# SSG3000X Series Signal Generator

## Programming Guide

PG0803X-E01B

SIGLENT TECHNOLOGIES CO.,LTD

Contents

# 1. Programming Overview

**SSG3000X** support both USB and LAN interfaces. By using these interfaces, in combination with NI-VISA and programming languages, users can remotely control the signal generator. Through LAN interface, VXI-11, Sockets and Telnet protocols can be used to communicate with the signal generator. This chapter introduces how to build communication between the signal generator and the PC. It also introduces the remote control capabilities.

## 1.1Build Communication

### 1.1.1Build Communication Using VISA

#### 1、Install NI-VISA

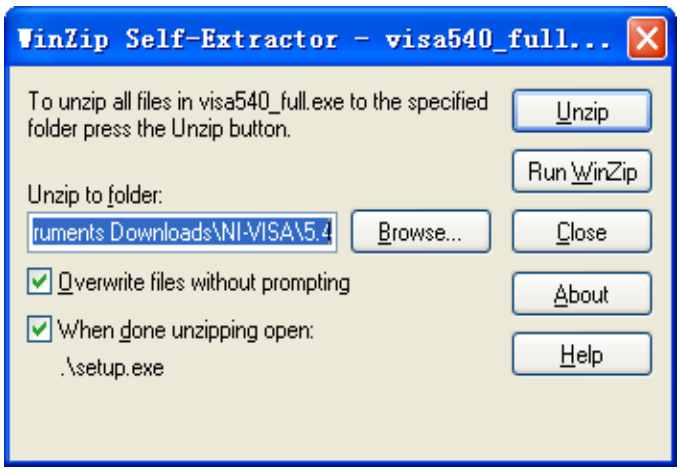
Before programming, you need to install NI-VISA, which you can download from the NI-VISA web site. About NI-VISA, there are full version and Run-Time Engine version. The full version includes the NI device driver and a tool named NI MAX that is a user interface to control the device. The Run-Time Engine version which is much smaller than the full version only include NI device driver.

For example, you can get NI-VISA 5.4 full version from: http://www.ni.com/download/ni-visa-5.4/4230/en/.

You can also download NI-VISA Run-Time Engine 5.4 to your PC and install it as default selection. Its installation process is similar with the full version.

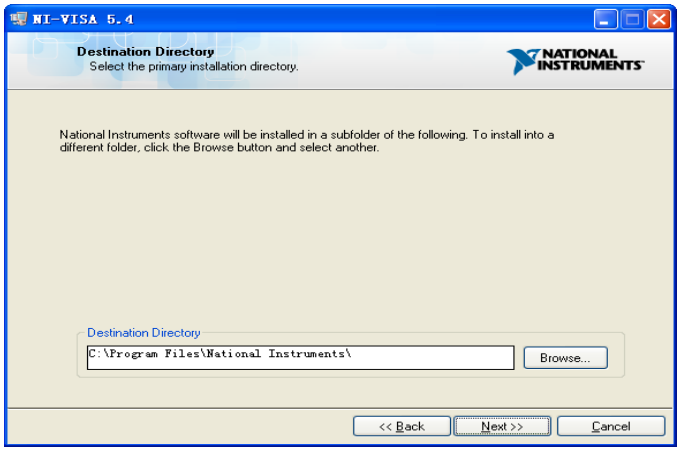
After you downloaded the file you can follow the steps below to install it:

a.Double click the visa540_full.exe, dialog shown as below:

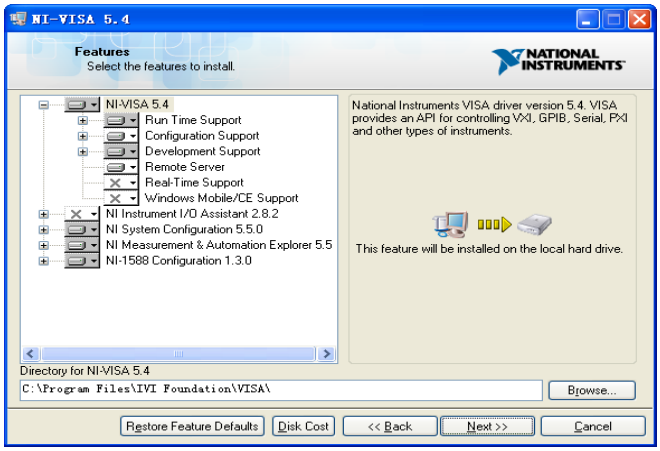


b.Click Unzip, the installation process will automatically launch after unzipping files. If your computer needs to install .NET Framework 4, its setup process will auto start.

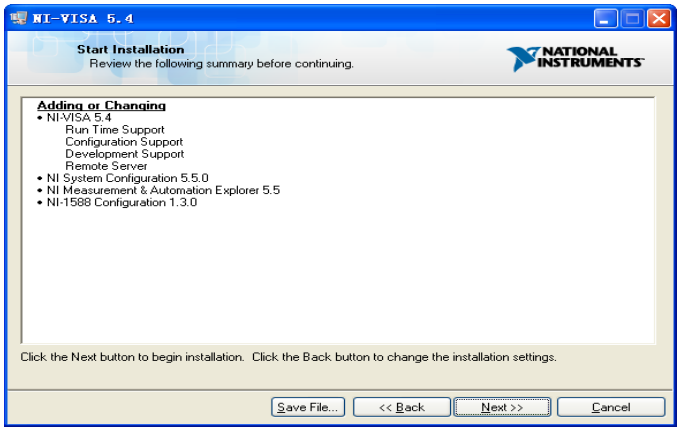

c.The NI-VISA installing dialog is shown above. Click Next to start the installation process.



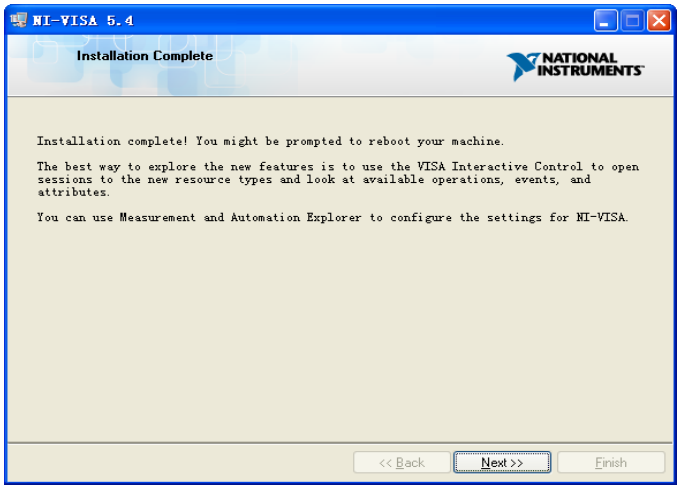

Set the install path, default path is “C:\Program Files\National Instruments\”, you can change it. Click Next, dialog shown as above.




d.Click Next twice, in the License Agreement dialog, select the “I accept the above 2 License Agreement(s).” ,and click Next, dialog shown as below:




e.Click Next to run installation.

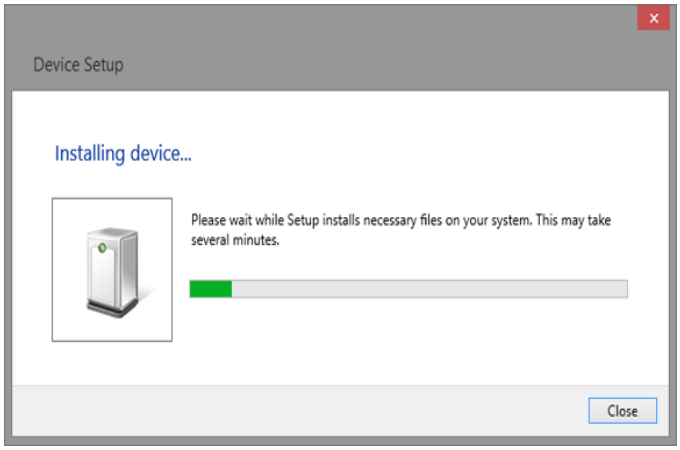



Now the installation is complete, reboot your PC.

## 2、Connect the Instrument

Depending on your specific model your signal generator may be able to communicate with a PC through the USB or LAN interface. This manual takes the USB as an example. (For instructions to communicate with a PC through the LAN interface see the User Manual.)

a.Connect the USB Device interface at the rear panel of the signal generator and the USB Host interface of the PC using a USB cable. Assuming your PC is already turned on, turn on your signal generator and your PC will display the "Device Setup" screen as it automatically installs the device driver as shown below.




b.Wait for the installation to complete and then proceed to the next step.

### 1.1.2Build Communication Using Sockets

Sockets LAN is a method used to communicate with the signal generator over the LAN interface using the Transmission Control Protocol/Internet Protocol (TCP/IP). A socket is a fundamental technology used for computer networking and allows applications to communicate using standard mechanisms built into network hardware and operating systems. The method accesses a port on the signal generator from which bidirectional communication with a network computer can be established.

Before you can use sockets LAN, you must select the signal generator's sockets port number to use:

- Standard mode. Available on port 5025. Use this port for simple programming.
- Telnet mode. The telnet SCPI service is available on port 5024.

### 1.1.3Connecting the signal generator via the USB Host port

Refer to the following steps to finish the connection via USB:

1. Install NI-VISA on your PC for GPIB driver.
2. Connect the signal generator USB Host port to a PC's GPIB card port, with SIGLENT USB-GPIB adaptor.

3. Switch on the signal generator
4. Press button on the front panel System→Interface→GPIB to enter the GPIB number.

The signal generator will be detected automatically as a new GPIB point.

# 1.2Remote Control Capabilities

## 1.2.1User-defined Programming

Users can use SCPI commands to program and control the signal generator. For details, refer to the introductions in "**Programming Examples**".

## 1.2.2Send SCPI Commands via NI-MAX

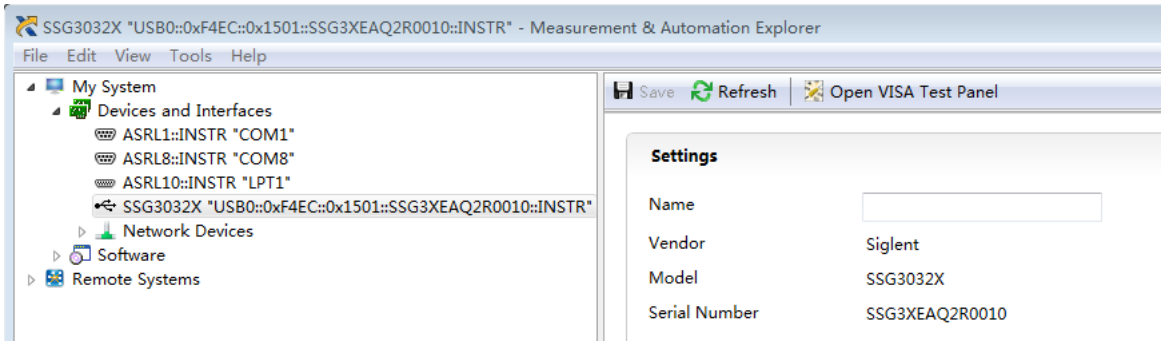
Users can control the signal generator remotely by sending SCPI commands via NI-MAX software.

### 1.2.2.1 Using USB

Run NI MAX software.

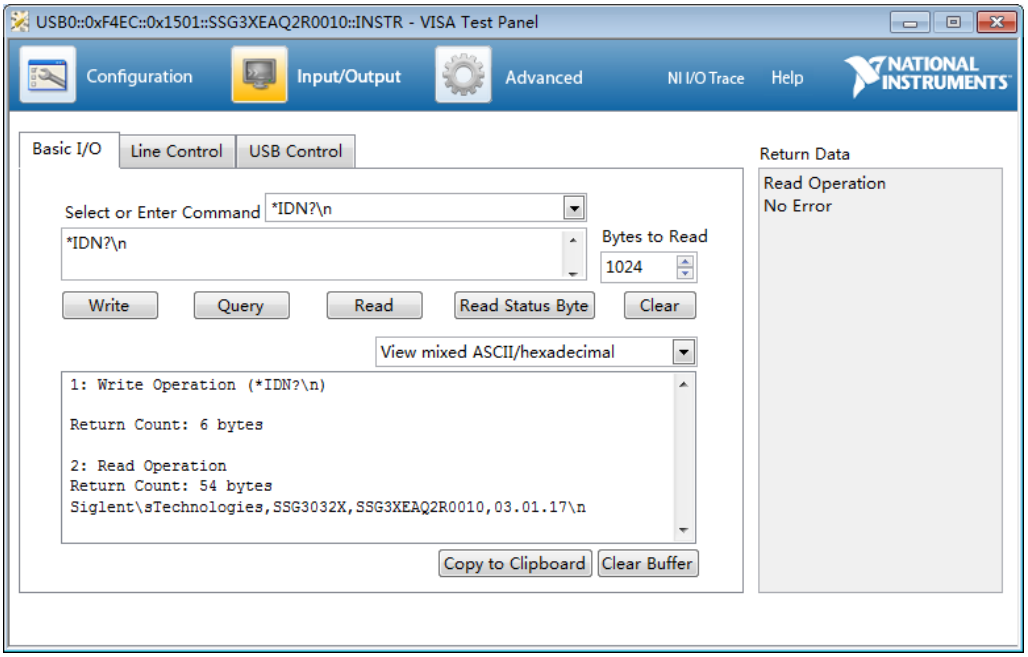
1, Click "Device and interface" at the upper left corner of the software;

2, Find the "USBTMC" device symbol;



3, Click "Open VISA Test Panel" option button, then the following interface will appear;

4, Click the "Input/Output" option button and click the "Query" option button in order to view the operation information.




**NOTE:** The *IDN? command (known as the Identification Query) returns the instrument manufacturer, instrument model, serial number, and other identification information.

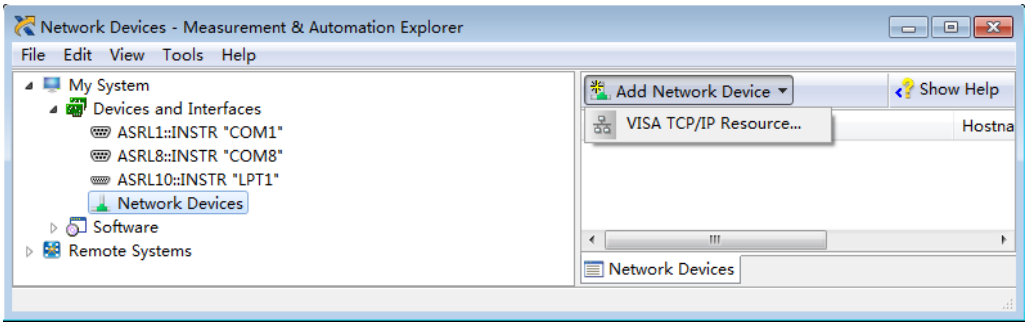
## 1.2.2.2 Using LAN

Add Network Device, and select VISA TCP/IP Resource as shown:.

Run NI MAX software.

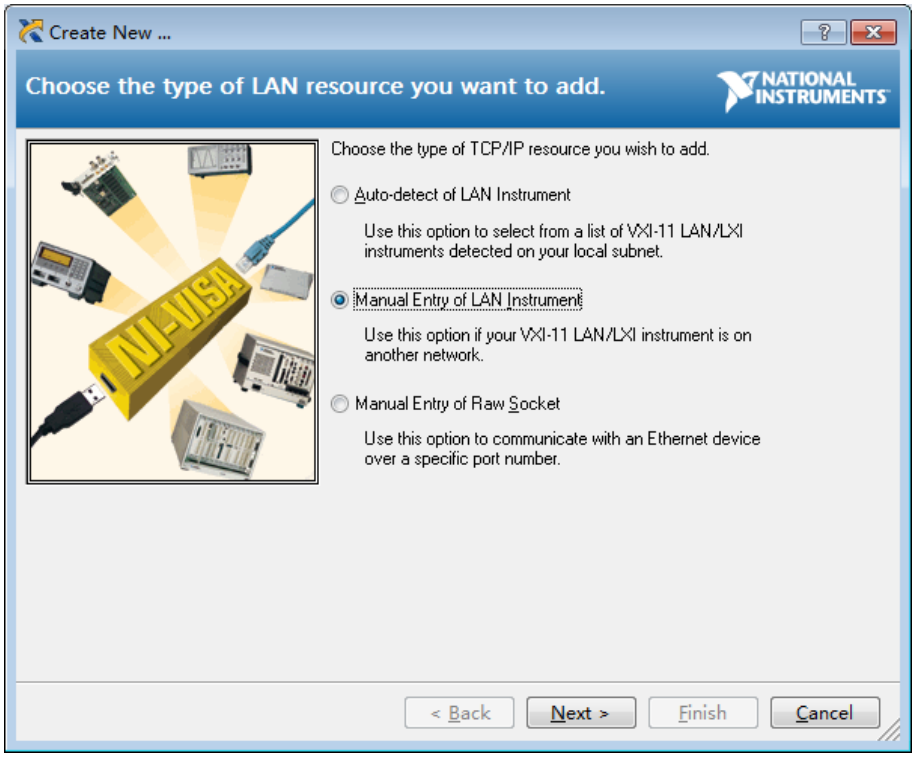
1, Click "Device and interface" at the upper left corner of the software;

2, Find the "Network Devices" symbol, click "Add Network Devices";

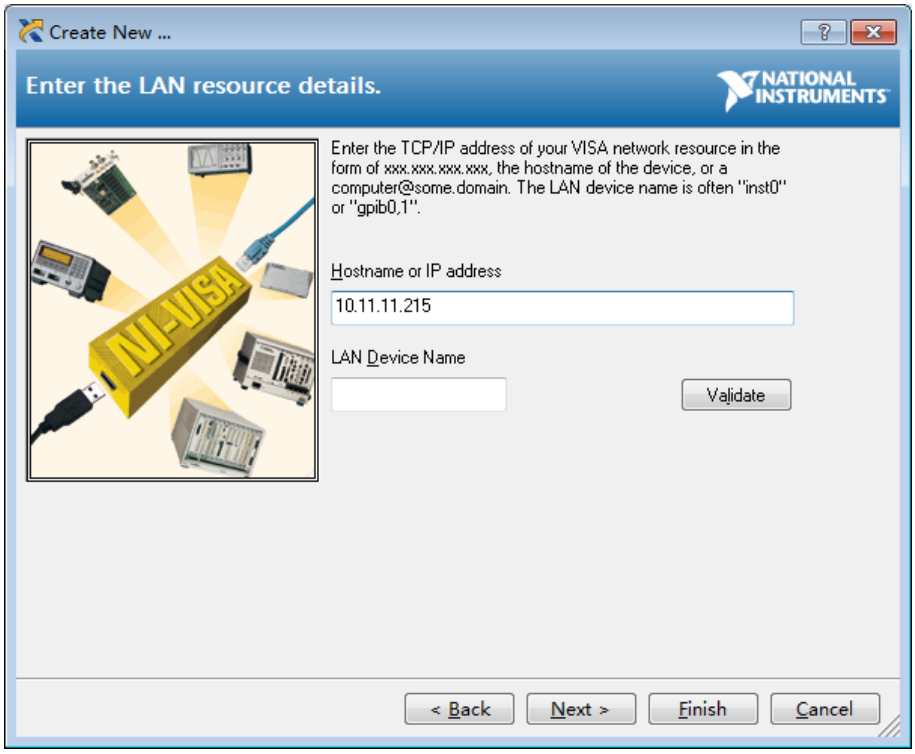



3. Select Manual Entry of LAN instrument, select Next, and enter the IP address as

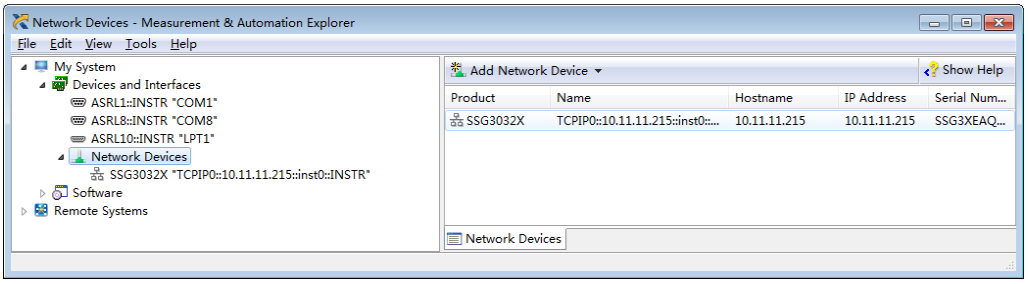
shown. Click Finish to establish the connection:



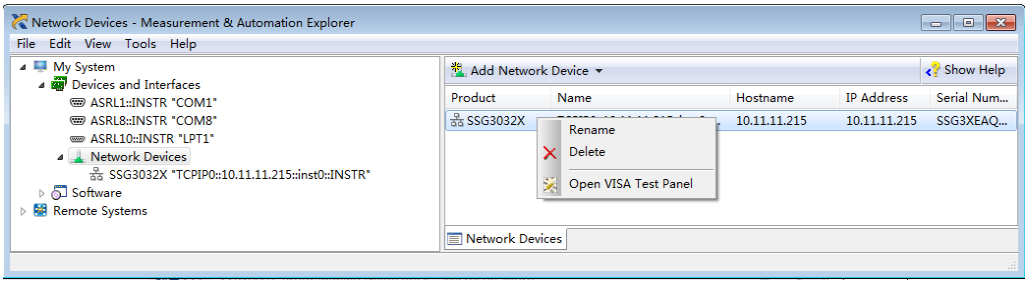

**NOTE:** Leave the LAN Device Name BLANK or the connection will fail.




4. After a brief scan, the connection should be shown under Network Devices:




5. Right-click on the product and select Open NI-VISA Test Panel:




6. Click "Input/Output" option button and click "Query" option button. If everything is OK, you will see the Read operation information returned as shown below.

# 2.SCPI Overview

## 2.1Command Format

SCPI commands present a hierarchical tree structure containing multiple subsystems, each of the subsystems is made up of a root keyword and several sub keywords. The command string usually starts with ":", the keywords are separated by ":" and the followed parameter settings are separated by space. Query commands add "?" at the end of the string.

For example:

:SOURce:FREQuency <freq>

:SOURce:FREQuency?

SOURce is the root key of the command, FREQuency is second. The command begins with ":", and separates the keywords at the same time, <freq> separated by space and represents the parameter available for setting; "?" represents a query.

## 2.2Symbol Instruction

The following four symbols are not the content of SCPI commands and can not be sent with the commands, but are usually used in the commands.

### 1, Triangle Brackets < >

The parameter in the triangle brackets must be replaced by an effective value. For example:

Send the "CALibration:SPC:TARGet <power>" command in "CALibration:SPC:TARGet 0".

### 2, Square Brackets [ ]

The content in the square brackets can be ignored. When the parameter is ignored, the instrument will set the parameter to its default. For example,

In the "[:SOURce]:POWer?" command, sending any of the four commands below can generate the same effect:

:SOURce:POWer?

:POWer?

### 3, Vertical Bar |

The vertical bar is used to separate multiple parameters and when sending the command, you can choose one of the parameters. For example,

In the "[:SOURce]:AM:STATe OFF|ON|0|1" command, the parameters available are "OFF", "ON", "0" or "1".

### 4, Braces { }

The parameters in the braces are optional which can be ignored or set for one or more times.

# 2.3Parameter Type

The parameters in the commands introduced in this manual include 6 types: boolean, enumeration, integer, float and string.

### 1, Boolean

The parameters in the commands could be "OFF", "ON", "0" or "1". For example:

[:SOURce]:FM:STATe OFF|ON|0|1

### 2, Enumeration

The parameter could be any of the values listed. For example:

[:SOURce]:SWEep:STATe OFF|FREQuency|LEVel|LEV_FREQ

The parameter is "OFF", "FREQuency", "LEVel" or LEV_FREQ.

### 3, Integer

Except other notes, the parameter can be any integer within the effective value range. For example:

[:SOURce]:SWEep:STEP:POINts <value>

The parameter <value> can be set to any integer between 2 and 65535.

### 4, Float

The parameter could be any value within the effective value range according to the accuracy requirement (the default accuracy contains up to 9 digits after the decimal

points). For example:

[:SOURce]:POWer:OFFSet <value>

The parameter <value> can be set to any real number between -100 and 100.

### 5, String

The parameter should be the combinations of ASCII characters. For example:

:SYSTem:COMMunicate:LAN:IPADdress < “xxx.xxx.xxx.xxx” >

The parameter can be set as “192.168.1.12” string.

# 2.4Command Abbreviation

All of the commands are not case sensitive, so you can use any of them. But if abbreviation is used, all the capital letters in the command must be written completely. For example:

:CORRection:FLATness:COUNt?

Can be abbreviated to:

:CORR:FLAT:COUN?

# 3.System Commands

This chapter introduces the Siglent Technologies SSG3000X SCPI commands, include:



# 3.1IEEE Common Commands

## 3.1.1Identification Query (*IDN)

| | |
|---|---|
| Command Format | *IDN? |
| Instruction | Returns an instrument identification information string. The string will contain the manufacturer, model number, serial number, software number, FPGA number and CPLD number. |
| Menu | None |
| Example | *IDN?<br>Return: Siglent Technologies,SSG3032X,1234567890, 03.01.16r2 |

## 3.1.2Reset (*RST)

| | |
|---|---|
| Command Format | *RST |
| Instruction | This command presets the instrument to a factory defined condition that is appropriate for remote programming operation. *RST is equivalent to performing the two commands :SOURce:PRESet and *CLS. This command always performs a factory preset. |
| Menu | None |

| | |
|---|---|
| Example | *RST |

### 3.1.3Clear Status (*CLS)

| | |
|---|---|
| Command Format | *CLS |
| Instruction | Clears the status byte register. It does this by emptying the error queue and clearing all bits in all of the event registers. The status byte register summarizes the states of the other registers. It is also responsible for generating service requests. |
| Menu | None |
| Example | *CLS |

### 3.1.4Standard Event Status Enable (*ESE)

| | |
|---|---|
| Command Format | *ESE <number><br>*ESE? |
| Instruction | Set the bits in the standard event status enable register. This register monitors I/O errors and synchronization conditions such as operation complete, request control, query error, device dependent error, execution error, command error and power on. A summary bit is generated on execution of the command.<br>The query returns the state of the standard event status enable register. |
| Menu | None |
| Example | *ESE 16 |

### 3.1.5Standard Event Status Register Query (*ESR)

| | |
|---|---|
| Command Format | *ESR? |
| Instruction | Queries and clears the standard event status event register. (This is a destructive read.) The value returned reflects the current state (0/1) of all the bits in the register. |
| Menu | None |
| Example | *ESR? |

### 3.1.6 Operation Complete Query (*OPC)

| | |
|---|---|
| Command Format | *OPC<br>*OPC? |
| Instruction | Set bit 0 in the standard event status register to "1" when all pending operations have finished.<br>The query stops any new commands from being processed until the current processing is complete. Then it returns a "1", and the program continues. This query can be used to synchronize events of other instruments on the external bus.<br>Returns a "1" if the last processing is complete. Use this query when there's a need to monitor the command execution status, such as a sweep execution. |
| Menu | None |
| Example | *OPC? |

### 3.1.7 Service Request Enable (*SRE)

| | |
|---|---|
| Command Format | *SRE <integer><br>*SRE? |
| Instruction | This command enables the desired bits of the service request enable register.<br>The query returns the value of the register, indicating which bits are currently enabled. |
| Menu | None |
| Example | *SRE 1 |

### 3.1.8 Status Byte Query (*STB)

| | |
|---|---|
| Command Format | *STB |
| Instruction | This query is used by some instruments for a self test. |
| Menu | None |
| Example | *STB |

### 3.1.9Wait-to-Continue (*WAI)

| | |
|---|---|
| Command Format | *WAI |
| Instruction | This command causes the instrument to wait until all pending commands are completed before executing any additional commands.<br>There is no query form to the command. |
| Menu | None |
| Example | *WAI |

### 3.1.10Self Test Query (*TST)

| | |
|---|---|
| Command Format | *TST? |
| Instruction | This query is used by some instruments for a self test. |
| Menu | None |
| Example | *TST? |

## 3.2System Subsystem

### 3.2.1System Time (:SYSTem:TIME)

| | |
|---|---|
| Command Format | :SYSTem:TIME <hhmmss><br>:SYSTem:TIME? |
| Instruction | Set System time.<br>Get System time. |
| Parameter Type | String |
| Parameter Range | hour(0 ~ 23), minute(0 ~ 59), second(0 ~ 59) |
| Return | String |
| Default | None |
| Menu | Utility > Setting > Time Setting |
| Example | Set System time:<br>:SYSTem:TIME 182559 |

| | |
|---|---|
| | Get System time:<br>:SYSTem:TIME? |

### 3.2.2System Date (:SYSTem:DATE)

| | |
|---|---|
| Command Format | :SYSTem:DATE <yyyymmdd><br>:SYSTem:DATE? |
| Instruction | Set system date.<br>Get system date. |
| Parameter Type | String |
| Parameter Range | year(four digits), month(1 ~ 12), date(1 ~ 31) |
| Return | String |
| Default | None |
| Menu | Utility > Setting > Time Setting |
| Example | Set System date:<br>:SYSTem:DATE 20050101<br>Get System date:<br>:SYSTem:DATE? |

### 3.2.3IP Address (:SYSTem:COMMunicate:LAN:IPADdress)

| | |
|---|---|
| Command Format | :SYSTem:COMMunicate:LAN:IPADdress < “xxx.xxx.xxx.xxx” ><br>:SYSTem:COMMunicate:LAN:IPADdress? |
| Instruction | Set a host name for the signal generator in network.<br>Get IP address. |
| Parameter Type | String |
| Parameter Range | Conform to the IP Sets standard(0-255:0-255:0-255:0-255) |
| Return | IP address string |
| Default | None |
| Menu | Utility > Interface > LAN Setting > IP Address |
| Example | :SYSTem:COMMunicate:LAN:IPADdress “192.168.1.12” |

| | |
|---|---|
| | :SYSTem:COMMunicate:LAN:IPADdress? |

## 3.2.4Gateway (:SYSTem:COMMunicate:LAN:GATeway)

| | |
|---|---|
| Command Format | :SYSTem:COMMunicate:LAN:GATeway < “xxx.xxx.xxx.xxx” ><br>:SYSTem:COMMunicate:LAN:GATeway? |
| Instruction | Set the gateway for the signal generator in the network. The gateway will be fetched automatically if the IP assignment is set to DHCP.<br>Get gateway. |
| Parameter Type | String |
| Parameter Range | Conform to the IP standard (0~255.0~255.0~255.0~255) |
| Return | Gateway string. |
| Default | None |
| Menu | Utility > Interface > LAN Setting > Gateway |
| Example | :SYSTem:COMMunicate:LAN:GATeway “192.168.1.1”<br>:SYSTem:COMMunicate:LAN:GATeway? |

## 3.2.5Subnet Mask (:SYSTem:COMMunicate:LAN:SMASk)

| | |
|---|---|
| Command Format | :SYSTem:COMMunicate:LAN:SMASk < “xxx.xxx.xxx.xxx” ><br>:SYSTem:COMMunicate:LAN:SMASk? |
| Instruction | Set the subnet mask according to the PC network Settings. The subnet mask will be set automatically if the IP assignment is set to DHCP. |
| Parameter Type | String |
| Parameter Range | Conform to the IP standard (0-255:0-255:0-255:0-255) |
| Return | Subnet mask string |
| Default | None |
| Menu | Utility > Interface > LAN Setting > Subnet Mask |
| Example | :SYSTem:COMMunicate:LAN:SMASk? |

## 3.2.6IP Config (:SYSTem:COMMunicate:LAN:TYPE)

| | |
|---|---|
| Command Format | :SYSTem:COMMunicate:LAN:TYPE STATIC\|DHCP<br>:SYSTem:COMMunicate:LAN:TYPE? |
| Instruction | Toggles the IP assignment Setting between static (manual) and DHCP (dynamic assignment) mode.<br>Get IP config. |
| Parameter Type | Enumeration |
| Parameter Range | STATIC\|DHCP |
| Return | Enumeration |
| Default | None |
| Menu | Utility > Interface > LAN Setting > DHCP State |
| Example | :SYSTem:COMMunicate:LAN:TYPE DHCP<br>:SYSTem:COMMunicate:LAN:TYPE? |

## 3.2.7Power On Type (:SYSTem:PON:TYPE)

| | |
|---|---|
| Command Format | :SYSTem:PON:TYPE DFT\|LAST<br>:SYSTem:PON:TYPE? |
| Instruction | Uses command to set signal generator to power on in default, last.<br>Get power on type. |
| Parameter Type | Enumeration |
| Parameter Range | DFT\|LAST<br>DFT: Default<br>LAST: Last |
| Return | Enumeration |
| Default | DFT |
| Menu | Utility > Setting > Power On |
| Example | SYSTem:PON:TYPE DFT |

## 3.2.810M Adjustment State (:SYSTem:REF:DAC:STAT)

| | |
|---|---|
| Command Format | :SYSTem:REF:DAC:STAT ON\|OFF\|1\|0<br>:SYSTem:REF:DAC:STAT? |

| | |
|---|---|
| Instruction | Set 10M Adjustment State.<br>Get 10M Adjustment State. |
| Parameter Type | Boolean |
| Parameter Range | ON\|OFF\|1\|0 |
| Return | Boolean |
| Default | 0 |
| Menu | Utility > Setting > 10M Adjustment |
| Example | :SYSTem:REF:DAC:STAT ON |

### 3.2.9 Ref Osc Code (:SYSTem:REF:DAC)

| | |
|---|---|
| Command Format | :SYSTem:REF:DAC <value><br>:SYSTem:REF:DAC? |
| Instruction | Set ref osc code.<br>Get ref osc code. |
| Parameter Type | Int |
| Parameter Range | 0 ~ 65535 |
| Return | Int |
| Default | 42885 |
| Menu | Utility > Setting > 10M Adjustment |
| Example | :SYSTem:REF:DAC 43000<br>:SYSTem:REF:DAC? |

### 3.2.10 Ref Osc Code Store (:SYSTem:REF:DAC:SAVE)

| | |
|---|---|
| Command Format | :SYSTem:REF:DAC:SAVE <file_name> |
| Instruction | Save the ref osc code in file. |
| Parameter Type | String |
| Parameter Range | None |

| | |
|---|---|
| Return | None |
| Default | None |
| Menu | Utility > Setting > 10M Adjustment |
| Example | :SYSTem:REF:DAC:SAVE test.dac |

### 3.2.11 Ref Osc Code Load (:SYSTem:REF:DAC:LOAD)

| | |
|---|---|
| Command Format | :SYSTem:REF:DAC:LOAD <file_name> |
| Instruction | Load existing ref osc code files. |
| Parameter Type | String |
| Parameter Range | None |
| Return | None |
| Default | None |
| Menu | Utility > Setting > 10M Adjustment > Recall Ref Osc Setting |
| Example | :SYSTem:REF:DAC:LOAD test.dac |

### 3.2.12 Reset Ref Osc Code to Default (:SYSTem:REF:DAC:DEFault)

| | |
|---|---|
| Command Format | :SYSTem:REF:DAC:DEFault |
| Instruction | Reset ref osc code to default value. |
| Parameter Type | None |
| Parameter Range | None |
| Return | None |
| Default | None |
| Menu | Utility > Setting > 10M Adjustment > Reset to Default |
| Example | :SYSTem:REF:DAC:DEFault |

# 3.3Preset Subsystem

## 3.3.1Preset (:SOURce:PRESet)

| | |
|---|---|
| Command Format | :SOURce:PRESet |
| Instruction | Presets all parameters which are related to the selected signal path. |
| Parameter Type | None |
| Return | None |
| Default | None |
| Menu | None |
| Example | SOUR:PRES |

## 3.3.1System Preset (:SYSTem:PRESet)

| | |
|---|---|
| Command Format | :SYSTem:PRESet |
| Instruction | Presets all parameters. |
| Parameter Type | None |
| Return | None |
| Default | None |
| Menu | Utility > Preset |
| Example | SYSTem:PRES |

## 3.3.2Preset Save (:SYSTem:PRESet:SAVE)

| | |
|---|---|
| Command Format | :SYSTem:PRESet:SAVE |
| Instruction | Save status for preset when preset type is user. |
| Parameter Type | None |
| Return | None |
| Default | None |

| | |
|---|---|
| Menu | Utility > Preset |
| Example | :SYSTem:PRESet:SAVE |

### 3.3.3Preset Path (:SYSTem:PRESet:PATH)

| | |
|---|---|
| Command Format | :SYSTem:PRESet:PATH <path> |
| Instruction | Set preset file when preset type is user. |
| Parameter Type | String |
| Return | None |
| Default | None |
| Menu | Utility > Preset |
| Example | :SYSTem:PRESet:PATH test.xml |

### 3.3.4Preset Type (:SYSTem:PRESet:TYPE)

| | |
|---|---|
| Command Format | :SYSTem:PRESet:TYPE DFT\|USER<br>:SYSTem:PRESet:TYPE? |
| Instruction | Uses this command to preset the signal generator to default, user.<br>Get preset type. |
| Parameter Type | Enumeration |
| Parameter Range | DFT: Default<br>USER: Custom Configuration |
| Return | Enumeration |
| Default | DFT |
| Menu | Utility > Setting > Preset Type |
| Example | :SYSTem:PRESet:TYPE DFT |

### 3.3.5Factory Reset (:SYSTem:FDEFault)

| | |
|---|---|
| Command Format | :SYSTem:FDEFault |
| Instruction | Set both the measure and setting parameters to factory preset parameters. |
| Parameter Type | None |

| | |
|---|---|
| Parameter Range | None |
| Return | None |
| Default | None |
| Menu | None |
| Example | :SYSTem:FDEFault |

# 3.4Output Subsystem

## 3.4.1RF Output (:OUTPut[:STATe])

| | |
|---|---|
| Command Format | :OUTPut[:STATe] ON\|OFF\|1\|0<br>:OUTPut[:STATe]? |
| Instruction | Activate/Deactivate the RF output.<br>Get the RF state. |
| Parameter Type | Boolean |
| Parameter Range | ON\|OFF\|1\|0 |
| Return | Boolean |
| Default | 0 |
| Menu | RF |
| Example | :OUTPut ON |

# 3.5Source Subsystem

## 3.5.1[:SOURce]:FREQuency Subsystem

### 3.5.1.1 Frequency Display ([:SOURce]:FREQuency:DISPlay)

| | |
|---|---|
| Command Format | [:SOURce]:FREQuency:DISPlay <freq><br>[:SOURce]:FREQuency:DISPlay? |
| Instruction | Set the frequency display on parameter bar. |

| | |
|---|---|
| | Get the frequency display on parameter bar. |
| Parameter Type | Float, unit: Hz, kHz, MHz, GHz, Default “Hz” |
| Parameter Range | Frequency offset + Full frequency range |
| Return | Float, unit: Hz |
| Default | Maximum frequency |
| Menu | Freq |
| Example | FREQuency:DISPlay 2 MHz |

### 3.5.1.2 Frequency ([:SOURce]:FREQuency[:FIX])

| | |
|---|---|
| Command Format | [:SOURce]:FREQuency[:FIX] <freq><br>[:SOURce]:FREQuency[:FIX]? |
| Instruction | Set the frequency of the RF output signal.<br>Get the frequency of the RF output signal. |
| Parameter Type | Float, unit: Hz, kHz, MHz, GHz, Default “Hz” |
| Parameter Range | Full frequency range |
| Return | Float, unit: Hz |
| Default | Maximum frequency |
| Menu | FREQ > Frequency |
| Example | FREQuency 2 MHz |

### 3.5.1.3 Frequency Offset ([:SOURce]:FREQuency:OFFSet)

| | |
|---|---|
| Command Format | [:SOURce]:FREQuency:OFFSet <freq><br>[:SOURce]:FREQuency:OFFSet? |
| Instruction | Set the frequency offset of a downstream instrument.<br>Get the frequency offset of a downstream instrument. |
| Parameter Type | Float, unit: Hz, kHz, MHz, GHz, Default “Hz” |
| Parameter Range | -200 GHz ~ 200 GHz |
| Return | Float, unit: Hz |

| | |
|---|---|
| Default | 0 Hz |
| Menu | FREQ > Freq Offset |
| Example | FREQuency:OFFSet 2 MHz |

### 3.5.1.4 Phase Offset ([:SOURce]:PHASe)

| | |
|---|---|
| Command Format | [:SOURce]:PHASe <phase><br>[:SOURce]:PHASe? |
| Instruction | Set the phase of the RF output signal.<br>Get the phase of the RF output signal. |
| Parameter Type | Float, unit: deg |
| Parameter Range | -360 deg ~ 360 deg |
| Return | Float, unit: deg |
| Default | 0 deg |
| Menu | FREQ > Phase Offset |
| Example | PHASe 20 |

### 3.5.1.5 Phase Reset ([:SOURce]:PHASe:RESet [:SOURce]:PHASe:REF)

| | |
|---|---|
| Command Format | [:SOURce]:PHASe:RESet<br>[:SOURce]:PHASe:REF |
| Instruction | Set the current phase to relatively zero. |
| Parameter Type | None |
| Parameter Range | None |
| Return | None |
| Default | None |
| Menu | FREQ > Reset phase delta display |
| Example | :PHASe:RESet |

# 3.5.2[:SOURce]:POWer Subsystem

## 3.5.2.1 Level Display ([:SOURce]:POWer:POWer)

| | |
|---|---|
| Command Format | [:SOURce]:POWer:POWer <power><br>[:SOURce]:POWer:POWer? |
| Instruction | Set the RF level display on parameter bar.<br>Get the RF level display on parameter bar. |
| Parameter Type | Float, unit: dBm, dBmV, dBuV, V, W, Default: dBm |
| Parameter Range | Level Offset + Full power range |
| Return | Float, unit: dBm |
| Default | -110 dBm |
| Menu | Level |
| Example | POWer:POWer 2 |

## 3.5.2.2 Level ([:SOURce]:POWer)

| | |
|---|---|
| Command Format | [:SOURce]:POWer <power><br>[:SOURce]:POWer? |
| Instruction | Set the RF output level.<br>Get the RF output level. |
| Parameter Type | Float, unit: dBm, dBmV, dBuV, V, W |
| Parameter Range | When IQ switched on: -100 dBm ~ 10 dBm<br>Frequency between 9 kHz ~ 100 kHz: -110 dBm ~ 9 dBm<br>Frequency between 100 kHz ~ 1 MHz: -110 dBm ~ 15 dBm<br>Frequency above 1 MHz: -110 dBm ~ 20 dBm |
| Return | Float, unit: dBm |
| Default | -110 dBm |
| Menu | LEVEL > Level |
| Example | POWer 2 |

### 3.5.2.3 Level Offset ([:SOURce]:POWer:OFFSet)

| | |
|---|---|
| Command Format | [:SOURce]:POWer:OFFSet <power><br>[:SOURce]:POWer:OFFSet? |
| Instruction | Set the RF offset level of the RF output connector.<br>Get the RF offset level of the RF output connector. |
| Parameter Type | Float |
| Parameter Range | -100 dB ~ 100 dB |
| Return | Float, unit: dB |
| Default | 0 dB |
| Menu | LEVEL > Level Offset |
| Example | POWer:OFFSet 2 |

### 3.5.2.4 ALC State ([:SOURce]:POWer:ALC)

| | |
|---|---|
| Command Format | [:SOURce]:POWer:ALC ON\|OFF\|AUTO<br>[:SOURce]:POWer:ALC? |
| Instruction | Activate/deactivate automatic level control.<br>Query ALC state. |
| Parameter Type | Enumeration |
| Parameter Range | ON\|OFF\|AUTO<br>ON<br>Internal level control is permanently activated.<br>OFF<br>Internal level control is deactivated; Sample & Hold mode is activated.<br>AUTO<br>Internal level control is activated/deactivated automatically depending on the operating state. |
| Return | Enumeration |
| Default | AUTO |
| Menu | LEVEL > ALC State |
| Example | POWer:ALC ON |

### 3.5.2.5 Flatness List State ([:SOURce]:CORRection[:FLATness])

| | |
|---|---|
| Command Format | [:SOURce]:CORRection[:FLATness] ON\|OFF\|1\|0<br>[:SOURce]:CORRection[:FLATness]? |
| Instruction | Activate/deactivate flatness. |
| Parameter Type | Boolean |
| Parameter Range | ON\|OFF\|1\|0 |
| Return | Boolean |
| Default | 0 |
| Menu | LEVEL > Flatness |
| Example | CORRection:FLATness ON |

### 3.5.2.6 Flatness List Add Row ([:SOURce]:CORRection:FLATness:PAIR)

| | |
|---|---|
| Command Format | [:SOURce]:CORRection:FLATness:PAIR <freq>,<power> |
| Instruction | Insert a new row. |
| Parameter Type | Float, Float |
| Parameter Range | Freq: Full freq range<br>Power: Full power range |
| Return | None |
| Default | None |
| Menu | LEVEL > Flatness > [+] |
| Example | CORRection:FLATness:PAIR 1 MHz,1 |

### 3.5.2.7 Flatness List Delete Row ([:SOURce]:CORRection:FLATness:DELete)

| | |
|---|---|
| Command Format | [:SOURce]:CORRection:FLATness:DELete <row> |
| Instruction | Delete the selected row. |

| | |
|---|---|
| Parameter Type | Integer |
| Parameter Range | Less than the total count of the flatness. |
| Return | None |
| Default | None |
| Menu | LEVEL > Flatness > [-] |
| Example | CORRection:FLATness:DELete 0 |

### 3.5.2.8 Flatness List Count ([:SOURce]:CORRection:FLATness:COUNt?)

| | |
|---|---|
| Command Format | [:SOURce]:CORRection:FLATness:COUNt? |
| Instruction | Indicates the total count of the flatness. |
| Parameter Type | None |
| Parameter Range | None |
| Return | Integer |
| Default | 0 |
| Menu | LEVEL > Flatness |
| Example | CORRection:FLATness:COUNt? |

### 3.5.2.9 Flatness List Store ([:SOURce]:CORRection:STORe)

| | |
|---|---|
| Command Format | [:SOURce]:CORRection:STORe <file_name> |
| Instruction | Save the correction data in the list. |
| Parameter Type | String |
| Parameter Range | None |
| Return | None |
| Default | None |

| | |
|---|---|
| Menu | LEVEL > Flatness > Store |
| Example | :CORRection:STORe test.uflt |

### 3.5.2.10 Flatness List Load ([:SOURce]:CORRection:LOAD)

| | |
|---|---|
| Command Format | [:SOURce]:CORRection:LOAD <file_name> |
| Instruction | Load existing flatness correction files. |
| Parameter Type | String |
| Parameter Range | None |
| Return | None |
| Default | None |
| Menu | LEVEL > Flatness > Load |
| Example | :CORRection:LOAD test.uflt |

### 3.5.2.11 Flatness List Clear ([:SOURce]:CORRection:FLATness:PRESet)

| | |
|---|---|
| Command Format | [:SOURce]:CORRection:FLATness:PRESet |
| Instruction | Clear flatness correction list. |
| Parameter Type | None |
| Parameter Range | None |
| Default | None |
| Menu | LEVEL > Flatness > Clear |
| Example | :CORRection:FLATness:PRESet |

### 3.5.2.12 Flatness List Fill Type ([:SOURce]:CORRection:FLATness:FILL:TYPE)

| | |
|---|---|
| Command | [:SOURce]:CORRection:FLATness:FILL:TYPE FLATness\|MANUal\|SWEEPlist |

| | |
|---|---|
| Format | [:SOURce]:CORRection:FLATness:FILL:TYPE? |
| Instruction | Set the Fill Type to generate flatness list.<br>Get the Fill Type to generate flatness list. |
| Parameter Type | Enumeration |
| Parameter Range | FLATness\|MANUal\|SWEEPlist |
| Return | Enumeration |
| Default | FLATness |
| Menu | LEVEL > Flatness > Set > Fill Type |
| Example | :CORRection:FLATness:FILL:TYPE FLATness |

### 3.5.2.13 Flatness List Start Freq ([:SOURce]:CORRection:FLATness:STARtfreq)

| | |
|---|---|
| Command Format | [:SOURce]:CORRection:FLATness:STARtfreq <freq><br>[:SOURce]:CORRection:FLATness:STARtfreq? |
| Instruction | Set the start frequency when you want to fill the flatness list with the sensor and filling type is “Manual Step”.<br>Get the start frequency when you want to fill the flatness list with the sensor and filling type is “Manual Step”. |
| Parameter Type | Float, unit: Hz, kHz, MHz, GHz, Default “Hz” |
| Parameter Range | Full frequency range |
| Return | Float, unit: Hz |
| Default | Maximum frequency |
| Menu | LEVEL > Flatness > Set > Fill Type > Manual Step |
| Example | :CORRection:FLATness:STARtfreq 200 MHz |

### 3.5.2.14 Flatness List Stop Freq ([:SOURce]:CORRection:FLATness:STOPfreq)

| | |
|---|---|
| Command Format | [:SOURce]:CORRection:FLATness:STOPfreq <freq><br>[:SOURce]:CORRection:FLATness:STOPfreq? |

| | |
|---|---|
| Instruction | Set the stop frequency when you want to fill the flatness list with the sensor and filling type is “Manual Step”.<br>Get the stop frequency when you want to fill the flatness list with the sensor and filling type is “Manual Step”. |
| Parameter Type | Float, unit: Hz, kHz, MHz, GHz, Default “Hz” |
| Parameter Range | Full frequency range |
| Return | Float, unit: Hz |
| Default | Maximum frequency |
| Menu | LEVEL > Flatness > Set > Fill Type > Manual Step |
| Example | :CORRection:FLATness:STOPfreq 500 MHz |

### 3.5.2.15 Flatness List Fill Space ([:SOURce]:CORRection:FLATness:LINStep)

| | |
|---|---|
| Command Format | [:SOURce]:CORRection:FLATness:SPACe LINear\|LOGarithmic<br>[:SOURce]:CORRection:FLATness:SPACe? |
| Instruction | Set the fill space in Manual Step Fill Type.<br>Get the fill space in Manual Step Fill Type. |
| Parameter Type | Enumeration |
| Parameter Range | LINear\|LOGarithmic |
| Return | Enumeration |
| Default | LINear |
| Menu | LEVEL > Flatness > Set > Fill Type > Manual Step |
| Example | :CORRection:FLATness:SPACe LINear |

### 3.5.2.16 Flatness List Linear Step ([:SOURce]:CORRection:FLATness:LINStep)

| | |
|---|---|
| Command Format | [:SOURce]:CORRection:FLATness:LINStep <freq><br>[:SOURce]:CORRection:FLATness:LINStep? |
| Instruction | Set the linear frequency step in Manual Step Fill Type. |

| | |
|---|---|
| | Get the linear frequency step in Manual Step Fill Type. |
| Parameter Type | Float, unit: Hz, kHz, MHz, GHz, Default “Hz” |
| Parameter Range | None |
| Return | Float, unit: Hz |
| Default | None |
| Menu | LEVEL > Flatness > Set > Fill Type > Manual Step |
| Example | :CORRection:FLATness:LINStep 200 MHz |

### 3.5.2.17 Flatness List Log Step ([:SOURce]:CORRection:FLATness:LOGStep)

| | |
|---|---|
| Command Format | [:SOURce]:CORRection:FLATness:LOGStep <value><br>[:SOURce]:CORRection:FLATness:LOGStep? |
| Instruction | Set the log frequency step in Manual Step Fill Type.<br>Get the log frequency step in Manual Step Fill Type. |
| Parameter Type | Float, unit: % |
| Parameter Range | None |
| Return | Float, unit: % |
| Default | None |
| Menu | LEVEL > Flatness > Set > Fill Type > Manual Step |
| Example | :CORRection:FLATness:LOGStep 20 |

### 3.5.2.18 Flatness List Points ([:SOURce]:CORRection:FLATness:POINt)

| | |
|---|---|
| Command Format | [:SOURce]:CORRection:FLATness:POINt <points><br>[:SOURce]:CORRection:FLATness:POINt? |
| Instruction | Set the points of flatness list in Manual Step Fill Type.<br>Get the points of flatness list in Manual Step Fill Type. |
| Parameter Type | Integer |

| | |
|---|---|
| Parameter Range | 2 ~ 500 |
| Return | Integer |
| Default | 11 |
| Menu | LEVEL > Flatness > Set > Fill Type > Manual Step |
| Example | :CORRection:FLATness:POINt 5 |

### 3.5.2.19 Level Control ([:SOURce]:POWer:SPC:STATe)

| | |
|---|---|
| Command Format | [:SOURce]:POWer:SPC:STATe ON\|OFF\|1\|0<br>[:SOURce]:POWer:SPC:STATe? |
| Instruction | Activate/Deactivate power control using the sensor.<br>Get level control state. |
| Parameter Type | Boolean |
| Parameter Range | ON\|OFF\|1\|0 |
| Return | Boolean |
| Default | 0 |
| Menu | SENSOR > Level Control |
| Example | POWer:SPC:STATe ON |

### 3.5.2.20 Target Level ([:SOURce]:POWer:SPC:TARGet)

| | |
|---|---|
| Command Format | [:SOURce]:POWer:SPC:TARGet <power><br>[:SOURce]:POWer:SPC:TARGet? |
| Instruction | Set the nominal level expected at the input of the sensor.<br>Get the nominal level expected at the input of the sensor. |
| Parameter Type | Float, unit: dBm, dBmV, dBuV, V, W, Default: dBm |
| Parameter Range | -120 dBm ~ 20 dBm |
| Return | Float, unit: dBm |
| Default | 0 dBm |
| Menu | SENSOR > Level Control > Target Level |
| Example | POWer:SPC:TARGet 0 |

### 3.5.2.21 Level Limit ([:SOURce]:POWer:LIMit)

| | |
|---|---|
| Command Format | [:SOURce]:POWer:LIMit <power><br>[:SOURce]:POWer:LIMit? |
| Instruction | Set an upper limit for the RF output power.<br>Get an upper limit for the RF output power. |
| Parameter Type | Float, unit: dBm, dBmV, dBuV, V, W, Default: dBm |
| Parameter Range | -110 dBm ~ 20 dBm |
| Return | Float, unit: dBm |
| Default | 0 dBm |
| Menu | SENSOR > Level Control > Level Limit |
| Example | POWer:LIMit 1 |

### 3.5.2.22 Catch Range ([:SOURce]:POWer:SPC:CRANge)

| | |
|---|---|
| Command Format | [SOURce]:POWer:SPC:CRANge <power><br>[SOURce]:POWer:SPC:CRANge? |
| Instruction | Set the capture range of the control system.<br>Get the capture range of the control system. |
| Parameter Type | Float |
| Parameter Range | 0 dB ~ 50 dB |
| Return | Float, unit: dB |
| Default | 0 dB |
| Menu | SENSOR > Level Control > Catch Range |
| Example | :POWer:SPC:CRANge 5 |

## 3.5.3[:SOURce]:SWEep Subsystem

### 3.5.3.1 Sweep State ([:SOURce]:SWEep:STATe)

| | |
|---|---|
| Command | [:SOURce]:SWEep:STATe OFF\|FREQuency\|LEVel\|LEV_FREQ |

| | |
|---|---|
| Format | [:SOURce]:SWEep:STATe? |
| Instruction | Activate frequency or/and level sweep. |
| Parameter Type | Enumeration |
| Parameter Range | OFF\|FREQuency\|LEVel\|LEV_FREQ |
| Return | Enumeration |
| Default | OFF |
| Menu | SWEEP > Sweep State |
| Example | :SWEep:STATe OFF |

### 3.5.3.2 Sweep Type ([:SOURce]:SWEep:TYPE)

| | |
|---|---|
| Command Format | [:SOURce]:SWEep:TYPE LIST\|STEP<br>[:SOURce]:SWEep:TYPE? |
| Instruction | Set sweep type.<br>Get sweep type. |
| Parameter Type | Enumeration |
| Parameter Range | LIST\|STEP |
| Return | Enumeration |
| Default | STEP |
| Menu | SWEEP > Step Sweep / List Sweep |
| Example | :SWEep:TYPE STEP |

### 3.5.3.3 Start Frequency ([:SOURce]:SWEep:STEP:STARt:FREQuency)

| | |
|---|---|
| Command Format | [:SOURce]:SWEep:STEP:STARt:FREQuency <freq><br>[:SOURce]:SWEep:STEP:STARt:FREQuency? |
| Instruction | Set the start frequency for the sweep mode.<br>Get the start frequency for the sweep mode. |
| Parameter Type | Float, unit: Hz, kHz, MHz, GHz, Default “Hz” |

| | |
|---|---|
| Parameter Range | Full frequency range. |
| Return | Float, unit: Hz |
| Default | Maximum frequency |
| Menu | SWEEP > Step Sweep > Start Freq |
| Example | :SWEep:STEP:STARt:FREQuency 1 GHz |

### 3.5.3.4 Stop Frequency ([:SOURce]:SWEep:STEP:STOP:FREQuency)

| | |
|---|---|
| Command Format | [:SOURce]:SWEep:STEP:STOP:FREQuency <freq><br>[:SOURce]:SWEep:STEP:STOP:FREQuency? |
| Instruction | Set the stop frequency for the sweep mode.<br>Get the stop frequency for the sweep mode. |
| Parameter Type | Float, unit: Hz, kHz, MHz, GHz, Default “Hz” |
| Parameter Range | Full frequency range. |
| Return | Float, unit: Hz |
| Default | Maximum frequency |
| Menu | SWEEP > Step Sweep > Stop Freq |
| Example | :SWEep:STEP:STOP:FREQuency 1 GHz |

### 3.5.3.5 Start Level ([:SOURce]:SWEep:STEP:STARt:LEVel)

| | |
|---|---|
| Command Format | [:SOURce]:SWEep:STEP:STARt:LEVel <level><br>[:SOURce]:SWEep:STEP:STARt:LEVel? |
| Instruction | Set the start level for the sweep mode.<br>Get the start level for the sweep mode. |
| Parameter Type | Float, unit: dBm, dBmV, dBuV, V, W, Default: dBm |
| Parameter Range | Full level range. |
| Return | Float, unit: dBm |
| Default | -110 dBm |
| Menu | SWEEP > Step Sweep > Start Level |
| Example | :SWEep:STEP:STARt:LEVel 0 dBm |

### 3.5.3.6 Stop Level ([:SOURce]:SWEep:STEP:STOP:LEVel)

| | |
|---|---|
| Command Format | [:SOURce]:SWEep:STEP:STOP:LEVel <level><br>[:SOURce]:SWEep:STEP:STOP:LEVel? |
| Instruction | Set the stop level for the sweep mode.<br>Get the stop level for the sweep mode. |
| Parameter Type | Float, unit: dBm, dBmV, dBuV, V, W, Default dBm |
| Parameter Range | Full level range. |
| Return | Float, unit: dBm |
| Default | -110 dBm |
| Menu | SWEEP > Step Sweep > Stop Level |
| Example | :SWEep:STEP:STOP:LEVel 0 dBm |

### 3.5.3.7 Dwell Time ([:SOURce]:SWEep:STEP:DWELl)

| | |
|---|---|
| Command Format | [:SOURce]:SWEep:STEP:DWELl <time><br>[:SOURce]:SWEep:STEP:DWELl? |
| Instruction | Set the duration of the individual sweep steps.<br>Get the duration of the individual sweep steps. |
| Parameter Type | Float, unit: ns, us, ms, s |
| Parameter Range | 10 ms ~ 100 s |
| Return | Float, unit: s |
| Default | 30 ms |
| Menu | SWEEP > Step Sweep > Dwell Time |
| Example | :SWEep:STEP:DWELl 20 ms |

### 3.5.3.8 Sweep Points ([:SOURce]:SWEep:STEP:POINts)

| | |
|---|---|
| Command Format | [:SOURce]:SWEep:STEP:POINts <points><br>[:SOURce]:SWEep:STEP:POINts? |
| Instruction | Set the number of steps in an RF sweep.<br>Get the number of steps in an RF sweep. |

| | |
|---|---|
| Parameter Type | Integer |
| Parameter Range | 2 ~ 65535 |
| Return | Integer |
| Default | 11 |
| Menu | SWEEP > Step Sweep > Sweep Points |
| Example | :SWEep:STEP:POINts 2 |

### 3.5.3.9 Sweep Shape ([:SOURce]:SWEep:STEP:SHAPe)

| | |
|---|---|
| Command Format | [:SOURce]:SWEep:STEP:SHAPe TRIangle\|SAWtooth<br>[:SOURce]:SWEep:STEP:SHAPe? |
| Instruction | Select the waveform shape of the sweep signal.<br>Get the waveform shape of the sweep signal. |
| Parameter Type | Enumeration |
| Parameter Range | TRIangle\|SAWtooth |
| Return | Enumeration |
| Default | SAWTooth |
| Menu | SWEEP > Step Sweep > Sweep Shape |
| Example | :SWEep:STEP:SHAPe TRIangle |

### 3.5.3.10 Sweep Space ([:SOURce]:SWEep:STEP:SPACe)

| | |
|---|---|
| Command Format | [:SOURce]:SWEep:STEP:SPACe LINear\|LOGarithmic<br>[:SOURce]:SWEep:STEP:SPACe? |
| Instruction | Select the sweep spacing.<br>Get the sweep spacing. |
| Parameter Type | Enumeration |
| Parameter Range | LINear\|LOGarithmic |
| Return | Enumeration |
| Default | LINear |

| | |
|---|---|
| Menu | SWEEP > Step Sweep > Sweep Space |
| Example | :SWEep:STEP:SPACe LOGarithmic |

### 3.5.3.11 Sweep Step in Linear Sweep Space ([:SOURce]:SWEep[:FREQuency]:STEP[:LINear])

| | |
|---|---|
| Command Format | [:SOURce]:SWEep[:FREQuency]:STEP[:LINear] <freq><br>[:SOURce]:SWEep[:FREQuency]:STEP[:LINear]? |
| Instruction | Set the sweep step in linear sweep space.<br>Get the sweep step in linear sweep space. |
| Parameter Type | Float, unit: Hz, kHz, MHz, GHz, Default “Hz” |
| Parameter Range | None |
| Return | Float, unit: Hz |
| Default | 0 |
| Menu | SWEEP > Step Sweep > Freq Step Linear |
| Example | :SWEep:STEP 200 MHz |

### 3.5.3.12 Sweep Step in Log Sweep Space ([:SOURce]:SWEep[:FREQuency]:STEP:LOGarithmic)

| | |
|---|---|
| Command Format | [:SOURce]:SWEep[:FREQuency]:STEP:LOGarithmic <value><br>[:SOURce]:SWEep[:FREQuency]:STEP:LOGarithmic? |
| Instruction | Set the sweep step in logarithmic sweep space.<br>Get the sweep step in logarithmic sweep space. |
| Parameter Type | Float, unit: % |
| Parameter Range | None |
| Return | Float, unit: % |
| Default | 0 |
| Menu | SWEEP > Step Sweep > Freq Step Log |
| Example | :SWEep:STEP:LOGarithmic 20 |

### 3.5.3.13 Sweep List Add Row ([:SOURce]:SWEep:LIST:ADDList)

| | |
|---|---|
| Command Format | [:SOURce]:SWEep:LIST:ADDList <freq>,<level>,<time> |
| Instruction | Insert a new row. |
| Parameter Type | Freq: Float, unit: Hz, kHz, MHz, GHz, Default “Hz”<br>Level: Float, unit: dBm<br>Time: Float, unit: ns, us, ms, s |
| Parameter Range | Full frequency range, full frequency range, 10.0 ms ~ 100.0 s |
| Return | None |
| Default | None |
| Menu | SWEEP > List Sweep > [+] |
| Example | :SWEep:LIST:ADDList 1 GHz,0 dBm,1 s |

### 3.5.3.14 Sweep List Delete Row ([:SOURce]:SWEep:LIST:DELete)

| | |
|---|---|
| Command Format | [:SOURce]:SWEep:LIST:DELete <row> |
| Instruction | Delete the sweep list pair. |
| Parameter Type | Integer |
| Parameter Range | 1 to count of sweep list. |
| Return | None |
| Default | None |
| Menu | SWEEP > List Sweep > [-] |
| Example | :SWEep:LIST:DELete 1 |

### 3.5.3.15 Sweep List Edit ([:SOURce]:SWEep:LIST:CHANGe)

| | |
|---|---|
| Command Format | [:SOURce]:SWEep:LIST:CHANGe <row>,<freq>,<power>,<time> |
| Instruction | Edit sweep list pair value. |
| Parameter Type | Integer, Float, unit: Hz, kHz, MHz, GHz, Float, unit: dBm, dBmV, dBuV, V, W,<br>Default: dBm, Float, unit: ns, us, ms, s |

| | |
|---|---|
| Parameter Range | Raw: 1 ~ count of pair.<br>Freq: Full frequency range.<br>Power: Full level range.<br>time: 10 ms ~ 100 s. |
| Return | None |
| Default | None |
| Menu | SWEEP > List Sweep |
| Example | :SWEep:LIST:CHANGe 1,1 GHz,1 dBm, 1 s |

### 3.5.3.16 Sweep List Row Count ([:SOURce]:SWEep:LIST:CPOint?)

| | |
|---|---|
| Command Format | [:SOURce]:SWEep:LIST:CPOint? |
| Instruction | Get how many rows in sweep list. |
| Parameter Type | None |
| Parameter Range | None |
| Return | Float |
| Default | 1 |
| Menu | SWEEP > List Sweep |
| Example | :SWEep:LIST:CPOint? |

### 3.5.3.17 Show Sweep List ([:SOURce]:SWEep:LIST:LIST?)

| | |
|---|---|
| Command Format | [:SOURce]:SWEep:LIST:LIST? <begin_ row>,<end_row> |
| Instruction | View starting row to end row data. |
| Parameter Type | Integer, Integer |
| Parameter Range | 1 to count of sweep list. |
| Return | String |
| Default | None |
| Menu | SWEEP > List Sweep |
| Example | :SWEep:LIST:LIST? 1,3 |

### 3.5.3.18 Sweep List Clear ([:SOURce]:SWEep:LIST:INITialize:PRESet)

| | |
|---|---|
| Command Format | [:SOURce]:SWEep:LIST:INITialize:PRESet |
| Instruction | Restore the scan list of the factory default settings. |
| Parameter Type | None |
| Parameter Range | None |
| Return | None |
| Default | None |
| Menu | SWEEP > List Sweep > Clear |
| Example | SWEep:LIST:INITialize:PRESet |

### 3.5.3.19 Sweep List Initialize From Step ([:SOURce]:SWEep:LIST:INITialize:FSTep)

| | |
|---|---|
| Command Format | [:SOURce]:SWEep:LIST:INITialize:FSTep |
| Instruction | Regenerate the sweep list based on the data points of the current step sweep settings. |
| Parameter Type | None |
| Parameter Range | None |
| Return | None |
| Default | None |
| Menu | SWEEP > List Sweep |
| Example | SWEep:LIST:INITialize:FSTep |

### 3.5.3.20 Sweep List Load ([:SOURce]:SWEep:LOAD)

| | |
|---|---|
| Command Format | [:SOURce]:CORRection:LOAD <file_name> |

| | |
|---|---|
| Instruction | Load existing sweep list file. |
| Parameter Type | String |
| Parameter Range | None |
| Return | None |
| Default | None |
| Menu | SWEEP > List Sweep > Load |
| Example | :SWEep:LOAD test.lsw |

### 3.5.3.21 Sweep List Store ([:SOURce]:SWEep:STORe)

| | |
|---|---|
| Command Format | [:SOURce]:CORRection:STORe <file_name> |
| Instruction | Save the sweep data in the list. |
| Parameter Type | String |
| Parameter Range | None |
| Return | None |
| Default | None |
| Menu | SWEEP > List Sweep > Store |
| Example | :SWEep:STORe test.lsw |

### 3.5.3.22 Sweep Direction ([:SOURce]:SWEep:DIRect)

| | |
|---|---|
| Command Format | [:SOURce]:SWEep:DIRect FWD\|REV<br>[:SOURce]:SWEep:DIRect? |
| Instruction | Select the direction for sweep. |
| Parameter Type | Enumeration |
| Parameter Range | FWD\|REV |
| Return | Enumeration |
| Default | FWD |
| Menu | SWEEP > Direction |

| | |
|---|---|
| Example | :SWEep:DIRect REV |

### 3.5.3.23 Sweep Mode ([:SOURce]:SWEep:MODE)

| | |
|---|---|
| Command Format | [:SOURce]:SWEep:MODE CONTinue\|SINGle<br>[:SOURce]:SWEep:MODE? |
| Instruction | Set the cycle mode of the sweep.<br>Get the cycle mode of the sweep. |
| Parameter Type | Enumeration |
| Parameter Range | CONTinue\|SINGle |
| Return | Enumeration |
| Default | CONTinue |
| Menu | SWEEP > Sweep Mode |
| Example | :SWEep:MODE SINGle |

### 3.5.3.24 Trigger Mode ([:SOURce]:SWEep:SWEep:TRIGger:TYPE)

| | |
|---|---|
| Command Format | [:SOURce]:SWEep:SWEep:TRIGger:TYPE AUTO\|KEY\|BUS\|EXT<br>[:SOURce]:SWEep:SWEep:TRIGger:TYPE? |
| Instruction | Select the trigger mode.<br>Get the trigger mode. |
| Parameter Type | Enumeration |
| Parameter Range | AUTO\|KEY\|BUS\|EXT |
| Return | Enumeration |
| Default | AUTO |
| Menu | SWEEP > Trigger Mode |
| Example | :SWEep:SWEep:TRIGger:TYPE KEY |

### 3.5.3.25 Point Trigger ([:SOURce]:SWEep:POINt:TRIGger:TYPE)

| | |
|---|---|
| Command Format | [:SOURce]:SWEep:POINt:TRIGger:TYPE AUTO\|KEY\|BUS\|EXT<br>[:SOURce]:SWEep:POINt:TRIGger:TYPE? |

| | |
|---|---|
| Instruction | Select the point trigger.<br>Get the point trigger. |
| Parameter Type | Enumeration |
| Parameter Range | AUTO\|KEY\|BUS\|EXT |
| Return | Enumeration |
| Default | AUTO |
| Menu | SWEEP > Point Trigger |
| Example | :SWEep:POINt:TRIGger:TYPE KEY |

### 3.5.3.26 Trigger Slope ([:SOURce]:INPut:TRIGger:SLOPe)

| | |
|---|---|
| Command Format | [:SOURce]:INPut:TRIGger:SLOPe POSitive\|NEGative<br>[:SOURce]:INPut:TRIGger:SLOPe? |
| Instruction | Select the trigger slope.<br>Get the trigger slope. |
| Parameter Type | Enumeration |
| Parameter Range | POSitive\|NEGative |
| Return | Enumeration |
| Default | POSitive |
| Menu | SWEEP > Trigger Slope |
| Example | :INPut:TRIGger:SLOPe NEGative |

## 3.5.4[:SOURce]:MODulation Subsystem

### 3.5.4.1 Modulation State ([:SOURce]:MODulation)

| | |
|---|---|
| Command Format | [:SOURce]:MODulation ON\|OFF\|1\|0<br>[:SOURce]:MODulation? |
| Instruction | Switch the modulations on and off.<br>Get the modulations state. |
| Parameter | Boolean |

| | |
|---|---|
| Type | |
| Parameter Range | ON\|OFF\|1\|0 |
| Return | Boolean |
| Default | 0 |
| Menu | MOD |
| Example | MODulation ON |

## 3.5.5[:SOURce]:AM Subsystem

### 3.5.5.1 AM State ([:SOURce]:AM:STATe)

| | |
|---|---|
| Command Format | [:SOURce]:AM:STATe ON\|OFF\|1\|0<br>[:SOURce]:AM:STATe? |
| Instruction | Activate/Deactivate amplitude modulation.<br>Get the AM state. |
| Parameter Type | Boolean |
| Parameter Range | ON\|OFF\|1\|0 |
| Return | Boolean |
| Default | 0 |
| Menu | AM > AM State |
| Example | :AM:STATe ON |

### 3.5.5.2 AM Shape ([:SOURce]:AM:WAVEform)

| | |
|---|---|
| Command Format | [:SOURce]:AM:WAVEform SINE\|SQUAre<br>[:SOURce]:AM:WAVEform? |
| Instruction | Set the AM modulation waveform.<br>Get the AM modulation waveform. |
| Parameter Type | Enumeration |
| Parameter Range | SINE\|SQUAre |

| | |
|---|---|
| Return | Enumeration |
| Default | SINE |
| Menu | AM > AM Shape |
| Example | :AM:WAVEform SINE |

### 3.5.5.3 AM Source ([:SOURce]:AM:SOURce)

| | |
|---|---|
| Command Format | [:SOURce]:AM:SOURce INTernal\|EXTernal\|INT+EXT<br>[:SOURce]:AM:SOURce? |
| Instruction | Select the modulation signal source for amplitude modulation.<br>Get the AM source. |
| Parameter Type | Enumeration |
| Parameter Range | INTernal\|EXTernal\|INT+EXT |
| Return | Enumeration |
| Default | INTernal |
| Menu | AM > AM Source |
| Example | :AM:SOURce EXTernal |

### 3.5.5.4 AM Depth ([:SOURce]:AM:DEPTh)

| | |
|---|---|
| Command Format | [:SOURce]:AM:DEPTh <value><br>[:SOURce]:AM:DEPTh? |
| Instruction | Set the overall modulation depth of the amplitude modulation in percent.<br>Get the AM depth. |
| Parameter Type | Float |
| Parameter Range | 0.1 % ~ 100 % |
| Return | Float |
| Default | 50 % |
| Menu | AM > AM Depth |
| Example | :AM:DEPTh 0.2 |

### 3.5.5.5 AM Rate ([:SOURce]:AM:FREQuency)

| | |
|---|---|
| Command Format | [:SOURce]:AM:FREQuency <value><br>[:SOURce]:AM:FREQuency? |
| Instruction | Set the AM modulation frequency.<br>Get the AM modulation frequency. |
| Parameter Type | Float, unit: Hz, kHz, MHz, GHz, Default “Hz” |
| Parameter Range | Sine: 0.01 Hz ~ 100 kHz<br>Square: 0.01 Hz ~ 20 kHz |
| Return | Float, unit: Hz |
| Default | 1 kHz |
| Menu | AM > AM Rate |
| Example | :AM:FREQuency 10 kHz |

### 3.5.5.6 AM Sensitivity ([:SOURce]:AM:SENSitivity)

| | |
|---|---|
| Command Format | [:SOURce]:AM:SENSitivity? |
| Instruction | Query the input sensitivity of the external modulation input in %/V. |
| Parameter Type | None |
| Parameter Range | None |
| Return | Float, unit: %/V |
| Default | 0 %/V |
| Menu | AM > AM Sensitivity |
| Example | AM:SENSitivity? |

## 3.5.6[:SOURce]:FM Subsystem

### 3.5.6.1 FM State ([:SOURce]:FM:STATe)

| | |
|---|---|
| Command Format | [:SOURce]:FM:STATe ON\|OFF\|1\|0<br>[:SOURce]:FM:STATe? |

| | |
|---|---|
| Instruction | Activate/Deactivate FM.<br>Get the FM state. |
| Parameter Type | Boolean |
| Parameter Range | ON\|OFF\|1\|0 |
| Return | Boolean |
| Default | 0 |
| Menu | FM > FM State |
| Example | :FM:STATe ON |

### 3.5.6.2 FM Shape ([:SOURce]:FM:WAVEform)

| | |
|---|---|
| Command Format | [:SOURce]:FM:WAVEform SINE\|SQUAre<br>[:SOURce]:FM:WAVEform? |
| Instruction | Selects the shape of FM.<br>Get the shape of FM. |
| Parameter Type | Enumeration |
| Parameter Range | SINE\|SQUAre |
| Return | Enumeration |
| Default | SINE |
| Menu | FM > FM Shape |
| Example | :FM:WAVEform SQUAre |

### 3.5.6.3 FM Source ([:SOURce]:FM:SOURce)

| | |
|---|---|
| Command Format | [:SOURce]:FM:SOURce INTernal\|EXTernal\|INT+EXT<br>[:SOURce]:FM:SOURce? |
| Instruction | Select the modulation signal source for frequency modulation.<br>Get the FM Source. |
| Parameter Type | Enumeration |
| Parameter Range | INTernal\|EXTernal\|INT+EXT |

| | |
|---|---|
| Return | Enumeration |
| Default | INTernal |
| Menu | FM > FM Source |
| Example | :FM:SOURce EXTernal |

### 3.5.6.4 FM Deviation ([:SOURce]:FM:DEViation)

| | |
|---|---|
| Command Format | [:SOURce]:FM:DEViation <value><br>[:SOURce]:FM:DEViation? |
| Instruction | Set deviation value<br>Get deviation value |
| Parameter Type | Float, unit: Hz, kHz, MHz, GHz |
| Parameter Range | 0.01 Hz ~ 1 MHz |
| Return | Float, unit: Hz |
| Default | 100 kHz |
| Menu | FM > FM Deviation |
| Example | :FM:DEViation 500 kHz |

### 3.5.6.5 FM Rate ([:SOURce]:FM:FREQuency)

| | |
|---|---|
| Command Format | [:SOURce]:FM:FREQuency <value><br>[:SOURce]:FM:FREQuency? |
| Instruction | Set the FM modulation frequency.<br>Get the FM modulation frequency. |
| Parameter Type | Float, unit: Hz, kHz, MHz, GHz, Default “Hz” |
| Parameter Range | INTernal: SQUAre: 0.01 Hz ~ 20 kHz SINE: 0.01 Hz ~ 100 kHz<br>INT+EXT: SQUAre: 0.01 Hz ~ 20 kHz SINE: 0.01 Hz ~ 100 kHz |
| Return | Float, unit: Hz |
| Default | 10 kHz |
| Menu | FM > FM Rate |
| Example | :FM:FREQuency 40 kHz |

### 3.5.6.6 FM Sensitivity ([:SOURce]:FM:SENSitivity)

| | |
|---|---|
| Command Format | [:SOURce]:FM:SENSitivity? |
| Instruction | Displays the input sensitivity of the FM EXT input in Hz/V. |
| Parameter Type | None |
| Parameter Range | None |
| Return | Float unit: Hz/V |
| Default | 0 Hz/V |
| Menu | FM > FM Sensitivity |
| Example | FM:SENSitivity? |

## 3.5.7[:SOURce]:PM Subsystem

### 3.5.7.1 PM State ([:SOURce]:PM:STATe)

| | |
|---|---|
| Command Format | [:SOURce]:PM:STATe ON\|OFF\|1\|0<br>[:SOURce]:PM:STATe? |
| Instruction | Activate/Deactivate phase modulation.<br>Get the PM state. |
| Parameter Type | Boolean |
| Parameter Range | ON\|OFF\|1\|0 |
| Return | Boolean |
| Default | 0 |
| Menu | PM > PM State |
| Example | :PM:STATe ON |

### 3.5.7.2 PM Shape ([:SOURce]:PM:WAVEform)

| | |
|---|---|
| Command Format | [:SOURce]:PM:WAVEform SINE\|SQUAre<br>[:SOURce]:PM:WAVEform? |

| | |
|---|---|
| Instruction | Selects the shape of PM.<br>Get the shape of PM. |
| Parameter Type | Enumeration |
| Parameter Range | SINE\|SQUAre |
| Return | Enumeration |
| Default | SINE |
| Menu | PM > PM Shape |
| Example | :PM:WAVEform SINE |

### 3.5.7.3 PM Source ([:SOURce]:PM:SOURce)

| | |
|---|---|
| Command Format | [:SOURce]:PM:SOURce INTernal\|EXTernal\|INT+EXT<br>[:SOURce]:PM:SOURce? |
| Instruction | Select the modulation signal source for phase modulation.<br>Get the PM source. |
| Parameter Type | Enumeration |
| Parameter Range | INTernal\|EXTernal\|INT+EXT |
| Return | Enumeration |
| Default | INTernal |
| Menu | PM > PM Source |
| Example | :PM:SOURce EXTernal |

### 3.5.7.4 PM Deviation ([:SOURce]:PM:DEViation)

| | |
|---|---|
| Command Format | [:SOURce]:PM:DEViation <value><br>[:SOURce]:PM:DEViation? |
| Instruction | Set the modulation deviation of the phase modulation.<br>Get the modulation deviation of the phase modulation. |
| Parameter Type | Float, unit: rad |
| Parameter Range | 0.00001 rad ~ 5 rad |

| | |
|---|---|
| Return | Float, unit: rad |
| Default | 1 rad |
| Menu | PM > PM Deviation |
| Example | :PM:DEViation 2 |

## 3.5.7.5 PM Rate ([:SOURce]:PM:FREQuency)

| | |
|---|---|
| Command Format | [:SOURce]:PM:FREQuency <value><br>[:SOURce]:PM:FREQuency? |
| Instruction | Set the PM modulation frequency.<br>Get the PM modulation frequency. |
| Parameter Type | Float, unit: Hz, kHz, MHz, GHz, Default “Hz” |
| Parameter Range | INTernal: SQUAre:0.01 Hz ~ 20 kHz SINE: 0.01 Hz ~ 100 kHz<br>INT+EXT: SQUAre:0.01 Hz ~ 20 kHz 0.01 Hz ~ 100 kHz |
| Return | Float, unit: Hz |
| Default | 10 kHz |
| Menu | PM > PM Rate |
| Example | :PM:FREQuency 10 kHz |

## 3.5.7.6 PM Sensitivity ([:SOURce]:PM:SENSitivity)

| | |
|---|---|
| Command Format | [:SOURce]:PM:SENSitivity? |
| Instruction | Query the input sensitivity of the EXT MOD input in rad/v. |
| Parameter Type | None |
| Parameter Range | None |
| Return | Float, unit: rad/V |
| Default | 0 rad/V |
| Menu | PM > PM Sensitivity |
| Example | PM:SENSitivity? |

## 3.5.8[:SOURce]:PULM Subsystem

### 3.5.8.1 Pulse State ([:SOURce]:PULM:STATe)

| | |
|---|---|
| Command Format | [:SOURce]:PULM:STATe ON\|OFF\|1\|0<br>[:SOURce]:PULM:STATe? |
| Instruction | Activate/Deactivate the pulse modulation.<br>Get the state of pulse modulation. |
| Parameter Type | Boolean |
| Parameter Range | ON\|OFF\|1\|0 |
| Return | Boolean |
| Default | 0 |
| Menu | PULSE > Pulse State |
| Example | PULM:STAT ON |

### 3.5.8.2 Pulse Out ([:SOURce]:PULM:OUT:STATe)

| | |
|---|---|
| Command Format | [:SOURce]:PULM:OUT:STATe ON\|OFF\|1\|0<br>[:SOURce]:PULM:OUT:STATe? |
| Instruction | Configures the signal at the PULSE OUT connector.<br>Get the Output status. |
| Parameter Type | Boolean |
| Parameter Range | ON\|OFF\|1\|0 |
| Return | Boolean |
| Default | 0 |
| Menu | PULSE > Pulse Out |
| Example | PULM:OUT ON |

### 3.5.8.3 Pulse Source ([:SOURce]:PULM:SOURce)

| | |
|---|---|
| Command | [:SOURce]:PULM:SOURce INTernal\|EXTernal |

| | |
|---|---|
| Format | [:SOURce]:PULM:SOURce? |
| Instruction | Select the source for the pulse modulation signal.<br>Get the source for the pulse modulation signal. |
| Parameter Type | Enumeration |
| Parameter Range | INTernal\|EXTernal |
| Return | Enumeration |
| Default | INTernal |
| Menu | PULSE > Pulse Source |
| Example | PULM:SOUR INTernal |

### 3.5.8.4 Pulse Polarity ([:SOURce]:PULM:POLarity)

| | |
|---|---|
| Command Format | [:SOURce]:PULM:POLarity NORMal\|INVerted<br>[:SOURce]:PULM:POLarity? |
| Instruction | Set the period of the generated pulse. The period determines the repetition frequency of the internal signal.<br>Get the period of the generated pulse. |
| Parameter Type | Enumeration |
| Parameter Range | NORMal\|INVerted |
| Return | Enumeration |
| Default | NORMal |
| Menu | PULSE > Pulse Polarity |
| Example | PULM:POL INV |

### 3.5.8.5 Pulse Mode ([:SOURce]:PULM:MODE)

| | |
|---|---|
| Command Format | [:SOURce]:PULM:MODE SINGle\|DOUBle\|PTRain<br>[:SOURce]:PULM:MODE? |
| Instruction | Set the mode of the pulse generator.<br>Get the mode of the pulse generator. |
| Parameter Type | Enumeration |

| | |
|---|---|
| Parameter Range | SINGle\|DOUBle\|PTRain<br>SINGle<br>Enables single pulse generation.<br>DOUBle<br>Enables double pulse generation. The two pulses are generated in one pulse period.<br>PTRain<br>A user-defined pulse train is generated The pulse train is defined by value pairs of on and off times that can be entered in a pulse train list. |
| Return | Enumeration |
| Default | SINGle |
| Menu | PULSE > Pulse Mode |
| Example | PULM:MODE DOUB |

### 3.5.8.6 Pulse Period ([:SOURce]:PULM:PERiod)

| | |
|---|---|
| Command Format | [:SOURce]:PULM:PERiod <value><br>[:SOURce]:PULM:PERiod? |
| Instruction | Set the period of the generated pulse. The period determines the repetition frequency of the internal signal.<br>Get the period of the generated pulse. |
| Parameter Type | Float, unit: ns, us, ms, s |
| Parameter Range | 40 ns ~ 300 s |
| Return | Float, unit: s |
| Default | 10 ms |
| Menu | PULSE > Pulse Period |
| Example | PULM:PER 220 us |

### 3.5.8.7 Pulse Width ([:SOURce]:PULM:WIDTh)

| | |
|---|---|
| Command Format | [:SOURce]:PULM:WIDTh <value><br>[:SOURce]:PULM:WIDTh? |
| Instruction | Set the width of the generated pulse.<br>Get the width of the generated pulse. |

| | |
|---|---|
| Parameter Type | Float, unit: ns, us, ms, s |
| Parameter Range | 20 ns ~ 300 s |
| Return | Float, unit: s |
| Default | 2 ms |
| Menu | PULSE > Pulse Width |
| Example | PULM:WIDT 33 us |

### 3.5.8.8 Double Pulse Delay ([:SOURce]:PULM:DOUBle:DELay)

| | |
|---|---|
| Command Format | [:SOURce]:PULM:DOUBle:DELay <value><br>[:SOURce]:PULM:DOUBle:DELay? |
| Instruction | Set the delay from the start of the first pulse to the start of the second pulse.<br>Get the delay from the start of the first pulse to the start of the second pulse. |
| Parameter Type | Float, unit: ns, us, ms, s |
| Parameter Range | 20 ns ~ 300 s |
| Return | Float, unit: s |
| Default | 4 ms |
| Menu | PULSE > Double Pulse Delay |
| Example | :PULM:DOUBle:DELay 2 ms |

### 3.5.8.9 #2 Width ([:SOURce]:PULM:DOUBle:WIDTh)

| | |
|---|---|
| Command Format | [:SOURce]:PULM:DOUBle:WIDTh <time><br>[:SOURce]:PULM:DOUBle:WIDTh? |
| Instruction | Set the width of the second pulse in case of double pulse generation.<br>Get the width of the second pulse in case of double pulse generation. |
| Parameter Type | Float, unit: ns, us, ms, s |
| Parameter Range | 20 ns ~ 300 s |

| | |
|---|---|
| Return | Float, unit: s |
| Default | 2 ms |
| Menu | PULSE > #2 Width |
| Example | PULM:DOUBle:WIDTh 2 s |

### 3.5.8.10 Pulse Train Add Row ([:SOURce]:PULM:TRAin:PAIR)

| | |
|---|---|
| Command Format | [:SOURce]:PULM:TRAin:PAIR |
| Instruction | Add default train pair value. |
| Parameter Type | None |
| Parameter Range | None |
| Return | None |
| Default | None |
| Menu | PULSE > Pulse Train > [+] |
| Example | PULM:TRAin:PAIR |

### 3.5.8.11 Pulse Train Delete ([:SOURce]:PULM:TRAin:DELete)

| | |
|---|---|
| Command Format | [:SOURce]:PULM:TRAin:DELete <row> |
| Instruction | Delete the train pair. |
| Parameter Type | Integer |
| Parameter Range | 1 ~ 2047 |
| Return | None |
| Default | None |
| Menu | PULSE > Pulse Train > [-] |
| Example | PULM:TRAin:DELete 5 |

### 3.5.8.12 Pulse Train Edit ([:SOURce]:PULM:TRAin:CHANGe)

| | |
|---|---|
| Command | [:SOURce]:PULM:TRAin:CHANGe <raw>,<on_time>,<off_time>,<count> |

| | |
|---|---|
| Format | |
| Instruction | Edit train pair value. |
| Parameter Type | Integer, Float, unit: ns, us, ms, s, Float, unit: ns, us, ms, s, Integer |
| Parameter Range | Raw: 1 ~ count of pair.<br>On time: 10 ns ~ 300 s.<br>Off time: 10 ns ~ 300 s.<br>Count: 1 ~ 65535 |
| Return | None |
| Default | None |
| Menu | PULSE > Pulse Train |
| Example | :PULM:TRAin:CHANGe 1,10 ms,20 ms,3 |

### 3.5.8.13 List Pulse Train ([:SOURce]:PULM:TRAin:LIST?)

| | |
|---|---|
| Command Format | [:SOURce]:PULM:TRAin:LIST? <begin_row>,<end_row> |
| Instruction | View starting row to end row data. |
| Parameter Type | Integer, Integer |
| Parameter Range | Begin_row: 1 ~ the count of pulse list.<br>End_row: Begin_row ~ the count of pulse list. |
| Return | String |
| Default | None |
| Menu | PULSE > Pulse Train |
| Example | :PULM:TRAin:LIST? 1,3 |

### 3.5.8.14 Pulse Train Count ([:SOURce]:PULM:TRAin:COUNt?)

| | |
|---|---|
| Command Format | [:SOURce]:PULM:TRAin:COUNt? |
| Instruction | Get count of train list. |
| Parameter Type | None |
| Parameter Range | None |

| | |
|---|---|
| Return | Integer |
| Default | 1 |
| Menu | PULSE > Pulse Train |
| Example | :PULM:TRAin:COUNt? |

### 3.5.8.15 Pulse Train Clear ([:SOURce]:PULM:TRAin:CLEAr)

| | |
|---|---|
| Command Format | [:SOURce]:PULM:TRAin:CLEAr |
| Instruction | Clear train pair list. |
| Parameter Type | None |
| Parameter Range | None |
| Return | None |
| Default | None |
| Menu | PULSE > Pulse Train > Store |
| Example | PULM:TRAin:CLEAr |

### 3.5.8.16 Pulse Train Load ([:SOURce]:PULM:TRAin:LOAD)

| | |
|---|---|
| Command Format | [:SOURce]:PULM:TRAin:LOAD <file> |
| Instruction | Load train pair list. |
| Parameter Type | String |
| Parameter Range | None |
| Return | None |
| Default | None |
| Menu | PULSE > Pulse Train > Load |
| Example | PULM:TRAin:LOAD test.pulstrn |

### 3.5.8.17 Pulse Train Store ([:SOURce]:PULM:TRAin:STORE)

| | |
|---|---|
| Command | [:SOURce]:PULM:TRAin:STORE <file> |

| | |
|---|---|
| Format | |
| Instruction | Store train pair list. |
| Parameter Type | String |
| Parameter Range | None |
| Return | None |
| Default | None |
| Menu | PULSE > Pulse Train |
| Example | PULM:TRAin:STORE test.pulstrn |

### 3.5.8.18 Trigger Out ([:SOURce]:PULM:TRIGger:STATe)

| | |
|---|---|
| Command Format | [:SOURce]:PULM:TRIGger:STATe ON\|OFF\|1\|0<br>[:SOURce]:PULM:TRIGger:STATe? |
| Instruction | Set the trigger output status.<br>Get the trigger output status. |
| Parameter Type | Boolean |
| Parameter Range | ON\|OFF\|1\|0 |
| Return | Boolean |
| Default | 1 |
| Menu | PULSE > Trigger Out |
| Example | PULM:TRIGger:STATe ON |

### 3.5.8.19 Pulse Trigger ([:SOURce]:PULM:TRIGger:MODE)

| | |
|---|---|
| Command Format | [:SOURce]:PULM:TRIGger:MODE AUTO\|KEY\|EXTernal\|EGATe<br>[:SOURce]:PULM:TRIGger:MODE? |
| Instruction | Select the trigger mode for pulse modulation.<br>Get the trigger mode for pulse modulation. |
| Parameter Type | Enumeration |
| Parameter Range | AUTO\| KEY\|EXTernal\|EGATe |

| | |
|---|---|
| Return | Enumeration |
| Default | AUTO |
| Menu | PULSE > Pulse Trigger |
| Example | PULM:TRIG:MODE EXTernal |

### 3.5.8.20 Trig Polarity ([:SOURce]:PULM:TRIGger:EXTernal:GATE:POLarity)

| | |
|---|---|
| Command Format | [:SOURce]:PULM:TRIGger:EXTernal:GATE:POLarity NORMal\|INVerted<br>[:SOURce]:PULM:TRIGger:EXTernal:GATE:POLarity? |
| Instruction | Select the polarity of the gate signal.<br>Get the polarity of the gate signal. |
| Parameter Type | Enumeration |
| Parameter Range | NORMal\|INVerted |
| Return | Enumeration |
| Default | NORMal |
| Menu | PULSE > Pulse Polarity |
| Example | PULM:TRIG:EXT:GATE:POL NORMal |

### 3.5.8.21 Trig Delay ([:SOURce]:PULM:DELay)

| | |
|---|---|
| Command Format | [:SOURce]:PULM:DELay <value><br>[:SOURce]:PULM:DELay? |
| Instruction | Set the pulse delay.<br>Get the pulse delay. |
| Parameter Type | Float, unit: ns, us, ms, s |
| Parameter Range | 140 ns ~ 300 s |
| Return | Float, unit: s |
| Default | 140 ns |
| Menu | PULSE > Trig Delay |
| Example | PULM:DEL 30 ms |

### 3.5.8.22 Trig Slope ([:SOURce]:PULM:TRIGger:EXTernal:SLOPe)

| | |
|---|---|
| Command Format | [:SOURce]:PULM:TRIGger:EXTernal:SLOPe NEGative\|POSitive<br>[:SOURce]:PULM:TRIGger:EXTernal:SLOPe? |
| Instruction | Set the polarity of the active slope of an applied trigger at the PULSE EXT connector.<br>Get the polarity of the active slope of an applied trigger at the PULSE EXT connector. |
| Parameter Type | Enumeration |
| Parameter Range | NEGative\|POSitive |
| Return | Enumeration |
| Default | POSitive |
| Menu | PULSE > Trig Slope |
| Example | PULM:TRIG:EXT:SLOP NEG |

## 3.5.9[:SOURce]:LFOutput Subsystem

### 3.5.9.1 LF State ([:SOURce]:LFOutput[:STATe])

| | |
|---|---|
| Command Format | [:SOURce]:LFOutput ON\|OFF\|1\|0<br>[:SOURce]:LFOutput? |
| Instruction | Activate/deactivate the LF output.<br>Get the LF output state. |
| Parameter Type | Boolean |
| Parameter Range | ON\|OFF\|1\|0 |
| Return | Boolean |
| Default | 0 |
| Menu | LF > LF State |
| Example | LFOutput ON |

### 3.5.9.2 LF Level ([:SOURce]:LFOutput:VOLTage)

| | |
|---|---|
| Command Format | [:SOURce]:LFOutput:VOLTage <volt><br>[:SOURce]:LFOutput:VOLTage? |
| Instruction | Set the voltage of the LF output signal.<br>Get the voltage of the LF output signal. |
| Parameter Type | Float, unit: dBm, dBmV, dBuV, V, W, Default: V (Here V is the peak-peak value) |
| Parameter Range | 1 mVpp ~ 3 Vpp |
| Return | Float, unit: Vpp |
| Default | 0.5 Vpp |
| Menu | LF > LF Voltage |
| Example | LFOutput:VOLTage 2 V |

### 3.5.9.3 LF Offset ([:SOURce]:LFOutput:OFFSEt)

| | |
|---|---|
| Command Format | [:SOURce]:LFOutput:OFFSEt <volt><br>[:SOURce]:LFOutput:OFFSEt? |
| Instruction | Set the voltage offset of the LF output signal.<br>Get the voltage offset of the LF output signal. |
| Parameter Type | Float, unit: dBm, dBmV, dBuV, V, W, Default: V |
| Parameter Range | $\left\|LFoffset\right\| \le \max(2.5V - \frac{1}{2}LEVEL, 2V)$ |
| Return | Float, unit: V |
| Default | 0 V |
| Menu | LF > LF Offset |
| Example | LFOutput:OFFSEt 1 V |

### 3.5.9.4 LF Frequency ([:SOURce]:LFOutput:FREQuency)

| | |
|---|---|
| Command Format | [:SOURce]:LFOutput:FREQuency <freq><br>[:SOURce]:LFOutput:FREQuency? |
| Instruction | Set LF out put frequency.<br>Get LF out put frequency. |

| | |
|---|---|
| | If signal source "Internal" is set, the instrument performs the analog modulations<br>(AM/FM /PM) with this frequency. |
| Parameter Type | Float, unit: Hz, kHz, MHz, GHz, Default “Hz” |
| Parameter Range | 0.01 Hz ~ 1 MHz |
| Return | Float, unit: Hz |
| Default | 1 kHz |
| Menu | LF > LF Frequency |
| Example | LFOutput:FREQuency 10 kHz |

### 3.5.9.5 LF Shape ([:SOURce]:LFOutput:SHAPe)

| | |
|---|---|
| Command Format | [:SOURce]:LFOutput:SHAPe SINE\|SQUare\|TRIangle\|SAWTooth\|DC<br>[:SOURce]:LFOutput:SHAPe? |
| Instruction | Select the shape of the LF signal.<br>Get the shape of the LF signal. |
| Parameter Type | Enumeration |
| Parameter Range | SINE\|SQUare\|TRIangle\|SAWTooth\|DC |
| Return | Enumeration |
| Default | SINE |
| Menu | LF > LF Shape |
| Example | LFOutput:SHAPe TRIangle |

### 3.5.9.6 LF Phase ([:SOURce]:LFOutput:PHASe)

| | |
|---|---|
| Command Format | [:SOURce]:LFOutput:PHASe <deg><br>[:SOURce]:LFOutput:PHASe? |
| Instruction | Set the phase of the LF output signal.<br>Get the phase of the LF output signal. |
| Parameter Type | Float, unit: deg |
| Parameter | -360 deg ~ 360 deg |

| | |
|---|---|
| Range | |
| Return | Float, unit: deg |
| Default | 0 deg |
| Menu | LF > LF Phase |
| Example | LFOutput:PHASe 20 |

## 3.5.10[:SOURce]:LFOutput:SWEep Subsystem

### 3.5.10.1 Sweep State ([:SOURce]:LFOutput:SWEep)

| | |
|---|---|
| Command Format | [:SOURce]:LFOutput:SWEep ON\|OFF\|0\|1<br>[:SOURce]:LFOutput:SWEep? |
| Instruction | Activate/Deactivate the LF frequency sweep signal generation.<br>Get the state of LF frequency sweep. |
| Parameter Type | Boolean |
| Parameter Range | ON\|OFF\|0\|1 |
| Return | Boolean |
| Default | 0 |
| Menu | LF Sweep > LF State |
| Example | :LFOutput:SWEep 1 |

### 3.5.10.2 Sweep Direction ([:SOURce]:LFOutput:SWEep:DIRect)

| | |
|---|---|
| Command Format | [:SOURce]:LFOutput:SWEep:DIRect UP\|DOWN<br>[:SOURce]:LFOutput:SWEep:DIRect? |
| Instruction | Set the sweep direction.<br>Get the sweep direction. |
| Parameter Type | Enumeration |
| Parameter Range | UP\|DOWN |
| Return | Enumeration |
| Default | UP |

| | |
|---|---|
| Menu | LF Sweep > Sweep Direction |
| Example | :LFOutput:SWEep:DIRect DOWN |

### 3.5.10.3 Start Freq ([:SOURce]:LFOutput:SWEep:STARt:FREQuency)

| | |
|---|---|
| Command Format | [:SOURce]:LFOutput:SWEep:STARt:FREQuency <freq><br>[:SOURce]:LFOutput:SWEep:STARt:FREQuency? |
| Instruction | Set the start frequency of sweep mode.<br>Get the start frequency of sweep mode. |
| Parameter Type | Float, unit: Hz, kHz, MHz, GHz, Default "Hz" |
| Parameter Range | 0.01 Hz ~ Stop frequency |
| Return | Float, unit: Hz |
| Default | 500 Hz |
| Menu | LF Sweep > Start Freq |
| Example | :LFOutput:SWEep:STARt:FREQuency 100 |

### 3.5.10.4 Stop Freq ([:SOURce]:LFOutput:SWEep:STOP:FREQuency)

| | |
|---|---|
| Command Format | [:SOURce]:LFOutput:SWEep:STOP:FREQuency <freq><br>[:SOURce]:LFOutput:SWEep:STOP:FREQuency? |
| Instruction | Set the stop frequency of sweep mode.<br>Get the stop frequency of sweep mode. |
| Parameter Type | Float, unit: Hz, kHz, MHz, GHz, Default "Hz" |
| Parameter Range | Start frequency ~ Maximum frequency of LF |
| Return | Float, unit: Hz |
| Default | 1.5 kHz |
| Menu | LF Sweep > Stop Freq |
| Example | :LFOutput:SWEep:STOP:FREQuency 1000 |

### 3.5.10.5 Center Freq ([:SOURce]:LFOutput:SWEep:CENTer:FREQuency)

| | |
|---|---|
| Command Format | [:SOURce]:LFOutput:SWEep:CENTer:FREQuency <freq><br>[:SOURce]:LFOutput:SWEep:CENTer:FREQuency? |
| Instruction | Set the center frequency of sweep mode.<br>Get the center frequency of sweep mode. |
| Parameter Type | Float, unit: Hz, kHz, MHz, GHz, Default "Hz" |
| Parameter Range | 0.01 Hz ~ Maximum frequency of LF |
| Return | Float, unit: Hz |
| Default | 1 kHz |
| Menu | LF Sweep > Center Freq |
| Example | :LFOutput:SWEep:CENTer:FREQuency 550 |

### 3.5.10.6 Freq Span ([:SOURce]:LFOutput:SWEep:SPAN:FREQuency)

| | |
|---|---|
| Command Format | [:SOURce]:LFOutput:SWEep:SPAN:FREQuency <freq><br>[:SOURce]:LFOutput:SWEep:SPAN:FREQuency? |
| Instruction | Set the center frequency of sweep mode.<br>Get the center frequency of sweep mode. |
| Parameter Type | Float, unit: Hz, kHz, MHz, GHz, Default "Hz" |
| Parameter Range | 0 Hz ~ Maximum frequency of LF – 0.01 Hz |
| Return | Float, unit: Hz |
| Default | 1 kHz |
| Menu | LF Sweep > Freq Span |
| Example | :LFOutput:SWEep:SPAN:FREQuency 550 |

### 3.5.10.7 Sweep Time ([:SOURce]:LFOutput:SWEep:DWELl)

| | |
|---|---|
| Command | [:SOURce]:LFOutput:SWEep:DWELl <time> |

| | |
|---|---|
| Format | [:SOURce]:LFOutput:SWEep:DWELl? |
| Instruction | Set the sweep time of sweep mode.<br>Get the sweep time of sweep mode. |
| Parameter Type | Float, unit: ns, us, ms, s |
| Parameter Range | 1 ms ~ 500 s |
| Return | Float, unit: s |
| Default | 1 s |
| Menu | LF Sweep > Sweep Time |
| Example | :LFOutput:SWEep:DWELl 2 s |

### 3.5.10.8 Trigger Mode ([:SOURce]:LFOutput:SWEep:TRIGger:TYPE)

| | |
|---|---|
| Command Format | [:SOURce]:LFOutput:SWEep:TRIGger:TYPE AUTO\|KEY\|BUS\|EXT<br>[:SOURce]:LFOutput:SWEep:TRIGger:TYPE? |
| Instruction | Select the LF frequency sweep trigger mode.<br>Get the LF frequency sweep trigger mode. |
| Parameter Type | Enumeration |
| Parameter Range | AUTO\|KEY\|BUS\|EXT |
| Return | Enumeration |
| Default | AUTO |
| Menu | LF Sweep > Trigger Mode |
| Example | :LFOutput:SWEep:TRIGger:TYPE KEY |

### 3.5.10.9 Sweep Shape ([:SOURce]:LFOutput:SWEep:SHAPe)

| | |
|---|---|
| Command Format | [:SOURce]:LFOutput:SWEep:SHAPe TRIangle\|SAWTooth<br>[:SOURce]:LFOutput:SWEep:SHAPe? |
| Instruction | Select the waveform shape of the sweep signal.<br>Get the waveform shape of the sweep signal. |
| Parameter Type | Enumeration |
| Parameter | TRIangle\|SAWTooth |

| | |
|---|---|
| Range | |
| Return | Enumeration |
| Default | SAWTooth |
| Menu | LF Sweep > Sweep Shape |
| Example | :LFOutput:SWEep:SHAPe TRIangle |

### 3.5.10.10 Sweep Space ([:SOURce]:LFOutput:SWEep:SPACing)

| | |
|---|---|
| Command Format | [:SOURce]:LFOutput:SWEep:SPACing LINear\|LOGarithmic<br>[:SOURce]:LFOutput:SWEep:SPACing? |
| Instruction | Select the mode for the calculation of the frequency sweep intervals.<br>Get the mode for the calculation of the frequency sweep intervals. |
| Parameter Type | Enumeration |
| Parameter Range | LINear\|LOGarithmic |
| Return | Enumeration |
| Default | LINear |
| Menu | LF Sweep > Sweep Space |
| Example | :LFOutput:SWEep:SPACing LOGarithmic |

# 3.6Sense Subsystem

## 3.6.1Sensor Info (:SENSe[:POWer]:TYPE)

| | |
|---|---|
| Command Format | :SENSe[:POWer]:TYPE? |
| Instruction | Query the type of sensor connected to the POWER SENSOR connector. |
| Parameter Type | None |
| Parameter Range | None |
| Return | String |
| Default | None |
| Menu | SENSOR > Sensor Info |

| | |
|---|---|
| Example | SENSe:TYPE? |

### 3.6.2Sensor State (:SENSe[:POWer]:STATus)

| | |
|---|---|
| Command Format | :SENSe[:POWer]:STATus OFF\|ON\|0\|1<br>:SENSe[:POWer]:STATus? |
| Instruction | Set the sensor state.<br>Get the sensor state. |
| Parameter Type | Boolean |
| Parameter Range | OFF\|ON\|0\|1 |
| Return | Boolean |
| Default | 0 |
| Menu | SENSOR > Sensor State |
| Example | SENSe:STATus ON |

### 3.6.3Measurement (:SENSe[:POWer]:VALue)

| | |
|---|---|
| Command Format | :SENSe[:POWer]:VALue? |
| Instruction | Indicate the current reading of the sensor. |
| Parameter Type | None |
| Parameter Range | None |
| Return | Float, unit: dBm |
| Default | None |
| Menu | SENSOR > Measurement |
| Example | SENSe:VALue? |

### 3.6.4Statistics State (:SENSe[:POWer]:STATIStics:STATe)

| | |
|---|---|
| Command Format | :SENSe[:POWer]:STATIStics:STATe ON\|OFF\|1\|0<br>:SENSe[:POWer]:STATIStics:STATe? |

| | |
|---|---|
| Instruction | Set statistics state.<br>Get statistics state. |
| Parameter Type | Boolean |
| Parameter Range | ON\|OFF\|1\|0 |
| Return | Boolean |
| Default | 0 |
| Menu | SENSOR > Statistics |
| Example | SENSe:STATIStics:STATe ON |

### 3.6.5Statistics Value (:READ[:POWer])

| | |
|---|---|
| Command Format | :READ[:POWer]? |
| Instruction | Indicate the measured mean value and maximum value. |
| Parameter Type | None |
| Parameter Range | None |
| Return | String |
| Default | None |
| Menu | SENSOR > Statistics |
| Example | READ? |

### 3.6.6Statistics Max Value (:SENSe[:POWer]:STATIStics:MAX?)

| | |
|---|---|
| Command Format | :SENSe[:POWer]:STATIStics:MAX? |
| Instruction | Indicate the measured maximum value. |
| Parameter Type | None |
| Parameter Range | None |
| Return | Float, unit: dBm |

| | |
|---|---|
| Default | None |
| Menu | SENSOR > Statistics |
| Example | SENSe:STATIStics:MAX? |

### 3.6.7Statistics Min Value (:SENSe[:POWer]:STATIStics:MIN?)

| | |
|---|---|
| Command Format | :SENSe[:POWer]:STATIStics:MIN? |
| Instruction | Indicate the measured minimum value. |
| Parameter Type | None |
| Parameter Range | None |
| Return | Float, unit: dBm |
| Default | None |
| Menu | SENSOR > Statistics |
| Example | SENSe:STATIStics:MIN? |

### 3.6.8Statistics Mean Value (:SENSe[:POWer]:STATIStics:AVG?)

| | |
|---|---|
| Command Format | :SENSe[:POWer]:STATIStics:AVG? |
| Instruction | Indicate the measured mean value. |
| Parameter Type | None |
| Parameter Range | None |
| Return | Float, unit: dBm |
| Default | None |
| Menu | SENSOR > Statistics |
| Example | SENSe:STATIStics:AVG? |

### 3.6.9 Statistics Count (:SENSe[:POWer]:STATIStics:COUNt?)

| | |
|---|---|
| Command Format | :SENSe[:POWer]:STATIStics:COUNt? |
| Instruction | Indicate the measured times. |
| Parameter Type | None |
| Parameter Range | None |
| Return | Integer |
| Default | None |
| Menu | SENSOR > Statistics |
| Example | SENSe:STATIStics:COUNt? |

### 3.6.10 Statistics Clear (:SENSe[:POWer]:STATIStics:CLEAr?)

| | |
|---|---|
| Command Format | :SENSe[:POWer]:STATIStics:CLEAr? |
| Instruction | Clear statistics. |
| Parameter Type | None |
| Parameter Range | None |
| Return | None |
| Default | None |
| Menu | SENSOR > Statistics |
| Example | SENSe:STATIStics:CLEAr? |

### 3.6.11 Auto Zero (:CALibration:ZERO:TYPE)

| | |
|---|---|
| Command Format | :CALibration:ZERO:TYPE INTernal\|EXTernal<br>:CALibration:ZERO:TYPE? |
| Instruction | Select zero type.<br>Get zero type. |

| | |
|---|---|
| Parameter Type | Enumeration |
| Parameter Range | INTernal\|EXTernal |
| Return | Enumeration |
| Default | INTernal |
| Menu | SENSOR > Auto Zero |
| Example | CALibration:ZERO:TYPE EXTernal |

### 3.6.12 Zeroing (:SENSe[:POWer]:ZERO)

| | |
|---|---|
| Command Format | :SENSe[:POWer]:ZERO |
| Instruction | Perform zeroing of the sensor. |
| Parameter Type | None |
| Parameter Range | None |
| Return | None |
| Default | None |
| Menu | SENSOR > Click to perform zeroing |
| Example | :SENSe:ZERO |

### 3.6.13 Frequency Type (:SENSe[:POWer]:SOURce)

| | |
|---|---|
| Command Format | :SENSe[:POWer]:SOURce RF\|USER<br>:SENSe[:POWer]:SOURce? |
| Instruction | Select the signal source for the measurement.<br>Get the signal source for the measurement. |
| Parameter Type | Enumeration |
| Parameter Range | RF\|USER |
| Return | Enumeration |
| Default | RF |
| Menu | SENSOR > Frequency |

| | |
|---|---|
| Example | SENSe:SOURce RF |

## 3.6.14Frequency (:SENSe[:POWer]:FREQuency)

| | |
|---|---|
| Command Format | :SENSe[:POWer]:FREQuency <type><br>:SENSe[:POWer]:FREQuency? |
| Instruction | Set the frequency for frequency type " USER".<br>Get the frequency for frequency type " USER". |
| Parameter Type | Float, unit: Hz, kHz, MHz, GHz, Default “Hz” |
| Parameter Range | 9 kHz ~ 3.2 GHz |
| Return | Float, unit: Hz |
| Default | None |
| Menu | SENSOR > Frequency |
| Example | SENSe:FREQuency 1 MHz |

## 3.6.15Level Offset State (:SENSe[:POWer]:OFFSet:STATe)

| | |
|---|---|
| Command Format | :SENSe[:POWer]:OFFSet:STATe ON\|OFF\|0\|1<br>:SENSe[:POWer]:OFFSet:STATe? |
| Instruction | Switch the power offset switch status.<br>Get the power offset switch status. |
| Parameter Type | Boolean |
| Parameter Range | ON\|OFF\|0\|1 |
| Return | Boolean |
| Default | 0 |
| Menu | SENSOR > Level Offset |
| Example | SENSe:OFFSet:STATe ON |

## 3.6.16Level Offset (:SENSe[:POWer]:OFFSet)

| | |
|---|---|
| Command | :SENSe[:POWer]:OFFSet <power> |

| | |
|---|---|
| Format | :SENSe[:POWer]:OFFSet? |
| Instruction | The command enters a level offset which is added to the measured level value.<br>Get the level offset which is added to the measured level value. |
| Parameter Type | Float |
| Parameter Range | Limit by power sensor. |
| Return | Float, unit: dB |
| Default | 0 dB |
| Menu | SENSOR > Level Offset |
| Example | SENSe:OFFSet 10 |

## 3.6.17 Average Type (:SENSe[:POWer]:FILTer:TYPE)

| | |
|---|---|
| Command Format | :SENSe[:POWer]:FILTer:TYPE AUTO\|USER\|NSRatio<br>:SENSe[:POWer]:FILTer:TYPE? |
| Instruction | Select the averaging mode.<br>Get the averaging mode. |
| Parameter Type | Enumeration |
| Parameter Range | AUTO\|USER\|NSRatio |
| Return | Enumeration |
| Default | None |
| Menu | SENSOR > Averaging |
| Example | SENSe:FILTer:TYPE AUTO |

## 3.6.18 Average Times (:SENSe[:POWer]:FILTer:LENGth)

| | |
|---|---|
| Command Format | :SENSe[:POWer]:FILTer:LENGth <length><br>:SENSe[:POWer]:FILTer:LENGth? |
| Instruction | Set the average number times. |
| Parameter Type | Integer |
| Parameter | Limit by power sensor |

| | |
|---|---|
| Range | |
| Return | Float |
| Default | None |
| Menu | SENSOR > Averaging |
| Example | SENSe:FILTer:LENGth 10 |

### 3.6.19 Internal Noise (:SENSe[:POWer]:FILTer:NSRatio)

| | |
|---|---|
| Command Format | :SENSe[:POWer]:FILTer:NSRatio <noise><br>:SENSe[:POWer]:FILTer:NSRatio? |
| Instruction | The power sensor will control the internal noise that does not exceed the set value of the fixed noise parameter. |
| Parameter Type | Float, unit: dB |
| Parameter Range | Limit by power sensor. |
| Return | Float, unit: dB |
| Default | None |
| Menu | SENSOR > Averaging |
| Example | SENSe:FILTer:NSRatio 1 |

### 3.6.20 Logging (:SENSe[:POWer]:LOGGing:STATe)

| | |
|---|---|
| Command Format | :SENSe[:POWer]:LOGGing:STATe <state><br>:SENSe[:POWer]:LOGGing:STATe? |
| Instruction | Set logging state.<br>Get logging state. |
| Parameter Type | Boolean |
| Parameter Range | ON\|OFF\|1\|0 |
| Return | Boolean |
| Default | 0 |
| Menu | SENSOR > Logging |
| Example | SENSe:LOGGing:STATe ON |

# 4.Programming Examples

This chapter gives some examples for the programmer. In these examples you can see how to use the VISA or sockets, in combination with the commands have been described above to control the signal generator. By following these examples, you can develop many more applications.

# 4.1Examples of Using VISA

## 4.1.1Example of VC++

**Environment:** Win7 32bit system, Visual Studio

**The functions of this example:** use the NI-VISA, to control the device with USBTMC or TCP/IP access to do a write and read.

Follow the steps to finish the example:

1、Open Visual Studio, create a new VC++ win32 console project.

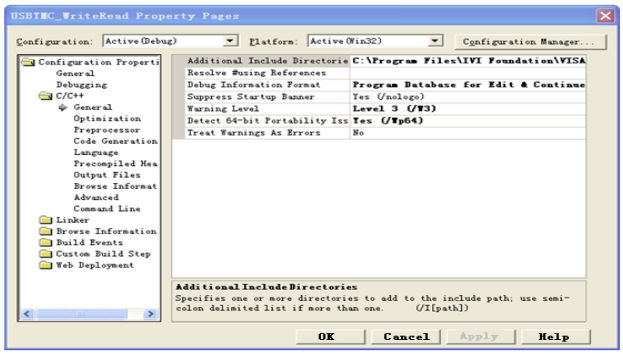
2、Set the project environment to use the NI-VISA lib, there are two ways to use NI-VISA, static or automatic:

(1) Static: find files: visa.h, visatype.h, visa32.lib in NI-VISA install path. Copy them to your project, and add them into project. In the projectname.cpp file, add the follow two lines:

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

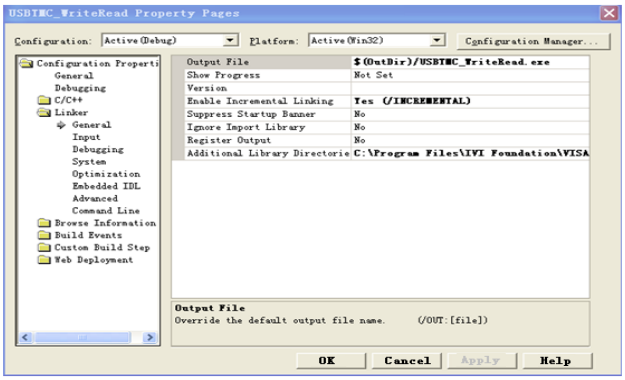
(2) Automatic:

Set the .h file include directory, the NI-VISA install path, in our computer we set the path is: C:\Program Files\IVI Foundation \VISA\WinNT\include. Set this path to project---properties---c/c++---General---Additional Include Directories: See the picture.

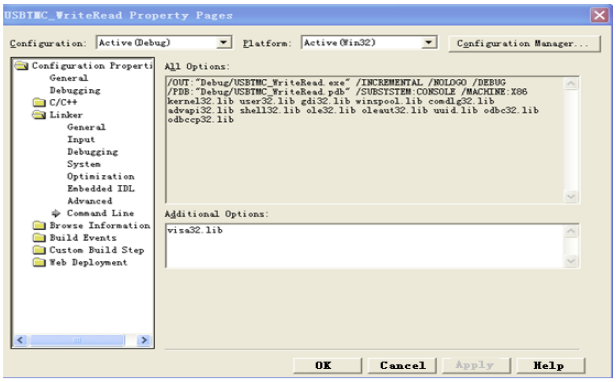



Set lib path set lib file:

Set lib path: the NI-VISA install path, in our computer we set the path is: C:\Program Files\IVI Foundation\VISA\WinNT\lib\msc. Set this path to project---properties---Linker---General---Additional Library Directories: as seen in the pictures below.




Set lib file: project---properties---Linker---Command Line---Additional Options: visa32.lib




Include visa.h file: In the projectname.cpp file:

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

3、Add codes:

(1) USBTMC access code.

Write a function Usbtmc_test:

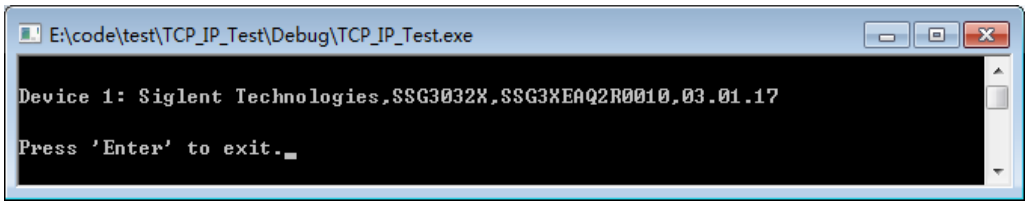
```
int Usbtmc_test()
{
 /* This code demonstrates sending synchronous read & write commands */
 /* to an USB Test & Measurement Class (USBTMC) instrument using  */
 /* NI-VISA      */
```

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

```
* device, we will get a handle to the instrument which we
* will use in later VISA functions. The AccessMode and Timeout
* parameters in this function are reserved for future
* functionality. These two parameters are given the value VI_NULL.*/
for (i=0; i<int(numInstrs); i++)
{
if (i> 0)
{
 viFindNext (findList, instrResourceString);
}
 status = viOpen (defaultRM, instrResourceString, VI_NULL, VI_NULL, &instr);
if (status<VI_SUCCESS)
{
 printf ("Cannot open a session to the device %d.\n", i+1);
 continue ;
}
/* * At this point we now have a session open to the USB TMC instrument.
* We will now use the viPrintf function to send the device the string "*IDN?\n",
* asking for the device's identification. */
char * cmmand ="*IDN?\n";
status = viPrintf (instr, cmmand);
if (status<VI_SUCCESS)
{
 printf ("Error writing to the device %d.\n", i+1);
 status = viClose (instr);
 continue;
}
/** Now we will attempt to read back a response from the device to
* the identification query that was sent. We will use the viScanf
* function to acquire the data.
* After the data has been read the response is displayed. */
status = viScanf(instr, "%t", buffer);
if (status<VI_SUCCESS)
{
 printf ("Error reading a response from the device %d.\n", i+1);
}
else
{
 printf ("\nDevice %d: %s\n", i+1, buffer);
}
 status = viClose (instr);
}
/** Now we will close the session to the instrument using
```

```
* viClose. This operation frees all system resources.      */
status = viClose (defaultRM);
printf("Press 'Enter' to exit.");
fflush(stdin);
getchar();
return 0;
}

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

**Run result.**



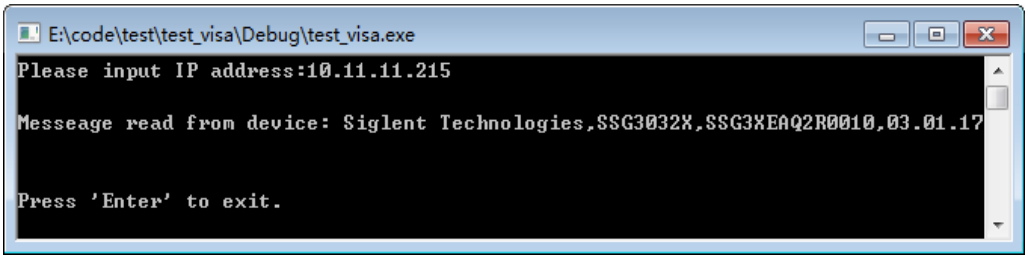

(2) TCP/IP access code.

Write a function TCP_IP_Test:

```
int TCP_IP_Test(char *pIP)
{
char outputBuffer[VI_FIND_BUFLEN];
ViSession defaultRM, instr;
ViStatus status;

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

strcat(head,pIP);
strcat(head,tail);
status = viOpen (defaultRM, head, VI_LOAD_CONFIG, VI_NULL, &instr);
if (status<VI_SUCCESS)
{
```

```
printf ("An error occurred opening the session\n");
viClose(defaultRM);
}
status = viPrintf(instr, "*idn?\n");
status = viScanf(instr, "%t", outputBuffer);
if (status<VI_SUCCESS)
{
printf("viRead failed with error code: %x \n",status);
viClose(defaultRM);
}
else
{
printf ("\nMesseage read from device: %*s\n", 0,outputBuffer);
}
status = viClose (instr);
status = viClose (defaultRM);
printf("Press 'Enter' to exit.");
fflush(stdin);
getchar();
return 0;
}

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

**Run result.**



## 4.1.2Example of VB

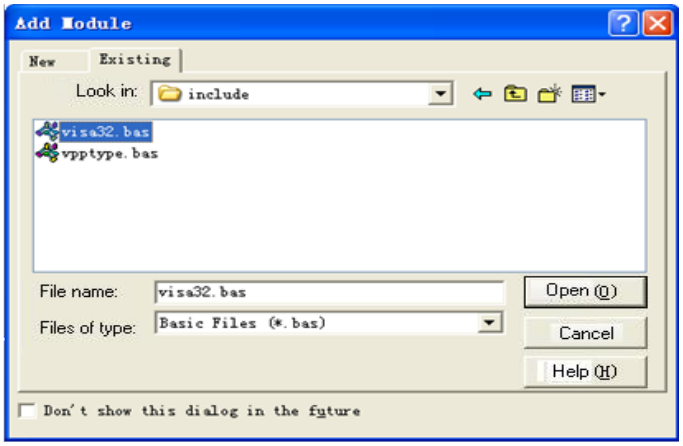
**Environment:** Win7 32bit system, Microsoft Visual Basic 6.0

**The function of this example:** Use the NI-VISA, to control the device with USBTMC and TCP/IP access to do a write and read.

Follow the steps to complete the example:

1、Open Visual Basic, build a standard application program project (Standard EXE)

2、Set the project environment to use the NI-VISA lib, Click the Existing tab of Project>>Add Existing Item. Search for the visa32.bas file in the include folder under the NI-VISA installation path and add the file.




This allows the VISA functions and VISA data types to be used in a program.

3、Add codes:

(1) USBTMC access code.

Write a function Usbtmc_test:

```
Private Function Usbtmc_test() As Long
  ' 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 As Long
  Dim instrsesn As Long
```

```
Dim numInstrs As Long
Dim findList As Long
Dim retCount As Long

Dim status As Long
Dim instrResourceString As String * VI_FIND_BUFLEN
Dim Buffer As String * MAX_CNT
Dim i As Integer
' First we must call viOpenDefaultRM to get the manager
' handle. We will store this handle in defaultRM.
status = viOpenDefaultRM(defaultRM)
If (status < VI_SUCCESS) Then
   resultTxt.Text = "Could not open a session to the VISA Resource Manager!"
   Usbtmc_test = status
   Exit Function
End If

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

' Now we will open VISA sessions to all USB TMC instruments.
' We must use the handle from viOpenDefaultRM and we must
' also use a string that indicates which instrument to open. This
' is called the instrument descriptor. The format for this string
' can be found in the function panel by right clicking on the
' descriptor parameter. After opening a session to the
' device, we will get a handle to the instrument which we
' will use in later VISA functions. The AccessMode and Timeout
' parameters in this function are reserved for future
' functionality. These two parameters are given the value VI_NULL.
For i = 0 To numInstrs
   If (i > 0) Then
      status = viFindNext(findList, instrResourceString)
   End If
   status = viOpen(defaultRM, instrResourceString, VI_NULL, VI_NULL, instrsesn)
   If (status < VI_SUCCESS) Then
```

```
        resultTxt.Text = "Cannot open a session to the device " + CStr(i + 1)
        GoTo NextFind
    End If

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

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

  Next i

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

(2) TCP/IP access code.

Write a function TCP_IP_Test:

```
Private Function TCP_IP_Test(ByVal ip As String) As Long
  Dim outputBuffer As String * VI_FIND_BUFLEN
  Dim defaultRM As Long
  Dim instrsesn As Long
  Dim status As Long
  Dim count As Long
```

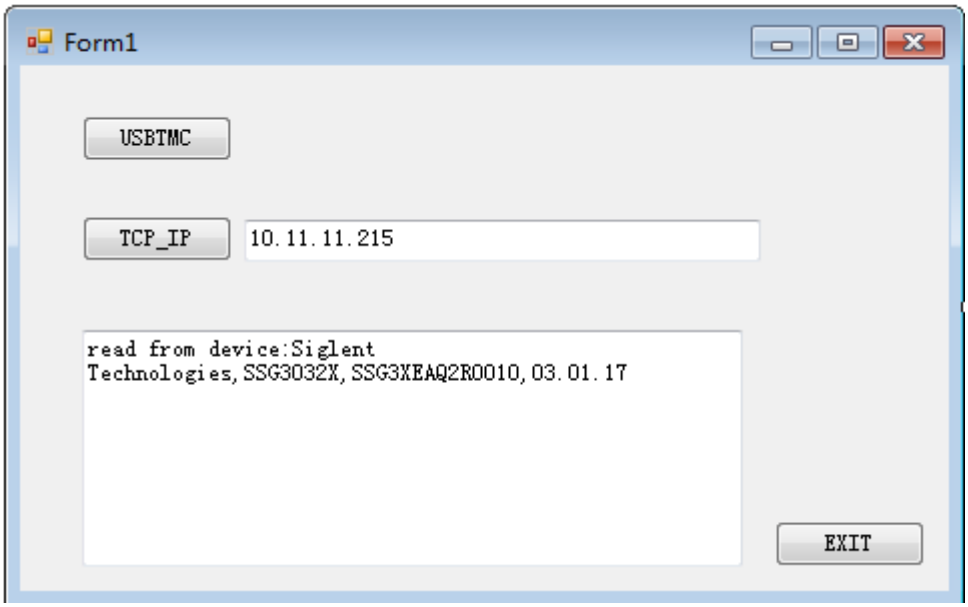
```
    ' First we will need to open the default resource manager.
    status = viOpenDefaultRM(defaultRM)
    If (status < VI_SUCCESS) Then
        resultTxt.Text = "Could not open a session to the VISA Resource Manager!"
        TCP_IP_Test = status
        Exit Function
    End If

    ' Now we will open a session via TCP/IP device
    status = viOpen(defaultRM, "TCPIP0::" + ip + "::INSTR", VI_LOAD_CONFIG, VI_NULL, instrsesn)
    If (status < VI_SUCCESS) Then
        resultTxt.Text = "An error occurred opening the session"
        viClose(defaultRM)
        TCP_IP_Test = status
        Exit Function
    End If

    status = viWrite(instrsesn, "*IDN?", 5, count)
    If (status < VI_SUCCESS) Then
        resultTxt.Text = "Error writing to the device."
    End If
    status = viRead(instrsesn, outputBuffer, VI_FIND_BUFLEN, count)
    If (status < VI_SUCCESS) Then
        resultTxt.Text = "Error reading a response from the device." + CStr(i + 1)
    Else
        resultTxt.Text = "read from device:" + outputBuffer
    End If
    status = viClose(instrsesn)
    status = viClose(defaultRM)
    TCP_IP_Test = 0
End Function
```

(3) Button control code:

```
Private Sub exitBtn_Click()
    End
End Sub
Private Sub tcpipBtn_Click()
    Dim stat As Long
    stat = TCP_IP_Test(ipTxt.Text)
    If (stat < VI_SUCCESS) Then
        resultTxt.Text = Hex(stat)
    End If
End Sub
```

```
Private Sub usbBtn_Click()
   Dim stat As Long
   stat = Usbtmc_test
   If (stat < VI_SUCCESS) Then
      resultTxt.Text = Hex(stat)
   End If
End Sub
```

**Run result:**




## 4.1.3Example of MATLAB

**Environment:** Win7 32bit system, MATLAB R2013a

**The function of this example:** Use the NI-VISA, to control the device with USBTMC or TCP/IP access to do a write and read.

Follow the steps to complete the example:

1、Open MATLAB, modify the **current directory**. In this demo, the current directory is modified to D:\USBTMC_TCPIP_Demo.

2、Click **File>>New>>Script** in the Matlab interface to create an empty M file

3、Add codes:

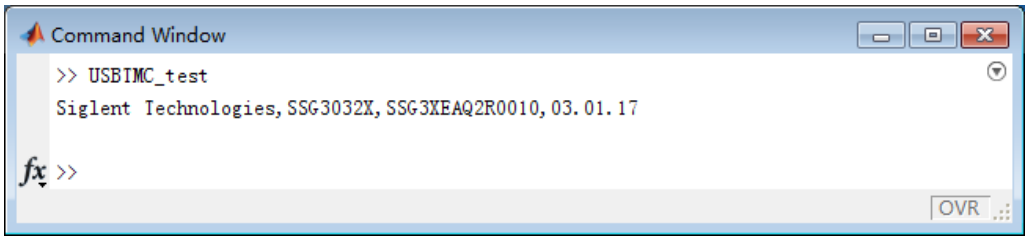
(1)USBTMC access code

Write a function Usbtmc_test.

```
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::0xF4EC::0x1501::0123456789::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
```

**Run result:**



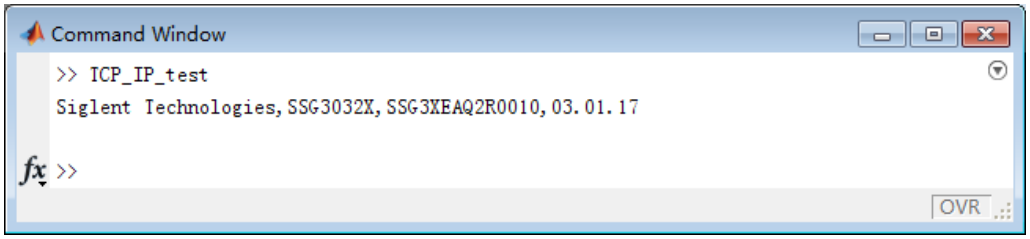

(2)TCP/IP access code.

Write a function TCP_IP_Test:

```
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::','10.11.11.215','::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
```

**Run result:**




## 4.1.4Example of LabVIEW

**Environment:** Win7 32bit system, LabVIEW 2011

**The functions of this example:** use the NI-VISA, to control the device with USBTMC and TCP/IP access to do a write and read.

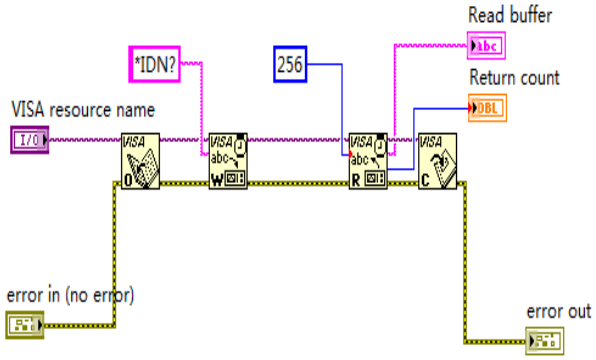
Follow the steps to complete the example:

1、Open LabVIEW, create a VI file.

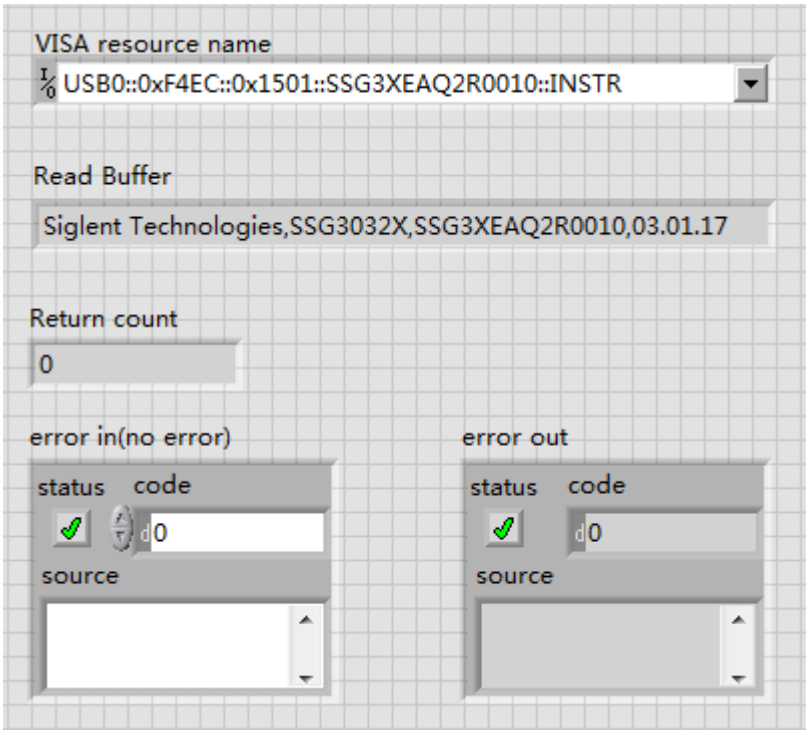
2、Add controls. Right-click in the **Front Panel** interface, select and add **VISA resource name**, error in, error out and some indicators from the Controls column.

3、Open the **Block Diagram** interface. Right-click on the **VISA resource name** and you can select and add the following functions from VISA Palette from the pop-up menu: **VISA Write**, **VISA Read**, **VISA Open** and **VISA Close**.

4、Connect them as shown in the figure below



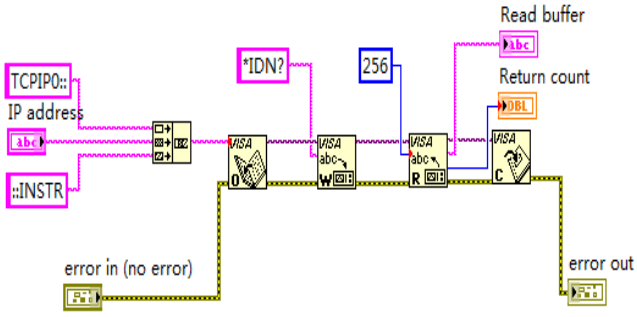
5、Select the device resource from the VISA Resource Name list box and run the program.



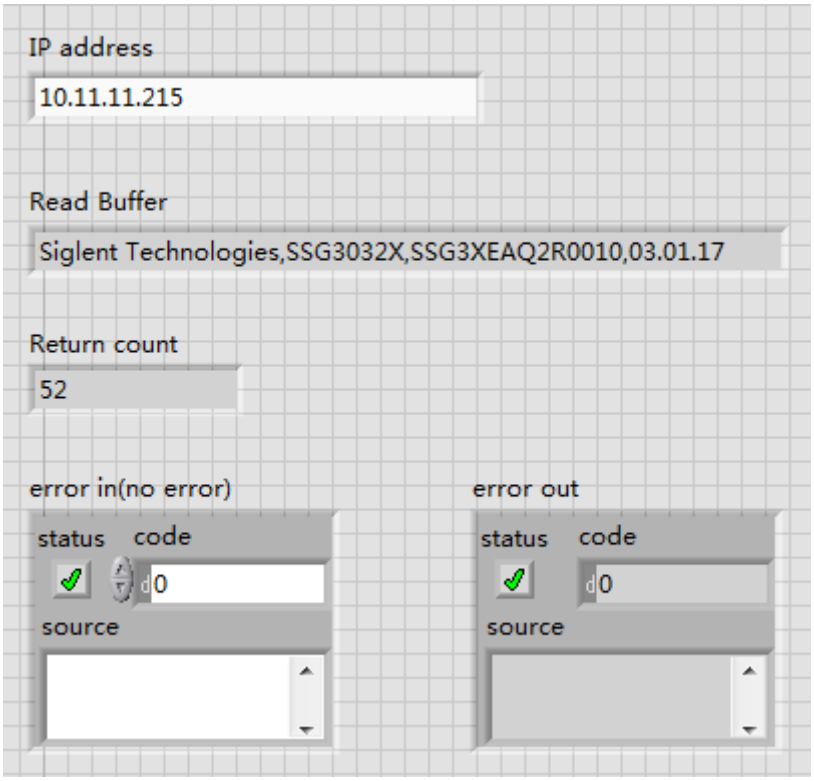

In this example, the VI opens a VISA session to a USBTMC device, writes a command to the device, and reads back the response. In this example, the specific command being sent is the device ID query. Check with your device manufacturer for the device command set. After all communication is complete, the VI closes the VISA session.

6、Communicating with the device via TCP/IP is similar to USBTMC. But you need to change VISA Write and VISA Read Function to Synchronous I/O. The LabVIEW default is asynchronous I/O. Right-click the node and select Synchronous I/O Mod>>Synchronous from the shortcut menu to write or read data synchronously.

7、Connect them as shown in the figure below




8、Input the IP address and run the program.




# 4.2Examples of Using Socket

## 4.2.1Example of Python

Python is an interpreted programming language that lets you work quickly and is very portable.Python has a low-level networking module that provides access to the socket interface.Python scripts can be written for sockets to do a variety of test and measurements tasks.

**Environment:** Win7 32bit system, Python v2.7.5

**The functions of this example:** Open a socket, sends a query, and closes the socket. It does this loop 10 times.

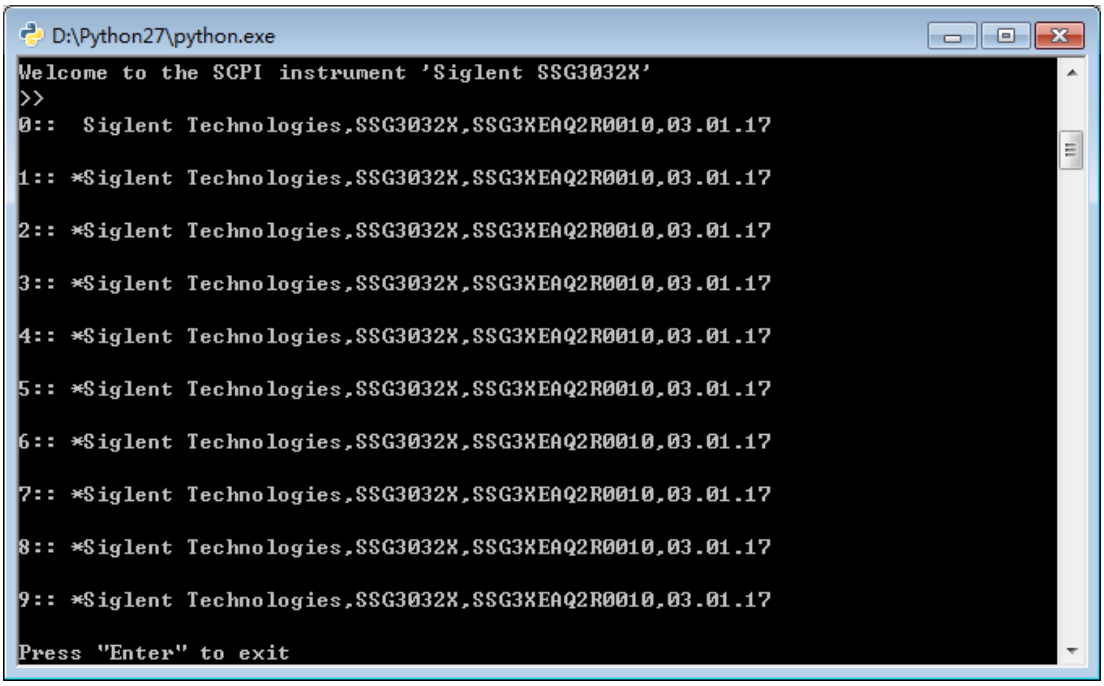
Below is the code of the script:

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-
#-----------------------------------------------------------------------------
# The short script is a example that open a socket, sends a query,
# print the return message and closes the socket.
```

```
#--------------------------------------------------------------------------
import socket   # for sockets
import sys  # for exit
import time # for sleep
#--------------------------------------------------------------------------
remote_ip = "10.11.13.32"  # should match the instrument's IP address
port = 5024 # the port number of the instrument service
count = 0

def SocketConnect():
    try:
        #create an AF_INET, STREAM socket (TCP)
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    except socket.error:
        print ('Failed to create socket.')
        sys.exit();
    try:
        #Connect to remote server
        s.connect((remote_ip , port))
        info = s.recv(4096)
        print (info)
    except socket.error:
        print ('failed to connect to ip ' + remote_ip)
    return s

def SocketQuery(Sock, cmd):
    try :
        #Send cmd string
        Sock.sendall(cmd)
        time.sleep(1)
    except socket.error:
        #Send failed
        print ('Send failed')
        sys.exit()
    reply = Sock.recv(4096)
    return reply

def SocketClose(Sock):
    #close the socket
    Sock.close()
    time.sleep(.300)

def main():
```

```
    global remote_ip
    global port
    global count

    # Body: send the SCPI commands *IDN? 10 times and print the return message
    s = SocketConnect()
    for i in range(10):
        qStr = SocketQuery(s, b'*IDN?')
        print (str(count) + ":: " + str(qStr))
        count = count + 1
    SocketClose(s)
    input('Press "Enter" to exit')

if __name__ == '__main__':
    proc = main()
```

**Run result:**




## 4.2.2Example of Telnet

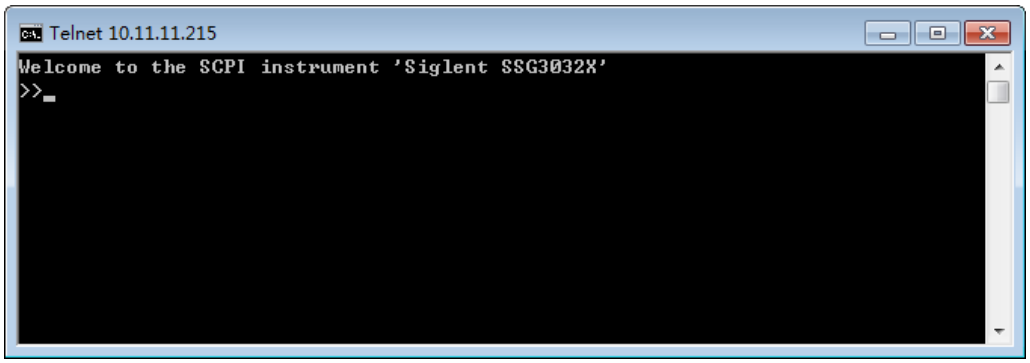
**Telnet SCPI:** Provides the ability to send single SCPI commands from a remote PC to the signal generator using LAN port number 5024.

How to send single SCPI commands using Telnet:

1. On the remote PC, click Start, then Run

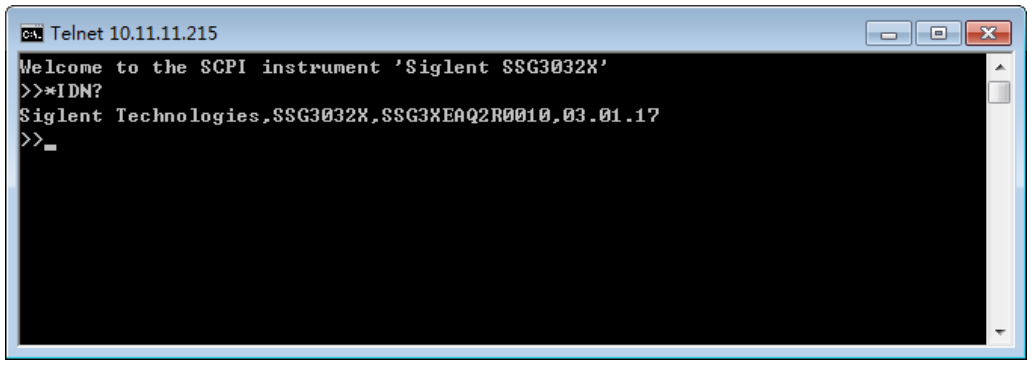
2. Type: **telnet** <ip address> **5024**

3. A Telnet window with a **>>** prompt should appear on the remote PC screen.



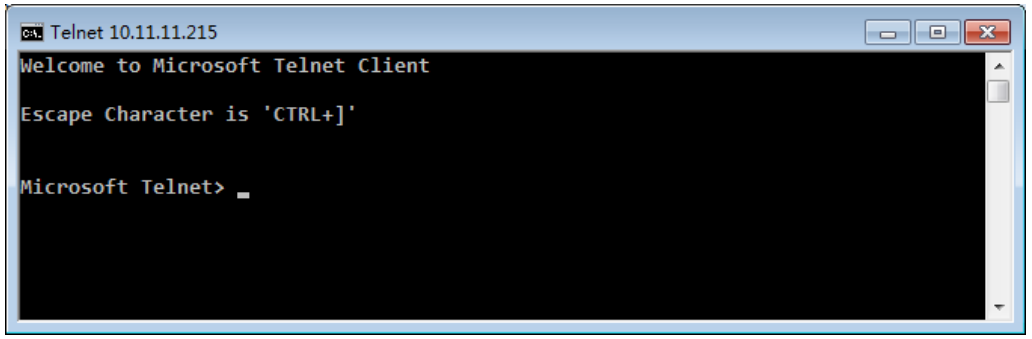

4. From the SCPI prompt:

- Type single SCPI commands. Press Enter to send the command.

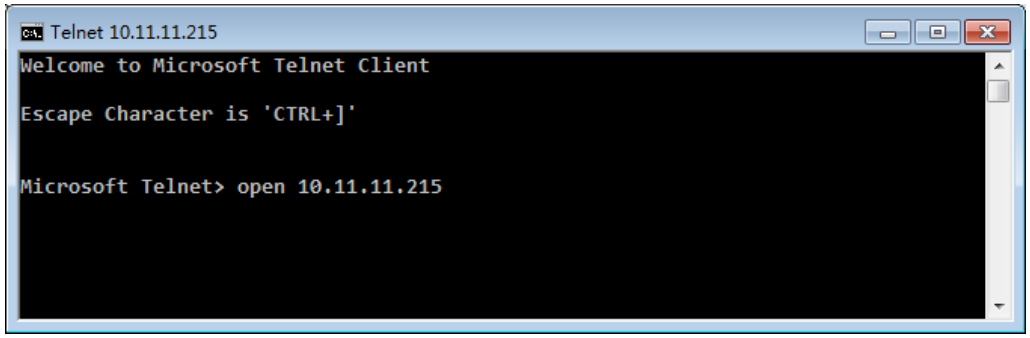



- To exit the telnet window click X in the upper-right corner.

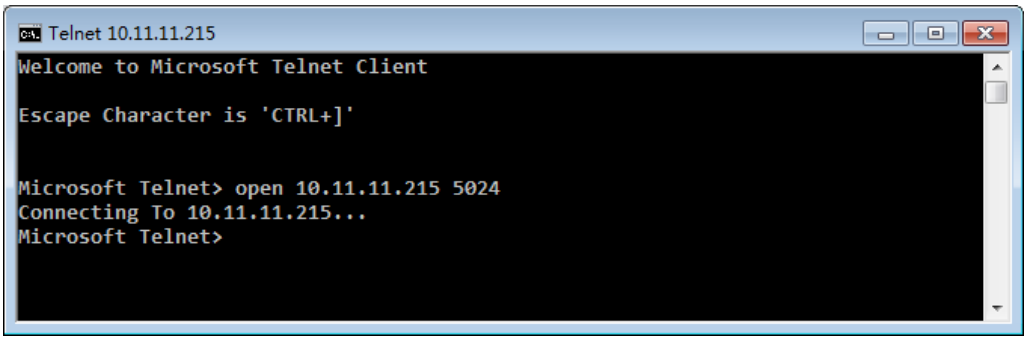
- To get a normal telnet prompt, press Ctrl + ] (closing bracket).




- To get SCPI prompt again, type open <ip Address> 5024.




Press **Enter**:




- To close the normal telnet window, type **Quit** and press **Enter**.