ZH

RIGOL

PNOISE Mode

For RSA6000 Series Spectrum Analyzer

Programming Guide

Jan. 2026

Guaranty and Declaration

Trademark Information
RIGOL® is the trademark of RIGOL TECHNOLOGIES CO., LTD.

Notices
- **RIGOL** products are covered by P.R.C. and foreign patents, issued and pending.
- **RIGOL** reserves the right to modify or change parts of or all the specifications and pricing policies at the company's sole decision.
- Information in this publication replaces all previously released materials.
- Information in this publication is subject to change without notice.
- **RIGOL** shall not be liable for either incidental or consequential losses in connection with the furnishing, use, or performance of this manual, as well as any information contained.
- Any part of this document is forbidden to be copied, photocopied, or rearranged without prior written approval of **RIGOL**.

Product Certification
RIGOL guarantees that this product conforms to the national and industrial standards in China as well as the ISO9001:2015 standard and the ISO14001:2015 standard. Others international standard conformance certifications are in progress

Contact Us
If you have any problem or requirement when using our products or this manual, please contact **RIGOL**.
E-mail: service@rigol.com
Website: *http://www.rigol.com*

Chapter 1 Document Overview

This manual introduces how to program and control **RIGOL** RSA6000 series spectrum analyzer (PNOISE mode) by using SCPI commands through the USB or LAN interface.

Tip

For the latest version of this manual, download it from the official website of RIGOL (*http://www.rigol.com*).

Publication Number

PGD31100-1110

Software Version

00.00.03

Software upgrade might change or add product features. Please acquire the latest version of the manual from RIGOL website or contact RIGOL to upgrade the software.

Format Conventions in this Manual

1. Key

The front panel key is denoted by the menu key icon. For example, 

indicates the **System** key.

2. Menu

The menu item is denoted by the format of "Menu Name (Bold) + Character Shading" in the manual. For example, "Setup" indicates clicking or tapping the "Setup" sub-menu under the "System" menu to view the basic configuration items.

3. Operation Procedures

The next step of the operation is denoted by an arrow ">" in the manual. For example,  > **Save** indicates that first clicking or tapping the icon  , then clicking or tapping **Save**.

4. Connector

The connectors on the front or rear panel are usually denoted by the format of "Connector Name (Bold) + Square Brackets (Bold)". For example, **[TRIG IN]**.

Content Conventions in this Manual

The RSA6000 series spectrum analyzer includes the following models. Unless otherwise specified, this manual takes RSA6265 as an example to illustrate the functions and operation methods of RSA5065 series spectrum analyzer.

Model	Frequency Range
RSA6265	5 kHz to 26.5 GHz
RSA6140	5 kHz to 14 GHz
RSA6085	5 kHz to 8.5 GHz

Contents

Chapter 2 Programming Overview

This chapter introduces how to set up remote communication between the spectrum analyzer and the PC, the remote control methods, the syntax, symbols, parameters, and abbreviation rules of the SCPI commands.

SCPI Command Overview

The remote control can be realized by using SCPI (Standard Commands for Programmable Instruments) commands. SCPI (Standard Commands for Programmable Instruments) is a standardized instrument programming language that is built upon the existing standard IEEE 488.1 and IEEE 488.2 and conforms to various standards, such as the floating point operation rule in IEEE 754 standard, ISO 646 7-bit coded character set for information interchange (equivalent to ASCII programming). The SCPI commands provide a hierarchical tree structure, and consist of multiple subsystems. Each command subsystem consists of one root keyword and one or more sub-keywords.

Syntax

The command line usually starts with a colon; the keywords are separated by colons, and following the keywords are the parameter settings available. The command ending with a quotation mark indicates querying a certain function and returns the query results. The keywords of the command and the first parameter are separated by a space.

For example,

:SYSTem:BEEPer <bool>

:SYSTem:BEEPer?

SYSTem is the root keyword of the command. BEEPer is the second-level keywords. The command line starts with a colon, and different levels of keywords are also separated by colons. <bool> indicates a settable parameter. The command ending with a quotation mark indicates querying a certain function. :SYSTem:BEEPer and the parameter <bool> are separated by a space.

In some commands with parameters, "," is often used to separate multiple parameters. For example,

:SYSTem:DATE <year>,<month>,<day>.

Symbol Description

The following symbols are not sent with the commands.

1. Braces { }

The contents in the braces can contain one or multiple parameters. These parameters can be omitted or used for several times. Parameters are usually separated by the vertical bar "|". When using the command, you must select one of the parameters.

2. Vertical Bar |

The vertical bar is used to separate multiple parameters. When using the command, you must select one of the parameters.

3. Square Brackets [ ]

The contents in the square brackets can be omitted.

4. Angle Brackets < >

The parameter enclosed in the angle brackets must be replaced by an effective value.

Parameter Type

1. Bool

The parameter can be set to ON, OFF, 1, or 0. For example,
:SYSTem:BEEPer <bool>
:SYSTem:BEEPer?
Wherein, <bool> can be set to {{1|ON}|{0|OFF}}. The query returns 1 or 0.

2. Discrete

The parameter can be any of the values listed. For example,
:SYSTem:PSTatus <status>
:SYSTem:PSTatus?
Wherein,
<status> can be set to DEFault|OPEN.
The query returns DEF or OPEN.

3. Integer

Unless otherwise specified, the parameter can be any integer (NR1 format) within the effective value range.
Note:
Do not set the parameter to a decimal, otherwise, errors will occur.

For example,
:SYSTem:GPIB <adr>
:SYSTem:GPIB?
Wherein, <adr> can be set to an integer ranging from 1 to 30. The query returns an integer ranging from 1 to 30.

4. Real

The parameter can be any real number within the effective value range, and this command accepts parameter input in decimal (NR2 format) and scientific notation (NR3 format). For example,
[:SENSe]:LPLot:SMOoth <percent>
[:SENSe]:LPLot:SMOoth?
Wherein, <percent> can be set to a real number ranging from 1% to 50%. The query returns the smoothing value in scientific notation.

5. ASCII String

The parameter can be the combinations of ASCII characters. For example,
In the command :CALCulate:LPLot:LLINe<n>:DESCription <string>, the parameter <string> can be set to
"Test Limit"

Command Abbreviation

All of the keywords of the commands are case-insensitive. They can all be in upper case or in lower case. If abbreviation is used, you must input all the capital letters in the command. For example,
[:SENSe]:LPLot:DETector?
can be abbreviated to
:SENS:LPL:DET?

Remote Control

The instrument can communicate with the PC via the USB interface and LAN interface to realize remote control of the instrument by using the SCPI (Standard Commands for Programmable Instruments) commands.

PC Software

Users usually need to use the PC software to send commands to control the instrument remotely. RIGOL Sigma is recommended. When the instrument is connected to the PC via the USB interface and USB, Ultra Sigma (only supported for Win 10 operating system and below) can be used to search the instrument resource and send the command.

Log in to the RIGOL official website. Click SUPPORT CENTER and select SOFTWARE and FIRMWARE to obtain the Ultra Sigma software package and help documentation (https://www.rigol.com/intl/support/firmware-software.html).

Web Control

When the instrument is connected to the PC via the LAN interface, you can use Web Control to send SCPI commands from the PC to the instrument. The operation procedures are as follows:

1. Obtain the instrument's IP address and input it in the browser address bar to log in to the Web Control page.
2. After you enter the Web Control interface, click the "SCPI Panel Control" button to enter the SCPI Command interface.
3. Input the specified SCPI command and then click **Send & Read** to send the command. The operation process and the returned value will be displayed in the current interface.

Remote Control via USB

1. **Connect the device**

   Use the USB cable to connect the rear-panel USB DEVICE interface of the instrument to the USB HOST interface of the PC.

2. **Search for the device resource**

   Start up Ultra Sigma and the software will automatically search for the resource currently connected to the PC via the USB interface. You can also click **USB-TMC** to search for the resource.

3. **View the device resource**

   The resources found will appear under the "RIGOL Online Resource" directory, and the model number and USB interface information of the instrument will also be displayed.

4. **Control the instrument remotely**

   Right-click the resource name and select "SCPI Panel Control" to open the remote command control panel. Then you can send commands and read data through the panel. For details about the SCPI commands and programming, refer to the Programming Guide of this instrument.

Remote Control via LAN

1. **Connect the load to the PC**

   Use the network cable to connect the instrument to your local area network (LAN).

2. **Configure network parameters**

   In **System** > **I/O** menu, configure the network parameters for the instrument.

3. **Search for the device resource**

   Start up Ultra Sigma and click **LAN** to open the panel. Click **Search** and the software searches for the instrument resources currently connected and the resources found are displayed at the right section of the window. Click **OK** to add it. Besides, you can input the IP address of the instrument manually into the text field under "Manual Input LAN Instrument IP", then click **TEST**. If the instrument passes the test, click **ADD** to add the instrument to the LAN instrument resource list in the right section; if the instrument fails the test, please check whether the IP address that you input is correct, or use the auto search method to add the instrument resource.

4. **View the device resource**

   The resources found are shown in the "RIGOL Online Resource" directory.

5. **Control the instrument remotely**

   Right-click the resource name and select "SCPI Panel Control" to open the remote command control panel. Then you can send commands and read data through the panel.

6. **Load LXI webpage**

   This instrument conforms to LXI CORE 2011 DEVICE standards, you can load LXI webpage through Ultra Sigma (right-click the instrument resource name and select "LXI-Web"). Various important information about the instrument (including the model, manufacturer, serial number, description, MAC address, and IP address) will be displayed on the webpage. You can also directly input the IP address of the instrument in the address bar of the PC browser to load the LXI webpage.

Chapter 3 Command System

This chapter introduces the commands of the RSA6000 series spectrum analyzer in PNOISE mode.

Remarks:

1. The commands concerning the phase noise measurement are only available for the RSA6000 model installed with the RSA6000-PNM option. For details, refer to remarks for each command subsystem.
2. For the command set, unless otherwise specified, the query command returns "N/A" (without quotations in its return format) if no specified option is installed. If the queried function is disabled or improper type match is found, the query command will return "Error" (without quotations in its return format).
3. This manual takes RSA6265 as an example to illustrate the range of the parameters in each command.

:CALCulate Commands

The :CALCulate commands mainly cover the functions of marker, trace, limit line, and spurious search.

:CALCulate:LLINe:SELect

Syntax
:CALCulate:LLINe:SELect <limit>
:CALCulate:LLINe:SELect?

Description
Sets or queries the selected limit line.

Parameter

Name	Type	Range	Default
<limit>	Discrete	LLINe1|LLINe2|LLINe3|LLINe4|LLINe5|LLINe6	——

Return Format
The query returns the currently selected limit line.

Example
:CALC:LLIN:SELect LLIN2 /*Selects Limit Line 2.*/
:CALC:LLIN:SELect? /*The query returns LLINe2.*/

:CALCulate:LLINe<n>:TYPE

Syntax
:CALCulate:LLINe<n>:TYPE <type>
:CALCulate:LLINe<n>:TYPE?

Description
Sets or queries the type of the specified limit line.

Parameter

Name	Type	Range	Default
<n>	Discrete	1|2|3|4|5|6	1
<type>	Discrete	UPPer|LOWer	Refer to "**Remarks**"

Remarks
Limit Line 1, Limit Line 3, and Limit Line 5 belong to the upper type; Limit Line 2, Limit Line 4, and Limit Line 6 belong to the lower type.

Return Format
The query returns UPP or LOW.

Example
:CALCulate:LPLot:LLINe2:TYPE UPPer /*Sets the type of Limit Line 2 to Upper.*/
:CALCulate:LPLot:LLINe2:TYPE? /*The query returns UPP.*/

:CALCulate:LPLot:LLINe:ALL:DELete

Syntax
:CALCulate:LPLot:LLINe:ALL:DELete

Description
Deletes all the limit line data.

:CALCulate:LPLot:LLINe:TEST[:STATe]

Syntax
:CALCulate:LPLot:LLINe:TEST[:STATe] <bool>
:CALCulate:LPLot:LLINe:TEST[:STATe]?

Description
Sets or queries whether the limit line test is enable or not.

Parameter

Name	Type	Range	Default
<bool>	Bool	OFF|ON|0|1	OFF|0

Return Format
The query returns 0 or 1.

Example
:CALCulate:LPLot:LLINe:TEST:STATe ON or :CALCulate:LPLot:LLINe:TEST:STATe 1
/*Enables the limit line test.*/
:CALCulate:LPLot:LLINe:TEST:STATe? /*The query returns 1.*/

:CALCulate:LPLot:LLINe<n>:AMPLitude:INTerpolate:TYPE

Syntax
:CALCulate:LPLot:LLINe<n>:AMPLitude:INTerpolate:TYPE <type>
:CALCulate:LPLot:LLINe<n>:AMPLitude:INTerpolate:TYPE?

Description
Sets or queries the amplitude interpolation type of the specified limit line.

Parameter

Name	Type	Range	Default
<n>	Discrete	1|2|3|4|5|6	1

<type>	Discrete	LINear|LOG	LOG

Remarks
LINear: linear type.
LOG: log type.

Return Format
The query returns LIN or LOG.

Example
:CALCulate:LPLot:LLINe2:AMPLitude:INTerpolate:TYPE LOG /*Sets the amplitude interpolation type of the limit line to LOG.*/
:CALCulate:LPLot:LLINe2:AMPLitude:INTerpolate:TYPE? /*The query returns LOG.*/

:CALCulate:LPLot:LLINe<n>:CONTrol:INTerpolate:TYPE

Syntax
:CALCulate:LPLot:LLINe<n>:CONTrol:INTerpolate:TYPE <type>
:CALCulate:LPLot:LLINe<n>:CONTrol:INTerpolate:TYPE?

Description
Sets or queries the type of the frequency interpolation type of the specified limit line.

Parameter

Name	Type	Range	Default
<n>	Discrete	1|2|3|4|5|6	1
<type>	Discrete	LINear|LOG	LOG

Remarks
LINear: linear type.
LOG: log type.

Return Format
The query returns LIN or LOG.

Example
:CALCulate:LPLot:LLINe2:CONTrol:INTerpolate:TYPE LINear
/*Sets the frequency interpolation type of the limit line to LINear.*/
:CALCulate:LPLot:LLINe2:CONTrol:INTerpolate:TYPE?/*The query returns LIN.*/

:CALCulate:LPLot:LLINe<n>:COPY

Syntax
:CALCulate:LPLot:LLINe<n>:COPY <copy>

Description
Copies the selected limit line to the current limit line.

Parameter

Name	Type	Range	Default
<n>	Discrete	1|2|3|4|5|6	1
<copy>	Discrete	LLINe1|LLINe2|LLINe3|LLINe4|LLINe5|LLINe6	——

Remarks

If the limit line to be copied that you select is the same as the current limit line, no operation should be performed.

Example

:CALCulate:LPLot:LLINe2:COPY LLINe1 /*Copies Limit Line 1 to Limit Line 2.*/

:CALCulate:LPLot:LLINe<n>:DISPlay

Syntax

:CALCulate:LPLot:LLINe<n>:DISPlay <bool>
:CALCulate:LPLot:LLINe<n>:DISPlay?

Description

Sets or queries whether to display the specified limit line.

Parameter

Name	Type	Range	Default
<n>	Discrete	1|2|3|4|5|6	1
<bool>	Bool	ON|OFF|1|0	OFF|0

Return Format

The query returns 1 or 0.

Example

:CALC:LPL:LLIN1:DISP ON /*Displays Limit Line 1.*/
:CALC:LPL:LLIN1:DISP? /*The query returns 1.*/

:CALCulate:LPLot:LLINe<n>:TRACe

Syntax

:CALCulate:LPLot:LLINe<n>:TRACe <trace>
:CALCulate:LPLot:LLINe<n>:TRACe?

Description

Sets or queries the target trace of the specified limit line in the limit line test.

Parameter

Name	Type	Range	Default
<n>	Discrete	1|2|3|4|5|6	1
<trace>	Discrete	1|2|3	1

Return Format
The query returns the trace number.

Example
:CALC:LPL:LLIN1:TRAC 1 /*Sets the test trace of Limit Line 1 to Trace 1.*/
:CALC:LPL:LLIN1:TRAC? /*The query returns 1.*/

:CALCulate:LPLot:LLINe<n>:DELete

Syntax
:CALCulate:LPLot:LLINe<n>:DELete

Description
Deletes the specified limit line.

Parameter

Name	Type	Range	Default
<n>	Discrete	1|2|3|4|5|6	1

Example
:CALC:LPL:LLIN1:DEL /*Deletes Limit Line 1.*/

:CALCulate:LPLot:LLINe<n>:DATA

Syntax
:CALCulate:LPLot:LLINe<n>:DATA <x>,<ampl>,<connect>{,<x>,<ampl>,<connect>}
:CALCulate:LPLot:LLINe<n>:DATA?

Description
Edits one limit line, and marks it with n.
Queries the limit line data that you are editing currently.

Parameter

Name	Type	Range	Default
<n>	Discrete	1|2|3|4|5|6	1
<x>	Real	0 Hz to 6.5 GHz (X-axis indicates frequency)	——
<ampl>	Real	-1000 dBm to 1000 dBm	——
<connect>	Discrete	0|1	0

Remarks
<x>: indicates the frequency.
<apml>: indicates the amplitude. By default, its unit is dBm. The same X value can be configured with at most two amplitude values.
<connect>: can be configured with 0 or 1. When it is configured with 1, it indicates that the current point connects with the previous point to determine the limit line; when configured with 0, it indicates that the current point is disconnected from the previous point. The <connect> value of the first point can be configured with 0.

Return Format
The query returns the limit line data that are being edited currently in strings.

Example
:CALC:LPL:LLIN1:DATA 50,100,0,100,150,1,200,200,1
/*Edits a limit line that contains three points, and marks it Limit Line 1.*/
:CALC:LPL:LLIN1:DATA?
/*The query returns 50,100.000000,0,100,150.000000,1,200,200.000000,1.*/

:CALCulate:LPLot:LLINe<n>:MARGin

Syntax
:CALCulate:LPLot:LLINe<n>:MARGin <margin>
:CALCulate:LPLot:LLINe<n>:MARGin?

Description
Sets or queries the margin value of the specified limit line.

Parameter

Name	Type	Range	Default
<n>	Discrete	1|2|3|4|5|6	1
<margin>	Real	-100 dB to 100 dB	0 dB

Return Format
The query returns the margin value of the specified limit line in real number.

Example
:CALC:LPL:LLIN1:MARG 3 /*Sets the margin value of Limit Line 1 to 3 dB.*/
:CALC:LPL:LLIN1:MARG? /*The query returns 3.*/

:CALCulate:LPLot:LLINe<n>:MARGin:STATe

Syntax
:CALCulate:LPLot:LLINe<n>:MARGin:STATe <bool>
:CALCulate:LPLot:LLINe<n>:MARGin:STATe?

Description
Sets or queries the margin state of the specified limit line.

Parameter

Name	Type	Range	Default
<n>	Discrete	1|2|3|4|5|6	1
<bool>	Bool	ON|OFF|1|0	OFF|0

Return Format
The query returns 1 or 0.

Example
:CALC:LPL:LLIN1:MARG:STAT ON /*Enables the margin state.*/
:CALC:LPL:LLIN1:MARG:STAT? /*The query returns 1.*/

:CALCulate:LPLot:LLINe<n>:OFFSet:X

Syntax
:CALCulate:LPLot:LLINe<n>:OFFSet:X <offset>
:CALCulate:LPLot:LLINe<n>:OFFSet:X?

Description
Sets or queries the X-axis offset of the specified limit line.

Parameter

Name	Type	Range	Default
<n>	Discrete	1|2|3|4|5|6	1
<offset>	Real	-500 GHz to 500 GHz	0

Return Format
The query returns the X-axis offset of the specified limit line in real number.

Example
:CALC:LPL:LLIN1:OFFS:X 100 /*Sets the X-axis offset to 100 Hz.*/
:CALC:LPL:LLIN1:OFFS:X? /*The query returns 100.*/

:CALCulate:LPLot:LLINe<n>:OFFSet:Y

Syntax
:CALCulate:LPLot:LLINe<n>:OFFSet:Y <offset>
:CALCulate:LPLot:LLINe<n>:OFFSet:Y?

Description
Sets or queries the Y-axis offset of the specified limit line.

Parameter

Name	Type	Range	Default
<n>	Discrete	1|2|3|4|5|6	1
<offset>	Real	——	0

Return Format
The query returns the Y-axis offset of the specified limit line in real number.

Example
:CALC:LPL:LLIN1:OFFS:Y 5 /*Sets the Y-axis offset of Limit Line 1 to 5 dB.*/
:CALC:LPL:LLIN1:OFFS:Y? /*The query returns 5.*/

:CALCulate:LPLot:LLINe<n>:OFFSet:UPDate

Syntax
:CALCulate:LPLot:LLINe<n>:OFFSet:UPDate

Description
Applies the offset to the specified limit line.

Parameter

Name	Type	Range	Default
<n>	Discrete	1|2|3|4|5|6	1

Return Format
N/A

Example
:CALCulate:LPLot:LLINe1:OFFSet:UPDate /*Applies the offset to Limit Line 1.*/

:CALCulate:LPLot:LLINe<n>:AMPLitude:CMODe:RELative

Syntax
:CALCulate:LPLot:LLINe<n>:AMPLitude:CMODe:RELative <bool>
:CALCulate:LPLot:LLINe<n>:AMPLitude:CMODe:RELative?

Description
Sets or queries the type for Y to RF.

Parameter

Name	Type	Range	Default
<n>	Discrete	1|2|3|4|5|6	1
<bool>	Bool	OFF|ON|0|1	OFF|0

Remarks
OFF|0: "Fixed" is selected under **Y to Ref**, and the amplitude of the current limit line editing point is not affected by the reference value.
ON|1: "Relative" is selected under **Y to Ref**, and the amplitude of the current editing point is the difference between the amplitude of the point and that of the current reference value. At this time, if the reference value changes, then the position of the current editing point changes along with the reference value.

Return Format
The query returns 0 or 1.

Example
:CALCulate:LPLot:LLINe2:AMPLitude:CMODe:RELative OFF
or :CALCulate:LPLot:LLINe2:AMPLitude:CMODe:RELative 0
/*Sets "Y to Ref" to Fixed for Limit Line 2.*/

:CALCulate:LPLot:LLINe2:AMPLitude:CMODe:RELative? /*The query returns 0.*/

:CALCulate:LPLot:LLINe<n>:BUILd

Syntax
:CALCulate:LPLot:LLINe<n>:BUILd <trace>

Description
Builds the limit line from the selected trace.

Parameter

Name	Type	Range	Default
<n>	Discrete	1|2|3|4|5|6	1
<trace>	Discrete	TRACe1|TRACe2|TRACe3	TRACe1

Example
:CALC:LPL:LLIN1:BUIL TRAC1 /*Builds Limit Line 1 from TRACe 1.*/

:CALCulate:LPLot:LLINe<n>:DESCription

Syntax
:CALCulate:LPLot:LLINe<n>:DESCription <string>
:CALCulate:LPLot:LLINe<n>:DESCription?

Description
Sets or queries the description of the specified limit line.

Parameter

Name	Type	Range	Default
<n>	Discrete	1|2|3|4|5|6	1
<string>	ASCII String	Refer to Remarks	——

Remarks
The length of the description cannot exceed 256 characters.

Example
:CALC:LPL:LLIN1:DESC "Test Limit" /*Sets the description of Limit Line 1 to "Test Limit".*/
:CALC:LPL:LLIN1:DESC? /*The query returns "Test Limit".*/

:CALCulate:LPLot:LLINe<n>:COMMent

Syntax
:CALCulate:LPLot:LLINe<n>:COMMent <string>
:CALCulate:LPLot:LLINe<n>:COMMent?

Description

Sets or queries the comment of the specified limit line.

Parameter

Name	Type	Range	Default
<n>	Discrete	1|2|3|4|5|6	1
<string>	ASCII String	Refer to Remarks	——

Remarks

The length of the comment cannot exceed 256 characters.

Example

:CALC:LPL:LLIN1:COMM "Phase Noise Limit" /*Sets the comment of Limit Line 1 to "Phase Noise Limit".*/
:CALC:LPL:LLIN1:COMM? /*The query returns "Phase Noise Limit".*/

:CALCulate:LPLot:MARKer<n>:MODE

Syntax

:CALCulate:LPLot:MARKer<n>:MODE <mode>
:CALCulate:LPLot:MARKer<n>:MODE?

Description

Sets or queries the marker mode of the specified marker.

Parameter

Name	Type	Range	Default
<n>	Discrete	1|2|3|4|5|6|7|8	1
<mode>	Discrete	POSition|DELTa|OFF	OFF

Remarks

POSition: indicates the normal marker.
DELTa: indicates difference between two data points.
OFF: turns off the selected marker.

Return Format

The query returns POS, DELT, or OFF.

Example

:CALCulate:LPLot:MARKer1:MODE POSition /*Sets the type of Marker 1 to POSition.*/
:CALCulate:LPLot:MARKer1:MODE? /*The query returns POS.*/

:CALCulate:LPLot:MARKer<n>:REFerence

Syntax

:CALCulate:LPLot:MARKer<n>:REFerence <ref_marker>
:CALCulate:LPLot:MARKer<n>:REFerence?

Description
Sets or queries the reference marker of the specified marker.

Parameter

Name	Type	Range	Default
<n>	Discrete	1|2|3|4|5|6|7|8	1
<ref_marker>	Integer	1 to 8	Next marker

Return Format
The query returns the reference marker for the specified marker in integer.

Example
:CALCulate:LPLot:MARKer1:REFerence 2 /*Sets the reference marker of Marker 1 to Marker 2.*/
:CALCulate:LPLot:MARKer1:REFerence? /*The query returns 2.*/

:CALCulate:LPLot:MARKer<n>:X

Syntax
:CALCulate:LPLot:MARKer<n>:X <freq>
:CALCulate:LPLot:MARKer<n>:X?

Description
Sets or queries the marker X value of the specified marker.

Parameter

Name	Type	Range	Default
<n>	Discrete	1|2|3|4|5|6|7|8	——
<freq>	Real	——	——

Return Format
The query returns the marker X value of the specified marker in real number. Its unit is Hz.

Example
:CALCulate:LPLot:MARKer1:X 20000 /*Sets the marker X value of Marker 1 to 20 kHz.*/
:CALCulate:LPLot:MARKer1:X? /*The query returns 20000.*/

:CALCulate:LPLot:MARKer<n>:TRACe

Syntax
:CALCulate:LPLot:MARKer<n>:TRACe <trace>
:CALCulate:LPLot:MARKer<n>:TRACe?

Description
Sets or queries the marker trace for the specified marker.

Parameter

Name	Type	Range	Default

<n>	Discrete	1|2|3|4|5|6|7|8	1
<trace>	Integer	1|2|3	1

Remarks
<trace> indicates the marker trace. It can be set to Trace1 through Trace3.

Return Format
The query returns the marker trace number for the specified marker in integer.

Example
:CALCulate:LPLot:MARKer1:TRACe 2 /*Sets the marker trace of Marker 1 to Trace 2.*/
:CALCulate:LPLot:MARKer1:TRACe? /*The query returns 2.*/

:CALCulate:LPLot:MARKer<n>:FUNCtion

Syntax
:CALCulate:LPLot:MARKer<n>:FUNCtion <func>
:CALCulate:LPLot:MARKer<n>:FUNCtion?

Description
Sets or queries the marker function type of the specified marker.

Parameter

Name	Type	Range	Default
<n>	Discrete	1|2|3|4|5|6|7|8	1
<func>	Real	OFF|PMDEgree|PMRAdian|PMDBC|JITTer|RFM|AVERage	OFF

Remarks
OFF: disables the marker function.
PMDEgree: indicates residual PM(deg).
PMRAdian: indicates residual PM(rad).
PMDBC: indicates residual PM(dBc).
JITTer: indicates jitter.
RFM: indicates Residual FM.
AVERage: indicates average noise density.

Return Format
The query returns the marker function type of the specified marker.

Example
:CALCulate:LPLot:MARKer1:FUNCtion JITTer /*Sets the marker function type of Marker 1 to JITTer.*/
:CALCulate:LPLot:MARKer1:FUNCtion? /*The query returns JITT.*/

:CALCulate:LPLot:MARKer<n>:FUNCtion:BAND:SPAN

Syntax
:CALCulate:LPLot:MARKer<n>:FUNCtion:BAND:SPAN <freq>
:CALCulate:LPLot:MARKer<n>:FUNCtion:BAND:SPAN?

Description
Sets or queries the band span of the specified marker.

Parameter

Name	Type	Range	Default
<n>	Discrete	1|2|3|4|5|6|7|8	1
<freq>	Real	Start Frequency Offset - Stop Frequency Offset[1]	(Stop Frequency Offset - Start Frequency Offset) x 5%

Note[1]: Ensure that the band left and band right is within the range between start frequency offset and stop frequency offset. If the band span you set is not within this range, a prompt message "Out of range" is displayed.

Remarks

- When the band span value is modified, the band left and band right values will be updated automatically.
- When the marker function is disabled, the band span is automatically set to 0.
- To set or query the start frequency offset, run the **[:SENSe]:LPLot:FREQuency:OFFSet:STARt** command; to set or query the stop frequency offset, run the **[:SENSe]:LPLot:FREQuency:OFFSet:STOP** command.

Return Format
The query returns the band span of the specified marker in real number. Its unit is Hz.

Example
:CALC:LPL:MARK1:FUNC:BAND:SPAN 1000 /*Sets the band span of Marker 1 to 1 kHz.*/
:CALC:LPL:MARK1:FUNC:BAND:SPAN? /*The query returns 1000.*/

:CALCulate:LPLot:MARKer<n>:FUNCtion:BAND:LEFT

Syntax
:CALCulate:LPLot:MARKer<n>:FUNCtion:BAND:LEFT <freq>
:CALCulate:LPLot:MARKer<n>:FUNCtion:BAND:LEFT?

Description
Sets the left edge frequency of the signal involved in the calculation for the band function.

Parameter

Name	Type	Range	Default
<n>	Discrete	1|2|3|4|5|6|7|8	1
<freq>	Real	Start Frequency Offset - Stop Frequency Offset	Marker Frequency - 5% of (Stop Frequency - Cutoff Frequency)/2

Remarks

- When the band left value is modified, the band span value will be updated automatically.
- When the marker function is disabled, the band span is automatically set to 0.
- When the marker function is disabled and band span is set to zero, the band left is set to the current center frequency.

Return Format

The query returns the left edge frequency for the band of the specified marker in real number. Its unit is Hz.

Example

```
:CALC:LPL:MARK1:FUNC:BAND:LEFT 1000     /*Sets the left edge frequency for the band of Marker 1 to 1 kHz.*/
:CALC:LPL:MARK1:FUNC:BAND:LEFT?         /*The query returns 1000.*/
```

:CALCulate:LPLot:MARKer<n>:FUNCtion:BAND:RIGHt

Syntax

```
:CALCulate:LPLot:MARKer<n>:FUNCtion:BAND:RIGHt <freq>
:CALCulate:LPLot:MARKer<n>:FUNCtion:BAND:RIGHt?
```

Description

Sets the right edge frequency of the signal involved in the calculation for the band function.

Parameter

Name	Type	Range	Default
<n>	Discrete	1|2|3|4|5|6|7|8	1
<freq>	Real	Start Frequency Offset - Stop Frequency Offset	Marker Frequency + 5% of (Stop Frequency - Cutoff Frequency)/2

Remarks

- When the band left value is modified, the band span value will be updated automatically.
- When the marker function is disabled, the band span is automatically set to 0.
- When the marker function is disabled and band span is set to zero, the band right is set to the current center frequency.

Return Format

The query returns the right edge frequency for the band of the specified marker in real number. Its unit is Hz.

Example

```
:CALC:LPL:MARK1:FUNC:BAND:RIGT 1000     /*Sets the right edge frequency for the band of Marker 1 to 1 kHz.*/
:CALC:LPL:MARK1:FUNC:BAND:RIGT?         /*The query returns 1000.*/
```

:CALCulate:LPLot:MARKer<n>:FUNCtion:RESult?

Syntax
:CALCulate:LPLot:MARKer<n>:FUNCtion:RESult?

Description
Queries the calculation results of the marker function for the specified marker.

Parameter

Name	Type	Range	Default
<n>	Discrete	1|2|3|4|5|6|7|8	1

Remarks
The query returns the calculation results of the marker function for the specified marker in scientific notation. The unit is determined by the marker function type that you select.

Example
:CALCulate:LPLot:MARKer<n>:FUNCtion:RESult?/*The query returns 5.76300385056E-13.*/

:CALCulate:LPLot:MARKer<n>:SPURious:MAXimum

Syntax
:CALCulate:LPLot:MARKer<n>:SPURious:MAXimum

Description
Moves the specified marker to the max. spurious signal.

Parameter

Name	Type	Range	Default
<n>	Discrete	1|2|3|4|5|6|7|8	1

Example
:CALC:LPL:MARK1:SPUR:MAX /*Moves Marker 1 to max. spurious signal.*/

:CALCulate:LPLot:MARKer<n>:SPURious:MAXimum:NEXT

Syntax
:CALCulate:LPLot:MARKer<n>:SPURious:MAXimum:NEXT

Description
Moves the specified marker to the next higher spurious signal.

Parameter

Name	Type	Range	Default
<n>	Discrete	1|2|3|4|5|6|7|8	1

Example

:CALC:LPL:MARK1:SPUR:MAX:NEXT /*Moves Marker 1 to the next higher spurious signal.*/

:CALCulate:LPLot:MARKer<n>:SPURious:MAXimum:NEXT:RIGHt

Syntax
:CALCulate:LPLot:MARKer<n>:SPURious:MAXimum:NEXT:RIGHt

Description
Moves the specified marker to the next spurious signal to the right of the current marker position.

Parameter

Name	Type	Range	Default
<n>	Discrete	1|2|3|4|5|6|7|8	1

Example
:CALC:LPL:MARK1:SPUR:MAX:NEXT:RIGH /*Moves Marker 1 to the next spurious signal to the right of the current marker position.*/

:CALCulate:LPLot:MARKer<n>:SPURious:MAXimum:NEXT:LEFT

Syntax
:CALCulate:LPLot:MARKer<n>:SPURious:MAXimum:NEXT:LEFT

Description
Moves the specified marker to the next spurious signal to the left of the current marker position.

Parameter

Name	Type	Range	Default
<n>	Discrete	1|2|3|4|5|6|7|8	1

Example
:CALC:LPL:MARK1:SPUR:MAX:NEXT:LEFT /*Moves Marker 1 to the next spurious signal to the left of the current marker position.*/

:CALCulate:LPLot:SPURious:SORTmode

Syntax
:CALCulate:LPLot:SPURious:SORTmode <mode>
:CALCulate:LPLot:SPURious:SORTmode?

Description
Sets or queries the sorting mode of the spurious table.

Parameter

Name	Type	Range	Default
<mode>	Discrete	FREQ|AMPL	AMPL

Remarks

- FREQ: sorted by frequency in ascending order.
- AMPL: sorted by amplitude in descending order.

Example

:CALC:LPL:SPUR:SORT FREQ /*Sorted by frequency.*/
:CALC:LPL:SPUR:SORT? /*The query returns FREQ.*/

:CALCulate:LPLot:SPURious:THReshold

Syntax

:CALCulate:LPLot:SPURious:THReshold <offset>
:CALCulate:LPLot:SPURious:THReshold?

Description

Sets or queries the spurious offset.

Parameter

Name	Type	Range	Default
<offset>	Real	0 dB to 50 dB	10 dB

Remarks

Only the value above the spurious offset can be identified as spurious.

Return Format

The query returns the spurious offset in scientific notation.

Example

:CALC:LPL:SPUR:THR 25 /*Sets the spurious offset to 25 dB.*/
:CALC:LPL:SPUR:THR? /*The query returns 2.5E+01.*/

:CALCulate:LPLot:SPURious:TABLe

Syntax

:CALCulate:LPLot:SPURious:TABLe <bool>
:CALCulate:LPLot:SPURious:TABLe?

Description

Sets or queries whether to display the spurious table.

Parameter

Name	Type	Range	Default
<bool>	Bool	ON|OFF|1|0	OFF|0

Return Format
The query returns 1 or 0.

Example
:CALC:LPL:SPUR:TABL ON /*Enables the display of the spurious table.*/
:CALC:LPL:SPUR:TABL? /*The query returns 1.*/

:CALCulate:MARKer:AOFF

Syntax
:CALCulate:MARKer:AOFF

Description
Turns off all the enabled markers.

:CALCulate:MARKer:COUPle[:STATe]

Syntax
:CALCulate:MARKer:COUPle[:STATe] <bool>
:CALCulate:MARKer:COUPle[:STATe]?

Description
Sets or queries the marker coupling state.

Parameter

Name	Type	Range	Default
<bool>	Bool	OFF|ON|0|1	OFF|0

Remarks
When this function enabled, moving any marker will enable other markers (except the OFF marker) to move with it.

Return Format
The query returns 0 or 1.

Example
:CALCulate:MARKer:COUPle:STATe OFF or :CALCulate:MARKer:COUPle:STATe 0
/*Disables the coupling state of the marker.*/
:CALCulate:MARKer:COUPle:STATe? /*The query returns 0.*/

:CALCulate:MARKer:NEXT

Syntax
:CALCulate:MARKer:NEXT

Description

Sets the next marker.

:CALCulate:MARKer<n>:LINes[:STATe]

Syntax
:CALCulate:MARKer<n>:LINes[:STATe] <bool>
:CALCulate:MARKer<n>:LINes[:STATe]?

Description
Sets or queries whether to display the marker line for the specified marker.

Parameter

Name	Type	Range	Default
<n>	Discrete	1|2|3|4|5|6|7|8	1
<bool>	Bool	OFF|ON|0|1	OFF|0

Return Format
The query returns 0 or 1.

Example
:CALCulate:MARKer1:LINes:STATe ON or :CALCulate:MARKer1:LINes:STATe 1 /*Enables the display of the marker line for Marker 1.*/
:CALCulate:MARKer1:LINes:STATe? /*The query returns 1.*/

:CALCulate:MARKer:TABLe:STATe

Syntax
:CALCulate:MARKer:TABLe:STATe <bool>
:CALCulate:MARKer:TABLe:STATe?

Description
Enables or disables the display of the marker table; queries whether the marker table is displayed.

Parameter

Name	Type	Range	Default
<bool>	Bool	OFF|ON|0|1	OFF|0

Return Format
The query returns 0 or 1.

Example
:CALCulate:MARKer:TABLe:STATe ON or :CALCulate:MARKer:TABLe:STATe 1 /*Enables the display of the marker table.*/
:CALCulate:MARKer:TABLe:STATe? /*The query returns 1.*/

:CALibration Commands

:CALibration[:ALL]

Syntax
:CALibration[:ALL]

Description
Executes self-calibration immediately.

Example
:CALibration:ALL /*Executes self-calibration immediately.*/

:CALibration:AUTO

Syntax
:CALibration:AUTO <bool>
:CALibration:AUTO?

Description
Enables or disables auto calibration.
Queries the setting status of auto calibration.

Parameter

Name	Type	Range	Default
<bool>	Bool	OFF|ON|0|1	OFF|0

Return Format
The query returns 0 or 1.

Example
:CALibration:AUTO ON or :CALibration:AUTO 1 /*Enables the auto calibration.*/
:CALibration:AUTO? /*The query returns 1.*/

:CONFigure Commands

:CONFigure:CATalog?

Syntax
:CONFigure:CATalog?

Description
Queries the current measurement function.

Return Format
The query returns PNOISE.

:COUPle Commands

:COUPle

Syntax
:COUPle <enum>

Description
Enables auto coupling.

Parameter

Name	Type	Range	Default
<enum>	Discrete	ALL|NONE	——

Remarks
ALL: enables auto coupling.
NONE: disables auto coupling. Couple manually.

Return Format
N/A

Example
:COUPle NONE /*Disables auto coupling function.*/

:DISPlay Commands

:DISPlay:LPLot:VIEW[:Select]

Syntax

:DISPlay:LPLot:VIEW[:Select] <view>
:DISPlay:LPLot:VIEW[:Select]?

Description

Sets or queries the current combination view.

Parameter

Name	Type	Range	Default
<view>	Discrete	NORMal|DECade|SPURious	NORMal

Remarks

NORMal: normal view. In this view, it displays the log plot graph, displaying the current noise measurement trace.
DECade: Decade view. When you set it to DECade, both log plot graph view and decade table view are displayed.
SPURious: Spurious view. When you set it to SPURious, both log plot graph view and spurious table view are displayed.

Return Format

The query returns NORM, DEC, or SPUR.

Example

:DISP:LPL:VIEW NORMal /*Sets the combination view to NORMal.*/
:DISP:LPL:VIEW? /*The query returns NORM.*/

:DISPlay:LPLot:WINDow[1]:TRACe:Y[:SCALe]:RLEVel

Syntax

:DISPlay:LPLot:WINDow[1]:TRACe:Y[:SCALe]:RLEVel <ampl>
:DISPlay:LPLot:WINDow[1]:TRACe:Y[:SCALe]:RLEVel?

Description

Sets or queries the reference value in the log plot graph.

Parameter

Name	Type	Range	Default
<ampl>	Real	-200 dBc/Hz to 200 dBc/Hz	-70 dBc/Hz

Return Format

The query returns the reference value in real number. The default unit is dBc/Hz.

Example

:DISP:LPL:WIND1:TRAC:Y:RLEV 0 /*Sets the reference value in log plot graph to 0 dBc/Hz.*/
:DISP:LPL:WIND1:TRAC:Y:RLEV? /*The query returns 0.*/

:DISPlay:LPLot:WINDow[1]:TRACe:Y[:SCALe]:PDIVision

Syntax
:DISPlay:LPLot:WINDow[1]:TRACe:Y[:SCALe]:PDIVision
:DISPlay:LPLot:WINDow[1]:TRACe:Y[:SCALe]:PDIVision?

Description
Sets or queries the scale per division in the log plot graph.

Parameter

Name	Type	Range	Default
	Real	0.1 dB to 20 dB	10 dB

Return Format
The query returns the Y-axis scale value in real number.

Example
:DISP:LPL:WIND1:TRAC:Y:PDIV 10 /*Sets the scale/div value to 10 dB.*/
:DISP:LPL:WIND1:TRAC:Y:PDIV? /*The query returns 10.*/

:DISPlay:LPLot:WINDow[1]:TRACe:Y[:SCALe]:COUPle

Syntax
:DISPlay:LPLot:WINDow[1]:TRACe:Y[:SCALe]:COUPle <bool>
:DISPlay:LPLot:WINDow[1]:TRACe:Y[:SCALe]:COUPle?

Description
Sets or queries whether to enable the auto scaling function.

Parameter

Name	Type	Range	Default
<bool>	Bool	ON|OFF|1|0	OFF

Return Format
The query returns 1 or 0.

Example
:DISP:LPL:WIND1:TRAC:Y:COUP ON /*Enables the auto scaling function.*/
:DISP:LPL:WIND1:TRAC:Y:COUP? /*The query returns 1.*/

IEEE 488.2 Common Commands

IEEE 488.2 common commands are used to operate or query the status registers.

*CLS

Syntax
*CLS

Description
Clears all the event registers and status byte registers.

*ESE

Syntax
*ESE <value>
*ESE?

Description
Sets or queries the enable register of the standard event status register.

Parameter

Name	Type	Range	Default
<value>	Integer	Refer to "**Remarks**"	0

Remarks
The bit 2, bit 3, bit 4, and bit 7 are reserved; you can set their values but they will not affect the system. Bit 1 and bit 6 are not used and are always treated as 0; therefore, the range of <value> are the decimal numbers corresponding to the binary numbers ranging from 00000000 (0 in decimal) to 11111111 (255 in decimal) and of which bit 1 and bit 6 are 0.

Return Format
The query returns an integer. The integer equals to the binary-weighted sum of all the bits set in the register. For example, the query returns 144 if Bit 4 (16 in decimal) and Bit 7 (128 in decimal) are enabled.

Example
The following command sets the enable register for the standard event status register to 16.
*ESE 16

The following query returns 16.
*ESE?

*ESR?

Syntax
*ESR?

Description
Queries and clears the event register for the standard event status register.

Remarks
Bit 1 and Bit 6 in the standard event status register are not in use, and are regarded as 0. The query returns a decimal value that corresponds to the binary values ranging from 00000000 (0 in decimal) to 11111111 (255 in decimal) and of which Bit 1 and Bit 6 are 0.

Return Format
The query returns an integer. The integer equals to the binary-weighted sum of all the bits set in the register. For example, the query returns 144 if Bit 4 (16 in decimal) and Bit 7 (128 in decimal) are enabled.

Example
The following query returns 24 (Bit 3 and Bit 4 have been set).
*ESR?

*IDN?

Syntax
*IDN?

Description
Queries the ID string of instrument.

Return Format
The query returns the ID string in the following format:
RIGOL TECHNOLOGIES,<model>,<serial number>,XX.XX.XX
<model>: instrument model
<serial number>: serial number of the instrument
XX.XX.XX: software version of the instrument

Example
The following query returns RIGOL TECHNOLOGIES,RSA6265,RSA60004000000,00.00.25.
*IDN?

*OPC

Syntax
*OPC
*OPC?

Description
Sets Bit 0 (Operation Complete, OPC) in the standard event status register to 1 after the current operation is finished.
Queries whether the current operation is finished.

Return Format

The query returns 1 after the current operation is finished; otherwise, the query returns 0.

*RCL

Syntax
*RCL <integer>

Description
Recalls the selected register.

Parameter

Name	Type	Range	Default
<integer>	Integer	1 to 16	——

Example
The following command recalls Register 1.
*RCL 1

*RST

Syntax
*RST

Description
Restores the instrument to its factory default settings.

*SAV

Syntax
*SAV <integer>

Description
Saves the current instrument state to the selected register.

Parameter

Name	Type	Range	Default
<integer>	Integer	1 to 16	——

Example
The following command saves the current instrument state to Register 1.
*SAV 1

*SRE

Syntax
*SRE <value>
*SRE?

Description
Sets or queries the enable register of the status byte register.

Parameter

Name	Type	Range	Default
<value>	Integer	Refer to "**Remarks**"	0

Remarks
Bit 0 and Bit 1 are not used and are always treated as 0; therefore, the range of <value> are the decimal numbers corresponding to the binary numbers ranging from 00000000 (0 in decimal) to 11111111 (255 in decimal) and of which Bit 0 and Bit 1 are 0.

Return Format
The query returns an integer. The integer equals to the binary-weighted sum of all the bits set in the register. For example, the query returns 144 if Bit 4 (16 in decimal) and Bit 7 (128 in decimal) are enabled.

Example
The following command sets the enable register for the status byte register to 16.
*SRE 16

The following query returns 16.
*SRE?

*STB?

Syntax
*STB?

Description
Queries the event register for the status byte register.

Remarks
Bit 0 and Bit 1 in the status byte register are not in use, and are regarded as 0. The query returns a decimal value that corresponds to the binary values ranging from 00000000 (0 in decimal) to 11111111 (255 in decimal) and of which Bit 0 and Bit 1 are 0.

Return Format
The query returns an integer. The integer equals to the binary-weighted sum of all the bits set in the register. For example, the query returns 144 if Bit 4 (16 in decimal) and Bit 7 (128 in decimal) are enabled.

Example
The following query returns 24 (Bit 3 and Bit 4 have been set).
*STB?

*TST?

Syntax
*TST?

Description
Queries whether the self-check operation is finished.

Remarks
The query returns 0 or 1. A zero is returned if the test is successful, 1 if it fails.

*WAI

Syntax
*WAI

Description
Waits for all the pending operations to complete before executing any additional commands.

:GPIB:PARSe:END

Syntax
:GPIB:PARSe:END

Description
The GPIB module command. This instruction set sends the "Parse End" command to the GPIB module. If this command is not sent, the USB-GPIB adapter module fails to work. You will receive no return value after sending a command.

:INITiate Commands

:INITiate:CONTinuous

Syntax
:INITiate:CONTinuous <bool>
:INITiate:CONTinuous?

Description
Selects continuous (ON|1) or single (OFF|0) measurement mode.
Queries the current measurement
mode.

Parameter

Name	Type	Range	Default
<bool>	Bool	OFF|ON|0|1	ON|1

Return Format
The query returns 0 or 1.

Example
:INITiate:CONTinuous ON or :INITiate:CONTinuous 1 /*Performs continuous measurement.*/
:INITiate:CONTinuous? /*The query returns 1.*/

:INITiate[:IMMediate]

Syntax
:INITiate[:IMMediate]

Description
Initializes one sweep in the non-measurement state. Triggers one measurement in the measurement state.

:INITiate:RESTart

Syntax
:INITiate:RESTart

Description
Restarts to initiate a sweep.

:INSTrument Commands

:INSTrument:SCReen:CATalog?

Syntax
:INSTrument:SCReen:CATalog?

Description
Queries the available measurement modes displayed at the bottom of the screen.

:INSTrument:SCReen:CREate

Syntax
:INSTrument:SCReen:CREate

Description
Creates a new measurement mode label at the bottom of the screen.

:INSTrument:SCReen:DELete

Syntax
:INSTrument:SCReen:DELete

Description
Deletes the currently selected measurement mode label at the bottom of the screen.

:INSTrument:SCReen:DELete:ALL

Syntax
:INSTrument:SCReen:DELete:ALL

Description
Deletes all the currently not selected measurement mode label at the bottom of the screen.

:INSTrument:SCReen:SELect

Syntax
:INSTrument:SCReen:SELect <name>
:INSTrument:SCReen:SELect?

Description
Switches to the specified screen.
Queries the currently selected screen.

Parameter

Name	Type	Range	Default

<name>	ASCII String	The label can contain English letters and numbers, as well as some symbols.	——

Remarks
You can run the *:INSTrument:SCReen:CATalog?* command to query the measurement mode label name.

Example
:INSTrument:SCReen:SELect PhaseNoise 1 /*Switches to the screen PhaseNoise 1.*/
:INSTrument:SCReen:SELect?/ /*The query returns PhaseNoise 1.*/

:INSTrument[:SELect]

Syntax
:INSTrument[:SELect] <enum>
:INSTrument[:SELect]?

Description
Selects the working mode of the instrument.
Queries the working mode of the instrument.

Parameter

Name	Type	Range	Default
<enum>	Discrete	SA|GPSA_TG|RTSA|RTSA_UFS|GPSA_ACP|GPSA_CNR|GPSA_CHPower|GPSA_TOI|GPSA_HARModist|GPSA_OBW|GPSA_TPOWer|EMI|VSA|AM|FM|PM|PNOISE|PULSE|VSA_ZIGBEE|VSA_LORA|VSA_TX|GPSA_IBEM	SA

Remarks
After running the command of switching the working mode, we recommend you set the timeout value to 8 s, or perform the next operation after a delay of 8 s.

Example
:INSTrument:SELect PNOISE /*Sets the instrument to work in PNOISE mode.*/
:INSTrument:SELect? /*The query returns PNOISE.*/

:LAN Commands

:LAN:DHCP

Syntax
:LAN:DHCP <bool>
:LAN:DHCP?

Description
Turns on or off the DHCP configuration mode; or queries the on/off status of the current DHCP configuration mode.

Parameter

Name	Type	Range	Default
<bool>	Bool	OFF|ON|0|1	ON|1

Remarks

When the three IP configuration types (DHCP, Auto IP, and Static IP) are all turned on, the priority of the parameter configuration from high to low is "DHCP", "Auto IP", and "Static IP". When DHPC is valid, the DHCP server in the current network will assign the network parameters (such as the IP address) for the oscilloscope.
After the :LAN:APPLy command is executed, the configuration type can take effect immediately.

Return Format

The query returns 1 or 0.

Exameple

:LAN:DHCP OFF /*Disables DHCP configuration mode.*/
:LAN:DHCP? /*The query returns 0.*/

:LAN:AUToip

Syntax

:LAN:AUToip <bool>
:LAN:AUToip?

Description

Turns on or off the Auto IP configuration mode; or queries the on/off status of the current Auto IP configuration mode.

Parameter

Name	Type	Range	Default
<bool>	Bool	OFF|ON|0|1	ON|1

Remarks

When the auto IP mode is valid, disable DHCP manually. In Auto IP mode, the instrument gets the IP address (ranging from 169.254.0.1 to 169.254.255.254) and the subnet mask (255.255.0.0) automatically according to the current network configuration.

Return Format

The query returns 1 or 0.

Example

:LAN:AUToip OFF /*Disables the Auto IP configuration mode.*/
:LAN:AUToip? /*The query returns 0.*/

:LAN:MANual

Syntax
:LAN:MANual <bool>
:LAN:MANual?

Description
Turns on or off the static IP configuration mode; or queries the on/off status of the static IP configuration mode.

Parameter

Name	Type	Range	Default
<bool>	Bool	OFF|ON|0|1	OFF|0

Remarks
When the static IP mode is valid, disable DHCP and Auto IP manually. You can self-define the network parameters of the oscilloscope, such as IP address, subnet mask, gateway, and DNS address. For the setting of the IP address, refer to the :LAN:IPADdress command. For the setting of the subnet mask, refer to the :LAN:SMASk command. For the setting of the gateway, refer to the :LAN:GATeway command. For the setting of DNS, refer to the :LAN:DNS command.

Return Format
The query returns 1 or 0.

Example
:LAN:MANual ON /*Enables the static IP configuration mode.*/
:LAN:MANual? /*The query returns 1.*/

:LAN:IPADdress

Syntax
:LAN:IPADdress <string>
:LAN:IPADdress?

Description
Sets or queries the IP address of the instrument.

Parameter

Name	Type	Range	Default
<string>	ASCII String	Refer to "**Remarks**"	——

Remarks
The format of the<string> is "nnn.nnn.nnn.nnn". The range for the first segment (nnn) of the address is from 0 to 223 (except 127); and the range for the other three segments is from 0 to 255.
Only when the IP configuration mode is static IP and both DHCP and Auto IP are disabled, can you use the set IP address.

Return Format
The query returns the current IP address in strings.

Example
:LAN:IPADdress 192.168.1.10 /*Sets the IP address to 192.168.1.10.*/
:LAN:IPADdress? /*The query returns 192.168.1.10.*/

:LAN:SMASk

Syntax
:LAN:SMASk <string>
:LAN:SMASk?

Description
Sets or queries the subnet mask.

Parameter

Name	Type	Range	Default
<string>	ASCII String	Refer to "**Remarks**"	——

Remarks
The format of the<string> is "nnn.nnn.nnn.nnn". The range for each segment (nnn) is from 0 to 255.
Only when the IP configuration mode is static IP and both DHCP and Auto IP are disabled, can you use the set subnet mask.

Return Format
The query returns the current subnet mask in strings.

Example
:LAN:SMASk 255.255.255.0 /*Sets the subnet mask to 255.255.255.0.*/
:LAN:SMASK? /*The query returns 255.255.255.0.*/

:LAN:GATeway

Syntax
:LAN:GATeway <string>
:LAN:GATeway?

Description
Sets or queries the default gateway.

Parameter

Name	Type	Range	Default
<string>	ASCII String	Refer to "**Remarks**"	——

Remarks
The format of the<string> is "nnn.nnn.nnn.nnn". The range for the first segment (nnn) of the address is from 0 to 223 (except 127); and the range for the other three segments is from 0 to 255.

Only when the IP configuration mode is static IP and both DHCP and Auto IP are disabled, can you use the set default gateway.

Return Format
The query returns the current gateway in strings.

Example
:LAN:GATeway 192.168.1.1 /*Sets the default gateway to 192.168.1.1.*/
:LAN:GATeway? /*The query returns 192.168.1.1.*/

:LAN:DNS

Syntax
:LAN:DNS <string>
:LAN:DNS?

Description
Sets or queries the DNS address.

Parameter

Name	Type	Range	Default
<string>	ASCII String	Refer to "**Remarks**"	——

Remarks
The format of the<string> is "nnn.nnn.nnn.nnn". The range for the first segment (nnn) of the address is from 0 to 223 (except 127); and the range for the other three segments is from 0 to 255.
Only when the IP configuration mode is static IP and both DHCP and Auto IP are disabled, can you use the set DNS.

Return Format
The query returns the current DNS address in strings.

Example
:LAN:DNS 192.168.1.1 /*Sets the DNS address to 192.168.1.1.*/
:LAN:DNS? /*The query returns 192.168.1.1.*/

:LAN:MAC?

Syntax
:LAN:MAC?

Description
Queries the MAC address.

Return Format
The query returns the MAC address in strings. For example, 00:19:AF:00:11:22.

:LAN:DSERver?

Syntax
:LAN:DSERver?

Description
Queries the DHCP server address.

Return Format
The query returns the DHCP server address in strings.

:LAN:STATus?

Syntax
:LAN:STATus?

Description
Queries the current network configuration status.

Remarks
- UNLINK: Not connected!
- CONNECTED: Connected successfully.
- INIT: acquiring IP address.
- IPCONFLICT: IP conflicted.
- BUSY: please wait...
- CONFIGURED: network configuration succeeded.
- DHCPFAILED: DHCP configuration failed.
- INVALIDIP: invalid IP.
- IPLOSE: IP lost.

Return Format
The query returns UNLINK, CONNECTED, INIT, IPCONFLICT, BUSY, CONFIGURED, DHCPFAILED, INVALIDIP, or IPLOSE.

:LAN:VISA?

Syntax
:LAN:VISA? [<type>]

Description
Queries the VISA address of the instrument.

Parameter

Name	Type	Range	Default
<type>	Discrete	{USB|LXI}	——

Remarks

This command contains a parameter "type" and it is used to set or query the address type. By default, it returns the LXI address.

Return Format
The query returns the VISA address in strings.

:LAN:MDNS

Syntax
:LAN:MDNS <bool>
:LAN:MDNS?

Description
Enables or disables mDNS; or queries the mDNS status.

Parameter

Name	Type	Range	Default
<bool>	Bool	OFF|ON|0|1	OFF|0

Return Format
The query returns 0 or 1.

Example
:LAN:MDNS ON /*Enables mDNS.*/
:LAN:MDNS? /*The query returns 1.*/

:LAN:APPLy

Syntax
:LAN:APPLy

Description
Applies the network configuration.

Remarks
After configuring the LAN parameters, run this command to apply the LAN settings.

:LAN:HOST:NAME

Syntax
:LAN:HOST:NAME <name>
:LAN:HOST:NAME?

Description
Sets or queries the host name.

Parameter

Name	Type	Range	Default
<name>	ASCII String	The label can contain English letters and numbers, as well as some symbols.	——

Return Format

The query returns the host name in ASCII strings.

:LAN:DESCription

Syntax

:LAN:DESCription <name>
:LAN:DESCription?

Description

Sets or queries the description.

Parameter

Name	Type	Range	Default
<name>	ASCII String	The label can contain English letters and numbers, as well as some symbols.	——

Return Format

The query returns the description in ASCII strings.

:MMEMory Commands

:MMEMory:STORe:STATe

Syntax

:MMEMory:STORe:STATe <file_name>

Description

Saves the current instrument state as a file with the specified file name suffixed with "*.sta" to the default path (/pnoise/state).

Parameter

Name	Type	Range	Default
<file_name>	ASCII String	——	——

Remarks

If the specified file already exists, overwrite it.

Example

:MMEMory:STORe:STATe state /*Saves the current instrument state as a file named state.sta to the /pnoise/state folder.*/

:MMEMory:STORe:TRACe

Syntax
:MMEMory:STORe:TRACe <label>,<file_name>

Description
Saves the specified trace and instrument state as a file with the specified file name suffixed with "*.trs" to the default path (/pnoise/trace).

Parameter

Name	Type	Range	Default
<label>	Discrete	TRACE1|TRACE2|TRACE3	——
<file_name>	ASCII String	——	——

Remarks
If the specified file already exists, overwrite it.

Example
:MMEMory:STORe:TRACe TRACE1,mydata
/*Saves Trace 1 data and instrument state as a file with the file name mydata.trs to the default path (/pnoise/trace).*/

:MMEMory:STORe:TRACe:DATA

Syntax
:MMEMory:STORe:TRACe:DATA <label>,<file_name>

Description
Saves the trace measurement data as a file with the specified file name suffixed with "*.csv" to the default path (/pnoise/measdata).

Parameter

Name	Type	Range	Default
<label>	Discrete	TRACE1|TRACE2|TRACE3	——
<file_name>	ASCII String	——	——

Remarks
If the specified file already exists, overwrite it.

Example
:MMEMory:STORe:TRACe:DATA TRACE1,mydata
/*Saves the Trace1 measurement data as a file with the file name mydata.csv to the default path (/pnoise/measdata).*/

:MMEMory:STORe:CORRection

Syntax

:MMEMory:STORe:CORRection <label>,<file_name>

Description

Saves the amplitude correction data of the current limit line with the specified filename suffixed with ".csv" by default (you do not have to add the suffix manually) to the default path (/pnoise/correction).

Parameter

Name	Type	Range	Default
<label>	Discrete	LLINE1|LLINE2|LLINE3|LLINE4|LLINE5|LLINE6	——
<file_name>	ASCII String	——	——

Remarks

If the specified file already exists, overwrite it.

Example

:MMEMory:STORe:CORRection LLINE1,low

Saves the amplitude correction data of Limit Line 1 with the filename "low" to the /pnoise/correction folder.*/

:MMEMory:STORe:LIMit

Syntax

:MMEMory:STORe:LIMit <label>,<file_name>

Description

Saves the currently edited limit line with the specified filename suffixed with ".csv" by default (you do not have to add the suffix manually) to the default path (/pnoise/limit).

Parameter

Name	Type	Range	Default
<label>	Discrete	LLINE1|LLINE2|LLINE3|LLINE4|LLINE5|LLINE6	——
<file_name>	ASCII String	——	——

Remarks

If the specified file already exists, overwrite it.

Example

:MMEMory:STORe:LIMit LLINE1,low

/*Saves the Limit Line 1 data with the filename "low" to the /pnoise/limit folder.*/

:MMEMory:STORe:IMG:PNG

Syntax
:MMEMory:STORe:IMG:PNG <path>

Description
Saves the screen image to the specified path.

Parameter

Name	Type	Range	Default
<file>	ASCII String	——	——

:MMEMory:STORe:SCReen

Syntax
:MMEMory:STORe:SCReen <file>

Description
Saves the current screenshot as a file with the specified file name suffixed with "*.jpg", "*.png", or "*.bmp" to the default path (/pnoise/screen).

Parameter

Name	Type	Range	Default
<file>	ASCII String	——	——

Remarks
If the specified file already exists, overwrite it.
If a suffix (.jpg/.png/.bmp) is added to the filename, you can save the current screen image with a different format based on its different suffix.
If no suffix is added to the filename, then by default, the current screen image is saved in the currently selected format.

Example
:MMEMory:STORe:SCReen screen.jpg
/*Saves the current screenshot as a file with the file name screen.jpg to the /pnoise/screen folder.*/

:MMEMory:STORe:MTABle

Syntax
:MMEMory:STORe:MTABle <file_name>

Description
Saves the marker table with a specified filename suffixed with ".csv" by default (you do not have to add the suffix manually) to the default path (/pnoise/measdata).

Parameter

Name	Type	Range	Default
<file_name>	ASCII String	——	——

Remarks
If the specified file already exists, overwrite it.

Example
The following command saves the marker table with the specified filename "MAK1" to the folder (/pnoise/measdata).
:MMEMory:STORe:MTABle MAK1

:MMEMory:STORe:PTABle

Syntax
:MMEMory:STORe:PTABle <file_name>

Description
Saves the peak table with a specified filename suffixed with ".csv" by default (you do not have to add the suffix manually) to the default path (/pnoise/measdata).

Parameter

Name	Type	Range	Default
<file_name>	ASCII String	——	——

Remarks
If the specified file already exists, overwrite it.

Example
The following command saves the spurious table with the specified filename "PT1" to the folder (/pnoise/measdata).
:MMEMory:STORe:PTABle PT1

:MMEMory:STORe:SCReen:DATA?

Syntax
:MMEMory:STORe:SCReen:DATA?

Description
Saves the screen data.

:MMEMory:STORe:STATe?

Syntax
:MMEMory:STORe:STATe?

Description
Obtains the status register.

:MMEMory:STORe:TRACe?

Syntax
:MMEMory:STORe:TRACe?

Description
Obtains the trace register.

:MMEMory:LOAD:STATe

Syntax
:MMEMory:LOAD:STATe <file_name>

Description
Loads the specified state file suffixed with "*.sta".

Parameter

Name	Type	Range	Default
<file_name>	ASCII String	——	——

Remarks
This operation fails if the file with the specified filename does not exist.

Example
:MMEMory:LOAD:STATe state1.sta /*Loads the state file named state1.sta to the instrument.*/

:MMEMory:LOAD:TRACe

Syntax
:MMEMory:LOAD:TRACe <label>,<file_name>

Description
Loads the specified trace+state file (suffixed with .trs).

Parameter

Name	Type	Range	Default
<label>	Discrete	TRACE1|TRACE2|TRACE3	——
<file_name>	ASCII String	——	——

Remarks
This operation fails if the file with the specified filename does not exist.

Example
:MMEMory:LOAD:TRACe TRACE1,mydata,trs
/*Loads the trace+state file named mydata.trs to Trace 1.*/

:MMEMory:LOAD:TRACe:DATA

Syntax
:MMEMory:LOAD:TRACe:DATA <label>,<file_name>

Description
Loads the specified measurement data file (suffixed with .csv).

Parameter

Name	Type	Range	Default
<label>	Discrete	TRACE1|TRACE2|TRACE3	——
<file_name>	ASCII String	——	——

Remarks
This operation fails if the file with the specified filename does not exist.

Example
:MMEMory:LOAD:TRACe:DATA TRACE1,measdata1.csv
/*Loads the measurement data file named trace1.csv to Trace 1.*/

:MMEMory:LOAD:LIMit

Syntax
:MMEMory:LOAD:LIMit <label>,<file_name>

Description
Loads the edited limit line file (.csv).

Parameter

Name	Type	Range	Default
<label>	Discrete	LLINE1|LLINE2|LLINE3|LLINE4|LLINE5|LLINE6	——
<file_name>	ASCII String	——	——

Remarks
This operation fails if the file with the specified filename does not exist.

Example
:MMEMory:LOAD:LIMit LLINE1,limlt1.csv
/*Loads the limit line file limlt1.csv to Limit Line 1.*/

:MMEMory:LOAD:CORRection

Syntax
:MMEMory:LOAD:CORRection <label>,<file_name>

Description
Loads the amplitude correction data of the specified file suffixed with "*.csv".

Parameter

Name	Type	Range	Default
<label>	Discrete	1|2|3|4	——
<file_name>	ASCII String	——	——

Remarks
The parameter <label> indicates the amplitude correction type. 1 indicates Antenna, 2 indicates Cable, 3 indicates Others, and 4 indicates User.
This operation fails if the file with the specified filename does not exist.

Example
:MMEMory:LOAD:CORRection 1,correction1.csv
/*Loads the antenna amplitude correction file correction1.csv to the instrument.*/

:MMEMory:USER:STORe

Syntax
:MMEMory:USER:STORe <user>
:MMEMory:USER:STORe?

Description
Sets or queries the preset file.

Parameter

Name	Type	Range	Default
<user>	Discrete	DEF|USER1|USER2|USER3|USER4|USER5|USER6	DEF

Example
:MMEMory:USER:STORe USER1 /*Sets the preset file to USER1.*/
:MMEMory:USER:STORe? /*The query returns USER1.*/

[:SENSe] Commands

[:SENSe]:AVERage:COUNt

Syntax

[:SENSe]:AVERage:COUNt <integer>
[:SENSe]:AVERage:COUNt?

Description

Sets or queries the trace average count of the current measurement.

Parameter

Name	Type	Range	Default
<integer>	Integer	1 to 1000	10

Return Format

The query returns the number of average times in integer.

Example

:SENSe:AVERage:COUNt 100 /*Sets the average count to 100.*/
:SENSe:AVERage:COUNt? /*The query returns 100.*/

[:SENSe]:AVERage[:STATe]

Syntax

[:SENSe]:AVERage[:STATe] <bool>
[:SENSe]:AVERage[:STATe]?

Description

Sets or queries the average state.

Parameter

Name	Type	Range	Default
<bool>	Bool	OFF|ON|0|1	OFF|0

Return Format

The query returns 0 or 1.

Example

:SENSe:AVERage:STATe ON or :SENSe:AVERage:STATe 1 /*Enables the average state.*/
:SENSe:AVERage:STATe? /*The query returns 1.*/

[:SENSe]:AVERage:TCONtrol

Syntax

[:SENSe]:AVERage:TCONtrol <enum>
[:SENSe]:AVERage:TCONtrol?

Description
Sets or queries the average mode.

Parameter

Name	Type	Range	Default
<enum>	Discrete	EXPonential|REPeat	——

Remarks
EXPonential: indicates the exponential average mode.
REPeat: indicates the repeated average mode.

Return Format
The query returns EXP or REP.

Example
:SENSe:AVERage:TCONtrol REPeat /*Sets the average mode to REPeat.*/
:SENSe:AVERage:TCONtrol? /*The query returns REP.*/

[:SENSe]:CORRection:IMPedance[:INPut][:MAGNitude]

Syntax
[:SENSe]:CORRection:IMPedance[:INPut][:MAGNitude] <enum>
[:SENSe]:CORRection:IMPedance[:INPut][:MAGNitude]?

Description
Sets or queries the input impedance. The unit is Ω.

Parameter

Name	Type	Range	Default
<enum>	Discrete	50|75	50

Remarks
If the output impedance of the system under test is 75 Ω, you should use a 75 Ω to 50 Ω adapter (option) supplied by RIGOL to connect the analyzer with the system under test, and then set the input impedance to 75 Ω.

Return Format
The query returns 50 or 75.

Example
:SENSe:CORRection:IMPedance:INPut:MAGNitude 75 /*Sets the input impedance to 75 Ω.*/
:SENSe:CORRection:IMPedance:INPut:MAGNitude? /*The query returns 75.*/

[:SENSe]:CORRection:SA[:RF]:GAIN

Syntax

[:SENSe]:CORRection:SA[:RF]:GAIN <rel_ampl>
[:SENSe]:CORRection:SA[:RF]:GAIN?

Description

Sets or queries the external gain.

Parameter

Name	Type	Range	Default
<rel_ampl>	Real	-120 dB to 120 dB	0 dB

Return Format

The query returns the external gain value in scientific notation. The unit is dB.

Example

:SENSe:CORRection:SA:RF:GAIN 20 /*Sets the external gain to 20 dB.*/
:SENSe:CORRection:SA:RF:GAIN? /*The query returns 2.0E +01.*/

[:SENSe]:CORRection:CSET<n>[:STATe]

Syntax

[:SENSe]:CORRection:CSET<n>[:STATe] <bool>
[:SENSe]:CORRection:CSET<n>[:STATe]?

Description

Sets or queries whether the correction function is enabled.

Parameter

Name	Type	Range	Default
<n>	Discrete	1|2|3|4	——
<bool>	Bool	OFF|ON|0|1	ON|1

Remarks

The parameter <n> sets the amplitude correction type. 1 indicates Antenna, 2 indicates Cable, 3 indicates Other, and 4 indicates User.

Return Format

The query returns 0 or 1.

Example

:SENSe:CORRection:CSET1:STATe 1 or :SENSe:CORRection:CSET1:STATe ON
/*Enables the antenna amplitude correction.*/
:SENSe:CORRection:CSET1:STATe? /*The query returns 1.*/

[:SENSe]:CORRection:CSET<n>:DESCription

Syntax
[:SENSe]:CORRection:CSET<n>:DESCription <string>
[:SENSe]:CORRection:CSET<n>:DESCription?

Description
Sets or queries the description of the amplitude correction.

Parameter

Name	Type	Range	Default
<n>	Discrete	1|2|3|4	——
<string>	ASCII String	The label can contain English letters and numbers, as well as some symbols.	——

Remarks
The parameter <n> sets the amplitude correction type. 1 indicates Antenna, 2 indicates Cable, 3 indicates Other, and 4 indicates User.

Return Format
The query returns the amplitude correction description in ASCII strings.

Example
:SENSe:CORRection:CSET1:DESCription Rigol /*Sets the description of the antenna amplitude correction to Rigol.*/
:SENSe:CORRection:CSET1:DESCription? /*The query returns Rigol.*/

[:SENSe]:CORRection:CSET<n>:COMMent

Syntax
[:SENSe]:CORRection:CSET<n>:COMMent <string>
[:SENSe]:CORRection:CSET<n>:COMMent?

Description
Sets or queries the comment of the amplitude correction.

Parameter

Name	Type	Range	Default
<n>	Discrete	1|2|3|4	——
<string>	ASCII String	The label can contain English letters and numbers, as well as some symbols.	——

Remarks
The parameter <n> sets the amplitude correction type. 1 indicates Antenna, 2 indicates Cable, 3 indicates Other, and 4 indicates User.

Return Format
The query returns the comment of the amplitude correction in strings.

Example
:SENSe:CORRection:CSET1:COMMent Rigol /*Sets the comment of the antenna amplitude correction to Rigol.*/
:SENSe:CORRection:CSET1:COMMent? /*The query returns Rigol.*/

[:SENSe]:CORRection:CSET<n>:X:SPACing

Syntax
[:SENSe]:CORRection:CSET<n>:X:SPACing <enum>
[:SENSe]:CORRection:CSET<n>:X:SPACing?

Description
Sets or queries the type of the frequency interpolation of the specified amplitude correction.

Parameter

Name	Type	Range	Default
<n>	Discrete	1|2|3|4	——
<enum>	Discrete	LINear|LOG	LINear

Remarks
The parameter <n> sets the amplitude correction type. 1 indicates Antenna, 2 indicates Cable, 3 indicates Other, and 4 indicates User.
LINear: linear type.
LOG: log type.

Return Format
The query returns LIN or LOG.

Example
:SENSe:CORRection:CSET1:X:SPACing LOG
/*Sets the type of the frequency interpolation of the antenna amplitude correction is LOG.*/
:SENSe:CORRection:CSET1:X:SPACing ?/*The query returns LOG.*/

[:SENSe]:CORRection:CSET<n>:DATA

Syntax
[:SENSe]:CORRection:CSET<n>:DATA <string>
[:SENSe]:CORRection:CSET<n>:DATA?

Description
Sets or queries the amplitude correction points in the table.

Parameter

Name	Type	Range	Default
<n>	Discrete	1|2|3|4	——

<string>	ASCII String	The label can contain English letters and numbers, as well as some symbols.	——

Remarks
The parameter <n> sets the amplitude correction type. 1 indicates Antenna, 2 indicates Cable, 3 indicates Other, and 4 indicates User.

Return Format
The query returns the data in the amplitude correction table in ASCII strings.

Example
:SENSe:CORRection:CSET1:DATA 5,2
/*Sets a group of data in the antenna amplitude correction table to 5 Hz frequency and 2 dB amplitude.*/
:SENSe:CORRection:CSET1:DATA? /*The query returns 5.000000000000000e+00,2.000000.*/

[:SENSe]:CORRection:CSET<n>:DATA:MERG

Syntax
[:SENSe]:CORRection:CSET<n>:DATA:MERG <string>

Description
Merges the amplitude correction.

Parameter

Name	Type	Range	Default
<n>	Discrete	1|2|3|4	——
<string>	ASCII String	The label can contain English letters and numbers, as well as some symbols.	——

Remarks
The parameter <n> sets the amplitude correction type. 1 indicates Antenna, 2 indicates Cable, 3 indicates Other, and 4 indicates User.

[:SENSe]:CORRection:CSET<n>:DELete

Syntax
[:SENSe]:CORRection:CSET<n>:DELete

Description
Deletes the single amplitude correction data.

Remarks
The parameter <n> sets the amplitude correction type. 1 indicates Antenna, 2 indicates Cable, 3 indicates Other, and 4 indicates User.

[:SENSe]:CORRection:CSET:ALL:DELete

Syntax
[:SENSe]:CORRection:CSET:ALL:DELete

Description
Delete all amplitude correction data.

[:SENSe]:CORRection:CSET:ALL[:STATe]

Syntax
[:SENSe]:CORRection:CSET:ALL[:STATe] <bool>
[:SENSe]:CORRection:CSET:ALL[:STATe]?

Description
Sets or queries whether to apply all the corrections.

Parameter

Name	Type	Range	Default
<bool>	Bool	OFF|ON|0|1	ON|1

Return Format
The query returns 0 or 1.

Example
:SENSe:CORRection:CSET:ALL:STATe 1 or :SENSe:CORRection:CSET:ALL:STATe ON
/*Enables applying all the corrections.*/
:SENSe:CORRection:CSET:STATe? /*The query returns 1.*/

[:SENSe]:CORRection:CSET[:ID]

Syntax
x[:SENSe]:CORRection:CSET>:ID <enum>
[:SENSe]:CORRection:CSET[:ID]?

Description
Sets or queries the type of the amplitude correction.

Parameter

Name	Type	Range	Default
<enum>	Discrete	OFF|ANTEnna|CABLe|OTHEr|USER	——

Remarks
ANTEnna: Antenna.
CABLe: Cable.
OTHEr: Other.

USER: indicates user-defined.

Return Format
The query returns ANTE, CABL, OTHE, or USER.

Example
:SENSe:CORRection:CSET:ID OTHEr /*Sets the type of the amplitude correction to OTHEr.*/
:SENSe:CORRection:CSET:ID? /*The query returns OTHE.*/

[:SENSe]:CARRier:THReshold:MINimum

Syntax
[:SENSe]:CARRier:THReshold:MINimum <amptd>
[:SENSe]:CARRier:THReshold:MINimum?

Description
Sets or queries the min. carrier level.

Parameter

Name	Type	Range	Default
<amptd>	Real	-200 dBm to 30 dBm	-50 dBm

Remarks
When the carrier power is below this value, the measurement may be incorrect.

Return Format
The query returns the min. carrier level in scientific notation.

Example
:SENS:CARR:THR:MIN -40 /*Sets the min. carrier level to -40 dBm.*/
:SENS:CARR:THR:MIN? /*The query returns -4.0E+01.*/

[:SENSe]:FREQuency:CARRier

Syntax
[:SENSe]:FREQuency:CARRier <freq>
[:SENSe]:FREQuency:CARRier?

Description
Sets or queries the nominal frequency of the phase noise measurement.

Parameter

Name	Type	Range	Default
<freq>	Real	5 kHz to F_{max}	1 GHz

Remarks

- Fmax is determined by the instrument model.
- When the signal track is enabled, the nominal frequency will be automatically adjusted based on the signal.

Return Format
The query returns the nominal frequency in scientific notation. The unit is Hz.

Example
:SENS:FREQ:CARR 20 /*Sets the nominal frequency to 20 Hz.*/
:SENS:FREQ:CARR? /*The query returns 2.0E+01.*/

[:SENSe]:FREQuency:CARRier:TRACk[:STATe]

Syntax
[:SENSe]:FREQuency:CARRier:TRACk[:STATe] <bool>
[:SENSe]:FREQuency:CARRier:TRACk[:STATe]?

Description
Sets or queries whether to enable the signal track function.

Parameter

Name	Type	Range	Default
<bool>	Bool	ON|OFF|1|0	OFF|0

Remarks
When the signal track function is enabled, the analyzer will track the signal frequency changes automatically.

Example
:FREQ:CARR:TRAC ON /*Enables the signal track function.*/
:FREQ:CARR:TRAC? /*The query returns 1.*/

[:SENSe]:FREQuency:VERify:TOLerance:ABSolute

Syntax
[:SENSe]:FREQuency:VERify:TOLerance:ABSolute <freq>
[:SENSe]:FREQuency:VERify:TOLerance:ABSolute?

Description
Sets or queries the absolute frequency tolerance in the signal verification.

Parameter

Name	Type	Range	Default
<freq>	Real	1 Hz to 1 GHz	1 kHz

Return Format
The query returns the absolute frequency tolerance in scientific notation. Its unit is Hz.

Example
:SENS:FREQ:VER:TOL:ABS 1000 /*Sets the absolute frequency tolerance to 1 kHz.*/
:SENS:FREQ:VER:TOL:ABS? /*The query returns 1.0E+03.*/

[:SENSe]:FREQuency:VERify:TOLerance[:RELative]

Syntax
[:SENSe]:FREQuency:VERify:TOLerance[:RELative] <percent>
[:SENSe]:FREQuency:VERify:TOLerance[:RELative]?

Description
Sets or queries the relative frequency tolerance in the signal verification, expressed in percentage.

Parameter

Name	Type	Range	Default
<percent>	Real	1% to 100%	10%

Return Format
The query returns the relative frequency tolerance in scientific notation. The unit is %.

Example
:SENS:FREQ:VER:TOL: 0.5 /*Sets the relative frequency tolerance to 0.5%.*/
:SENS:FREQ:VER:TOL? /*The query returns 5.0E-01.*/

[:SENSe]:FREQuency:VERify[:STATe]

Syntax
[:SENSe]:FREQuency:VERify[:STATe] <bool>
[:SENSe]:FREQuency:VERify[:STATe]?

Description
Sets or queries whether to enable the frequency verification.

Parameter

Name	Type	Range	Default
<bool>	Bool	ON|OFF|1|0	OFF|0

Return Format
The query returns 1 or 0.

Example
:SENS:FREQ:VER ON /*Enables the frequency verification.*/
:SENS:FREQ:VER? /*The query returns 1.*/

[:SENSe]:LPLot:FREQuency:OFFSet:STARt

Syntax

[:SENSe]:LPLot:FREQuency:OFFSet:STARt <freq>
[:SENSe]:LPLot:FREQuency:OFFSet:STARt?

Description

Sets or queries the start frequency offset for the log plot graph.

Parameter

Name	Type	Range	Default
<freq>	Real	10 Hz to (max. stop frequency offset/10)	30 MHz

Remarks

- The max. start frequency offset is (max. stop frequency offset/10). If the start frequency offset is greater than the stop frequency offset, then the stop frequency offset is automatically set to (start frequency offset x 10).
- The frequency offset value must be an integer number of decades apart. If the value is an integer number of decades, it will be automatically adjusted to the effective value that is closest to the proper value.

Return Format

The query returns the start offset frequency in scientific notation. The unit is Hz.

Example

:SENS:LPL:FREQ:OFFS:STAR 100 /*Sets the start frequency offset for the log plot graph to 100 Hz.*/
:SENS:LPL:FREQ:OFFS:STAR? /*The query returns 1.0E+02.*/

[:SENSe]:LPLot:FREQuency:OFFSet:STOP

Syntax

[:SENSe]:LPLot:FREQuency:OFFSet:STOP <freq>
[:SENSe]:LPLot:FREQuency:OFFSet:STOP?

Description

Sets or queries the stop frequency offset for the log plot graph.

Parameter

Name	Type	Range	Default
<freq>	Real	Right sideband: stop frequency offset < Fmax - nominal frequency Left sideband: stop frequency offset < nominal frequency - 5 kHz	30 MHz

Remarks

- The max. stop frequency offset is (max. start frequency offset x 10). If the stop frequency offset is smaller than the start frequency offset, then the stop frequency offset is automatically set to (start frequency offset/10).
- The frequency offset value must be an integer number of decades apart. If the value is an integer number of decades, it will be automatically adjusted to the effective value that is closest to the proper value.

Return Format

The query returns the stop frequency offset in scientific notation. The unit is Hz.

Example

:SENS:LPL:FREQ:OFFS:STOP 1000 /*Sets the stop frequency offset for the log plot graph to 1 kHz.*/
:SENS:LPL:FREQ:OFFS:STOP? /*The query returns 1.0E+03.*/

[:SENSe]:LPLot:CARRier:MEASure:DisPlay

Syntax

[:SENSe]:LPLot:CARRier:MEASure:DisPlay <state>
[:SENSe]:LPLot:CARRier:MEASure:DisPlay?

Description

Sets or queries the display of the carrier power measurement results.

Parameter

Name	Type	Range	Default
<state>	Bool	OFF|ON|0|1	ON|1

Return Format

The query returns 1 or 0.

Example

:SENSe:LPLot:CARRier:MEASure:DisPlay 1 or :SENSe LPLot:CARRier:MEASure:DisPlay ON
/*Enables the display of the carrier power measurement results.*/
SENSe:LPLot:CARRier:MEASure:DisPlay?/*The query returns 1.*/

[:SENSe]:LPLot:OFFSet:SIDE

Syntax

[:SENSe]:LPLot:OFFSet:SIDE <side>
[:SENSe]:LPLot:OFFSet:SIDE?

Description

Sets or queries the offset side of the phase noise measurement.

Parameter

Name	Type	Range	Default
<side>	Discrete	USBand|LSBand	USBand

Remarks
USBand: upper sideband.
LSBand: lower sideband.

Return Format
The query returns USBand or LSBand.

Example
:SENS:LPL:OFFS:SIDE USB /*Sets the offset side to USBand.*/
:SENS:LPL:OFFS:SIDE? /*The query returns USBand.*/

[:SENSe]:LPLot:DETector

Syntax
[:SENSe]:LPLot:DETector <type>
[:SENSe]:LPLot:DETector?

Description
Sets or queries the detector type of the phase noise measurement.

Parameter

Name	Type	Range	Default
<type>	Discrete	AVERage|SAMPle|RAVerage	AVERage

Remarks
AVERage: indicates the voltage average detector.
SAMPle: indicates the sample detector.
RAVerage: indicates the RMS average detector.

Return Format
The query returns SAMP, AVER, or RAV.

Example
:SENS:LPL:DET AVER /*Sets the detector type to AVERage.*/
:SENS:LPL:DET? /*The query returns AVER.*/

[:SENSe]:LPLot:SMOoth

Syntax
[:SENSe]:LPLot:SMOoth <percent>
[:SENSe]:LPLot:SMOoth?

Description

Sets or queries smoothing percentage. This value controls the smoothing degree of the smoothed trace in the log plot graph.

Parameter

Name	Type	Range	Default
<percent>	Real	1% to 50%	2%

Remarks

- The greater the value, the smoother the trace.
- Appropriate smoothing reduces noise, but excessive smoothing can cause the loss of signal details.

Return Format

The query returns the smoothing value in scientific notation. The unit is %.

Example

:SENS:LPL:SMO: 5 /*Sets the smoothing value to 5%.*/
:SENS:LPL:SMO? /*The query returns 5.0E+00.*/

[:SENSe]:LPLot:CANCellation[:STATe]

Syntax

[:SENSe]:LPLot:CANCellation[:STATe] <bool>
[:SENSe]:LPLot:CANCellation[:STATe]?

Description

Sets or queries whether to enable the noise cancellation feature.

Parameter

Name	Type	Range	Default
<bool>	Bool	ON|OFF|1|0	OFF|0

Remarks

- When the noise cancellation feature is enabled, the noise floor of the measurement system can be removed.
- Before noise cancellation is used, ensure that the DANL of the instrument have been measured and stored in a reference trace.

Return Format

The query returns 0 or 1.

Example

:SENS:LPL:CANC ON /*Enables the noise cancellation feature.*/
:SENS:LPL:CANC? /*The query returns 1.*/

[:SENSe]:LPLot:CANCellation:TRACe

Syntax

[:SENSe]:LPLot:CANCellation:TRACe <trace>
[:SENSe]:LPLot:CANCellation:TRACe?

Description
Sets or queries the reference trace for the noise cancellation feature in the log plot graph.

Parameter

Name	Type	Range	Default
<trace>	Integer	1|2|3	1

Remarks
The reference trace is used as the reference DANL floor trace for the noise cancellation feature in the log plot graph.

Return Format
The query returns the reference trace number in integer.

Example
:SENS:LPL:CANC:TRAC 2 /*Sets the reference trace to Trace 2.*/
:SENS:LPL:CANC:TRAC? /*The query returns 2.*/

[:SENSe]:LPLot:CANCellation:DELTa

Syntax
[:SENSe]:LPLot:CANCellation:DELTa <threshold>
[:SENSe]:LPLot:CANCellation:DELTa?

Description
Sets or queries the threshold for the noise cancellation feature in the log plot graph.

Parameter

Name	Type	Range	Default
<threshold>	Real	0.001 dB to 5 dB	0.01 dB

Remarks
This value determines the minimum delta between the reference and pre-cancellation traces that must exist before any cancellation is performed.

Return Format
The query returns the threshold in scientific notation.

Example
:SENS:LPL:CANC:DELT 15 /*Sets the threshold to 15 dB.*/
:SENS:LPL:CANC:DELT? /*The query returns 1.5E+01.*/

[:SENSe]:POWer:RLEVel

Syntax
[:SENSe]:POWer:RLEVel <level>
[:SENSe]:POWer:RLEVel?

Description
Sets or queries the nominal level in the signal verification.

Parameter

Name	Type	Range	Default
<level>	Real	-200 dBm to 200 dBm	0 dBm

Return Format
The query returns the nominal level in scientific notation. The unit is dBm.

Example
:SENS:POW:RLEV -10 /*Sets the nominal level to -10 dBm.*/
:SENS:POW:RLEV? /*The query returns -1.0E+01.*/

[:SENSe]:POWer:RLEVel:VERify:TOLerance

Syntax
[:SENSe]:POWer:RLEVel:VERify:TOLerance <tolerance>
[:SENSe]:POWer:RLEVel:VERify:TOLerance?

Description
Sets or queries the absolute level tolerance.

Parameter

Name	Type	Range	Default
<tolerance>	Real	0.1 dB to 100 dB	10 dB

Return Format
The query returns the absolute level tolerance in scientific notation. The unit is dB.

Example
:SENS:POW:RLEV:VER:TOL 5 /*Sets the absolute level tolerance to 5 dB.*/
:SENS:POW:RLEV:VER:TOL? /*The query returns 5.0E+00.*/

[:SENSe]:POWer:RLEVel:VERify[:STATe]

Syntax
[:SENSe]:POWer:RLEVel:VERify[:STATe] <bool>
[:SENSe]:POWer:RLEVel:VERify[:STATe]?

Description
Sets or queries whether to enable the level verification.

Parameter

Name	Type	Range	Default
<bool>	Bool	ON|OFF|1|0	OFF|0

Return Format
The query returns 1 or 0.

Example
:SENS:POW:RLEV:VER ON /*Enables the level verification.*/
:SENS:POW:RLEV:VER? /*The query returns 1.*/

[:SENSe]:POWer[:RF]:ATTenuation

Syntax
[:SENSe]:POWer[:RF]:ATTenuation <real>
[:SENSe]:POWer[:RF]:ATTenuation?

Description
Sets or queries the attenuation of the RF front-end attenuator.

Parameter

Name	Type	Range	Default
<real>	Integer	0 dB to 40 dB	10 dB

Return Format
The query returns the attenuation in integer. The unit is dB.

Example
:SENSe:POWer:RF:ATTenuation 20 /*Sets the attenuation value to 20 dB.*/
:SENSe:POWer:RF:ATTenuation? /*The query returns 20.*/

[:SENSe]:POWer[:RF]:ATTenuation:AUTO

Syntax
[:SENSe]:POWer[:RF]:ATTenuation:AUTO <bool>
[:SENSe]:POWer[:RF]:ATTenuation:AUTO?

Description
Enables or disables the auto setting mode of the input attenuation.
Queries the status of the auto setting mode of the input attenuation.

Parameter

Name	Type	Range	Default
<bool>	Bool	OFF|ON|0|1	ON|1

Return Format
The query returns 0 or 1.

Example
:SENSe:POWer:RF:ATTenuation:AUTO OFF or :SENSe:POWer:RF:ATTenuation:AUTO 0
/*Disables the auto attenuation mode.*/
:SENSe:POWer:RF:ATTenuation:AUTO? /*The query returns 0.*/

[:SENSe]:POWer[:RF]:GAIN[:STATe]

Syntax
[:SENSe]:POWer[:RF]:GAIN[:STATe] <bool>
[:SENSe]:POWer[:RF]:GAIN[:STATe]?

Description
Sets or queries the on/off status of the preamplifier (PA).

Parameter

Name	Type	Range	Default
<bool>	Bool	OFF|ON|0|1	OFF|0

Return Format
The query returns 0 or 1.

Example
:SENSe:POWer:RF:GAIN:STATe ON or :SENSe:POWer:RF:GAIN:STATe 1
/*Enables the preamplifier (PA).*/
:SENSe:POWer:RF:GAIN:STATe? /*The query returns 1.*/

[:SENSe]:POWer[:RF]:MIXer:RANGe[:UPPer]

Syntax
[:SENSe]:POWer[:RF]:MIXer:RANGe[:UPPer] <ampl>
[:SENSe]:POWer[:RF]:MIXer:RANGe[:UPPer]?

Description
Sets or queries the maximum level of the input mixer.

Parameter

Name	Type	Range	Default
<ampl>	Real	-50 dBm to -10 dBm	-10 dBm

Return Format
The query returns the maximum level of the input mixer in scientific notation. The unit is dBm.

Example
:SENSe:POWer:RF:MIXer:RANGe:UPPer -20 /*Sets the max. mixer level to -20 dBm*/
:SENSe:POWer:RF:MIXer:RANGe:UPPer? /*The query returns -2.0E+01.*/

[:SENSe]:SWEep:SVFailed

Syntax

[:SENSe]:SWEep:SVFailed <action>
[:SENSe]:SWEep:SVFailed?

Description

Sets or queries the action taken when the signal verification fails.

Parameter

Name	Type	Range	Default
<action>	Discrete	OFF|ON	OFF

Remarks

- OFF: restarts the measurement when signal verification fails.
- ON: stops the measurement immediately when signal verification fails.

Return Format

The query returns ON or OFF.

Example

:SENS:SWE:SVF ON /*Stops the measurement immediately after the signal verification fails.*/
:SENS:SWE:SVF? /*The query returns ON.*/

:SYSTem Commands

:SYSTem:BEEPer

Syntax

:SYSTem:BEEPer <bool>
:SYSTem:BEEPer?

Description

Enables or disables the beeper; queries the on/off status of the beeper.

Parameter

Name	Type	Range	Default
<bool>	Bool	OFF|ON|0|1	OFF|0

Return Format

The query returns 0 or 1.

Example

:SYSTem:BEEPer ON or :SYSTem:BEEPer 1 /*Enables the beeper.*/
:SYSTem:BEEPer? /*The query returns 1.*/

:SYSTem:BRIGhtness

Syntax
:SYSTem:BRIGhtness <brightness>
:SYSTem:BRIGhtness?

Description
Sets or queries the screen brightness.

Parameter

Name	Type	Range	Default
<brightness>	Integer	0% to 100%	80%

Return Format
The query returns the screen brightness in integer.

Example
:SYSTem:BRIGhtness 50 /*Sets the screen brightness to 50%.*/
:SYSTem:BRIGhtness? /*The query returns 50.*/

:SYSTem:DATE

Syntax
:SYSTem:DATE <year>,<month>,<day>
:SYSTem:DATE?

Description
Sets or queries the system date of the instrument.

Parameter

Name	Type	Range	Default
<year>	ASCII String	2000 to 2099	——
<month>	ASCII String	01 to 12	——
<day>	ASCII String	01 to 31	——

Return Format
The query returns the current system date in the format of "YYYY-MM-DD".

Example
:SYSTem:DATE 2017,11,16 /*Sets the system date of the instrument to Nov. 16th, 2017.*/
:SYSTem:DATE? /*The query returns 2017-11-16.*/

:SYSTem:GPIB

Syntax
:SYSTem:GPIB <adr>
:SYSTem:GPIB?

Description
Sets or queries the GPIB address.

Parameter

Name	Type	Range	Default
<adr>	Integer	1 to 30	1

Return Format
The query returns an integer ranging from 1 to 30.

Example
:SYSTem:GPIB 2 /*Sets the GPIB address to 2.*/
:SYSTem:GPIB? /*The query returns 2.*/

:SYSTem:LANGuage

Syntax
:SYSTem:LANGuage <enum>

Description
Sets or queries the system language of the instrument.

Parameter

Name	Type	Range	Default
<enum>	Discrete	SCHinese|TCHinese|KORean|JAPanese|ENGLish|GERMan|PORTuguese|POLish|FRENch|RUSSian|SPAN|THAI|INDonesian	ENGLish

Remarks
SCHinese: Simplified Chinese.
TCHinese: Traditional Chinese.
KORean: Korean.
JAPanese: Japanese.
ENGLish: English.
GERMan: German.
PORTuguese: Portuguese.
POLish: Polish.
FRENch: French.
RUSSian: Russian.
SPAN: Spanish.
THAI: Thai.
INDonesian: Indonesian.

Return Format
The query returns SCH, TCH, KOR, JAP, ENGL, GERM, PORT, POL, FREN, RUSS, SPAN, THAI, or IND.

Example
:SYSTem:LANGuage ENGLish /*Sets the system language to ENGLish.*/
:SYSTem:LANGuage? /*The query returns ENGL.*/

:SYSTem:LOCKed

Syntax
:SYSTem:LOCKed<bool>
:SYSTem:LOCKed?

Description
Locks or unlocks the keypad.
Queries whether the keypad is locked or not.

Parameter

Name	Type	Range	Default
<bool>	Bool	OFF|ON|0|1	OFF|0

Return Format
The query returns 0 or 1.

Example
:SYSTem:LOCKed 1 or :SYSTem:LOCKed ON /*Disables the touch screen operation.*/
:SYSTem:LOCKed? /*The query returns 1.*/

:SYSTem:OPTion:INSTall

Syntax
:SYSTem:OPTion:INSTall <option info>@<license info>

Description
Installs and activates the specified option.

Parameter

Name	Type	Range	Default
<option info>	ASCII String	——	——
<license info>	ASCII String	——	——

Remarks
The parameter <option info> indicates the order number of the option. <license info> indicates the serial number of the option.

Example
:SYSTem:OPTion:INSTall RSA6000-PA@8AD12B8EBC5DF492D1D4289B7CBA5B6150BF6F5D752D645C36D74530B05F39B49C461B23A50D6C94A34E06782AC4380070B0D1A86BA84E02768391FFD70C2103
/*Installs the option RSA6000-PA.*/

:SYSTem:OPTion:STATus?

Syntax
:SYSTem:OPTion:STATus? <option name>

Description
Queries whether the specified option is activated.

Parameter

Name	Type	Range	Default
<option name>	Discrete	BND|PA|P08|P14|P26|B200|RB200|EMI|T08|ADM|AWK|VSA|UBW|PNM|PMA|LORA|ZIGBEE|BLE_TX	——

Return Format
The query returns 0 (not activated) or 1 (activated).

Example
:SYSTem:OPTion:STATus? RSA6000-PA /*Queries whether the RSA6000-PA option is activated.*/

:SYSTem:OPTion:UNINstall

Syntax
:SYSTem:OPTion:UNINstall

Description
Uninstalls all the official options.

:SYSTem:PON

Syntax
:SYSTem:PON <power_on>
:SYSTem:PON?

Description
Sets or queries the setting type the instrument recalls at power-on.

Parameter

Name	Type	Range	Default
<power_on>	Discrete	DEFault|LATest	PRESet

Remarks
DEFault: indicates preset settings, including factory mode and 6 user-defined settings.
LATest: last settings.

Return Format

The query returns DEF or LAT.

Example
:SYSTem:PON LAT /*/*Sets the instrument to recall last settings at next power-on.*/
:SYSTem:PON? /*The query returns LAT.*/

:SYSTem:RESet

Syntax
:SYSTem:RESet

Description
Restarts the instrument.

:SYSTem:PRESet

Syntax
:SYSTem:PRESet

Description
Restores the instrument to its preset settings.

:SYSTem:PSTatus

Syntax
:SYSTem:PSTatus <status>
:SYSTem:PSTatus?

Description
Sets or queries the on/off state of the power switch.

Parameter

Name	Type	Range	Default
<status>	Discrete	DEFault|OPEN	DEFault

Remarks
DEFault: switches off.
OPEN: switches on.

Return Format
The query returns DEF or OPEN.

Example
:SYSTem:PSTatus OPEN /*Sets the power switch to On.*/
:SYSTem:PSTatus? /*The query returns OPEN.*/

:SYSTem:PWRD

Syntax
:SYSTem:PWRD

Description
Sets the instrument to power off.

:SYSTem:STIME

Syntax
:SYSTem:STIME <bool>
:SYSTem:STIME?

Description
Sets or queries whether to display the system time.

Parameter

Name	Type	Range	Default
<bool>	Bool	OFF|ON|0|1	OFF|0

Return Format
The query returns 0 or 1.

Example
:SYSTem:STIME 1 or :SYSTem:STIME ON /*Sets to display the system date.*/
:SYSTem:STIME? /*The query returns 1.*/

:SYSTem:TIME

Syntax
:SYSTem:TIME <hour>,<minute>,<second>
:SYSTem:TIME?

Description
Sets or queries the system time of the instrument.

Parameter

Name	Type	Range	Default
<hour>	ASCII String	00 to 23	——
<minute>	ASCII String	00 to 59	——
<second>	ASCII String	00 to 59	——

Return Format
The query returns the current system time in the format of "HH:MM:SS".

Example

:SYSTem:TIME 15,10,30 /*Sets the system time of the instrument to 15:10:30.*/
:SYSTem:TIME? /*The query returns 15:10:30.*/

:SYSTem:VERSion?

Syntax
:SYSTem:VERSion?

Description
Queries the version number of the SCPI used by the system.

:TRACe Commands

:TRACe:LPLot:SELect

Syntax
:TRACe:LPLot:SELect <trace>
:TRACe:LPLot:SELect?

Description
Sets or queries the current active trace.

Parameter

Name	Type	Range	Default
<trace>	Discrete	TRACe1|TRACe2|TRACe3	TRACe1

Return Format
The query returns TRACe1, TRACe2, or TRACe3.

Example
:TRAC:LPL:SEL TRAC1 /*Selects Trace 1 as the active trace.*/
:TRAC:LPL:SEL? /*The query returns TRACe1.*/

:TRACe<n>:LPLot:TYPE

Syntax
:TRACe<n>:LPLot:TYPE <type>
:TRACe<n>:LPLot:TYPE?

Description
Sets or queries the type of the specified trace.

Parameter

Name	Type	Range	Default
<n>	Discrete	1|2|3	——
<type>	Discrete	RAW|SMOothed|VIEW|BLANk	RAW

Remarks

RAW: raw trace. Displays the raw trace. The raw trace presents the directly acquired measurement data, without any smoothing applied.
SMOothed: smoothed trace. Displays the smoothed trace, with the spurious removed.
VIEW: displays the trace. Only displays the trace and not updates the trace.
BLANk: blank trace. Not displays the trace. When you enable multiple traces (Trace1, Trace2, and Trace3) at the same time, you can set the unwanted traces to Blank.
The default trace type for Trace 1 is Raw; the default trace type for Trace 2 is Smoothed; and the default trace type for Trace 3 is Blank.

Return Format

The query returns RAW, SMO, VIEW, or BLAN.

Example

```
:TRAC1:LPL:TYPE RAW                /*Sets Trace 1 as the raw trace.*/
:TRAC1:LPL:TYPE?                   /*The query returns RAW.*/
```

:TRACe<n>:LPLot:HOLD

Syntax

:TRACe<n>:LPLot:HOLD <hold>
:TRACe<n>:LPLot:HOLD?

Description

Sets or queries the trace hold mode.

Parameter

Name	Type	Range	Default
<n>	Discrete	1|2|3	1
<hold>	Discrete	OFF|MAXHold|MINHold	OFF

Remarks

- OFF: disables the trace hold. The trace restores to the updated state.
- MAXHold: indicates the maximum hold. Maintains and displays a max hold trace, which represents the maximum data value on a point-by-point basis. When a new maximum value is generated, data will be updated, and the newly updated maximum value prevails.
- MINHold: indicates the minimum hold. Maintains and displays a min hold trace, which represents the minimum data value on a point-by-point basis. When a new minimum value is generated, data will be updated, and the newly updated minimum value prevails.

Return Format

The query returns OFF, MAXH, or MINH.

Example

```
:TRAC1:LPL:HOLD MAXH               /*Sets Trace 1 to Max Hold.*/
:TRAC1:LPL:HOLD?                   /*The query returns MAXH.*/
```

Chapter 4 Programming Examples

This chapter lists some programming examples to illustrate how to use commands to realize the common functions of the spectrum analyzer in the development environments such as Visual C++ 6.0, Visual Basic 6.0, and LabVIEW 2010. Also, the chapter lists some examples to illustrate how to control the spectrum analyzer to realize the common functions in Linux operating system. These examples are programmed based on NI-VISA library.

NI-VISA (National Instrument-Virtual Instrument Software Architecture), developed by NI (National Instrument), provides an advanced programming interface to communicate with various instruments through their bus lines. NI-VISA enables you to realize the communication between the analyzer and PC through instrument buses (such as USB). VISA defines a set of software commands, with which users can control the instrument without knowing the working principle of the interface buses. For details, refer to the help information of NI-VISA.

Programming Instructions

This section introduces the problems that might occur during the programming process as well as their solutions. If these problems occur, please resolve them according to the corresponding instructions.

1. When you build a working environment via the network, it is recommended that you build a pure local area network.

2. If the local area network environment is complicated (e.g. many devices and broadcast messages exist), it is recommended that you add some fault tolerance during the programming process. For the details, refer to the instrument write/read operations with exception handling functions "InstrWriteEx()" and "InstrReadEx()" in "*Visual C++ 6.0* Programming Example".

3. The Socket programming port No. of this device is 5025.

Programming Preparations

The programming preparations introduced here are only applicable to programming by using Visual C++ 6.0, Visual Basic 6.0, and LabVIEW 2010 development tools in Windows operating system. For the preparations of programming in Linux operating system, refer to "*Linux* Programming Example" in "*Programming* Preparations".

First, check whether your PC has installed NI's VISA library. If not, download it from *http://www.ni.com/visa/*. In this manual, the default installation path is C:\Program Files\IVI Foundation\VISA.

Connect spectrum analyzer to the PC via the USB interface of the analyzer. Use the USB cable to connect the USB DEVICE interface on the rear panel of the analyzer to the USB interface of the PC.

After the analyzer is connected to the PC properly, start the analyzer. In this case, "Found New Hardware Wizard" dialog box appears on the PC. Please install "USB Test and Measurement Device (IVI)" according to the wizard.

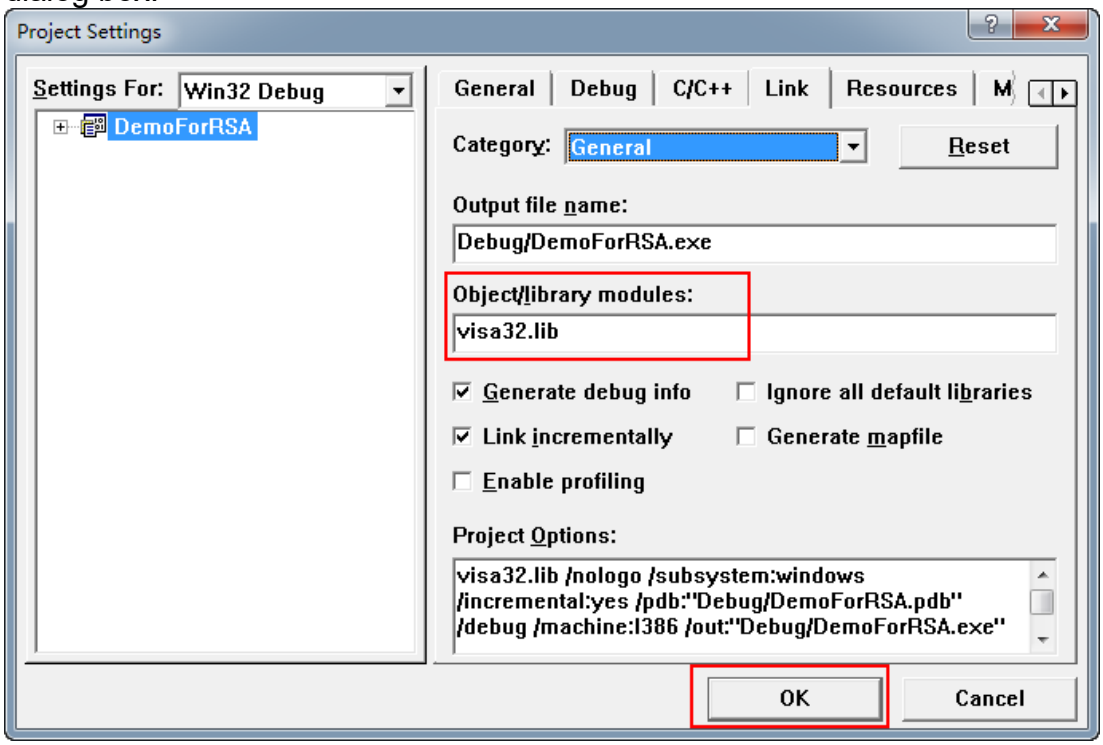
By now, the programming preparations are complete. The following parts will make a detailed introduction about the programming instances in the Visual C++ 6.0, Visual Basic 6.0, and LabVIEW 2010 development environment.

Visual C++ 6.0 Programming Example

Enter the Visual C++6.0 programming environment, and perform the following procedures.

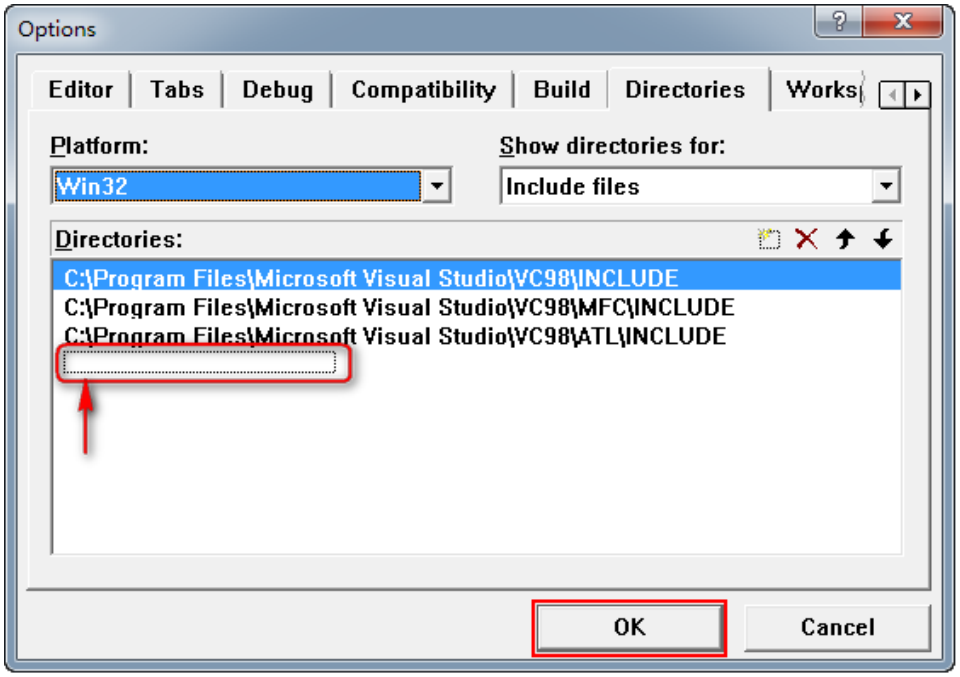
1. Create a MFC project based on a dialog box and name it "DemoForDSA" in this example.

2. Click **Project** → **Settings** to open the Project Setting dialog box. In the dialog box, click the **Link** tab, add "visa32.lib" under **Object/library modules**, then click **OK** to close the dialog box.



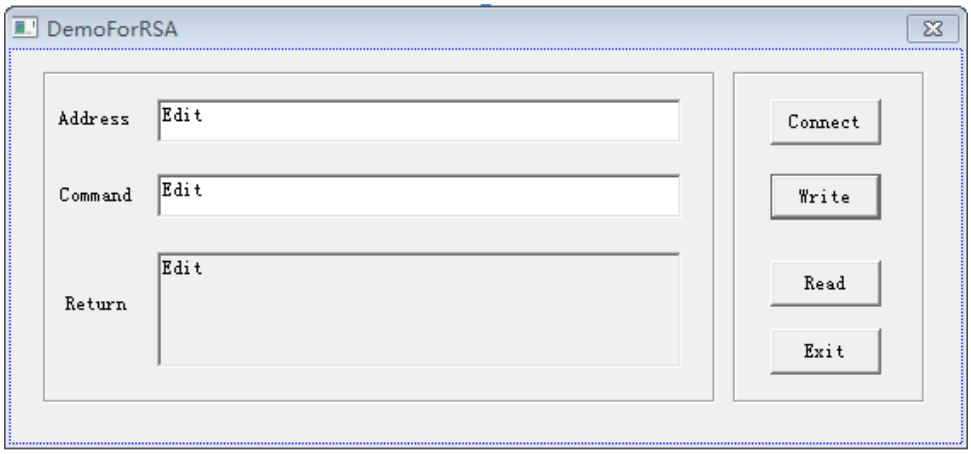
3. Click **Tools** → **Options** to open the Options dialog box. Then click the **Directories** tab. Select Include files from the drop-down list under **Show directories for**. Double click the empty space under **Directories** to enter the specified path of Include files: C:\Program Files\IVI Foundation\VISA\WinNT\include. Click **OK** to close the dialog box. Select Library files from the drop-down list under Show directories for. Double click the empty space under Directories to enter the specified path of Library files：C:\Program Files\IVI Foundation\VISA\WinNT\lib\msc. Click OK to close the dialog box.

   Note: The two paths added here are related to the installation path of NI-VISA on your PC. By default, NI-VISA is installed under C:\Program Files\IVI Foundation\VISA.



By now, VISA library has been added.

4. Add the **Text**, **Edit**, and **Button** controls. The layout interface for adding controls is as follows:



5. Add the control variables.
   Click **View → ClassWizard**, and then click on the "Member Variables" tab to add the following three variables:
   Instrument address: CString m_strInstrAddr
   Command: CString m_strCommand
   Returned value: CString m_strResult

6. Encapsulate the read and write operations of VISA.
   1) Encapsulate the write operation of VISA for easier operation.

```
bool CDemoForRSADlg::InstrWrite(CString strAddr, CString strContent) //Write operation
 {
   ViSession defaultRM,instr;
```

```
    ViStatus status;
    ViUInt32 retCount;
    char * SendBuf = NULL;
    char * SendAddr = NULL;
    bool bWriteOK = false;
    CString str;

    // Change the address's data style from CString to char*
    SendAddr = strAddr.GetBuffer(strAddr.GetLength());
    strcpy(SendAddr,strAddr);
    strAddr.ReleaseBuffer();

    // Change the command's data style from CString to char*
    SendBuf = strContent.GetBuffer(strContent.GetLength());
    strcpy(SendBuf,strContent);
    strContent.ReleaseBuffer();

    //Open a VISA resource
    status = viOpenDefaultRM(&defaultRM);
    if (status < VI_SUCCESS)
    {
        AfxMessageBox("No VISA resource was opened!");
        return false;
     }

    status = viOpen(defaultRM, SendAddr, VI_NULL, VI_NULL, &instr);

    //Write command to the instrument
    status = viWrite(instr, (unsigned char *)SendBuf, strlen(SendBuf), &retCount);

    //Close the system
    status = viClose(instr);
    status = viClose(defaultRM);

    return bWriteOK;
  }
```

2) Encapsulate the read operation of VISA for easier operation.

```
bool CDemoForRSADlg::InstrRead(CString strAddr, CString *pstrResult) //Read operation
{
  ViSession defaultRM,instr;
  ViStatus status;
  ViUInt32 retCount;
  char * SendAddr = NULL;
  unsigned char RecBuf[MAX_REC_SIZE];
  bool bReadOK = false;
  CString str;

  // Change the address's data style from CString to char*
```

```
SendAddr = strAddr.GetBuffer(strAddr.GetLength());
strcpy(SendAddr,strAddr);
strAddr.ReleaseBuffer();

memset(RecBuf,0,MAX_REC_SIZE);

//Open a VISA resource
status = viOpenDefaultRM(&defaultRM);
if (status < VI_SUCCESS)
{
    // Error Initializing VISA...exiting
    AfxMessageBox("No VISA resource was opened!");
    return false;
 }

//Open the instrument
status = viOpen(defaultRM, SendAddr, VI_NULL, VI_NULL, &instr);

//Read from the instrument
status = viRead(instr, RecBuf, MAX_REC_SIZE, &retCount);

//close the system
status = viClose(instr);
status = viClose(defaultRM);

(*pstrResult).Format("%s",RecBuf);

return bReadOK;
}
```

3) Encapsulate the read operation with exception handling function of VISA.

```
ViStatus CDemoForRSADlg::OpenVisaDevice(CString strAddr) //Open a VISA device
{
    ViStatus status;
    char * SendAddr = NULL;

    // Change the address's data style from CString to char*
    SendAddr = strAddr.GetBuffer(strAddr.GetLength());
    strcpy(SendAddr,strAddr);
    strAddr.ReleaseBuffer();

    //Open a VISA resource
    status = viOpenDefaultRM(&m_SessRM);

    if (status == 0)
    {
        //Open the device
```

```
                status = viOpen(m_SessRM, SendAddr, VI_NULL, VI_NULL,
&m_SessInstr);

                //If you fails to open the connection, close the resource
                if (status != 0)
                {
                    viClose(m_SessRM);
                 }
             }

            return status;
        }


        ViStatus CDemoForRSADlg::CloseVisaDevice()        //Close a VISA device

        {
            ViStatus status;

            //Close the device
            status = viClose(m_SessInstr);

            if (status == 0)
            {
                //close the resource
                status = viClose(m_SessRM);
            }

            return status;
        }


        bool CDemoForRSADlg::InstrWriteEx(CString strAddr, CString strContent) //Write
        operation with exception handling
        {
            ViStatus status;
            ViUInt32 retCount;
            char * SendBuf = NULL;
            bool bWriteOK = true;

            // Change the address's data style from CString to char*
            SendBuf = strContent.GetBuffer(strContent.GetLength());
            strcpy(SendBuf,strContent);
            strContent.ReleaseBuffer();

            do
            {
                //Write command to the instrument
                status = viWrite(m_SessInstr, (unsigned char *)SendBuf, strlen(SendBuf),
&retCount);
```

```
            //If an error occurs, perform error handing
            if (status < 0)
            {
                //If the time exceed the limit value, resend the command after a delay of
1s
                if (VI_ERROR_TMO == status)
                {
                    Sleep(1000);
                    status = viWrite(m_SessInstr, (unsigned char *)SendBuf,
                    strlen(SendBuf), &retCount);
                }
                else
                {
                    //If another error occurs, reopen the connection after the connection
                    is closed and resend the command
                    status = CloseVisaDevice();
                    Sleep(1000);
                    status = OpenVisaDevice(m_strInstrAddr);
                    if (status == 0)
                    {
                        status = viWrite(m_SessInstr, (unsigned char *)SendBuf,
                        strlen(SendBuf), &retCount);
                    }
                }
            }
        } while (status < 0);

        return bWriteOK;
    }


    bool CDemoForRSADlg::InstrReadEx(CString strAddr, CString *pstrResult) //Read
    operation with exception handling
    {
        ViStatus status;
        ViUInt32 retCount;
        char * SendAddr = NULL;
        unsigned char RecBuf[MAX_REC_SIZE];
        bool bReadOK = true;

        // Change the address's data style from CString to char*
        SendAddr = strAddr.GetBuffer(strAddr.GetLength());
        strcpy(SendAddr,strAddr);
        strAddr.ReleaseBuffer();

        memset(RecBuf,0,MAX_REC_SIZE);

        do
```

```
            {
                //Read from the instrument
                status = viRead(m_SessInstr, RecBuf, MAX_REC_SIZE, &retCount);
                if (status < 0)
                {
                    //If the time exceeds the limit value, read from the instrument after a
delay of 1s
                    if (VI_ERROR_TMO == status)
                    {
                        Sleep(1000);
                        status = viRead(m_SessInstr, RecBuf, MAX_REC_SIZE,
&retCount);
                    }
                    else
                    {
                        //If another error occurs, reopen the connection after the connection
                        is closed and reread from instrument
                        status = CloseVisaDevice();
                        Sleep(1000);
                        status = OpenVisaDevice(m_strInstrAddr);
                        if (status == 0)
                        {
                            status = viRead(m_SessInstr, RecBuf, MAX_REC_SIZE,
&retCount);
                        }
                    }
                }

            } while (status < 0);


            (*pstrResult).Format("%s",RecBuf);

            return bReadOK;
        }
```

7. Add the control message response codes.

   1) Connect to the instrument

```
        void CDemoForRSADlg::OnBtConnectInstr()              // Connect to the
instrument
        {
        //TODO: Add your control notification handler code here
        ViStatus status;
        ViSession defaultRM;
        ViString expr = "?*";
        ViPFindList findList = new unsigned long;
        ViPUInt32 retcnt = new unsigned long;
        ViChar instrDesc[1000];
        CString strSrc = "";
        CString strInstr = "";
```

```
        unsigned long i = 0;
        bool bFindRSA = false;

        status = viOpenDefaultRM(&defaultRM);
        if (status < VI_SUCCESS)
        {
        // Error Initializing VISA...exiting
        MessageBox("No VISA instrument was opened ! ");
        return ;
        }

        memset(instrDesc,0,1000);

        // Find resource
        status = viFindRsrc(defaultRM,expr,findList, retcnt, instrDesc);

        for (i = 0;i < (*retcnt);i++)
        {
            // Get instrument name
            strSrc.Format("%s",instrDesc);
            InstrWrite(strSrc,"*IDN?");
            ::Sleep(200);
            InstrRead(strSrc,&strInstr);

           // If the instrument(resource) belongs to the RSA series then jump out //from the
loop
             strInstr.MakeUpper();
             if (strInstr.Find("RSA") >= 0)
             {
                 bFindRSA = true;
                 m_strInstrAddr = strSrc;
                 break;
             }

            //Find next instrument
            status = viFindNext(*findList,instrDesc);
        }

        if (bFindRSA == false)
        {
           MessageBox("Didn't find any RSA!");
        }
        UpdateData(false);
    }
```

2) Write Operation

```
    void CDemoForRSADlg::OnBtWrite()                //Write operation
     {
      //TODO: Add your control notification handler code here
```

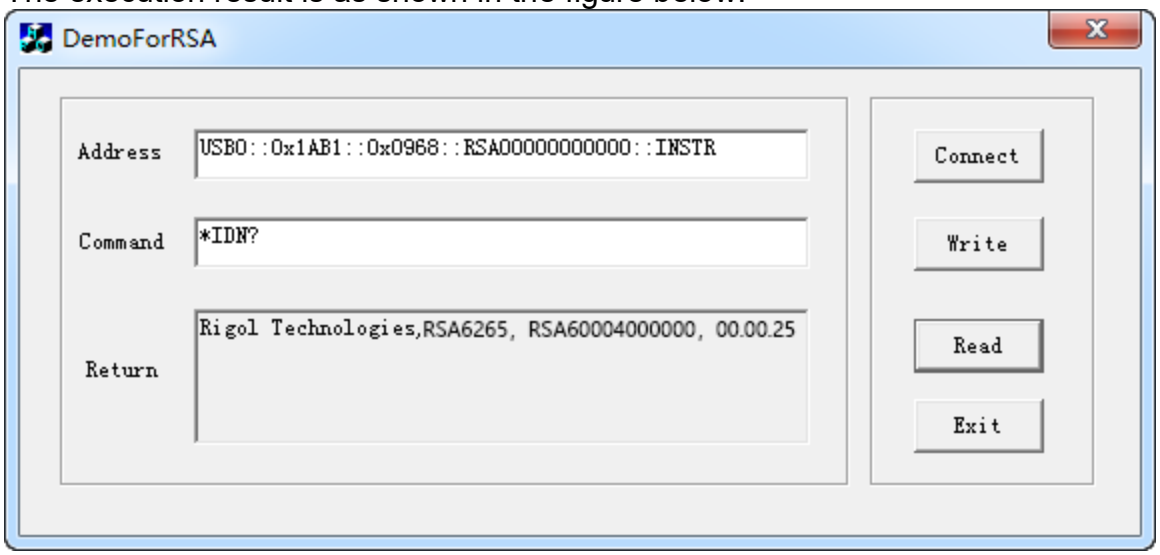
```
    UpdateData(true);
    if (m_strInstrAddr.IsEmpty())
    {
        MessageBox("Please connect to the instrument first!");
     }
    InstrWrite(m_strInstrAddr,m_strCommand);
    m_strResult.Empty();
    UpdateData(false);
}
```

3) Read Operation

```
void CDemoForRSADlg::OnBtRead()                    //Read operation
{
  //TODO: Add your control notification handler code here
  UpdateData(true);
  InstrRead(m_strInstrAddr,&m_strResult);
  UpdateData(false);
}
```

8. Run the results.
   1) Click **Connect** to search for the spectrum analyzer;
   2) Input "*IDN?" in the "Command" edit box;
   3) Click **Write** to write the command into the spectrum analyzer;
   4) Click **Read** to read the return value.

The execution result is as shown in the figure below.

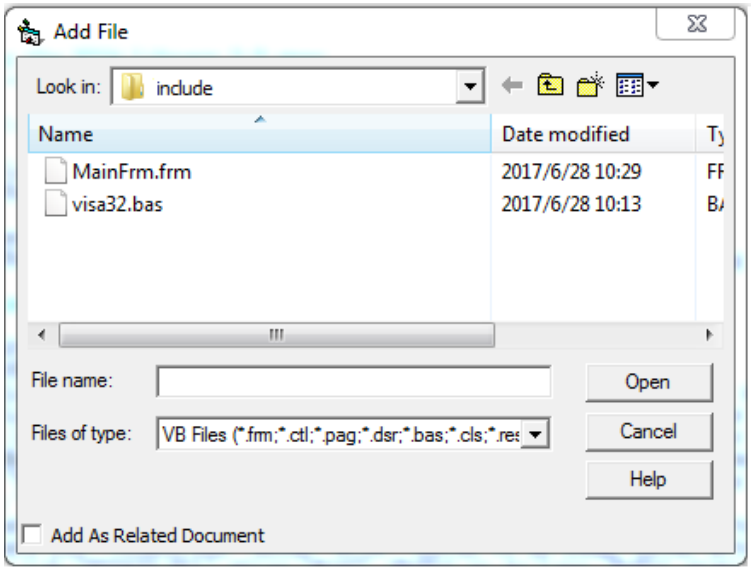


Visual Basic 6.0 Programming Example

Enter the Visual Basic 6.0 programming environment, and perform the following procedures.

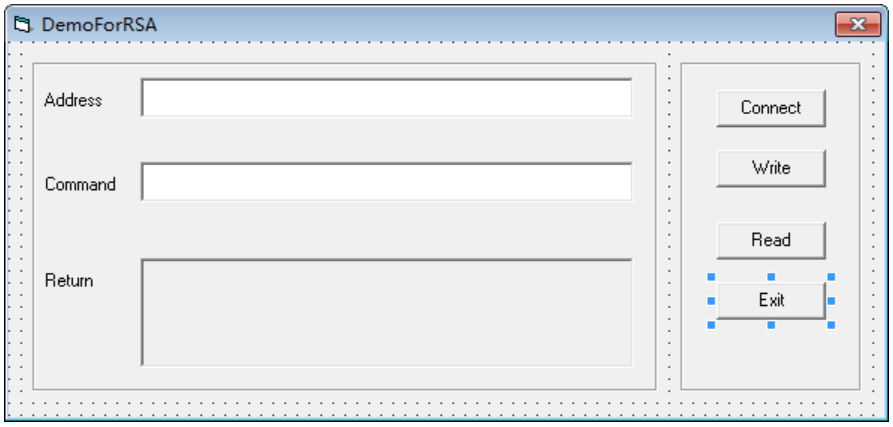
1. Build a standard application program project (Standard EXE), and name it "DemoForRSA".

2. Open **Project → Add File…**. Search for the visa32.bas file from the include folder in the installation path of NI-VISA, and then add the file to the project. The visa32.bas module contains all VISA functions and constant statements.



Then, add the Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long) statement into the visa32.bas module; or you can also create a new module to declare the Sleep function.

3. Add the **Label**, **Text**, and **Button** controls. The layout interface for adding controls is as follows:



4. Encapsulate the read and write operations of VISA.
   1) Encapsulate the write operation of VISA for easier operation.

```
'------------------------------------------------------------
'Function Name：InstrWrite
```

```
'Function：Send command to the instrument
'Input：rsrcName,instrument(resource) name
        strCmd,Command
'-------------------------------------------------------
Public Sub InstrWrite(rsrcName As String, strCmd As String)
   Dim status As Long
   Dim dfltRM As Long
   Dim sesn As Long
   Dim rSize As Long

   'Initialize the system
   status = viOpenDefaultRM(dfltRM)
   'Failed to initialize the system
   If (status < VI_SUCCESS) Then
      MsgBox " No VISA resource was opened！ "
      Exit Sub
   End If
   'Open the VISA instrument
   status = viOpen(dfltRM, rsrcName, VI_NULL, VI_NULL, sesn)
   'Failed to open the instrument
   If (status < VI_SUCCESS) Then
      MsgBox "Failed to open the instrument！ "
      Exit Sub
   End If

   'Write command to the instrument
   status = viWrite(sesn, strCmd, Len(strCmd), rSize)
   'Failed to write to the instrument
   If (status < VI_SUCCESS) Then
      MsgBox " Faild to write to the instrument！ "
      Exit Sub
   End If

   'Close the system
   status = viClose(sesn)
   status = viClose(dfltRM)

End Sub
```

2) Encapsulate the read operation of VISA for easier operation.

```
'-------------------------------------------------------
'Function Name：InstrRead
'Function：Read the return value from the instrument
'Input：rsrcName,Resource name
'Return：The string gotten from the instrument
'-------------------------------------------------------
Public Function InstrRead(rsrcName As String) As String
   Dim status As Long
   Dim dfltRM As Long
```

```
    Dim sesn As Long
    Dim strTemp0 As String * 256
    Dim strTemp1 As String
    Dim rSize As Long

    'Begin by initializing the system
    status = viOpenDefaultRM(dfltRM)
    'Initialization failed
    If (status < VI_SUCCESS) Then
        MsgBox " Failed to open the instrument! "
        Exit Function
    End If
    'Open the instrument
    status = viOpen(dfltRM, rsrcName, VI_NULL, VI_NULL, sesn)
    'Failed to open the instrument
    If (status < VI_SUCCESS) Then
        MsgBox " Failed to open the instrument! "
        Exit Function
    End If

    'Read from the instrument
    stasus = viRead(sesn, strTemp0, 256, rSize)
    'Reading failed
    If (status < VI_SUCCESS) Then
        MsgBox " Failed to read from the instrument! "
        Exit Function
    End If

    'Close the system
    status = viClose(sesn)
    status = viClose(dfltRM)

    'Remove the space at the end of the string
    strTemp1 = Left(strTemp0, rSize)
    InstrRead = strTemp1
End Function
```

5. Add the control event codes.

   1) Connect to the instrument

```
'Connect to the instrument
Private Sub CmdConnect_Click()
    Const MAX_CNT = 200
    Dim status As Long
    Dim dfltRM As Long
    Dim sesn As Long
    Dim fList As Long
    Dim buffer As String * MAX_CNT, Desc As String * 256
    Dim nList As Long, retCount As Long
```

```
        Dim rsrcName(19) As String * VI_FIND_BUFLEN, instrDesc As String *
VI_FIND_BUFLEN
        Dim i, j As Long
        Dim strRet As String
        Dim bFindRSA As Boolean

        'Initialize the system
        status = viOpenDefaultRM(dfltRM)
        'Initialization failed
        If (status < VI_SUCCESS) Then
            MsgBox " No VISA resource was opened ！ "
            Exit Sub
        End If

        'Find instrument resource
        Call viFindRsrc(dfltRM, "USB?*INSTR", fList, nList, rsrcName(0))
        'Get the list of the instruments (resources)
        strRet = ""
        bFindRSA = False
        For i = 0 To nList - 1
        'Get the instrument name
        InstrWrite rsrcName(i), "*IDN?"
        Sleep 200
        strRet = InstrRead(rsrcName(i))
        'Continuing searching for the resource until an RSA instrument is found
        strRet = UCase(strRet)
        j = InStr(strRet, "RSA")
        If (j >= 0) Then
           bFindRSA = True
           Exit For
        End If

        Call viFindNext(fList + i - 1, rsrcName(i))
        Next i
        'Display
        If (bFindRSA = True) Then
           TxtInsAddr.Text = rsrcName(i)
        Else
           TxtInsAddr.Text = ""
        End If
    End Sub
```

2) Write Operation

```
    'Write the command to the instrument
    Private Sub CmdWrite_Click()
       If (TxtInsAddr.Text = "") Then
           MsgBox ("Please write the instrument address！ ")
       End If

       InstrWrite TxtInsAddr.Text, TxtCommand.Text
```

```
    End Sub
```

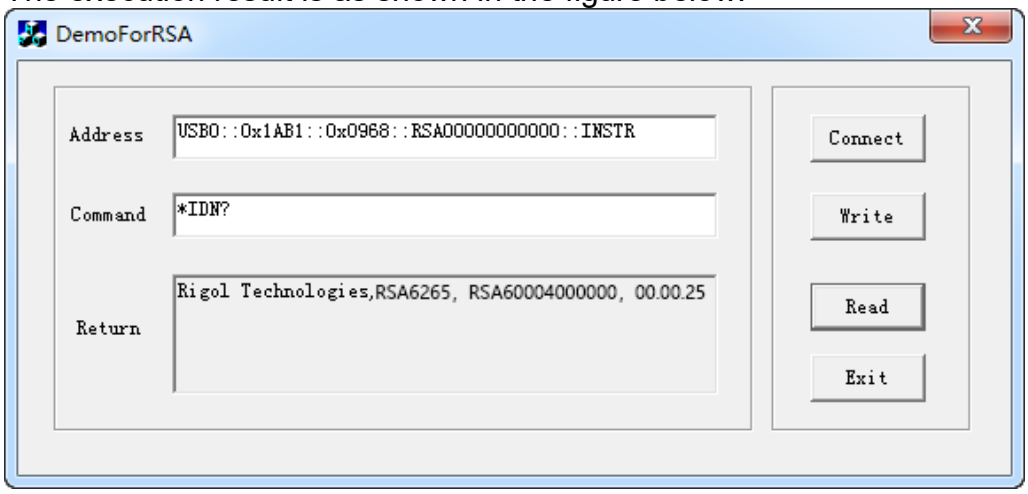
3) Read Operation

```
'Read the return value from the instrument
Private Sub CmdRead_Click()
   Dim strTemp As String
   strTemp = InstrRead(TxtInsAddr.Text)
   TxtReturn.Text = strTemp
End Sub
```

6. Run the results.

1) Click **Connect** to search for the spectrum analyzer;
2) Input "*IDN?" in the "Command" edit box;
3) Click **Write** to write the command into the spectrum analyzer;
4) Click **Read** to read the return value.

The execution result is as shown in the figure below.

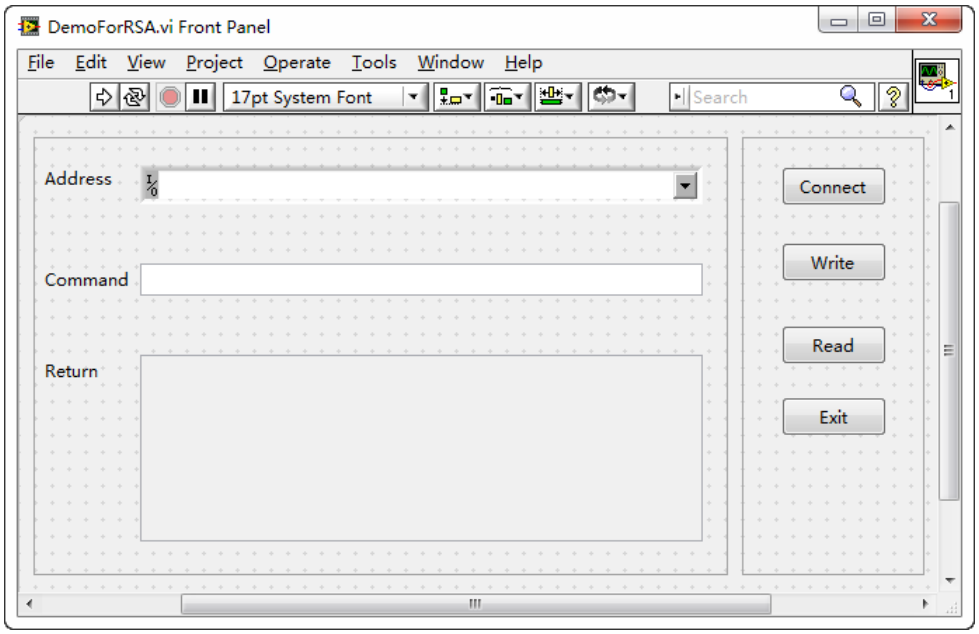


LabVIEW 2010 Programming Example

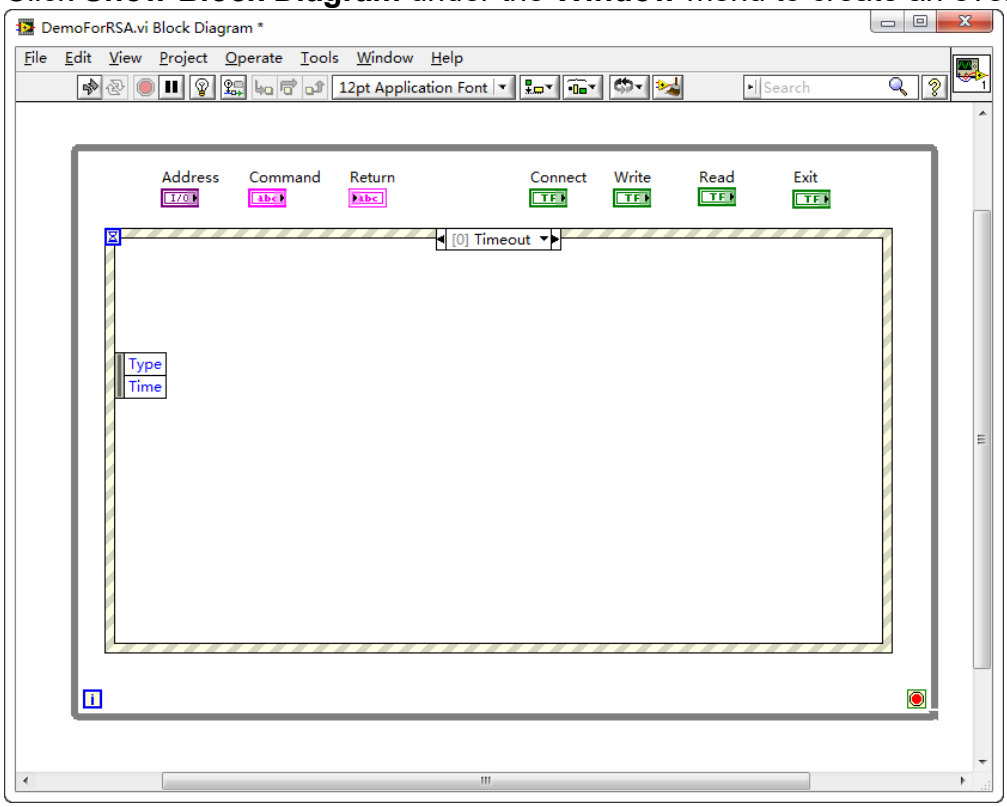
Enter the Labview 2010 programming environment, and perform the following procedures.

1. Create a VI file, and name it "DemoForRSA".

2. Add controls to the front panel interface, including the Address field, Command field, and Return field, the Connect button, the Write button, the Read button, and the Exit button.

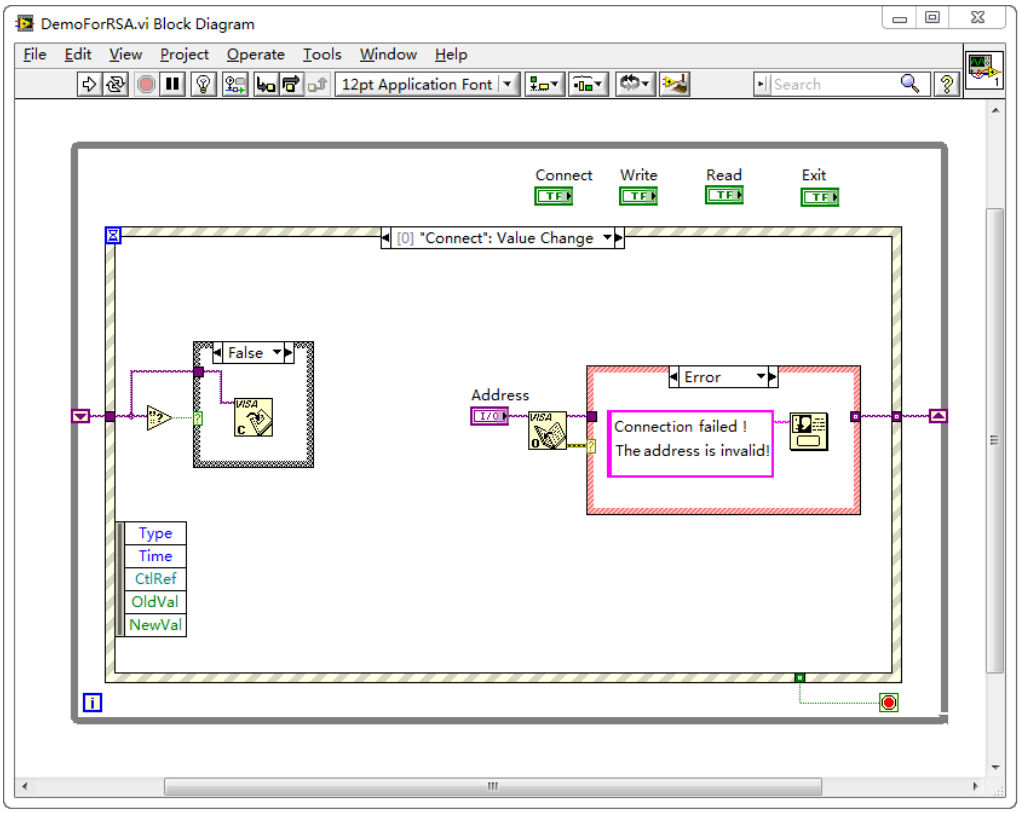


3. Click **Show Block Diagram** under the **Window** menu to create an event structure.

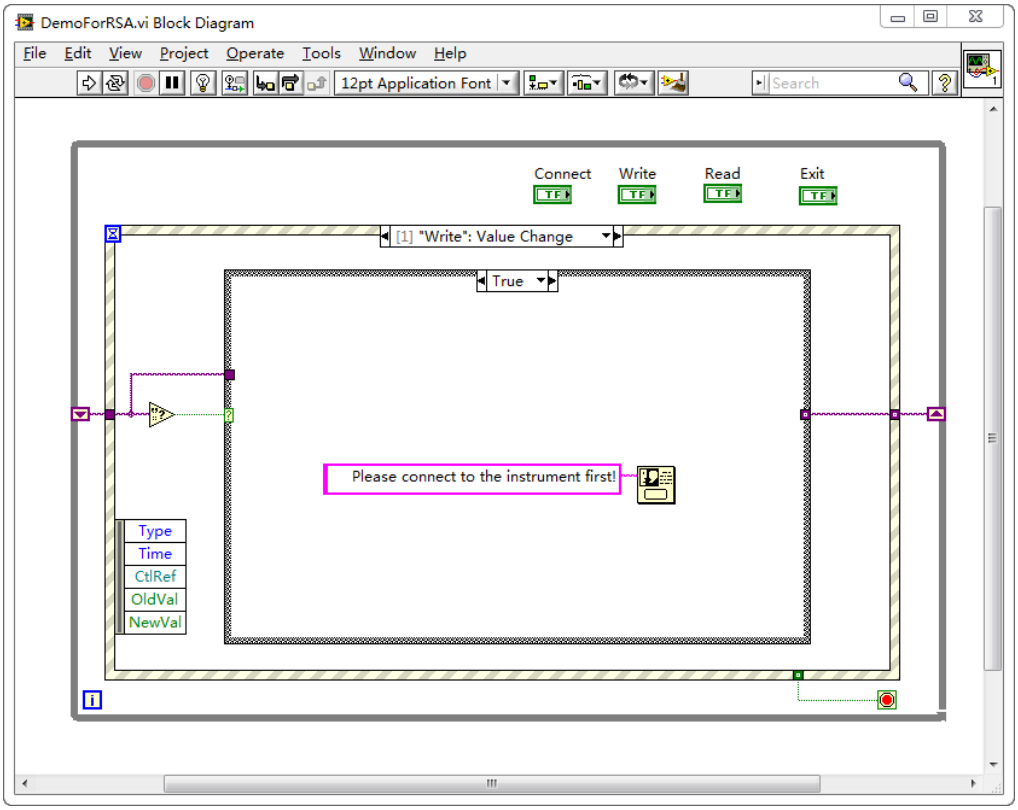


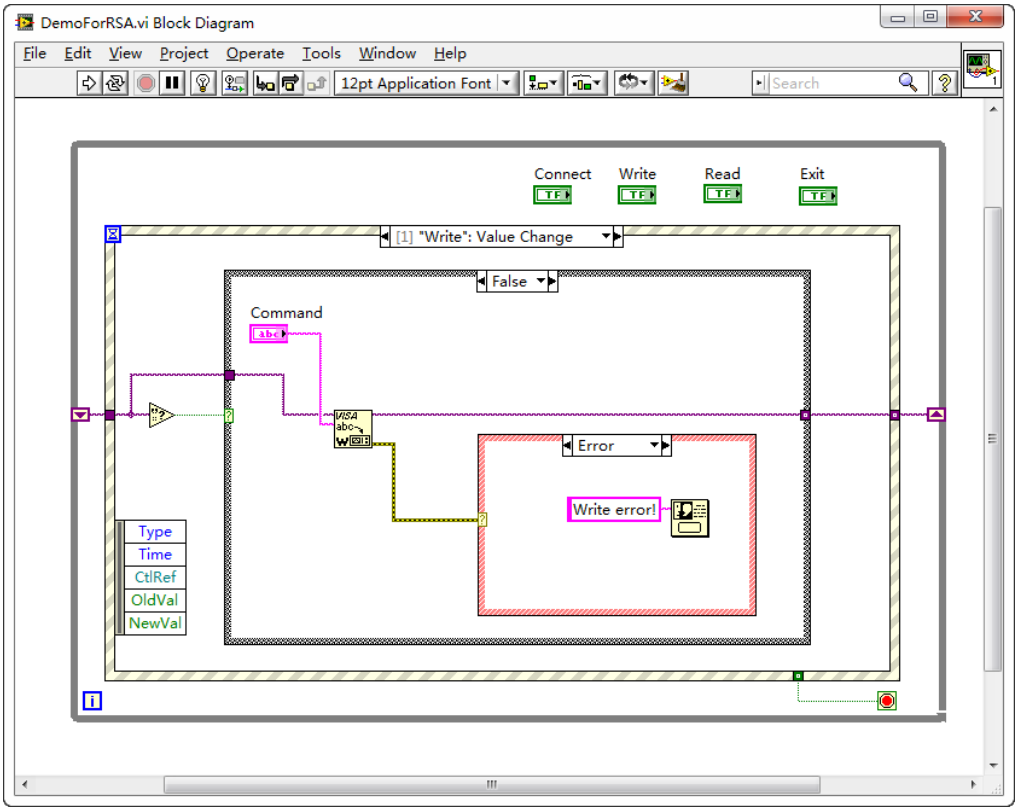
4. Add the events (including connecting to the instrument, write operation, read operation, and exit)

   1) Connect to the instrument

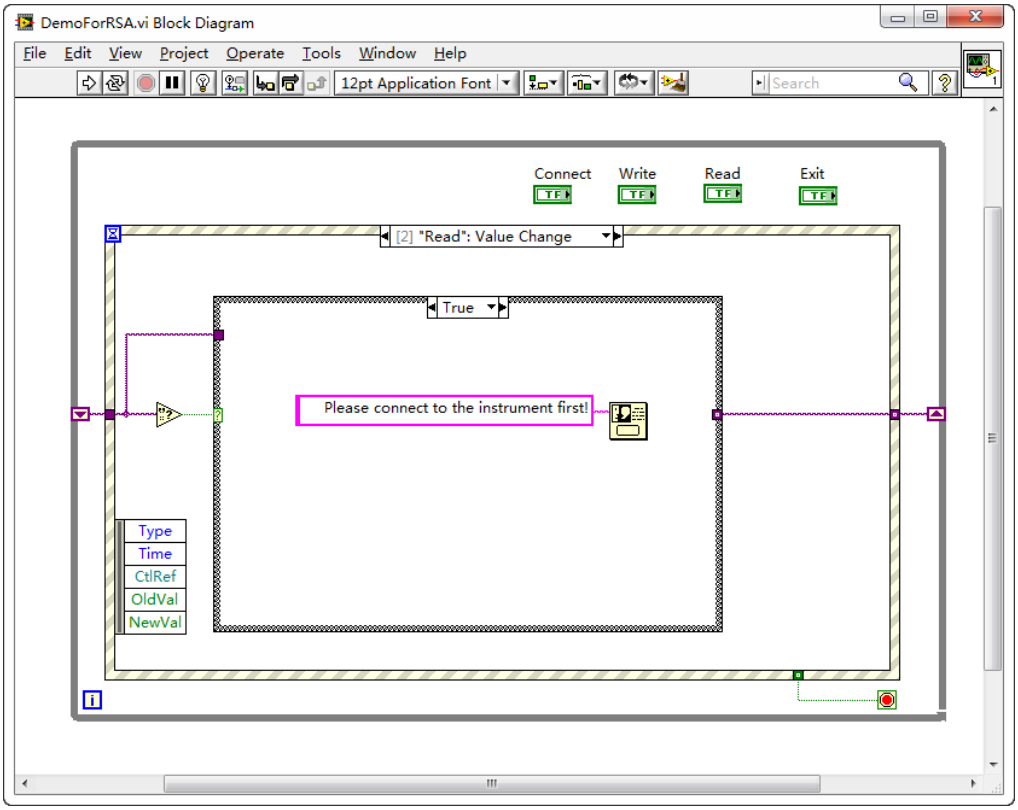


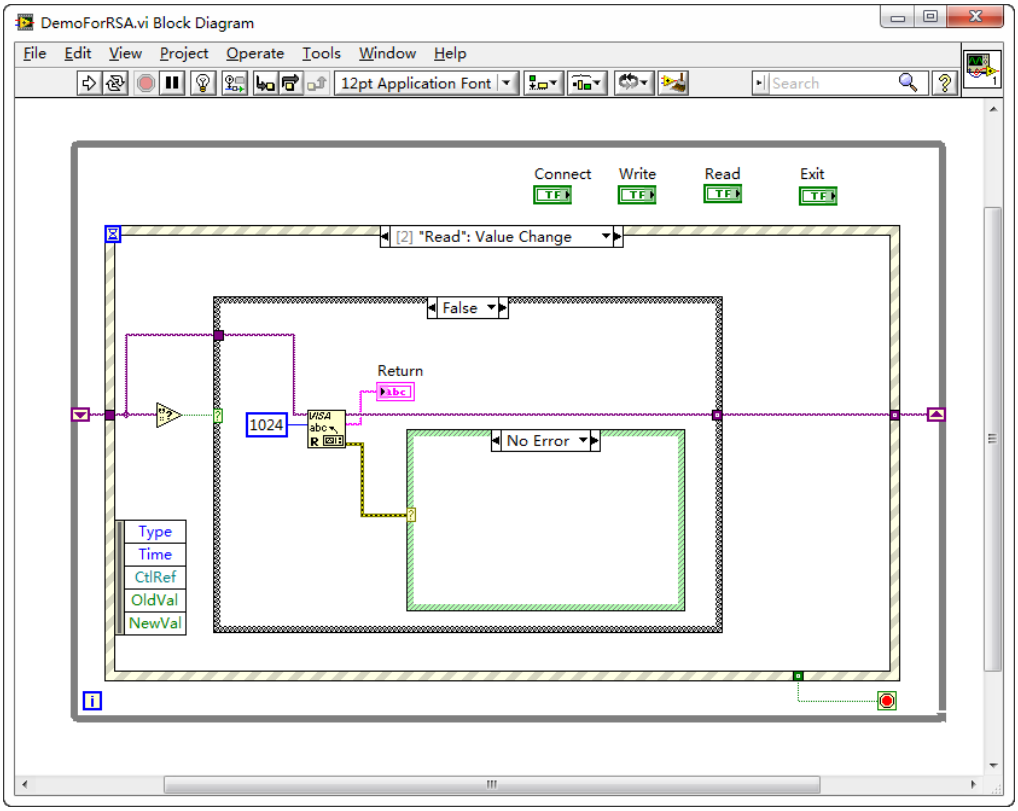
   2) Write operation (including error confirmation)



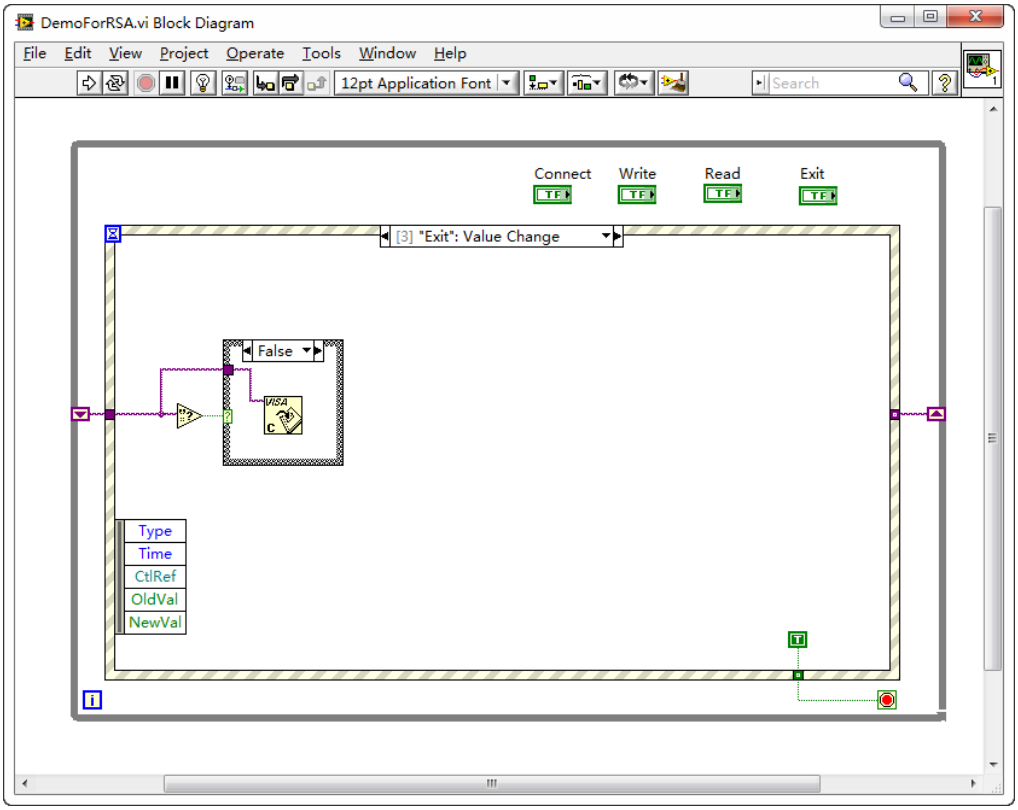


3) Read operation (including error correction advice)

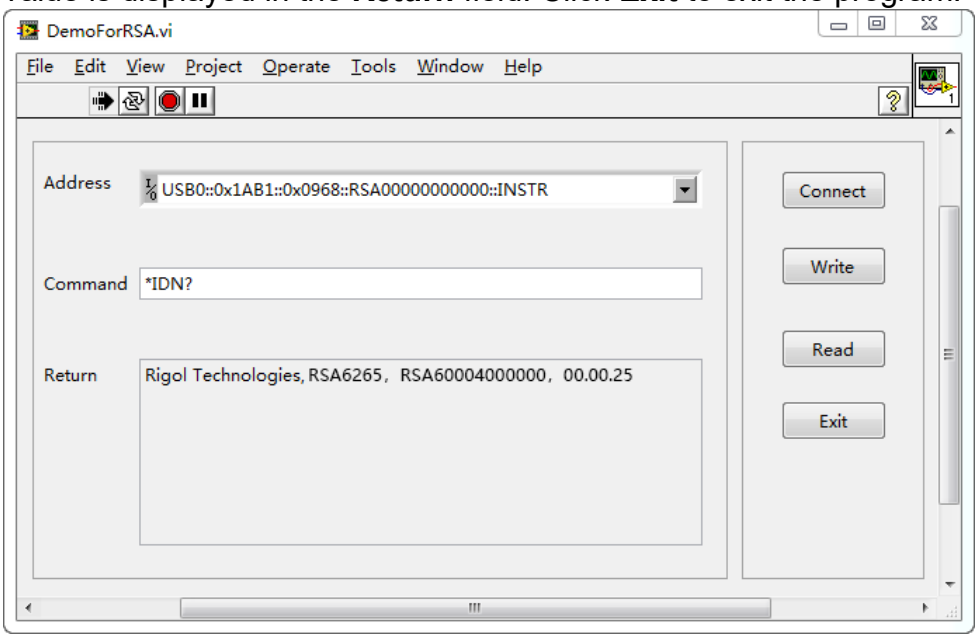




4) Exit



5) Run the program, and then the following interface is displayed below. Click the VISA resource name from the drop-down list under **Address**, and click **Connect** to connect the instrument. Then, input a command in the **Command** field. Click **Write** to write the command to the instrument. If the command is a query (e.g.*IDN?), click **Write** to write the command into the instrument, and then click **Read**. The return value is displayed in the **Return** field. Click **Exit** to exit the program.



Linux Programming Example

This section illustrates how to program and control the spectrum analyzer to realize the common functions in Linux operating system.

Programming Preparations

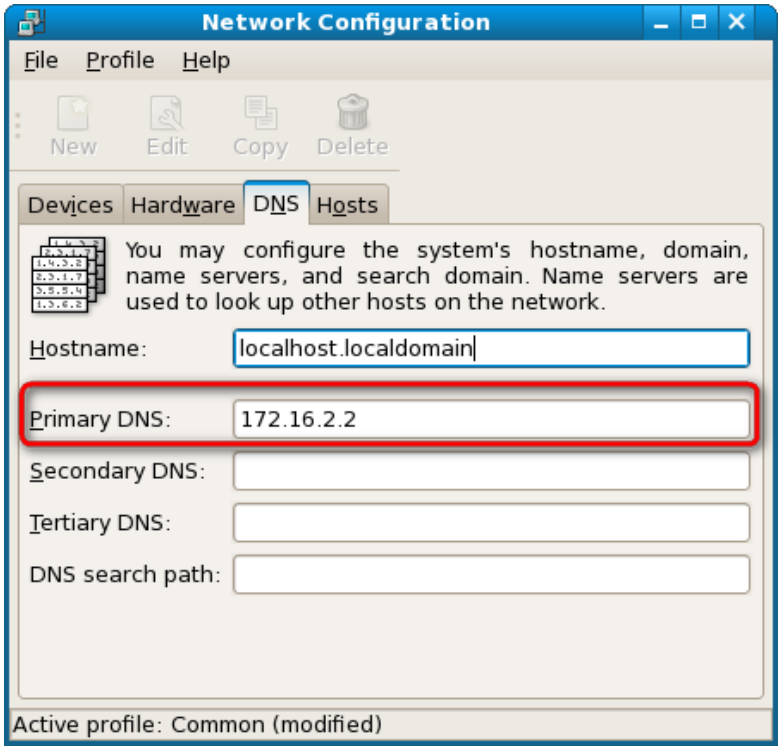
1. Programming environment:
   Operating system: Fedroa 8 (Linux-2.6.23)
   GCC version: gcc-4.1.2

2. Install the VISA library. First, check whether your PC has installed NI's VISA library. If not, download it from NI website (*http://www.ni.com/visa/*). The installation procedures are as follows:
   Download the VISA library NI-VISA-4.4.0.ISO from the NI website.

   Create a new directory.
   #mkdir NI_VISA
   Mount the iso file.
   #mount -o loop -t iso9660 NI-VISA-4.4.0.iso NI_VISA
   Enter the NI_VISA directory to install
   #cd NI_VISA
   #./INSTALL
   Unmount the iso file.
   #umount NI_VISA
   After the installation is finished, the default installation path is /usr/local.

3. Connect spectrum analyzer to the PC via the LAN interface of the analyzer. Use the USB cable to connect the analyzer to the PC via the LAN interface on the rear panel of the analyzer. You can also use a network cable to connect the spectrum analyzer to the local area network where the PC resides.

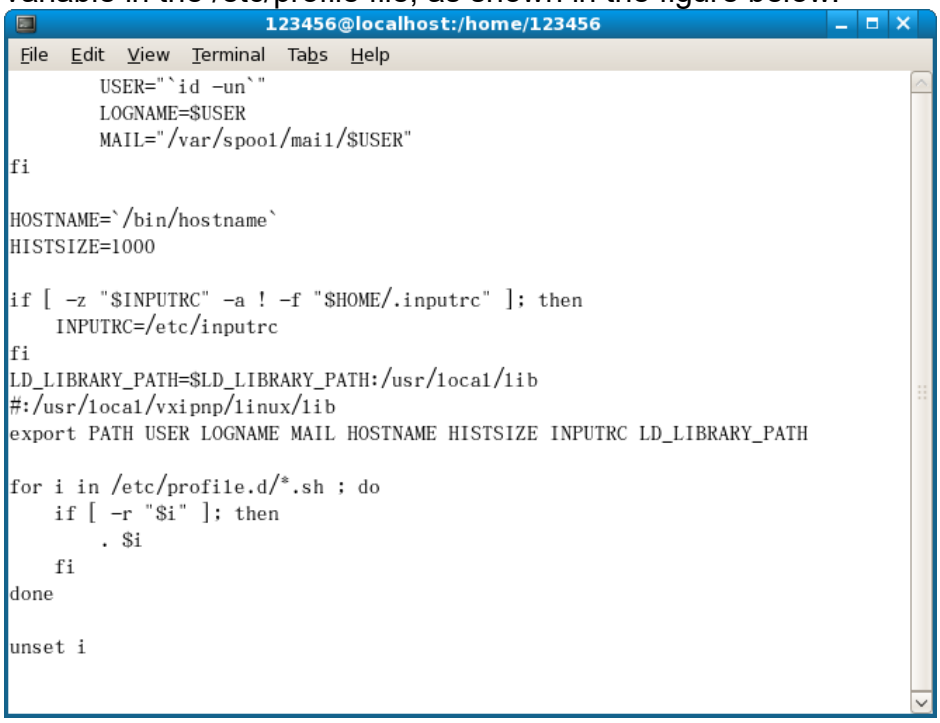
   After the spectrum analyzer is connected to the PC properly, configure the network address for the spectrum analyzer to make its address to be within the same network segment where the PC resides. For example, if the network address and DNS setting configured for the PC are as shown in the figures below, then, the network address of the spectrum analyzer should be configured as follows.
   IP Address: 172.16.3.X*
   Default Gateway: 172.16.3.1
   Subnet Mask: 255.255.255.0
   DNS: 172.16.2.2
   Note*: X can be any value not used ranging from 2 to 254.



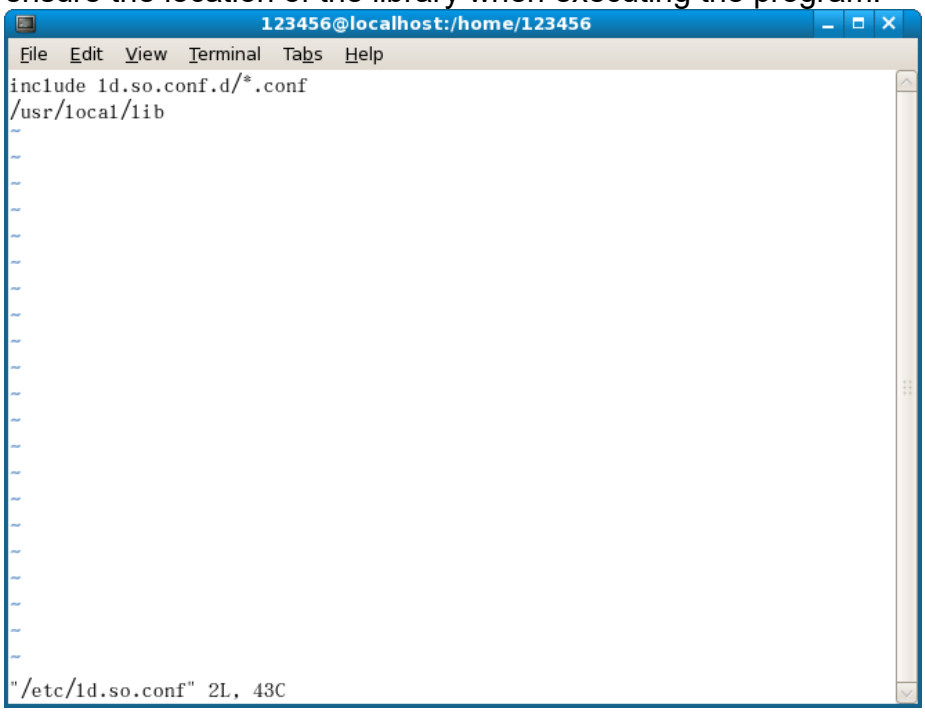
4. Use either of the following two methods to add the library location to the search path of the library, so that the program can load the installed library file automatically.

   Method 1: Specify the search path of the library in the environment variable LD_LIBRARY_PATH.

Operation Method: Add the library file path /usr/local/lib to the LD_LIBRARY_PATH variable in the /etc/profile file, as shown in the figure below.



Method 2: Add the search path of the library in the /etc/ld.so.conf file.
Operation Method: #echo "/usr/local/lib" >> /etc/ld.so.conf, as shown in the figure below.
After setting the search path of the library in /etc/ld.so.conf, run the /sbin/ldconfig command to update /etc/ld.so.cache (this command should have the root permission) to ensure the location of the library when executing the program.



Linux Programming Procedures

1. Edit the DemoForDSA.h header file and declare a class to encapsulate the operation and property of the instrument.

```
#ifndef DEMO_FOR_RSA_H
#define DEMO_FOR_RSA_H

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <iostream>
//#include <syswait.h>
using namespace std;

#define MAX_SEND_BUF_SIZE 50
#define MAX_REC_SIZE 300

class DemoForRSA
{
// Construction
public:
DemoForRSA();
bool InstrRead(string strAddr, string & pstrResult);
bool InstrWrite(string strAddr, string strContent);
bool ConnectInstr();

string m_strInstrAddr;
string m_strResult;
string m_strCommand;

};

void makeupper(string & instr);

#endif
```

2. Edit the DemoForDSA.cpp file to realize various operations of the instrument.

```
#include "visa.h"
#include "DemoForRSA.h"

DemoForRSA::DemoForRSA()
{
m_strInstrAddr = "";
m_strResult = "";
m_strCommand = "";
}

bool DemoForRSA::ConnectInstr()
```

```
{
ViUInt32 retCount;
ViStatus status;
ViSession defaultRM;
ViString expr = "?*";
ViPFindList findList = new unsigned long;
ViPUInt32 retcnt = new unsigned long;
string strSrc = "";
string strInstr = "";
ViChar instrDesc[1000];

unsigned long i = 0;
bool bFindRSA = false;
memset(instrDesc,0,1000);

//Turn on the VISA device
status = viOpenDefaultRM(&defaultRM);

if (status < VI_SUCCESS)
{
   cout<<"No VISA equipment!"<<endl;
   return false;
}

//Search for resources
status = viFindRsrc(defaultRM,expr,findList, retcnt, instrDesc);

for (i = 0;i < (*retcnt);i++)
{
   //Acquire the instrument name
   strSrc = instrDesc;

   InstrWrite(strSrc,"*IDN?");
   usleep(200);
   InstrRead(strSrc,strInstr);

   // If the RSA series is found, then exit
   makeupper(strInstr);
   if (strInstr.find("RSA",0) > 0)
   {
      bFindRSA = true;
      m_strInstrAddr = strSrc;
      break;
    }

   //Acquire the next device
   status = viFindNext(*findList,instrDesc);
}
```

```
    if (bFindRSA == false)
    {
        printf("RSA device not found!\n");
        return false;
    }

        return true;
    }

    bool DemoForRSA::InstrWrite(string strAddr, string strContent) //Write operation
    {
    ViSession defaultRM,instr;
    ViStatus status;
    ViUInt32 retCount;
    char * SendBuf = NULL;
    char * SendAddr = NULL;
    bool bWriteOK = false;
    string str;
    //Address conversion, convert the string type to char*
    SendAddr = const_cast<char*>(strAddr.c_str());

    //Address conversion, convert the string type to char*
    SendBuf = const_cast<char*>(strContent.c_str());

    //Turn on the specified device□
    status = viOpenDefaultRM(&defaultRM);
    if (status < VI_SUCCESS)
    {
       cout<<"No VISA equipment!"<<endl;
       return false;
    }

    status = viOpen(defaultRM, SendAddr, VI_NULL, VI_NULL, &instr);

    //Write command to the device
    status = viWrite(instr, (unsigned char *)SendBuf, strlen(SendBuf), &retCount);

    //Turn off the device□
    status = viClose(instr);
    status = viClose(defaultRM);
    return bWriteOK;
    }

    bool DemoForRSA::InstrRead(string strAddr, string & pstrResult) //Read operation
    {
    ViSession defaultRM,instr;
    ViStatus status;
    ViUInt32 retCount;
    char* SendAddr = NULL;
    char * result = NULL;
```

```
bool bReadOK = false;
unsigned char RecBuf[MAX_REC_SIZE];
string str;
memset(RecBuf,0,MAX_REC_SIZE);

result=char*)malloc(MAX_REC_SIZE*sizeof(char));
memset(result,0,MAX_REC_SIZE);

//Address conversion, convert the string type to char*
SendAddr=const_cast<char*>(strAddr.c_str());

//Turn on the VISA device
status = viOpenDefaultRM(&defaultRM);
if (status < VI_SUCCESS)
{
    // Error Initializing VISA...exiting
    cout<<"No VISA equipment!"<<endl;
    return false;
}

//Turn on the specified device□
status = viOpen(defaultRM, SendAddr, VI_NULL, VI_NULL, &instr);

//Read from the device□
status = viRead(instr, RecBuf, MAX_REC_SIZE, &retCount);

//Turn off the device□
status = viClose(instr);
status = viClose(defaultRM);
sprintf(result,"%s",RecBuf);
pstrResult = result;
free(result);
return bReadOK;
}

void makeupper( string &instr)
{

   string outstr = "";
   if(instr == "")
   {
      exit(0);
   }

   for(int i = 0;i < instr.length();i++)
   {
      instr[i] = toupper(instr[i]);
   }
```

```
}
```

3. Edit the function file mainloop.cpp to complete the flow control.

```
#include "DemoForRSA.h"

void menudisplay()
{
   cout<<"\t\t   Please operate the instrument:\n read write quit"<<endl;
}

int main()
{
   DemoForRSA demo;
   char temp[50];
   if(!demo.ConnectInstr())
   {
      cout<<"can not connect the equipment!"<<endl;
      return 0;
    }
   else
   {

      cout<<"\n connect equipment success!"<<endl;
      cout<<" the equipment address is :"<<demo.m_strInstrAddr<<endl;
    }

   while(1)
   {
      menudisplay();
      //cin>>demo.m_strCommand;
      cin.getline(temp,50);
      demo.m_strCommand=temp;
      if(demo.m_strCommand[0]='r' && demo.m_strCommand[1]='e'
         && demo.m_strCommand[2]='a' && demo.m_strCommand[3]='d')
      {
         //demo.InstrWrite(demo.m_strInstrAddr,"*IDN?");
         //demo.InstrRead(demo.m_strInstrAddr,demo.m_strResult);
         cout<<"read result:"<<demo.m_strResult<<endl;
         demo.m_strResult="";

       }

       else if (demo.m_strCommand[0]='w' && demo.m_strCommand[1]='r'
                 && demo.m_strCommand[2]='i' && demo.m_strCommand[3]='t' &&
                demo.m_strCommand[4]='e')
       {
                if (demo.m_strInstrAddr="")
                {
                    cout<<"Please connect the instrument！ \ n";
                 }
```

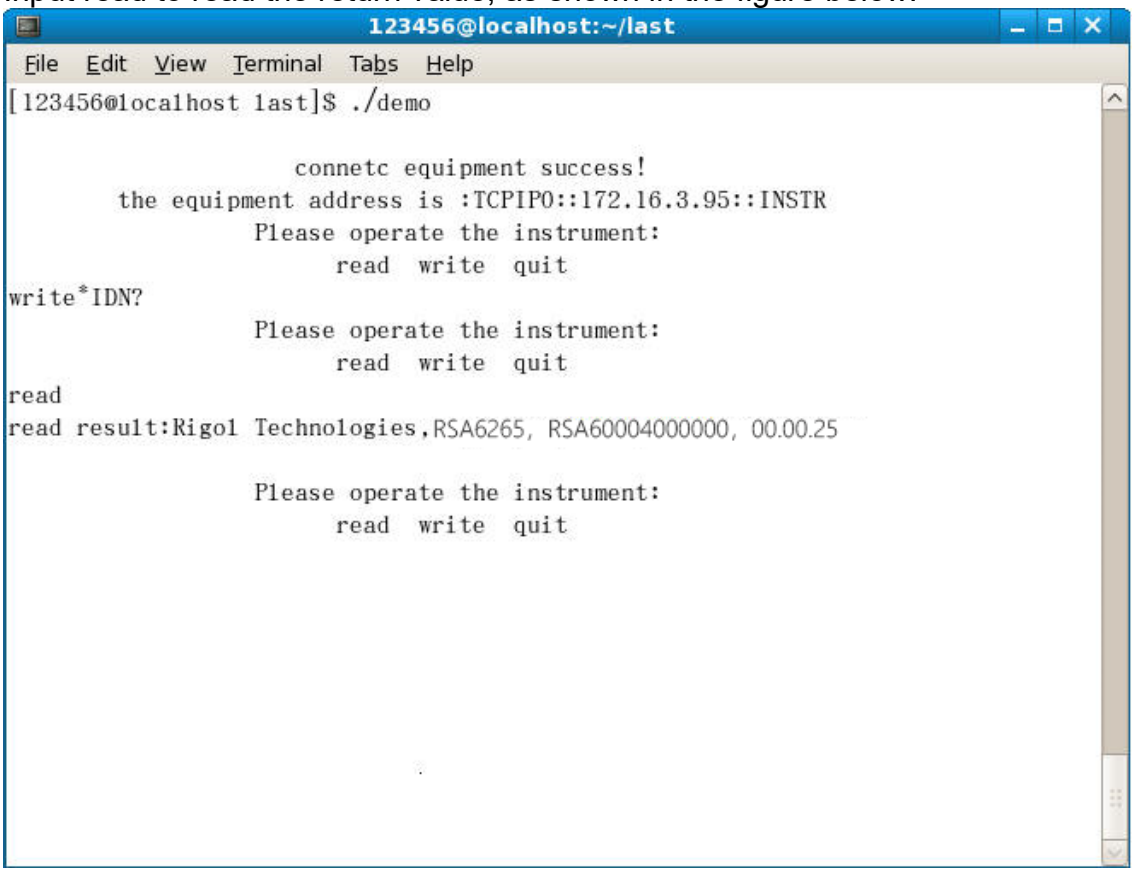
```
                demo.InstrWrite(demo.m_strInstrAddr,demo.m_strCommand.substr(5,40));
                usleep(200);

              //Read operation
             demo.InstrRead(demo.m_strInstrAddr,demo.m_strResult);

        }

      else if (demo.m_strCommand[0] == 'q' && demo.m_strCommand[1] == 'u'
               && demo.m_strCommand[2] == 'i' && demo.m_strCommand[3] == 't')

      {
            break;
        }
      else if(demo.m_strCommand != "")
      {
            cout<<"Bad command!"<<endl;
        }
    }
    return 1;

}
```

4. makefile file

```
src = DemoForRSA.cpp mainloop.cpp DemoForRSA.h

obj = DemoForRSA.o mainloop.o
INCLUDE= -I/usr/local/vxipnp/linux/include
LIB= -lvisa -lc -lpthread
CC=
demo : $(obj)
$(CC) $(INCLUDE) $(LIB) -o demo $(obj)

mainloop.o : mainloop.cpp DemoForRSA.h
$(CC) -c $< -o $@
DemoForRSA.o: DemoForRSA.cpp DemoForRSA.h
$(CC) -c $< -o $@

.PHONY : clean
clean:
rm demo $(obj)
```

5. Run the results.
   1) #make
   2) ./demo
   3) When the program runs, the instrument is connected automatically. If no instrument is found, a prompt message "No VISA equipment!" is displayed, and the system exits the program. If the instrument is found and successfully connected, the following interface is displayed.

4) Input write<command> (for example, write<*IDN?>) to write the command into the spectrum analyzer.
5) Input read to read the return value, as shown in the figure below.



全面助力智慧世界和科技创新

- 蜂窝-5G/WIFI
- UWB/RFID/ ZIGBEE
- 数字总线/以太网
- 光通信

- 数字/模拟/射频芯片
- 存储器及MCU芯片
- 第三代半导体
- 太阳能光伏电池

- 新能源汽车
- 光伏/逆变器
- 电源测试
- 汽车电子

为行业客户提供测试测量产品和解决方案

RIGOL开放实验室

地　　址：北京、苏州、深圳、西安
开放时间：工作日 9:00 am~6:00 pm
预约电话：400-620-0002
RIGOL客服热线： 400-620-0002
官网预约网址：
https://www.rigol.com/quote/Lab-appoint.html

RIGOL开放实验室预约

RIGOL实验室视频号

www.rigol.com

RIGOL官方微信

RIGOL官网