusive-brands
Farnell.com/exclusive-brands
Element14.com/exclusive-brands

™

Page <21>   25/06/25   V1.0

VISA programming

Some programming examples are described in this section. Through these examples, you will learn how to use VISA and control the instrument by the command of programming manual. More applications can be developed through the examples below:

VC++

» Environment: Window system, Visual Studio.
» Description: Access to the instrument through USBTMC and TCP/IP, and query the device information by sending "*IDN?" command in NI-VISA.
» Steps
    1. Open Visual Studio software, and create a VC++ win32 console project.
    2. Set the project environments that recall NI-VISA library to static library and dynamic library respectively.

a) Static library

Find visa.h, visatype.h, and visa32.lib files in NI-VISA installation path, then copy them to the root path of VC++ project and add to the project. Add two lines of codes below in projectname.cpp file:
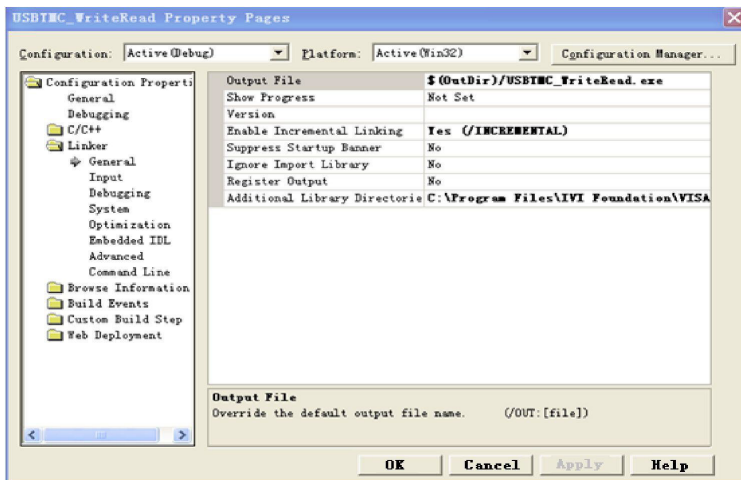
    #include "visa.h"
    #pragma comment(lib,"visa32.lib")

b) Dynamic library

Click "project>>properties", select "c/c++---General" on the left side of the properties dialog box, then set the value of the option "Additional Include Directories" as the installation path of NI-VISA, (For example, C:\ProgramFiles\IVI Foundation\ VISA\WinNT\include), As shown in the Figure below.
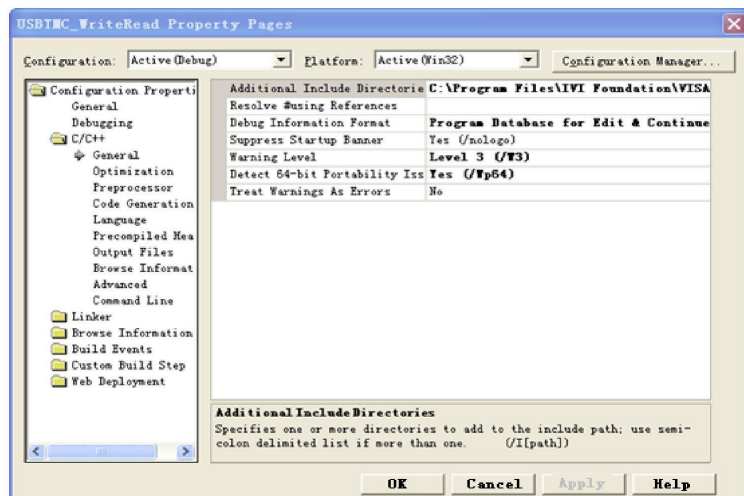
Select "Linker-General" on the left side of the properties dialog box, then set the value of the option "Additional Library Directories" as the installation path of NI-VISA, (For example, C:\Program Files\IVIFoundation\VISA\WinNT\include), As shown in the Figure below.

Newark.com/exclusive-brands
Farnell.com/exclusive-brands
Element14.com/exclusive-brands

Page <22>   25/06/25   V1.0

Select "Linker-Command Line" on the left side of the properties dialog box, then set the value of the option "Additional" as visa32.lib, As shown in the Figure below.
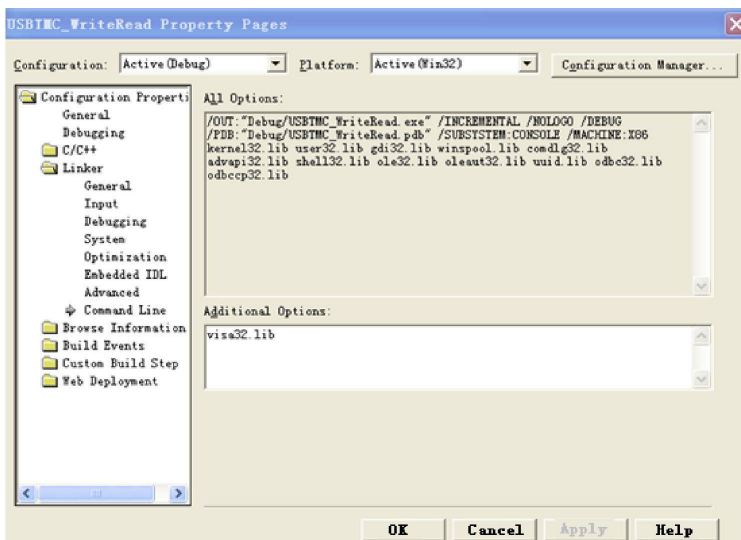
Add visa.h file on the projectname.cpp file:
#include <visa.h>

1. Source code
a) USBTMC
　　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

Newark.com/exclusive-brands
Farnell.com/exclusive-brands
Element14.com/exclusive-brands

Page <23>   25/06/25   V1.0

```
* 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";
```

Newark.com/exclusive-brands
Farnell.com/exclusive-brands
Element14.com/exclusive-brands

™

Page <24>   25/06/25   V1.0

```
            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
   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);
```

Newark.com/exclusive-brands
Farnell.com/exclusive-brands
Element14.com/exclusive-brands

™

Page <25>   25/06/25   V1.0

```
            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#

» Environment: Window system, Visual Studio.
» Description: Access to the instrument via USBTMC and TCP/IP, and query the device information by sending "*IDN?" command in NI-VISA.
» Steps
1. Open the Visual Studio software and create a C# console project.
2. Add the C# references (Ivi.Visa.dll"and "NationalInstruments.Visa.dll)of VISA.
3. Source code
a) USBTMC

```
    class Program
    {
        void usbtmc_test()
        {
            using (var rmSession = new ResourceManager())
            {
                var resources = rmSession.Find("USB?*INSTR");
                foreach (string s in resources)
```

Newark.com/exclusive-brands
Farnell.com/exclusive-brands
Element14.com/exclusive-brands

™

Page <26>   25/06/25   V1.0

```
        {
            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

```
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");
    }
}
```

Newark.com/exclusive-brands
Farnell.com/exclusive-brands
Element14.com/exclusive-brands

™

Page <27>   25/06/25   V1.0

VB

- » Environment: Window system, Microsoft Visual Basic 6.0
- » Description: Access to the instrument via USBTMC and TCP/IP, and query the device information by sending "*IDN?" command in NI-VISA.
- » Steps
1. Open the Visual Basic software and create a standard application project.
2. Set recalling the project environment of NI-VISA library: Click Existing tab of Project>>Add Existing Item, search the visa32.bas file in the "include" file folder on the NI-VISA installation path, then add the file. As shown in the Figure below.
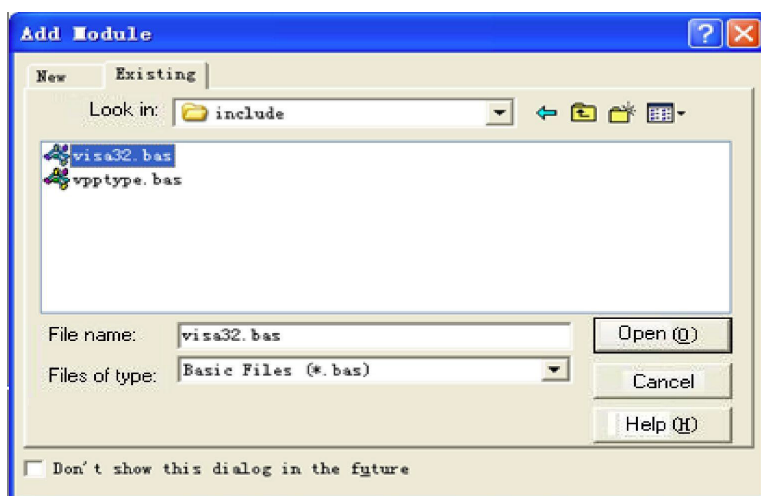
3. Source code

a) USBTMC

```
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
```

Newark.com/exclusive-brands
Farnell.com/exclusive-brands
Element14.com/exclusive-brands

Page <28>   25/06/25   V1.0

```
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
```

Newark.com/exclusive-brands
Farnell.com/exclusive-brands
Element14.com/exclusive-brands

TENMA™

Page <29>   25/06/25   V1.0

```
' 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

```
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
```

Newark.com/exclusive-brands
Farnell.com/exclusive-brands
Element14.com/exclusive-brands

TENMA™

Page <30>   25/06/25   V1.0

```
EndIf
    status = viClose(instrsesn)
    status = viClose(defaultRM)
    tcp_ip_test = 0
EndFunction
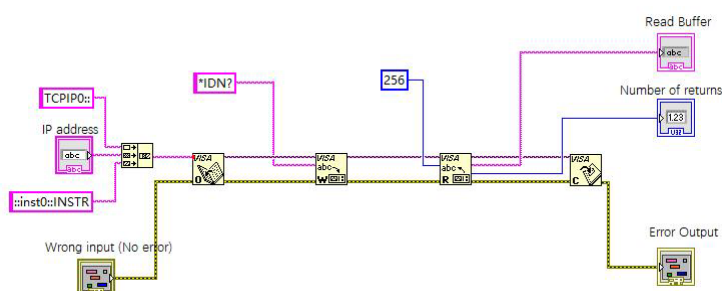```

## LabVIEW

» Environment: Window system, LabVIEW.

» Description: Access to the instrument via USBTMC and TCP/IP, and query the device information by sending "*IDN?" command in NI-VISA.

» Steps

» 1. Open the LabVIEW software and create a VI file.

» 2. Add a control, right-click the interface of front panel, then select and add VISA resource name, wrong input, wrong output and partial indicators.

» 3. Open the block diagram interface, right-click the VISA resource name, then select and add the functions below from the VISA panel of the popup menu: VISA Write, VISA Read, VISA Open, and VISA Close.

» 4. VI opens a VISA session of USBTMC device, and writes the *IDN? command for the device and reads back the response value. VI will close the VISA session after all communications are completed. As shown in the Figure below.
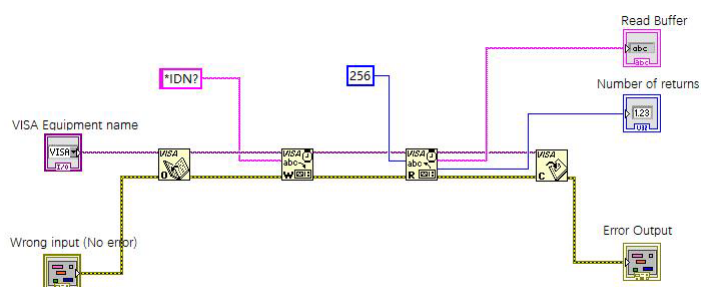


5. Communicating with the device via TCP/IP resembles USBTMC, but the VISA Write and VISA READ functions shall be set as synchronous I/O. The default setting of LabVIEW is asynchronous IO. Right-click the node, then select "Synchronous I/O Mode>>Synchronous" form the shortcut menu, so as to achieve writing or reading data synchronously. As shown in the Figure below.



## MATLAB

» Environment: Window system, MATLAB.

» Description: Access to the instrument via USBTMC and TCP/IP, and query the device information by sending "*IDN?" command in NI-VISA.

» Steps

Newark.com/exclusive-brands
Farnell.com/exclusive-brands
Element14.com/exclusive-brands

Page <31>                                        25/06/25   V1.0

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

2. Source code

a) USBTMC

```
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

```
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
```

Newark.com/exclusive-brands
Farnell.com/exclusive-brands
Element14.com/exclusive-brands

TENMA™

Page <32>   25/06/25   V1.0

Python

» Environment: Window system, Python3.8, PyVISA 1.11.0

» Description: Access to the instrument via USBTMC and TCP/IP, and query the device information by sending "*IDN?" command in NI-VISA.

» Steps

1. Install python, open the Python batch compiler, and create an empty test.py file.

2. Install PyVISA by using the pip install PyVISA command. If failure to install occurs, please access the link below for operating instructions: https://pyvisa.readthedocs.io/en/latest/

3. Source code

a) USBTMC

```
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

```
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?'))
```

Examples of programming

Configuring sine wave

This section introduces how to configure sine wave function.

Description

There are amplitude, offset, and phase relative to synchronous pulse for sine wave. The amplitude and offset of a sine wave can be set by using high and low voltage values.

Examples

The waveform below can be set by SCIP command series, and the high and low levels can be used to replace :CHANnel1:BASE:AMPLitude and :CHANnel1:BASE:OFFSet.

The commands below 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
```

Newark.com/exclusive-brands
Farnell.com/exclusive-brands
Element14.com/exclusive-brands

Page <33>   25/06/25   V1.0

Configuring square wave

Description

There are amplitude, offset, phase relative to synchronous pulse, duty cycle, and period for square wave. The amplitude and offset for square wave can be set by using high and low voltage values.

Example

The waveform below can be set by SCPI command series.
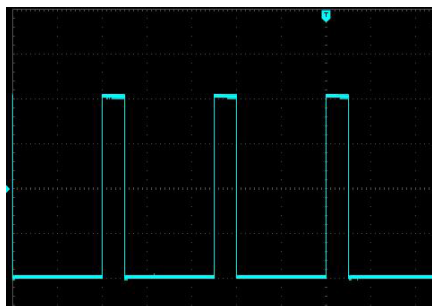
The commands below can generate the square wave as shown above.
:CHANnel1:MODe CONTinue
:CHANnel1:BASE:WAVe SQUare
:CHANnel1:BASE:FREQuency 40000
:CHANnel1:BASE:AMPLitude 2
:CHANnel1:BASE:OFFSet 0
:CHANnel1:BASE:PHAse 90
:CHANnel1:BASE:DUTY 20
:CHANnel1:OUTPut ON

Configuring sawtooth wave

Description

For sawtooth wave, there are amplitude, offset, phase relative to synchronous pulse, and symmetry used to create triangular waveform and other similar waveforms. The amplitude and offset of a sawtooth wave can be set by using high and low voltage values.

Examples

The waveform below can be set by SCPI command series, and the high and low levels can be used to replace :CHANnel1:BASE:AMPLitude and :CHANnel1:BASE:OFFSet.
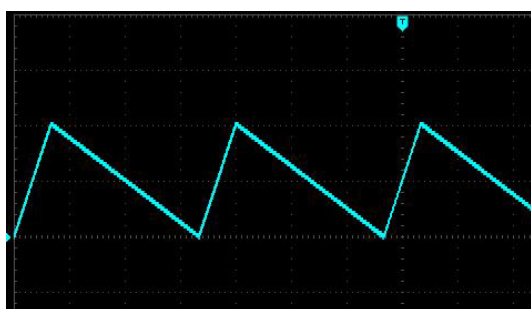
The commands below can generate the sawtooth wave as shown above.
:CHANnel1:MODe CONTinue
:CHANnel1:BASE:WAVe RAMP
:CHANnel1:BASE:FREQuency 30000

Newark.com/exclusive-brands
Farnell.com/exclusive-brands
Element14.com/exclusive-brands

TENMA™

Page <34>   25/06/25   V1.0

:CHANnel1:BASE:HIGH 2
:CHANnel1:BASE:LOW 0
:CHANnel1:BASE:PHAse 90
:CHANnel1:RAMP:SYMMetry 20
:CHANnel1:OUTPut ON

Configuring pulse wave

Description

For pulse width, there are amplitude, offset, and phase relative to synchronous pulse. It also adds edge slope and duty cycle (or pulse width). The amplitude and offset of a pulse wave can be set by using high and low voltage values.

Examples

The waveform below can be set by SCPI command series, and the high and low levels can be used to replace :CHANnel1:BASE:AMPLitude and :CHANnel1:BASE:OFFSet.

The commands below 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 introduces how to configure arbitrary wave.

Description

For harmonic wave, there are frequency, amplitude, offset, and phase. It also adds mode and waveform file.

Examples

The codes below can load and modify built-in arbitrary waveforms.
: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

Newark.com/exclusive-brands
Farnell.com/exclusive-brands
Element14.com/exclusive-brands

™

Page <35>   25/06/25   V1.0

:CHANnel1:BASE:PHAse 90
:CHANnel1:OUTPut ON

The waveform generated from these commands is shown in the Figure below.
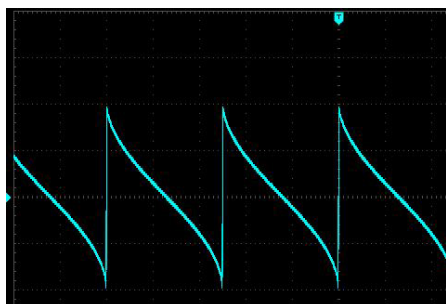
Appendix 1: <key> list

Keyword of Key Command	Functional descriptions	LED Light
CH1	Channel 1 Key	√
CH2	Channel 2 key	√
RIGHT	Direction Key (Right)	
LEFT	Direction Key (Left)	
OK	Confirmation Key	
SYMBOL	Symbol (Numeric Key)	
NUM0	Numeric Key (Number: 0)	
NUM1	Numeric Key (Number: 1)	
NUM2	Numeric Key (Number: 2)	
NUM3	Numeric Key (Number: 3)	
NUM4	Numeric Key (Number: 4)	
NUM5	Numeric Key (Number: 5)	
NUM6	Numeric Key (Number: 6)	
NUM7	Numeric Key (Number: 7)	
NUM8	Numeric Key (Number: 8)	
NUM9	Numeric Key (Number: 9)	
DOT	Decimal Point (Number Key)	
F1	Horizontal Function Key 1	
F2	Horizontal Function Key 2	
F3	Horizontal Function Key 3	
F4	Horizontal Function Key 4	
F5	Horizontal Function Key 5	
F6	Horizontal Function Key 6	
MENU	Menu Key	
VF1	Vertical Function Key 1	

Newark.com/exclusive-brands
Farnell.com/exclusive-brands
Element14.com/exclusive-brands

Page <36>   25/06/25   V1.0

Keyword of Key Command	Functional descriptions	LED Light
VF2	Vertical Function Key 2	
VF3	Vertical Function Key 3	
VF4	Vertical Function Key 4	
UTILity	System	
TRIGGER	Trigger Key	√
UP	Turn the knob clockwise	
DOWN	Turn the knob anticlockwise	

Newark.com/exclusive-brands
Farnell.com/exclusive-brands
Element14.com/exclusive-brands

Page <37>   25/06/25   V1.0