UNI-T®

Instruments.uni-trend.com

USG Series Signal Generators

Programming Manual

V1.0

August 2024

Warranty and Statement

Copyright

Brand Information

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

File Number

August 27th, 2024

Software Version

V1.03.0026

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

Statement

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

Product Certification

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

Contact Us

If you have any questions or problems, please contact **UNI-T**.

Website: https://www.uni-trend.com

SCPI

SCPI (Standard Commands for Programmable Instruments) is a standard command set based on the existing standards IEEE 488.1 and IEEE 488.2. And follow the IEEE754 standard floating-point arithmetic rules, ISO646 information exchange 7 bits code symbol (equivalent to ASCII programming) and other standard standardized instrument programming language.

Command Format

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

Symbol Description

The following four symbols are not part of SCPI command, it cannot be sent with the command. It is usually used as a supplementary description of command parameters.

- **Braces { }**
  The braces usually contain multiple optional parameters; one of the parameters should be selected when sending the command.
  For example, :DISPlay:GRID:MODE { FULL | GRID | CROSS | NONE}
- **Vertical Bar |**
  The vertical bar is used to separate multiple parameters; one of the parameters should be selected when sending the command.
  For example, :DISPlay:GRID:MODE { FULL | GRID | CROSS | NONE}
- **Square Brackets [ ]**
  The contents in square brackets (command keywords) can be omissible. If the parameter is omitted, the instrument will set the parameter as the default value.
  For example, for the command :MEASure:NDUTy? [<source>], [<source>] indicates the current channel.
- **Triangular Brackets < >**
  The parameter in the brackets must be replaced with valid value.
  For example, send the command DISPlay:GRID:BRIGhtness 30 in the format of DISPlay:GRID:BRIGhtness <count>.

Parameter Description

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

- **Boolean**

  The parameter can be set to “ON” (1) or “OFF” (0).

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

- **Integer**

  Unless otherwise specified, the parameter can take any valid integer value.

  Note: Do not set decimal as parameter, otherwise, errors may occur.

  For example, <count> in the command :DISPlay:GRID:BRIGhtness <count> can take any integer value from 0 to 100.

- **Real**

  Unless otherwise specified, the parameter can take any valid integer value.

  For example, for CH1, <offset> in the command CHANnel1:OFFSet <offset> can take a real number as its value.

- **Discrete**

  The parameter can only take specified numbers or characters.

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

- **ASCII String**

  A string parameter can contain any ASCII character. Strings must begin and end with paired quotation marks, which can be either single or double quotes. To include a quotation mark or delimiter within the string, type it twice without adding any other characters.

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

Abbreviation

All commands are case-sensitive. If a command is written in an abbreviated format, all capital letters in the command must be input completely.

Date Return

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

For example, #3123xxxxxxxxxxxxxxxxxxx\n indicates 123 strings batch data return format, ‘3’ indicates “123” occupies three character bits.

SCPI Explanation

IEEE488.2 Common Command

*CLS

- **Command Format**

  *CLS

- **Functional Description**

  Clear the values of all event registers and state byte registers.

*ESE

- **Command Format**

  *ESE <integer>

  *ESE?

- **Functional Description**

  Set the enable value for standard event register. The decimal sum of the register bits, the default is 0. For example, to enable 2-bit (value: 4), 3-bit (value: 8), and bit-7 (value: 128), so the decimal sum should be 140 (4+8+128). 1-bit and 6-bit of the standard event register are not enabled.

  <integer>: An integer value

- **Return Format**

  The query returns the enable value of the standard event register as an integer value.

- **For Example**

*ESE 16	Enable the fourth bit for the standard event register.
*ESE?	The query returns 16.

*ESR?

- **Command Format**

  *ESR?

- **Functional Description**

  Query the standard event register. The event register is read-only register, lock the event from the condition register.

- **Return Format**

  The query returns the event value of the standard event register as an integer value.

- **For Example**

*ESR?	The query returns 24, with 3-bit and 4-bit enabled.

***IDN?**

- **Command Format**

  *IDN?

- **Functional Description**

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

- **Return Format**

  The query returns the manufacture name, product model, and product serial number. The software version is separated by commas.

  Note: The return model number should be consistent with the nameplate.

- **For Example**

  UNI-TREND,USG5000M,ASA322214A002,SW V1.03.0027

***OPC**

- **Command Format**

  *OPC

  *OPC?

- **Functional Description**

  Set the standard event state register to 1 after the current operation is finished.

- **Return Format**

  The query returns whether the current operation is finished. 0 indicates that the current operation is unfinished. 1 indicates that the current operation is unfinished.

- **For Example**

*OPC	Set the standard event state register to 1.
*OPC?	The query returns 1, indicating that the current operation is unfinished; otherwise, the query returns 0.

***RST**

- **Command Format**

  *RST

- **Functional Description**

  Restore the instrument to its factory default settings, clear all the error messages, and send and receive queue buffers.

***SRE**

- **Command Format**

  *SRE <integer>

  *SRE?

- **Functional Description**

  Set the enable value of the state byte register. Service Request Enable is the bit in the enable register of the state register. The enable register defines which bit in the event register will be reported to the "state byte" register group. The enable register is readable and writeable.

  <integer>: An integer value

- **Return Format**

  The query returns the enable value of the state byte register as an integer.

- **For Example**

*SRE 24	Enable 3-bit and 4-bit for the enable register.
*SRE?	The query returns 24.

***STB?**

- **Command Format**

  *STB?

- **Functional Description**

  Query the state value of the state byte register.

- **Return Format**

  The query returns the state value of the state byte register as an integer.

- **For Example**

*STB?	40, set 3-bit and 5-bit for the state byte register.

SYSTem Command

This command is used to set the basic operations of the RF, including the full QWERTY lock and system data settings.

:SYSTem:NETAdpt

- **Command Format**

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

  :SYSTem:NETAdpt?

- **Functional Description**

Query the state of network adapter to determine whether it is ON (1) or OFF (0).

1|ON: On

0|OFF: Off

- **Return Format**

The query returns the state of the network adapter: 0 or 1.

- **For Example**

:SYSTem:NETAdpt ON	The network adapter is enabled.
:SYSTem:NETAdpt?	The query returns 1.

:SYSTem:IPV4

- **Command Format**

:SYSTem:IPV4 <ip>

:SYSTem:IPV4?

- **Functional Description**

Configure the IP address for the network settings.

<ip>: Four fields in dotted decimal format, represented as xxx.xxx.xxx.xxx.

- **Return Format**

The query returns the current IP address in the format xxx.xxx.xxx.xxx.

- **For Example**

:SYSTem:IPV4 “192.168.20.111”	Set the IP address to “192.168.20.111”.
:SYSTem:IPV4?	The query returns the IP address “192.168.20.111”.

:SYSTem:DHCP

- Command Format

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

:SYSTem:IPV4:DHCP?

- **Functional Description**

Query the state of DHCP to determine whether it is ON (1) or OFF (0).

1|ON: On

0|OFF: Off

- **Return Format**

The query returns the state of the DHCP: 0 or 1.

- **For Example**

:SYSTem:DHCP ON	The DHCP is enabled.
:SYSTem:DHCP?	The query returns 1.

:SYSTem:GATEway

- **Command Format**

  :SYSTem:GATEway <gateway>

  :SYSTem:GATEway?

- **Functional Description**

  Configure the gateway for the network settings.

  <gateway>: Four fields in dotted decimal format, represented as xxx.xxx.xxx.xxx.

- **Return Format**

  The query returns the gateway data in the format xxx.xxx.xxx.xxx.

- **For Example**

  :SYSTem:GATEway “192.168.20.1” Set the gateway to "192.168.20.1".

  :SYSTem::GATEway? The query returns the gateway "192.168.20.1".

:SYSTem:IPMAsk

- **Command Format**

  :SYSTem:IPMAsk <mask>

  :SYSTem:IPMAsk?

- **Functional Description**

  Configure the mask for the network settings.

  <mask>: Four fields in dotted decimal format, represented as xxx.xxx.xxx.xxx.

- **Return Format**

  The query returns mask data in the format xxx.xxx.xxx.xxx.

- **For Example**

  :SYSTem:IPMAsk “255.255.255.0” Set the mask to "255.255.255.0".

  :SYSTem:IPMAsk? The query returns "255.255.255.0".

:SYSTem:MAC?

- **Command Format**

  :SYSTem:MAC?

- **Functional Description**

  Query the MAC address of the device.

- **Return Format**

  The query returns the MAC address as a character string.

- **For Example**

  :SYSTem:MAC? The query returns "3E:A4:20:3D:82:83".

:SYSTem:DATE

- **Command Format**

  :SYSTem:DATE <yyyymmdd>

  :SYSTem:DATE?

- **Functional Description**

  Set the system date in the format YYYYMMDD, without any separators. For example, to set the date to January 1, 2000, the parameter is expressed as 20000101.

- **Return Format**

  The query returns the system date.

- **For Example**

  :SYSTem:DATE? The query returns 20000101 (January 1, 2000).

:SYSTem:BACKlight

- **Command Format**

  :SYSTem:BACKlight <integer>

  :SYSTem:BACKlight?

- **Functional Description**

  Adjust the backlight brightness of the system.

  <integer>: A continuous integer, with the default unit in %. The value range is from 0 to 100.

- **Return Format**

  The query returns the backlight brightness of the system, with the unit in %.

- **For Example**

  :SYSTem:BACKlight 80 Set the backlight brightness of the system to 80%.

  :SYSTem:BACKlight? The query returns 80.

:SYSTem:CFORmat

- **Command Format**

  :SYSTem:CFORmat {HR12|HR24}

  :SYSTem:CFORmat?

- **Functional Description**

  Set the system time format.

  HR12: 12-hour system

  HR24: 24-hour system

- **Return Format**

  The query returns the system time format: HR12 or HR24.

➢ **For Example**

:SYSTem:CFORmat HR24	Set the system time format to HR24.
:SYSTem:CFORmat?	The query returns HR24.

:SYSTem:LANGuage

➢ **Command Format**

:SYSTem:LANGuage {CHINese|ENGLish|GERMan}

:SYSTem:LANGuage?

➢ **Functional Description**

Set the system language.

CHINese: Simplified-Chinese

ENGLish: English

GERMan: German

➢ **Return Format**

The query returns the system language: CHINese, ENGLish, or GERMan.

➢ **For Example**

:SYSTem:LANGuage CHINese	Set the system language to CHINese.
:SYSTem:LANGuage?	The query returns CHINese.

:SYSTem:LOCK

➢ **Command Format**

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

:SYSTem:LOCK?

➢ **Functional Description**

Lock or unlock the keyboard and touchscreen.

1|ON: Lock

0|OFF: Unlock

➢ **Return Format**

The query returns the lock state of keyboard and touchscreen. 0 indicates that the keyboard and touchscreen are unlocked. 1 indicates that the keyboard and touchscreen are locked.

➢ **For Example**

:SYSTem:LOCK ON	The keyboard and touchscreen are locked.
:SYSTem:LOCK OFF	The keyboard and touchscreen are unlocked.
:SYSTem:LOCK?	The query returns 0, indicating that the keyboard and touchscreen are unlocked.

:SYSTem:FORMat

- **Command Format**

  :SYSTem:FORMat {BMP|PNG}

  :SYSTem:FORMat?

- **Functional Description**

  Select the format for saving system screenshots.

- **Return Format**

  The query returns the system picture format: BMP, PNG.

- **For Example**

:SYSTem:FORMat BMP	Select the system picture format to BMP.
:SYSTem:FORMat?	The query returns BMP.

:SYSTem:PICInvert

- **Command Format**

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

  :SYSTem:PICInvert?

- **Functional Description**

  Query the state of invert color setting to determine whether it is ON (1) or OFF (0).

  1|ON: On

  0|OFF: Off

- **Return Format**

  The query returns the state of invert color setting: 0 or 1.

- **For Example**

:SYSTem:PICInvert ON	The invert color setting is enabled.
:SYSTem:PICInvert?	The query returns 1.

:SYSTem:TIME

- **Command Format**

  :SYSTem:TIME <hhmmss>

  :SYSTem:TIME?

- **Functional Description**

  Configure the system time in a 24-hour format. For example, to set the time to 13:01:01, the parameter is expressed as 130101.

- **Return Format**

  The query returns the system time.

- **For Example**

  :SYSTem:TIME? The query returns 130101.

:SYSTem:TEMPerature?

- **Command Format**

  :SYSTem:TEMPerature?

- **Functional Description**

  Query the instrument temperature.

- **Return Format**

  The query returns the instrument temperature as 30°C.

- **For Example**

  :SYSTem:TEMPerature? The query returns 30°C.

:SYSTem:GPIB

- **Command Format**

  :SYSTem:GPIB<value>

- **Functional Description**

  Configure the GPIB address without specifying a unit.

- **Return Format**

  The query returns the GPIB address.

- **For Example**

  :SYSTem:GPIB 2 Set the GPIB address to 2.

  :SYSTem:GPIB? The query returns 2.

KEY Command

:KEY:<key>

- **Command Format**

  :KEY:<key>

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

  :KEY:<key>:LOCK?

  :KEY:LED?

- **Functional Description**

  Enable the key function and lock/unlock the key. The definition and functional description, referring to Appendix 1: <key> Table.

- **Return Format**

  The query returns the key lock state or LED state.

  LED state: 0 indicates the LED is off, and 1 indicates the LED is illuminated (green).

- **For Example**

  :KEY:FREQ Enter the frequency setting menu.

  :KEY:LED? The query returns LED state: 0 indicates the LED is off.

:KEY:LED?

- **Command Format**

  :KEY:LED?

- **Functional Description**

  Query the state of all keys with indicator light.

- **Return Format**

  The query returns the state of all keys with indicator lights as a character sequence, where each character indicates the state of one key. Illuminated: ASCII '1'; Extinguished: ASCII '0'. The four keys with indicator lights are the Touch/Lock key, LF key, MOD key, and RF key. The query returns a 4-bit string consisting of '1' or '0'.

- **For Example**

  The query returns ASCII 1000, indicating that the Touch/Lock is illuminated, while LF key, MOD key, and RF key are off.

:KEY:LOCK?

- **Command Format**

  :KEY:LOCK?

- **Functional Description**

  Query the lock state of all keys.

- **Return Format**

  The query returns the lock state of all keys. It returns a character sequence where each character indicates the lock state of one key. "Lock" is represented by ASCII '1' and "Unlock" by ASCII '0'. The lock state is returned according to Appendix 1: <key> Table.

- **For Example**

  The query returns ASCII 00011000000000000000000000000000000000000, indicating that out of the total 41 keys, only the fourth and fifth keys are locked.

unction Command

This command is used to set the modulation source, analog modulation, RF, function generation, and modulation input functions.

SOURce Command

Modulation Source

:MOGEn:STATe

- **Command Format**

  [:SOURce]:MOGEn:STATe {{0 | OFF | {1 | ON}}

  [:SOURce]:MOGEn:STATe?

- **Functional Description**

  Query the state of modulation source to determine whether it is ON (1) or OFF (0).

- **Return Format**

  The query returns the state of the modulation source: 0 or 1.

- **For Example**

:SOURce:MOGEn:STATe ON	The modulation source is enabled.
:SOURce:MOGEn:STATe?	The query returns 1.

:MOGEn:TYPE

- **Command Format**

  [:SOURce]:MOGEn:TYPE {{0 | SINE} | {1 | SQUAre} | {2 | RAMP}}

  [:SOURce]:MOGEn:TYPE?

- **Functional Description**

  Select the modulation wave.

- **Return Format**

  The query returns the modulation wave type: SINE, SQUAre, or RAMP.

- **For Example**

:SOURce:MOGEn:TYPE SQUAre	Select the modulation wave to SQUAre.
:SOURce:MOGEn:TYPE?	The query returns SQUAre.

:MOGEn:FREQ

- **Command Format**

  [:SOURce]:MOGEn:FREQ <value><unit>

[:SOURce]:MOGEn:FREQ?

- **Functional Description**

  Set the frequency for the modulation source.

- **Return Format**

  The query returns the frequency value, with the unit in Hz.

- **For Example**

:SOURce:MOGEn:FREQ 1kHz	Set the frequency for the modulation source to 1 kHz.
:SOURce:MOGEn:FREQ?	The query returns 1000.

:MOGEn:AMPT

- **Command Format**

  [:SOURce]:MOGEn:AMPT <value><unit>

  [:SOURce]:MOGEn:AMPT?

- **Functional Description**

  Set the amplitude for the modulation source, with the unit specified as Vpp, mVpp, Vrms, or mVrms.

- **Return Format**

  The query returns the amplitude value along with its unit.

- **For Example**

:SOURce:MOGEn:AMPT 100mVpp	Set the amplitude for the modulation source to 100 mVpp.
:SOURce:MOGEn:AMPT?	The query returns 100 mVpp.

:MOGEn:PHASe

- **Command Format**

  [:SOURce]:MOGEn:PHASe <value><unit>

  [:SOURce]:MOGEn:PHASe?

- **Functional Description**

  Set the phase for the modulation source, with the unit specified as either degrees (deg) or radians (rad).

- **Return Format**

  The query returns the phase value along with its unit.

- **For Example**

:SOURce:MOGEn:PHASe 30deg	Set the phase for the modulation source to 30 deg.
:SOURce:MOGEn:PHASe ?	The query returns 30 deg.

MOGEn:DCOFst

- **Command Format**

  [:SOURce]:MOGEn:DCOFst <value><unit>

  [:SOURce]:MOGEn:DCOFst?

- **Functional Description**

  Set the DC offset for the modulation source, with the unit specified as either V or mV.

- **Return Format**

  The query returns the DC offset value along with its unit.

- **For Example**

  :SOURce:MOGEn:DCOFst 500mv Set the DC offset for the modulation source to 500 mv.

  :SOURce:MOGEn:DCOFst? The query returns 500 mv.

Analog Modulation

:MOD:STATe

- **Command Format**

  [:SOURce]:MOD:STATe {{1|ON} | {0|OFF}}

  [:SOURce]:MOD:STATe?

- **Functional Description**

  Query the state of analog modulation to determine whether it is ON (1) or OFF (0).

- **Return Format**

  The query returns the state of the analog modulation: 0 or 1.

- **For Example**

  :SOURce:MOD:STATe ON The analog modulation is enabled.

  :SOURce:MOD:STATe? The query returns 1.

:MOD:AM:EN

- **Command Format**

  [:SOURce]:MOD:AM:EN {{1|ON} | {0|OFF}}

  [:SOURce]:MOD:AM:EN?

- **Functional Description**

  Set the amplitude modulation to ON (1) or OFF (0).

- **Return Format**

  The query returns the state of amplitude modulation: 0 or 1.

- **For Example**

[:SOURce]:MOD:AM:EN ON	Enable the amplitude modulation.
[:SOURce]:MOD:AM:EN?	The query returns 1.

:MOD:AM:SOURce

- **Command Format**

  [:SOURce]:MOD:AM:SOURce {{0 | INT} | {1 | EXT} | {2 | INT+EXT}}

  [:SOURce]:MOD:AM:SOURce?

- **Functional Description**

  Select the modulation source for amplitude modulation to internal, external, or internal and external.

- **Return Format**

  The query returns the modulation source type: INT (Internal), EXT (External), or INT+EXT (Internal and External).

- **For Example**

:SOURce:MOD:AM:SOURce INT	Select the modulation source for amplitude modulation to INT.
:SOURce:MOD:AM:SOURce?	The query returns INT.

:MOD:AM:DEPTh

- **Command Format**

  [:SOURce]:MOD:AM:DEPTh <value>

  [:SOURce]:MOD:AM:DEPTh?

- **Functional Description**

  Set the modulation depth for amplitude modulation, with the value as a floating-point number.

- **Return Format**

  The query returns the modulation depth value, with the unit in %.

- **For Example**

:SOURce:MOD:AM:DEPTh 60.5	Set the modulation depth to 60.5%.
:SOURce:MOD:AM:DEPTh?	The query returns the modulation depth value of 60.5.

:MOD:FM:EN

- **Command Format**

  [:SOURce]:MOD:FM:EN {{1|ON} | {0|OFF}}

  [:SOURce]:MOD:FM:EN?

- **Functional Description**

Set the frequency modulation to ON (1) or OFF (0).

- **Return Format**

  The query returns the state of frequency modulation: 0 or 1.

- **For Example**

:SOURce:MOD:FM:EN ON	Enable the frequency modulation.
:SOURce:MOD:FM:EN?	The query returns 1.

:MOD:FM:SOURce

- **Command Format**

  [:SOURce]:MOD:FM:SOURce {{0 | INT} | {1 | EXT} | {2 | INT+EXT}}

  [:SOURce]:MOD:FM:SOURce?

- **Functional Description**

  Select the modulation source for frequency modulation to internal, external, or internal and external.

- **Return Format**

  The query returns the modulation source type: INT (Internal), EXT (External), or INT+EXT (Internal and External).

- **For Example**

:SOURce:MOD:FM:SOURce INT	Select the modulation source for frequency modulation to INT.
:SOURce:MOD:FM:SOURce?	The query returns INT.

:MOD:FM:DEV

- **Command Format**

  [:SOURce]:MOD:FM:DEV <value><unit>

  [:SOURce]:MOD:FM:DEV?

- **Functional Description**

  Set the frequency offset for frequency modulation.

- **Return Format**

  The query returns the frequency offset, with the unit in Hz.

- **For Example**

:SOURce:MOD:FM:DEV 1kHz	Set the frequency offset to 1 kHz.
:SOURce:MOD:FM:DEV?	The query returns 1.000000e+03.

:MOD:PM:EN

- **Command Format**

  [:SOURce]:MOD:PM:EN {{1|ON} | {0|OFF}}

  [:SOURce]:MOD:PM:EN?

- **Functional Description**

  Set the phase modulation to ON (1) or OFF (0).

- **Return Format**

  The query returns the state of phase modulation: 0 or 1.

- **For Example**

:SOURce:MOD:PM:EN ON	Enable the phase modulation.
:SOURce:MOD:PM:EN?	The query returns 1.

:MOD:PM:SOURce

- **Command Format**

  [:SOURce]:MOD:PM:SOURce {{0 | INT} | {1 | EXT} | {2 | INT+EXT}}

  [:SOURce]:MOD:PM:SOURce?

- **Functional Description**

  Select the modulation source for phase modulation to internal, external, or internal and external.

- **Return Format**

  The query returns the modulation source type: INT (Internal), EXT (External), or INT+EXT (Internal and External).

- **For Example**

:SOURce:MOD:PM:SOURce INT	Select the modulation source for phase modulation to INT.
:SOURce:MOD:PM:SOURce?	The query returns INT.

:MOD:PM:PHASe

- **Command Format**

  [:SOURce]:MOD:PM:PHASe <value><unit>

  [:SOURce]:MOD:PM:PHASe?

- **Functional Description**

  Set the phase offset for phase modulation, with the unit specified as either degrees (deg) or radians (rad).

- **Return Format**

  The query returns the phase offset along with its unit.

➢ **For Example**

:SOURce:MOD:PM:PHASe 50deg	Set the phase offset to 50 deg.
:SOURce:MOD:PM:PHASe?	The query returns 50 deg.

:MOD:PULSe:EN

➢ **Command Format**

[:SOURce]:MOD:PULSe:EN {{1|ON} | {0|OFF}}

[:SOURce]:MOD:PULSe:EN?

➢ **Functional Description**

Set the pulse modulation to ON (1) or OFF (0).

➢ **Return Format**

The query returns the state of pulse modulation: 0 or 1.

➢ **For Example**

:SOURce:MOD:PULSe:EN ON	Enable the pulse modulation.
:SOURce:MOD:PULSe:EN?	The query returns 1.

:MOD:PULSe:TYPE

➢ **Command Format**

[:SOURce]:MOD:PULSe:TYPE {{FREErun | 0} | {SQUAre | 1} | {TRIGgered | 2} | {ADJDoublet | 3} | {TRIDoublet | 4} | {GATEd | 5} | {EXTPulse | 6} | {PULSetrain | 7}}

[:SOURce]:MOD:PULSe:TYPE?

➢ **Functional Description**

Set the modulation type for pulse modulation.

➢ **Return Format**

The query returns the modulation type: FREErun, SQUAre, TRIGgered, ADJDoublet, TRIDoublet, GATEd, EXTPulse, PULSetrain.

➢ **For Example**

:SOURce:MOD:PULSe:TYPE FREErun	Set the modulation type for pulse modulation to FREErun.
:SOURce:MOD:PULSe:TYPE?	The query returns FREErun.

:MOD:PULSe:PERIod

➢ **Command Format**

[:SOURce]:MOD:PULSe:PERIod <value><unit>

[:SOURce]:MOD:PULSe:PERIod?

- **Functional Description**

  Set the period for pulse modulation, with the unit specified as s, ms, us, or ns.

- **Return Format**

  The query returns the period of pulse modulation, with the unit in seconds (s).

- **For Example**

:SOURce:MOD:PULSe:PERIod 60ns	Set the period for pulse modulation to 60 ns.
:SOURce:MOD:PULSe:PERIod?	The query returns 6.000000e-08.

:MOD:PULSe:RATE

- **Command Format**

  [:SOURce]:MOD:PULSe:RATE <value><unit>

  [:SOURce]:MOD:PULSe:RATE?

- **Functional Description**

  Set the square wave rate in pulse modulation.

- **Return Format**

  The query returns the square wave rate value, with the unit in Hz.

- **For Example**

:SOURce:MOD:PULSe:RATE 1kHz	Set the square wave rate to 1 kHz.
:SOURce:MOD:PULSe:RATE?	The query returns 1000.

:MOD:PULSe:DELAy

- **Command Format**

  [:SOURce]:MOD:PULSe:DELAy <value><unit>

  [:SOURce]:MOD:PULSe:DELAy?

- **Functional Description**

  Set the delay time for pulse modulation, with the unit specified as s, ms, us, or ns.

- **Return Format**

  The query returns the delay time of pulse modulation, with the unit in seconds (s).

- **For Example**

:SOURce:MOD:PULSe:DELAy 20ns	Set the delay time for pulse modulation to 20.
:SOURce:MOD:PULSe:DELAy?	The query returns 2.000000e-08.

:MOD:PULSe:WIDTh

- **Command Format**

  [:SOURce]:MOD:PULSe:WIDTh <value><unit>

[:SOURce]:MOD:PULSe:WIDTh?

- **Functional Description**

  Set the pulse width for pulse modulation, with the unit specified as s, ms, us, or ns.

- **Return Format**

  The query returns the pulse width of pulse modulation, with the unit in seconds (s).

- **For Example**

:SOURce:MOD:PULSe:WIDTh 50ns	Set the pulse width for pulse modulation to 50 ns.
:SOURce:MOD:PULSe:WIDTh?	The query returns 5.000000e-08.

:MOD:PULSe:DELAys

- **Command Format**

  [:SOURce]:MOD:PULSe:DELAys <value><unit>

  [:SOURce]:MOD:PULSe:DELAys?

- **Functional Description**

  Set the time for pulse delay 2, with the unit specified as s, ms, us, or ns.

- **Return Format**

  The query returns the time of pulse delay 2, with the unit in seconds (s).

- **For Example**

:SOURce:MOD:PULSe:DELAys 20ns	Set the time for pulse delay 2 to 20 ns.
:SOURce:MOD:PULSe:DELAys?	The query returns 2.000000e-08.

:MOD:PULSe:WIDThs

- **Command Format**

  [:SOURce]:MOD:PULSe:WIDThs <value><unit>

  [:SOURce]:MOD:PULSe:WIDThs?

- **Functional Description**

  Set the pulse width 2 for pulse modulation, with the unit specified as s, ms, us, or ns.

- **Return Format**

  The query returns the pulse width 2 of pulse modulation, with the unit in seconds (s).

- **For Example**

:SOURce:MOD:PULSe:WIDThs 50ns	Set the pulse width 2 for pulse modulation to 50 ns.
:SOURce:MOD:PULSe:WIDThs?	The query returns 5.000000e-08.

:MOD:PULSe:SYNCwidth

- **Command Format**

[:SOURce]:MOD:PULSe:SYNCwidth <value><unit>

[:SOURce]:MOD:PULSe:SYNCwidth?

- **Functional Description**

Set the sync pulse width for pulse modulation, with the unit specified as s, ms, us, or ns.

- **Return Format**

The query returns the sync pulse width of pulse modulation, with the unit in seconds (s).

- **For Example**

:SOURce:MOD:PULSe:SYNCwidth 20ns	Set the sync pulse width for pulse modulation to 20 ns.
:SOURce:MOD:PULSe:SYNCwidth?	The query returns 2.000000e-08.

:MOD:PULSe:EXTPolarity

- **Command Format**

[:SOURce]:MOD:PULSe:EXTPolarity {{1|INVErt} | {0|NORMal}}

[:SOURce]:MOD:PULSe:EXTPolarity?

- **Functional Description**

Set the external polarity to normal (0) or invert (1).

- **Return Format**

The query returns the external polarity: 0 or 1.

- **For Example**

:SOURce:MOD:PULSe:EXTPolarity INVErt	Set the external polarity to INVErt.
:SOURce:MOD:PULSe:EXTPolarity?	The query returns 1.

:MOD:PULSe:TRIGgermode

- **Command Format**

[:SOURce]:MOD:PULSe:TRIGgermode {FREErun | TRIGgered | GATEd}

[:SOURce]:MOD:PULSe:TRIGgermode?

- **Functional Description**

Set the trigger mode to free run, external, or gating when the pulse mode is pulse train.

- **Return Format**

The query returns the trigger mode: 0, 1, or 2.

- **For Example**

:SOURce:MOD:PULSe:TRIGgermode FREErun	Set the trigger mode to FREErun.
:SOURce:MOD:PULSe:TRIGgermode?	The query returns 0.

:MOD:PULSe:LIST

- **Command Format**

  [:SOURce]:MOD:PULSe:LIST {{1|ON} | {0|OFF}}

  [:SOURce]:MOD:PULSe:LIST?

- **Functional Description**

  Set the pulse train list to ON (1) or OFF (0).

  1 | ON: When the pulse trains list is enabled, the list can be edited, and data can be sent.

  0 | OFF: When the pulse trains list is disabled, the list cannot be edited, and data cannot be sent.

- **Return Format**

  The query returns the state of pulse train list: 0 or 1.

- **For Example**

:SOURce:MOD:PULSe:LIST ON	Enable the pulse train list.
:SOURce:MOD:PULSe:LIST?	The query returns 1.

:MOD:PULSe:LIST:ONTIme

- **Command Format**

  [:SOURce]:MOD:PULSe:LIST:ONTIme <value><unit>

  [:SOURce]:MOD:PULSe:LIST:ONTIme?

- **Functional Description**

  Set the high level of the current row in the pulse train list, with the unit specified as s, ms, us, or ns.

- **Return Format**

  The query returns the high level of the current row in the pulse train list, with the unit in seconds (s).

- **For Example**

:SOURce:MOD:PULSe:LIST:ONTIme 20ns	Set the high level time to 20 ns.
:SOURce:MOD:PULSe:LIST:ONTIme?	The query returns 2.000000e-08.

:MOD:PULSe:LIST:OFFTime

- **Command Format**

  [:SOURce]:MOD:PULSe:LIST:OFFTime <value><unit>

  [:SOURce]:MOD:PULSe:LIST:OFFTime?

- **Functional Description**

  Set the low level of the current row in the pulse train list, with the unit specified as s, ms, us, or ns.

- **Return Format**

  The query returns the low level of the current row in the pulse train list, with the unit in seconds (s).

- **For Example**

:SOURce:MOD:PULSe:LIST:OFFTime 20ns	Set the low level time to 20 ns.
:SOURce:MOD:PULSe:LIST:OFFTime?	The query returns 2.000000e-08.

:MOD:PULSe:LIST:REPEat

- **Command Format**

  [:SOURce]:MOD:PULSe:LIST:REPEat <value>

  [:SOURce]:MOD:PULSe:LIST:REPEat?

- **Functional Description**

  Set the repeat count of the current line in the pulse train list without specifying a unit.

- **Return Format**

  The query returns the repeat count of the current line in the pulse train list.

- **For Example**

:SOURce:MOD:PULSe:LIST:REPEat 10	Set the repeat count to 10.
:SOURce:MOD:PULSe:LIST:REPEat?	The query returns 10.

:MOD:PULSe:LIST:CURRow

- **Command Format**

  [:SOURce]:MOD:PULSe:LIST:CURRow <value>

- **Functional Description**

  Set the current editable row in the pulse train list.

- **Return Format**

  This command does not return a value.

- **For Example**

:SOURce:MOD:PULSe:LIST:CURRow 1	Set row 1 as the row currently being edited.

:MOD:PULSe:LIST:ADDRow

- **Command Format**

  [:SOURce]:MOD:PULSe:LIST:ADDRow <value>

- **Functional Description**

  Add a row to the pulse train list without specifying a unit.

- **Return Format**

This command does not return a value.

- **For Example**

:SOURce:MOD:PULSe:LIST:ADDRow 2 Add two rows to the pulse train list.

:MOD:PULSe:LIST:SUBRow

- **Command Format**

[:SOURce]:MOD:PULSe:LIST:SUBRow <value>

- **Functional Description**

Remove the last two rows from the pulse train list.

- **Return Format**

This command does not return a value.

- **For Example**

:SOURce:MOD:PULSe:LIST:SUBRow 2 Remove the last two rows from the pulse train list.

Radio Frequency

:OUTPut

- **Command Format**

[:SOURce]:OUTPut[:STATe] {{1|ON} | {0|OFF}}

[:SOURce]:OUTPut[:STATe]?

- **Functional Description**

Set the radio frequency to ON (1) or OFF (0).

- **Return Format**

The query returns the state of radio frequency: 0 or 1.

- **For Example**

:SOURce:OUTPut:STATe ON Enable the radio frequency.

:SOURce:OUTPut:STATe? The query returns 1.

:FREQuency

- **Command Format**

[:SOURce]:FREQuency[:CW] <value><unit>

[:SOURce]:FREQuency[:CW]?

- **Functional Description**

Set the output frequency for radio frequency.

- **Return Format**

The query returns the output frequency, with the unit in Hz.

- **For Example**

:SOURce:FREQuency:CW 1GHz	Set the output frequency to 1 GHz.
:SOURce:FREQuency:CW?	The query returns 1e+9.

:FREQuency:OFFSet

- **Command Format**

[:SOURce]:FREQuency:OFFSet <value><unit>

[:SOURce]:FREQuency:OFFSet?

- **Functional Description**

Set the frequency offset for radio frequency.

- **Return Format**

The query returns the frequency offset, with the unit in Hz.

- **For Example**

:SOURce:FREQuency:OFFSet 1kHz	Set the frequency offset to 1 kHz.
:SOURce:FREQuency:OFFSet?	The query returns 1000.

:FREQuency:OFFSet:STATe

- **Command Format**

[:SOURce]:FREQuency:OFFSet:STATe {{1|ON} | {0|OFF}}

[:SOURce]:FREQuency:OFFSet:STATe?

- **Functional Description**

Set the frequency offset to ON (1) or OFF (0).

- **Return Format**

The query returns the state of frequency offset: 0 or 1.

- **For Example**

:SOURce:FREQuency:OFFSet:STATe ON	Enable the frequency offset.
:SOURce:FREQuency:OFFSet:STATe?	The query returns 1.

:RADIo:PHASeofst

- **Command Format**

[:SOURce]:RADIo:PHASeofst <value><unit>

[:SOURce]:RADIo:PHASeofst?

- **Functional Description**

Set the phase offset for radio frequency, with the unit specified as either degrees (deg) or

radians (rad).

- **Return Format**

  The query returns the phase offset along with its unit.

- **For Example**

  :SOURce:RADIo:PHASeofst 30deg Set the phase offset to 30 deg.

  :SOURce:RADIo:PHASeofst? The query returns 30 deg.

:RADIo:INTBcal

- **Command Format**

  [:SOURce]:RADIo:INTBcal <value><unit>

  [:SOURce]:RADIo:INTBcal?

- **Functional Description**

  Set the internal TB calibration, with the unit in ppb or ppm.

- **Return Format**

  The query returns the internal TB calibration along with its unit.

- **For Example**

  :SOURce:RADIo:INTBcal 10ppb Set the internal TB calibration to 10 ppb.

  :SOURce:RADIo:INTBcal? The query returns 10 ppb.

:RADIo:OSREf

- **Command Format**

  [:SOURce]:RADIo:OSREf {{0 | INT} | {1 | EXT} | {2 | AUTO}}

  [:SOURce]:RADIo:OSREf?

- **Functional Description**

  Set the reference source.

- **Return Format**

  The query returns the reference source: 0 (INT),1 (EXT), or 2 (Auto).

- **For Example**

  :SOURce:RADIo:OSREf EXT Set the reference source to EXT.

  :SOURce:RADIo:OSREf? The query returns 1.

:FREQuency:REFerence:STATe

- **Command Format**

  [:SOURce]:FREQuency:REFerence:STATe {{1|ON} | {0|OFF}}

  [:SOURce]:FREQuency:REFerence:STATe?

- **Functional Description**

  Set the frequency reference to ON (1) or OFF (0).

- **Return Format**

  The query returns the state of frequency reference: 0 or 1.

- **For Example**

:SOURce:FREQuency:REFerence:STATe ON	Enable the frequency reference.
:SOURce:FREQuency:REFerence:STATe?	The query returns 1.

:PHASe:REFerence

- **Command Format**

  [:SOURce]:PHASe:REFerence {{1|ON} | {0|OFF}}

  [:SOURce]:PHASe:REFerence?

- **Functional Description**

  Set the phase reference to ON (1) or OFF (0).

- **Return Format**

  The query returns the state of phase reference: 0 or 1.

- **For Example**

:SOURce:PHASe:REFerence ON	Enable the phase reference.
:SOURce:PHASe:REFerence?	The query returns 1.

:ROSCillator:OVEN:STATe

- **Command Format**

  [:SOURce]:ROSCillator:OVEN:STATe

  [:SOURce]:ROSCillator:OVEN:STATe?

- **Functional Description**

  Set the 10MHz Out to ON or OFF.

- **Return Format**

  The query returns the state of 10MHz Out: ON or OFF.

- **For Example**

:SOURce:ROSCillator:OVEN:STATe ON	Enable the 10MHz Out.
:SOURce:ROSCillator:OVEN:STATe?	The query returns 1.

:POWer

- **Command Format**

  [:SOURce]:POWer[:LEVel][:IMMediate][:AMPLitude] <value><unit>

[:SOURce]:POWer[:LEVel][:IMMediate][:AMPLitude]?

- **Functional Description**

  Set the output amplitude, with units specified as dBm, dBuV, uV, mV, V, nW, uW, or mW.

- **Return Format**

  The query returns the output amplitude, with the unit in dBm.

- **For Example**

:SOURce:POWer:LEVel:IMMediate:AMPLitude 1V	Set the output amplitude to 1 V.
:SOURce:POWer:LEVel:IMMediate:AMPLitude?	The query returns 13.

:POWer:OFFSet

- **Command Format**

  [:SOURce]:POWer[:LEVel][:IMMediate]:OFFSet <value><unit>

  [:SOURce]:POWer[:LEVel][:IMMediate]:OFFSet?

- **Functional Description**

  Set the amplitude offset, with the unit in dB.

- **Return Format**

  The query returns the amplitude offset along with its unit.

- **For Example**

:SOURce:POWer:LEVel:IMMediate:OFFSet 1dB	Set the amplitude offset to 1 dB.
:SOURce:POWer:LEVel:IMMediate:OFFSet?	The query returns 1 dB.

:RADIo:MAXPower

- **Command Format**

  [:SOURce]:RADIo:MAXPower <value><unit>

  [:SOURce]:RADIo:MAXPower?

- **Functional Description**

  Set the maximum user-defined power, with units specified as dBm, dBuV, uV, mV, V, nW, uW, or mW.

- **Return Format**

  The query returns the set power value, with the unit in dBm.

- **For Example**

:SOURce:RADIo:MAXPower 1V	Set the maximum user-defined power to 1 V.
:SOURce:RADIo:MAXPower?	The query returns 13.

:POWer:ATTenuation

- **Command Format**

  [:SOURce]:POWer:ATTenuation <value><unit>

  [:SOURce]:POWer:ATTenuation?

- **Functional Description**

  Set the attenuation value, with the unit in dB.

- **Return Format**

  The query returns the set attenuation value along with its unit.

- **For Example**

:SOURce:POWer:ATTenuation 10dB	Set the attenuation value to 10 dB.
:SOURce:POWer:ATTenuation?	The query returns 10 dB.

:POWer:ALC:LEVel

- **Command Format**

  [:SOURce]:POWer:ALC:LEVel <value><unit>

  [:SOURce]:POWer:ALC:LEVel?

- **Functional Description**

  Set the ALC (Automatic Level Control) amplitude, with units specified as dBm, dBuV, uV, mV, V, nW, uW, or mW.

- **Return Format**

  The query returns the ALC amplitude, with the unit in dBm.

- **For Example**

:SOURce:POWer:ALC:LEVel 10dBm	Set the ALC amplitude to 10 dBm.
:SOURce:POWer:ALC:LEVel ?	The query returns 10.

:POWer:ALC

- **Command Format**

  [:SOURce]:POWer:ALC[:STATe] {{0 | ON} | {1 | OFF} | {2 | AUTO}}

  [:SOURce]:POWer:ALC[:STATe]?

- **Functional Description**

  Set the ALC state to ON (0), OFF (1), or AUTO (2).

- **Return Format**

  The query returns the ALC state: 0, 1, or 2.

- **For Example**

:SOURce:POWer:ALC:STATe AUTO	Set the ALC state to AUTO.
:SOURce:POWer:ALC:STATe?	The query returns AUTO.

:RADIo:LEVElofst:EN

- **Command Format**

  [:SOURce]:RADIo:LEVElofst:EN {{1|ON} | {0|OFF}}

  [:SOURce]:RADIo:LEVElofst:EN?

- **Functional Description**

  Set the amplitude offset to ON (1) or OFF (0).

- **Return Format**

  The query returns the state of amplitude offset: 0 or 1.

- **For Example**

:SOURce:RADIo:LEVElofst:EN ON	Enable the amplitude offset.
:SOURce:RADIo:LEVElofst:EN?	The query returns 1.

:POWer:REFerence:STATe

- **Command Format**

  [:SOURce]:POWer:REFerence:STATe {{1|ON} | {0|OFF}}

  [:SOURce]:POWer:REFerence:STATe?

- **Functional Description**

  Set the amplitude reference to ON (1) or OFF (0).

- **Return Format**

  The query returns the state of amplitude reference: 0 or 1.

- **For Example**

:SOURce:POWer:REFerence:STATe ON	Enable the amplitude reference.
:SOURce:POWer:REFerence:STATe?	The query returns 1.

:RADIo:MAXPower:EN

- **Command Format**

  [:SOURce]:RADIo:MAXPower:EN {{1|ON} | {0|OFF}}

  [:SOURce]:RADIo:MAXPower:EN?

- **Functional Description**

  Set the maximum user-defined power to ON (1) or OFF (0).

- **Return Format**

  The query returns the state of maximum user-defined power: 0 or 1.

- **For Example**

:SOURce:RADIo:MAXPower:EN ON	Enable the maximum user-defined power.
:SOURce:RADIo:MAXPower:EN?	The query returns 1.

:RADIo:ATTEnhold:EN

- **Command Format**

  [:SOURce]:RADIo:ATTEnhold:EN {{1|ON} | {0|OFF}}

  [:SOURce]:RADIo:ATTEnhold:EN?

- **Functional Description**

  Set the manual attenuation to ON (1) or OFF (0).

- **Return Format**

  The query returns the state of manual attenuation: 0 or 1.

- **For Example**

:SOURce:RADIo:ATTEnhold:EN ON	Enable the manual attenuation.
:SOURce:RADIo:ATTEnhold:EN?	The query returns 1.

:RADIo:LEVElflat:EN

- **Command Format**

  [:SOURce]:RADIo:LEVElflat:EN {{1|ON} | {0|OFF}}

  [:SOURce]:RADIo:LEVElflat:EN?

- **Functional Description**

  Set the flatness list to ON (1) or OFF (0).

- **Return Format**

  The query returns the state of the flatness list 1: 0 or 1.

- **For Example**

:SOURce:RADIo:LEVElflat:EN ON	Enable the flatness list.
:SOURce:RADIo:LEVElflat:EN?	The query returns 1.

:RADIo:FLATness:FREQ

- **Command Format**

  [:SOURce]:RADIo:FLATness:FREQ <value><unit>

  [:SOURce]:RADIo:FLATness:FREQ?

- **Functional Description**

  Set the frequency of the current row.

- **Return Format**

The query returns the frequency of the current row, with the unit in Hz.

- **For Example**

:SOURce:RADIo:FLATness:FREQ 10kHz	Set the frequency of the current row to 10 kHz.
:SOURce:RADIo:FLATness:FREQ?	The query returns 10000.

:RADIo:FLATness:CORR

- **Command Format**

[:SOURce]:RADIo:FLATness:CORR <value><unit>

[:SOURce]:RADIo:FLATness:CORR?

- **Functional Description**

Set the calibrated value for the current row.

- **Return Format**

The query returns the calibrated value of the current row, with the unit in dB.

- **For Example**

:SOURce:RADIo:FLATness:CORR 10dB	Set the calibrated value of the current row to 10 dB.
:SOURce:RADIo:FLATness:CORR?	The query returns 10.

:RADIo:FLATness:CURRow

- **Command Format**

[:SOURce]:RADIo:FLATness:CURRow <value>

- **Functional Description**

Select a row to display and configure in the flatness list.

- **Return Format**

This command does not return a value.

- **For Example**

:SOURce:RADIo:FLATness:CURRow 2	Display and configure the second row in the flatness list.

:RADIo:FLATness:ADDRow

- **Command Format**

[:SOURce]:RADIo:FLATness:ADDRow <value>

- **Functional Description**

Append two rows to the end of the flatness list.

- **Return Format**

This command does not return a value.

- **For Example**

  :SOURce:RADIo:FLATness:ADDRow 2 Append two rows to the end of the flatness list.

:RADIo:FLATness:SUBRow

- **Command Format**

  [:SOURce]:RADIo:FLATness:SUBRow <value>

- **Functional Description**

  Remove the last two rows from the flatness list.

- **Return Format**

  This command does not return a value.

- **For Example**

  :SOURce:RADIo:FLATness:SUBRow 2 Remove the last two rows from the flatness list.

:RADIo:SWEEp:FREQ:EN

- **Command Format**

  [:SOURce]:RADIo:SWEEp:FREQ:EN {{1|ON} | {0|OFF}}

  [:SOURce]:RADIo:SWEEp:FREQ:EN?

- **Functional Description**

  Set the frequency sweep to ON (1) or OFF (0).

- **Return Format**

  The query returns the state of frequency sweep: 0 or 1.

- **For Example**

  :SOURce:RADIo:SWEEp:FREQ:EN ON Enable the frequency sweep.

  :SOURce:RADIo:SWEEp:FREQ:EN? The query returns 1.

:RADIo:SWEEp:LEVEl:EN

- **Command Format**

  [:SOURce]:RADIo:SWEEp:LEVEl:EN {{1|ON} | {0|OFF}}

  [:SOURce]:RADIo:SWEEp:LEVEl:EN?

- **Functional Description**

  Set the amplitude sweep to ON (1) or OFF (0).

- **Return Format**

  The query returns the state of amplitude sweep: 0 or 1.

- **For Example**

:SOURce:RADIo:SWEEp:LEVEl:EN ON	Enable the amplitude sweep.
:SOURce:RADIo:SWEEp:LEVEl:EN?	The query returns 1.

:RADIo:SWEEp:STEP:EN

- **Command Format**

  [:SOURce]:RADIo:SWEEp:STEP:EN {{1|ON} | {0|OFF}}

  [:SOURce]:RADIo:SWEEp:STEP:EN?

- **Functional Description**

  Set the step sweep to ON (1) or OFF (0).

- **Return Format**

  The query returns the state of step sweep: 0 or 1.

- **For Example**

:SOURce:RADIo:SWEEp:STEP:EN ON	Enable the step sweep.
:SOURce:RADIo:SWEEp:STEP:EN?	The query returns 1.

:RADIo:SWEEp:LIST:EN

- **Command Format**

  [:SOURce]:RADIo:SWEEp:LIST:EN {{1|ON} | {0|OFF}}

  [:SOURce]:RADIo:SWEEp:LIST:EN?

- **Functional Description**

  Set the list sweep to ON (1) or OFF (0).

- **Return Format**

  The query returns the state of list sweep: 0 or 1.

- **For Example**

:SOURce:RADIo:SWEEp:LIST:EN ON	Enable the list sweep.
:SOURce:RADIo:SWEEp:LIST:EN?	The query returns 1.

:LIST:DIRection

- **Command Format**

  [:SOURce]:LIST:DIRection {{1|DOWN} | {0|UP}}

  [:SOURce]:LIST:DIRection?

- **Functional Description**

  Set the sweep direction to up (0) or down (1).

- **Return Format**

  The query returns the sweep direction: 0 or 1.

- **For Example**

:SOURce:LIST:DIRection DOWN	Set the sweep direction to down.
:SOURce:LIST:DIRection?	The query returns 1.

:RADIo:SWEEp:MODE

- **Command Format**

[:SOURce]:RADIo:SWEEp:MODE {{1|SINGle} | {0|COUNt}}

[:SOURce]:RADIo:SWEEp:MODE?

- **Functional Description**

Set the sweep mode to continuous (0) or single (1).

- **Return Format**

The query returns the sweep mode: 0 or 1.

- **For Example**

:SOURce:RADIo:SWEEp:MODE SINGle	Set the sweep mode to SINGle.
:SOURce:RADIo:SWEEp:MODE?	The query returns 1.

:RADIo:SWEEp:TRIGedge

- **Command Format**

[:SOURce]:RADIo:SWEEp:TRIGedge {{0| RISEedge} | {1 | FALLedge}}

[:SOURce]:RADIo:SWEEp:TRIGedge?

- **Functional Description**

Set the sweep edge trigger to rising edge (0) or falling edge (1).

- **Return Format**

The query returns the sweep triggering edge: 0 or 1.

- **For Example**

:SOURce:RADIo:SWEEp:TRIGedge RISEedge	Set the sweep edge trigger to rising edge.
:SOURce:RADIo:SWEEp:TRIGedge ?	The query returns 0.

:RADIo:POINt:TRIGger

- **Command Format**

[:SOURce]:RADIo:POINt:TRIGger {AUTO | NKEY | BUS | EXT}

[:SOURce]:RADIo:POINt:TRIGger?

- **Functional Description**

Set the point trigger to auto (0), key (1), bus (2), or external (3).

- **Return Format**

  The query returns the point trigger: 0, 1, 2, or 3.

- **For Example**

:SOURce:RADIo:POINt:TRIGger NKEY	Set the point trigger to key.
:SOURce:RADIo:POINt:TRIGger?	The query returns 1.

:RADIo:TRIGger:MODE

- **Command Format**

  [:SOURce]:RADIo:TRIGger:MODE {AUTO | NKEY | BUS | EXT}

  [:SOURce]:RADIo:TRIGger:MODE?

- **Functional Description**

  Set the trigger mode to auto (0), key (1), bus (2), or external (3).

- **Return Format**

  The query returns the point mode: 0, 1, 2, or 3.

- **For Example**

:SOURce:RADIo:TRIGger:MODE NKEY	Set the trigger mode to key.
:SOURce:RADIo:TRIGger:MODE?	The query returns 1.

:FREQuency:STARt

- **Command Format**

  [:SOURce]:FREQuency:STARt <value><unit>

  [:SOURce]:FREQuency:STARt?

- **Functional Description**

  Set the start frequency for sweep.

- **Return Format**

  The query returns the start frequency of sweep, with the unit in Hz.

- **For Example**

:SOURce:FREQuency:STARt 1kHz	Set the start frequency for sweep to 1 kHz.
:SOURce:FREQuency:STARt?	The query returns 1.000000e+03.

:FREQuency:STOP

- **Command Format**

  [:SOURce]:FREQuency:STOP <value><unit>

  [:SOURce]:FREQuency:STOP?

- **Functional Description**

Set the stop frequency for sweep.

- **Return Format**

  The query returns the stop frequency of sweep, with the unit in Hz.

- **For Example**

:SOURce:FREQuency:STOP 1kHz	Set the stop frequency for sweep to 1 kHz.
:SOURce:FREQuency:STOP?	The query returns 1.000000e+03.

:POWer:STARt

- **Command Format**

  [:SOURce]:POWer:STARt <value><unit>

  [:SOURce]:POWer:STARt?

- **Functional Description**

  Set the start amplitude for sweep, with units specified as dBm, dBuV, uV, mV, V, nW, uW, or mW.

- **Return Format**

  The query returns the start amplitude of sweep, with the unit in dBm.

- **For Example**

:SOURce:POWer:STARt 1V	Set the start amplitude for sweep to 1 V.
:SOURce:POWer:STARt?	The query returns 13.

:POWer:STOP

- **Command Format**

  [:SOURce]:POWer:STOP <value><unit>

  [:SOURce]:POWer:STOP?

- **Functional Description**

  Set the stop amplitude for sweep， with units specified as dBm, dBuV, uV, mV, V, nW, uW, or mW.

- **Return Format**

  The query returns the stop amplitude of sweep, with the unit in dBm.

- **For Example**

:SOURce:POWer:STOP 1V	Set the stop amplitude for sweep to 1 V.
:SOURce:POWer:STOP?	The query returns 13.

:SWEep:DWELl

- **Command Format**

  [:SOURce]:SWEep:DWELl <value><unit>

  [:SOURce]:SWEep:DWELl?

- **Functional Description**

  Set the dwell time for step sweep, with the unit specified as s, ms, us, or ns.

- **Return Format**

  The query returns the dwell time, with the unit in seconds (s).

- **For Example**

:SOURce:SWEep:DWELl 50ms	Set the dwell time for step sweep to 50 ms.
:SOURce:SWEep:DWELl?	The query returns 5.000000e-02.

:SWEep:POINts

- **Command Format**

  [:SOURce]:SWEep:POINts <value>

  [:SOURce]:SWEep:POINts?

- **Functional Description**

  Set the sweep point without specifying a unit.

- **Return Format**

  The query returns the sweep point.

- **For Example**

:SOURce:SWEep:POINts 5	Set the sweep point to 5.
:SOURce:SWEep:POINts?	The query returns 5.

:RADIo:SWEEp:SHAPe

- **Command Format**

  [:SOURce]:RADIo:SWEEp:SHAPe {SAWTooth | TRIAngel}

  [:SOURce]:RADIo:SWEEp:SHAPe?

- **Functional Description**

  Set the sweep shape to sawtooth (0) or triangle (1).

- **Return Format**

  The query returns the sweep shape: 0 or 1.

- **For Example**

:SOURce:RADIo:SWEEp:SHAPe SAWTooth	Set the sweep shape to sawtooth.

:SOURce:RADIo:SWEEp:SHAPe? The query returns 0.

:SWEep:SPACing

- **Command Format**

  [:SOURce]:SWEep:SPACing {LINEar | LOG}

  [:SOURce]:SWEep:SPACing?

- **Functional Description**

  Set the sweep mode to linear (0) or logarithm (1).

- **Return Format**

  The query returns the sweep mode: 0 or 1.

- **For Example**

  :SOURce:SWEep:SPACing LOG Set the sweep mode to logarithm.

  :SOURce:SWEep:SPACing? The query returns 1.

:LIST:FREQuency

- **Command Format**

  [:SOURce]:LIST:FREQuency <value><unit>

  [:SOURce]:LIST:FREQuency?

- **Functional Description**

  Set the frequency for list sweep.

- **Return Format**

  The query returns the frequency of list sweep, with the unit in Hz.

- **For Example**

  :SOURce:LIST:FREQuency 1kHz Set the frequency for list sweep to 1 kHz.

  :SOURce:LIST:FREQuency? The query returns 1.000000e+03.

:LIST:POWer

- **Command Format**

  [:SOURce]:LIST:POWer <value><unit>

  [:SOURce]:LIST:POWer?

- **Functional Description**

  Set the amplitude for list sweep, with the unit in dBm.

- **Return Format**

  The query returns the amplitude of list sweep, with the unit in dBm.

- **For Example**

:SOURce:LIST:POWer 10dBm	Set the amplitude for list sweep to 10 dBm.
:SOURce:LIST:POWer?	The query returns 10.

:LIST:DWELl

- **Command Format**

  [:SOURce]:LIST:DWELl <value><unit>

  [:SOURce]:LIST:DWELl?

- **Functional Description**

  Set the dwell time of the current row for the list sweep.

- **Return Format**

  The query returns the current row, with the unit in seconds (s).

- **For Example**

:SOURce:LIST:DWELl 500ms	Set the dwell time of the current row to 500 ms.
:SOURce:LIST:DWELl?	The query returns 0.5.

:RADIo:SWEEp:CURRow

- **Command Format**

  [:SOURce]:RADIo:SWEEp:CURRow <value>

- **Functional Description**

  Set the current editable row in the list sweep table.

- **Return Format**

  This command does not return a value.

- **For Example**

:SOURce:RADIo:SWEEp:CURRow 2	Set row 2 as the row currently being edited.

:RADIo:SWEEp:ADDRow

- **Command Format**

  [:SOURce]:RADIo:SWEEp:ADDRow <value>

- **Functional Description**

  Add a row to the list sweep table.

- **Return Format**

  This command does not return a value.

- **For Example**

:SOURce:RADIo:SWEEp:ADDRow 2 Add two rows to the list sweep table.

:RADIo:SWEEp:SUBRow

- **Command Format**

  [:SOURce]:RADIo:SWEEp:SUBRow <value>

- **Functional Description**

  Remove the last two rows from the list sweep table.

- **Return Format**

  This command does not return a value.

- **For Example**

  :SOURce:RADIo:SWEEp:SUBRow 2 Remove the last two rows from the list sweep table.

:RADIo:SENSor:DEVIce

- **Command Format**

  [:SOURce]:RADIo:SENSor:DEVIce <value>

  [:SOURce]:RADIo:SENSor:DEVIce?

- **Functional Description**

  Select the connected power meter device, starting from serial number 0.

- **Return Format**

  The query returns the serial number of the connected device, without specifying a unit.

- **For Example**

  :SOURce:RADIo:SENSor:DEVIce 1 Select the second connected power meter device.

  :SOURce:RADIo:SENSor:DEVIce? The query returns 1.

:RADIo:SENSor:FREQ

- **Command Format**

  [:SOURce]:RADIo:SENSor:FREQ <value><unit>

  [:SOURce]:RADIo:SENSor:FREQ?

- **Functional Description**

  Set the measurement frequency for power meter.

- **Return Format**

  The query returns the measurement frequency, with the unit in Hz.

- **For Example**

  :SOURce:RADIo:SENSor:FREQ 10kHz Set the measurement frequency to 10 kHz.

:SOURce:RADIo:SENSor:FREQ?	The query returns 10000.

:RADIo:SWEEp:BUS:TRIGger

- **Command Format**

  [:SOURce]:RADIo:SWEEp:BUS:TRIGger

- **Functional Description**

  Set the bus triggering for radio frequency. No parameter is specified.

- **Return Format**

  This command does not return a value.

- **For Example**

  :SOURce:RADIo:SWEEp:BUS:TRIGger

  Set the bus triggering for radio frequency to trigger once.

:RADIo:SENSor:AMPTofst

- **Command Format**

  [:SOURce]:RADIo:SENSor:AMPTofst <value><unit>

  [:SOURce]:RADIo:SENSor:AMPTofst?

- **Functional Description**

  Set the amplitude offset for power meter, with the unit in dB.

- **Return Format**

  The query returns the amplitude offset along with its unit.

- **For Example**

:SOURce:RADIo:SENSor:AMPTofst 10dB	Set the amplitude offset to 10 dB.
:SOURce:RADIo:SENSor:AMPTofst?	The query returns 10 dB.

:RADIo:SENSor:FREQmul

- **Command Format**

  [:SOURce]:RADIo:SENSor:FREQmul <value>

  [:SOURce]:RADIo:SENSor:FREQmul?

- **Functional Description**

  Set the frequency multiplier ratio without specifying a unit.

- **Return Format**

  The query returns the frequency multiplier ratio.

- **For Example**

:SOURce:RADIo:SENSor:FREQmul 2	Set the frequency multiplier ratio to 2.
:SOURce:RADIo:SENSor:FREQmul?	The query returns 2.

:RADIo:SENSor:FREQofst

- **Command Format**

  [:SOURce]:RADIo:SENSor:FREQofst <value><unit>

  [:SOURce]:RADIo:SENSor:FREQofst?

- **Functional Description**

  Set the frequency offset for power meter.

- **Return Format**

  The query returns the frequency offset, with the unit in Hz.

- **For Example**

:SOURce:RADIo:SENSor:FREQofst 100Hz	Set the frequency offset for power meter to 100 Hz.
:SOURce:RADIo:SENSor:FREQofst?	The query returns 100.

:RADIo:SENSor:AVECount

- **Command Format**

  [:SOURce]:RADIo:SENSor:AVECount <value>

  [:SOURce]:RADIo:SENSor:AVECount?

- **Functional Description**

  Set the average count for power meter.

- **Return Format**

  The query returns the average count without specifying a unit.

- **For Example**

:SOURce:RADIo:SENSor:AVECount 10	Set the average count to 10.
:SOURce:RADIo:SENSor:AVECount?	The query returns 10.

:RADIo:SENSor:FREQcouple

- **Command Format**

  [:SOURce]:RADIo:SENSor:FREQcouple {{1|ON} | {0|OFF}}

  [:SOURce]:RADIo:SENSor:FREQcouple?

- **Functional Description**

  Set the frequency coupling for power meter to ON (1) or OFF (0).

- **Return Format**

  The query returns the state of frequency coupling: 0 or 1.

- **For Example**

  :SOURce:RADIo:SENSor:FREQcouple ON Enable the frequency coupling of power meter.

  :SOURce:RADIo:SENSor:FREQcouple? The query returns 1.

:RADIo:SENSor:LEVEl

- **Command Format**

  [:SOURce]:RADIo:SENSor:LEVEl?

- **Functional Description**

  Query the measurement amplitude of power meter.

- **Return Format**

  The query returns the measured amplitude along with its unit.

- **For Example**

  :SOURce:RADIo:SENSor:LEVEl? The query returns 0.00 dBm.

Function Generation

:LFOutput:STATe

- **Command Format**

  [:SOURce]:LFOutput:STATe {{1|ON} | {0|OFF}}

  [:SOURce]:LFOutput:STATe?

- **Functional Description**

  Set the low-frequency output to ON (1) or OFF (0).

- **Return Format**

  The query returns the state of low-frequency output: 0 or 1.

- **For Example**

  :SOURce:LFOutput:STATe ON Enable the low-frequency output.

  :SOURce:LFOutput:STATe? The query returns 1.

:LOWF:NOISesum

- **Command Format**

  [:SOURce]:LOWF:NOISesum {{0|OFF} | {1|ON}}

  [:SOURce]:LOWF:NOISesum?

- **Functional Description**

Set the low-frequency noise to ON (1) or OFF (0).

- **Return Format**

  The query returns the state of low-frequency noise: 0 or 1.

- **For Example**

:SOURce:LOWF:NOISesum ON	Enable the low-frequency noise.
:SOURce:LOWF:NOISesum?	The query returns 1.

:LOWF:TYPE

- **Command Format**

  [:SOURce]:LOWF:TYPE {{0 | SINE} | {1 | SQUAre} | {2 | PULSe} | {3 | RAMP} | {4 | ARB} | {5 | DC} | {6 | NOISe}}

  [:SOURce]:LOWF:TYPE?

- **Functional Description**

  Set the carrier wave type.

- **Return Format**

  The query returns the carrier wave type.

- **For Example**

:SOURce:LOWF:TYPE SINE	Set the carrier wave type to sine wave.
:SOURce:LOWF:TYPE?	The query returns SINE

:LFOutput:LOAD:IMPedance

- **Command Format**

  [:SOURce]:LFOutput:LOAD:IMPedance {50 | 600 | 1000000}

  [:SOURce]:LFOutput:LOAD:IMPedance?

- **Functional Description**

  Set the load for low-frequence output to 50 Ω, 600 Ω, or high resistance.

- **Return Format**

  The query returns the load of low-frequence output: 50, 600, or 1000000.

- **For Example**

:SOURce:LFOutput:LOAD:IMPedance 50	Set the load of low-frequence output to 50 Ω.
:SOURce:LFOutput:LOAD:IMPedance?	The query returns 50.

:LOWF:FREQ

- **Command Format**

[:SOURce]:LOWF:FREQ <value><unit>

[:SOURce]:LOWF:FREQ?

- **Functional Description**

  Set the frequency for low-frequence output.

- **Return Format**

  The query returns the frequency of low-frequence output, with the unit in Hz.

- **For Example**

:SOURce:LOWF:FREQ 10kHz	Set the frequency of low-frequence output to 10 kHz.
:SOURce:LOWF:FREQ?	The query returns 1.000000e+04.

:LFOutput:AMPLitude

- **Command Format**

  [:SOURce]:LFOutput:AMPLitude <value><unit>

  [:SOURce]:LFOutput:AMPLitude?

- **Functional Description**

  Set the amplitude for low-frequence output, with the unit specified as Vpp, mVpp, Vrms, or mVrms.

- **Return Format**

  The query returns the amplitude of low-frequence output, with the unit specified as Vpp, mVpp, Vrms, or mVrms.

- **For Example**

:SOURce:LFOutput:AMPLitude 1Vpp	Set the amplitude of low-frequence output to 1 Vpp.
:SOURce:LFOutput:AMPLitude?	The query returns 1 Vpp.

:LOWF:PHASe

- **Command Format**

  [:SOURce]:LOWF:PHASe <value><unit>

  [:SOURce]:LOWF:PHASe?

- **Functional Description**

  Set the phase for low-frequence output, with the unit specified as either degrees (deg) or radians (rad).

- **Return Format**

  The query returns the phase of low-frequence output, with the unit in degrees (deg).

- **For Example**

:SOURce:LOWF:PHASe 30deg	Set the phase of low-frequence output to 30 deg.
:SOURce:LOWF:PHASe?	The query returns 30 deg.

:LFOutput:OFFset

- **Command Format**

  [:SOURce]:LFOutput:OFFset <value><unit>

  [:SOURce]:LFOutput:OFFset?

- **Functional Description**

  Set the DC amplitude offset for low-frequence output, with the unit specified as either V or mV.

- **Return Format**

  The query returns the DC amplitude offset, with the unit in mV.

- **For Example**

:SOURce:LFOutput:OFFset 100mV	Set the DC amplitude offset to 100 mV.
:SOURce:LFOutput:OFFset?	The query returns 100 mV.

:LOWF:DUTY

- **Command Format**

  [:SOURce]:LOWF:DUTY <value>

  [:SOURce]:LOWF:DUTY?

- **Functional Description**

  Set the duty cycle for a square wave or pulse wave.

- **Return Format**

  The query returns the duty cycle, with the unit in %.

- **For Example**

[:SOURce]:LOWF:DUTY 20	Set the duty cycle of a square wave or pulse wave to 20%.
[:SOURce]:LOWF:DUTY?	The query returns 20.

:LOWF:SYMMetry

- **Command Format**

  [:SOURce]:LOWF:SYMMetry <value>

  [:SOURce]:LOWF:SYMMetry?

- **Functional Description**

  Set the symmetry for triangular wave.

- **Return Format**

The query returns the symmetry of triangular wave, with the unit in %.

- **For Example**

[:SOURce]:LOWF:SYMMetry 20	Set the symmetry of triangular wave to 20%.
[:SOURce]:LOWF:SYMMetry?	The query returns 20.

:LOWF:RISEedge

- **Command Format**

[:SOURce]:LOWF:RISEedge<value><unit>

[:SOURce]:LOWF:RISEedge?

- **Functional Description**

Set the rising time for pulse wave edge, with the unit specified as s, ms, us, or ns.

- **Return Format**

The query returns the rising time of pulse wave edge, with the unit in seconds (s).

- **For Example**

:SOURce:LOWF:RISEedge20ns	Set the rising time of edge to 20 ns.
:SOURce:LOWF:RISEedge?	The query returns 2.000000e-08.

:LOWF:FALLedge

- **Command Format**

[:SOURce]:LOWF:FALLedge <value><unit>

[:SOURce]:LOWF:FALLedge?

- **Functional Description**

Set the falling time for pulse wave edge, with the unit specified as s, ms, us, or ns.

- **Return Format**

The query returns the falling time of pulse wave edge, with the unit in seconds (s).

- **For Example**

:SOURce:LOWF:FALLedge 20ns	Set the falling time of edge to 20 ns.
:SOURce:LOWF:FALLedge?	The query returns 2.000000e-08.

:LOWF:NOISebw

- **Command Format**

[:SOURce]:LOWF:NOISebw <value><unit>

[:SOURce]:LOWF:NOISebw?

- **Functional Description**

Set the noise bandwidth, with the unit specified as Hz, kHz, MHz, or GHz.

- **Return Format**

  The query returns the noise bandwidth, with the unit in Hz.

- **For Example**

:SOURce:LOWF:NOISebw 10kHz	Set the noise bandwidth to 10 kHz.
:SOURce:LOWF:NOISebw?	The query returns 1.000000e+04.

:LOWF:NOISeamp

- **Command Format**

  [:SOURce]:LOWF:NOISeamp <value><unit>

  [:SOURce]:LOWF:NOISeamp?

- **Functional Description**

  Set the amplitude of noise, with the unit specified as mV or V.

- **Return Format**

  The query returns the amplitude value along with its unit.

- **For Example**

:SOURce:LOWF:NOISeamp 10mV	Set the amplitude of noise to 10 mV.
:SOURce:LOWF:NOISeamp?	The query returns 10 mV.

:LOWF:MOD:EN

- **Command Format**

  [:SOURce]:LOWF:MOD:EN {{1|ON} | {0|OFF}}

  [:SOURce]:LOWF:MOD:EN?

- **Functional Description**

  Set the low-frequency modulation to ON (1) or OFF (0).

- **Return Format**

  The query returns the state of low-frequency modulation: 0 or 1.

- **For Example**

:SOURce:LOWF:MOD:EN ON	Enable the low-frequency modulation.
:SOURce:LOWF:MOD:EN?	The query returns 1.

:LOWF:MOD:TYPE

- **Command Format**

  [:SOURce]:LOWF:MOD:TYPE {{0 | AM} | {1 | FM} | {2 | PHM} | {3 | PM} | {4 | ASK} | {5 | FSK} | {6

| PSK} | {7 | QAM}}

[:SOURce]:LOWF:MOD:TYPE?

- **Functional Description**

Set the low-frequency modulation type: 0 (AM), 1 (FM), 2 (PHM), 3 (PM), 4 (ASK), 5 (FSK), 6 (PSK), or 7 (QAM).

- **Return Format**

The query returns the modulation type of low-frequency modulation: 0, 1, 2, 3, 4, 5, 6, or 7.

- **For Example**

:SOURce:LOWF:MOD:TYPE AM	Set the low-frequency modulation type to AM.
:SOURce:LOWF:MOD:TYPE?	The query returns 0.

:LOWF:MOD:WAVE

- **Command Format**

[:SOURce]:LOWF:MOD:WAVE {{0 | SINE} | {1 | SQUAre} | {2 | RAMP} | {3 | ARB}}

[:SOURce]:LOWF:MOD:WAVE?

- **Functional Description**

Set the modulation wave for low-frequency modulation: 0 (SINE), 1 (SQUAre), 2 (RAMP), or 3 (ARB).

- **Return Format**

The query returns the modulation wave of low-frequency modulation: 0, 1, 2, or 3.

- **For Example**

:SOURce:LOWF:MOD:WAVE SINE

Set the modulation wave of low-frequency modulation to SINE.

:SOURce:LOWF:MOD:WAVE?	The query returns 0.

:LOWF:MOD:FREQ

- **Command Format**

[:SOURce]:LOWF:MOD:FREQ <value><unit>

[:SOURce]:LOWF:MOD:FREQ?

- **Functional Description**

Set the modulation frequency for low-frequency modulation.

- **Return Format**

The query returns the modulation frequency of low-frequency modulation, with the unit in Hz.

- **For Example**

:SOURce:LOWF:MOD:FREQ 10kHz

Set the modulation frequency of low-frequency modulation to 10 kHz.

:SOURce:LOWF:MOD:FREQ? The query returns 1.000000e+04.

:LOWF:MOD:DEPTh

- **Command Format**

  [:SOURce]:LOWF:MOD:DEPTh <value>

  [:SOURce]:LOWF:MOD:DEPTh?

- **Functional Description**

  Set the modulation depth for low-frequency modulation, with the range from 0 to 120%.

- **Return Format**

  The query returns the modulation depth of low-frequency modulation, without specifying a unit.

- **For Example**

  :SOURce:LOWF:MOD:DEPTh 50

  Set the modulation depth of low-frequency modulation to 50%.

  :SOURce:LOWF:MOD:DEPTh? The query returns 50.

:LOWF:MOD:FREQdev

- **Command Format**

  [:SOURce]:LOWF:MOD:FREQdev <value><unit>

  [:SOURce]:LOWF:MOD:FREQdev?

- **Functional Description**

  Set the frequency offset for frequency modulation.

- **Return Format**

  The query returns the frequency offset, with the unit in Hz.

- **For Example**

  :SOURce:LOWF:MOD:FREQdev 10kHz Set the frequency offset to 10 kHz.

  :SOURce:LOWF:MOD:FREQdev? The query returns 1.000000e+04.

:LOWF:MOD:PHASe1

- **Command Format**

  [:SOURce]:LOWF:MOD:PHASe1 <value><unit>

  [:SOURce]:LOWF:MOD:PHASe1?

- **Functional Description**

  Set the phase for phase modulation or phase shift keying 1, with the unit specified as either

degrees (deg) or radians (rad).

- **Return Format**

  The query returns the phase value along with its unit.

- **For Example**

:SOURce:LOWF:MOD:PHASe1 30deg	Set the phase 1 to 30 deg.
:SOURce:LOWF:MOD:PHASe1?	The query returns 30 deg.

:LOWF:MOD:PHASe2

- **Command Format**

  [:SOURce]:LOWF:MOD:PHASe2 <value><unit>

  [:SOURce]:LOWF:MOD:PHASe2?

- **Functional Description**

  Set the phase for phase shift keying 2, with the unit specified as either degrees (deg) or radians (rad).

- **Return Format**

  The query returns the phase value along with its unit.

- **For Example**

:SOURce:LOWF:MOD:PHASe2 30deg	Set the phase 2 to 30 deg.
:SOURce:LOWF:MOD:PHASe2?	The query returns 30 deg.

:LOWF:MOD:PHASe3

- **Command Format**

  [:SOURce]:LOWF:MOD:PHASe3 <value><unit>

  [:SOURce]:LOWF:MOD:PHASe3?

- **Functional Description**

  Set the phase for phase shift keying 3, with the unit specified as either degrees (deg) or radians (rad).

- **Return Format**

  The query returns the phase value along with its unit.

- **For Example**

:SOURce:LOWF:MOD:PHASe3 30deg	Set the phase 3 to 30 deg.
:SOURce:LOWF:MOD:PHASe3?	The query returns 30 deg.

:LOWF:MOD:PHASe4

- **Command Format**

  [:SOURce]:LOWF:MOD:PHASe4 <value><unit>

  [:SOURce]:LOWF:MOD:PHASe4?

- **Functional Description**

  Set the phase for phase shift keying 4, with the unit specified as either degrees (deg) or radians (rad).

- **Return Format**

  The query returns the phase value along with its unit.

- **For Example**

:SOURce:LOWF:MOD:PHASe4 30deg	Set the phase 4 to 30 deg.
:SOURce:LOWF:MOD:PHASe4?	The query returns 30 deg.

:LOWF:MOD:PMFReq

- **Command Format**

  [:SOURce]:LOWF:MOD:PMFReq <value><unit>

  [:SOURce]:LOWF:MOD:PMFReq?

- **Functional Description**

  Set the pulse frequency for pulse modulation.

- **Return Format**

  The query returns the pulse frequency, with the unit in Hz.

- **For Example**

:SOURce:LOWF:MOD:PMFReq 10kHz	Set the pulse frequency of pulse modulation to 10 kHz.
:SOURce:LOWF:MOD:PMFReq?	The query returns 1.000000e+04.

:LOWF:MOD:PMDUty

- **Command Format**

  [:SOURce]:LOWF:MOD:PMDUty <value>

  [:SOURce]:LOWF:MOD:PMDUty?

- **Functional Description**

  Set the pulse wave duty cycle for pulse modulation.

- **Return Format**

  The query returns the pulse wave duty cycle of pulse modulation, with the unit in %.

- **For Example**

:SOURce:LOWF:MOD:PMDUty 50	Set the pulse wave duty cycle of pulse modulation to 50%.
:SOURce:LOWF:MOD:PMDUty?	The query returns 50.

:LOWF:MOD:RATE

- **Command Format**

  [:SOURce]:LOWF:MOD:RATE <value><unit>

  [:SOURce]:LOWF:MOD:RATE?

- **Functional Description**

  Set the modulation rate for digital modulation.

- **Return Format**

  The query returns the modulation rate of digital modulation, with the unit in Hz.

- **For Example**

:SOURce:LOWF:MOD:RATE 10kHz	Set the modulation rate of digital modulation to 10 kHz.
:SOURce:LOWF:MOD:RATE?	The query returns 1.000000e+04.

:LOWF:HOP:FREQ1

- **Command Format**

  [:SOURce]:LOWF:HOP:FREQ1 <value><unit>

  [:SOURce]:LOWF:HOP:FREQ1?

- **Functional Description**

  Set the hopping frequency 1 for frequency shift keying.

- **Return Format**

  The query returns the hopping frequency 1 of frequency shift keying, with the unit in Hz.

- **For Example**

:SOURce:LOWF:HOP:FREQ1 10kHz	Set the hopping frequency 1 of frequency shift keying to 10 kHz.
:SOURce:LOWF:HOP:FREQ1?	The query returns 1.000000e+04.

:LOWF:HOP:FREQ2

- **Command Format**

  [:SOURce]:LOWF:HOP:FREQ2 <value><unit>

[:SOURce]:LOWF:HOP:FREQ2?

- **Functional Description**

  Set the hopping frequency 2 for frequency shift keying.

- **Return Format**

  The query returns the hopping frequency 2 of frequency shift keying, with the unit in Hz.

- **For Example**

:SOURce:LOWF:HOP:FREQ2 10kHz	Set the hopping frequency 2 of frequency shift keying to 10 kHz.
:SOURce:LOWF:HOP:FREQ2?	The query returns 1.000000e+04.

:LOWF:HOP:FREQ3

- **Command Format**

  [:SOURce]:LOWF:HOP:FREQ3 <value><unit>

  [:SOURce]:LOWF:HOP:FREQ3?

- **Functional Description**

  Set the hopping frequency 3 for frequency shift keying.

- **Return Format**

  The query returns the hopping frequency 3 of frequency shift keying, with the unit in Hz.

- **For Example**

:SOURce:LOWF:HOP:FREQ3 10kHz	Set the hopping frequency 3 of frequency shift keying to 10 kHz.
:SOURce:LOWF:HOP:FREQ3?	The query returns 1.000000e+04.

:LOWF:MOD:FSKMode

- **Command Format**

  [:SOURce]:LOWF:MOD:FSKMode {{0 | 2FSK} | {1 | 4FSK}}

  [:SOURce]:LOWF:MOD:FSKMode?

- **Functional Description**

  Set the frequency shift keying mode.

- **Return Format**

  The query returns the frequency shift keying mode: 0 (2FSK) or 1 (4FSK), without specifying a unit.

- **For Example**

:SOURce:LOWF:MOD:FSKMode 2FSK	Set the frequency shift keying mode to 2FSK.

:SOURce:LOWF:MOD:FSKMode? The query returns 2FSK.

:LOWF:MOD:PSKMode

- **Command Format**

  [:SOURce]:LOWF:MOD:PSKMode {{0 | 2PSK} | {1 | 4PSK}}

  [:SOURce]:LOWF:MOD:PSKMode?

- **Functional Description**

  Set the phase shift keying mode.

- **Return Format**

  The query returns the phase shift keying mode: 0 (2PSK) or 1 (4PSK), without specifying a unit.

- **For Example**

  :SOURce:LOWF:MOD:PSKMode 2PSK Set the phase shift keying mode to 2PSK.

  :SOURce:LOWF:MOD:PSKMode? The query returns 2PSK.

:LOWF:MOD:SOURce

- **Command Format**

  [:SOURce]:LOWF:MOD:SOURce {{0 | PN7} | {1 | PN9} | {2 | PN11} | {3 | PN15} | {4 | PN17} | {5 | PN21} | {6 | PN23} | {7 | PN25}}

  [:SOURce]:LOWF:MOD:SOURce?

- **Functional Description**

  Set the symbol mode.

- **Return Format**

  The query returns the symbol mode: 0 (PN7), 1 (PN9), 2 (PN11), 3 (PN15), 4 (PN17), 5 (PN21), 6 (PN23), or 7 (PN25), without specifying a unit.

- **For Example**

  :SOURce:LOWF:MOD:SOURce PN7 Set the symbol mode to PN7.

  :SOURce:LOWF:MOD:SOURce? The query returns 0.

:LOWF:MOD:QAMMode

- **Command Format**

  [:SOURce]:LOWF:MOD:QAMMode {{0 | QAM4} | {1 | QAM8} | {2 | QAM16} | {3 | QAM32} | {4 | QAM64} | {5 | QAM128} | {6 | QAM256}}

  [:SOURce]:LOWF:MOD:QAMMode?

- **Functional Description**

Set the QAM mode.

- **Return Format**

  The query returns the QAM mode: 0 (QAM4), 1 (QAM8), 2 (QAM16), 3 (QAM32), 4 (QAM64), 5 (QAM128), or 6 (QAM256), without specifying a unit.

- **For Example**

:SOURce:LOWF:MOD:QAMMode QAM4	Set the QAM mode to QAM4.
:SOURce:LOWF:MOD:QAMMode?	The query returns 0.

:LOWF:SWEEp:EN

- **Command Format**

  [:SOURce]:LOWF:SWEEp:EN {{1|ON} | {0|OFF}}

  [:SOURce]:LOWF:SWEEp:EN?

- **Functional Description**

  Set the sweep function to ON (1) or OFF (2).

- **Return Format**

  The query returns the state of sweep function: 0 or 1.

- **For Example**

:SOURce:LOWF:SWEEp:EN ON	Enable the sweep function.
:SOURce:LOWF:SWEEp:EN?	The query returns 1.

:LOWF:SWEEp:MODE

- **Command Format**

  [:SOURce]:LOWF:SWEEp:MODE {{0 | LINEar} | {1 | LOG} | {2 | STEP}}

  [:SOURce]:LOWF:SWEEp:MODE?

- **Functional Description**

  Set the sweep mode.

- **Return Format**

  The query returns the sweep mode: 0 (LINEar), 1 (LOG), or 2 (STEP).

- **For Example**

:SOURce:LOWF:SWEEp:MODE LINEar	Set the sweep mode to LINEar.
:SOURce:LOWF:SWEEp:MODE?	The query returns 0.

:LOWF:SWEEp:SHAP

- **Command Format**

[:SOURce]:LOWF:SWEEp:SHAP {{0 | POSSAW} | {1 | NEGSAW} | {2 | POSTRI} | {3 | NEGTRI}}

[:SOURce]:LOWF:SWEEp:SHAP?

- **Functional Description**

  Set the sweep shape to positive sawtooth, negative sawtooth, positive triangle, or negative triangle.

- **Return Format**

  The query returns the sweep shape: 0 (POSSAW), 1 (NEGSAW), 2 (POSTRI), or 3 (NEGTRI).

- **For Example**

:SOURce:LOWF:SWEEp:SHAP POSSAW	Set the sweep shape to POSSAW.
:SOURce:LOWF:SWEEp:SHAP?	The query returns POSSAW.

:LOWF:SWEEp:TRIGout

- **Command Format**

  [:SOURce]:LOWF:SWEEp:TRIGout {{0 | CLOSe} | {1 | RISEedge} | {2 | FALLedge}}

  [:SOURce]:LOWF:SWEEp:TRIGout?

- **Functional Description**

  Set the mode of sweep trigger output to close, rising edge, or falling edge.

- **Return Format**

  The query returns the mode of sweep trigger output: 0 (CLOSe), 1 (RISEedge), or 3 (FALLedge).

- **For Example**

:SOURce:LOWF:SWEEp:TRIGout CLOSe	Set the mode of sweep trigger output to close.
:SOURce:LOWF:SWEEp:TRIGout?	The query returns CLOSe.

:LOWF:SWEEp:TRIGin

- **Command Format**

  [:SOURce]:LOWF:SWEEp:TRIGin {{0 | AUTO} | {1 | NKEY} | {2 | BUS} | {3 | EXT}}

  [:SOURce]:LOWF:SWEEp:TRIGin?

- **Functional Description**

  Set the mode of sweep trigger input to auto, key, bus, or external.

- **Return Format**

  The query returns the mode of sweep trigger input: 0 (AUTO), 1 (NKEY), 2 (BUS), or 3 (EXT).

- **For Example**

:SOURce:LOWF:SWEEp:TRIGin AUTO	Set the mode of sweep trigger input to auto.
:SOURce:LOWF:SWEEp:TRIGin?	The query returns AUTO.

:LOWF:SWEEp:TRIGedge

- **Command Format**

  [:SOURce]:LOWF:SWEEp:TRIGedge {{0| RISEedge} | {1 | FALLedge}}

  [:SOURce]:LOWF:SWEEp:TRIGedge?

- **Functional Description**

  Set the edge of the sweep external trigger output to rising or falling.

- **Return Format**

  The query returns the edge of the sweep external trigger output: 0 (RISEedge) or 1 (FALLedge).

- **For Example**

:SOURce:LOWF:SWEEp:TRIGedge RISEedge	Set the edge of the sweep external trigger output to rising.
:SOURce:LOWF:SWEEp:TRIGedge ?	The query returns RISEedge.

:LOWF:FREQ:STAR

- **Command Format**

  [:SOURce]:LOWF:FREQ:STARt <value><unit>

  [:SOURce]:LOWF:FREQ:STARt?

- **Functional Description**

  Set the start frequency for sweep.

- **Return Format**

  The query returns the start frequency of sweep, with the unit in Hz.

- **For Example**

:SOURce:LOWF:FREQ:STARt 10kHz	Set the start frequency of sweep to 10 kHz.
:SOURce:LOWF:FREQ:STARt?	The query returns 1.000000e+04.

:LOWF:FREQ:STOP

- **Command Format**

  [:SOURce]:LOWF:FREQ:STOP <value><unit>

  [:SOURce]:LOWF:FREQ:STOP?

- **Functional Description**

  Set the stop frequency for sweep.

- **Return Format**

  The query returns the stop frequency of sweep, with the unit in Hz.

- **For Example**

:SOURce:LOWF:FREQ:STOP 10kHz	Set the stop frequency of sweep to 10 kHz.
:SOURce:LOWF:FREQ:STOP?	The query returns 1.000000e+04.

:LOWF:SWEEp:TIME

- **Command Format**

  [:SOURce]:LOWF:SWEEp:TIME <value><unit>

  [:SOURce]:LOWF:SWEEp:TIME?

- **Functional Description**

  Set the sweep time.

- **Return Format**

  The query returns the sweep time, with the unit in seconds (s).

- **For Example**

:SOURce:LOWF:SWEEp:TIME 2s	Set the sweep time to 2 s.
:SOURce:LOWF:SWEEp:TIME?	The query returns 2.000000e+00.

:LOWF:DWELl:TIME

- **Command Format**

  [:SOURce]:LOWF:DWELl:TIME <value><unit>

  [:SOURce]:LOWF:DWELl:TIME?

- **Functional Description**

  Set the dwell time.

- **Return Format**

  The query returns the dwell time, with the unit in seconds (s).

- **For Example**

:SOURce:LOWF:DWELl:TIME 2s	Set the dwell time to 2 s.
:SOURce:LOWF:DWELl:TIME?	The query returns 2.000000e+00.

:LOWF:STEP:COUNt

- **Command Format**

  [:SOURce]:LOWF:STEP:COUNt <value>

  [:SOURce]:LOWF:STEP:COUNt?

- **Functional Description**

  Set the sweep count without specifying a unit.

- **Return Format**

The query returns the sweep count

- **For Example**

:SOURce:LOWF:STEP:COUNt 5 Set the sweep count to 5.

:SOURce:LOWF:STEP:COUNt? The query returns 5.

:LOWF:SWEEp:BUS:TRIGger

- **Command Format**

[:SOURce]:LOWF:SWEEp:BUS:TRIGger

- **Functional Description**

Set the bus triggering for low-frequency sweep without specifying a parameter.

- **Return Format**

This command does not return a value.

- **For Example**

:SOURce:LOWF:SWEEp:BUS:TRIGger Set the bus triggering for low-frequency sweep to trigger once.

WARB Command

:WARB:CARRier

- **Command Format**

:WARB:CARRier <arb file>

- **Functional Description**

Write the carrier arbitrary waveform. First, send this command, then transmit the arbitrary waveform file to the signal generator.

<arb file> refers to the arbitrary waveform file name, which must be in .bsv format.

- **Return Format**

This command does not return a value.

- **For Example**

:WARB:CARRier test.bsv Write the carrier low-frequency arbitrary waveform file.

:WARB:MODulate

- **Command Format**

:WARB:MODulate <arb file>

- **Functional Description**

Write the modulation arbitrary waveform. First, send this command, then transmit the arbitrary

waveform file to the signal generator.

<arb file> refers to the arbitrary waveform file name, which must be in .bsv format.

- **Return Format**

  This command does not return a value.

- **For Example**

  :WARB:MODulate test.bsv Write the modulation low-frequency arbitrary waveform file.

Modulation Input

:MODIn:STATe

- **Command Format**

  [:SOURce]:MODIn:STATe {{1|ON} | {0|OFF}}

  [:SOURce]:MODIn:STATe?

- **Functional Description**

  Set the modulation input to ON (1) or OFF (1).

- **Return Format**

  The query returns the state of modulation input: 0 or 1.

- **For Example**

  :SOURce:MODIn:STATe ON Enable the modulation input.

  :SOURce:MODIn:STATe? The query returns 1.

:MODIn:LOAD

- **Command Format**

  [:SOURce]:MODIn:LOAD {50 | 600 | 1000000}

  [:SOURce]:MODIn:LOAD?

- **Functional Description**

  Set the load for modulation input to 50 Ω, 600 Ω, or high resistance.

- **Return Format**

  The query returns the load of modulation input: 50, 600, or 1000000.

- **For Example**

  :SOURce:MODIn:LOAD 50 Set the load of modulation input to 50 Ω.

  :SOURce:MODIn:LOAD? The query returns 5.0

DISPlay Command

:DISPlay:DATA?

- **Command Format**

  :DISPlay:DATA?

- **Functional Description**

  Capture the screen image.

- **Return Format**

  The query returns the screen image data.

- **For Example**

  :DISPlay:DATA? Capture the screen image.

Programming Explanation

This chapter describes troubleshooting during the programming process. If the user encounters any of the following problems, please follow the related instructions.

Programming Preparation

Users can remotely control signal generator via USB, GPIB, or LAN interfaces and integrate it with NI-VISA and programming languages. Programming is applicable only when using Visual Studio and LabVIEW development tools under the Windows operating system.

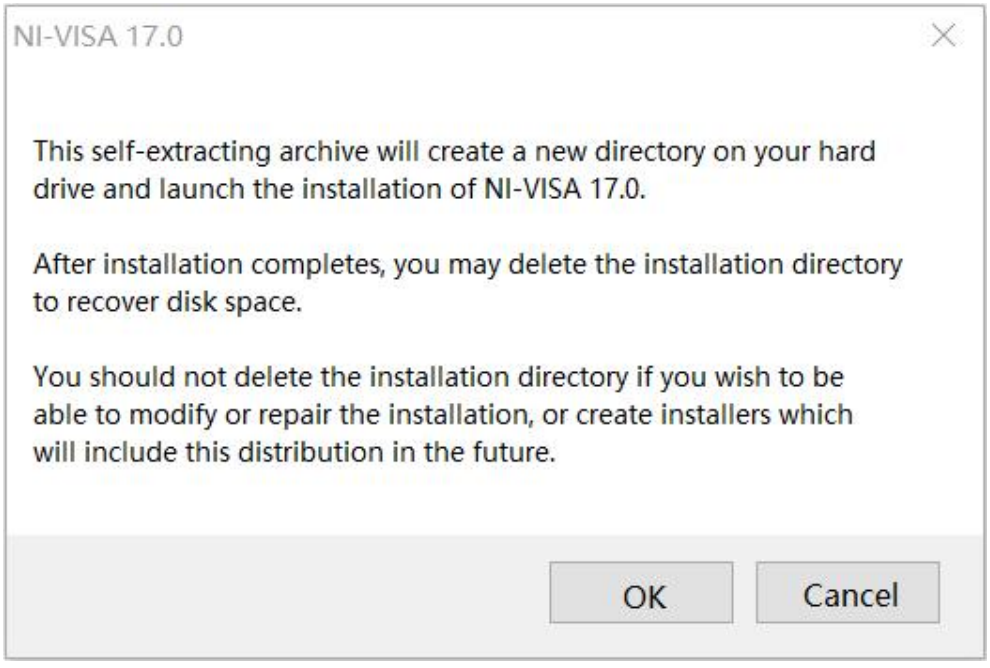
1. Setup Communication

NI-VISA is the communication library for communication between computers and devices. NI software has two valid VISA installation packages: Full version and the Run-Time Engine version. The full version includes the NI device driver and the NI MAX tool, NI MAX is the used to control user interface of the device. While the driver and NI MAX are useful, they are not used for remote control. Run-Time Engine is a smaller file than the full version, and it is primarily used for remote control. You can download the latest NI-VISA Run-Time Engine or the full version from the NI website. The installation steps are the same.

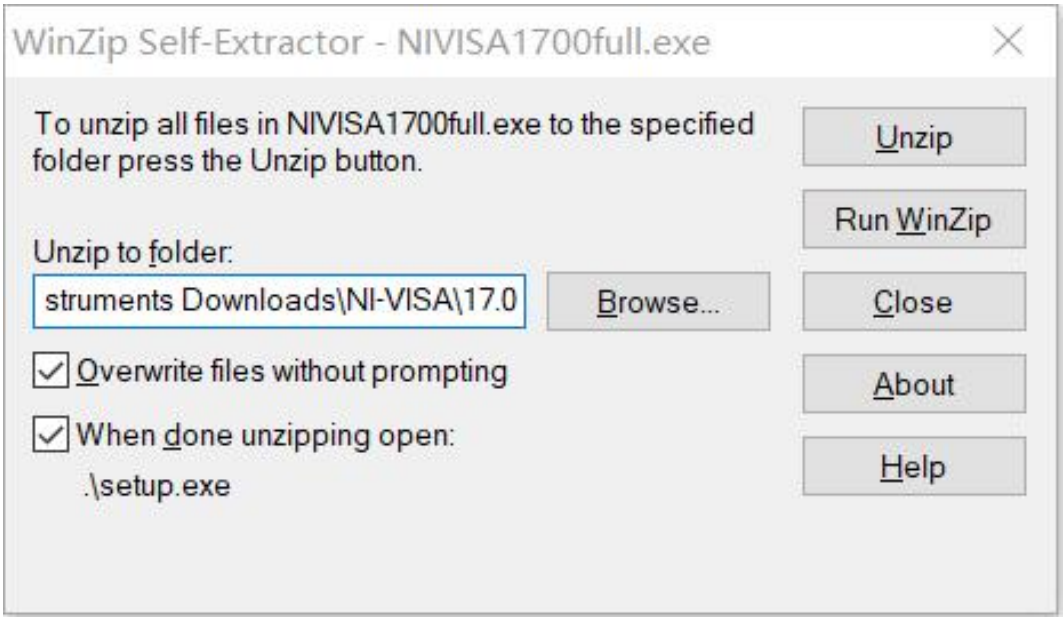
Follow these steps to install NI-VISA (take NI-VISA 17.0 full version as an example).

a. Download the proper version of NI-VISA.

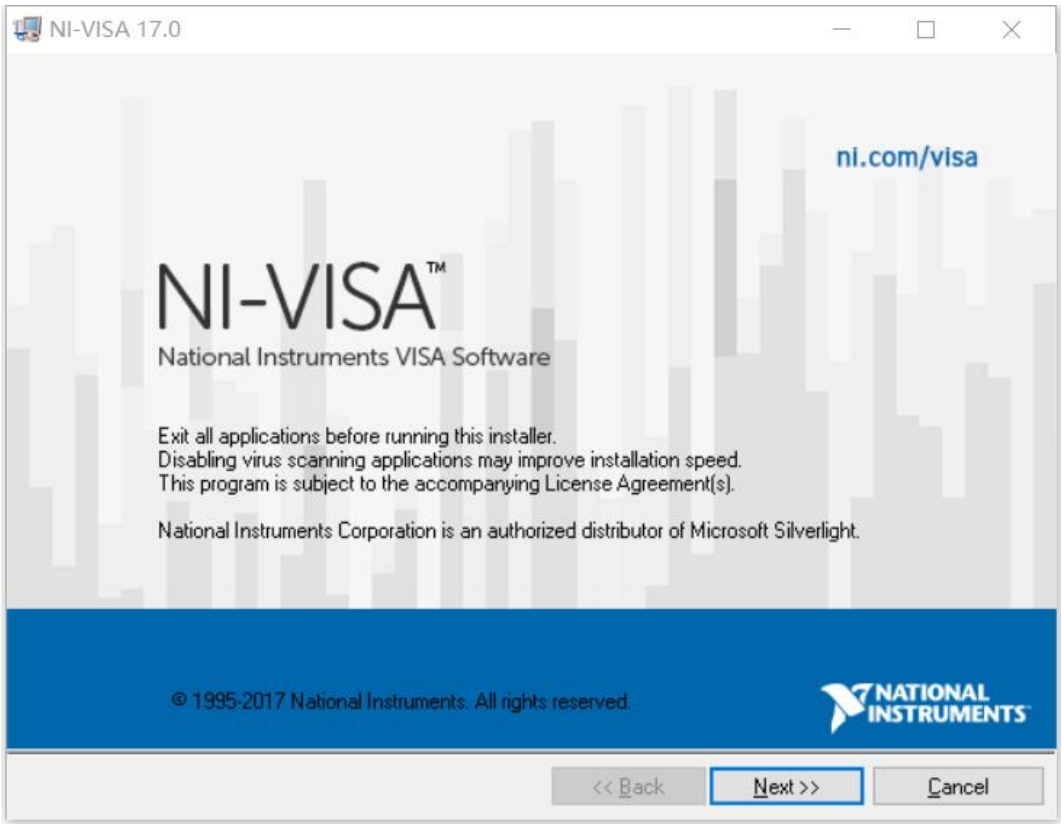
b. Double-click NIVISA1700full.exe to open the dialogue.



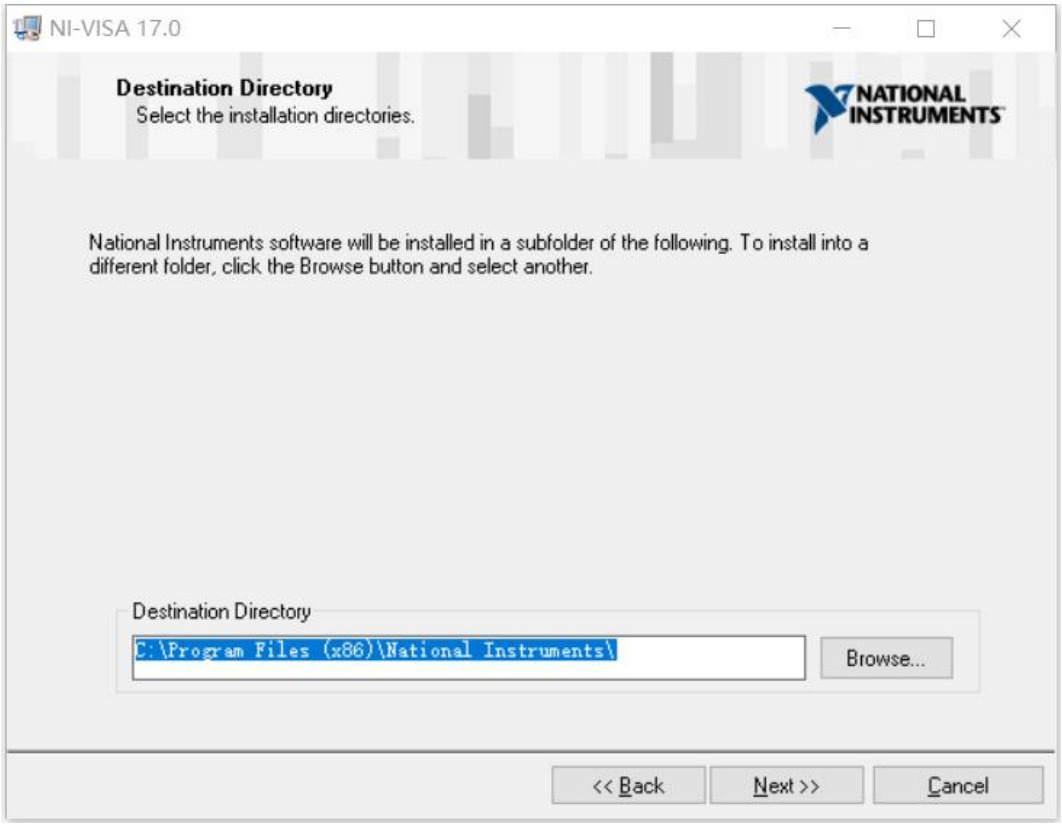
c. Click Yes to open the dialogue.



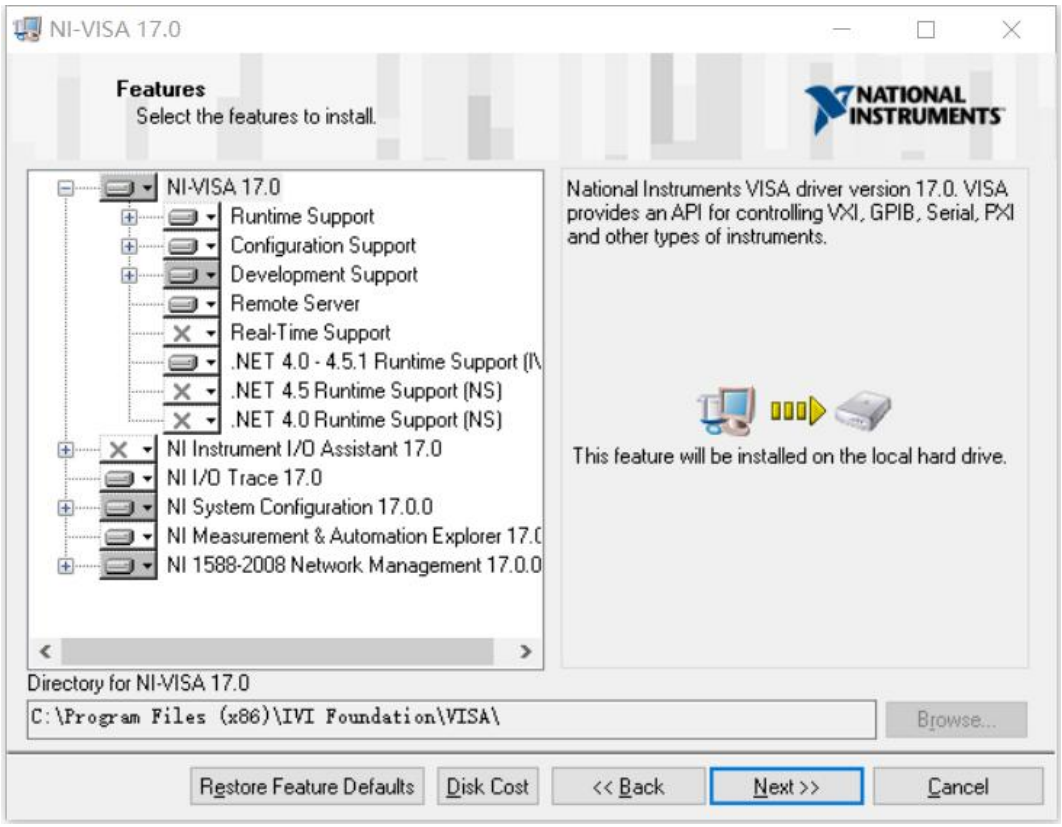
d. Click “Unzip” to decompress the files. After decompression is complete, the installation program will run automatically. If the user's computer requires .NET Framework 4, it will be installed automatically during the installation process.



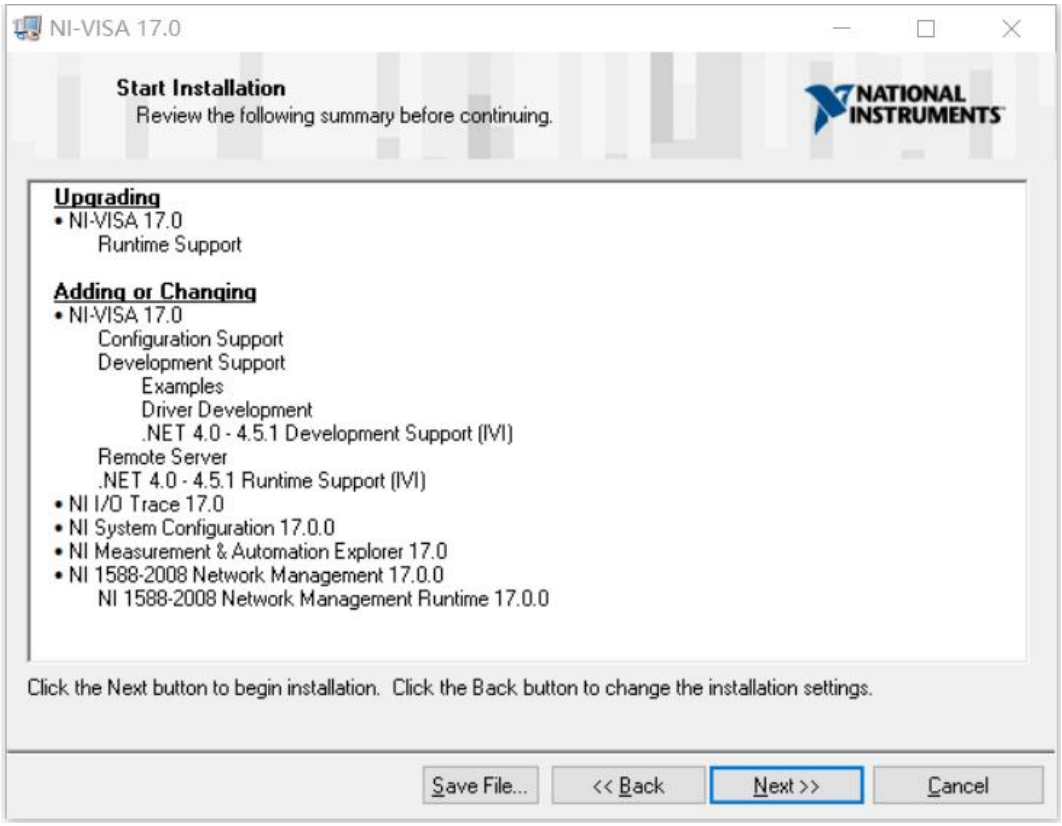
e. Installation dialogue of NI-VISA as shown in the figure above. Click Next to start the installation.



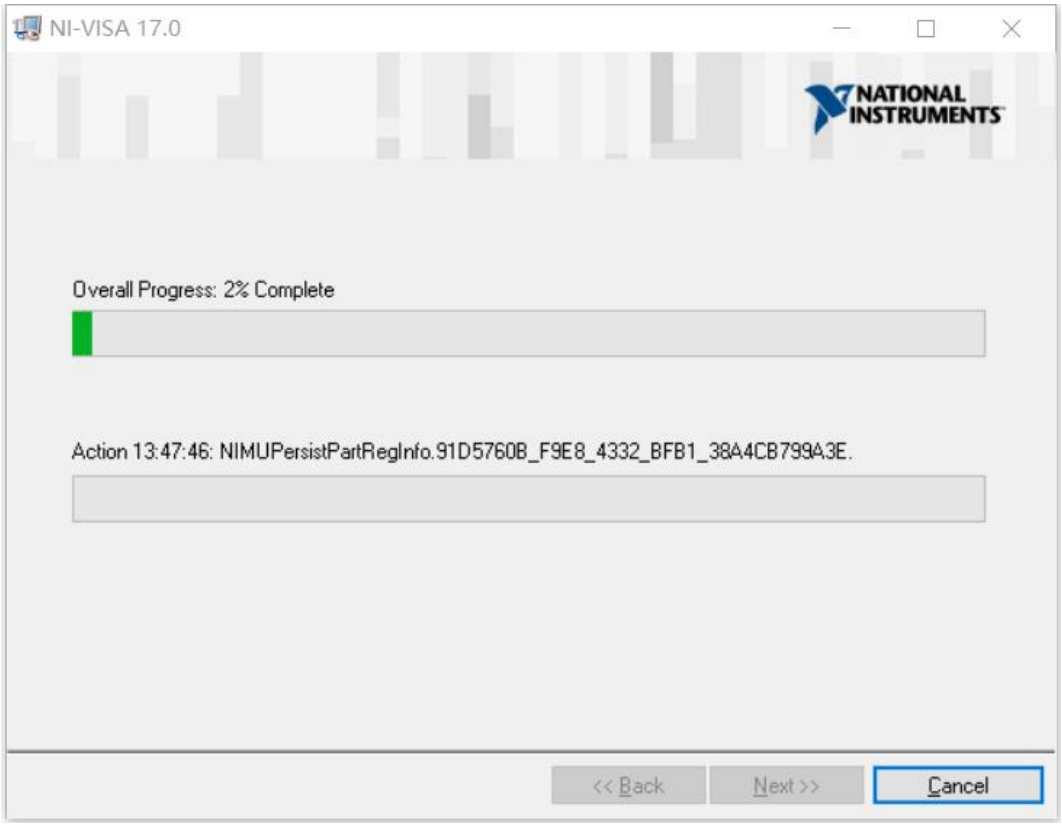
f. Set the installation path, the default path is "C:\Program Files (x86)\National Instruments\", or you can click Next to change the installation path, as shown in the following figure.



g. Double-click Next in the license agreement dialogue and select "I accept the above 2 License Agreement(s)." and then click Next, the dialogue as shown in the following figure.



h. Click Next to start the installation.



j. Restart the computer after the installation is completed.

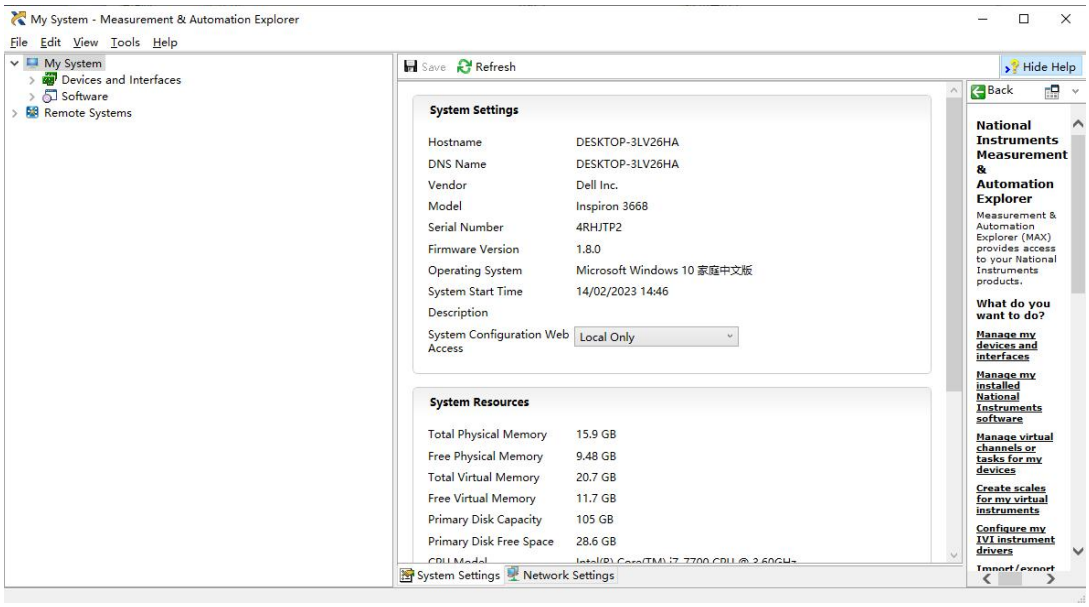
2. Connecting Device

Use the USB method as an example to introduce the connection process.

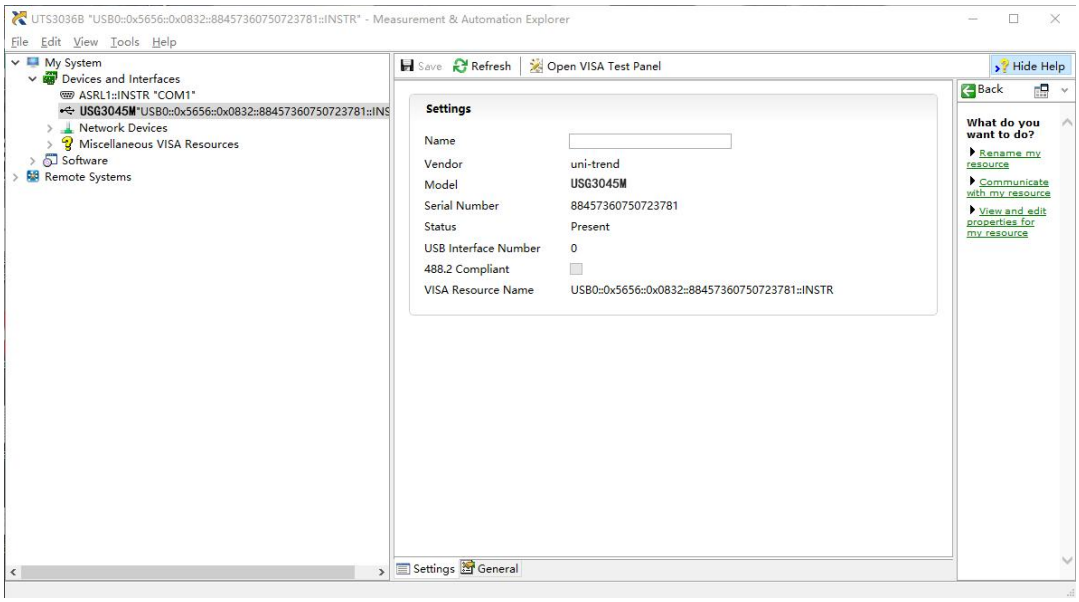
a. Turn on the RF signal generator.

b. Use USB wire to connect the USB Device port of the RF signal generator with USB Host port of the computer, as shown in the following figure.

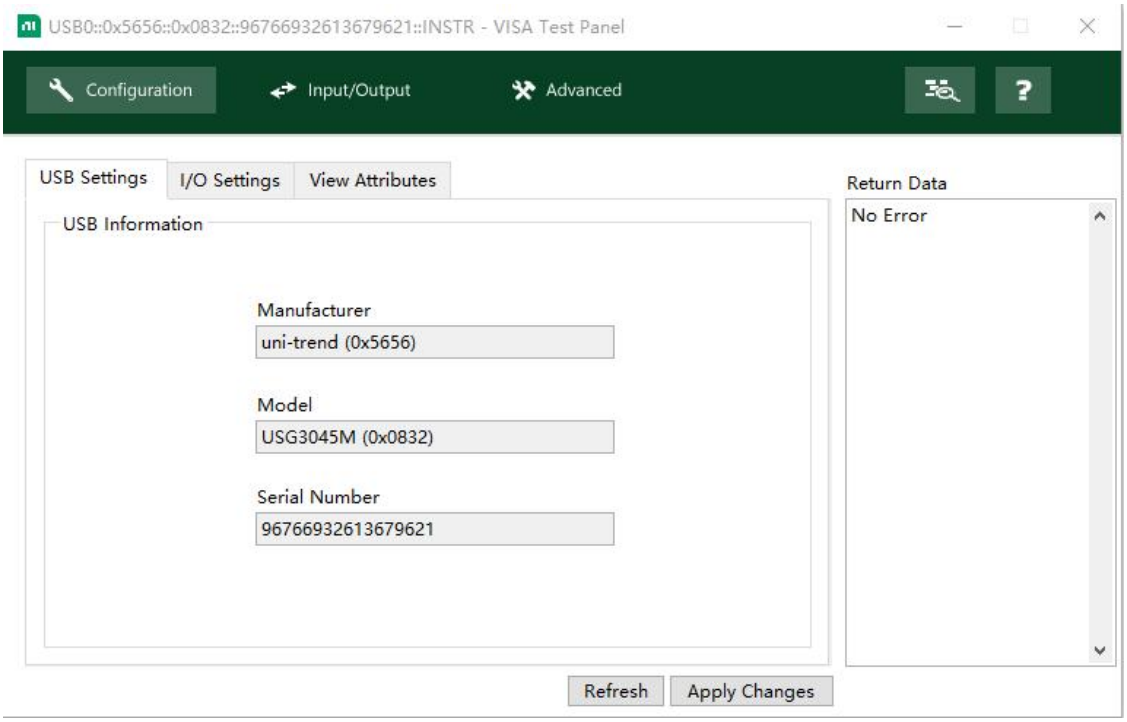
c. Lauch NI MAX on the computer. The dialogue is shown in the following figure.



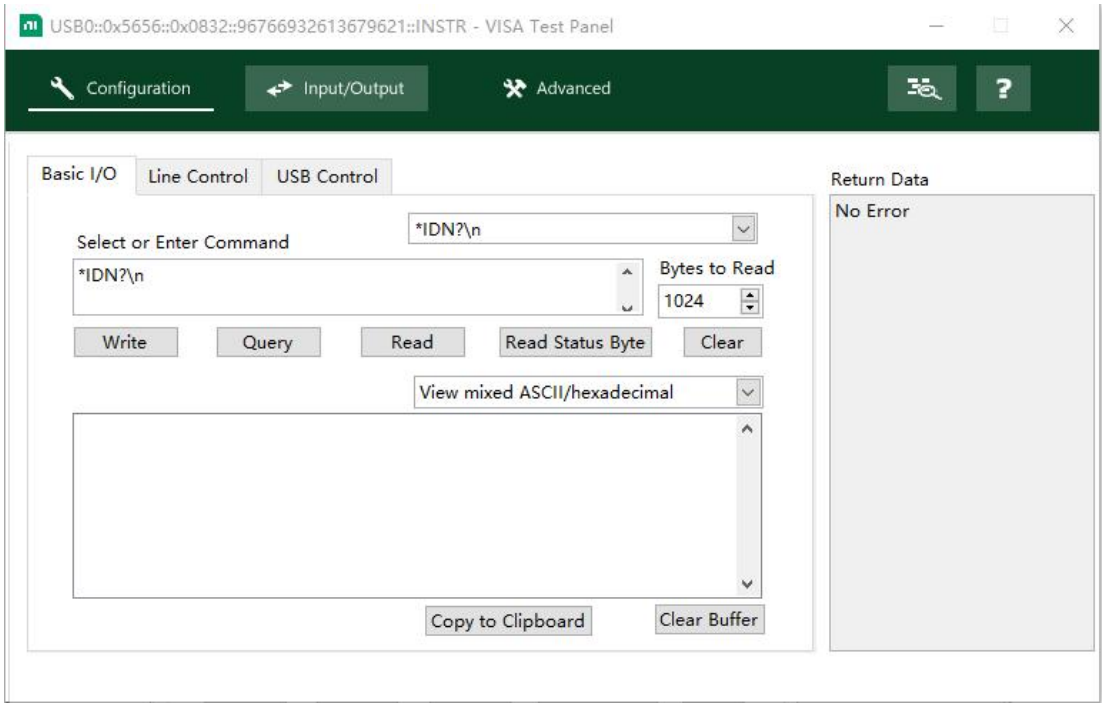
d. Turn on the device and scroll down the options to select the RF signal generator drive, as shown in the following figure.



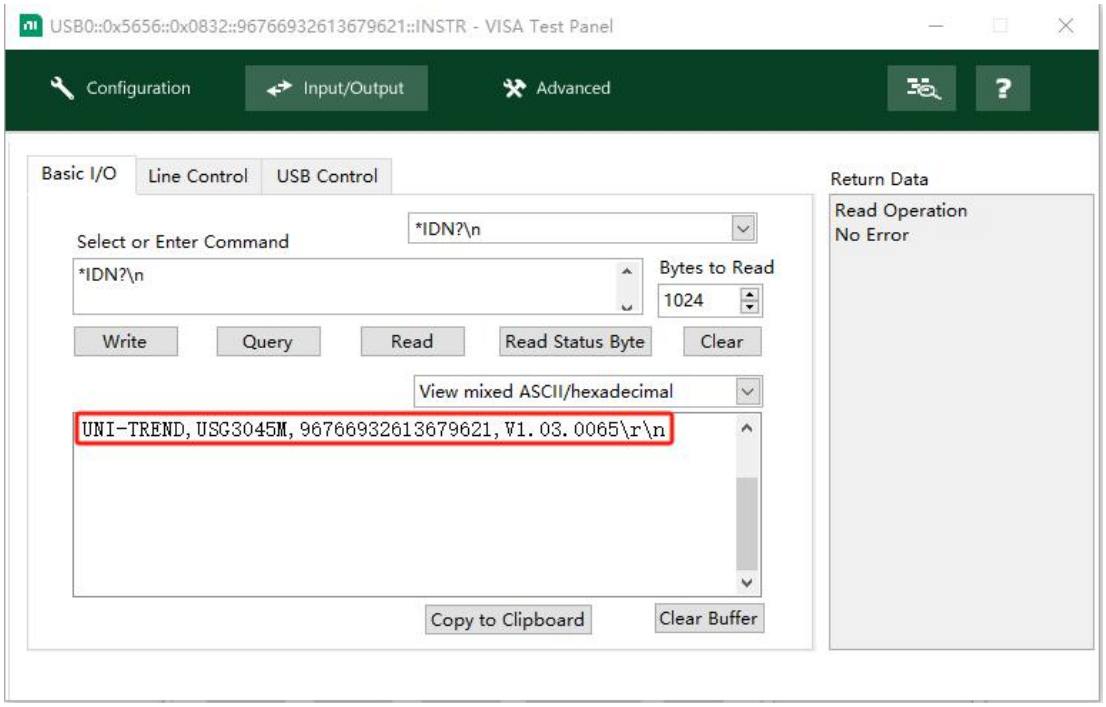
e. Use the mouse to click VISA test panel to open the dialogue as shown in the following figure.



f. Use the mouse to click the option Input/Output, as shown in the following figure.



g. Use the mouse to click Query IDN of the RF signal generator, the query result will display at the red area, as shown in the following figure.



h. If it can query the relevant information of the RF signal generator, it means the RF signal generator is communicating with the computer.

VISA Programming Example

This section contains examples. Through these examples, users can learn how to use VISA and combine it with the commands in the programming manual to control the instrument. With these examples, users 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 Query the device information.
- **Steps**

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

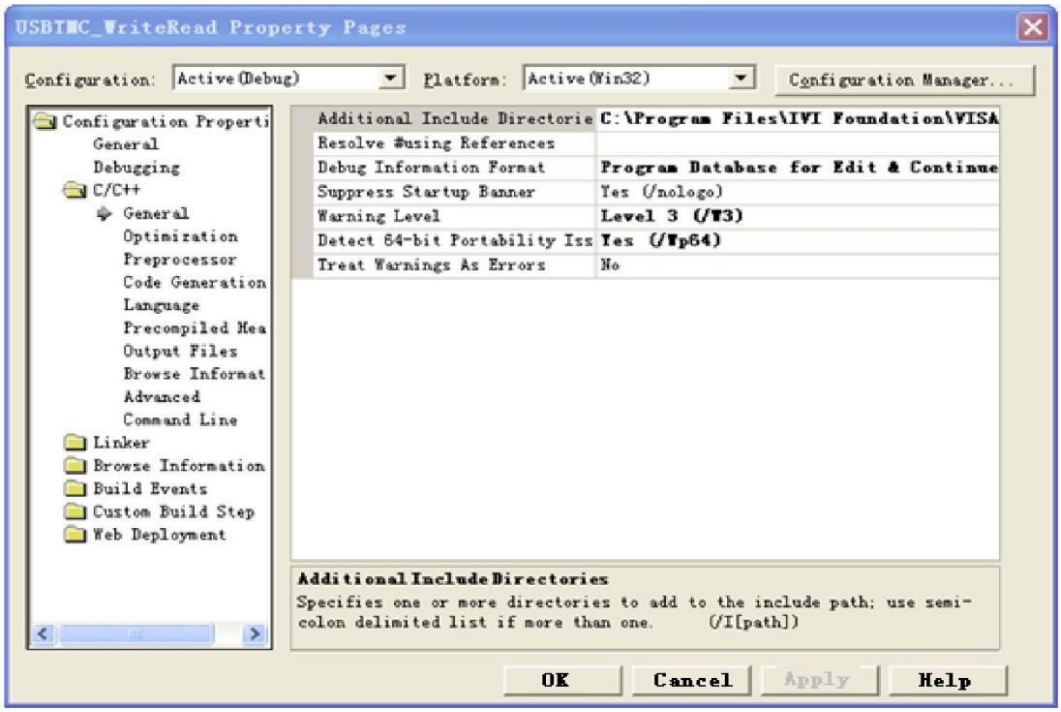
a) Static library

Locate the files “visa.h, visatype.h, and visa32.lib” in the NI-VISA installation path. Copy them to the root directory of the VC++ project and add them to the project. Add the following two lines of code to the projectname.cpp file:

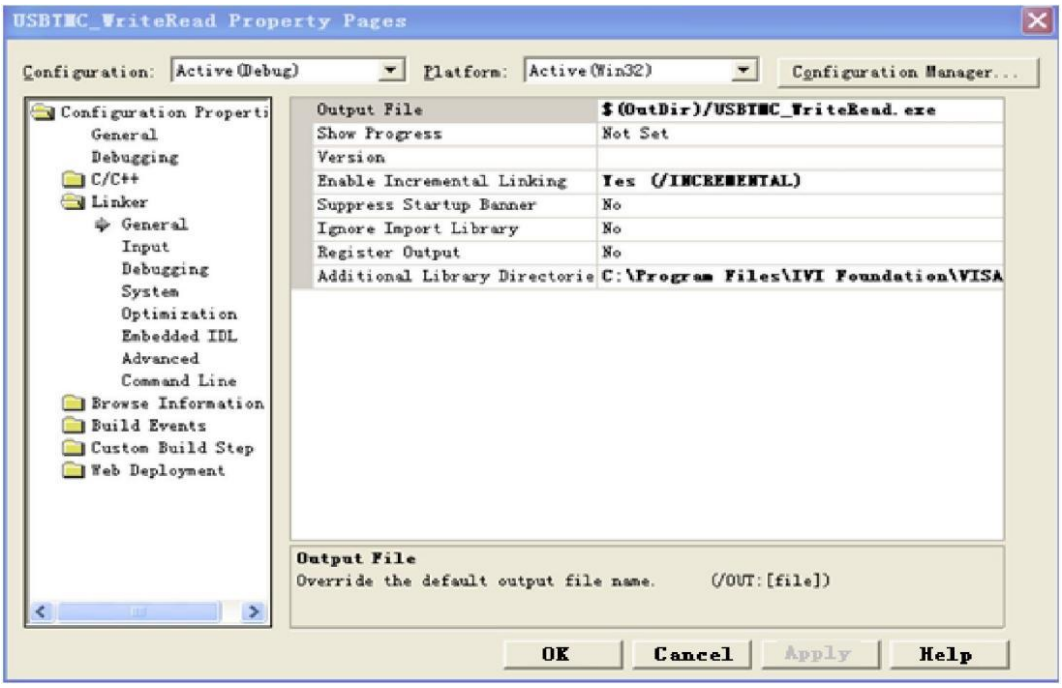
```
#include "visa.h"
#pragma comment(lib,"visa32.lib")
```

b) Dynamic library

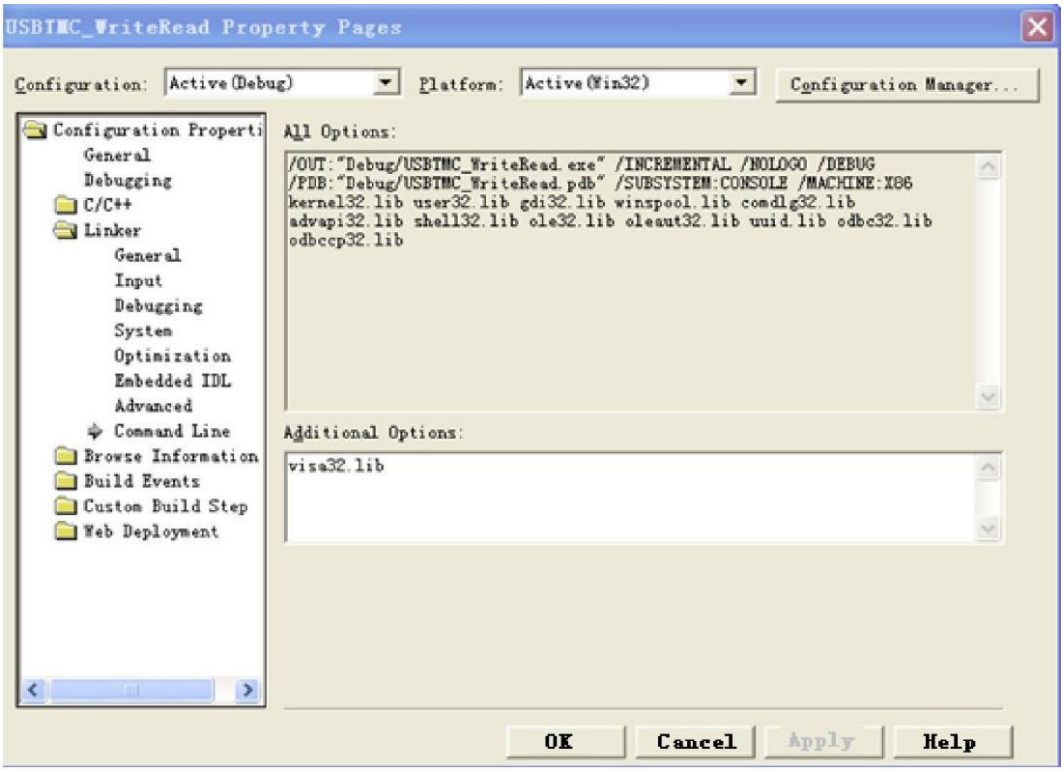
Press “project>>properties”, select "c/c++---General" in the attribute dialog on the left side, and set the value of "Additional Include Directories" to the installment path of NI-VIS (for example, C:\ProgramFiles\IVI Foundation\VISA\WinNT\include), as shown in the following figure.



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



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



Add the file visa.h in the 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;
    ViState state;
    char instrResourceString[VI_FIND_BUFLEN];
    unsigned char buffer[100];
    int i;
    state = viOpenDefaultRM(&defaultRM);
    if (state < VI_SUCCESS)
    {
```

```
        printf("Could not open a session to the VISA Resource Manager!\n");
        return state;
 }
/*Find all the USB TMC VISA resources in our system and store the number of resources in the system in numInstrs.*/
 state = viFindRsrc(defaultRM, "USB?*INSTR", &findList, &numInstrs, instrResourceString);
 if (state<VI_SUCCESS)
 {
        printf("An error occurred while finding resources. \nPress Enter to continue.");
        fflush(stdin);
        getchar();
        viClose(defaultRM);
        return state;
 }
 /** 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);
        }
        state = viOpen(defaultRM, instrResourceString, VI_NULL, VI_NULL, &instr);
        if (state < 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";
        state = viPrintf(instr, cmmand);
        if (state < VI_SUCCESS)
        {
                printf("Error writing to the device %d. \n", i + 1);
                state = 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. */
            state = viScanf(instr, "%t", buffer);
            if (state < VI_SUCCESS)
            {
                printf("Error reading a response from the device %d. \n", i + 1);
            }
            else
            {
                printf("\nDevice %d: %s\n", i + 1, buffer);
            }
            state = viClose(instr);
        }
        /*Now we will close the session to the instrument using viClose. This operation frees all
        system resources.*/
        state = 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;
        ViState state;
        /* First we will need to open the default resource manager. */
        state = viOpenDefaultRM(&defaultRM);
        if (state < 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);
        state = viOpen(defaultRM, head, VI_LOAD_CONFIG, VI_NULL, &instr);
        if (state < VI_SUCCESS)
        {
            printf("An error occurred opening the session\n");
            viClose(defaultRM);
        }
        state = viPrintf(instr, "*idn?\n");
        state = viScanf(instr, "%t", outputBuffer);
        if (state < VI_SUCCESS)
        {
            printf("viRead failed with error code: %x \n", state);
            viClose(defaultRM);
        }
        else
        {
            printf("\nMesseage read from device: %*s\n", 0, outputBuffer);
        }
        state = viClose(instr);
        state = 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 Query the device information.
- **Steps**

1. Open the 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

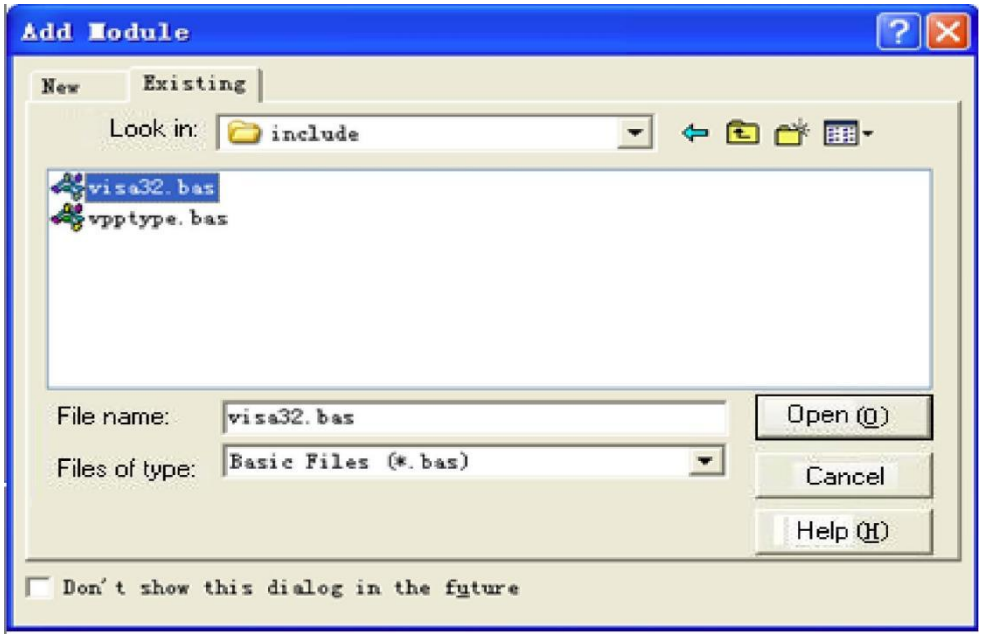
```
class Program
  {
    void tcp_ip_test(string ip)
    {
        using (var rmSession = new ResourceManager())
        {
            try
            {
                var resource = string.Format("TCPIP0::{0}::inst0::INSTR", ip);
                var mbSession = (MessageBasedSession)rmSession.Open(resource);
                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 the Visual Basic software and create a new standard application program project.
2. Set the project environment that can adjust NI-VISA library, press “Existing tab of Project>>Add Existing Item”, find the “visa32.bas file” in the "include" folder under the NI-VISA installation path, and add it, as shown in the following figure.



3. **Source Code**

a) USBTMC Example

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

```
' results using the write and read functions.
' The general flow of the code is
' Open Resource Manager
' Open VISA Session to an Instrument
' Write the Identification Query Using viWrite
' Try to Read a Response With viRead
' Close the VISA Session

Const MAX_CNT = 200
Dim defaultRM AsLong
Dim instrsesn AsLong
Dim numInstrs AsLong
Dim findList AsLong
Dim retCount AsLong
Dim state 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.
state = viOpenDefaultRM(defaultRM)
If(state < VI_SUCCESS) Then
    resultTxt.Text = "Could not open a session to the VISA Resource Manager!"
    usbtmc_test = state
ExitFunction
EndIf

' Find all the USB TMC VISA resources in our system and store the
' number of resources in the system in numInstrs.
state = viFindRsrc(defaultRM, "USB?*INSTR", findList, numInstrs, instrResourceString)
If (state < VI_SUCCESS) Then
    resultTxt.Text = "An error occurred while finding resources."
    viClose(defaultRM)
    usbtmc_test = state
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
    state = viFindNext(findList, instrResourceString)
EndIf
    state = viOpen(defaultRM, instrResourceString, VI_NULL, VI_NULL, instrsesn)
If (state < 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.
state = viWrite(instrsesn, "*IDN?", 5, retCount)
If (state < VI_SUCCESS) Then
    resultTxt.Text = "Error writing to the device."
    state = 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.
state = viRead(instrsesn, Buffer, MAX_CNT, retCount)
If (state < 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
    state = viClose(instrsesn)
Next i

' Now we will close the session to the instrument using
' viClose. This operation frees all system resources.
state = 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
```

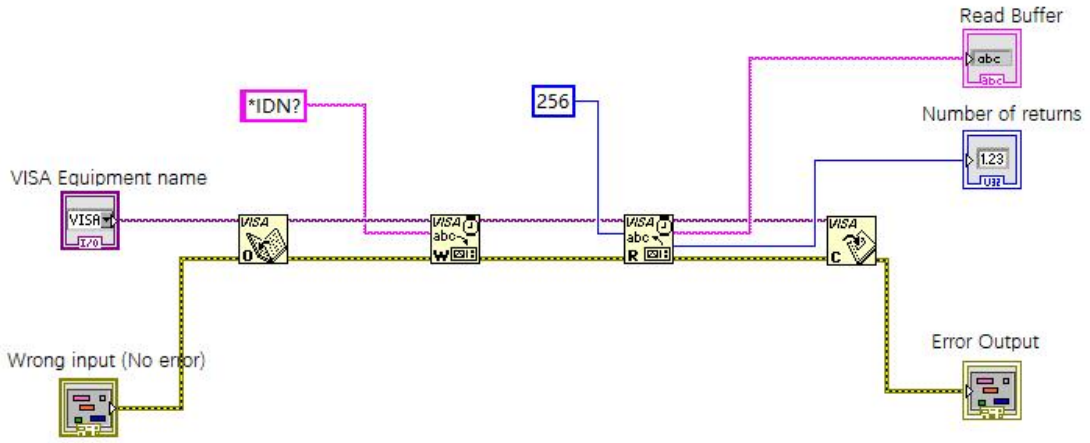
```
Dim state AsLong
Dim count AsLong

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

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

LabVIEW Example

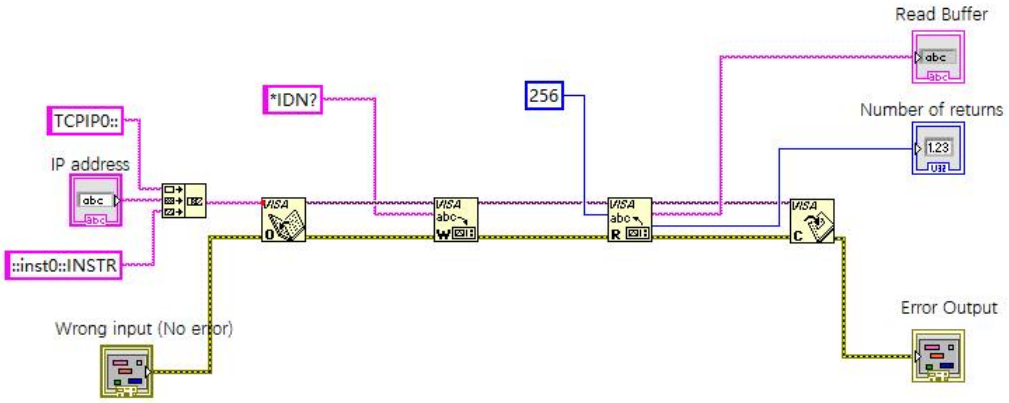
- **Environment:** Window System, LabVIEW.
- **Description:** Access the instrument via USBTMC and TCP/IP, and send "*IDN?" command on NI-VISA to query the device information.
- **Steps**

1. Open the LabVIEW software and create a VI file.
2. Add the control, click the front panel to select and add the VISA source name, error input, error output, and part of the indicator from the control column.
3. Open the diagram, click the VISA source name, and then select and add these functions VISA Write, VISA Read, VISA Open, and VISA Close on the VISA menu.

4. The VI opens a VISA session for a USBTMC device, writes the *IDN? command to the device, and reads back the response value. When all communication is complete, the VI closes the VISA session, as shown in the following figure.



5. Communication with the device via TCP/IP is similar to USBTMC. You need to set the VISA Write and Read functions to synchronous I/O, as LabVIEW uses asynchronous I/O by default. Right-click on the node and select “Synchronous I/O Mode >> Synchronous” from the shortcut menu to enable synchronous writing or reading of data, as shown in the following figure.



MATLAB Example

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

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

a) USBTMC Example

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

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

%Open the VISA object created
fopen(vu);

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

%Request the data

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

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

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, and send "*IDN?" command on NI-VISA to query the device information.
- **Steps**

1. Install python first, then open a Python script editor to create an empte test.py file.
2. Use the command “pip install PyVISA” to install PyVISA. If the installation fails, 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

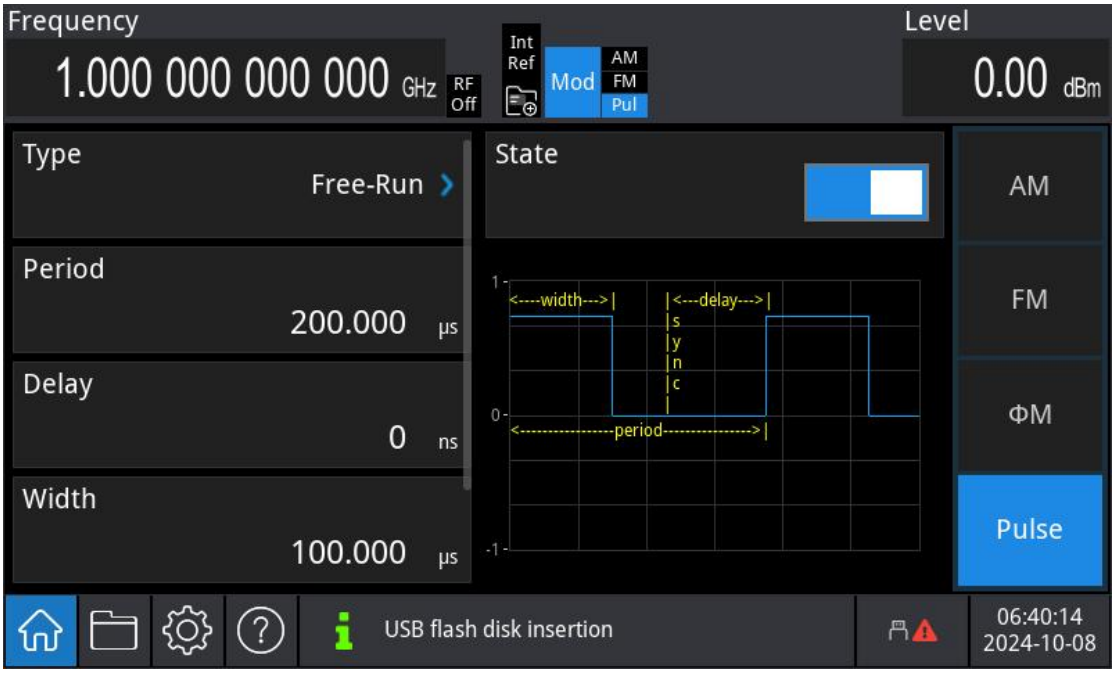
```
import pyvisa
rm = pyvisa.ResourceManager()
rm.list_resources()
my_instrument = rm.open_resource('TCPIP0::192.168.20.11::inst0::INSTR')
print(my_instrument.query('*IDN?'))
```

Programming Application Example

This section explains how to output the pulse modulation signal from the RF signal generator using SCPI commands.

Configure RF signal generator parameters:

1. Press the Default key to restore the parameters to their default settings.
2. Enable the radio-frequency output and configure the parameters: frequency and amplitude.
3. Enable the analog modulation, select the pulse modulation, and set the pulse type to FREErun.
4. Configure pulse parameters: period, delay, pulse width, sync pulse width.
5. Enable the pulse output.



The following commands can be used to perform the operations described above to obtain the output signals, as shown in the figure above.

```
:KEY:DEFault        // Restore to the default settings.
:OUTPut ON          // Enable the radio-frequency output.
:FREQuency 1GHz     // Set the output frequency to 1 GHz.
:POWer -10dBm       // Set the amplitude to -10 dBm.
:MOD:STATe ON       // Enable the analog modulation.
:MOD:PULSe:TYPE FREErun     // Set the pulse type to FREErun.
:MOD:PULSe:PERIod 200us     // Set the pulse period to 200 us.
:MOD:PULSe:DELAy 20ns       // Set the pulse delay to 20 ns.
:MOD:PULSe:WIDTh 100us      // Set the pulse width to 100 us.
:MOD:PULSe:SYNCwidth 1us    // Set the sync pulse width to 1 us.
:MOD:PULSe:EN ON            // Enable the pulse modulation.
```

Appendix 1: <key> Table

Key	Functional Description	LED
HOMe	Home page	
UTILity	System settings	
DEFAult	Restore to the default settings	
TLock	Lock screen	√
FREQ	Frequency	
AMPT	Amplitude	
SWEep	Sweep	
MODe	Mode	
AM	Amplitude modulation	
FM	Frequency modulation	
PULSe	Pulse modulation	
IQ	Vector source	
NUM7	Numeric key 7	
NUM4	Numeric key 4	
NUM1	Numeric key 1	
DOT	Decimal point key	
ESC	Exit	
NUM8	Numeric key 8	
NUM5	Numeric key 5	
NUM2	Numeric key 2	
NUM0	Numeric key 0	
BACKspace	Backspace	
NUM9	Numeric key 9	
NUM6	Numeric key 6	
NUM3	Numeric key 3	
SYMBOL	Numeric key	
ENTER	Carriage return	
GN	Unit	
MU	Unit	
KM	Unit	
DBM	Unit	
FKNob	OK	
UP	Move up	
LEFT	Move to the left	
TRIGger	Trigger	

LF	Low-frequency	√
MODS	Modulation	√
RF	Radio-frequency	√
RIGHT	Move to the right	
DOWN	Move down	
FKNLeft	Rotate to the left	
FKNRight	Rotate to the right	

Appendix 2: Key Lock State

No.	Key	State
0	HOMe	0 indicates that the key is unlocked. 1 indicates that the key is locked.
1	UTILity	0 indicates that the key is unlocked. 1 indicates that the key is locked.
2	DEFAult	0 indicates that the key is unlocked. 1 indicates that the key is locked.
3	TLock	0 indicates that the key is unlocked. 1 indicates that the key is locked.
4	FREQ	0 indicates that the key is unlocked. 1 indicates that the key is locked.
5	AMPT	0 indicates that the key is unlocked. 1 indicates that the key is locked.
6	SWEep	0 indicates that the key is unlocked. 1 indicates that the key is locked.
7	MODe	0 indicates that the key is unlocked. 1 indicates that the key is locked.
8	AM	0 indicates that the key is unlocked. 1 indicates that the key is locked.
9	FM	0 indicates that the key is unlocked. 1 indicates that the key is locked.
10	PULSe	0 indicates that the key is unlocked. 1 indicates that the key is locked.
11	IQ	0 indicates that the key is unlocked. 1 indicates that the key is locked.
12	NUM7	0 indicates that the key is unlocked. 1 indicates that the key is locked.
13	NUM4	0 indicates that the key is unlocked.

		1 indicates that the key is locked.
14	NUM1	0 indicates that the key is unlocked. 1 indicates that the key is locked.
15	DOT	0 indicates that the key is unlocked. 1 indicates that the key is locked.
16	ESC	0 indicates that the key is unlocked. 1 indicates that the key is locked.
17	NUM8	0 indicates that the key is unlocked. 1 indicates that the key is locked.
18	NUM5	0 indicates that the key is unlocked. 1 indicates that the key is locked.
19	NUM2	0 indicates that the key is unlocked. 1 indicates that the key is locked.
20	NUM0	0 indicates that the key is unlocked. 1 indicates that the key is locked.
21	BACKspace	0 indicates that the key is unlocked. 1 indicates that the key is locked.
22	NUM9	0 indicates that the key is unlocked. 1 indicates that the key is locked.
23	NUM6	0 indicates that the key is unlocked. 1 indicates that the key is locked.
24	NUM3	0 indicates that the key is unlocked. 1 indicates that the key is locked.
25	SYMBOL	0 indicates that the key is unlocked. 1 indicates that the key is locked.
26	ENTER	0 indicates that the key is unlocked. 1 indicates that the key is locked.
27	GN	0 indicates that the key is unlocked. 1 indicates that the key is locked.
28	MU	0 indicates that the key is unlocked. 1 indicates that the key is locked.
29	KM	0 indicates that the key is unlocked. 1 indicates that the key is locked.
30	DBM	0 indicates that the key is unlocked. 1 indicates that the key is locked.
31	FKNob	0 indicates that the key is unlocked. 1 indicates that the key is locked.
32	UP	0 indicates that the key is unlocked. 1 indicates that the key is locked.

33	LEFT	0 indicates that the key is unlocked. 1 indicates that the key is locked.
34	TRIGger	0 indicates that the key is unlocked. 1 indicates that the key is locked.
35	LF	0 indicates that the key is unlocked. 1 indicates that the key is locked.
36	MODS	0 indicates that the key is unlocked. 1 indicates that the key is locked.
37	RF	0 indicates that the key is unlocked. 1 indicates that the key is locked.
38	RIGHT	0 indicates that the key is unlocked. 1 indicates that the key is locked.
39	DOWN	0 indicates that the key is unlocked. 1 indicates that the key is locked.
40	FKNLeft	0 indicates that the key is unlocked. 1 indicates that the key is locked.
41	FKNRight	0 indicates that the key is unlocked. 1 indicates that the key is locked.