SSG6000A Series

Microwave Signal Generator

Programming Guide

EN01A

SIGLENT TECHNOLOGIES CO.,LTD

Contents

1 Programming Overview

The SSG6000A Series Signal Generator supports USB, LAN and USB-GPIB interfaces. Through these interfaces, combined with NI-VISA and corresponding programming language, users can use the command set based on SCPI (Standard Commands for Programmable Instruments) to remotely program and control the instrument, and interact with other programmable instruments that support the SCPI command set.

Through the LAN interface, VXI-11, Sockets, and Telnet protocols can be used to communicate with the instrument.

This chapter describes how to establish communication between the signal generator and the computer, and how to remotely control the signal generator.

1.1 Build communication via VISA

1.1.1 Install NI-VISA

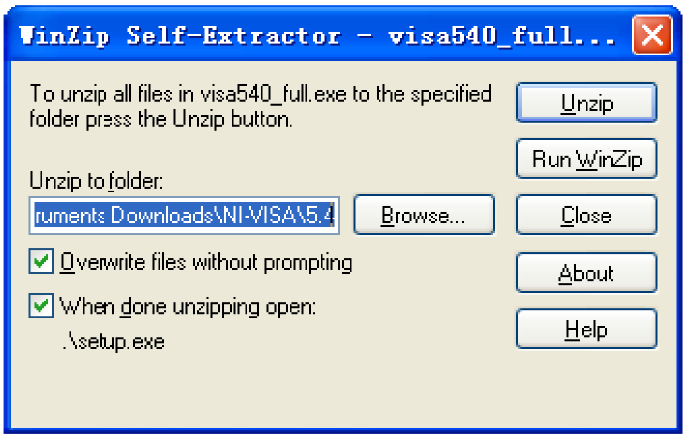
Before programming, please make sure that you have properly installed the latest version of National Instruments NI-VISA Software.

NI-VISA is a communication library for communication between computers and devices. NI software has two valid VISA installation packages: full version and run-time engine version (Run-Time Engine). The runtime engine version provides NI device drivers such as USB-TMC, VXI and GPIB, etc. It is mainly used for remote control. The full version includes the runtime engine and NI MAX tools, where NI MAX is the user interface for controlling the device.

You can download the latest NI-VISA runtime engine or full version from the NI official website. Their installation steps are basically the same.

Please refer to the following steps to install NI-VISA (the example uses the full version of NI-VISA5.4):

a. Download the appropriate version of NI-VISA.
b. Double-click visa540_full.exe, and the dialog box will pop up as follows:


WinZip Self-Extractor - visa540_full...

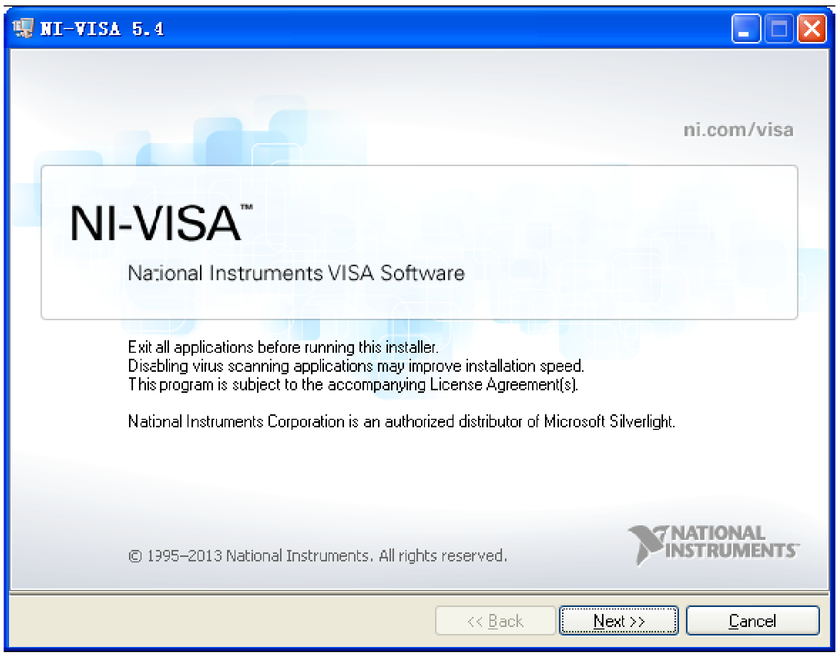
To unzip all files in visa540_full.exe to the specified folder press the Unzip button.

Unzip to folder:
ruments Downloads\NI-VISA\5.4

☑ Overwrite files without prompting

☑ When done unzipping open:
.\setup.exe

c. Click Unzip. Then the installation process will launch automatically after unzipping files. If your computer needs to install the .NET Framework 4, it shall auto-start.


NI-VISA 5.4

ni.com/visa

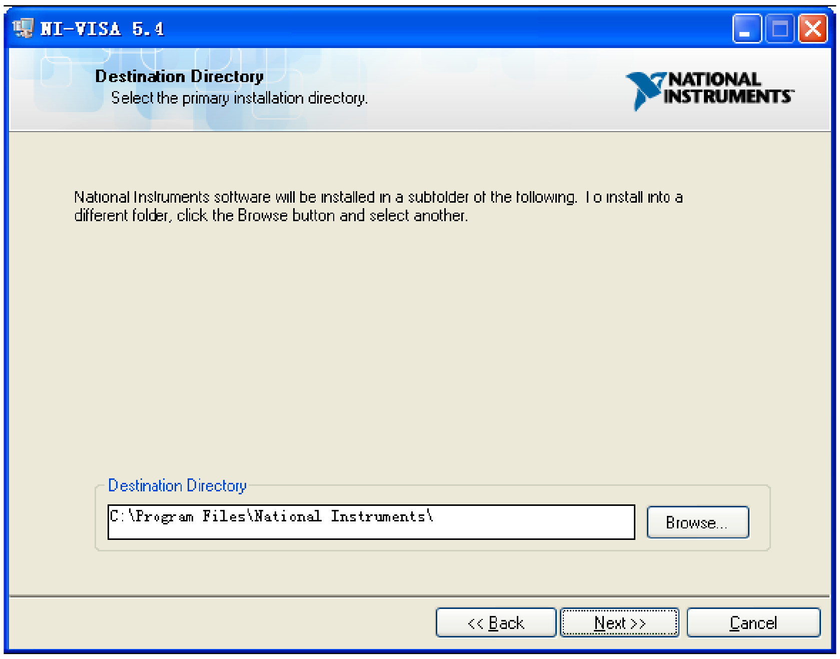
NI-VISA™
National Instruments VISA Software

Exit all applications before running this installer.
Disabling virus scanning applications may improve installation speed.
This program is subject to the accompanying License Agreement(s).

National Instruments Corporation is an authorized distributor of Microsoft Silverlight.

© 1995–2013 National Instruments. All rights reserved.

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

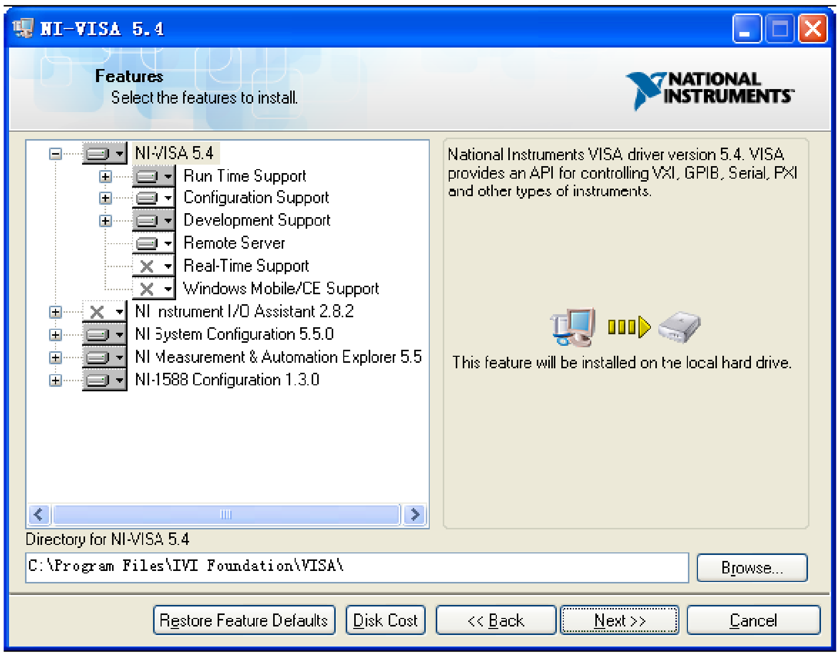


Destination Directory
Select the primary installation directory.

National Instruments software will be installed in a subfolder of the following. To install into a different folder, click the Browse button and select another.

Destination Directory
C:\Program Files\National Instruments\

e. Set the installation path. The default path is “C:\Program Files\National Instruments\”. You can also modify the installation path. Click Next and the dialog box will appear as shown below.



Features
Select the features to install.

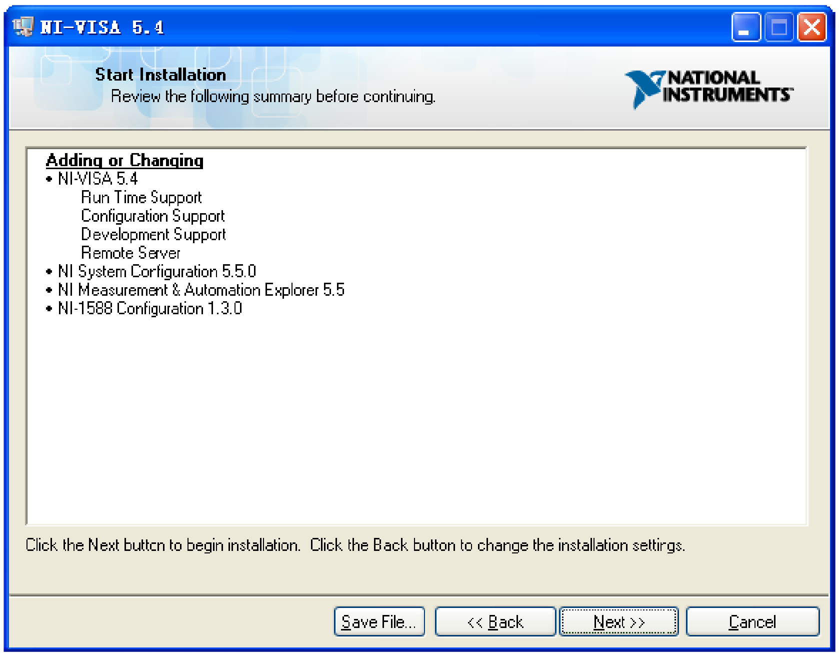
- NI-VISA 5.4
  - Run Time Support
  - Configuration Support
  - Development Support
  - Remote Server
  - Real-Time Support
  - Windows Mobile/CE Support
- NI Instrument I/O Assistant 2.8.2
- NI System Configuration 5.5.0
- NI Measurement & Automation Explorer 5.5
- NI-1588 Configuration 1.3.0

National Instruments VISA driver version 5.4. VISA provides an API for controlling VXI, GPIB, Serial, PXI and other types of instruments.

This feature will be installed on the local hard drive.

Directory for NI-VISA 5.4
C:\Program Files\IVI Foundation\VISA\

f. Click Next twice. In the License Agreement dialog, select “I accept the above 2 License Agreement(s).” and click Next. The dialog box will appear as shown below:

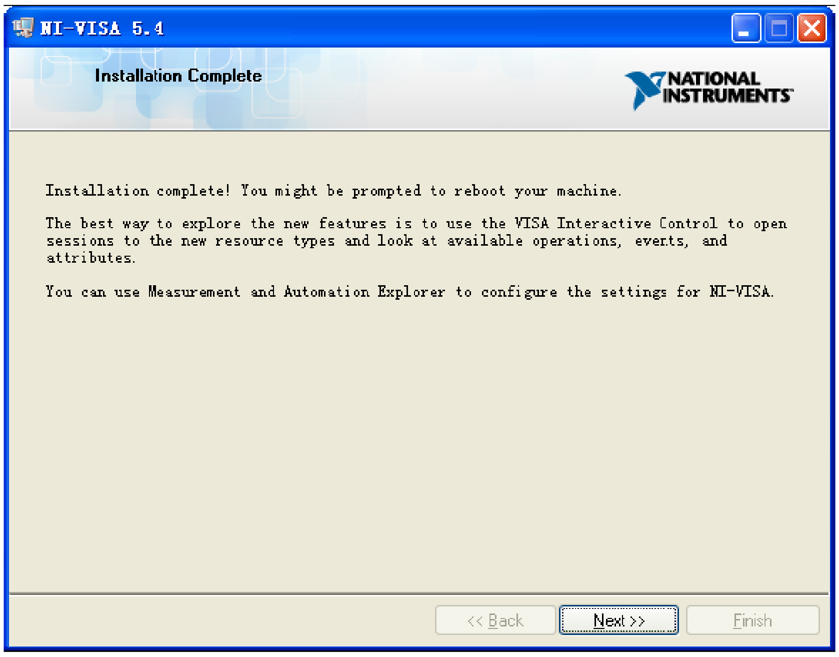


Start Installation
Review the following summary before continuing.

Adding or Changing
• NI-VISA 5.4
Run Time Support
Configuration Support
Development Support
Remote Server
• NI System Configuration 5.5.0
• NI Measurement & Automation Explorer 5.5
• NI-1588 Configuration 1.3.0

Click the Next button to begin installation. Click the Back button to change the installation settings.

g. Click Next to begin installation.



Installation Complete

Installation complete! You might be prompted to reboot your machine.

The best way to explore the new features is to use the VISA Interactive Control to open sessions to the new resource types and look at available operations, events, and attributes.

You can use Measurement and Automation Explorer to configure the settings for NI-VISA.

h. Now the installation is complete. Reboot your computer.

1.1.2 Connect the instrument to computer

The signal generator may be able to communicate with the computer through the USB, LAN or USB-GPIB interface.

1.1.2.1 Connect using USB interface

Please refer to the following steps to finish the connection via the USB interface:

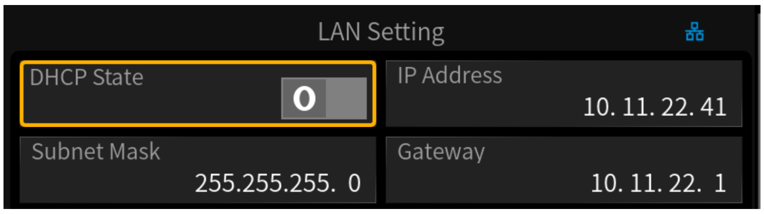
1. Install NI-VISA on your computer to obtain the USB-TMC driver.
2. Connect the USB Device interface of the signal generator to the USB Host interface of the computer with a USB A-B cable.
3. Turn on the signal generator.

The signal generator will be automatically detected as a new USB device.

1.1.2.2 Connect using LAN interface

Please refer to the following steps to finish the connection via the LAN interface:

1. Install NI-VISA on your computer to obtain the VXI driver.
2. Use a network cable to connect the signal generator to your computer or the local area network.
3. Turn on the signal generator.
4. Press UTILITY → Interface to enter the LAN Setting menu.
5. Set the LAN as Static or DHCP:
   - ◆ DHCP: The DHCP server in the current network will automatically assign network parameters (IP address, subnet mask, gateway) to the signal generator.
   - ◆ Static: You can manually set the IP address, subnet mask, and gateway.


LAN Setting

DHCP State O

IP Address 10. 11. 22. 41

Subnet Mask 255.255.255. 0

Gateway 10. 11. 22. 1

The signal generator will be automatically or manually detected as a new LAN device.

1.1.2.3 Connect using USB Host interface (With USB-GPIB Adaptor)

Please refer to the following steps to finish the connection via the LAN interface:

1. Install the NI-VISA GPIB driver on the computer.
2. Use the SIGLENT USB-GPIB adapter to connect the USB Host interface of the signal generator to the GPIB interface of the computer.

3. Turn on the signal generator.
4. Press $\boxed{\text{UTILITY}}$ → Interface and enter the GPIB number in GPIB Address.

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

1.2 Remote Control

1.2.1 User-defined Programming

Users can send SCPI commands through the computer to program and control the signal generator. For details, please refer to the introduction in the “Programming Examples” chapter.

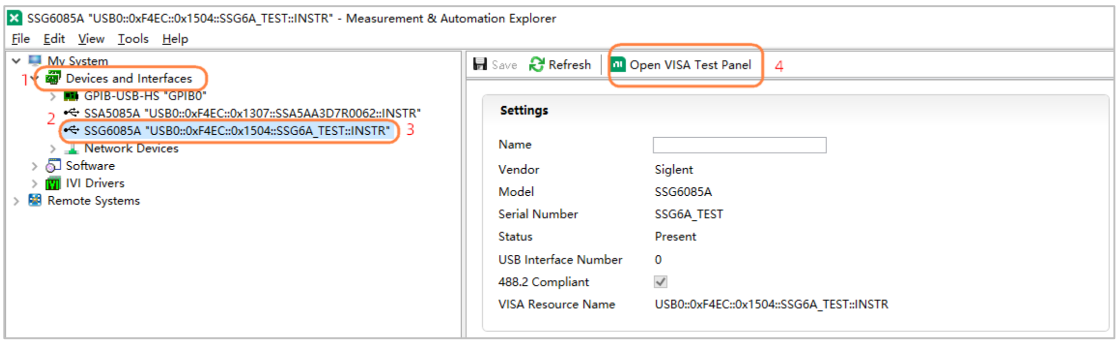
1.2.2 Send SCPI via NI-MAX

NI-MAX is a program created and maintained by National Instruments. It provides basic remote control interfaces for VXI, LAN, USB, GPIB, and Serial communications. Users can send SCPI commands through NI-MAX to remotely control the signal generator.

The following describes the steps for connecting a device through a USB, LAN, or GPIB interface in NI-MAX and sending SCPI to the device.

1.2.2.1 Using USB

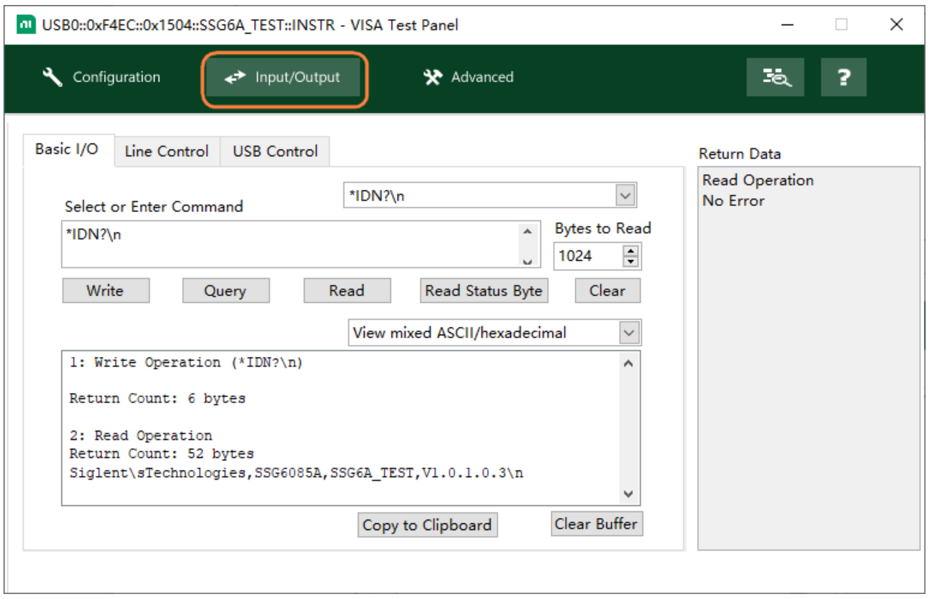
1. Run NI-MAX.
2. Click “Devices and Interfaces” in the upper left corner of the software.
3. Find the USBTMC device symbol: ⭠ .
4. Select the signal generator device and click the “Open VISA Test Panel” button.


SSG6085A "USB0::0xF4EC::0x1504::SSG6A_TEST::INSTR" - Measurement & Automation Explorer
File Edit View Tools Help
My System
1
Devices and Interfaces
GPIB-USB-HS "GPIB0"
SSA5085A "USB0::0xF4EC::0x1307::SSA5AA3D7R0062::INSTR"
2
SSG6085A "USB0::0xF4EC::0x1504::SSG6A_TEST::INSTR"
3
Network Devices
Software
IVI Drivers
Remote Systems
Save
Refresh
Open VISA Test Panel
4

Settings

Name	
Vendor	Siglent
Model	SSG6085A
Serial Number	SSG6A_TEST
Status	Present
USB Interface Number	0
488.2 Compliant	✓
VISA Resource Name	USB0::0xF4EC::0x1504::SSG6A_TEST::INSTR

5. Select the “Input/Output” page in the VISA test panel. At this time, you can enter SCPI in the input field to write or query. Click the “Query” button as shown below to query the device’s IDN:



Select or Enter Command: *IDN?\n

Bytes to Read: 1024

View mixed ASCII/hexadecimal

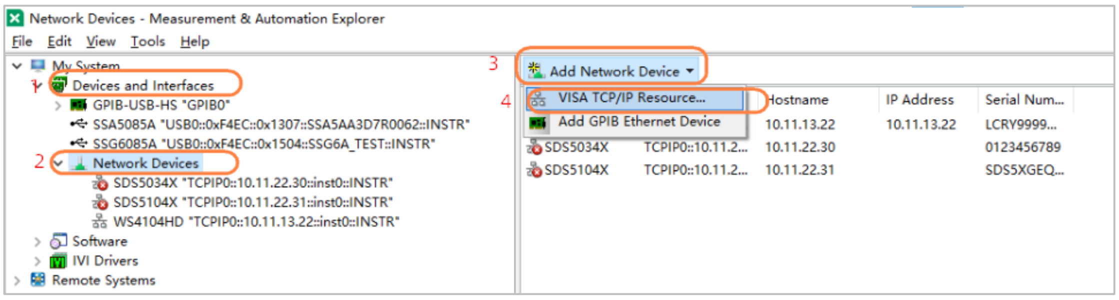
```
1: Write Operation (*IDN?\n)

Return Count: 6 bytes

2: Read Operation
Return Count: 52 bytes
Siglent\sTechnologies,SSG6085A,SSG6A_TEST,V1.0.1.0.3\n
```

Return Data

```
Read Operation
No Error
```

1.2.2.2 Using LAN

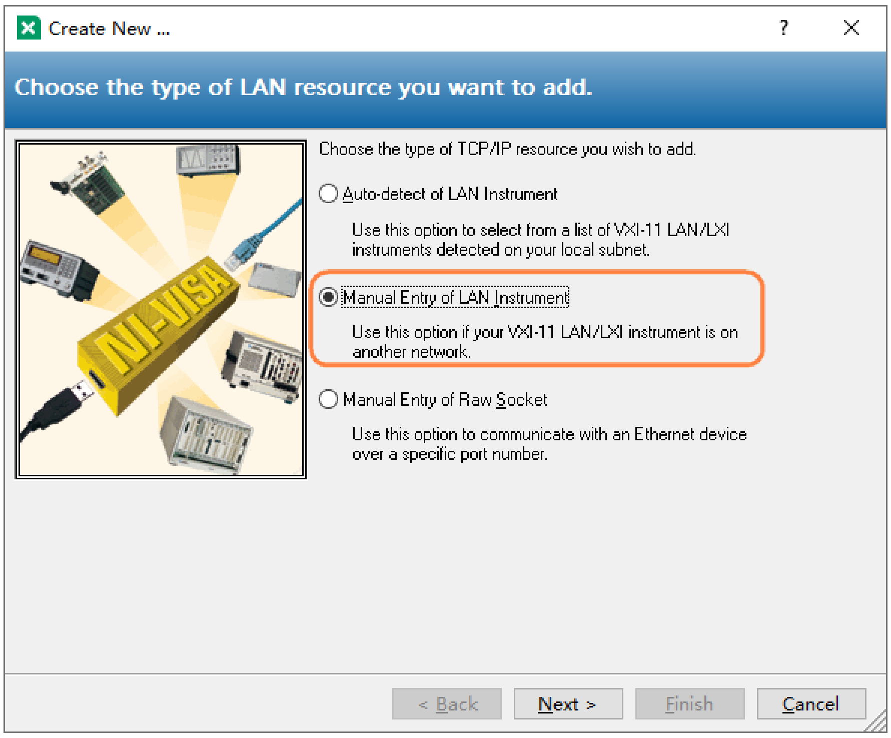
1. Run NI-MAX.
2. Click “Devices and Interfaces” > “Network Devices” in the upper left corner of the software.
3. Click “Add Network Device” > “VISA TCP/IP Resource…”.


Network Devices - Measurement & Automation Explorer
File Edit View Tools Help
3
Add Network Device
4
VISA TCP/IP Resource...
Add GPIB Ethernet Device
2
1

```
My System
  Devices and Interfaces
    GPIB-USB-HS "GPIB0"
    SSA5085A "USB0::0xF4EC::0x1307::SSA5AA3D7R0062::INSTR"
    SSG6085A "USB0::0xF4EC::0x1504::SSG6A_TEST::INSTR"
    Network Devices
      SDS5034X "TCPIP0::10.11.22.30::inst0::INSTR"
      SDS5104X "TCPIP0::10.11.22.31::inst0::INSTR"
      WS4104HD "TCPIP0::10.11.13.22::inst0::INSTR"
  Software
  IVI Drivers
Remote Systems
```

		Hostname	IP Address	Serial Num...
		10.11.13.22	10.11.13.22	LCRY9999...
SDS5034X	TCPIP0::10.11.2...	10.11.22.30		0123456789
SDS5104X	TCPIP0::10.11.2...	10.11.22.31		SDS5XGEQ...

4. In the pop-up “Create New...” window, select "Manual Entry of LAN Instrument" and then click Next.


Create New ...

Choose the type of LAN resource you want to add.

Choose the type of TCP/IP resource you wish to add.

○ Auto-detect of LAN Instrument

Use this option to select from a list of VXI-11 LAN/LXI instruments detected on your local subnet.

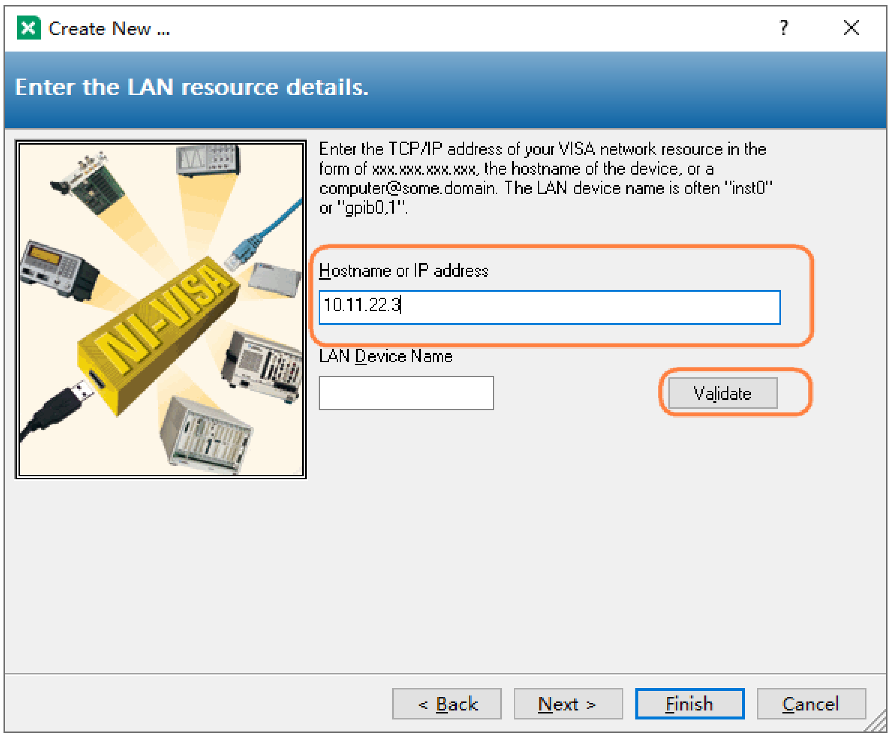
● Manual Entry of LAN Instrument

Use this option if your VXI-11 LAN/LXI instrument is on another network.

○ Manual Entry of Raw Socket

Use this option to communicate with an Ethernet device over a specific port number.

5. Enter the IP address of the signal generator in "Hostname or IP address". You can click "Validate" to verify whether the device can be connected via the entered IP.


Create New ...

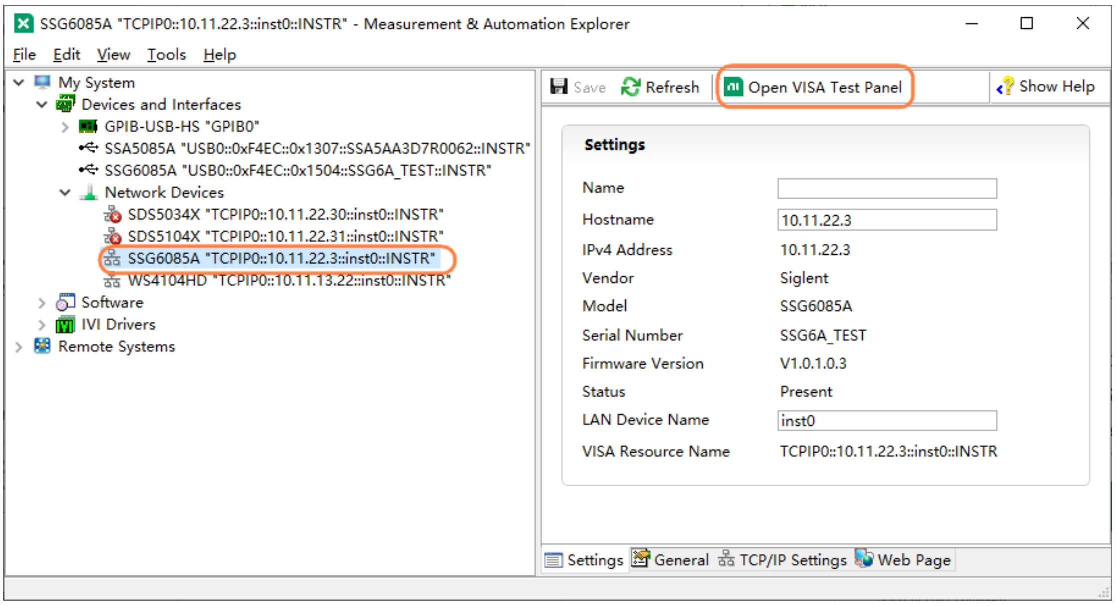
Enter the LAN resource details.

Enter the TCP/IP address of your VISA network resource in the form of xxx.xxx.xxx.xxx, the hostname of the device, or a computer@some.domain. The LAN device name is often "inst0" or "gpib0,1".

Hostname or IP address: 10.11.22.3

LAN Device Name:

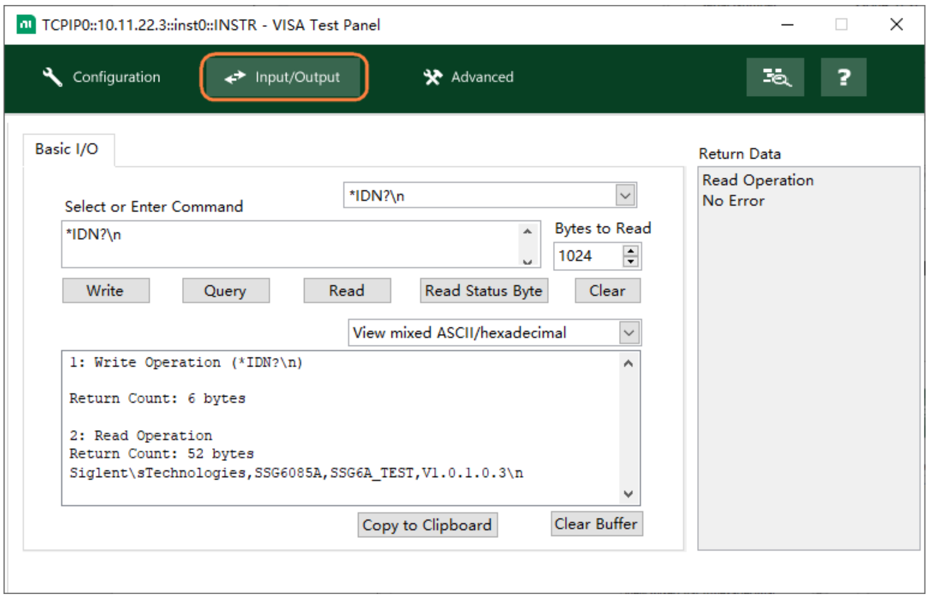
6. Click "Finish" to establish the connection.
7. After a short scan, the resource name of the signal generator should be displayed under "Network Devices".


SSG6085A "TCPIP0::10.11.22.3::inst0::INSTR" - Measurement & Automation Explorer
File Edit View Tools Help
Save
Refresh
Open VISA Test Panel
Show Help
My System
Devices and Interfaces
GPIB-USB-HS "GPIB0"
SSA5085A "USB0::0xF4EC::0x1307::SSA5AA3D7R0062::INSTR"
SSG6085A "USB0::0xF4EC::0x1504::SSG6A_TEST::INSTR"
Network Devices
SDS5034X "TCPIP0::10.11.22.30::inst0::INSTR"
SDS5104X "TCPIP0::10.11.22.31::inst0::INSTR"
SSG6085A "TCPIP0::10.11.22.3::inst0::INSTR"
WS4104HD "TCPIP0::10.11.13.22::inst0::INSTR"
Software
IVI Drivers
Remote Systems
Settings
General
TCP/IP Settings
Web Page

Settings

Name	
Hostname	10.11.22.3
IPv4 Address	10.11.22.3
Vendor	Siglent
Model	SSG6085A
Serial Number	SSG6A_TEST
Firmware Version	V1.0.1.0.3
Status	Present
LAN Device Name	inst0
VISA Resource Name	TCPIP0::10.11.22.3::inst0::INSTR

8. Select the signal generator device and click the “Open VISA Test Panel” button.
9. Select the “Input/Output” page in the VISA test panel. At this time, you can enter SCPI in the input field to write or query. Click the “Query” button as shown below to query the device’s IDN:



Select or Enter Command: *IDN?\n

*IDN?\n

Bytes to Read: 1024

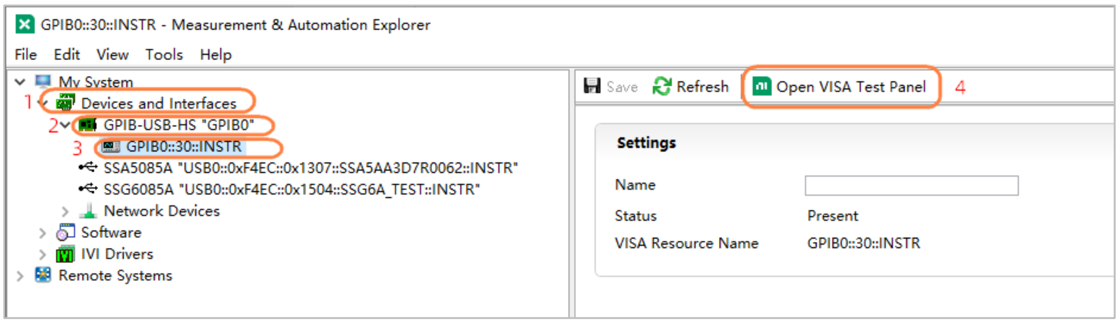
```
1: Write Operation (*IDN?\n)

Return Count: 6 bytes

2: Read Operation
Return Count: 52 bytes
Siglent\sTechnologies,SSG6085A,SSG6A_TEST,V1.0.1.0.3\n
```

Return Data

Read Operation
No Error

1.2.2.3 Using USB-GPIB

1. Run NI-MAX.
2. Click “Devices and Interfaces” in the upper left corner of the software.
3. Find the “GPIB-USB-HS” device symbol.
4. Select the signal generator device and click the “Open VISA Test Panel” button.

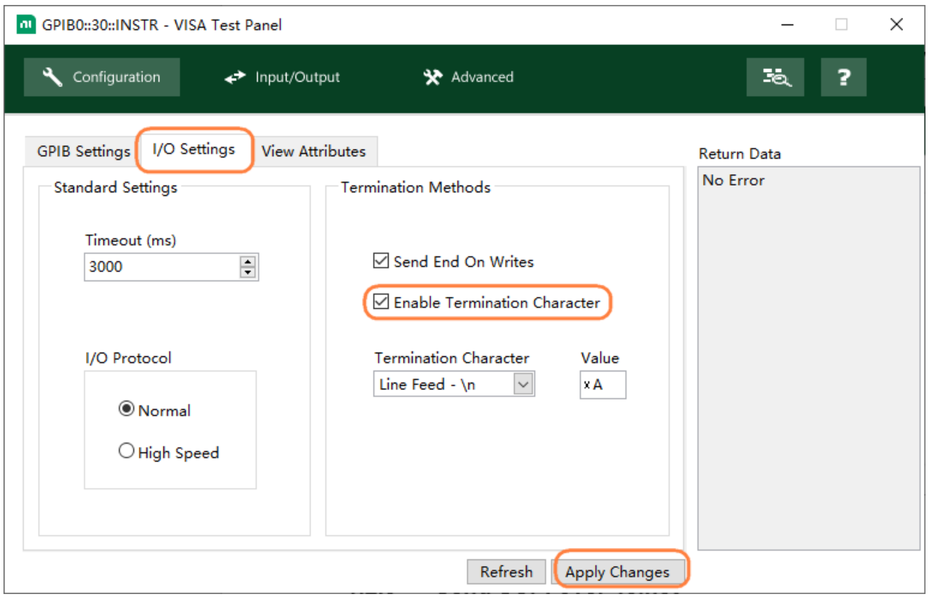

GPIB0::30::INSTR - Measurement & Automation Explorer
File Edit View Tools Help
My System
1 Devices and Interfaces
2 GPIB-USB-HS "GPIB0"
3 GPIB0::30::INSTR
SSA5085A "USB0::0xF4EC::0x1307::SSA5AA3D7R0062::INSTR"
SSG6085A "USB0::0xF4EC::0x1504::SSG6A_TEST::INSTR"
Network Devices
Software
IVI Drivers
Remote Systems
Save
Refresh
Open VISA Test Panel
4
Settings

Name

Status: Present

VISA Resource Name: GPIB0::30::INSTR

5. Select the “Configuration” > “I/O Settings” in the VISA test panel. Check the Enable Termination Character option, and click Apply Changes button.



Timeout (ms): 3000

I/O Protocol: [✓ Normal] High Speed

☑ Send End On Writes

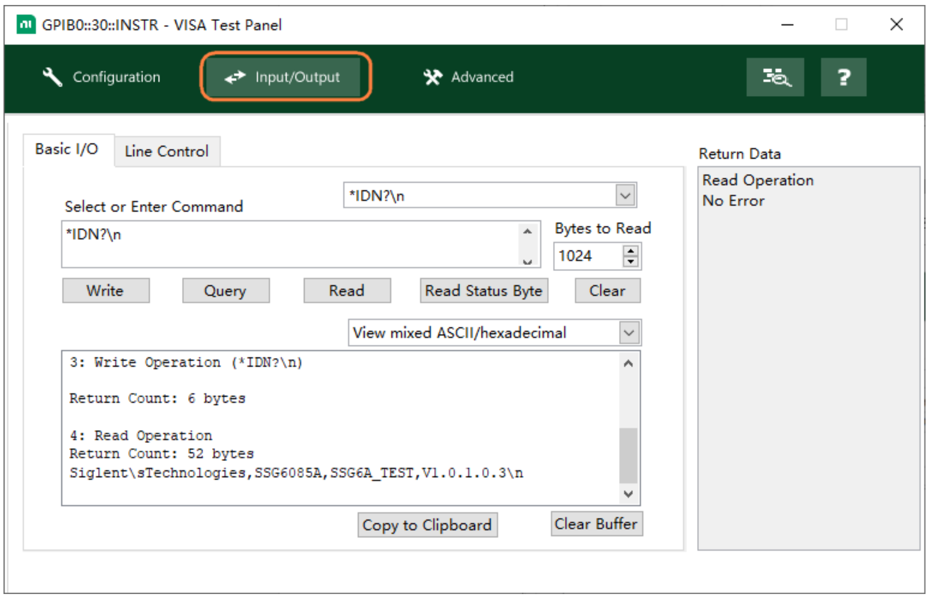
☑ Enable Termination Character

Termination Character: Line Feed - \n

Value: xA

No Error

6. Select the “Input/Output” page in the VISA test panel. At this time, you can enter SCPI in the input field to write or query. Click the “Query” button as shown below to query the device’s IDN:



Select or Enter Command: *IDN?\n

*IDN?\n

Bytes to Read: 1024

View mixed ASCII/hexadecimal

```
3: Write Operation (*IDN?\n)

Return Count: 6 bytes

4: Read Operation
Return Count: 52 bytes
Siglent\sTechnologies,SSG6085A,SSG6A_TEST,V1.0.1.0.3\n
```

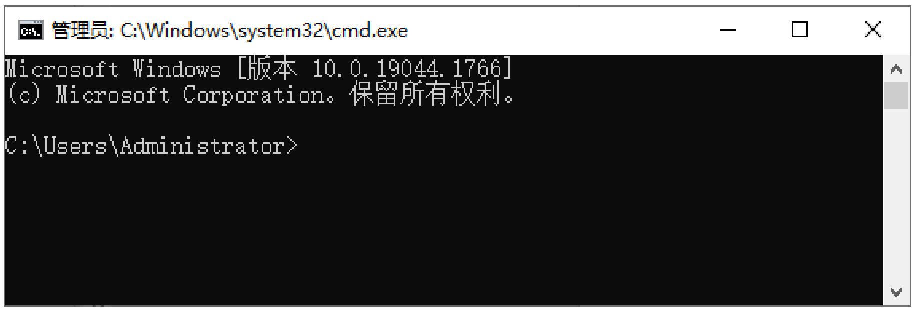
Read Operation
No Error

1.2.3 Send SCPI over Telnet

Telnet provides a way to communicate with the signal generator through the LAN interface. The Telnet protocol supports sending SCPI commands from the computer to the signal generator in a manner similar to communicating with the signal generator via USB. Sending and receiving information is interactive, and only one command can be sent at a time. Windows operating systems use a command prompt style interface as a Telnet client.

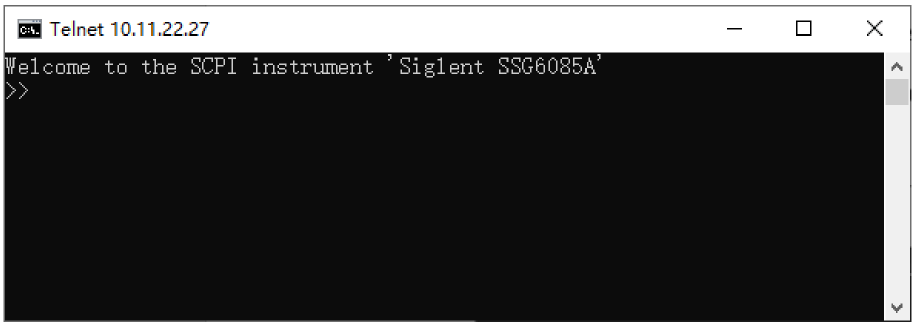
The steps are as follows:

1. On the computer desktop, click Start, then right-click and select Run. Enter *cmd* in the run window and click OK. The command prompt window opens.



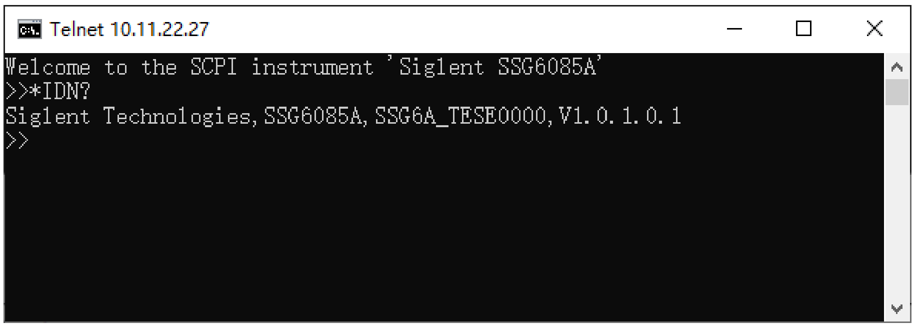
```
管理员: C:\Windows\system32\cmd.exe
Microsoft Windows [版本 10.0.19044.1766]
(c) Microsoft Corporation。保留所有权利。

C:\Users\Administrator>
```

2. In the command prompt window, enter *telnet <ip address> 5024* and press Enter. A Telnet window that can communicate with the instrument will pop up:



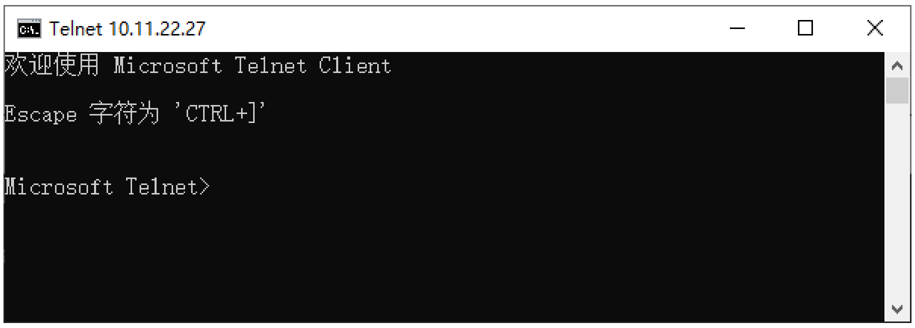
```
Telnet 10.11.22.27
Welcome to the SCPI instrument 'Siglent SSG6085A'
>>
```

3. After the “>>” prompt, you can enter a SCPI command to remotely control the signal generator. For example, enter **IDN?*, this command will return the company name, machine model, serial number and firmware version number.



```
Telnet 10.11.22.27
Welcome to the SCPI instrument 'Siglent SSG6085A'
>>*IDN?
Siglent Technologies,SSG6085A,SSG6A_TESE0000,V1.0.1.0.1
>>
```

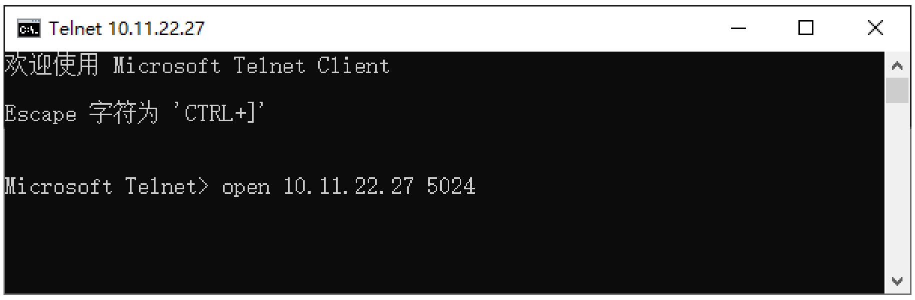
4. Press Ctrl+] keys simultaneously to exit the SCPI session with the instrument.



Telnet 10.11.22.27

```
欢迎使用 Microsoft Telnet Client

Escape 字符为 'CTRL+]'

Microsoft Telnet>
```

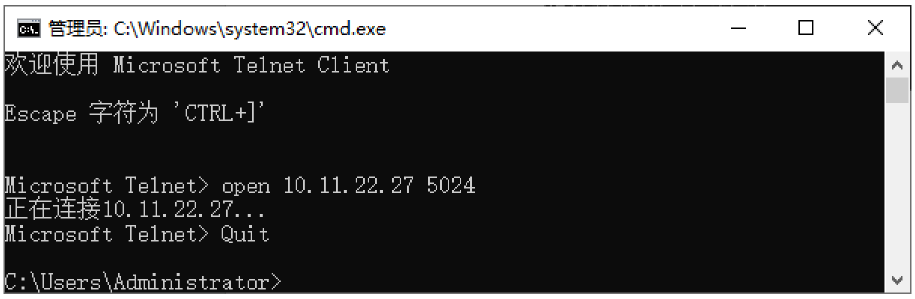
5. To re-enter the SCPI session with the instrument, you can enter *open <ip Address> 5024* and press Enter.



Telnet 10.11.22.27

```
欢迎使用 Microsoft Telnet Client

Escape 字符为 'CTRL+]'

Microsoft Telnet> open 10.11.22.27 5024
```

6. To close the Telnet window, type *Quit* and press Enter.



管理员: C:\Windows\system32\cmd.exe

```
欢迎使用 Microsoft Telnet Client

Escape 字符为 'CTRL+]'

Microsoft Telnet> open 10.11.22.27 5024
正在连接10.11.22.27...
Microsoft Telnet> Quit

C:\Users\Administrator>
```

1.2.4 Send SCPI over Socket

Socket API can be used to control the signal generator through the LAN interface without installing any other libraries, which can reduce the complexity of programming.
For detailed information, please refer to the “Socket Examples” chapter of “Programming Examples”.

SOCKET ADDRESS	IP address + port number
IP ADDRESS	SSG IP address
PORT NUMBER	5025

2 Introduction to the SCPI Language

2.1 Command Format

SCPI commands are tree-like hierarchical structures, including multiple subsystems. Each subsystem consists of a root keyword and one or several hierarchical keywords. The command line usually starts with a colon ":" and keywords are separated by a colon ":". The keyword is followed by optional parameter settings. The command and parameters are separated by "space". For multiple parameters, the parameters are separated by commas ",". Add a question mark "?" after the command line to indicate querying this function.

For example:
:SOURce:FREQuency <freq>
:SOURce:FREQuency?

SOURce is the root keyword of the command, and FREQuency is the second-level keyword. The command line starts with a colon ":", and colons separate keywords at each level. <freq> represents a settable parameter. The command: SOURce:FREQuency and the parameter <freq> are separated by a "space". The question mark "?" indicates query, and the instrument will return a response string after receiving the query command.

2.2 Symbol Instruction

The following are the symbols used in the SCPI commands:

1. Angular brackets < >

The contents in angular brackets < > are command parameters and must be replaced with a valid value. For example:
POWer:SPC:TARGet <power> command, you can send as POWer:SPC:TARGet 0.

2. Square brackets [ ]

Contents in square brackets (command keywords or default parameters) can be omitted. If you omit the keyword in square brackets, the command still produces the same effect. If a default parameter in square brackets is omitted, the default parameter still takes effect.

3. Vertical lines |

Vertical bars are used to separate multiple enumeration values, one of which must be selected when sending a command. For example:

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

4. Braces { }

Parameters in braces are optional and may not be set, or may be set once or multiple times. For example:

In the :CALCulate:LLINe[1]|2:DATA <x-axis>,<ampl>{,<x-axis>,<ampl>} command,

{,<x-axis>,<ampl>} in braces can be omitted, or one or more pairs of frequency and amplitude parameters can be set.

2.3 Parameter Type

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

1. Boolean

The parameter in the command could be “OFF”, “ON”, “0” or “1”.

For example:

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

2. Enumeration

Parameter should be one of the listed values.

For example:

In [:SOURce]:SWEep:STATe OFF|FREQuency|LEVel|LEV_FREQ command, valid parameters are “OFF”, “FREQuency”, “LEVel” or “LEV_FREQ”.

3. Integer

Unless otherwise stated, the parameter can be any integer within the valid value range.

For example:

In [:SOURce]:SWEep:STEP:POINts <value> command, the parameter <value> can be set to any integer between 2 and 65535.

4. Float

Parameters can take any value within the valid range according to accuracy requirements (usually the default accuracy is nine decimal places).

For example:
In [:SOURce]:POWer:OFFSet <value> command, the parameter <value> can be set to any real number between -100 and 100.

5. String

The parameter should be the combination of ASCII characters.

For example:
In :SYSTem:COMMunicate:LAN:IPADdress <“xxx.xxx.xxx.xxx”> command, the IP address can be set as the string “192.168.1.12”.

6. Discrete

The parameter could only be one of the specified values and these values are discontinuous.

For example:
In [:SENSe]:BWIDth:VIDeo:RATio <number> command, the parameter <number> could only be one of 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1.0, 3.0, 10.0, 30.0, 100.0, 300.0, 1000.0.

2.4 Command Abbreviation

All SCPI commands are case-insensitive. You can enter the complete command in all uppercase or all lowercase. You can also use abbreviations, in which case the abbreviated command must contain all uppercase letters in the command format.
For example:
:CORRection:FLATness:COUNt?

You can send in any of the following writing methods:
:CORRection:FLATness:COUNt?
:CORRECTION:FLATNESS:COUNT?
:correction:flatness:count?

You can also abbreviate it to:
:CORR:FLAT:COUN?

3 Commands

3.1 IEEE 488.2 Common Commands

The IEEE standards defined the common commands used for querying the basic information of the instrument and performing basic operations. These commands usually start with "*" and the length of the command keyword is usually 3 characters.

3.1.1 Identification Query (*IDN?)

SYNTAX	*IDN?
DESCRIPTION	This command reads the product information (manufacturer, device model, serial number, and firmware revision number) of the device.
EQUIVALENT MENU	None
EXAMPLE	**IDN?* Return: *Siglent Technologies,SSG6085A,SSG6A_TESE0000,V1.0.1.0.1\n*

3.1.2 Reset (*RST)

SYNTAX	*RST
DESCRIPTION	Restore the device state to its initial state.
EQUIVALENT MENU	None
EXAMPLE	**RST*

3.1.3 Clear Status (*CLS)

SYNTAX	*CLS
DESCRIPTION	Clear the values of all status event registers and clear the error queue.
EQUIVALENT MENU	None
EXAMPLE	**CLS*

3.1.4 Standard Event Status Enable (*ESE)

SYNTAX	*ESE <number> *ESE?
DESCRIPTION	This command sets/gets the value of the Standard Event Status Enable Register.

DATA TYPE	Integer
RANGE	0 to 255
EQUIVALENT MENU	None
EXAMPLE	**ESE 16* **ESE?* Return: *16\n*

3.1.5 Standard Event Status Register Query (*ESR?)

SYNTAX	*ESR?
DESCRIPTION	This command reads the value of the Standard Event Status Register. Execution of this command clears the register value.
EQUIVALENT MENU	None
EXAMPLE	**ESR?* Return: *0\n*

3.1.6 Operation Complete Query (*OPC)

SYNTAX	*OPC *OPC?
DESCRIPTION	This command sets/gets 1 the OPC bit (bit 0) of the Standard Event Status Register when all of pending operations complete.
EQUIVALENT MENU	None
EXAMPLE	**OPC* **OPC?* Return: *1\n*

3.1.7 Service Request Enable (*SRE)

SYNTAX	*SRE <integer> *SRE?
DESCRIPTION	This command sets/gets the value of Service Request Enable Register.
DATA TYPE	Integer
RANGE	0 to 255
EQUIVALENT MENU	None
EXAMPLE	**SRE 24*

	**SRE?* Return: *24\n*

3.1.8 Status Byte Query (*STB?)

SYNTAX	*STB?
DESCRIPTION	This command reads the value of Status Byte Register.
EQUIVALENT MENU	None
EXAMPLE	**STB?* Return: *72\n*

3.1.9 Wait-to-Continue (*WAI)

SYNTAX	*WAI
DESCRIPTION	This command waits for the execution of all objects sent before this command to be completed.
EQUIVALENT MENU	None
EXAMPLE	**WAI*

3.1.10 Self Test Query (*TST?)

SYNTAX	*TST?
DESCRIPTION	This command queries the instrument self-test results..
EQUIVALENT MENU	None
EXAMPLE	**TST?* Return: *0\n*

3.2 SYSTem Commands

3.2.1 System Configuration

3.2.1.1 System Time (:SYSTem:TIME)

SYNTAX	:SYSTem:TIME <hhmmss> :SYSTem:TIME?
DESCRIPTION	This command sets/gets the system time.
DATA TYPE	String
RANGE	Hours(0 ~ 23), minutes(0 ~ 59), seconds(0 ~ 59)
RETURN	String
EQUIVALENT MENU	UTILITY > Setting > Time Setting
EXAMPLE	*:SYSTem:TIME 182559* *:SYSTem:TIME?* Return: *182613\n*

3.2.1.2 System Date (:SYSTem:DATE)

SYNTAX	:SYSTem:DATE <yyyymmdd> :SYSTem:DATE?
DESCRIPTION	This command sets/gets the system date.
DATA TYPE	String
RANGE	Years(four digits), month(1 ~ 12), date(1 ~ 31)
RETURN	String
EQUIVALENT MENU	UTILITY > Setting > Time Setting
EXAMPLE	*:SYSTem:DATE 20050101* *:SYSTem:DATE?* Return: *20050101\n*

3.2.1.3 IP Address (:SYSTem:COMMunicate:LAN:IPADdress)

SYNTAX	:SYSTem:COMMunicate:LAN:IPADdress <“xxx.xxx.xxx.xxx”> :SYSTem:COMMunicate:LAN:IPADdress?
DESCRIPTION	This command sets/gets the IP address of the device when the IP assignment is set to Static.
DATA TYPE	String
RANGE	Conforms to the IP address standard (0-255:0-255:0-255:0-255)
RETURN	String
EQUIVALENT MENU	UTILITY > Interface > LAN Setting > IP Address

EXAMPLE	*:SYSTem:COMMunicate:LAN:IPADdress "192.168.1.12"* *:SYSTem:COMMunicate:LAN:IPADdress?* Return: *"192.168.1.12"\n*

3.2.1.4 Gateway (:SYSTem:COMMunicate:LAN:GATeway)

SYNTAX	:SYSTem:COMMunicate:LAN:GATeway <"xxx.xxx.xxx.xxx"> :SYSTem:COMMunicate:LAN:GATeway?
DESCRIPTION	This command sets/gets the gateway of the device when the IP assignment is set to Static.
DATA TYPE	String
RANGE	Conforms to the IP address standard (0-255:0-255:0-255:0-255)
RETURN	String
EQUIVALENT MENU	UTILITY > Interface > LAN Setting > Gateway
EXAMPLE	*:SYSTem:COMMunicate:LAN:GATeway "192.168.1.1"* *:SYSTem:COMMunicate:LAN:GATeway?* Return: *"192.168.1.1"\n*

3.2.1.5 Subnet Mask (:SYSTem:COMMunicate:LAN:SMASk)

SYNTAX	:SYSTem:COMMunicate:LAN:SMASk <"xxx.xxx.xxx.xxx"> :SYSTem:COMMunicate:LAN:SMASk?
DESCRIPTION	This command sets/gets the subnet mask of the device when the IP assignment is set to Static.
DATA TYPE	String
RANGE	Conforms to the IP address standard (0-255:0-255:0-255:0-255)
RETURN	String
EQUIVALENT MENU	UTILITY > Interface > LAN Setting > Subnet Mask
EXAMPLE	*:SYSTem:COMMunicate:LAN:SMASk "255.255.255.0"* *:SYSTem:COMMunicate:LAN:SMASk?* Return: *"255.255.255.0"\n*

3.2.1.6 IP Config (:SYSTem:COMMunicate:LAN:TYPE)

SYNTAX	:SYSTem:COMMunicate:LAN:TYPE STATIC|DHCP :SYSTem:COMMunicate:LAN:TYPE?
DESCRIPTION	This command sets/gets IP assignment type.
DATA TYPE	Enumeration
RANGE	STATIC|DHCP
RETURN	Enumeration
EQUIVALENT MENU	UTILITY > Interface > LAN Setting > DHCP State

EXAMPLE	*:SYSTem:COMMunicate:LAN:TYPE STATIC* *:SYSTem:COMMunicate:LAN:TYPE?* Return: *STATIC\n*

3.2.1.7 Language (SYSTem:LANGuage)

SYNTAX	:SYSTem:LANGuage CHINese|ENGLish :SYSTem:LANGuage?
DESCRIPTION	This command sets/gets the system language.
DATA TYPE	Enumeration
RANGE	CHINese|ENGLish
RETURN	Enumeration
EQUIVALENT MENU	UTILITY > Setting > Language
EXAMPLE	*:SYSTem:LANGuage CHINese* *:SYSTem:LANGuage?* Return: *CHINese\n*

3.2.1.8 Screen Saver (SYSTem:SCReen:SAVer)

SYNTAX	:SYSTem:SCReen:SAVer OFF|10S|1MIN|5MIN|15MIN|30MIN|1HOUR|2HOUR :SYSTem:SCReen:SAVer?
DESCRIPTION	This command sets/gets the type of system screen saver.
DATA TYPE	Enumeration
RANGE	OFF|10S|1MIN|5MIN|15MIN|30MIN|1HOUR|2HOUR
RETURN	Enumeration
DEFAULT VALUE	OFF
EQUIVALENT MENU	UTILITY > Setting > Screen Saver
EXAMPLE	*:SYSTem:SCReen:SAVer 30MIN* *:SYSTem:SCReen:SAVer?* Return: *30MIN\n*

3.2.1.9 Beeper (SYSTem:ALARm)

SYNTAX	:SYSTem:ALARm ON|OFF|1|0 :SYSTem:ALARm?
DESCRIPTION	This command sets/gets the state of system beeper.
DATA TYPE	Boolean
RANGE	ON|OFF|1|0

RETURN	1|0
DEFAULT VALUE	0
EQUIVALENT MENU	UTILITY > Setting > Beeper
EXAMPLE	*:SYSTem:ALARm ON* *:SYSTem:ALARm?* Return: *1\n*

3.2.1.10 Setup Type (:SYSTem:PON:TYPE)

SYNTAX	:SYSTem:PON:TYPE DFT|LAST :SYSTem:PON:TYPE?
DESCRIPTION	This command sets/gets the system startup type.
DATA TYPE	Enumeration
RANGE	DFT|LAST
RETURN	Enumeration
DEFAULT VALUE	None
EQUIVALENT MENU	UTILITY > Setting > Setup Type
EXAMPLE	*:SYSTem:PON:TYPE LAST* *:SYSTem:PON:TYPE?* Return: *LAST\n*

3.2.1.11 Power On Line (SYSTem:POWeron:TYPE)

SYNTAX	:SYSTem:POWeron:TYPE ON|OFF|1|0 :SYSTem:POWeron:TYPE?
DESCRIPTION	This command sets/gets whether the system is powered on or not.
DATA TYPE	Boolean
RANGE	ON|OFF|1|0
RETURN	1|0
DEFAULT VALUE	0
EQUIVALENT MENU	UTILITY > Setting > Power On Line
EXAMPLE	*:SYSTem:POWeron:TYPE ON* *:SYSTem:POWeron:TYPE?* Return: *1\n*

3.2.1.12 10M Adjustment State (:SYSTem:REF:DAC:STAT)

SYNTAX	:SYSTem:REF:DAC:STAT ON|OFF|1|0 :SYSTem:REF:DAC:STAT?

DESCRIPTION	This command sets/gets the state of system 10M adjustment.
DATA TYPE	Boolean
RANGE	ON|OFF|1|0
RETURN	1|0
DEFAULT VALUE	0
EQUIVALENT MENU	UTILITY > Setting > 10M Adjustment
EXAMPLE	*:SYSTem:REF:DAC:STAT ON* *:SYSTem:REF:DAC:STAT?* Return: *1\n*

3.2.1.13 Ref Osc Code (:SYSTem:REF:DAC)

SYNTAX	:SYSTem:REF:DAC <value> :SYSTem:REF:DAC?
DESCRIPTION	This command sets/gets the system reference oscillator code.
DATA TYPE	Integer
RANGE	0 to 65535
RETURN	Integer
DEFAULT VALUE	None
EQUIVALENT MENU	UTILITY > Setting > 10M Adjustment > Ref Osc Code
EXAMPLE	*:SYSTem:REF:DAC 43000* *:SYSTem:REF:DAC?* Return: *43000\n*

3.2.1.14 Ref Osc Code Store (:SYSTem:REF:DAC:SAVE)

SYNTAX	:SYSTem:REF:DAC:SAVE <"file_name">
DESCRIPTION	This command saves the system's reference oscillator code into the file.
DATA TYPE	String
RANGE	None
EQUIVALENT MENU	UTILITY > Setting > 10M Adjustment > Save Ref Osc Setting
EXAMPLE	*:SYSTem:REF:DAC:SAVE "U-disk3/test.dac"*

3.2.1.15 Ref Osc Code Load (:SYSTem:REF:DAC:LOAD)

SYNTAX	:SYSTem:REF:DAC:LOAD <"file_name">
DESCRIPTION	This command loads the reference oscillator code from the file.

DATA TYPE	String
RANGE	None
EQUIVALENT MENU	UTILITY > Setting > 10M Adjustment > Recall Ref Osc Setting
EXAMPLE	*:SYSTem:REF:DAC:LOAD "U-disk3/test.dac"*

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

SYNTAX	:SYSTem:REF:DAC:DEFault
DESCRIPTION	This command resets the reference oscillator code to default value.
EQUIVALENT MENU	UTILITY > Setting > 10M Adjustment > Reset to Default
EXAMPLE	*:SYSTem:REF:DAC:DEFault*

3.2.1.17 GPIB Address (SYSTem:GPIB)

SYNTAX	:SYSTem:GPIB <value> :SYSTem:GPIB?
DESCRIPTION	This command sets/gets the GPIB address of the device.
DATA TYPE	Integer
RANGE	1 to 30
RETURN	Integer
DEFAULT VALUE	18
EQUIVALENT MENU	UTILITY > Interface > GPIB Address
EXAMPLE	*:SYSTem:GPIB 10* *:SYSTem:GPIB?* Return: *10\n*

3.2.1.18 Quit Remote Control State (SYSTem:REMote 0)

SYNTAX	:SYSTem:REMote 0
DESCRIPTION	Send this command to exit the remote mode of the system.
EQUIVALENT MENU	ESC
EXAMPLE	*:SYSTem:REMote 0*

3.2.2 System Reset/Preset

3.2.2.1 System Preset (:SYSTem:PRESet)

SYNTAX	:SYSTem:PRESet
DESCRIPTION	This command presets the status of the instrument based on the preset type.
EQUIVALENT MENU	UTILITY > Preset, or PRESET
EXAMPLE	Reset the instrument to its default configuration: *:SYSTem:PRESet:TYPE DFT* *:SYSTem:PRESet* Reset the instrument to its current configuration: *:SYSTem:PRESet:TYPE USER* *:SYSTem:PRESet:SAVE* *:SYSTem:PRESet* Reset the instrument to the configuration in the file: *:SYSTem:PRESet:TYPE USER* *:SYSTem:PRESet:PATH “Local/test.xml”* *:SYSTem:PRESet*

3.2.2.2 Preset Save (:SYSTem:PRESet:SAVE)

SYNTAX	:SYSTem:PRESet:SAVE
DESCRIPTION	This command saves the current system configuration.
EQUIVALENT MENU	None
EXAMPLE	Reset the instrument to its current configuration: *:SYSTem:PRESet:TYPE USER* *:SYSTem:PRESet:SAVE* *:SYSTem:PRESet*

3.2.2.3 Preset Path (:SYSTem:PRESet:PATH)

SYNTAX	:SYSTem:PRESet:PATH <”file_name”>
DESCRIPTION	This command saves the current system configuration into a file.
DATA TYPE	String
RANGE	None
EQUIVALENT MENU	UTILITY > Setting > Preset Type (User) > Save
EXAMPLE	*:SYSTem:PRESet:PATH “Local/test.xml”* *:SYSTem:PRESet:PATH “U-disk1/test.xml”*

3.2.2.4 Preset Type (:SYSTem:PRESet:TYPE)

SYNTAX	:SYSTem:PRESet:TYPE DFT|USER :SYSTem:PRESet:TYPE?
DESCRIPTION	This command sets/gets the preset type of the system.
DATA TYPE	Enumeration
RANGE	DFT|USER
RETURN	Enumeration
DEFAULT VALUE	DFT
EQUIVALENT MENU	UTILITY > Setting > Preset Type
EXAMPLE	*:SYSTem:PRESet:TYPE DFT* *:SYSTem:PRESet:TYPE?* Return: *DFT\n*

3.2.2.5 Factory Reset (:SYSTem:FDEFault)

SYNTAX	:SYSTem:FDEFault
DESCRIPTION	This command restores the instrument status to factory settings.
EQUIVALENT MENU	UTILITY > Setting > Factory Reset
EXAMPLE	*:SYSTem:FDEFault*

3.2.2.6 Reset & Clear (SYSTem:RESet:CLEar)

SYNTAX	:SYSTem:RESet:CLEar
DESCRIPTION	Restore the instrument status to factory settings and clear the user files in the Local folder.
EQUIVALENT MENU	UTILITY > Setting > Reset & Clear
EXAMPLE	*:SYSTem:RESet:CLEar*

3.3 OUTPut Commands

3.3.1 RF Output (:OUTPut[:STATe])

SYNTAX	:OUTPut[:STATe] ON|OFF|1|0 :OUTPut[:STATe]?
DESCRIPTION	This command sets/gets the output status of the RF port.
DATA TYPE	Boolean
RANGE	ON|OFF|1|0
RETURN	1|0
DEFAULT VALUE	0
EQUIVALENT SCPI	[:SOURce]:OUTPut ON|OFF|1|0
EQUIVALENT MENU	RF ON/OFF or FREQ > RF State
EXAMPLE	*OUTPut ON* *:OUTPut?* Return: *1\n*

3.3.2 Anolog Modulation State (:OUTPut:MODulation[:STATe])

SYNTAX	:OUTPut:MODulation[:STATe] ON|OFF|1|0 :OUTPut:MODulation[:STATe]?
DESCRIPTION	This command sets/gets the switch status of analog modulation.
DATA TYPE	Boolean
RANGE	ON|OFF|1|0
RETURN	1|0
DEFAULT VALUE	0
EQUIVALENT SCPI	[:SOURce]:MODulation ON|OFF|1|0 [:SOURce]:MODulation?
EQUIVALENT MENU	ANALOG MOD > On
EXAMPLE	*:OUTPut:MODulation ON* *:OUTPut:MODulation?* Return: *1\n*

3.4 SOURce Commands

3.4.1 RF Output ([:SOURce]:OUTPut)

SYNTAX	[:SOURce]:OUTPut ON|OFF|1|0
DESCRIPTION	This command sets the output status of the RF port.
DATA TYPE	Boolean
RANGE	ON|OFF|1|0
EQUIVALENT SCPI	:OUTPut[:STATe] ON|OFF|1|0
EQUIVALENT MENU	RF ON/OFF or FREQ > RF State
EXAMPLE	*:SOURce:OUTPut ON*

3.4.2 Software Trigger([:SOURce]:*TRG)

SYNTAX	[:SOURce]:*TRG
DESCRIPTION	When the trigger source is Bus, execute software trigger. Note: the trigger functions involved include sweep trigger, point sweep trigger, LF sweep trigger, pulse modulation trigger, ARB trigger and IoT trigger, etc.
EQUIVALENT MENU	None
EXAMPLE	**TRG*

3.4.3 Frequency

3.4.3.1 Frequency Display ([:SOURce]:FREQuency:DISPlay)

SYNTAX	[:SOURce]:FREQuency:DISPlay <freq> [:SOURce]:FREQuency:DISPlay?
DESCRIPTION	This command sets/gets the frequency display value in the parameter bar at the top of the screen.
DATA TYPE	Float, unit: Hz, kHz, MHz or GHz, default is Hz
RANGE	Frequency offset + Full frequency range
RETURN	Float, unit: Hz
DEFAULT VALUE	Maximum frequency
EQUIVALENT MENU	Freq
EXAMPLE	*:FREQuency:DISPlay 2 MHz* *:FREQuency:DISPlay?* Return: *2000000\n*

3.4.3.2 Frequency ([:SOURce]:FREQuency)

SYNTAX	[:SOURce]:FREQuency <freq> [:SOURce]:FREQuency?
DESCRIPTION	This command sets/gets the frequency of the RF output signal.
DATA TYPE	Float, unit: Hz, kHz, MHz or GHz, default is Hz
RANGE	Full frequency range
RETURN	Float, unit: Hz
DEFAULT VALUE	Maximum frequency
EQUIVALENT MENU	FREQ > Frequency
EXAMPLE	*:FREQuency 2 MHz* *:FREQuency?* Return: *2000000\n*

3.4.3.3 Frequency Offset ([:SOURce]:FREQuency:OFFSet)

SYNTAX	[:SOURce]:FREQuency:OFFSet <freq> [:SOURce]:FREQuency:OFFSet?
DESCRIPTION	This command sets/gets the frequency offset of the RF output signal.
DATA TYPE	Float, unit: Hz, kHz, MHz or GHz, default is Hz
RANGE	-200 GHz to 200 GHz
RETURN	Float, unit: Hz
DEFAULT VALUE	0
EQUIVALENT MENU	FREQ > Freq Offset
EXAMPLE	*:FREQuency:OFFSet -2 MHz* *:FREQuency:OFFSet?* Return: *-2000000\n*

3.4.4 Level

3.4.4.1 Level Display ([:SOURce]:POWer:POWer)

SYNTAX	[:SOURce]:POWer:POWer <power> [:SOURce]:POWer:POWer?
DESCRIPTION	This command sets/gets the level display value in the parameter bar at the top of the screen.
DATA TYPE	Float, unit: dBm, dBuV, uV, mV, V, nW, uW, mW or W, default is dBm
RANGE	Level Offset + Full power range
RETURN	Float, unit: dBm

DEFAULT VALUE	-130 dBm
EQUIVALENT MENU	Level
EXAMPLE	*:POWer:POWer -2* *:POWer:POWer?* Return: *-2\n*

3.4.4.2 Level ([:SOURce]:POWer)

SYNTAX	[:SOURce]:POWer <power> [:SOURce]:POWer?
DESCRIPTION	This command sets/gets the level of the RF output signal.
DATA TYPE	Float, unit: dBm, dBuV, uV, mV, V, nW, uW, mW or W, default is dBm
RANGE	Please refer to SSG6000A datasheet
RETURN	Float, unit: dBm
DEFAULT VALUE	-130 dBm
EQUIVALENT MENU	LEVEL > Level
EXAMPLE	*:POWer 2* *:POWer?* Return: *2\n*

3.4.4.3 Level ([:SOURce]:POWer[:LEVel][:IMMediate][:AMPLitude])

SYNTAX	[:SOURce]:POWer[:LEVel][:IMMediate][:AMPLitude] <power> [:SOURce]:POWer[:LEVel][:IMMediate][:AMPLitude]?
DESCRIPTION	This command sets/gets the level of the RF output signal.
DATA TYPE	Float, unit: dBm, dBuV, uV, mV, V, nW, uW, mW or W, default is dBm
RANGE	Please refer to SSG6000A datasheet
RETURN	Float, unit: dBm
DEFAULT VALUE	-130 dBm
EQUIVALENT MENU	LEVEL > Level
EXAMPLE	*:POWer:LEVel -5* *:POWer:LEVel?* Return: *-5\n*

3.4.4.4 Level Offset ([:SOURce]:POWer:OFFSet)

SYNTAX	[:SOURce]:POWer:OFFSet <power> [:SOURce]:POWer:OFFSet?
DESCRIPTION	This command sets/gets the level offset of the RF output signal.

DATA TYPE	Float, unit: dB
RANGE	-100 dB to 100 dB
RETURN	Float, unit: dB
DEFAULT VALUE	0
EQUIVALENT MENU	LEVEL > Level Offset
EXAMPLE	*:POWer:OFFSet 2* *:POWer:OFFSet?* Return: *2\n*

3.4.4.5 ALC State ([:SOURce]:POWer:ALC)

SYNTAX	[:SOURce]:POWer:ALC ON|OFF|AUTO [:SOURce]:POWer:ALC?
DESCRIPTION	This command sets/gets the ALC state.
DATA TYPE	Enumeration
RANGE	ON|OFF|AUTO
RETURN	Enumeration
DEFAULT VALUE	AUTO
EQUIVALENT MENU	LEVEL > ALC State
EXAMPLE	*:POWer:ALC ON* *:POWer:ALC?* Return: *ON\n*

3.4.4.6 Flatness List State ([:SOURce]:CORRection[:FLATness])

SYNTAX	[:SOURce]:CORRection[:FLATness] ON|OFF|1|0 [:SOURce]:CORRection[:FLATness]?
DESCRIPTION	This command sets/gets the state of flatness correction.
DATA TYPE	Boolean
RANGE	ON|OFF|1|0
RETURN	1|0
DEFAULT VALUE	0
EQUIVALENT MENU	LEVEL > Flatness
EXAMPLE	*:CORRection:FLATness ON* *:CORRection?* Return: *1\n*

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

SYNTAX	[:SOURce]:CORRection:FLATness:PAIR <freq>,<power>
DESCRIPTION	This command adds a line to the flatness list.
DATA TYPE	Freq: float, unit: Hz, kHz, MHz or GHz, default is Hz, Power: float, unit: dB
RANGE	Freq: Full frequency range, Power: -100 to 100
EQUIVALENT MENU	LEVEL > Flatness > Add
EXAMPLE	*CORRection:FLATness:PAIR 3 MHz,-3*

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

SYNTAX	[:SOURce]:CORRection:FLATness:DELete <row>
DESCRIPTION	This command removes a specified row from the flatness list.
DATA TYPE	Integer
RANGE	1 ~ total number of rows in the flatness list
EQUIVALENT MENU	LEVEL > Flatness > Delete
EXAMPLE	*:CORRection:FLATness:DELete 1*

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

SYNTAX	[:SOURce]:CORRection:FLATness:COUNt?
DESCRIPTION	This command queries the total number of rows of the flatness list.
RETURN	Integer
DEFAULT VALUE	0
EQUIVALENT MENU	LEVEL > Flatness
EXAMPLE	*:CORRection:FLATness:COUNt?* Return: *5\n*

3.4.4.10 Flatness List Store ([:SOURce]:CORRection:STORe)

SYNTAX	[:SOURce]:CORRection:STORe <"file_name">
DESCRIPTION	This command saves the flatness list into file.
DATA TYPE	String
RANGE	None
EQUIVALENT MENU	LEVEL > Flatness > Save
EXAMPLE	*:CORRection:STORe "U-disk3/test.uflt"*

3.4.4.11 Flatness List Load ([:SOURce]:CORRection:LOAD)

SYNTAX	[:SOURce]:CORRection:LOAD <"file_name">
DESCRIPTION	This command loads the flatness list from the file.
DATA TYPE	String
RANGE	None
EQUIVALENT MENU	LEVEL > Flatness > Open
EXAMPLE	*:CORRection:LOAD "U-disk3/test.uflt"*

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

SYNTAX	[:SOURce]:CORRection:FLATness:PRESet
DESCRIPTION	This command clears the flatness list.
EQUIVALENT MENU	LEVEL > Flatness > Clear
EXAMPLE	*:CORRection:FLATness:PRESet*

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

SYNTAX	[:SOURce]:CORRection:FLATness:FILL:TYPE FLATness|MANUal|SWEEPlist [:SOURce]:CORRection:FLATness:FILL:TYPE?
DESCRIPTION	This command sets/gets the fill type of the flatness list.
DATA TYPE	Enumeration
RANGE	FLATness|MANUal|SWEEPlist
RETURN	Enumeration
DEFAULT VALUE	FLATness
EQUIVALENT MENU	LEVEL > Flatness > Setting > Fill Type
EXAMPLE	*:CORRection:FLATness:FILL:TYPE FLATness* *:CORRection:FLATness:FILL:TYPE?* Return: *FLATness\n*

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

SYNTAX	[:SOURce]:CORRection:FLATness:STARtfreq <freq> [:SOURce]:CORRection:FLATness:STARtfreq?
DESCRIPTION	When the flatness list needs to be filled with a power sensor and the filling type is "Manual Step", this command sets/queries the starting frequency of manual step filling.
DATA TYPE	Float, unit: Hz, kHz, MHz or GHz, default is Hz
RANGE	Full frequency range

RETURN	Float, unit: Hz
DEFAULT VALUE	Maximum frequency
EQUIVALENT MENU	LEVEL > Flatness > Setting > Fill Type (Manual Step) > Start Freq
EXAMPLE	*:CORRection:FLATness:STARtfreq 200 MHz* *:CORRection:FLATness:STARtfreq?* Return: *200000000\n*

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

SYNTAX	[:SOURce]:CORRection:FLATness:STOPfreq <freq> [:SOURce]:CORRection:FLATness:STOPfreq?
DESCRIPTION	When the flatness list needs to be filled with a power sensor and the filling type is “Manual Step”, this command sets/queries the end frequency of manual step filling.
DATA TYPE	Float, unit: Hz, kHz, MHz or GHz, default is Hz
RANGE	Full frequency range
RETURN	Float, unit: Hz
DEFAULT VALUE	Maximum frequency
EQUIVALENT MENU	LEVEL > Flatness > Setting > Fill Type (Manual Step) > Stop Freq
EXAMPLE	*:CORRection:FLATness:STOPfreq 500 MHz* *:CORRection:FLATness:STOPfreq?* Return: *500000000\n*

3.4.4.16 Flatness List Fill Space ([:SOURce]:CORRection:FLATness:SPACe)

SYNTAX	[:SOURce]:CORRection:FLATness:SPACe LINear|LOGarithmic [:SOURce]:CORRection:FLATness:SPACe?
DESCRIPTION	When the flatness list needs to be filled with a power sensor and the filling type is “Manual Step”, this command sets/queries the frequency step method of manual step filling.
DATA TYPE	Enumeration
RANGE	LINear|LOGarithmic
RETURN	Enumeration
DEFAULT VALUE	LINear
EQUIVALENT MENU	LEVEL > Flatness > Setting > Fill Type (Manual Step) > Fill Space
EXAMPLE	*:CORRection:FLATness:SPACe LINear* *:CORRection:FLATness:SPACe?* Return: *LINear\n*

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

SYNTAX	[:SOURce]:CORRection:FLATness:LINStep <freq> [:SOURce]:CORRection:FLATness:LINStep?
DESCRIPTION	This command sets/queries the linear frequency step of manual step filling.
DATA TYPE	Float, unit: Hz, kHz, MHz or GHz, default is Hz
RETURN	Float, unit: Hz
DEFAULT VALUE	0
EQUIVALENT MENU	LEVEL > Flatness > Setting > Fill Type (Manual Step) > Step Linear
EXAMPLE	*:CORRection:FLATness:LINStep 200 MHz* *:CORRection:FLATness:LINStep?* Return: *200000000\n*

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

SYNTAX	[:SOURce]:CORRection:FLATness:LOGStep <value> [:SOURce]:CORRection:FLATness:LOGStep?
DESCRIPTION	This command sets/queries the logarithmic frequency step of manual step filling.
DATA TYPE	Float, unit: %
RETURN	Float, unit: %
DEFAULT VALUE	0
EQUIVALENT MENU	LEVEL > Flatness > Setting > Fill Type (Manual Step) > Step Log
EXAMPLE	*:CORRection:FLATness:LOGStep 20* *:CORRection:FLATness:LOGStep?* Return: *20\n*

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

SYNTAX	[:SOURce]:CORRection:FLATness:POINt <points> [:SOURce]:CORRection:FLATness:POINt?
DESCRIPTION	This command sets/queries the sweep points of manual step filling.
DATA TYPE	Integer
RANGE	2 to 500
RETURN	Integer
DEFAULT VALUE	2
EQUIVALENT MENU	LEVEL > Flatness > Setting > Fill Type (Manual Step) > Points
EXAMPLE	*:CORRection:FLATness:POINt 5* *:CORRection:FLATness:POINt?*

	Return: *5\n*

3.4.4.20 Fill Flatness with Sensor ([:SOURce]:CORRection:CSET:DATA[:SENSor][:POWer]:SONCe)

SYNTAX	[:SOURce]:CORRection:CSET:DATA[:SENSor][:POWer]:SONCe
DESCRIPTION	Amplitude correction values to populate flatness list with power sensor.
EQUIVALENT MENU	LEVEL > Flatness > Setting > Fill Flatness with Sensor
EXAMPLE	*:CORRection:CSET:DATA:SONCe*

3.4.5 Sweep

3.4.5.1 Sweep State ([:SOURce]:SWEep:STATe)

SYNTAX	[:SOURce]:SWEep:STATe OFF|FREQuency|LEVel|LEV_FREQ [:SOURce]:SWEep:STATe?
DESCRIPTION	This command sets/gets the sweep state.
DATA TYPE	Enumeration
RANGE	OFF|FREQuency|LEVel|LEV_FREQ
RETURN	Enumeration
DEFAULT VALUE	OFF
EQUIVALENT MENU	SWEEP > Sweep State
EXAMPLE	*:SWEep:STATe LEV_FREQ* *:SWEep:STATe?* Return: *LEV_FREQ\n*

3.4.5.2 Sweep Type ([:SOURce]:SWEep:TYPE)

SYNTAX	[:SOURce]:SWEep:TYPE LIST|STEP [:SOURce]:SWEep:TYPE?
DESCRIPTION	This command sets the sweep type to step sweep or list sweep. This command queries whether the sweep type is step sweep or list sweep.
DATA TYPE	Enumeration
RANGE	LIST|STEP
RETURN	Enumeration
DEFAULT VALUE	STEP
EQUIVALENT MENU	SWEEP > Step Sweep / List Sweep

EXAMPLE	*:SWEep:TYPE STEP* *:SWEep:TYPE?* Return: *STEP\n*

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

SYNTAX	[:SOURce]:SWEep:STEP:STARt:FREQuency <freq> [:SOURce]:SWEep:STEP:STARt:FREQuency?
DESCRIPTION	This command sets/queries the starting frequency of step sweep.
DATA TYPE	Float, unit: Hz, kHz, MHz or GHz, default is Hz
RANGE	Full frequency range
RETURN	Float, unit: Hz
DEFAULT VALUE	Maximum frequency
EQUIVALENT MENU	SWEEP > Step Sweep > Start Freq
EXAMPLE	*:SWEep:STEP:STARt:FREQuency 1 GHz* *:SWEep:STEP:STARt:FREQuency?* Return: *1000000000\n*

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

SYNTAX	[:SOURce]:SWEep:STEP:STOP:FREQuency <freq> [:SOURce]:SWEep:STEP:STOP:FREQuency?
DESCRIPTION	This command sets/queries the stop frequency of step sweep.
DATA TYPE	Float, unit: Hz, kHz, MHz or GHz, default is Hz
RANGE	Full frequency range
RETURN	Float, unit: Hz
DEFAULT VALUE	Maximum frequency
EQUIVALENT MENU	SWEEP > Step Sweep > Stop Freq
EXAMPLE	*:SWEep:STEP:STOP:FREQuency 1 GHz* *:SWEep:STEP:STOP:FREQuency?* Return: *1000000000\n*

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

SYNTAX	[:SOURce]:SWEep:STEP:STARt:LEVel <level> [:SOURce]:SWEep:STEP:STARt:LEVel?
DESCRIPTION	This command sets/queries the starting level of step sweep.
DATA TYPE	Float, unit: dBm, dBuV, uV, mV, V, nW, uW, mW or W, default is dBm
RANGE	Please refer to SSG6000A datasheet

RETURN	Float, unit: dBm
DEFAULT VALUE	-130 dBm
EQUIVALENT MENU	SWEEP > Step Sweep > Start Level
EXAMPLE	*:SWEep:STEP:STARt:LEVel 0 dBm* *:SWEep:STEP:STARt:LEVel?* Return: *0\n*

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

SYNTAX	[:SOURce]:SWEep:STEP:STOP:LEVel <level> [:SOURce]:SWEep:STEP:STOP:LEVel?
DESCRIPTION	This command sets/queries the stop level of step sweep.
DATA TYPE	Float, unit: dBm, dBuV, uV, mV, V, nW, uW, mW or W, default is dBm
RANGE	Please refer to SSG6000A datasheet
RETURN	Float, unit: dBm
DEFAULT VALUE	-130 dBm
EQUIVALENT MENU	SWEEP > Step Sweep > Stop Level
EXAMPLE	*:SWEep:STEP:STOP:LEVel 0 dBm* *:SWEep:STEP:STOP:LEVel?* Return: *0\n*

3.4.5.7 Dwell Time ([:SOURce]:SWEep:STEP:DWELl)

SYNTAX	[:SOURce]:SWEep:STEP:DWELl <time> [:SOURce]:SWEep:STEP:DWELl?
DESCRIPTION	This command sets/queries the dwell time of step sweep.
DATA TYPE	Float, unit: ns, us, ms or s, default is s
RANGE	10 ms ~ 100 s
RETURN	Float, unit: s
DEFAULT VALUE	30 ms
EQUIVALENT MENU	SWEEP > Step Sweep > Dwell Time
EXAMPLE	*:SWEep:STEP:DWELl 20 ms* *:SWEep:STEP:DWELl?* Return: *0.02\n*

3.4.5.8 Sweep Points ([:SOURce]:SWEep:STEP:POINts)

SYNTAX	[:SOURce]:SWEep:STEP:POINts <points> [:SOURce]:SWEep:STEP:POINts?

DESCRIPTION	This command sets/queries the sweep points of step sweep.
DATA TYPE	Integer
RANGE	2 to 65535
RETURN	Integer
DEFAULT VALUE	11
EQUIVALENT MENU	SWEEP > Step Sweep > Sweep Points
EXAMPLE	*:SWEep:STEP:POINts 2* *:SWEep:STEP:POINts?* Return: *2\n*

3.4.5.9 Sweep Shape ([:SOURce]:SWEep:STEP:SHAPe)

SYNTAX	[:SOURce]:SWEep:STEP:SHAPe TRIangle|SAWtooth [:SOURce]:SWEep:STEP:SHAPe?
DESCRIPTION	This command sets/queries the sweep shape of step sweep.
DATA TYPE	Enumeration
RANGE	TRIangle|SAWtooth
RETURN	Enumeration
DEFAULT VALUE	SAWtooth
EQUIVALENT MENU	SWEEP > Step Sweep > Sweep Shape
EXAMPLE	*:SWEep:STEP:SHAPe TRIangle* *:SWEep:STEP:SHAPe?* Return: *TRIangle\n*

3.4.5.10 Sweep Space ([:SOURce]:SWEep:STEP:SPACe)

SYNTAX	[:SOURce]:SWEep:STEP:SPACe LINear|LOGarithmic [:SOURce]:SWEep:STEP:SPACe?
DESCRIPTION	This command sets/queries the frequency sweep space of step sweep.
DATA TYPE	Enumeration
RANGE	LINear|LOGarithmic
RETURN	Enumeration
DEFAULT VALUE	LINear
EQUIVALENT MENU	SWEEP > Step Sweep > Sweep Space
EXAMPLE	*:SWEep:STEP:SPACe LOGarithmic* *:SWEep:STEP:SPACe?* Return: *LOGarithmic\n*

3.4.5.11 Linear Sweep Step ([:SOURce]:SWEep[:FREQuency]:STEP[:LINear])

SYNTAX	[:SOURce]:SWEep[:FREQuency]:STEP[:LINear] <freq> [:SOURce]:SWEep[:FREQuency]:STEP[:LINear]?
DESCRIPTION	This command sets/queries the linear frequency step of the step sweep.
DATA TYPE	Float, unit: Hz, kHz, MHz or GHz, default is Hz
RANGE	None
RETURN	Float, unit: Hz
DEFAULT VALUE	0
EQUIVALENT MENU	SWEEP > Step Sweep > Freq Step Linear
EXAMPLE	*:SWEep:STEP 200 MHz* *:SWEep:STEP?* Return: *200000000\n*

3.4.5.12 Logarithmic Sweep Step ([:SOURce]:SWEep[:FREQuency]:STEP:LOGarithmic)

SYNTAX	[:SOURce]:SWEep[:FREQuency]:STEP:LOGarithmic <value> [:SOURce]:SWEep[:FREQuency]:STEP:LOGarithmic?
DESCRIPTION	This command sets/queries the logarithmic frequency step of the step sweep.
DATA TYPE	Float, unit: %
RANGE	None
RETURN	Float, unit: %
DEFAULT VALUE	0
EQUIVALENT MENU	SWEEP > Step Sweep > Freq Step Log
EXAMPLE	*:SWEep:STEP:LOGarithmic 20* *:SWEep:STEP:LOGarithmic?* Return: *20\n*

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

SYNTAX	[:SOURce]:SWEep:LIST:ADDList <freq>,<level>,<time>
DESCRIPTION	This command adds a line to the sweep list.
DATA TYPE	Freq: float, unit: Hz, kHz, MHz or GHz, default is Hz, Level: float, unit: dBm, dBuV, uV, mV, V, nW, uW, mW or W, default is dBm, Time: float, unit: ns, us, ms or s, default is s
RANGE	Freq: Full frequency range, Level: Full level range, Time: 10.0 ms ~ 100.0 s
EQUIVALENT MENU	SWEEP > List Sweep > Add

EXAMPLE	*:SWEep:LIST:ADDList 1 GHz,0 dBm,1 s*

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

SYNTAX	[:SOURce]:SWEep:LIST:DELete <row>
DESCRIPTION	This command removes a specified row from the sweep list.
DATA TYPE	Integer
RANGE	1 ~ total number of rows in the sweep list
EQUIVALENT MENU	SWEEP > List Sweep > Delete
EXAMPLE	*:SWEep:LIST:DELete 1*

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

SYNTAX	[:SOURce]:SWEep:LIST:CHANGe <row>,<freq>,<level>,<time>
DESCRIPTION	This command edits the specified row in the sweep list.
DATA TYPE	Row: Integer, Freq: float, unit: Hz, kHz, MHz or GHz, default is Hz, Level: float, unit: dBm, dBuV, uV, mV, V, nW, uW, mW or W, default is dBm, Time: float, unit: ns, us, ms or s, default is s
RANGE	Row: 1 ~ total number of rows in the sweep list, Freq: Full frequency range, Level: Full level range, Time: 10.0 ms ~ 100.0 s
EQUIVALENT MENU	SWEEP > List Sweep
EXAMPLE	*:SWEep:LIST:CHANGe 1,1 GHz,1 dBm, 1 s*

3.4.5.16 Sweep List Count ([:SOURce]:SWEep:LIST:CPOint?)

SYNTAX	[:SOURce]:SWEep:LIST:CPOint?
DESCRIPTION	This command queries the total number of rows of the sweep list.
RETURN	Integer
DEFAULT VALUE	1
EQUIVALENT MENU	SWEEP > List Sweep
EXAMPLE	*:SWEep:LIST:CPOint?* Return: *5\n*

3.4.5.17 Sweep List Data ([:SOURce]:SWEep:LIST:LIST?)

SYNTAX	[:SOURce]:SWEep:LIST:LIST? <begin_row>,<end_row>

DESCRIPTION	This command queries the data from begin_row to end_row in the sweep list.
DATA TYPE	Integer, Integer
RANGE	1 ~ total number of rows in the sweep list, Start row ~ total number of rows in the sweep list
RETURN	String
DEFAULT VALUE	None
EQUIVALENT MENU	SWEEP > List Sweep
EXAMPLE	*:SWEep:LIST:LIST? 1,3* Return: *1000000000,0,0.03\s2000000000,-2.4,0.03\s3000000000,-4.8,0.03\n*

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

SYNTAX	[:SOURce]:SWEep:LIST:INITialize:PRESet
DESCRIPTION	This command clears the sweep list.
EQUIVALENT MENU	SWEEP > List Sweep > Clear
EXAMPLE	*:SWEep:LIST:INITialize:PRESet*

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

SYNTAX	[:SOURce]:SWEep:LIST:INITialize:FSTep
DESCRIPTION	This command imports the sweep list from step sweep.
EQUIVALENT MENU	SWEEP > List Sweep > Import
EXAMPLE	*:SWEep:LIST:INITialize:FSTep*

3.4.5.20 Sweep List Load ([:SOURce]:SWEep:LOAD)

SYNTAX	[:SOURce]:SWEep:LOAD <”file_name”>
DESCRIPTION	This command loads the sweep list from the file.
DATA TYPE	String
RANGE	None
EQUIVALENT MENU	SWEEP > List Sweep > Open
EXAMPLE	*:SWEep:LOAD “U-disk3/test.lsw”*

3.4.5.21 Sweep List Store ([:SOURce]:SWEep:STORe)

SYNTAX	[:SOURce]:SWEep:STORe <”file_name”>
DESCRIPTION	This command saves the sweep list into file.

DATA TYPE	String
RANGE	None
EQUIVALENT MENU	SWEEP > List Sweep > Save
EXAMPLE	*:SWEep:STORe “U-disk3/test.lsw”*

3.4.5.22 Sweep Direction ([:SOURce]:SWEep:DIRect)

SYNTAX	[:SOURce]:SWEep:DIRect FWD|REV [:SOURce]:SWEep:DIRect?
DESCRIPTION	This command sets/queries the sweep direction.
DATA TYPE	Enumeration
RANGE	FWD|REV
RETURN	Enumeration
DEFAULT VALUE	FWD
EQUIVALENT MENU	SWEEP > Direction
EXAMPLE	*:SWEep:DIRect REV* *:SWEep:DIRect?* Return: *REV\n*

3.4.5.23 Sweep Mode ([:SOURce]:SWEep:MODE)

SYNTAX	[:SOURce]:SWEep:MODE CONTinue|SINGle [:SOURce]:SWEep:MODE?
DESCRIPTION	This command sets the sweep mode to continuous or single. This command queries whether the sweep mode is continuous or single.
DATA TYPE	Enumeration
RANGE	CONTinue|SINGle
RETURN	Enumeration
DEFAULT VALUE	CONTinue
EQUIVALENT MENU	SWEEP > Sweep Mode
EXAMPLE	*:SWEep:MODE SINGle* *:SWEep:MODE?* Return: *SINGle\n*

3.4.5.24 Execute Single Sweep ([:SOURce]:SWEep:EXECute)

SYNTAX	[:SOURce]:SWEep:EXECute
DESCRIPTION	When the sweep mode is single, this command can perform a single

	sweep.
EQUIVALENT MENU	SWEEP > Execute single sweep
EXAMPLE	*:SWEep:EXECute*

3.4.5.25 Trigger Mode ([:SOURce]:SWEep:SWEep:TRIGger:TYPE)

SYNTAX	[:SOURce]:SWEep:SWEep:TRIGger:TYPE AUTO|KEY|BUS|EXT [:SOURce]:SWEep:SWEep:TRIGger:TYPE?
DESCRIPTION	This command sets/queries the sweep trigger mode.
DATA TYPE	Enumeration
RANGE	AUTO|KEY|BUS|EXT
RETURN	Enumeration
DEFAULT VALUE	AUTO
EQUIVALENT MENU	SWEEP > Trigger Mode
EXAMPLE	*:SWEep:SWEep:TRIGger:TYPE KEY* *:SWEep:SWEep:TRIGger:TYPE?* Return: *KEY\n*

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

SYNTAX	[:SOURce]:SWEep:POINt:TRIGger:TYPE AUTO|KEY|BUS|EXT [:SOURce]:SWEep:POINt:TRIGger:TYPE?
DESCRIPTION	This command sets/queries the point sweep trigger mode.
DATA TYPE	Enumeration
RANGE	AUTO|KEY|BUS|EXT
RETURN	Enumeration
DEFAULT VALUE	AUTO
EQUIVALENT MENU	SWEEP > Point Trigger
EXAMPLE	*:SWEep:POINt:TRIGger:TYPE KEY* *:SWEep:POINt:TRIGger:TYPE?* Return: *KEY\n*

3.4.5.27 Trigger Slope ([:SOURce]:INPut:TRIGger:SLOPe)

SYNTAX	[:SOURce]:INPut:TRIGger:SLOPe POSitive|NEGative [:SOURce]:INPut:TRIGger:SLOPe?
DESCRIPTION	This command sets/queries the trigger edge of the external trigger signal for the sweep function.
DATA TYPE	Enumeration

RANGE	POSitive|NEGative
RETURN	Enumeration
DEFAULT VALUE	POSitive
EQUIVALENT MENU	SWEEP > Trigger Slope
EXAMPLE	*:INPut:TRIGger:SLOPe NEGative* *:INPut:TRIGger:SLOPe?* Return: *NEGative\n*

3.4.5.28 Get Current Sweep Point ([:SOURce]:SWEep:CURRent:DATA?)

SYNTAX	[:SOURce]:SWEep:CURRent:DATA?
DESCRIPTION	This command queries the current sweep point.
RETURN	String The string format: index,{freq,level,time} Index: interger, Freq: float, unit: Hz, Level: float, unit: dBm, Time: float, unit: s.
DEFAULT VALUE	None
EQUIVALENT MENU	Display frequency and display level at the top of the screen
EXAMPLE	*:SWEep:CURRent:DATA?* Return: *1,{1000000000,0,0.03}\n*

3.4.5.29 Get the Frequency of the Current Sweep Point ([:SOURce]:SWEep:CURRent:FREQuency?)

SYNTAX	[:SOURce]:SWEep:CURRent:FREQuency?
DESCRIPTION	This command queries the frequency of the current sweep point.
RETURN	Float, unit: Hz
DEFAULT VALUE	None
EQUIVALENT MENU	Display frequency at the top of the screen
EXAMPLE	*:SWEep:CURRent:FREQuency?* Return: *1000000000.000000\n*

3.4.5.30 Get the Level of the Current Sweep Point ([:SOURce]:SWEep:CURRent:LEVel?)

SYNTAX	[:SOURce]:SWEep:CURRent:LEVel?
DESCRIPTION	This command queries the level of the current sweep point.
RETURN	Float, unit: dBm

DEFAULT VALUE	None
EQUIVALENT MENU	Display level at the top of the screen
EXAMPLE	*:SWEep:CURRent:LEVel?* Return: *0.000000\n*

3.4.6 Sensor

3.4.6.1 Level Control ([:SOURce]:POWer:SPC:STATe)

SYNTAX	[:SOURce]:POWer:SPC:STATe ON|OFF|1|0 [:SOURce]:POWer:SPC:STATe?
DESCRIPTION	This command sets/queries the level control state of the power sensor.
DATA TYPE	Boolean
RANGE	ON|OFF|1|0
RETURN	1|0
DEFAULT VALUE	0
EQUIVALENT SCPI	:SENSe[:POWer]:LEV:CTL:STATe ON|OFF|1|0 :SENSe[:POWer]:LEV:CTL:STATe?
EQUIVALENT MENU	HOME > POWER SENSOR > Level Control
EXAMPLE	*:POWer:SPC:STATe ON* *:POWer:SPC:STATe?* Return: *1\n*

3.4.6.2 Target Level ([:SOURce]:POWer:SPC:TARGet)

SYNTAX	[:SOURce]:POWer:SPC:TARGet <power> [:SOURce]:POWer:SPC:TARGet?
DESCRIPTION	This command sets/queries the targrt level of the power sensor level control.
DATA TYPE	Float, unit: dBm, dBuV, uV, mV, V, nW, uW, mW or W, default is dBm
RANGE	-120 dBm ~ 20 dBm
RETURN	Float, unit: dBm
DEFAULT VALUE	0
EQUIVALENT SCPI	:SENSe[:POWer]:SPC:TARGet <power> :SENSe[:POWer]:SPC:TARGet?
EQUIVALENT MENU	HOME > POWER SENSOR > Level Control > Target Level
EXAMPLE	*:POWer:SPC:TARGet 0* *:POWer:SPC:TARGet?* Return: *0\n*

3.4.6.3 Limit Level ([:SOURce]:POWer:LIMit)

SYNTAX	[:SOURce]:POWer:LIMit <power> [:SOURce]:POWer:LIMit?
DESCRIPTION	This command sets/queries the Limit level of the power sensor level control.
DATA TYPE	Float, unit: dBm, dBuV, uV, mV, V, nW, uW, mW or W, default is dBm
RANGE	-120 dBm ~ 20 dBm
RETURN	Float, unit: dBm
DEFAULT VALUE	0
EQUIVALENT SCPI	:SENSe[:POWer]:LIMit <power> :SENSe[:POWer]:LIMit?
EQUIVALENT MENU	HOME > POWER SENSOR > Level Control > Limit Level
EXAMPLE	*POWer:LIMit 1* *POWer:LIMit?* Return: *1\n*

3.4.6.4 Catch Range ([:SOURce]:POWer:SPC:CRANge)

SYNTAX	[SOURce]:POWer:SPC:CRANge <power> [SOURce]:POWer:SPC:CRANge?
DESCRIPTION	This command sets/queries the catch range of the power sensor level control.
DATA TYPE	Float, unit: dB
RANGE	0 ~ 50
RETURN	Float, unit: dB
DEFAULT VALUE	0
EQUIVALENT SCPI	:SENSe[:POWer]:SPC:CRANge <power> :SENSe[:POWer]:SPC:CRANge?
EQUIVALENT MENU	HOME > POWER SENSOR > Level Control > Catch Range
EXAMPLE	*:POWer:SPC:CRANge 5* *:POWer:SPC:CRANge?* Return: *5\n*

3.4.7 Anolog Modulation

3.4.7.1 Anolog Modulation State ([:SOURce]:MODulation)

SYNTAX	[:SOURce]:MODulation ON|OFF|1|0 [:SOURce]:MODulation?
DESCRIPTION	This command sets/gets the switch status of analog modulation.

DATA TYPE	Boolean
RANGE	ON|OFF|1|0
RETURN	1|0
DEFAULT VALUE	0
EQUIVALENT SCPI	:OUTPut:MODulation[:STATe] ON|OFF|1|0 :OUTPut:MODulation[:STATe]?
EQUIVALENT MENU	ANALOG MOD > On
EXAMPLE	*:MODulation ON* *:MODulation?* Return: *1\n*

3.4.8 AM

3.4.8.1 AM State ([:SOURce]:AM:STATe)

SYNTAX	[:SOURce]:AM:STATe ON|OFF|1|0 [:SOURce]:AM:STATe?
DESCRIPTION	This command sets/gets the switch status of amplitude modulation.
DATA TYPE	Boolean
RANGE	ON|OFF|1|0
RETURN	1|0
DEFAULT VALUE	0
EQUIVALENT MENU	MOD > AM > AM State
EXAMPLE	*:AM:STATe ON* *:AM:STATe?* Return: *1\n*

3.4.8.2 AM Shape ([:SOURce]:AM:WAVEform)

SYNTAX	[:SOURce]:AM:WAVEform SINE|SQUAre [:SOURce]:AM:WAVEform?
DESCRIPTION	This command sets/queries the modulation waveform of AM modulation.
DATA TYPE	Enumeration
RANGE	SINE|SQUAre
RETURN	Enumeration
DEFAULT VALUE	SINE
EQUIVALENT MENU	MOD > AM > AM Shape

EXAMPLE	*:AM:WAVEform SQUAre* *:AM:WAVEform?* Return: *SQUAre\n*

3.4.8.3 AM Source ([:SOURce]:AM:SOURce)

SYNTAX	[:SOURce]:AM:SOURce INTernal|EXTernal|INT,EXT [:SOURce]:AM:SOURce?
DESCRIPTION	This command sets/queries the modulation source of AM modulation.
DATA TYPE	Enumeration
RANGE	INTernal|EXTernal|INT,EXT
RETURN	Enumeration
DEFAULT VALUE	INTernal
EQUIVALENT MENU	MOD > AM > AM Source
EXAMPLE	*:AM:SOURce EXTernal* *:AM:SOURce?* Return: *EXTernal\n*

3.4.8.4 AM Depth ([:SOURce]:AM:DEPTh)

SYNTAX	[:SOURce]:AM:DEPTh <value> [:SOURce]:AM:DEPTh?
DESCRIPTION	This command sets/queries the modulation depth of AM modulation.
DATA TYPE	Float
RANGE	0 ~ 1
RETURN	Float
DEFAULT VALUE	0.5
EQUIVALENT MENU	MOD > AM > AM Depth
EXAMPLE	*:AM:DEPTh 0.2* *:AM:DEPTh?* Return: *0.2\n*

3.4.8.5 AM Rate ([:SOURce]:AM:FREQuency)

SYNTAX	[:SOURce]:AM:FREQuency <value> [:SOURce]:AM:FREQuency?
DESCRIPTION	This command sets/queries the modulation rate of AM modulation.
DATA TYPE	Float, unit: Hz, kHz, MHz or GHz, default is Hz
RANGE	Modulation wave is Sine: 0.01 Hz ~ 100 kHz,

	Modulation wave is Square: 0.01 Hz ~ 20 kHz.
RETURN	Float, unit: Hz
DEFAULT VALUE	1 kHz
EQUIVALENT MENU	MOD > AM > AM Rate
EXAMPLE	*:AM:FREQuency 10 kHz* *:AM:FREQuency?* Return: *10000\n*

3.4.8.6 AM Sensitivity ([:SOURce]:AM:SENSitivity?)

SYNTAX	[:SOURce]:AM:SENSitivity?
DESCRIPTION	This command queries the sensitivity of AM external modulation.
RETURN	Float, unit: /V
DEFAULT VALUE	0.125/V
EQUIVALENT MENU	MOD > AM > AM Sensitivity
EXAMPLE	*AM:SENSitivity?* Return: *0.125\n*

3.4.9 Pulse Modulation

3.4.9.1 Pulse State ([:SOURce]:PULM:STATe)

SYNTAX	[:SOURce]:PULM:STATe ON|OFF|1|0 [:SOURce]:PULM:STATe?
DESCRIPTION	This command sets/gets the switch status of pulse modulation.
DATA TYPE	Boolean
RANGE	ON|OFF|1|0
RETURN	1|0
DEFAULT VALUE	0
EQUIVALENT MENU	MOD > PULSE > Pulse State
EXAMPLE	*:PULM:STATe ON* *:PULM:STATe?* Return: *1\n*

3.4.9.2 Pulse Source ([:SOURce]:PULM:SOURce)

SYNTAX	[:SOURce]:PULM:SOURce INTernal|EXTernal [:SOURce]:PULM:SOURce?

DESCRIPTION	This command sets/queries the modulation source of pulse modulation.
DATA TYPE	Enumeration
RANGE	INTernal|EXTernal
RETURN	Enumeration
DEFAULT VALUE	INTernal
EQUIVALENT SCPI	[:SOURce]:PULM:SOURce:INT INTernal|EXTernal [:SOURce]:PULM:SOURce:INT?
EQUIVALENT MENU	MOD > PULSE > Pulse Source
EXAMPLE	*:PULM:SOURce INTernal* *:PULM:SOURce?* Return: *INTernal\n*

3.4.9.3 Pulse Source ([:SOURce]:PULM:SOURce:INT)

SYNTAX	[:SOURce]:PULM:SOURce:INT INTernal|EXTernal [:SOURce]:PULM:SOURce:INT?
DESCRIPTION	This command sets/queries the modulation source of pulse modulation.
DATA TYPE	Enumeration
RANGE	INTernal|EXTernal
RETURN	Enumeration
DEFAULT VALUE	INTernal
EQUIVALENT SCPI	[:SOURce]:PULM:SOURce INTernal|EXTernal [:SOURce]:PULM:SOURce?
EQUIVALENT MENU	MOD > PULSE > Pulse Source
EXAMPLE	*:PULM:SOURce:INT EXTernal* *:PULM:SOURce:INT?* Return: *EXTernal\n*

3.4.9.4 Pulse Out ([:SOURce]:PULM:OUT:STATe)

SYNTAX	[:SOURce]:PULM:OUT:STATe ON|OFF|1|0 [:SOURce]:PULM:OUT:STATe?
DESCRIPTION	This command sets/queries the pulse output status of pulse modulation.
DATA TYPE	Boolean
RANGE	ON|OFF|1|0
RETURN	1|0
DEFAULT VALUE	0

EQUIVALENT MENU	MOD > PULSE > Pulse Out
EXAMPLE	*:PULM:OUT:STATe ON* *:PULM:OUT:STATe?* Return: *1\n*

3.4.9.5 Pulse Out Polarity ([:SOURce]:PULM:POLarity)

SYNTAX	[:SOURce]:PULM:POLarity NORMal|INVerted [:SOURce]:PULM:POLarity?
DESCRIPTION	This command sets/queries the pulse output polarity.
DATA TYPE	Enumeration
RANGE	NORMal|INVerted
RETURN	Enumeration
DEFAULT VALUE	NORMal
EQUIVALENT MENU	MOD > PULSE > Pulse Out Polarity
EXAMPLE	*:PULM:POL INV* *:PULM:POLarity?* Return: *INVerted\n*

3.4.9.6 Pulse Mode ([:SOURce]:PULM:MODE)

SYNTAX	[:SOURce]:PULM:MODE SINGle|DOUBle|PTRain [:SOURce]:PULM:MODE?
DESCRIPTION	This command sets the pulse mode to Single pulse, Double pulse or pulse Train. This command queries the pulse mode.
DATA TYPE	Enumeration
RANGE	SINGle|DOUBle|PTRain
RETURN	Enumeration
DEFAULT VALUE	SINGle
EQUIVALENT MENU	MOD > PULSE > Pulse Mode
EXAMPLE	*PULM:MODE DOUB* *PULM:MODE?* Return: *DOUBle\n*

3.4.9.7 Pulse Period ([:SOURce]:PULM:PERiod)

SYNTAX	[:SOURce]:PULM:PERiod <value> [:SOURce]:PULM:PERiod?
DESCRIPTION	When the pulse mode is Single or Double, this command

	sets/queries the pulse period.
DATA TYPE	Float, unit: ns, us, ms or s, default is s
RANGE	40 ns ~ 300 s
RETURN	Float, unit: s
DEFAULT VALUE	10 ms
EQUIVALENT SCPI	[:SOURce]:PULM:INT[1]:PERiod <value> [:SOURce]:PULM:INT[1]:PERiod?
EQUIVALENT MENU	MOD > PULSE > Pulse Period
EXAMPLE	*PULM:PER 220 us* *PULM:PER?* Return: *0.00022\n*

3.4.9.8 Pulse Period ([:SOURce]:PULM:INT[1]:PERiod)

SYNTAX	[:SOURce]:PULM:INT[1]:PERiod <value> [:SOURce]:PULM:INT[1]:PERiod?
DESCRIPTION	When the pulse mode is Single or Double, this command sets/queries the pulse period.
DATA TYPE	Float, unit: ns, us, ms or s, default is s
RANGE	40 ns ~ 300 s
RETURN	Float, unit: s
DEFAULT VALUE	10 ms
EQUIVALENT SCPI	[:SOURce]:PULM:PERiod <value> [:SOURce]:PULM:PERiod?
EQUIVALENT MENU	MOD > PULSE > Pulse Period
EXAMPLE	*:PULM:INT1:PERiod 900 ns* *:PULM:INT1:PERiod?* Return: *9e-07\n*

3.4.9.9 Pulse Width ([:SOURce]:PULM:WIDTh)

SYNTAX	[:SOURce]:PULM:WIDTh <value> [:SOURce]:PULM:WIDTh?
DESCRIPTION	When the pulse mode is Single, this command sets/queries the pulse width; when the pulse mode is Double, this command sets/queries the pulse width of the first pulse.
DATA TYPE	Float, unit: ns, us, ms or s, default is s
RANGE	20 ns ~ 300 s
RETURN	Float, unit: s
DEFAULT VALUE	2 ms

EQUIVALENT SCPI	[:SOURce]:PULM:INT[1]:PWIDth <value> [:SOURce]:PULM:INT[1]:PWIDth?
EQUIVALENT MENU	MOD > PULSE > Pulse Width
EXAMPLE	*PULM:WIDT 33 us* *PULM:WIDT?* Return: *3.3e-05\n*

3.4.9.10 Pulse Width ([:SOURce]:PULM:INT[1]:PWIDth)

SYNTAX	[:SOURce]:PULM:INT[1]:PWIDth <value> [:SOURce]:PULM:INT[1]:PWIDth?
DESCRIPTION	When the pulse mode is Single, this command sets/queries the pulse width; when the pulse mode is Double, this command sets/queries the pulse width of the first pulse.
DATA TYPE	Float, unit: ns, us, ms or s, default is s
RANGE	20 ns ~ 300 s
RETURN	Float, unit: s
DEFAULT VALUE	2 ms
EQUIVALENT SCPI	[:SOURce]:PULM:WIDTh <value> [:SOURce]:PULM:WIDTh?
EQUIVALENT MENU	MOD > PULSE > Pulse Width
EXAMPLE	*:PULM:INT:PWIDth 400 ns* *:PULM:INT:PWIDth?* Return: *4e-07\n*

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

SYNTAX	[:SOURce]:PULM:DOUBle:DELay <value> [:SOURce]:PULM:DOUBle:DELay?
DESCRIPTION	When the pulse mode is Double, this command sets/queries the double pulse delay.
DATA TYPE	Float, unit: ns, us, ms or s, default is s
RANGE	20 ns ~ 300 s
RETURN	Float, unit: s
DEFAULT VALUE	4 ms
EQUIVALENT MENU	MOD > PULSE > Double Pulse Delay
EXAMPLE	*:PULM:DOUBle:DELay 2 ms* *:PULM:DOUBle:DELay?* Return: *0.002\n*

3.4.9.12 #2 Width ([:SOURce]:PULM:DOUBle:WIDTh)

SYNTAX	[:SOURce]:PULM:DOUBle:WIDTh <time> [:SOURce]:PULM:DOUBle:WIDTh?
DESCRIPTION	When the pulse mode is Double, this command sets/queries the pulse width of the second pulse.
DATA TYPE	Float, unit: ns, us, ms or s, default is s
RANGE	20 ns ~ 300 s
RETURN	Float, unit: s
DEFAULT VALUE	2 ms
EQUIVALENT MENU	MOD > PULSE > #2 Width
EXAMPLE	*PULM:DOUBle:WIDTh 5 ms* *PULM:DOUBle:WIDTh?* Return: *0.005\n*

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

SYNTAX	[:SOURce]:PULM:TRAin:PAIR <row>
DESCRIPTION	This command copies the specified line of the pulse train and pastes it in front of the specified line.
DATA TYPE	Integer
RANGE	1 ~ total number of rows in the pulse train
EQUIVALENT MENU	MOD > PULSE > Pulse Train > Add
EXAMPLE	*PULM:TRAin:PAIR 1*

3.4.9.14 Pulse Train Delete Row ([:SOURce]:PULM:TRAin:DELete)

SYNTAX	[:SOURce]:PULM:TRAin:DELete <row>
DESCRIPTION	This command removes a specified row from the pulse train.
DATA TYPE	Integer
RANGE	1 ~ total number of rows in the pulse train
EQUIVALENT MENU	MOD > PULSE > Pulse Train > Delete
EXAMPLE	*PULM:TRAin:DELete 5*

3.4.9.15 Pulse Train Edit On Time ([:SOURce]:PULM:TRAin:DATA:ONTIme)

SYNTAX	[:SOURce]:PULM:TRAin:DATA:ONTIme <raw>,<on_time>
DESCRIPTION	This command edits the positive pulse width of the specified row in the pulse train.

DATA TYPE	Row: Integer, On_time: float, unit: ns, us, ms or s, default is s
RANGE	Row: 1 ~ total number of rows in the pulse train, On_time: 20 ns ~ 300 s
EQUIVALENT MENU	MOD > PULSE > Pulse Train
EXAMPLE	*:PULM:TRAin:DATA:ONTIme 1,10 ms*

3.4.9.16 Pulse Train Edit Off Time ([:SOURce]:PULM:TRAin:DATA:OFFTime)

SYNTAX	[:SOURce]:PULM:TRAin:DATA:OFFTime <raw>,<off_time>
DESCRIPTION	This command edits the negative pulse width of the specified row in the pulse train.
DATA TYPE	Row: Integer, Off_time: float, unit: ns, us, ms or s, default is s
RANGE	Row: 1 ~ total number of rows in the pulse train, Off_time: 20 ns ~ 300 s
EQUIVALENT MENU	MOD > PULSE > Pulse Train
EXAMPLE	*:PULM:TRAin:DATA:OFFTime 1, 20 ms*

3.4.9.17 Pulse Train Edit Count ([:SOURce]:PULM:TRAin:DATA:COUNt)

SYNTAX	[:SOURce]:PULM:TRAin:DATA:COUNt <raw>,<count>
DESCRIPTION	This command edits the number of repetitions for a specified row of the pulse train.
DATA TYPE	Row: integer, Count: integer
RANGE	Row: 1 ~ total number of rows in the pulse train, Count: 1 ~ 65535
EQUIVALENT MENU	MOD > PULSE > Pulse Train
EXAMPLE	*:PULM:TRAin:DATA:COUNt 1,3*

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

SYNTAX	[:SOURce]:PULM:TRAin:CHANGe <row>,<on_time>,<off_time>,<count>
DESCRIPTION	This command edits the specified row of the pulse train.
DATA TYPE	Row: integer, On_time: float, unit: ns, us, ms or s, default is s, Off_time: float, unit: ns, us, ms or s, default is s, Count: integer
RANGE	Row: 1 ~ total number of rows in the pulse train, On_time: 20 ns ~ 300 s, Off_time: 20 ns ~ 300 s,

	Count: 1 ~ 65535
EQUIVALENT MENU	MOD > PULSE > Pulse Train
EXAMPLE	*:PULM:TRAin:CHANGe 2,3 ms,500 ns,4*

3.4.9.19 Pulse Train Data ([:SOURce]:PULM:TRAin:LIST?)

SYNTAX	[:SOURce]:PULM:TRAin:LIST? <begin_row>,<end_row>
DESCRIPTION	This command queries the data from begin_row to end_row in the pulse train.
DATA TYPE	Integer, Integer
RANGE	1 ~ total number of rows in the pulse train, Start row ~ total number of rows in the pulse train
RETURN	String
DEFAULT VALUE	None
EQUIVALENT MENU	MOD > PULSE > Pulse Train
EXAMPLE	*:PULM:TRAin:LIST? 1,3* Return: *0.001,0.005,1\s0.003,0.003,2\s0.004,0.002,1\n*

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

SYNTAX	[:SOURce]:PULM:TRAin:COUNt?
DESCRIPTION	This command queries the number of rows in the pulse train.
RETURN	String
DEFAULT VALUE	None
EQUIVALENT MENU	MOD > PULSE > Pulse Train
EXAMPLE	*:PULM:TRAin:COUNt?* Return: *5\n*

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

SYNTAX	[:SOURce]:PULM:TRAin:CLEAr
DESCRIPTION	This command clears the pulse train and restores its default value.
EQUIVALENT MENU	MOD > PULSE > Pulse Train > Clear
EXAMPLE	*PULM:TRAin:CLEAr*

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

SYNTAX	[:SOURce]:PULM:TRAin:LOAD <"file_name">

DESCRIPTION	This command loads the pulse train from the file.
DATA TYPE	String
RANGE	None
EQUIVALENT MENU	MOD > PULSE > Pulse Train > Load
EXAMPLE	*PULM:TRAin:LOAD "U-disk3/test.pulstrn"*

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

SYNTAX	[:SOURce]:PULM:TRAin:STORE <"file_name">
DESCRIPTION	This command saves the pulse train into file.
DATA TYPE	String
RANGE	None
EQUIVALENT MENU	MOD > PULSE > Pulse Train > Save
EXAMPLE	*PULM:TRAin:STORE "test.pulstrn"* *PULM:TRAin:STORE "U-disk1/test.pulstrn"*

3.4.9.24 Trigger Out ([:SOURce]:PULM:TRIGger:STATe)

SYNTAX	[:SOURce]:PULM:TRIGger:STATe ON|OFF|1|0 [:SOURce]:PULM:TRIGger:STATe?
DESCRIPTION	This command sets/gets the pulse trigger output status.
DATA TYPE	Boolean
RANGE	ON|OFF|1|0
RETURN	1|0
DEFAULT VALUE	1
EQUIVALENT MENU	MOD > PULSE > Trigger Out
EXAMPLE	*:PULM:TRIGger:STATe OFF* *:PULM:TRIGger:STATe?* Return: *0\n*

3.4.9.25 Pulse Trigger ([:SOURce]:PULM:TRIGger:MODE)

SYNTAX	[:SOURce]:PULM:TRIGger:MODE AUTO|KEY|EXTernal|EGATe [:SOURce]:PULM:TRIGger:MODE?
DESCRIPTION	This command sets/queries the pulse trigger mode.
DATA TYPE	Enumeration
RANGE	AUTO|KEY|EXTernal|EGATe
RETURN	Enumeration

DEFAULT VALUE	AUTO
EQUIVALENT MENU	MOD > PULSE > Pulse Trigger
EXAMPLE	*:PULM:TRIG:MODE EXTernal* *:PULM:TRIGger:MODE?* Return: *EXTernal\n*

3.4.9.26 Trigger Delay ([:SOURce]:PULM:DELay)

SYNTAX	[:SOURce]:PULM:DELay <value> [:SOURce]:PULM:DELay?
DESCRIPTION	When the pulse trigger mode is EXTernal, this command sets/queries the trigger delay.
DATA TYPE	Float, unit: ns, us, ms or s, default is s
RANGE	140 ns ~ 300 s
RETURN	Float, unit: s
DEFAULT VALUE	140 ns
EQUIVALENT MENU	MOD > PULSE > Trig Delay
EXAMPLE	*:PULM:DEL 30 ms* *:PULM:DELay?* Return: *0.03\n*

3.4.9.27 Trigger Slope ([:SOURce]:PULM:TRIGger:EXTernal:SLOPe)

SYNTAX	[:SOURce]:PULM:TRIGger:EXTernal:SLOPe NEGative|POSitive [:SOURce]:PULM:TRIGger:EXTernal:SLOPe?
DESCRIPTION	When the pulse trigger mode is EXTernal, this command sets/queries the trigger edge of the external trigger signal.
DATA TYPE	Enumeration
RANGE	NEGative|POSitive
RETURN	Enumeration
DEFAULT VALUE	POSitive
EQUIVALENT MENU	MOD > PULSE > Trigger Slope
EXAMPLE	*PULM:TRIG:EXT:SLOP NEG* *PULM:TRIG:EXT:SLOP?* Return: *NEGative\n*

3.4.9.28 Trigger Polarity ([:SOURce]:PULM:TRIGger:EXTernal:GATE:POLarity)

SYNTAX	[:SOURce]:PULM:TRIGger:EXTernal:GATE:POLarity NORMal|INVerted

	[:SOURce]:PULM:TRIGger:EXTernal:GATE:POLarity?
DESCRIPTION	This command sets/queries the trigger polarity of the external gating signal for the pulse function.
DATA TYPE	Enumeration
RANGE	NORMal|INVerted
RETURN	Enumeration
DEFAULT VALUE	NORMal
EQUIVALENT MENU	MOD > PULSE > Trig Polarity
EXAMPLE	*:PULM:TRIG:EXT:GATE:POL INVerted* *:PULM:TRIGger:EXTernal:GATE:POLarity?* Return: *INVerted\n*

3.4.10 LF Source

3.4.10.1 LF State ([:SOURce]:LFOutput)

SYNTAX	[:SOURce]:LFOutput ON|OFF|1|0 [:SOURce]:LFOutput?
DESCRIPTION	This command sets/queries the output status of LF Source.
DATA TYPE	Boolean
RANGE	ON|OFF|1|0
RETURN	1|0
DEFAULT VALUE	0
EQUIVALENT MENU	LF > LF Source > LF State
EXAMPLE	*LFOutput ON* *LFOutput?* Return: *1\n*

3.4.10.2 LF Level ([:SOURce]:LFOutput:VOLTage)

SYNTAX	[:SOURce]:LFOutput:VOLTage <voltage> [:SOURce]:LFOutput:VOLTage?
DESCRIPTION	This command sets/queries the level of the LF output signal.
DATA TYPE	Float, unit: dBm, uVpp, mVpp, Vpp, nW, uW or mW, default is Vpp
RANGE	1 mVpp ~ 3 Vpp
RETURN	Float, unit: Vpp
DEFAULT VALUE	0.5 Vpp

EQUIVALENT MENU	LF > LF Source > LF Level
EXAMPLE	*LFOutput:VOLTage 2 Vpp* *LFOutput:VOLTage?* Return: *2\n*

3.4.10.3 LF Offset ([:SOURce]:LFOutput:OFFSEt)

SYNTAX	[:SOURce]:LFOutput:OFFSEt <voltage> [:SOURce]:LFOutput:OFFSEt?
DESCRIPTION	This command sets/queries the amplitude offset of the LF output signal.
DATA TYPE	Float, unit: dBm, uV, mV, V, nW, uW or mW, default is V
RANGE	$\lvert LFoffset \rvert \le \min\,(2.5V - \frac{1}{2}LEVEL, 2V)$
RETURN	Float, unit: V
DEFAULT VALUE	0
EQUIVALENT MENU	LF > LF Source > LF Offset
EXAMPLE	*LFOutput:OFFSEt 1 V* *LFOutput:OFFSEt?* Return: *1\n*

3.4.10.4 LF Frequency ([:SOURce]:LFOutput:FREQuency)

SYNTAX	[:SOURce]:LFOutput:FREQuency <freq> [:SOURce]:LFOutput:FREQuency?
DESCRIPTION	This command sets/queries the frequency of the LF output signal.
DATA TYPE	Float, unit: Hz, kHz, MHz or GHz, default is Hz
RANGE	LF wave is Sine: 0.01 Hz ~ 1 MHz, LF wave is Square\Triangle\Sawtooth: 0.01 Hz ~ 20 kHz.
RETURN	Float, unit: Hz
DEFAULT VALUE	1 kHz
EQUIVALENT MENU	LF > LF Source > LF Frequency
EXAMPLE	*LFOutput:FREQuency 10 kHz* *LFOutput:FREQuency?* Return: *10000\n*

3.4.10.5 LF Shape ([:SOURce]:LFOutput:SHAPe)

SYNTAX	[:SOURce]:LFOutput:SHAPe SINE|SQUare|TRIangle|SAWTooth|DC [:SOURce]:LFOutput:SHAPe?

DESCRIPTION	This command sets/queries the wave shape of the LF output signal.
DATA TYPE	Enumeration
RANGE	SINE|SQUare|TRIangle|SAWTooth|DC
RETURN	Enumeration
DEFAULT VALUE	SINE
EQUIVALENT MENU	LF > LF Source > LF Shape
EXAMPLE	*:LFOutput:SHAPe TRIangle* *:LFOutput:SHAPe?* Return: *TRIangle\n*

3.4.10.6 LF Phase ([:SOURce]:LFOutput:PHASe)

SYNTAX	[:SOURce]:LFOutput:PHASe <deg> [:SOURce]:LFOutput:PHASe?
DESCRIPTION	This command sets/queries the phase of the LF output signal.
DATA TYPE	Float, unit: degree (°)
RANGE	-360 ~ 360
RETURN	Float, unit: degree (°)
DEFAULT VALUE	0
EQUIVALENT MENU	LF > LF Source > LF Phase
EXAMPLE	*LFOutput:PHASe 20* *LFOutput:PHASe?* Return: *20\n*

3.4.11 LF Sweep

3.4.11.1 Sweep State ([:SOURce]:LFOutput:SWEep)

SYNTAX	[:SOURce]:LFOutput:SWEep ON|OFF|1|0 [:SOURce]:LFOutput:SWEep?
DESCRIPTION	This command sets/queries the sweep status of LF signal.
DATA TYPE	Boolean
RANGE	ON|OFF|1|0
RETURN	1|0
DEFAULT VALUE	0
EQUIVALENT MENU	LF > LF Sweep > Sweep State
EXAMPLE	*:LFOutput:SWEep 1* *:LFOutput:SWEep?*

	Return: *1\n*

3.4.11.2 Sweep Direction ([:SOURce]:LFOutput:SWEep:DIRect)

SYNTAX	[:SOURce]:LFOutput:SWEep:DIRect UP|DOWN [:SOURce]:LFOutput:SWEep:DIRect?
DESCRIPTION	This command sets/queries the direction of LF sweep.
DATA TYPE	Enumeration
RANGE	UP|DOWN
RETURN	Enumeration
DEFAULT VALUE	UP
EQUIVALENT MENU	LF > LF Sweep > Direction
EXAMPLE	*:LFOutput:SWEep:DIRect DOWN* *:LFOutput:SWEep:DIRect?* Return: *DOWN\n*

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

SYNTAX	[:SOURce]:LFOutput:SWEep:STARt:FREQuency <freq> [:SOURce]:LFOutput:SWEep:STARt:FREQuency?
DESCRIPTION	This command sets/queries the starting frequency of LF sweep.
DATA TYPE	Float, unit: Hz, kHz, MHz or GHz, default is Hz
RANGE	LF wave is Sine: 0.01 Hz ~ 1 MHz, LF wave is Square/Triangle/Sawtooth: 0.01 Hz ~ 20 kHz.
RETURN	Float, unit: Hz
DEFAULT VALUE	500 Hz
EQUIVALENT MENU	LF > LF Sweep > Start Freq
EXAMPLE	*:LFOutput:SWEep:STARt:FREQuency 100* *:LFOutput:SWEep:STARt:FREQuency?* Return: *100\n*

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

SYNTAX	[:SOURce]:LFOutput:SWEep:STOP:FREQuency <freq> [:SOURce]:LFOutput:SWEep:STOP:FREQuency?
DESCRIPTION	This command sets/queries the stop frequency of LF sweep.
DATA TYPE	Float, unit: Hz, kHz, MHz or GHz, default is Hz
RANGE	LF wave is Sine: 0.01 Hz ~ 1 MHz, LF wave is Square/Triangle/Sawtooth: 0.01 Hz ~ 20 kHz.

RETURN	Float, unit: Hz
DEFAULT VALUE	1.5 kHz
EQUIVALENT MENU	LF > LF Sweep > Stop Freq
EXAMPLE	*:LFOutput:SWEep:STOP:FREQuency 1000* *:LFOutput:SWEep:STOP:FREQuency?* Return: *1000\n*

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

SYNTAX	[:SOURce]:LFOutput:SWEep:CENTer:FREQuency <freq> [:SOURce]:LFOutput:SWEep:CENTer:FREQuency?
DESCRIPTION	This command sets/queries the center frequency of LF sweep.
DATA TYPE	Float, unit: Hz, kHz, MHz or GHz, default is Hz
RANGE	LF wave is Sine: 0.01 Hz ~ 1 MHz, LF wave is Square/Triangle/Sawtooth: 0.01 Hz ~ 20 kHz.
RETURN	Float, unit: Hz
DEFAULT VALUE	1 kHz
EQUIVALENT MENU	LF > LF Sweep > Center Freq
EXAMPLE	*:LFOutput:SWEep:CENTer:FREQuency 550* *:LFOutput:SWEep:CENTer:FREQuency?* Return: *550\n*

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

SYNTAX	[:SOURce]:LFOutput:SWEep:SPAN:FREQuency <freq> [:SOURce]:LFOutput:SWEep:SPAN:FREQuency?
DESCRIPTION	This command sets/queries the frequency san of LF sweep.
DATA TYPE	Float, unit: Hz, kHz, MHz or GHz, default is Hz
RANGE	LF wave is Sine: 0.00 Hz ~ 999.99999 kHz, LF wave is Square/Triangle/Sawtooth: 0.00 Hz ~ 19.99999 kHz.
RETURN	Float, unit: Hz
DEFAULT VALUE	1 kHz
EQUIVALENT MENU	LF > LF Sweep > Freq Span
EXAMPLE	*:LFOutput:SWEep:SPAN:FREQuency 550* *:LFOutput:SWEep:SPAN:FREQuency?* Return: *550\n*

3.4.11.7 Sweep Time ([:SOURce]:LFOutput:SWEep:DWELl)

SYNTAX	[:SOURce]:LFOutput:SWEep:DWELl <time> [:SOURce]:LFOutput:SWEep:DWELl?
DESCRIPTION	This command sets/queries the sweep time of LF sweep.
DATA TYPE	Float, unit: ns, us, ms or s, default is s
RANGE	1 ms ~ 500 s
RETURN	Float, unit: s
DEFAULT VALUE	1 s
EQUIVALENT MENU	LF > LF Sweep > Sweep Time
EXAMPLE	*:LFOutput:SWEep:DWELl 2 s* *:LFOutput:SWEep:DWELl?* Return: *2\n*

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

SYNTAX	[:SOURce]:LFOutput:SWEep:TRIGger:TYPE AUTO|KEY|BUS|EXT [:SOURce]:LFOutput:SWEep:TRIGger:TYPE?
DESCRIPTION	This command sets/queries the sweep trigger mode of LF sweep.
DATA TYPE	Enumeration
RANGE	AUTO|KEY|BUS|EXT
RETURN	Enumeration
DEFAULT VALUE	AUTO
EQUIVALENT MENU	LF > LF Sweep > Trigger Mode
EXAMPLE	*:LFOutput:SWEep:TRIGger:TYPE KEY* *:LFOutput:SWEep:TRIGger:TYPE?* Return: *KEY\n*

3.4.11.9 Trigger Slope ([:SOURce]:LFOutput:SWEep:XPOLar)

SYNTAX	[:SOURce]:LFOutput:SWEep:XPOLar POS|NEG [:SOURce]:LFOutput:SWEep:XPOLar?
DESCRIPTION	This command sets/queries the trigger edge of the external trigger signal for the LF sweep function.
DATA TYPE	Enumeration
RANGE	POS|NEG
RETURN	Enumeration
DEFAULT VALUE	POS
EQUIVALENT MENU	LF > LF Sweep > Trigger Slope
EXAMPLE	*:LFOutput:SWEep:XPOLar POS*

	:LFOutput:SWEep:XPOLar? Return: *POS\n*

3.4.11.10 Sweep Shape ([:SOURce]:LFOutput:SWEep:SHAPe)

SYNTAX	[:SOURce]:LFOutput:SWEep:SHAPe TRIangle|SAWTooth [:SOURce]:LFOutput:SWEep:SHAPe?
DESCRIPTION	This command sets/queries the sweep shape of LF sweep.
DATA TYPE	Enumeration
RANGE	TRIangle|SAWTooth
RETURN	Enumeration
DEFAULT VALUE	SAWTooth
EQUIVALENT MENU	LF > LF Sweep > Sweep Shape
EXAMPLE	*:LFOutput:SWEep:SHAPe TRIangle* *:LFOutput:SWEep:SHAPe?* Return: *TRIangle\n*

3.4.11.11 Sweep Space ([:SOURce]:LFOutput:SWEep:SPACing)

SYNTAX	[:SOURce]:LFOutput:SWEep:SPACing LINear|LOGarithmic [:SOURce]:LFOutput:SWEep:SPACing?
DESCRIPTION	This command sets/queries the frequency sweep space of LF sweep.
DATA TYPE	Enumeration
RANGE	LINear|LOGarithmic
RETURN	Enumeration
DEFAULT VALUE	LINear
EQUIVALENT MENU	LF > LF Sweep > Sweep Space
EXAMPLE	*:LFOutput:SWEep:SPACing LOGarithmic* *:LFOutput:SWEep:SPACing?* Return: *LOGarithmic\n*

3.4.12 System Preset

3.4.12.1 Preset (:SOURce:PRESet)

SYNTAX	:SOURce:PRESet
DESCRIPTION	This command sets the instrument status to factory settings.
EQUIVALENT MENU	None
EXAMPLE	*:SOURce:PRESet*

3.5 SENSe Commands

3.5.1 Power Sensor

3.5.1.1 Sensor Info (:SENSe[:POWer]:TYPE?)

SYNTAX	:SENSe[:POWer]:TYPE?
DESCRIPTION	This command queries the model of the power sensor connected to the signal generator.
RETURN	String
DEFAULT VALUE	None
EQUIVALENT MENU	HOME > POWER SENSOR > Sensor Info
EXAMPLE	*SENSe:TYPE?* Return: *NRP6A\n*

3.5.1.2 Sensor State (:SENSe[:POWer]:STATus)

SYNTAX	:SENSe[:POWer]:STATus OFF|ON|0|1 :SENSe[:POWer]:STATus?
DESCRIPTION	This command sets/queries the measurement status of the power sensor.
DATA TYPE	Boolean
RANGE	ON|OFF|1|0
RETURN	1|0
DEFAULT VALUE	0
EQUIVALENT MENU	HOME > POWER SENSOR > Sensor State
EXAMPLE	*SENSe:STATus ON* *SENSe:STATus?* Return: *1\n*

3.5.1.3 Measurement (:SENSe[:POWer]:VALue?)

SYNTAX	:SENSe[:POWer]:VALue?
DESCRIPTION	This command queries the measured value of the power sensor connected to the signal generator.
RETURN	Float, unit: dBm
DEFAULT VALUE	None
EQUIVALENT MENU	HOME > POWER SENSOR > Measurement
EXAMPLE	*SENSe:VALue?* Return: *-0.02600282\n*

3.5.1.4 Statistics State (:SENSe[:POWer]:STATIStics:STATe)

SYNTAX	:SENSe[:POWer]:STATIStics:STATe ON|OFF|1|0 :SENSe[:POWer]:STATIStics:STATe?
DESCRIPTION	This command sets/queries the measurement statistics status of the power sensor.
DATA TYPE	Boolean
RANGE	ON|OFF|1|0
RETURN	1|0
DEFAULT VALUE	0
EQUIVALENT MENU	HOME > POWER SENSOR > Statistics
EXAMPLE	*SENSe:STATIStics:STATe ON* *SENSe:STATIStics:STATe?* Return: *1\n*

3.5.1.5 Statistics Value (:READ[:POWer]?)

SYNTAX	:READ[:POWer]?
DESCRIPTION	This command queries the average and maximum values of the power sensor measurement statistics.
RETURN	String The string format: average,maximum Average: float, unit: dBm, Maximum: float, unit: dBm.
DEFAULT VALUE	None
EQUIVALENT MENU	HOME > POWER SENSOR > Statistics > mean/max
EXAMPLE	*READ?* Return: *-0.05,-0.04\n*

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

SYNTAX	:SENSe[:POWer]:STATIStics:MAX?
DESCRIPTION	This command queries the maximum value of the power sensor measurement statistics.
RETURN	Float, unit: dBm.
DEFAULT VALUE	None
EQUIVALENT MENU	HOME > POWER SENSOR > Statistics > max
EXAMPLE	*SENSe:STATIStics:MAX?* Return: *-0.03117205\n*

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

SYNTAX	:SENSe[:POWer]:STATIStics:MIN?
DESCRIPTION	This command queries the minimum value of the power sensor measurement statistics.
RETURN	Float, unit: dBm.
DEFAULT VALUE	None
EQUIVALENT MENU	HOME > POWER SENSOR > Statistics > min
EXAMPLE	*SENSe:STATIStics:MIN?* Return: *-0.06101395\n*

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

SYNTAX	:SENSe[:POWer]:STATIStics:AVG?
DESCRIPTION	This command queries the average value of the power sensor measurement statistics.
RETURN	Float, unit: dBm.
DEFAULT VALUE	None
EQUIVALENT MENU	HOME > POWER SENSOR > Statistics > mean
EXAMPLE	*SENSe:STATIStics:AVG?* Return: *-0.0322136383619456\n*

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

SYNTAX	:SENSe[:POWer]:STATIStics:COUNt?
DESCRIPTION	This command queries the count of the power sensor measurement statistics.
RETURN	Integer
DEFAULT VALUE	None
EQUIVALENT MENU	HOME > POWER SENSOR > Statistics > count
EXAMPLE	*SENSe:STATIStics:COUNt?* Return: *2035\n*

3.5.1.10 Statistics Clear (:SENSe[:POWer]:STATIStics:CLEAr)

SYNTAX	:SENSe[:POWer]:STATIStics:CLEAr
DESCRIPTION	This command clears the measurement statistics of the power sensor.
EQUIVALENT MENU	HOME > POWER SENSOR > Statistics > clear

EXAMPLE	*SENSe:STATIStics:CLEAr*

3.5.1.11 Level Control (:SENSe[:POWer]:LEV:CTL:STATe)

SYNTAX	:SENSe[:POWer]:LEV:CTL:STATe ON|OFF|1|0 :SENSe[:POWer]:LEV:CTL:STATe?
DESCRIPTION	This command sets/queries the level control state of the power sensor.
DATA TYPE	Boolean
RANGE	ON|OFF|1|0
RETURN	1|0
DEFAULT VALUE	0
EQUIVALENT SCPI	[:SOURce]:POWer:SPC:STATe ON|OFF|1|0 [:SOURce]:POWer:SPC:STATe?
EQUIVALENT MENU	HOME > POWER SENSOR > Level Control
EXAMPLE	*:SENSe:LEV:CTL:STATe OFF* *:SENSe:LEV:CTL:STATe?* Return: *0\n*

3.5.1.12 Target Level (:SENSe[:POWer]:SPC:TARGet)

SYNTAX	:SENSe[:POWer]:SPC:TARGet <power> :SENSe[:POWer]:SPC:TARGet?
DESCRIPTION	This command sets/queries the targrt level of the power sensor level control.
DATA TYPE	Float, unit: dBm, dBuV, uV, mV, V, nW, uW, mW or W, default is dBm
RANGE	-120 dBm ~ 20 dBm
RETURN	Float, unit: dBm
DEFAULT VALUE	0
EQUIVALENT SCPI	[:SOURce]:POWer:SPC:TARGet <power> [:SOURce]:POWer:SPC:TARGet?
EQUIVALENT MENU	HOME > POWER SENSOR > Level Control > Target Level
EXAMPLE	*SENSe:SPC:TARGet -6* *SENSe:SPC:TARGet?* Return: *-6\n*

3.5.1.13 Limit Level (:SENSe[:POWer]:LIMit)

SYNTAX	:SENSe[:POWer]:LIMit <power> :SENSe[:POWer]:LIMit?

DESCRIPTION	This command sets/queries the Limit level of the power sensor level control.
DATA TYPE	Float, unit: dBm, dBuV, uV, mV, V, nW, uW, mW or W, default is dBm
RANGE	-120 dBm ~ 20 dBm
RETURN	Float, unit: dBm
DEFAULT VALUE	0
EQUIVALENT SCPI	[:SOURce]:POWer:LIMit <power> [:SOURce]:POWer:LIMit?
EQUIVALENT MENU	HOME > POWER SENSOR > Level Control > Limit Level
EXAMPLE	*SENSe:LIMit 2* *SENSe:LIMit?* Return: *2\n*

3.5.1.14 Catch Range (:SENSe[:POWer]:SPC:CRANge)

SYNTAX	:SENSe[:POWer]:SPC:CRANge <power> :SENSe[:POWer]:SPC:CRANge?
DESCRIPTION	This command sets/queries the catch range of the power sensor level control.
DATA TYPE	Float, unit: dB
RANGE	0 ~ 50
RETURN	Float, unit: dB
DEFAULT VALUE	0
EQUIVALENT SCPI	[SOURce]:POWer:SPC:CRANge <power> [SOURce]:POWer:SPC:CRANge?
EQUIVALENT MENU	HOME > POWER SENSOR > Level Control > Catch Range
EXAMPLE	*:SENSe:SPC:CRANge 10* *:SENSe:SPC:CRANge?* Return: *10\n*

3.5.1.15 Auto Zero (:CALibration:ZERO:TYPE)

SYNTAX	:CALibration:ZERO:TYPE INTernal|EXTernal :CALibration:ZERO:TYPE?
DESCRIPTION	This command sets/queries the automatic zero adjustment type of the power sensor.
DATA TYPE	Enumeration
RANGE	INTernal|EXTernal
RETURN	Enumeration
DEFAULT VALUE	INTernal

EQUIVALENT MENU	HOME > POWER SENSOR > Auto Zero
EXAMPLE	*:CALibration:ZERO:TYPE EXTernal* *:CALibration:ZERO:TYPE?* Return: *EXTernal\n*

3.5.1.16 Zeroing (:SENSe[:POWer]:ZERO)

SYNTAX	:SENSe[:POWer]:ZERO
DESCRIPTION	This command performs a zeroing operation on the power sensor.
EQUIVALENT MENU	HOME > POWER SENSOR > Click to perform zeroing
EXAMPLE	*:SENSe:ZERO*

3.5.1.17 Frequency Type (:SENSe[:POWer]:SOURce)

SYNTAX	:SENSe[:POWer]:SOURce RF|USER :SENSe[:POWer]:SOURce?
DESCRIPTION	This command sets/queries the measurement frequency type of the power sensor.
DATA TYPE	Enumeration
RANGE	RF|USER
RETURN	Enumeration
DEFAULT VALUE	RF
EQUIVALENT MENU	HOME > POWER SENSOR > Frequency
EXAMPLE	*SENSe:SOURce USER* *SENSe:SOURce?* Return: *USER\n*

3.5.1.18 Frequency (:SENSe[:POWer]:FREQuency)

SYNTAX	:SENSe[:POWer]:FREQuency <freq> :SENSe[:POWer]:FREQuency?
DESCRIPTION	This command sets/queries the manual measurement frequency of the power sensor.
DATA TYPE	Float, unit: Hz, kHz, MHz or GHz, default is Hz
RANGE	Power sensor measurement range
RETURN	Float, unit: Hz
DEFAULT VALUE	None
EQUIVALENT MENU	HOME > POWER SENSOR > Frequency
EXAMPLE	*SENSe:FREQuency 1 MHz* *SENSe:FREQuency?*

	Return: *1000000\n*

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

SYNTAX	:SENSe[:POWer]:OFFSet:STATe ON|OFF|1|0 :SENSe[:POWer]:OFFSet:STATe?
DESCRIPTION	This command sets/queries the level offset state of the power sensor.
DATA TYPE	Boolean
RANGE	ON|OFF|1|0
RETURN	1|0
DEFAULT VALUE	0
EQUIVALENT MENU	HOME > POWER SENSOR > Level Offset
EXAMPLE	*SENSe:OFFSet:STATe ON* *SENSe:OFFSet:STATe?* Return: *1\n*

3.5.1.20 Level Offset (:SENSe[:POWer]:OFFSet)

SYNTAX	:SENSe[:POWer]:OFFSet <power> :SENSe[:POWer]:OFFSet?
DESCRIPTION	This command sets/queries the level offset value of the power sensor.
DATA TYPE	Float, unit: dB
RANGE	-200 ~ 200
RETURN	Float, unit: dB
DEFAULT VALUE	0
EQUIVALENT MENU	HOME > POWER SENSOR > Level Offset
EXAMPLE	*SENSe:OFFSet 10* *SENSe:OFFSet?* Return: *10\n*

3.5.1.21 Average Type (:SENSe[:POWer]:FILTer:TYPE)

SYNTAX	:SENSe[:POWer]:FILTer:TYPE AUTO|USER|NSRatio :SENSe[:POWer]:FILTer:TYPE?
DESCRIPTION	This command sets/queries the measurement averaging type of the power sensor.
DATA TYPE	Enumeration
RANGE	AUTO|USER|NSRatio

RETURN	Enumeration
DEFAULT VALUE	AUTO
EQUIVALENT MENU	HOME > POWER SENSOR > Averaging
EXAMPLE	*SENSe:FILTer:TYPE USER* *SENSe:FILTer:TYPE?* Return: *USER\n*

3.5.1.22 Average Times (:SENSe[:POWer]:FILTer:LENGth)

SYNTAX	:SENSe[:POWer]:FILTer:LENGth <length> :SENSe[:POWer]:FILTer:LENGth?
DESCRIPTION	This command sets/queries the average number of measurements of the power sensor.
DATA TYPE	Integer
RANGE	1 ~ 65536
RETURN	Integer
DEFAULT VALUE	None
EQUIVALENT MENU	HOME > POWER SENSOR > Averaging Count
EXAMPLE	*SENSe:FILTer:LENGth 10* *SENSe:FILTer:LENGth?* Return: *10\n*

3.5.1.23 Internal Noise (:SENSe[:POWer]:FILTer:NSRatio)

SYNTAX	:SENSe[:POWer]:FILTer:NSRatio <noise> :SENSe[:POWer]:FILTer:NSRatio?
DESCRIPTION	When the measurement averaging type of the power sensor is fixed noise, this command sets/queries the fixed noise of the power sensor.
DATA TYPE	Float, unit: dB
RANGE	None
RETURN	Float, unit: dB
DEFAULT VALUE	None
EQUIVALENT MENU	HOME > POWER SENSOR > Internal Noise
EXAMPLE	*SENSe:FILTer:NSRatio 1* *SENSe:FILTer:NSRatio?* Return: *1\n*

3.5.1.24 Logging (:SENSe[:POWer]:LOGGing:STATe)

SYNTAX	:SENSe[:POWer]:LOGGing:STATe ON|OFF|1|0 :SENSe[:POWer]:LOGGing:STATe?
DESCRIPTION	This command sets/queries the logging status of the power sensor.
DATA TYPE	Boolean
RANGE	ON|OFF|1|0
RETURN	1|0
DEFAULT VALUE	0
EQUIVALENT MENU	HOME > POWER SENSOR > Logging
EXAMPLE	*SENSe:LOGGing:STATe ON* *SENSe:LOGGing:STATe?* Return: *1\n*

4 Programming Examples

This chapter provides some examples for programmers. In these examples you can see how to use VISA or Sockets and the above SCPI commands to control a signal generator. By following these examples, you can develop more applications.

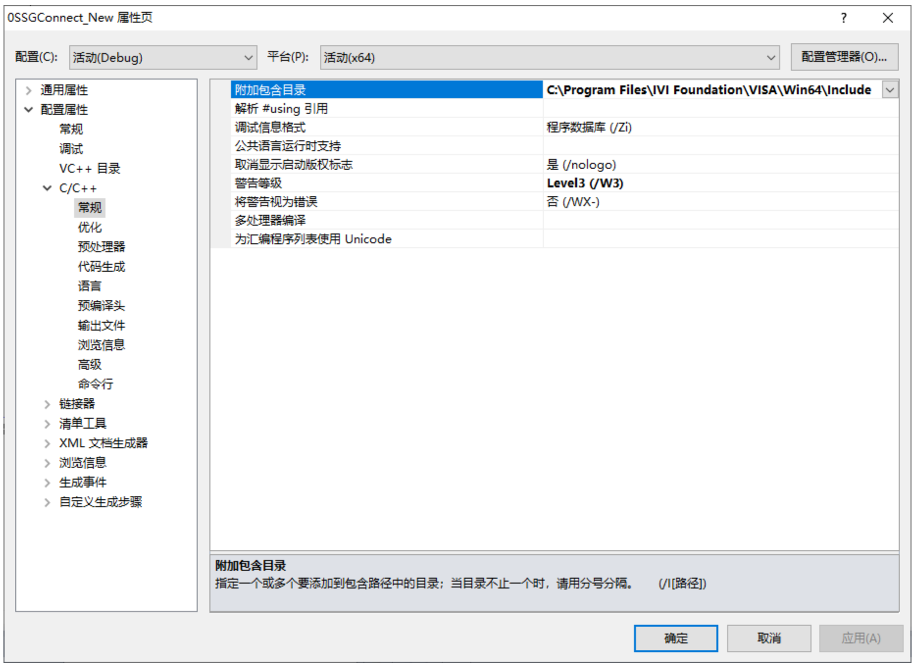
4.1 VISA Examples

4.1.1 VC++ Example

System environment: Windows 10, 64-bit operating system
Programming software: Visual Studio
Example content: Use NI-VISA to access and control the device through USBTMC or TCP/IP and perform read and write operations.

Please follow below steps to complete the example:

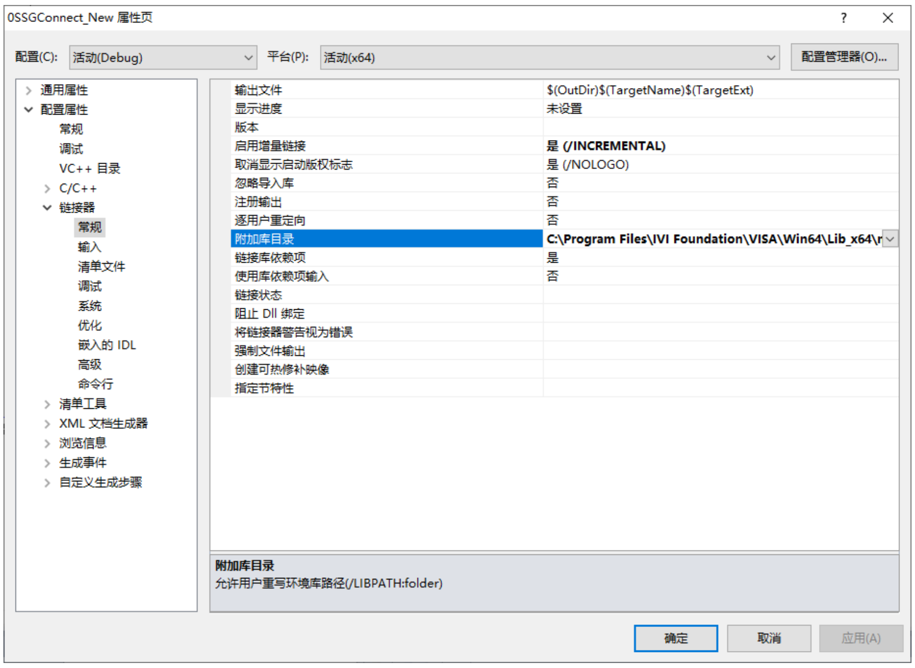
1. Open Visual Studio, and create a new VC++ win32 console project.
2. Set up the project environment to use the NI-VISA library. There are two ways to use the NI-VISA library, static or automatic:
   (1) Static:
   Find the files in the NI-VISA installation path: visa.h, visatype.h, visa32.lib. The default installation path of NI-VISA is “C:\Program Files\IVI Foundation\VISA\Win64\”. Copy the above files into your project and add them to the project. Then add the following two lines of code in the project.cpp file:
   ```
   #include "visa.h"
   #pragma comment(lib,"visa32.lib")
   ```
   (2) Automatic:
   Set the include directory for header files. The default installation path of NI-VISA header files is “C:\Program Files\IVI Foundation\VISA\Win64\Include”. Fill in the header file installation path in Project --- Properties --- C/C++ --- General --- Additional Include Directories, as shown below:



附加包含目录	C:\Program Files\IVI Foundation\VISA\Win64\Include
解析 #using 引用	
调试信息格式	程序数据库 (/Zi)
公共语言运行时支持	
取消显示启动版权标志	是 (/nologo)
警告等级	Level3 (/W3)
将警告视为错误	否 (/WX-)
多处理器编译	
为汇编程序列表使用 Unicode	

附加包含目录
指定一个或多个要添加到包含路径中的目录；当目录不止一个时，请用分号分隔。 (/I[路径])

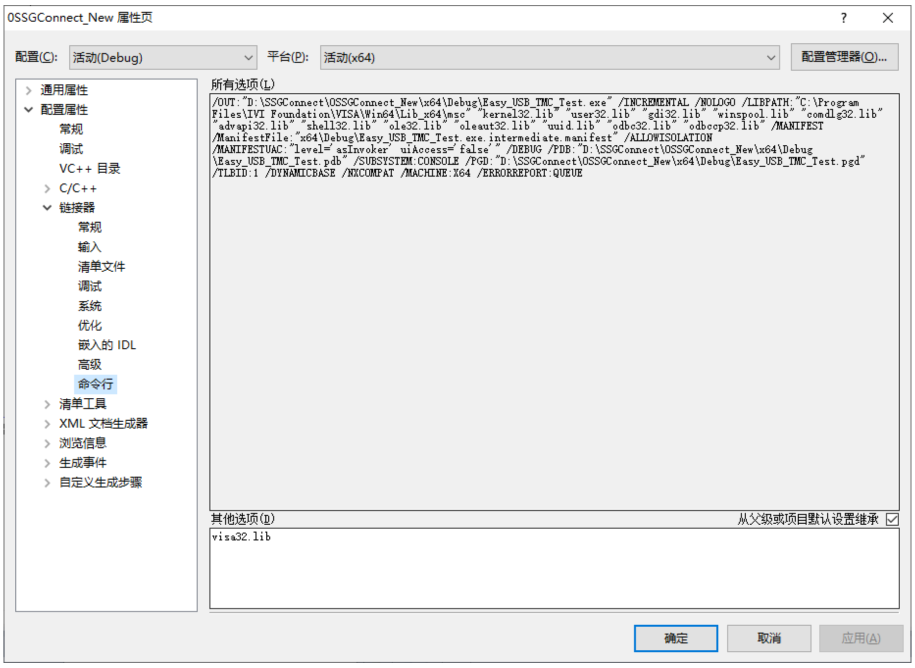
Set the include directory of library files. The default installation path of NI-VISA library files is “C:\Program Files\IVI Foundation\VISA\Win64\Lib_x64\msc”. Fill in the library file installation path in Project --- Properties --- Linker --- General --- Additional library directory, as shown below:



输出文件	$(OutDir)$(TargetName)$(TargetExt)
显示进度	未设置
版本	
启用增量链接	是 (/INCREMENTAL)
取消显示启动版权标志	是 (/NOLOGO)
忽略导入库	否
注册输出	否
逐用户重定向	否
附加库目录	C:\Program Files\IVI Foundation\VISA\Win64\Lib_x64\r
链接库依赖项	是
使用库依赖项输入	否
链接状态	
阻止 Dll 绑定	
将链接器警告视为错误	
强制文件输出	
创建可热修补映像	
指定节特性	

附加库目录
允许用户重写环境库路径(/LIBPATH:folder)

Set up library files. Fill in the library file name in Project --- Properties --- Linker --- Command Line --- Additional Options: visa32.lib, as shown below:



```
/OUT:"D:\SSGConnect\OSSGConnect_New\x64\Debug\Easy_USB_TMC_Test.exe" /INCREMENTAL /NOLOGO /LIBPATH:"C:\Program
Files\IVI Foundation\VISA\Win64\Lib_x64\msc" "kernel32.lib" "user32.lib" "gdi32.lib" "winspool.lib" "comdlg32.lib"
"advapi32.lib" "shell32.lib" "ole32.lib" "oleaut32.lib" "uuid.lib" "odbc32.lib" "odbccp32.lib" /MANIFEST
/ManifestFile:"x64\Debug\Easy_USB_TMC_Test.exe.intermediate.manifest" /ALLOWISOLATION
/MANIFESTUAC:"level='asInvoker' uiAccess='false'" /DEBUG /PDB:"D:\SSGConnect\OSSGConnect_New\x64\Debug
\Easy_USB_TMC_Test.pdb" /SUBSYSTEM:CONSOLE /PGD:"D:\SSGConnect\OSSGConnect_New\x64\Debug\Easy_USB_TMC_Test.pgd"
/TLBID:1 /DYNAMICBASE /NXCOMPAT /MACHINE:X64 /ERRORREPORT:QUEUE
```

```
visa32.lib
```

Finally, reference the header file in the project .cpp file:

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

3. Add code

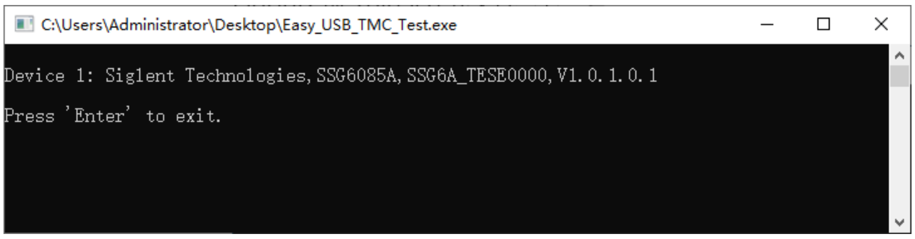
(1) USBTMC access code

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;
}
```

The running result:



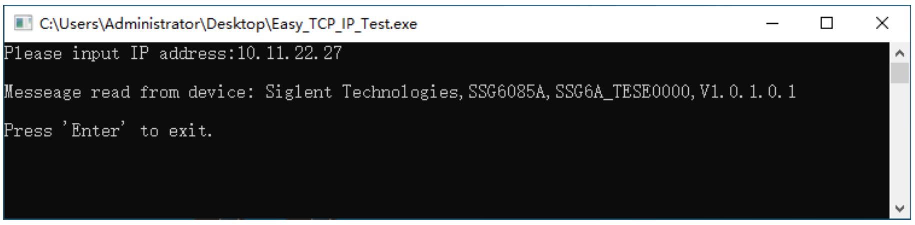
```
Device 1: Siglent Technologies,SSG6085A,SSG6A_TESE0000,V1.0.1.0.1
Press 'Enter' to exit.
```

(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");
        if (status<VI_SUCCESS)
        {
                printf("Error writing to the device.\n");
                viClose(defaultRM);
        }
        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;
}
```

The running result:



```
C:\Users\Administrator\Desktop\Easy_TCP_IP_Test.exe
Please input IP address:10.11.22.27

Messeage read from device: Siglent Technologies,SSG6085A,SSG6A_TESE0000,V1.0.1.0.1

Press 'Enter' to exit.
```

4.1.2 VB Example

System environment: Windows 7

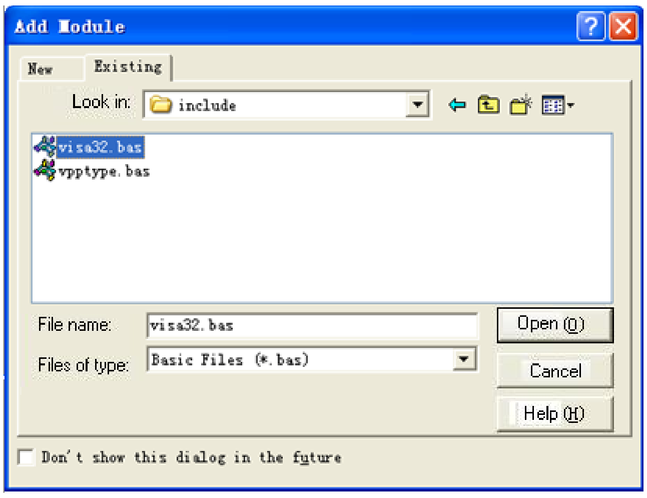
Programming software: Microsoft Visual Basic 6.0

Example content: Use NI-VISA to access and control the device through USBTMC or TCP/IP and perform read and write operations.

Please follow below steps to complete the example:

1. Open Visual Basic and build a standard application program project (standard EXE).
2. Set up the project environment to use the NI-VISA library. Click the Existing tag 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 will enable VISA functions and VISA data types to be used in the

program.



visa32.bas
vpptype.bas

File name: visa32.bas

Files of type: Basic Files (*.bas)

3. Add code

(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 numlnstrs 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:

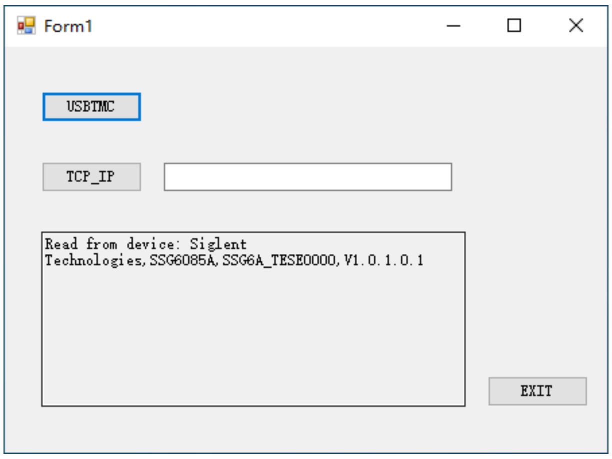
```
Private Function TCP_IP_Test(ByVal ip As String) As Long
    Dim outputBuffer As String * VI_FIND_BUFLEN
    Dim defaultRM As Long
    Dim instrsesn As Long
    Dim status As Long
    Dim count As Long

    ' 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
```

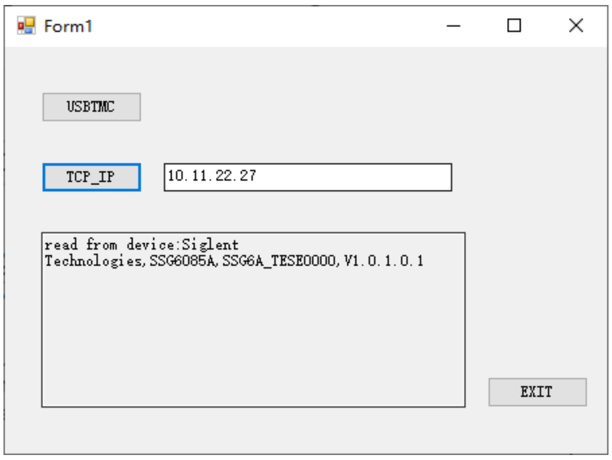
(3) Button control code:

```
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
```

The running result:



Read from device: Siglent Technologies,SSG6085A,SSG6A_TESE0000,V1.0.1.0.1



10.11.22.27

read from device:Siglent Technologies,SSG6085A,SSG6A_TESE0000,V1.0.1.0.1

4.1.3 MATLAB Example

System environment: Windows 7

Programming software: MATLAB R2013a

Example content: Use NI-VISA to access and control the device through USBTMC or TCP/IP and perform read and write operations.

Please follow below steps to complete the example:

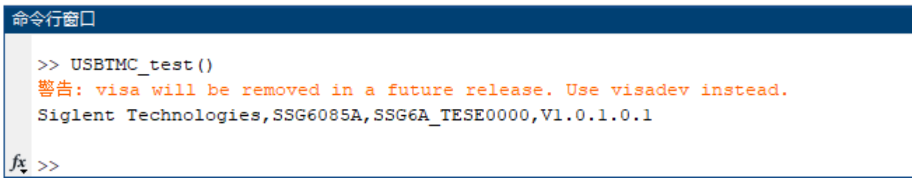
1. Open MATLAB and modify the current directory. In this example, change the current directory to D:\USBTMC_TCPIP_Demo.
2. Click File>>New>>Script in the MATLAB interface to create an empty M document.
3. Add code

   (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::0x1504::SSG6A_TESE0000::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
```

The running result:



```
>> USBTMC_test()
警告: visa will be removed in a future release. Use visadev instead.
Siglent Technologies,SSG6085A,SSG6A_TESE0000,V1.0.1.0.1

>>
```

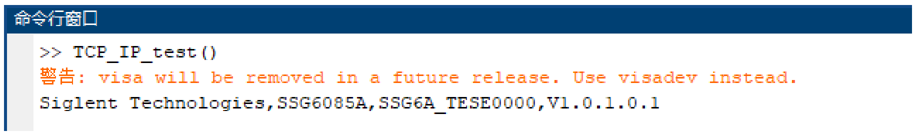
(2) TCP/IP access code

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::','10.11.22.27','::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
```

The running result:



命令行窗口

```
>> TCP_IP_test()
警告: visa will be removed in a future release. Use visadev instead.
Siglent Technologies,SSG6085A,SSG6A_TESE0000,V1.0.1.0.1
```

4.1.4 LabVIEW Example

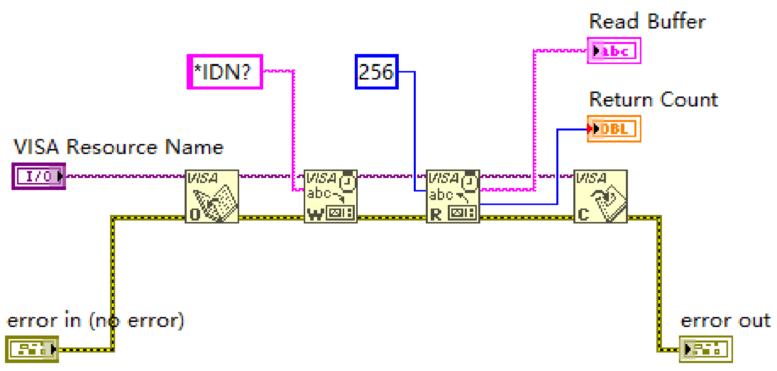
System environment: Windows 7

Programming software: LabVIEW 2011

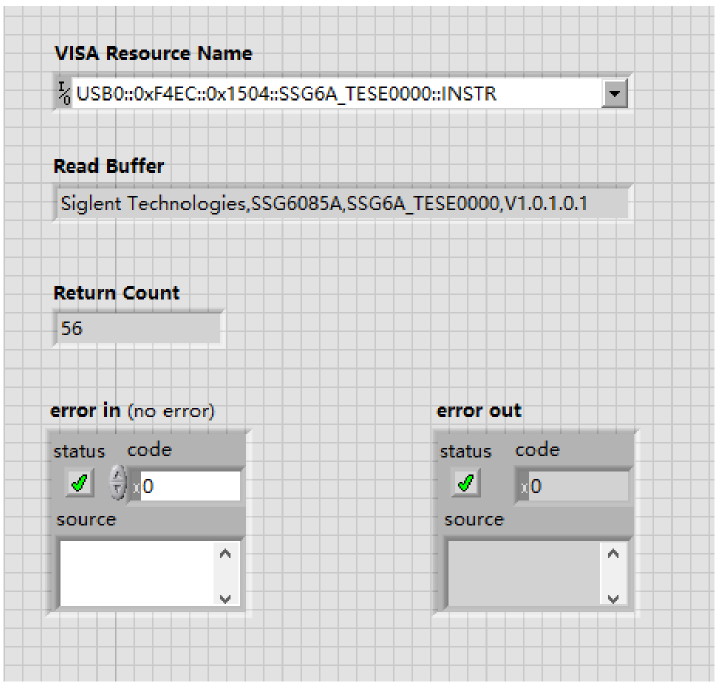
Example content: Use NI-VISA to access and control the device through USBTMC or TCP/IP and perform read and write operations.

Please follow below steps to complete the example:

1. Open LabVIEW and create a VI file.
2. Add controls. Right-click the front panel interface, select and add VISA resource name, error input, error out and some indicators in the controls column.
3. Open the block diagram interface. Right-click the VISA resource name and select from the shortcut menu of the VISA palette to add the following functions: 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.



VISA Resource Name

USB0::0xF4EC::0x1504::SSG6A_TESE0000::INSTR

Read Buffer

Siglent Technologies,SSG6085A,SSG6A_TESE0000,V1.0.1.0.1

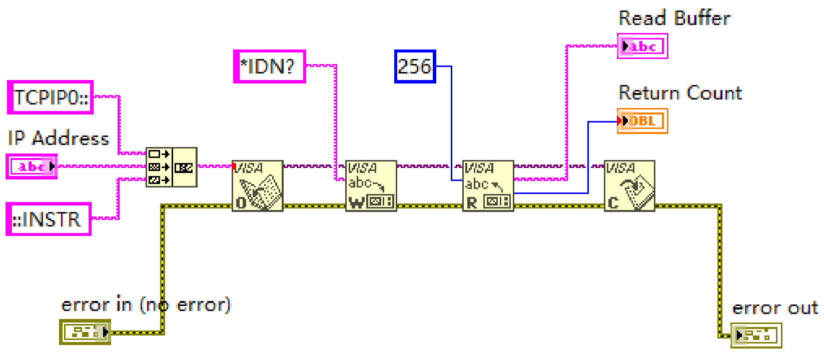
Return Count

56

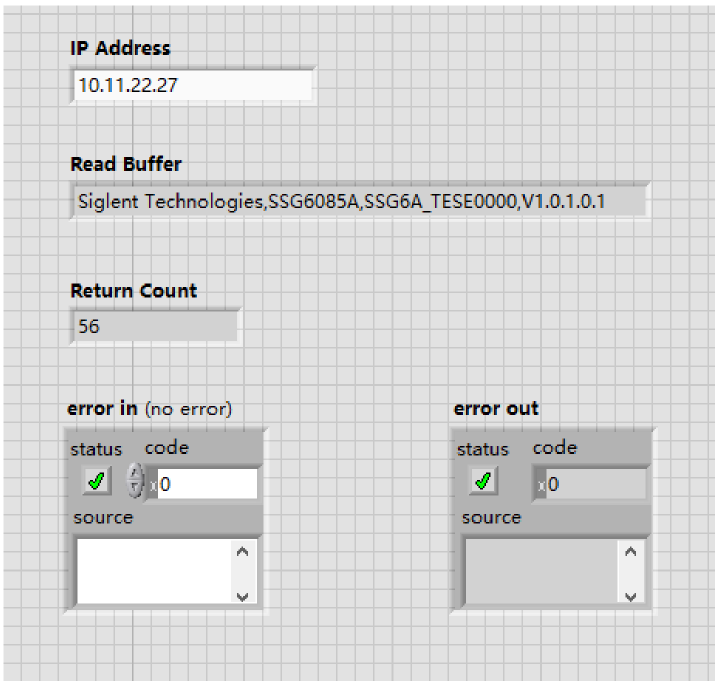
In this example, the VI opens a VISA session to the USBTMC device, writes a command to the device, and then reads back the response. In this example, the specific command sent is a device ID query. Please check the device command set with your device manufacturer. After all communication is complete,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

Communicating with the device over TCP/IP is similar to USBTMC. However, you need to change the VISA write and VISA read functions to synchronize the I/O. LabVIEW’s default is asynchronous I/O. Right-click the node and select Synchronous I/O Mode>>Sync from the shortcut menu to write or read synchronized data.

1. Connect them as shown in the figure below:



2. Enter the IP address and run the program:



IP Address
10.11.22.27

Read Buffer
Siglent Technologies,SSG6085A,SSG6A_TESE0000,V1.0.1.0.1

Return Count
56

error in (no error)
status code
✓ 0
source

error out
status code
✓ 0
source

4.2 Socket Examples

The Windows operating system itself supports Sockets communication, and this communication method is also relatively simple. It should be noted that "\n" (newline character) needs to be added to the end of the SCPI command string.

4.2.1 Python Example

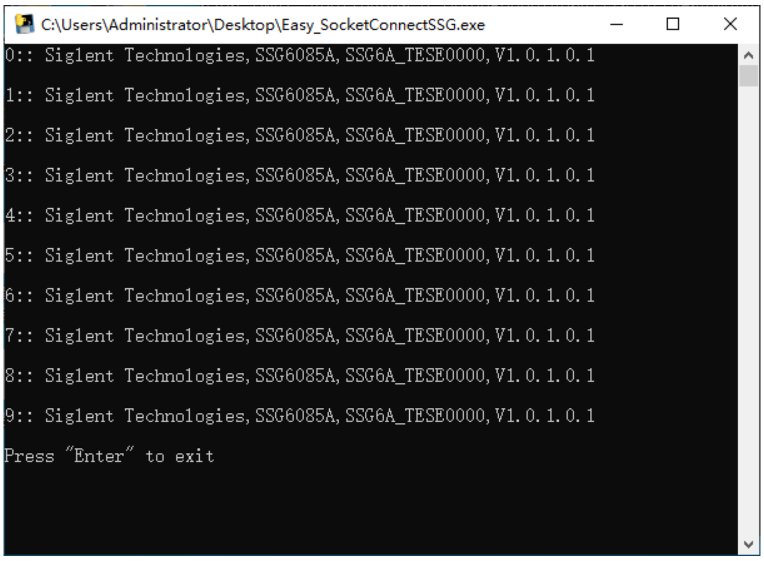
Python is an interpreted programming language that allows you to work quickly and is very portable. Python has an underlying network module that provides access to the Socket interface, port 5025. Python scripts can be written for the Socket interface to perform various test and measurement tasks.

System environment: Windows 10, 64-bit operating system
Programming software: Python v3.6.5
Example content: Open a Socket, send a SCPI query, then close the Socket, loop ten times.

Below is the code of the script:

```
#!/usr/bin/env python
#-*- coding:utf-8 –*-
#-----------------------------------------------------------------------------
# The short script is an 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 = "10.11.22.27"    # should match the instrument’s IP address
port = 5025 # the port number of the instrument service

def SocketConnect():
    try:
        #create an AF_INET, STREAM socket (TCP)
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    except socket.error:
        input ('Failed to create socket. \nPress "Enter" to exit: ')
        sys.exit()
    try:
        #Connect to remote server
        s.connect((remote_ip , port))
    except socket.error:
        input('Failed to connect to ip %s!\nPress "Enter" to exit: ' % remote_ip)
        s.close()
        sys.exit()
    return s

def SocketQuery(Sock, cmd):
    try :
        #Send cmd string
        Sock.sendall(cmd)
        time.sleep(1)
    except socket.error:
        #Send failed
        input('Send failed!\nPress "Enter" to exit: ')
        SocketClose(Sock)
        sys.exit()
    reply = Sock.recv(4096)
    reply = reply.decode()
    return reply

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

def main():
    # Body: send the SCPI commands *IDN? 10 times and print the return message
    s = SocketConnect()

    count = 0
    for i in range(10):
        qStr = SocketQuery(s, b'*IDN?\n')
        print (str(count) + ":: " + qStr)
        count = count + 1
    SocketClose(s)
    input('Press "Enter" to exit')

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

The running result:



```
0:: Siglent Technologies,SSG6085A,SSG6A_TESE0000,V1.0.1.0.1
1:: Siglent Technologies,SSG6085A,SSG6A_TESE0000,V1.0.1.0.1
2:: Siglent Technologies,SSG6085A,SSG6A_TESE0000,V1.0.1.0.1
3:: Siglent Technologies,SSG6085A,SSG6A_TESE0000,V1.0.1.0.1
4:: Siglent Technologies,SSG6085A,SSG6A_TESE0000,V1.0.1.0.1
5:: Siglent Technologies,SSG6085A,SSG6A_TESE0000,V1.0.1.0.1
6:: Siglent Technologies,SSG6085A,SSG6A_TESE0000,V1.0.1.0.1
7:: Siglent Technologies,SSG6085A,SSG6A_TESE0000,V1.0.1.0.1
8:: Siglent Technologies,SSG6085A,SSG6A_TESE0000,V1.0.1.0.1
9:: Siglent Technologies,SSG6085A,SSG6A_TESE0000,V1.0.1.0.1
Press "Enter" to exit
```

About SIGLENT

SIGLENT is an international high-tech company, concentrating on R&D, sales, production and services of electronic test & measurement instruments.

SIGLENT first began developing digital oscilloscopes independently in 2002. After more than a decade of continuous development, SIGLENT has extended its product line to include digital oscilloscopes, isolated handheld oscilloscopes, function/arbitrary waveform generators, RF/MW signal generators, spectrum analyzers, vector network analyzers, digital multimeters, DC power supplies, electronic loads and other general purpose test instrumentation. Since its first oscilloscope was launched in 2005, SIGLENT has become the fastest growing manufacturer of digital oscilloscopes. We firmly believe that today SIGLENT is the best value in electronic test & measurement.

Headquarters:
SIGLENT Technologies Co., Ltd
Add: Bldg No.4 & No.5, Antongda Industrial Zone, 3rd Liuxian Road, Bao'an District, Shenzhen, 518101, China
Tel: + 86 755 3688 7876
Fax: + 86 755 3359 1582
Email: sales@siglent.com
Website: int.siglent.com

North America:
SIGLENT Technologies America, Inc
6557 Cochran Rd Solon, Ohio 44139
Tel: 440-398-5800
Toll Free: 877-515-5551
Fax: 440-399-1211
Email: info@siglentna.com
Website: www.siglentna.com

Europe:
SIGLENT Technologies Germany GmbH
Add: Staetzlinger Str. 70
86165 Augsburg, Germany
Tel: +49(0)-821-666 0 111 0
Fax: +49(0)-821-666 0 111 22
Email: info-eu@siglent.com
Website: www.siglenteu.com

Follow us on Facebook: SiglentTech