RIGOL

# VSA Mode

## For RSA6000 Series Spectrum Analyzer

Programming Guide

Aug. 2025

# 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*

# 1 Document Overview

This manual gives you a quick review about the front and rear panel of RSA6000 series, the user interface, and its basic operation method.

**Tip**

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

**Publication Number**
PGD28101-1110

**Software Version**
00.00.25
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 configuration items.

3. **Operation Procedures:**

The next step of the operation is denoted by ">" 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 |

# 2 Programming Overview

## 2.1 SCPI Command Overview

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,

**:ACQuire:TYPE** <*type*>

**:ACQuire:TYPE?**

**ACQuire** is the root keyword of the command, **TYPE** is the second-level keyword. The command line starts with a colon "**:**", and different levels of keywords are also separated by colons. <*type*> indicates a settable parameter. The command ending with a quotation mark "**?**" indicates querying a certain function. The command keywords **:ACQuire:TYPE** and the parameter <*type*> 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** <*sat*>

   **:SYSTem:PSTatus?**

   Wherein,

   - <*sat*> can be set to DEFault|OPEN.
   - The query returns an abbreviated form: DEF or OPEN.

3. **Integer**

   Unless otherwise specified, the parameter can be any integer (NR1 format) within the effective value range.

**CAUTION**

**Do not set the parameter to a decimal, otherwise, errors will occur.**

For example,

**:DISPlay:GBRightness** <*brightness*>

**:DISPlay:GBRightness?**

Wherein, <*brightness*> can be set to an integer ranging from 1 to 100. The query returns an integer ranging from 1 to 100.

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,

**:TRIGger:TIMeout:TIME** <*time*>

**:TRIGger:TIMeout:TIME?**

Wherein, <*time*> can be set to any real number ranging from 1.6E-8 (16 ns) to 1E +1 (10 s). The query returns a real number in scientific notation.

**5. ASCII String**

The parameter can be the combinations of ASCII characters. For example,

**:LAN:GATeway** <*string*>

Wherein, <*string*> can be set to

```
192.168.1.1
```

**Command Abbreviation**

The keywords of all the commands are case-insensitive. They can all be in upper case or in lower case. If an abbreviation is used, you must input all the capital letters in the command. For example,

**:DISPlay:GBRightness?**

can be abbreviated as

**:DISP:GBR?**

## 2.2 Remote Control

This instrument can be connected to the PC via the USB, or LAN interface to set up communication and realize remote control through the PC. The remote control can be realized by using 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 Ultra Sigma is recommended. When the instrument is connected to the PC via the USB, or LAN interface, the Ultra Sigma software can search for instrument resources and enable command interaction(Ultra Sigma is only supported on the PC installed with the win10 operating system or below).

Log in to the RIGOL official website. Click **Support** and select **Soft/Firmware** to obtain the Ultra Sigma software package and help documentation.

**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.

### 2.2.1 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 device resource name and select "SCPI Panel Control" to open the remotely 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.

### 2.2.2 Remote Control via LAN

1. **Connect the device**

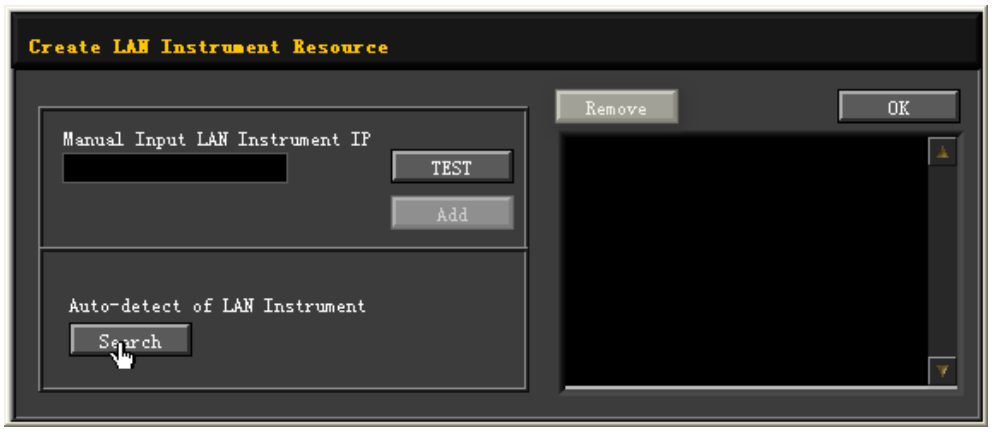
   Use the network cable to connect the instrument to your local area network (LAN).

2. **Configure network parameters**

   Configure the network parameters of the instrument in **Utility**>**IO** menu.

3. **Search for Search device resource**

Start up Ultra Sigma and click **LAN** to open the panel as shown in the figure below. Click **Search** and the software searches for the instrument resources currently connected to the LAN and the resources found are displayed at the right section of the window as shown in the figure below. 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 will appear under the "RIGOL Online Resource" directory.

5. **Control the instrument remotely**

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

6. **Load LXI webpage**

   As this instrument conforms to LXI CORE 2011 DEVICE standards, you can load LXI web page 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 web page. You can also directly input the IP address of the instrument in the address bar of the PC browser to load the LXI web page.

# 3 Command System

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

**Remarks:**

1. 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).
2. This manual takes RSA6265 as an example to illustrate the range of the parameters in each command.

# :ABORt Commands

## :ABORt

**Syntax**
:ABORt

**Description**
Aborts the current measurement.

# :CALCulate Commands

## :CALCulate:DDEMod[:SEGMent1]:FILTer:COEFficient:MEASure[:DATA]

**Syntax**
:CALCulate:DDEMod[:SEGMent1]:FILTer:COEFficient:MEASure[:DATA] <string>
:CALCulate:DDEMod[:SEGMent1]:FILTer:COEFficient:MEASure[:DATA]?

**Description**
Sets the coefficient of the user-defined measurement filter.
Queries the coefficient of the user-defined measurement filter.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <string> | ASCII String | -- | -- |

**Remarks**
N/A

**Return Format**
The query returns the coefficient of the user-defined measurement filter in strings.

**Example**
The following command sets the coefficient of the user-defined measurement filter to 12111 024.
:CALCulate:DDEMod:SEGMent1:FILTer:COEFficient:MEASure 12111 024
The following query returns 12111 024.
:CALCulate:DDEMod:SEGMent1:FILTer:COEFficient:MEASure?

## :CALCulate:DDEMod[:SEGMent1]:FILTer:COEFficient:REFerence[:DATA]

**Syntax**
:CALCulate:DDEMod[:SEGMent1]:FILTer:COEFficient:REFerence[:DATA] <string>
:CALCulate:DDEMod[:SEGMent1]:FILTer:COEFficient:REFerence[:DATA]?

**Description**
Sets the coefficient of the user-defined reference filter.
Queries the coefficient of the user-defined reference filter.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <string> | ASCII String | -- | -- |

**Remarks**
N/A

**Return Format**
The query returns the coefficient of the user-defined reference filter in strings.

**Example**
The following command sets coefficient of the user-defined reference filter to 12000012.
:CALCulate:DDEMod:SEGMent1:FILTer:COEFficient:REFerence 12000012
The following query returns 12000012.
:CALCulate:DDEMod:SEGMent1:FILTer:COEFficient:REFerence?

## :CALCulate:DDEMod:MARKer<n>:MODE

**Syntax**
:CALCulate:DDEMod:MARKer<n>:MODE <mode>
:CALCulate:DDEMod:MARKer<n>:MODE?

**Description**
Sets the type of the specified marker.
Queries the type of the specified marker.

**Parameter**

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

**Remarks**
POSition: indicates the normal marker.
DELTa: indicates difference between two data points.
FIXed: indicates that the marker is fixed.
OFF: turns off the selected marker.

**Return Format**
The query returns POS, DELT, FIX, or OFF.

**Example**
The following command sets the type of Marker 1 to Position.
:CALCulate:DDEMod:MARKer1:MODE POSition

The following query returns POS.
:CALCulate:DDEMod:MARKer1:MODE?

## :CALCulate:DDEMod:MARKer<n>:X

**Syntax**
:CALCulate:DDEMod:MARKer<n>:X <real>
:CALCulate:DDEMod:MARKer<n>:X?

**Description**
Sets the X-axis value of the specified marker.
Queries the X-axis value of the specified marker.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <n> | Discrete | 1\|2\|3\|4\|5\|6\|7\|8 | —— |
| <real> | Real | Refer to "**Remarks**" | —— |

**Remarks**
<real> can be any value within the available range of the current X axis. When you select the frequency-domain data, X-axis indicates frequency; when you select the time-domain data, X-axis indicates time.
If the marker mode is set to Position or Fixed, this value sets the X value at the marker.
If the marker mode is set to Delta, this value sets the X value of the Delta marker in relative to that of the reference marker.

**Return Format**
The query returns the X-axis value of the specified marker in scientific notation.

**Example**
The following command sets the X-axis value of Marker 1 to 150 MHz.

:CALCulate:DDEMod:MARKer1:X 150000000

The following query returns 1.500000000e+08.
:CALCulate:DDEMod:MARKer1:X?

## :CALCulate:DDEMod:MARKer<n>:Y[:REAL]

**Syntax**
:CALCulate:DDEMod:MARKer<n>:Y[:REAL] <real>
:CALCulate:DDEMod:MARKer<n>:Y[:REAL]?

**Description**
Sets the Y value of the specified marker.
Queries the Y value of the specified marker.

**Parameter**

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

**Remarks**
This command is used to set the marker's in-phase component (real part) of the Y value.

**Return Format**
The query returns the Y value or the marker's in-phase component of the Y value in scientific notation.

**Example**
The following command sets the Y value of Marker 1 to 0.325.
:CALCulate:DDEMod:MARKer1:Y:REAL 0.325

The following query returns 3.250000000e-01.
:CALCulate:DDEMod:MARKer1:Y:REAL?

## :CALCulate:DDEMod:MARKer<n>:Y:IMAGinary

**Syntax**
:CALCulate:DDEMod:MARKer<n>:Y:IMAGinary <real>
:CALCulate:DDEMod:MARKer<n>:Y:IMAGinary?

**Description**
Sets the quadrature component (imaginary) Y value of the specified marker.
Queries the quadrature component (imaginary) Y value of the specified marker.

**Parameter**

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

**Remarks**
Only when "Fixed" is selected for the marker mode and the marker is in the constellation or I-Q view, can these commands be valid.

**Return Format**
The query returns the quadrature component (imaginary) Y value of the specified marker in scientific notation.

**Example**
The following command sets the quadrature component Y value of Marker 1 to 0.435.

:CALCulate:DDEMod:MARKer1:Y:IMAGinary 0.435

The following query returns 4.350000000e-01.
:CALCulate:DDEMod:MARKer1:Y:IMAGinary?

## :CALCulate:DDEMod:MARKer<n>:WINDow

**Syntax**
:CALCulate:DDEMod:MARKer<n>:WINDow <enum>
:CALCulate:DDEMod:MARKer<n>:WINDow?

**Description**
Sets the marker window.
Queries the marker window.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <n> | Discrete | 1\|2\|3\|4\|5\|6\|7\|8 | —— |
| <enum> | Discrete | RTIMe\|SPECtrum\|MLOG\|MLIN\|MREAl\|MIMAg\|MPHAse\|MPHUnwrap\|MIQ\|MCONs\|MIEYe\|MQEYe\|MSPEctrum\|RLOG\|RLIN\|RREAl\|RIMAg\|RPHAse\|RPHUnwrap\|RIQ\|RCONs\|RIEYe\|RQEYe\|RSPEctrum\|ETIMe\|ESPEctrum\|EMag\|EPHAse | POSition |

**Remarks**
For detailed description of the parameter <enum>, refer to "Window ID Table" in Display Commands.

**Return Format**
The query returns RTIM, SPEC, MLOG, MLIN, MREA, MIMA, MPHA, MPHU, MIQ, MCON, MIEY, MQEY, MSPE, RLOG, RLIN, RREAl, RIMA, RPHA, RPHU, RIQ, RCON, RIEY, RQEY, RSPE, ETIM, ESPE, EM, or EPHA.

**Example**
The following command sets the Marker 1 window to SPECtrum.
:CALCulate:DDEMod:MARKer1:WINDow SPECtrum

The following query returns SPEC.
:CALCulate:DDEMod:MARKer1:WINDow?

## :CALCulate:DDEMod:MARKer<n>:REFerence

**Syntax**
:CALCulate:DDEMod:MARKer<n>:REFerence <integer>
:CALCulate:DDEMod:MARKer<n>:REFerence?

**Description**
Sets the reference marker for the specified marker.
Queries the reference marker for the specified marker.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <n> | Discrete | 1\|2\|3\|4\|5\|6\|7\|8 | —— |
| <integer> | Integer | 1 to 8 | By default, the reference marker is the marker next to it. |

**Remarks**
Each marker can have another marker to be its reference marker.
It the current marker is a Delta marker, the measurement result of the marker will be determined by the reference marker.
Any marker cannot have itself to be the reference marker.

**Example**
The following command sets the reference marker for Marker 1 to 2.
:CALCulate:DDEMod:MARKer1:REFerence 2

The following query returns 2.
:CALCulate:DDEMod:MARKer1:REFerence?

## :CALCulate:DDEMod:MARKer<n>:FUNCtion

**Syntax**
:CALCulate:DDEMod:MARKer<n>:FUNCtion <func>
:CALCulate:DDEMod:MARKer<n>:FUNCtion?

**Description**
Sets the band function type.
Queries the band function type.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <n> | Discrete | 1\|2\|3\|4\|5\|6\|7\|8 | —— |
| <func> | Discrete | BPOWer\|OFF | OFF |

**Remarks**
BPOWer: indicates the band power.
OFF: turns off all the measurements.
The commands are valid only when you enable the marker of the frequency-domain data.

**Return Format**
The query returns BPOW or OFF.

**Example**
The following command sets the measurement type of Marker 1 to band power.
:CALCulate:DDEMod:MARKer1:FUNCtion BPOW

The following query returns BPOW.
:CALCulate:DDEMod:MARKer1:FUNCtion?

## :CALCulate:DDEMod:MARKer<n>:FUNCtion:BAND:LEFT

**Syntax**
:CALCulate:DDEMod:MARKer<n>:FUNCtion:BAND:LEFT <real>
:CALCulate:DDEMod:MARKer<n>:FUNCtion:BAND:LEFT?

**Description**
Sets the left edge frequency or time for the band of the selected marker.
Queries the left edge frequency or time for the band of the selected marker.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <n> | Discrete | 1\|2\|3\|4\|5\|6\|7\|8 | —— |
| <freq> | Real | 0 to band right | center frequency-bandwidth/2 |

**Description**
The commands are valid only when you enable the marker of the frequency-domain data and enable the band function.

**Return Format**
The query returns the left edge frequency of the signal in scientific notation.

**Example**
The following command sets the left edge frequency of the signal involved in the calculation for the Marker 1 band function.
:CALCulate:DDEMod:MARKer1:FUNCtion:BAND:LEFT 2000000

The following query returns 2.000000000e+06.
:CALCulate:DDEMod:MARKer1:FUNCtion:BAND:LEFT?

## :CALCulate:DDEMod:MARKer<n>:FUNCtion:BAND:RIGHt

**Syntax**
:CALCulate:DDEMod:MARKer<n>:FUNCtion:BAND:RIGHt <real>
:CALCulate:DDEMod:MARKer<n>:FUNCtion:BAND:RIGHt?

**Description**
Sets the right edge frequency or time for the band of the selected marker.
Queries the right edge frequency or time for the band of the selected marker.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <n> | Discrete | 1\|2\|3\|4\|5\|6\|7\|8 | —— |
| <freq> | Real | band left to +∞ | center frequency+bandwidth/2 |

**Description**
The commands are valid only when you enable the marker of the frequency-domain data and enable the band function.

**Return Format**
The query returns the right edge frequency of the signal in scientific notation.

**Example**
The following command sets the right edge frequency of the signal involved in the calculation for the Marker 1 band function to 4 GHz.
:CALCulate:DDEMod:MARKer1:FUNCtion:BAND:RIGHt 4000000000

The following query returns 4.000000000e+09.
:CALCulate:DDEMod:MARKer1:FUNCtion:BAND:RIGHt?

## :CALCulate:DDEMod:MARKer<n>:FUNCtion:BAND:SPAN

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

**Description**
Sets the band span of the specified marker.
Queries the band span of the specified marker.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <n> | Discrete | 1\|2\|3\|4\|5\|6\|7\|8 | —— |
| <freq> | Real | 0 to +∞ | span/20 |

**Description**
The commands are valid only when you enable the marker of the frequency-domain data and enable the band function.

**Return Format**
The query returns the band span involved in the calculation for the band function in scientific notation.

**Example**
The following command sets the bandwidth of the signal involved in the calculation for the Marker 1 band function to 500 MHz.
:CALCulate:DDEMod:MARKer1:FUNCtion:BAND:SPAN 500000000

The following query returns 5.000000000e+08.
:CALCulate:DDEMod:MARKer1:FUNCtion:BAND:SPAN?

## :CALCulate:DDEMod:MARKer<n>[:SET]:CENTer

**Syntax**
:CALCulate:DDEMod:MARKer<n>[:SET]:CENTer

**Description**
Sets the frequency of the specified marker to center frequency of the analyzer.

**Parameter**

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

**Remarks**
If the marker mode of the specified marker is Position or Fixed, the center frequency will be set to the frequency of the marker.
If the specified marker mode is Delta, the center frequency will be set to the frequency of the Delta marker.

**Example**
The following command sets the frequency of Marker 1 (Position) to the center frequency of the analyzer.
:CALCulate:DDEMod:MARKer1:SET:CENTer

## :CALCulate:DDEMod:MARKer<n>[:SET]:DELTa:CENTer

**Syntax**
:CALCulate:DDEMod:MARKer<n>[:SET]:DELTa:CENTer

**Description**
Sets the center frequency of the analyzer to the frequency difference between the two Delta markers.

**Parameter**

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

**Remarks**
It is only valid when the current marker mode is "Delta".

**Example**
The following command sets the center frequency of the analyzer to the frequency difference between the two Delta markers.
:CALCulate:DDEMod:MARKer1:SET:DELTa:CENTer

## :CALCulate:DDEMod:MARKer<n>[:SET]:RLEVel

**Syntax**
:CALCulate:DDEMod:MARKer<n>[:SET]:RLEVel

**Description**
Sets the reference level of the analyzer to the amplitude of the specified marker.

**Parameter**

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

**Remarks**
If the marker mode of the specified marker is Position or Fixed, the reference level will be set to the amplitude of the marker.
If the specified marker mode is Delta and the current marker is the reference marker, then the reference level is set to the amplitude of the reference marker; if the current marker is the Delta marker, then the reference level is set to the amplitude of the Delta marker.

**Example**
The following command sets the reference level of the analyzer to the amplitude of Marker 2 (Position).
:CALCulate:DDEMod:MARKer2:SET:RLEVel

## :CALCulate:DDEMod:MARKer<n>[:SET]:STEP

**Syntax**
:CALCulate:DDEMod:MARKer<n>[:SET]:STEP

**Description**
Sets the frequency at the specified marker to the CF step of the analyzer.

**Parameter**

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

**Remarks**
If the marker mode of the specified marker is Position or Fixed, the center frequency step will be set to the frequency of the marker.
If the specified marker mode is Delta, the CF step is set to the frequency of the Delta marker.

**Example**
The following command sets the frequency at Marker 4 (Position) to the CF step of the analyzer.
:CALCulate:DDEMod:MARKer4:SET:STEP

## :CALCulate:DDEMod:MARKer<n>:MAXimum

**Syntax**
:CALCulate:DDEMod:MARKer<n>:MAXimum

**Description**
Performs one peak search and marks it with the specified marker.

**Parameter**

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

**Remarks**
When no peak is found, a prompt message "No peak found" is displayed on the screen.

**Example**
The following command performs one peak search, and marks with Marker 2.
:CALCulate:DDEMod:MARKer2:MAXimum

## :CALCulate:DDEMod:MARKer<n>:MAXimum:LEFT

**Syntax**
:CALCulate:DDEMod:MARKer<n>:MAXimum:LEFT

**Description**
Searches for and marks the peak closest to the left side of the current peak.

**Parameter**

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

**Remarks**
When no peak is found, a prompt message "No peak found" is displayed on the screen.

**Example**
The following command performs one left peak search, and marks with Marker 2.
:CALCulate:DDEMod:MARKer2:MAXimum:LEFT

## :CALCulate:DDEMod:MARKer<n>:MAXimum:NEXT

**Syntax**
:CALCulate:DDEMod:MARKer<n>:MAXimum:NEXT

**Description**
Searches for and marks the peak whose amplitude on the trace is next lower than that of the current peak.

**Parameter**

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

**Remarks**
When no peak is found, a prompt message "No peak found" is displayed on the screen.

**Example**
The following command performs one next peak search, and marks with Marker 2.
:CALCulate:DDEMod:MARKer2:MAXimum:NEXT

## :CALCulate:DDEMod:MARKer<n>:MAXimum:PREVious

**Syntax**
:CALCulate:DDEMod:MARKer<n>:MAXimum:PREVious

**Description**
Searches for and marks the peak whose amplitude on the trace is next higher than that of the current peak.

**Parameter**

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

**Remarks**
When no peak is found, a prompt message "No peak found" is displayed on the screen.

**Example**
The following command performs one next peak search, and marks with Marker 2.
:CALCulate:DDEMod:MARKer2:MAXimum:PREVious

## :CALCulate:DDEMod:MARKer<n>:MAXimum:RIGHt

**Syntax**
:CALCulate:DDEMod:MARKer<n>:MAXimum:RIGHt

**Description**
Searches for and marks the peak closest to the right side of the current peak.

**Parameter**

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

**Remarks**
When no peak is found, a prompt message "No peak found" is displayed on the screen.

**Example**
The following command performs one right peak search, and marks with Marker 2.
:CALCulate:DDEMod:MARKer2:MAXimum:RIGHt

## :CALCulate:DDEMod:MARKer<n>:MINimum

**Syntax**
:CALCulate:DDEMod:MARKer<n>:MINimum

**Description**
Searches for and marks the peak with the minimum amplitude on the trace.

**Parameter**

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

**Remarks**
When no peak is found, a prompt message "No peak found" is displayed on the screen.

**Example**
The following command performs one minimum search, and marks it with Marker 2.
:CALCulate:DDEMod:MARKer2:MINimum

## :CALCulate:DDEMod:MARKer<n>:CPSearch[:STATe]

**Syntax**
:CALCulate:DDEMod:MARKer<n>:CPSearch[:STATe] <bool>
:CALCulate:DDEMod:MARKer<n>:CPSearch[:STATe]?

**Description**
Sets the on/off status of continuous peak search.
Queries the on/off status of continuous peak search.

**Parameter**

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

**Remarks**
N/A

**Example**
The following command enables the continuous peak search of Marker 1.
:CALCulate:DDEMod:MARKer1:CPSearch:STATe ON or :CALCulate:DDEMod:MARKer1:CPSearch:STATe 1

The following query returns 1.
:CALCulate:DDEMod:MARKer1:CPSearch:STATe?

## :CALCulate:DDEMod:MARKer:AOFF

**Syntax**
:CALCulate:DDEMod:MARKer:AOFF

**Description**
Turns off all the enabled markers.

## :CALCulate:DDEMod:MARKer:COUPle[:STATe]

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

**Description**
Enables or disables the couple marker function.
Queries the state of the couple marker function.

**Parameter**

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

**Remarks**
When you enable the couple marker function, moving any marker will make other markers (that are not fixed or off) move along with it.

**Return Format**
The query returns 0 or 1.

**Example**
The following command disables the couple marker function.
:CALCulate:DDEMod:MARKer:COUPle:STATe OFF or :CALCulate:DDEMod:MARKer:COUPle:STATe 0

The following query returns 0.
:CALCulate:DDEMod:MARKer:COUPle:STATe?

## :CALCulate:DDEMod:MARKer:SELect

**Syntax**
:CALCulate:DDEMod:MARKer:SELect <num>
:CALCulate:DDEMod:MARKer:SELect?

**Description**
Sets the selected marker.
Queries the currently selected marker.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <num> | Integer | 1\|2\|3\|4\|5\|6\|7\|8 | —— |

**Return Format**
The query returns the current marker in integer.

**Example**
The following command sets Marker 2 to be the selected marker.
:CALCulate:DDEMod:MARKer:SELect 2

The following query command returns 2.
:CALCulate:DDEMod:MARKer:SELect?

## :CALCulate:DDEMod:MARKer:NEXT

**Syntax**
:CALCulate:DDEMod:MARKer:NEXT

**Description**
Selects the next marker.

# :CALibration Commands

## :CALibration[:ALL]

**Syntax**
:CALibration[:ALL]

**Description**
Executes self-calibration immediately.

**Example**
The following command executes the self-calibration immediately.
:CALibration:ALL

## :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**
The following command enables auto calibration.
:CALibration:AUTO ON or :CALibration:AUTO 1

The following query returns 1.
:CALibration:AUTO?

# :CONFigure Commands

## :CONFigure?

**Syntax**
:CONFigure?

**Description**
Queries the current measurement function.

**Return Format**
The query returns SA, RTSA, EMI, VSA, AM, FM, PM, GPSA_TPOW, GPSA_ACP, GPSA_CHP, GPSA_OBW, GPSA_CNR, GPSA_HARM, GPSA_TOI, GPSA_TG, or RTSA_UFS.

## :CONFigure:CATalog?

**Syntax**
:CONFigure:CATalog?

**Description**
Queries the currently supported measurement application.

# :DISPlay Commands

**Window ID Table**

| Window ID | Window Name | Window Parameter | Window ID | Window Name | Window Parameter |
|---|---|---|---|---|---|
| 23 | Raw Time | RTIMe | 41 | RefReal | RREAl |
| 26 | Spectrum | SPECtrum | 42 | RefImag | RIMAg |
| 27 | MeasLog | MLOG | 43 | RefPhase | RPHAse |
| 28 | MeasLin | MLIN | 44 | RefPhaseUnwrap | RPHUnwrap |
| 29 | MeasReal | MREAl | 45 | RefIQ | RIQ |
| 30 | MeasImag | MIMAg | 46 | RefCons | RCONs |
| 31 | MeasPhase | MPHAse | 47 | RefIEye | RIEYe |
| 32 | MeasPhaseUnwrap | MPHUnwrap | 48 | RefQEye | RQEYe |
| 33 | MeasIQ | MIQ | 50 | RefSpectrum | RSPEctrum |
| 34 | MeasCons | MCONs | 51 | EVMErrorTime | ETIMe |
| 35 | MeasIEye | MIEYe | 52 | EVMErrorSpectrum | ESPEctrum |
| 36 | MeasQEye | MQEYe | 53 | MagError | EMag |
| 38 | MeasSpectrum | MSPEctrum | 54 | PhaseError | EPHAse |
| 39 | RefLog | RLOG | 55 | Meas | |
| 40 | RefLin | RLIN | 56 | Demod Bits | |

## :DISPlay:WINDow<n>:SWITch

**Syntax**
:DISPlay:WINDow<n>:SWITch

**Description**
Switches the window.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <n> | Discrete | Refer to "Remarks" | - |
| <integer> | Integer | Refer to "Remarks" | - |

**Remarks**
The range of parameter <n> and <integer> ranges from 23 to 56. For details, refer to "Window ID Table" in Display Commands.

**Example**
The following command switches the window view from 35 to 36.
:DISPlay:WINDow35:SWITch 36

## :DISPlay:DDEMod:VIEW[:SELect]

**Syntax**
:DISPlay:DDEMod:VIEW[:SELect] <enum>
:DISPlay:DDEMod:VIEW[:SELect]?

**Description**
Sets the preset combination view.
Queries the preset combination view.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <enum> | Discrete | NORMal\|DTRace\|DERRor\|NRESults | NORMal |

**Remarks**
NORMal: indicates normal view.
DTRace: indicates Demod Trace view.
DERRor: indicates Demod Error view.
NRESults: indicates Result Summary view.

**Return Format**
The query returns NORM, DTR, DERR, or NRES.

**Example**
The following command sets the preset combination view to DTRace.
:DISPlay:DDEMod:VIEW:SELect DTRace
The following query returns DTR.
:DISPlay:DDEMod:VIEW:SELect?

## :DISPlay:DDEMod:<n> X[:SCALe]:COUPle

**Syntax**
:DISPlay:DDEMod:WINDOW<n>:X[:SCALe]:COUPle <bool>
:DISPlay:DDEMod:WINDOW<n>:X[:SCALe]:COUPle?

**Description**
Enables or disables auto adjustment of X Scale.
Queries the on/off status of auto adjustment of X Scale.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <n> | Discrete | Refer to "Remarks" | —— |
| <bool> | Bool | OFF\|ON\|0\|1 | ON\|1 |

**Remarks**
ON: enables auto adjustment of X Scale. The instrument auto adjusts the reference value and width of X scale to an optimal state.
OFF: disables auto adjustment of X Scale. The reference value and width of X scale need to be adjusted manually.
The range of parameter <n> ranges from 23 to 56. For details, refer to "Window ID Table" in :DISPlay Commands.

**Return Format**
The query returns 0 or 1.

**Example**
The following command enables the auto adjustment of X scale.
:DISPlay:DDEMod:WINDOW 23:X:SCALe:COUPle ON or :DISPlay:DDEMod:WINDOW 23:X:SCALe:COUPle 1

The following query returns 1.
:DISPlay:DDEMod:WINDOW 23:X:SCALe:COUPle?

## :DISPlay:DDEMod:WINDOW<n>:X[:SCALe]:RLEVel

**Syntax**
:DISPlay:DDEMod:WINDOW<n>:X[:SCALe]:RLEVel <real>
:DISPlay:DDEMod:WINDOW<n>:X[:SCALe]:RLEVel?

**Description**
Sets the reference value of X scale.
Queries the reference value of X scale.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <n> | Discrete | Refer to "Remarks" | —— |
| <real> | Real | (-1e+12) to (1e+12) (Time-Domain Data)<br>-1 THz to 1 THz (Frequency-Domain Data) | 25 (19.1406 us)<br>/1 GHz |

**Remarks**

The command is valid when you disable the auto adjustment of X scale.
When the time-domain data source is selected, the reference value unit is symbol or s; when the frequency-domain data source is selected, the reference value unit is Hz.
The range of parameter <n> ranges from 23 to 56. For details, refer to "Window ID Table" in :DISPlay Commands.

**Return Format**

The query returns the X-axis reference of the selected trace in scientific notation.

**Example**

The following command sets the reference value of X scale to 25.
:DISPlay:DDEMod:WINDOW23:X:SCALe:RLEVel 25

The following query returns 2.500000000e+01.
:DISPlay:DDEMod:WINDOW23:X:SCALe:RLEVel?

## :DISPlay:DDEMod:WINDOW<n>:X[:SCALe]:RPOSition

**Syntax**

:DISPlay:DDEMod:WINDOW<n>:X[:SCALe]:RPOSition <enum>
:DISPlay:DDEMod:WINDOW<n>:X[:SCALe]:RPOSition?

**Description**

Sets the reference position of X scale.
Queries the reference position of X scale.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <n> | Discrete | Refer to "Remarks" | —— |
| <enum> | Discrete | LEFT\|CENTer\|RIGHt | LEFT |

**Remarks**

The range of parameter <n> ranges from 23 to 56. For details, refer to "Window ID Table" in :DISPlay Commands.

**Return Format**

The query returns LEFT, CENT, or RIGH.

**Example**

The following command sets the reference position of X scale to Left.
:DISPlay:DDEMod:WINDOW23:X:SCALe:RPOSition LEFT

The following query returns LEFT.
:DISPlay:DDEMod:WINDOW23:X:SCALe:RPOSition?

## :DISPlay:DDEMod:WINDOW<n>:X[:SCALe]:WIDTh

**Syntax**

:DISPlay:DDEMod:WINDOW<n>:X[:SCALe]:WIDTh <real>
:DISPlay:DDEMod:WINDOW<n>:X[:SCALe]:WIDTh?

**Description**
Sets the width of X scale.
Queries the width of X scale.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <n> | Discrete | Refer to "Remarks" | —— |
| <real> | Real | 0 to (1e+12) (Time-Domain Data)<br>0 Hz to 1 THz (Frequency-Domain Data) | 50 (38.2812 us)/3.125 MHz |

**Remarks**
The command is valid when you disable the auto adjustment of X scale.
When the time-domain data source is selected, the reference value unit is symbol or s; when the frequency-domain data source is selected, the reference value unit is Hz.
The range of parameter <n> ranges from 23 to 56. For details, refer to "Window ID Table" in :DISPlay Commands..

**Return Format**
The query returns the width of X scale in scientific notation.

**Example**
The following command sets the width of X scale to 50.
:DISPlay:DDEMod:WINDOW23:X:SCALe:WIDTh 50

The following query returns 5.000000000e+01.
:DISPlay:DDEMod:WINDOW23:X:SCALe:WIDTh?

## :DISPlay:DDEMod:WINDOW<n>:X[:SCALe]:LAST

**Syntax**
:DISPlay:DDEMod:WINDOW<n>:X[:SCALe]:LAST

**Description**
Sets the last scale.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <n> | Discrete | Refer to "Remarks" | —— |

**Remarks**
The range of parameter <n> ranges from 23 to 56. For details, refer to "Window ID Table" in :DISPlay Commands.

**Example**
The following command sets the last scale.
:DISPlay:DDEMod:WINDOW23:X:SCALe:LAST

## :DISPlay:DDEMod:WINDOW<n>:X[:SCALe]:COPY

**Syntax**
:DISPlay:DDEMod:WINDOW<n>:X[:SCALe]:COPY <enum>

**Description**
Copies X Scale.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <n> | Discrete | Refer to "Remarks" | —— |
| <enum> | Discrete | RTIMe\|SPECtrum\|MLOG\|MLIN\|MREAl\|MIMAg\|MPHAse\|MPHUnwrap\|MIQ\|MCONs\|MIEYe\|MQEYe\|MSPEctrum\|RLOG\|RLIN\|RREAl\|RIMAg\|RPHAse\|RPHUnwrap\|RIQ\|RCONs\|RIEYe\|RQEYe\|RSPEctrum\|ETIMe\|ESPEctrum\|EMag\|EPHAse | —— |

**Remarks**

The range of parameter <n> ranges from 23 to 56. For details, refer to "Window ID Table" in :DISPlay Commands.

For detailed description of the parameter <enum>, refer to "Window ID Table" in :DISPlay Commands.

**Example**

The following command copies the X Scale of Spectrum view (26) to the MeasSpectrum view.

:DISPlay:DDEMod:WINDOW26:X:SCALe:COPY MSPEctrum

## :DISPlay:DDEMod:WINDOW<n>:Y[:SCALe]:AUTO:ONCE

**Syntax**

:DISPlay:DDEMod:WINDOW<n>:Y[:SCALe]:AUTO:ONCE

**Description**

Enables the auto adjustment of Y scale.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <n> | Discrete | Refer to "Remarks" | —— |

**Remarks**

The range of parameter <n> ranges from 23 to 56. For details, refer to "Window ID Table" in :DISPlay Commands..

**Example**

The following command performs the auto adjustment of Y scale.

:DISPlay:DDEMod:WINDOW23:Y:SCALe:AUTO:ONCE

## :DISPlay:DDEMod:WINDOW<n>:Y[:SCALe]:RLEVel

**Syntax**

:DISPlay:DDEMod:WINDOW<n>:Y[:SCALe]:RLEVel <real>

:DISPlay:DDEMod:WINDOW<n>:Y[:SCALe]:RLEVel?

**Description**

Sets the reference value of Y scale.

Queries the reference value of Y scale.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <n> | Discrete | Refer to "Remarks" | —— |
| <ampl> | Real | (-1e+12) to (1e+12) | 0 |

**Remarks**

The unit of the reference value is determined by the Y scale unit.

The range of parameter <n> ranges from 23 to 56. For details, refer to "Window ID Table" in :DISPlay Commands.

**Return Format**
The query returns the reference value in scientific notation.

**Example**
The following command sets the reference value of Y scale to -10.
:DISPlay:DDEMod:WINDOW23:Y:SCALe:RLEVel -10

The following query returns -1.000000000e+01.
:DISPlay:DDEMod:WINDOW23:Y:SCALe:RLEVel?

## :DISPlay:DDEMod:WINDOW<n>:Y[:SCALe]:RPOSition

**Syntax**
:DISPlay:DDEMod:WINDOW<n>:Y[:SCALe]:RPOSition <enum>
:DISPlay:DDEMod:WINDOW<n>:Y[:SCALe]:RPOSition?

**Description**
Sets the reference position of Y scale.
Queries the reference position of Y scale.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <n> | Discrete | Refer to "Remarks" | —— |
| <enum> | Discrete | TOP\|CENTer\|BOTTom | CENTer |

**Remarks**
The range of parameter <n> ranges from 23 to 56. For details, refer to "Window ID Table" in :DISPlay Commands.

**Return Format**
The query returns TOP, CENT, or BOTT.

**Example**
The following command sets the reference position of Y scale to TOP.
:DISPlay:DDEMod:WINDOW 23:Y:SCALe:RPOSition TOP

The following query returns TOP.
:DISPlay:DDEMod:WINDOW23:Y:SCALe:RPOSition?

## :DISPlay:DDEMod:WINDOW<n>:Y[:SCALe]:PDIVision

**Syntax**
:DISPlay:DDEMod:WINDOW<n>:Y[:SCALe]:PDIVision <rel>
:DISPlay:DDEMod:WINDOW<n>:Y[:SCALe]:PDIVision?

**Description**
Sets the Y scale/div.
Queries the Y scale/div.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <n> | Discrete | Refer to "Remarks" | —— |
| <rel> | Real | (-1e+09) to (1e+12) | 0.3 |

**Remarks**
The range of parameter <n> ranges from 23 to 56. For details, refer to "Window ID Table" in :DISPlay Commands..

**Return Format**
The query returns Y scale/div in scientific notation.

**Example**
The following command sets the Y scale/div to 0.3.
:DISPlay:DDEMod:WINDOW23:Y:SCALe:PDIVision 0.3

The following command returns 3.000000000e-01.
:DISPlay:DDEMod:WINDOW23:Y:SCALe:PDIVision?

## :DISPlay:DDEMod:WINDOW<n>:Y:UNIT:PREFerence

**Syntax**
:DISPlay:DDEMod:WINDOW<n>:Y:UNIT:PREFerence <enum>
:DISPlay:DDEMod:WINDOW<n>:Y:UNIT:PREFerence?

**Description**
Sets the Y-axis unit of the specified trace.
Queries the Y-axis unit of the specified trace.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <n> | Discrete | Refer to "Remarks" | —— |
| <enum> | Discrete | PEAK\|RMS\|POWer | PEAK |

**Remarks**
The range of parameter <n> ranges from 23 to 56. For details, refer to "Window ID Table" in :DISPlay Commands..
This command is valid when the window is "RawTime" or "Spectrum".

**Return Format**
The query returns PEAK, RMS, or POW.

**Example**
The following command sets Y-axis unit of the Trace 1 to Peak.
:DISPlay:DDEMod:WINDOW23:Y:UNIT:PREFerence PEAK

The following query returns PEAK.
:DISPlay:DDEMod:WINDOW23:Y:UNIT:PREFerence?

## :DISPlay:DDEMod:WINDOW<n>:Y[:SCALe]:LAST

**Syntax**
:DISPlay:DDEMod:WINDOW<n>:Y[:SCALe]:LAST

**Description**
Sets last scale of Y-axis.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <n> | Discrete | Refer to "Remarks" | —— |

**Remarks**
The range of parameter <n> ranges from 23 to 56. For details, refer to "Window ID Table" in :DISPlay Commands..

## :DISPlay:DDEMod:WINDOW<n>:Y[:SCALe]:COPY

**Syntax**
:DISPlay:DDEMod:WINDOW<n>:Y[:SCALe]:COPY <enum>

**Description**
Copies Y scale.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <n> | Discrete | Refer to "Remarks" | —— |
| <enum> | Discrete | RTIMe\|SPECtrum\|MLOG\|MLIN\|MREAl\|MIMAg\|MPHAse\| MPHUnwrap\|MIQ\|MCONs\|MIEYe\|MQEYe\|MSPEctrum\|RLOG\| RLIN\|RREAl\|RIMAg\|RPHAse\|RPHUnwrap\|RIQ\|RCONs\|RIEYe\| RQEYe\|RSPEctrum\|ETIMe\|ESPEctrum\|EMag\|EPHAse | —— |

**Remarks**
The range of parameter <n> ranges from 23 to 56. For details, refer to "Window ID Table" in :DISPlay Commands..
For detailed description of the parameter <enum>, refer to "Window ID Table" in :DISPlay Commands..

**Example**
The following command copies the Y Scale of Spectrum view (26) to the MeasSpectrum view.
:DISPlay:DDEMod:WINDOW26:Y:SCALe:COPY MSPEctrum

## :DISPlay:DDEMod:WINDOW<n>:EYE:INTerval

**Syntax**
:DISPlay:DDEMod:WINDOW<n>:EYE:INTerval <integer>
:DISPlay:DDEMod:WINDOW<n>:EYE:INTerval?

**Description**
Sets the eye length.
Queries the eye length.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <n> | Discrete | Refer to "Remarks" | —— |
| <integer> | Integer | 1 to 40 | 2 |

**Remarks**
The range of parameter <n> and <integer> ranges from 23 to 56. This command is valid when <n> is set to 35, 36, 47, or 48. For details, refer to "Window ID Table" in :DISPlay Commands..
.

**Return Format**
The query returns the eye length in integer.

**Example**
The following command sets the eye length to 20.
:DISPlay:DDEMod:WINDOW36:EYE:INTerval 20

The following query returns 20.
:DISPlay:DDEMod:WINDOW36:EYE:INTerval?

## :DISPlay:DDEMod:BITS:FORMat

**Syntax**
:DISPlay:DDEMod:BITS:FORMat <enum>
:DISPlay:DDEMod:BITS:FORMat?

**Description**
Sets the bit format.
Queries the bit format.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <enum> | Discrete | HEX\|BINary | - |

**Remarks**
Hex: indicates Hexadecimal;.
BINary: indicates the binary format.

**Return Format**
The query returns HEX or BIN.

**Example**
The following command sets the bit format to HEX.
:DISPlay:DDEMod:BITS:FORMat HEX

The following query returns HEX.
:DISPlay:DDEMod:BITS:FORMat?

## :DISPlay:DDEMod:TXBit:PATTern

**Syntax**
:DISPlay:DDEMod:TXBit:PATTern <bool>
:DISPlay:DDEMod:TXBit:PATTern?

**Description**
Enables or disables the display of the reference bits.
Queries the on/off status of the reference bits.

**Parameter**

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

**Return Format**
The query returns 0 or 1.

**Example**
The following command enables the display of the reference bits.
:DISPlay:DDEMod:TXBit:PATTern ON or :DISPlay:DDEMod:TXBit:PATTern 1

The following query returns 1.
:DISPlay:ENABle?

# :FETCh Commands

## :FETCh:DDEMod<n>?

**Syntax**
:FETCh:DDEMod<n>?

**Description**
Queries the digital demodulation measurement result.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <n> | Integer | Refer to **"Remarks"** | - |

**Remarks**
The relationship between parameter n and the return value is as follows:

| Parameter n | Return Value | Parameter n | Return Value |
|---|---|---|---|
| 0 | Raw Time | 16 | RefLin |
| 1 | Measurement Result | 17 | RefReal |
| 2 | Spectrum | 18 | RefImag |
| 3 | MeasLog | 19 | RefPhase |
| 4 | MeasLin | 20 | RefPhaseUnwrap |
| 5 | MeasReal | 21 | RefIQ |
| 6 | MeasImag | 22 | RefCons |
| 7 | MeasPhase | 23 | RefIEye |
| 8 | MeasPhaseUnwrap | 24 | RefQEye |
| 9 | MeasIQ | 25 | Reserve |
| 10 | MeasCons | 26 | RefSpectrum |
| 11 | MeasIEye | 27 | EVMErrorTime |
| 12 | MeasQEye | 28 | EVMErrorSpectrum |
| 13 | Reserve | 29 | MagError |
| 14 | MeasSpectrum | 30 | PhaseError |
| 15 | RefLog | 31 | Demod Bits |

# 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 the enable register for the standard event status register.
Queries the enable register for 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 the enable register for the status byte register.
Queries the bits in the service request 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 set, 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**
The following command sets the instrument to sweep continuously.
:INITiate:CONTinuous ON or :INITiate:CONTinuous 1

The following query returns 1.
:INITiate:CONTinuous?

## :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 the sweep.

# :INPut Commands

## :INPut:IF:OVERload?

Syntax
:INPut:IF:OVERload?

**Description**
Queries the IF overload.

**Remarks**
The query returns 0 or 1.
0 indicates not overloaded; 1 indicates overloaded.

# :INSTrument Commands

## :INSTrument:COUPle:FREQuency:CENTer

**Syntax**
:INSTrument:COUPle:FREQuency:CENTer <enum>
:INSTrument:COUPle:FREQuency:CENTer?

**Description**
Sets the setting status of the global center frequency of the instrument.
Queries the setting status of the global center frequency of the instrument.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <enum> | Discrete | ALL\|NONE | NONE |

**Remarks**
NONE: turns off the global center frequency.
ALL: turns on the global center frequency.
If you execute this command in any mode, the center frequency of the current mode is set to the global center frequency. Adjusting the center frequency in a mode, while the global center frequency is on, will modify the global center frequency.

**Return Format**
The query returns ALL or NONE.

**Example**
The following command enables the global center frequency of the instrument.
:INSTrument:COUPle:FREQuency:CENTer ALL

The following query returns ALL.
:INSTrument:COUPle:FREQuency:CENTer?

## :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:REName

**Syntax**
:INSTrument:SCReen:REName<name>

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

**Parameter**

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

## :INSTrument:SCReen:SELect

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

**Description**
Selects the specified measurement mode label at the bottom of the screen.
Queries the measurement mode label at the bottom of the screen that you select currently.

**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**
The following command sets to select measurement mode label RTSA 1 at the bottom of the screen.
:INSTrument:SCReen:SELect RTSA 1
The following query returns RTSA 1.
:INSTrument:SCReen:SELect?

## :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\|<br>GPSA_CNR\|GPSA_CHPower\|GPSA_TOI\|GPSA_HARModist\|<br>GPSA_OBW\|GPSA_TPOWer\|EMI\|VSA\|AM\|FM\|PM | 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**
The following command sets the working mode of the instrument to GPSA.
:INSTrument:SELect SA

The following query returns 1 or SA.
:INSTrument:SELect?

# :MMEMory Commands

## :MMEMory:DELete

**Syntax**
:MMEMory:DELete <file_name>

**Description**
Deletes a specified file.

**Parameter**

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

**Remarks**
<file_name> should contain the path and the filename.
This operation fails if the file with the specified filename does not exist.

**Example**
The following command deletes the "state1.sta" file from the "/ VSA/state" folder.
:MMEMory:DELete /VSA/state/state1.sta

## :MMEMory:LOAD:STATe

**Syntax**
:MMEMory:LOAD:STATe <file_name>

**Description**
Loads the specified state file (.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**
The following command loads the state file (state1.sta) to the instrument.
:MMEMory:LOAD:STATe state1.sta

## :MMEMory:LOAD:RESults

**Syntax**
:MMEMory:LOAD:RESults <label>,<path>

**Description**
Loads the measurement results.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <label> | Discrete | RAWDATA | —— |
| <path> | Real | —— | -- |

**Remarks**

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

**Example**
The following command loads the rawdata.csv file to the raw data.
:MMEMory:LOAD:RESults RAWDATA1,rawdata1.csv

## :MMEMory:MOVE

**Syntax**
:MMEMory:MOVE <file_name1>,<file_name2>

**Description**
Renames the specified file <file_name1> as <file_name2>.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <file_name1> | ASCII String | —— | -- |
| <file_name2> | ASCII String | —— | -- |

**Remarks**
<file_name1> and <file_name2> should contain the path and the filename.
This operation fails if the file with the specified filename does not exist.

**Example**
The following command renames the state file (state1.sta) in the folder (/GPSA/state) as "state2.sta".
:MMEMory:MOVE /GPSA/state/state1.sta,/GPSA/state/state2.sta

## :MMEMory:STORe:IMG:PNG

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

**Description**
Saves the screen image to the specified path.

**Parameter**

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

**Remarks**
<path> should contain the path and the filename.

**Example**
The following command sets the saving path of the image to "/data/UserData/vsa/image.png".
:MMEMory:STORe:IMG:PNG /data/UserData/vsa/image.png

## :MMEMory:STORe:STATe

**Syntax**
:MMEMory:STORe:STATe <file_name>

:MMEMory:STORe:STATe?

**Description**
Saves the current instrument state with the specified filename suffixed with ".sta" to the default path (/"mode name"/state).

**Parameter**

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

**Remarks**
If the specified file already exists, overwrite it.

**Example**
The following command saves the current instrument state with the filename "state.sta" to the folder (/VSA/state).
:MMEMory:STORe:STATe state

The following command returns /VSA/state/state.sta.
:MMEMory:STORe:STATe?

## :MMEMory:STORe:RESults

**Syntax**
:MMEMory:STORe:RESults <file_name>

**Description**
Saves the raw data of the current measurement results with a specified filename suffixed with ".csv" by default (you do not have to add the suffix manually) to the default path (/"mode name"/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 current measurement results with the specified filename "data" to the folder (/VSA/measdata).
:MMEMory:STORe:RESults data

## :MMEMory:STORe:RETurn?

**Syntax**
:MMEMory:STORe:RETurn?

**Description**
Saves the returned results.

## :MMEMory:STORe:SCReen

**Syntax**
:MMEMory:STORe:SCReen <file_name>

**Description**
Saves the current screen image with the specified filename suffixed with "*.jpg", "*.png/", or "*.bmp" to the default path (/"mode name"/screen).

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <file_name> | 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**
The following command saves the current screen image with the filename "screen.jpg" to the folder (/VSA/screen).
:MMEMory:STORe:SCReen screen.jpg

## :MMEMory:STORe:SCReen:DATA?

**Syntax**
:MMEMory:STORe:SCReen:DATA?

**Description**
Saves the screen data.

## :MMEMory:USER:STORe

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

**Description**
Sets the preset settings.
Queries the preset settings.

**Parameter**

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

**Remarks**
N/A

**Example**
The following command selects USER1.
:MMEMory:USER:STORe USER1

The following query command returns USER1.
:MMEMory:USER:STORe?

# :SAVE:MEASure

**Syntax**
:SAVE:MEASure <type>,<path>

**Description**
Saves the current measurement data with a specified filename suffixed with ".csv" by default (you do not have to add the suffix manually) to the default path (/"mode name"/measdata).

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <type> | Discrete | RESULT\|RAWDATA | —— |
| <path> | ASCII String | —— | —— |

**Remarks**
If the specified file already exists, overwrite it.
RESULT: measurement results.
RAWDATA: raw data.

**Example**
The following command saves the current measurement results with the specified filename "measdata1" to the storage path "/VSA/measdata".
:SAVE:MEASure RESULT,measdata1

# [:SENSe] Commands

## [:SENSe]:POWer[:RF]:RANGe

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

Description
Sets the amplitude of the largest sinusoidal input signal without being clipped by the IF ADC.
Queries the amplitude of the largest sinusoidal input signal without being clipped by the IF ADC.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <real> | Real | -15 dBm to 25 dBm | 20 dBm |

**Return Format**
The query returns the signal amplitude value in scientific notation. The unit is dBm.

**Example**
The following command sets the amplitude of the largest sinusoidal input signal without being clipped by the IF ADC to 20 dBm.
:SENSe:POWer:RF:RANGe 20

The following query returns 2.000000000e+01.
:SENSe:POWer:RF:RANGe?

## [:SENSe]:DDEMod[:SEGMent1]:FFT:WINDow[:TYPE]

**Syntax**
[:SENSe]:DDEMod[:SEGMent1]:FFT:WINDow[:TYPE] <type>
[:SENSe]:DDEMod[:SEGMent1]:FFT:WINDow[:TYPE]?

**Description**
Sets the type of the FFT window function.
Queries the type of the FFT window function.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <type> | Keyword | UNIForm\|HANNing\|GAUSsian\|FLATtop | UNIForm |

**Return Format**
The query returns UNIF, HANN, GAUS, or FLAT.

**Example**
The following command sets the FFT window type to Gaussian.
:SENSe:DDEMod:FFT:WINDow:TYPE GAUSsian

The following query returns GAUS.
:SENSe:DDEMod:FFT:WINDow:TYPE?

## [ :SENSe]:DDEMod[:SEGMent1]:MODulation

**Syntax**
[:SENSe]:DDEMod[:SEGMent1]:MODulation <type>
[:SENSe]:DDEMod[:SEGMent1]:MODulation?

**Description**
Sets the modulation format.
Queries the modulation format.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <type> | Discrete | ASK2\|ASK4\|FSK2\|FSK4\|FSK8\|MSK1\|MSK2\|BPSK\|QPSK\|<br>PSK8\|DQPSK\|DPSK8\|PI4DQPSK\|PI8DPSK8\|OQPSK\|QAM16\|<br>QAM32\|QAM64\|QAM128\|QAM256\|QAM512\|QAM1024 | - |

**Return Format**
The query returns ASK2, ASK4, FSK2, FSK4, FSK8, MSK1, MSK2, BPSK, QPSK, PSK8, DQPSK, DPSK8, PI4DQPSK, PI8DPSK8, OQPSK, QAM16, QAM32, QAM64, QAM128, QAM256, QAM512, or QAM1024.

**Example**
The following command sets the modulation format to ASK2.
:SENSe:DDEMod:MODulation ASK2

The following query returns ASK2.
:SENSe:DDEMod:MODulation?

## [:SENSe]:DDEMod[:SEGMent1]:SRATe

**Syntax**
[:SENSe]:DDEMod[:SEGMent1]:SRATe <frequency>
[:SENSe]:DDEMod[:SEGMent1]:SRATe?

**Description**
Sets the symbol rate.
Queries the symbol rate.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <frequency> | Integer | Refer to "Remarks" | 1 Msps |

**Remarks**
Min Symbol Rate = 1 kHz.
Max. Symbol Rate SRmax = SPmax x 1.28/(Points/Symbol). Wherein, SPmax indicates the max span.

**Return Format**
The query returns the symbol rate in integer.

**Example**
The following command sets the symbol rate to 1 Msps.
:SENSe:DDEMod:SRATe 1 Msps

The following query returns 1000000.
:SENSe:DDEMod:SRATe?

## [:SENSe]:DDEMod[:SEGMent1]:PPSYmbol

**Syntax**
[:SENSe]:DDEMod[:SEGMent1]:PPSYmbol <integer>
[:SENSe]:DDEMod[:SEGMent1]:PPSYmbol?

**Description**
Sets Points/symbol of the digital demodulation.

Queries Points/symbol of the digital demodulation.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <integer> | Integer | 4\|8\|16 | 4 |

**Return Format**
The query returns Points/symbol in integer.

**Example**
The following command sets the number of sample points/symbol to 4.
:SENSe:DDEMod:PPSYmbol 4

The following query returns 4.
:SENSe:DDEMod:PPSYmbol?

## [:SENSe]:DDEMod[:SEGMent1]SWEep:POINts

**Syntax**
[:SENSe]:DDEMod[:SEGMent1]:SWEep:POINts <integer>
[:SENSe]:DDEMod[:SEGMent1]:SWEep:POINts?

**Description**
Sets the measurement interval.
Queries the measurement interval.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <integer> | Integer | 10 to 4,096 | 50 |

**Remarks**
N/A

**Return Format**
The query returns the measurement interval in integer.

**Example**
The following command sets the measurement interval to 200.
:SENSe:DDEMod:SWEep:POINts 200

The following query returns 200.
:SENSe:DDEMod:SWEep:POINts?

## [:SENSe]:DDEMod[:SEGMent1]:FILTer:MEASurement

**Syntax**
[:SENSe]:DDEMod[:SEGMent1]:FILTer:MEASurement <type>
[:SENSe]:DDEMod[:SEGMent1]:FILTer:MEASurement?

**Description**
Sets the measurement filter type.
Queries the measurement filter type.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <type> | Discrete | NONE\|RRCosine\|GAUSsian\|RECTangle\|<br>USER1\|USER2\|USER3\|USER4\|USER5\|USER6 | RRCosine |

**Remarks**
NONE: disables the filter.
RRCosine: indicates the root raised cosine.
GAUSsian: indicates the Gaussian filter.
RECTangle: indicates the rectangular filter.
USER<n>: indicates the user-defined filter. <n> ranges from 1 to 6.

**Return Format**
The query returns NONE, RRC, GAUS, RECT, USER1, USER2, USER3, USER4, USER5, or USER6.。

**Example**
The following command sets the measurement filter to Gaussian.
:SENSe:DDEMod:FILTer:MEASurement GAUSsian

The following query returns GAUS.
:SENSe:DDEMod:FILTer:MEASurement?

## [:SENSe]:DDEMod[:SEGMent1]:FILTer:REFerence

**Syntax**
[:SENSe]:DDEMod[:SEGMent1]:FILTer:REFerence <type>
[:SENSe]:DDEMod[:SEGMent1]:FILTer:REFerence?

**Description**
Sets the reference filter type.
Queries the reference filter type.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <type> | Discrete | RCOSine\|RRCosine\|GAUSsian\|RECTangle\|<br>HSINe\|USER2\|USER3\|USER4\|USER5\|USER6 | RCOSine |

**Remarks**
RCOSine: indicates the raised cosine filter.
RRCosine: indicates the root raised cosine.
GAUSsian: indicates the Gaussian filter.
RECTangle: indicates the rectangular filter.
HSINe: indicates the half-sine filter.
USER<n>: indicates the user-defined reference filter. <n> ranges from 1 to 6.

**Return Format**
The query returns RCOS, RRC, GAUS, RECT, HSIN, USER1, USER2, USER3, USER4, USER5, or USER6.

**Example**
The following command sets the reference filter to Gaussian.
:SENSe:DDEMod:FILTer:REFerence GAUSsian

The following query returns GAUS.
:SENSe:DDEMod:FILTer:REFerence?

## [:SENSe]:DDEMod[:SEGMent1]:FILTer:ALPHa

**Syntax**
[:SENSe]:DDEMod[:SEGMent1]:FILTer:ALPHa <real>
[:SENSe]:DDEMod[:SEGMent1]:FILTer:ALPHa?

**Description**
Sets the Alpha/BT value of the filter.
Queries the Alpha/BT value of the filter.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <real> | Real | 0.05 to 100 | —— |

**Return Format**
The query returns the Alpha/BT value of the filter in scientific notation.

**Example**
The following command sets the Alpha/BT value of the filter to 20.
:SENSe:DDEMod:FILTer:ALPHa 20

The following query returns 2.0E+01.
:SENSe:DDEMod:FILTer:ALPHa?

## [:SENSe]:DDEMod[:SEGMent1]:FSK:DEViation:REFerence

**Syntax**
[:SENSe]:DDEMod[:SEGMent1]:FSK:DEViation:REFerence <freq>
[:SENSe]:DDEMod[:SEGMent1]:FSK:DEViation:REFerence?

**Description**
Sets the reference deviation of FSK modulation.
Queries the reference deviation of FSK modulation.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <freq> | Real | 1 kHz to 1 THz | 1 kHz |

**Remarks**
This command is valid when you select FSK modulation format and set the reference deviation mode to Manual.

**Return Format**
The query returns the reference deviation in scientific notation. The unit is Hz.

**Example**
The following command sets the reference deviation to 1 MHz.
:SENSe:DDEMod:FSK:DEViation:REFerence 1000000

The following query returns 1.000000000e+06.
:SENSe:DDEMod:FSK:DEViation:REFerence?

## [:SENSe]:DDEMod[:SEGMent1]:FSK:DEViation:REFerence:AUTO

**Syntax**
[:SENSe]:DDEMod[:SEGMent1]:FSK:DEViation:REFerence:AUTO <bool>
[:SENSe]:DDEMod[:SEGMent1]:FSK:DEViation:REFerence:AUTO?

**Description**
Enables or disables the auto reference deviation mode.
Queries whether the reference deviation mode is Auto or Manual.

**Parameter**

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

**Remarks**
ON|1: sets the reference deviation mode to Auto.
OFF|0: sets the reference deviation mode to Manual.
This command is valid when you select FSK modulation format.

**Return Format**
The query returns 0 or 1.

**Example**
The following command sets the reference deviation mode of FSK modulation to Auto.
:SENSe:DDEMod:FSK:DEViation:REFerence:AUTO ON
or :SENSe:DDEMod:FSK:DEViation:REFerence:AUTO 1

The following query returns 1.
:SENSe:DDEMod:FSK:DEViation:REFerence:AUTO?

## [:SENSe]:DDEMod[:SEGMent1]:BER:STATe

**Syntax**
[:SENSe]:DDEMod[:SEGMent1]:BER:STATe <bool>
[:SENSe]:DDEMod[:SEGMent1]:BER:STATe?

**Description**
Enables or disables the display of BER.
Queries the on/off status of BER.

**Parameter**

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

**Remarks**
N/A

**Return Format**
The query returns 0 or 1.

**Example**
The following command enables the display of BER.
:SENSe:DDEMod:SEGMent1:BER:STATe 1 or :SENSe:DDEMod:SEGMent1:BER:STATe ON

The following query returns 1.
:SENSe:DDEMod:SEGMent1:BER:STATe?

## [:SENSe]:DDEMod[:SEGMent1]:BER:COUNt:BITS

**Syntax**
[:SENSe]:DDEMod[:SEGMent1]:BER:COUNt:BITS <real>
[:SENSe]:DDEMod[:SEGMent1]:BER:COUNt:BITS?

**Description**
Sets the total bits.
Queries the total bits..

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <real> | Integer | 0 to 40960 | 200 |

**Remarks**
N/A

**Return Format**
The query returns the total bits in scientific notation.

**Example**
The following command sets the total bits to 100.
:SENSe:DDEMod:SEGMent1:BER:COUNt:BITS 100

The following query returns 100.
:SENSe:DDEMod:SEGMent1:BER:COUNt:BITS?

## [:SENSe]:DDEMod:SYNC:SLENgth

**Syntax**
[:SENSe]:DDEMod:SYNC:SLENgth <time>
[:SENSe]:DDEMod:SYNC:SLENgth?

**Description**
Sets the burst search length.
Queries the burst search length.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <time> | Real | 50 us to 16.3840 ms | 50 us |

**Return Format**
The query returns the burst search length in scientific notation.

**Example**
The following command sets the burst search length to 200 us.
:SENSe:DDEMod:SYNC:SLENgth 0.0002

The following query returns 2.000000000e-04.
:SENSe:DDEMod:SYNC:SLENgth?

## [:SENSe]:DDEMod:SYNC:SLENgth:AUTO

**Syntax**
[:SENSe]:DDEMod:SYNC:SLENgth:AUTO <bool>
[:SENSe]:DDEMod:SYNC:SLENgth:AUTO?

**Description**
Enables or disables the auto setting of burst search length.
Queries the on/off status of auto setting of burst search length.

**Parameter**

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

**Return Format**
The query returns 0 or 1.

**Example**
The following command enables or disables the auto setting of burst search length.
:SENSe:DDEMod:SYNC:SLENgth:AUTO 1 or :SENSe:DDEMod:SYNC:SLENgth:AUTO ON
The following query returns 1.
:SENSe:DDEMod:SYNC:SLENgth:AUTO?

## [:SENSe]:DDEMod:SYNC:ALENgth

**Syntax**
[:SENSe]:DDEMod:SYNC:ALENgth <interger>
[:SENSe]:DDEMod:SYNC:ALENgth?

**Description**
Sets the sync analysis length.
Queries the sync analysis length.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <interger> | Integer | 50 to 4,096 | 50 |

**Remarks**
This command is only valid when you enable the sync search function and disable the burst search function.

**Return Format**
The query returns the sync analysis length in integer.

**Example**
The following command sets the sync analysis length to 50.
:SENSe:DDEMod:SYNC:ALENgth 50

The following query returns 50.
:SENSe:DDEMod:SYNC:ALENgth?

## [:SENSe]:DDEMod:SYNC:BURSt:RUNIn

**Syntax**
[:SENSe]:DDEMod:SYNC:BURSt:RUNIn <interger>
[:SENSe]:DDEMod:SYNC:BURSt:RUNIn?

**Description**
Sets the burst search run-in.
Queries the burst search run-in.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <integer> | Integer | 0 to 16384 | 0 |

**Remarks**
This command is only valid when you enable the burst search function.

**Return Format**
The query returns the burst search run-in in integer.

**Example**
The following command sets the burst search run-in to 50.
:SENSe:DDEMod:SYNC:BURSt:RUNIn 50

The following query command returns 50.
:SENSe:DDEMod:SYNC:BURSt:RUNIn?

## [:SENSe]:DDEMod:SYNC:BURSt:[STATe]

**Syntax**
[:SENSe]:DDEMod:SYNC:BURSt:[STATe] <bool>
[:SENSe]:DDEMod:SYNC:BURSt:[STATe]?

**Description**
Enables or disables the burst search.
Queries the on/off status of the burst search.

**Parameter**

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

**Return Format**
The query returns 0 or 1.

**Example**
The following command enables the burst search.
:SENSe:DDEMod:SYNC:BURSt:STATe ON or :SENSe:DDEMod:SYNC:BURSt:STATe 1

The following query returns 1.
:SENSe:DDEMod:SYNC:BURSt:STATe?

## [:SENSe]:DDEMod:SYNC:SWORd:OFFSet

**Syntax**
[:SENSe]:DDEMod:SYNC:SWORd:OFFSet <integer>
[:SENSe]:DDEMod:SYNC:SWORd:OFFSet?

**Description**
Sets the sync offset of the sync search.
Queries the sync offset of the sync search.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <integer> | Real | -2048 to 2048 | 0 |

**Return Format**
The query returns the sync offset in integer.

**Example**
The following command sets the sync offset to -3.
:SENSe:DDEMod:SYNC:SWORd:OFFSet -3

The following query returns -3.
:SENSe:DDEMod:SYNC:SWORd:OFFSet?

## [:SENSe]:DDEMod:SYNC:SWORd:PATTern

**Syntax**
[:SENSe]:DDEMod:SYNC:SWORd:PATTern <string>
[:SENSe]:DDEMod:SYNC:SWORd:PATTern?

**Description**
Sets the sync pattern.
Queries the sync pattern.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <string> | Real | 0 to 32 symbols | —— |

**Return Format**
The query returns the sync pattern in strings.

**Example**
The following command sets the sync pattern to 1011010.
:SENSe:DDEMod:SYNC:SWORd:PATTern 1011010

The following query returns 1011010.
:SENSe:DDEMod:SYNC:SWORd:PATTern?

## [:SENSe]:DDEMod:SYNC:SWORd:[STATe]

**Syntax**
[:SENSe]:DDEMod:SYNC:SWORd:[STATe] <bool>
[:SENSe]:DDEMod:SYNC:SWORd:[STATe]?

**Description**
Enables or disables the sync word search.
Queries the setting status of the sync word search.

**Parameter**

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

**Return Format**
The query returns 0 or 1.

**Example**
The following command enables the sync word search.
:SENSe:DDEMod:SYNC:SWORd:STATe ON or :SENSe:DDEMod:SYNC:SWORd:STATe 1

The following query returns 1.
:SENSe:DDEMod:SYNC:SWORd:STATe?

## [:SENSe]:DDEMod:LOAD:KNOWndata

**Syntax**
[:SENSe]:DDEMod:LOAD:KNOWndata <path>

**Description**
Loads the known data.

**Parameter**

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

**Example**
The following command sets to load the known data from the specified path.
:SENSe:DDEMod:LOAD:KNOWndata /data/UserData/BER_KnownData-QPSK.xml

## [:SENSe]:RADio:STANdard:PRESet

**Syntax**
[:SENSe]:RADio:STANdard:PRESet <type>
[:SENSe]:RADio:STANdard:PRESet?

**Description**
Sets the preset standard of the digital demodulation.
Queries the preset standard of the digital demodulation.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <type> | Discrete | NONE\|GSM\|NADC\|WCDMA\|PDC\|PHP\|BLUETOOTH\|WLAN802\|ZIGBEE868M\|ZIGBEE915M\|ZIGBEE2450M\|TETRA\|DECT\|APCO25 | NONE |

**Return Format**
The query returns NONE, GSM, NADC, WCDMA, PDC, PHP, BLUETOOTH, WLAN802, ZIGBEE868M, ZIGBEE915M, ZIGBEE2450M, TETRA, DECT, or APCO25.

**Example**
The following command sets the preset standard of the digital demodulation to GSM.
:SENSe:RADio:STANdard:PRESet GSM

The following query returns GSM.
:SENSe:RADio:STANdard:PRESet?

## [:SENSe]:FREQuency:CENTer

**Syntax**
[:SENSe]:FREQuency:CENTer <freq>
[:SENSe]:FREQuency:CENTer?

**Description**
Sets the center frequency.
Queries the center frequency.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <freq> | Real | (Smin[2]/2) to (Fmax - Smin/2) | Fmax[1]/2 |

Note[1]: Fmax is determined by the instrument model. RSA6000 includes 8.5 GHz model, 14 GHz model, and 26.5 GHz model.
Note[2]: Smin indicates the minimum span in non-zero mode.

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

**Example**
The following command sets the center frequency to 1 MHz.

:SENSe:FREQuency:CENTer 1000000

The following query returns 1.000000000e+06.
:SENSe:FREQuency:CENTer?

## [:SENSe]:FREQuency:CENTer:STEP:AUTO

**Syntax**
[:SENSe]:FREQuency:CENTer:STEP:AUTO <bool>
[:SENSe]:FREQuency:CENTer:STEP:AUTO?

**Description**
Enables or disables the auto setting mode of the CF step.
Queries the status of the auto setting mode of the CF step.

**Parameter**

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

**Return Format**
The query returns 0 or 1.

**Example**
The following command enables the auto setting mode of the CF step.
:SENSe:FREQuency:CENTer:STEP:AUTO ON or :SENSe:FREQuency:CENTer:STEP:AUTO 1

The following query returns 1.
:SENSe:FREQuency:CENTer:STEP:AUTO?

## [:SENSe]:FREQuency:CENTer:STEP[:INCRement]

**Syntax**
[:SENSe]:FREQuency:CENTer:STEP[:INCRement] <freq>
[:SENSe]:FREQuency:CENTer:STEP[:INCRement]?

**Description**
Sets the CF step.
Queries the CF step.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <freq> | Real | -Fmax to Fmax | Fmax/10 |

**Return Format**
The query returns the center frequency step in scientific notation. The unit is Hz.

**Example**
The following command sets the CF step to 100 kHz.
:SENSe:FREQuency:CENTer:STEP:INCRement 100000

The following query returns 1.000000000e+05.
:SENSe:FREQuency:CENTer:STEP:INCRement?

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

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

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

**Description**
Sets the input impedance. Its unit is Ω.
Queries the input impedance.

**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**
The following command sets the input impedance to 75 Ω.
:SENSe:CORRection:IMPedance:INPut:MAGNitude 75

The following query returns 75.
:SENSe:CORRection:IMPedance:INPut:MAGNitude?

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

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

**Description**
Sets the external gain.
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**
The following command sets the external gain value to 20 dB.
:SENSe:CORRection:SA:RF:GAIN 20

The following query returns 2.000000000e+01.
:SENSe:CORRection:SA:RF:GAIN?

# :SYSTem Commands

## :SYSTem:BEEPer

**Syntax**
:SYSTem:BEEPer <bool>
:SYSTem:BEEPer?

**Description**
Turns on or off 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**
The following command turns on the beeper.
:SYSTem:BEEPer ON or :SYSTem:BEEPer 1

The following query returns 1.
:SYSTem:BEEPer?

## :SYSTem:DATE

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

**Description**
Sets the date of the instrument.
Queries the 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 date in the format of "YYYY,MM,DD".

**Example**
The following command sets the date of the instrument to 2017/11/16.
:SYSTem:DATE 2017,11,16

The following query returns 2017,11,16
:SYSTem:DATE?

## :SYSTem:TIME

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

**Description**
Sets the system time of the instrument.
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**
The following command sets the system time to "15:10:30".
:SYSTem:TIME 15,10,30

The following query returns 15,10,30.
:SYSTem:TIME?

## :SYSTem:STIME

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

**Description**
Sets or queries whether to display the system time.
Queries whether to display the system date and time.

**Parameter**

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

**Return Format**
The query returns 0 or 1.

**Example**
The following command enables the display of the system date.
:SYSTem:STIME 1 or :SYSTem:STIME ON

The following query returns 1.
:SYSTem:STIME?

## :SYSTem:LANGuage

**Syntax**
:SYSTem:LANGuage <enum>
:SYSTem:LANGuage?
**Description**

Sets the language of the instrument.
Queries the 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**
The following command sets the language to English.
:SYSTem:LANGuage ENGLish

The following query returns ENGL.
:SYSTem:LANGuage?

## :SYSTem:PON

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

**Description**
Selects the setting type the instrument recalls at power-on.
Queries what 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**
The following command sets the instrument to recall the last setting.
:SYSTem:PON LAT

The following query returns LAT.
:SYSTem:PON?

## :SYSTem:PSTatus

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

**Description**
Sets the power status.
Queries the power status.

**Parameter**

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

**Return Format**
DEFault: switch off.
OPEN: switch on.

**Example**
The following command sets the power switch on the front panel to be turned off.
:SYSTem:PSTatus OFF or :SYSTem:PSTatus 0

The following query returns 0.
:SYSTem:PSTatus?

## :SYSTem:PRESet

**Syntax**
:SYSTem:PRESet

**Description**
Resets the system to power on.

## :SYSTem:PWRD

**Syntax**
:SYSTem:PWRD

**Description**
Sets the system to power off.

## :SYSTem:VERSion?

**Syntax**
:SYSTem:VERSion?

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

## :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**
The following command locks the keypad.
:SYSTem:LOCKed 1 or :SYSTem:LOCKed ON

The following query returns 1.
:SYSTem:LOCKed?

## :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 |

**Remarks**
N/A

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

**Example**
The following command sets the GPIB address to "2".
:SYSTem:GPIB 2

The following query returns 2.
:SYSTem:GPIB?

## :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**
The following command installs the option RSA6000-PA.
:SYSTem:LKEY RSA6000-PA@8AD12B8EBC5DF492D1D4289B7CBA5B6150BF6F5D752D645C36D74530B05F39B49C461B23A50D6C94A34E06782AC4380070B0D1A86BA84E02768391FFD70C2103

## :SYSTem:OPTion:STATus?

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

**Description**
Queries whether an option is activated or not.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <option name> | Discrete | BND\|PA\|P08\|P14\|P26\|B200\|RB200\|EMI\|T08\|ADM\|AWK\|VSA\|UBW | —— |

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

**Example**
The following command queries whether the option RSA6000-PA is activated.
:SYSTem:OPTion:STATus? RSA6000-PA

## :SYSTem:OPTion:UNINstall

**Syntax**
:SYSTem:OPTion:UNINstall

**Description**
Uninstalls all the official options.

## :SYSTem:OPTion?

**Syntax**
:SYSTem:OPTion?

**Description**
Queries the option.

# :TRIGger Commands

## :TRIGger[:SEQuence]:SOURce

**Syntax**
:TRIGger[:SEQuence]:SOURce <type>
:TRIGger[:SEQuence]:SOURce?

**Description**
Sets the trigger source.
Queries the trigger source.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <type> | Discrete | IMMediate\|EXTernal1\|POWer | IMMediate |

**Remarks**
IMMediate: indicates the free-run trigger.
EXTernal1: indicates the external trigger.
POWer: indicates the IF power trigger.

**Return Format**
The query returns IMM, EXT1, or POW.

**Example**
The following command sets the trigger source to external trigger.
:TRIGger:SEQuence:SOURce EXTernal1

The following query returns EXT1.
:TRIGger:SEQuence:SOURce?

## :TRIGger[:SEQuence]:ATRigger

**Syntax**
:TRIGger[:SEQuence]:ATRigger <time>
:TRIGger[:SEQuence]:ATRigger?

**Description**
Sets the time that the analyzer will wait for the trigger to be initiated automatically.
Queries the time that the analyzer will wait for the trigger to be initiated automatically.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <time> | Real | 1 ms to 100 s | 100 ms |

**Remarks**
This command is only valid when the auto triggering function is enabled.

**Return Format**
The query returns the time value in scientific notation. The unit is s.

**Example**
The following command sets the time to 10 ms.
:TRIGger:SEQuence:ATRigger 0.01

The following query returns 1.000000000e-02.
:TRIGger:SEQuence:ATRigger?

## :TRIGger[:SEQuence]:ATRigger:STATe

**Syntax**
:TRIGger[:SEQuence]:ATRigger:STATe <bool>
:TRIGger[:SEQuence]:ATRigger:STATe?

**Description**
Enables or disables the auto trigger function.
Queries the setting status of auto trigger function.

**Parameter**

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

**Return Format**
The query returns 1 or 0.

**Example**
The following command enables the auto trigger function.
:TRIGger:SEQuence:ATRigger:STATe ON or :TRIGger:SEQuence:ATRigger:STATe 1

The following query returns 1.
:TRIGger:SEQuence:ATRigger:STATe?

## :TRIGger[:SEQuence]:EXTernal1:DELay

**Syntax**
:TRIGger[:SEQuence]:EXTernal1:DELay <time>
:TRIGger[:SEQuence]:EXTernal1:DELay?

**Description**
Sets the delay time for the external trigger.
Queries the delay time for the external trigger.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <n> | Discrete | 1\|2 | —— |
| <time> | Real | 0 us to 500 ms | 1 us |

**Remarks**
When the parameter n is set to 1, it indicates External Trigger 1; when set to 2, it indicates External Trigger 2.
This command is only valid when the external trigger delay function is enabled.

**Return Format**
The query returns the delay time for the external trigger in scientific notation. The unit is s.

**Example**
The following command sets the delay time for External Trigger 1 to 100 ms.
:TRIGger:SEQuence:EXTernal1:DELay 0.1

The following query returns 1.000000000e-01.
:TRIGger:SEQuence:EXTernal1:DELay?

## :TRIGger[:SEQuence]:EXTernal1:DELay:STATe

**Syntax**
:TRIGger[:SEQuence]:EXTernal1:DELay:STATe <bool>
:TRIGger[:SEQuence]:EXTernal1:DELay:STATe?

**Description**
Enables or disables the external trigger delay function.
Queries the setting state of the external trigger delay function.

**Parameter**

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

**Remarks**
When the parameter n is set to 1, it indicates External Trigger 1; when set to 2, it indicates External Trigger 2.

**Return Format**
The query returns 1 or 0.

**Example**
The following command enables the delay function of External Trigger 1.
:TRIGger:SEQuence:EXTernal1:DELay:STATe ON or :TRIGger:SEQuence:EXTernal1:DELay:STATe 1

The following query returns 1.
:TRIGger:SEQuence:EXTernal1:DELay:STATe?

## :TRIGger[:SEQuence]:EXTernal1:SLOPe

**Syntax**
:TRIGger:SEQuence:EXTernal1:SLOPe <type>
:TRIGger:SEQuence:EXTernal1:SLOPe?

**Description**
Sets the trigger edge for the external trigger.
Queries the trigger edge for the external trigger.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <n> | Discrete | 1\|2 | —— |
| <type> | Discrete | POSitive\|NEGative | POSitive |

**Remarks**
When the parameter n is set to 1, it indicates External Trigger 1; when set to 2, it indicates External Trigger 2.
POSitive: indicates the rising edge.
NEGative: indicates the falling edge.

**Return Format**
The query returns POS or NEG.

**Example**
The following command sets the trigger edge of External Trigger 1 to POSitive.
:TRIGger:SEQuence:EXTernal1:SLOPe POSitive

The following query returns POS.
:TRIGger:SEQuence:EXTernal1:SLOPe?

## :TRIGger[:SEQuence]:IF:DELay

**Syntax**
:TRIGger[:SEQuence]:IF:DELay <time>
:TRIGger[:SEQuence]:IF:DELay?

**Description**
Sets the delay time for the IF power trigger.
Queries the delay time for the IF power trigger.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <time> | Real | 1 ms to 100 s | 100 ms |

**Remarks**
This command is only valid when the IF power trigger function is enabled.

**Return Format**
The query returns the time value in scientific notation. The unit is s.

**Example**
The following command sets the delay time for the IF power trigger to 10 ms.
:TRIGger:SEQuence:IF:DELay 0.01

The following query returns 1.000000000e-02.
:TRIGger:SEQuence:IF:DELay?

## :TRIGger[:SEQuence]:IF:DELay:STATe

**Syntax**
:TRIGger[:SEQuence]:IF:DELay:STATe <bool>
:TRIGger[:SEQuence]:IF:DELay:STATe?

**Description**
Sets the delay state for the IF power trigger.
Queries the delay state for the IF power trigger.

**Parameter**

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

**Return Format**
The query returns 1 or 0.

**Example**
The following command enables the delay function for the IF power trigger.
:TRIGger:SEQuence:IF:DELay:STATe ON or :TRIGger:SEQuence:IF:DELay:STATe 1

The following query returns 1.
:TRIGger:SEQuence:IF:DELay:STATe?

## :TRIGger[:SEQuence]:IF:LEVel

**Syntax**
:TRIGger[:SEQuence]:IF:LEVel <time>
:TRIGger[:SEQuence]:IF:LEVel?

**Description**
Sets the trigger level of the IF power trigger.
Queries the trigger level of the IF power trigger.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <time> | Real | (-140+Level Offset) to (30+Level Offset) | -25 dBm |

**Remarks**
This command is only valid when the IF power trigger function is enabled.

**Return Format**
The query returns the trigger level of the IF power trigger in scientific notation.

**Example**
The following command sets the trigger level of IF power trigger to 10 dBm.
:TRIGger:SEQuence:IF:LEVel 10

The following query returns 1.000000000e+01.
:TRIGger:SEQuence:IF:LEVel?

## :TRIGger[:SEQuence]:HOLDoff

**Syntax**
:TRIGger[:SEQuence]:HOLDoff <time>
:TRIGger[:SEQuence]:HOLDoff?

**Description**
Sets the trigger holdoff time.
Queries the trigger holdoff time.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <time> | Real | 100 us to 500 ms (GPSA mode)<br>0 us to 10 s (RTSA mode) | 100 ms |

**Remarks**
This command is only valid when the trigger holdoff function is enabled.

**Return Format**
The query returns the trigger holdoff time in scientific notation. The unit is s.

**Example**
The following command sets the sync holdoff time to 100 ms.
:TRIGger:SEQuence:HOLDoff 0.1

The following query returns 1.000000000e-01.
:TRIGger:SEQuence:HOLDoff?

## :TRIGger[:SEQuence]:HOLDoff:STATe

**Syntax**
:TRIGger[:SEQuence]:HOLDoff:STATe <bool>
:TRIGger[:SEQuence]:HOLDoff:STATe?

**Description**
Turns on or off the trigger holdoff function.
Queries the status of the trigger holdoff function.

**Parameter**

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

**Return Format**
The query returns 1 or 0.

**Example**
The following command enables the trigger holdoff function.
:TRIGger:SEQuence:HOLDoff:STATe ON or :TRIGger:SEQuence:HOLDoff:STATe 1

The following query returns 1.
:TRIGger:SEQuence:HOLDoff:STATe?

# :LAN Commands

The **:LAN** commands are used to set or query the LAN-related parameters.

**NOTE**

After configuring all the other :LAN commands, you need to send *:LAN:APPLy* to make all the LAN configurations take effect.

## :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 | {{1\|ON}\|{0\|OFF}} | 1\|ON |

### 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". The three IP configuration types cannot be all turned off at the same time.
- 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.

### Example

```
: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 | {{1\|ON}\|{0\|OFF}} | 1\|ON |

### Remarks

When the auto IP mode is valid, disable DHCP manually. You can self-define the gateway and DNS address for the oscilloscope.

### 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: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 <string> is nnn.nnn.nnn.nnn. The range of the first section of "nnn" is from 0 to 223 (except 127), and the ranges of the other three sections of "nnn" are from 0 to 255.
- When you use this command, the IP configuration mode should be Auto IP or Static IP mode.

**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 <string> is nnn.nnn.nnn.nnn. The range of the first section of "nnn" is from 0 to 223 (except 127), and the ranges of the other three sections of "nnn" are from 0 to 255.
- When you use this command, the IP configuration mode should be Auto IP or Static IP mode.

**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 of the instrument.

### Parameter

N/A

### Remarks

N/A

### Return Format

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

### Example

```
N/A
```

## :LAN:DSERver?

### Syntax

```
:LAN:DSERver?
```

### Description

Queries the address of the DHCP server.

### Parameter

N/A

### Remarks

N/A

### Return Format

The query returns the address of the DHCP server in strings.

### Example

```
N/A
```

## :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 | {{1\|ON}\|{0\|OFF}} | 0\|OFF |

**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 <string> is nnn.nnn.nnn.nnn. The range of the first section of "nnn" is from 0 to 223 (except 127), and the ranges of the other three sections of "nnn" are from 0 to 255.

- When you use the command, the IP configuration mode should be static IP. Besides, the DHCP and auto IP should be disabled.

**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 <string> is nnn.nnn.nnn.nnn. The range of the section "nnn" is from 0 to 255.
- When you use the command, the IP configuration mode should be static IP. Besides, the DHCP and auto IP should be disabled.

**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:STATus?

### Syntax

**:LAN:STATus?**

### Description

Queries the current network configuration status.

### Parameter

N/A

### Remarks

- **UNLINK:** not connected.
- **CONNECTED:** the network is successfully connected.
- **INIT:** the instrument is acquiring an IP address.
- **IPCONFLICT:** there is an IP address conflict.
- **BUSY:** please wait...
- **CONFIGURED:** the network configuration has been successfully configured.
- **DHCPFAILED:** the DHCP configuration has failed.
- **INVALIDIP:** invalid IP.
- **IPLOSE:** IP lost.

### Return Format

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

### Example

```
N/A
```

## :LAN:VISA?

### Syntax

**:LAN:VISA?** [<*type*>]

### Description

Queries the VISA address of the instrument.

**Parameter**

| Name | Type | Range | Default |
|---|---|---|---|
| <type> | Discrete | {USB\|LXI\|SOCKet} | - |

**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.

**Example**

```
N/A
```

## :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 | {{1\|ON}\|{0\|OFF}} | 0\|OFF |

**Remarks**

N/A

**Return Format**

The query returns 1 or 0.

**Example**

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

## :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. | - |

**Remarks**

N/A

**Return Format**

The query returns the host name in ASCII strings.

**Example**

```
N/A
```

## :LAN:DESCription

**Syntax**

**:LAN:DESCription** <*name*>

**:LAN:DESCription?**

**Description**

Sets or queries the description.

**Parameter**

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

**Remarks**

N/A

**Return Format**

The query returns the description in ASCII strings.

**Example**

```
N/A
```

# :LAN:APPLy

**Syntax**

**:LAN:APPLy**

**Description**

Applies the network configuration.

**Parameter**

N/A

**Remarks**

After configuring all the LAN-related parameters with the :LAN commands, you need to send this command to make all the LAN configurations take effect.

**Return Format**

N/A

**Example**

```
N/A
```

# 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 (e.g. USB). VISA defines a set of software commands with which users can control the instrument without the need to understand how the interface bus works. For details, please refer to the NI-VISA Help.

# 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 5555.

# 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.

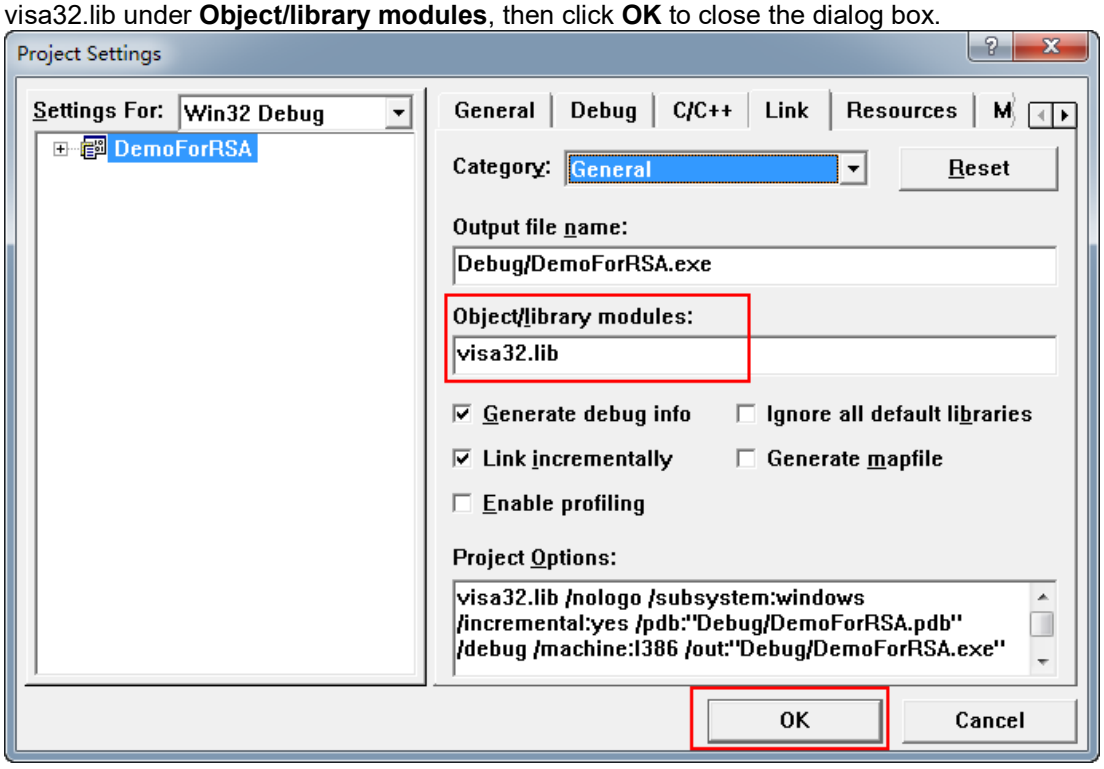
By now, the programming preparations are complete. The following parts will make a detailed introduction about the programming instances in the Visual C++ 6.0, Visual Basic 6.0, and LabVIEW 2010 development environment.

# Visual C++ 6.0 Programming Example

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

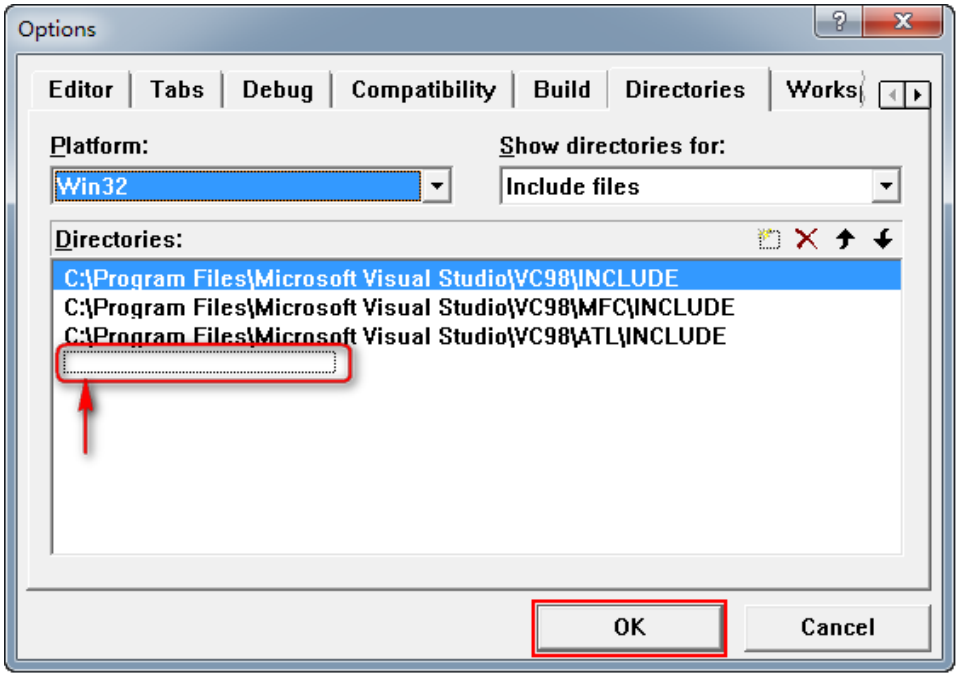
1. Create a MFC project based on a dialog box and name it "DemoForDSA" in this example.

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.



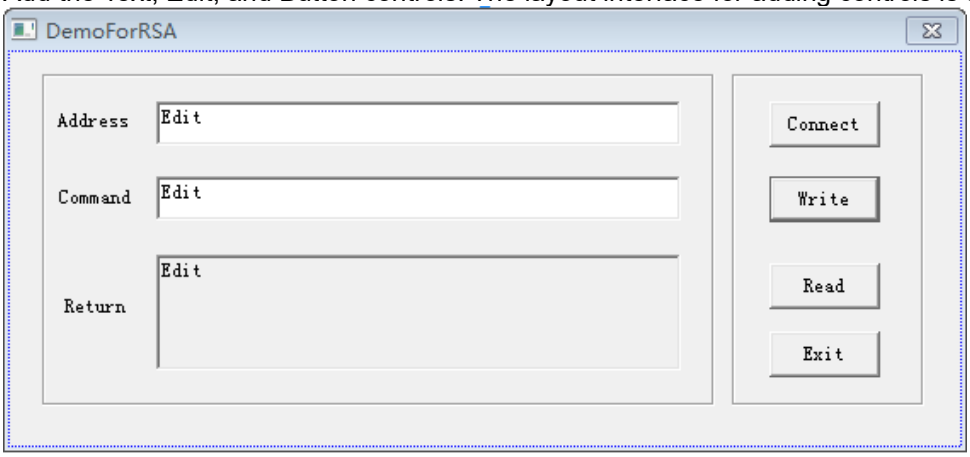

3. Click **Tools** → **Options** to open the **Options** dialog box. Then, click the **Directories** tab.
   Select **Include files** from the drop-down list under **Show directories for**. Double click the empty space under **Directories** to enter the specified path of **Include files**: C:\Program 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);
    }
```

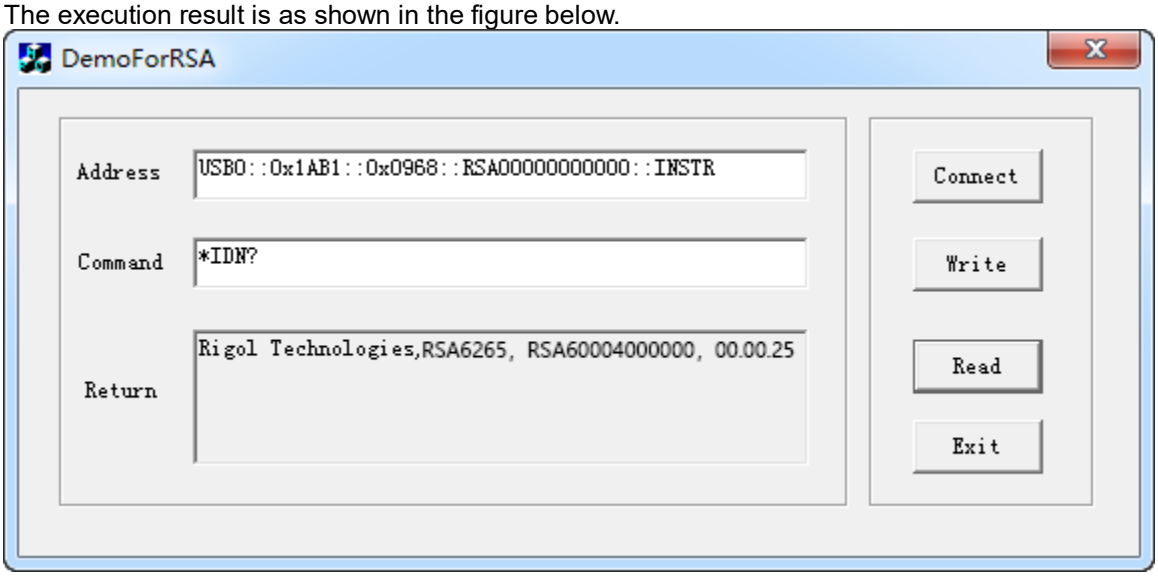
3) Read Operation

```
    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.

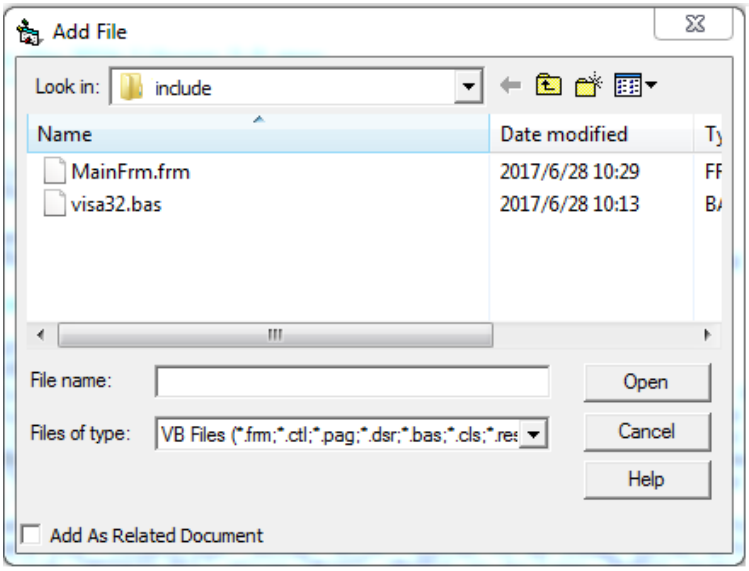


# Visual Basic 6.0 Programming Example

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

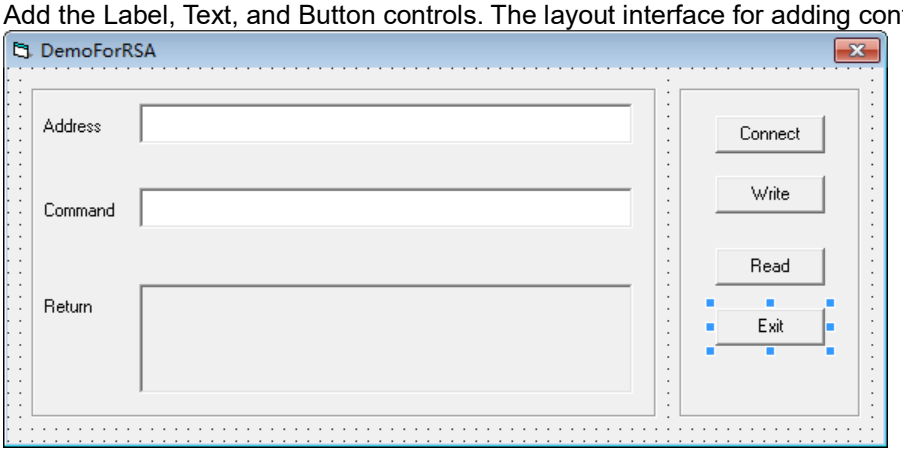
1. Build a standard application program project (Standard EXE), and name it "DemoForRSA".

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
```

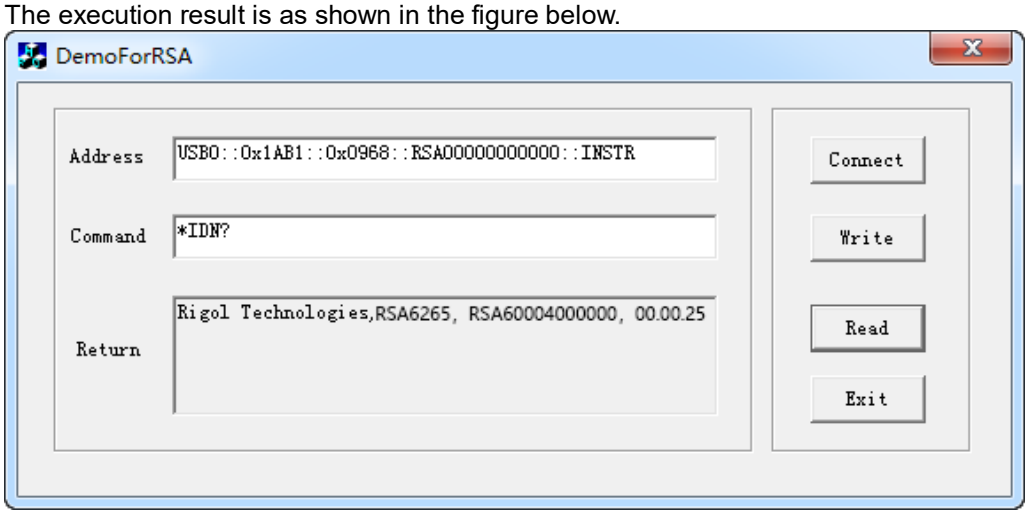
3) Read Operation

```
    '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.

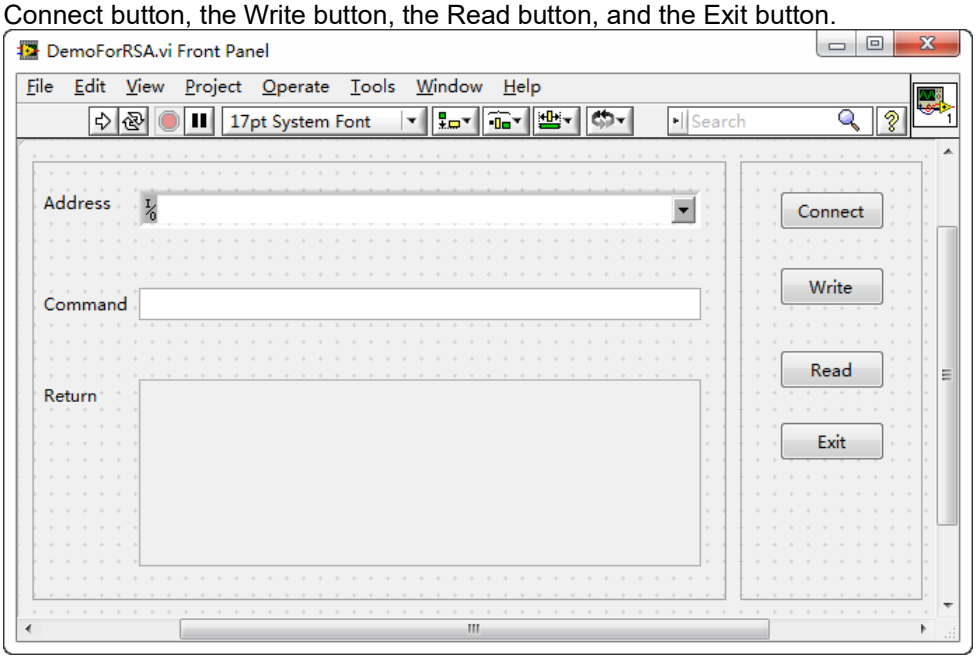


# LabVIEW 2010 Programming Example

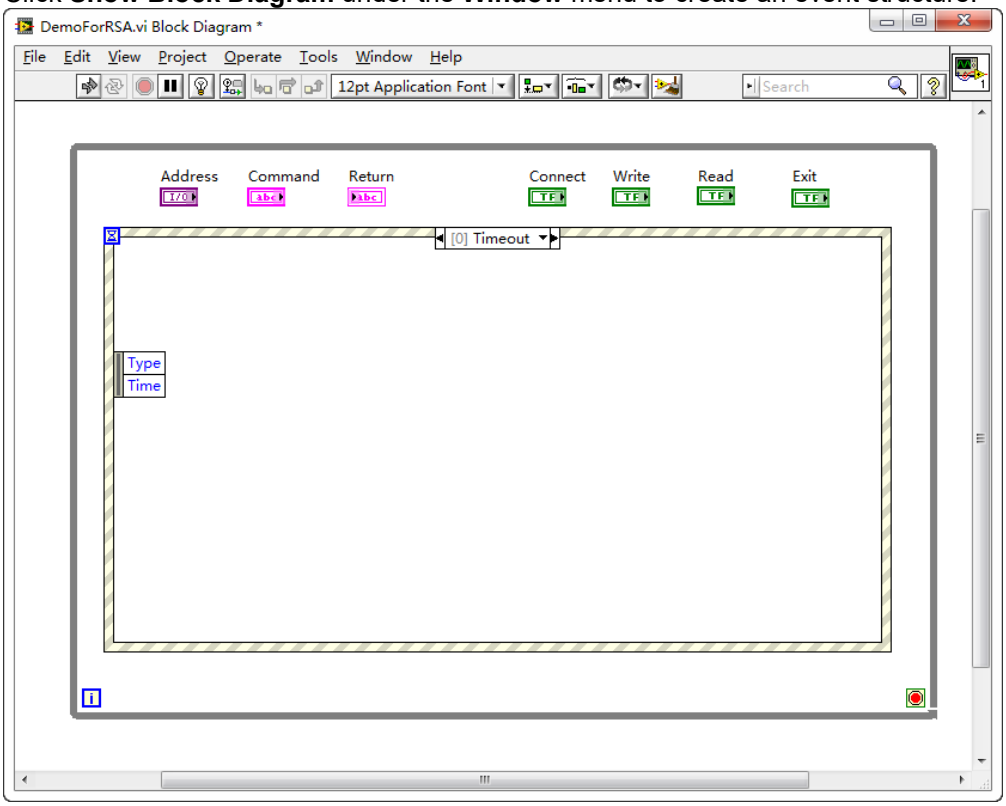
Enter the Labview 2010 programming environment, and perform the following procedures.

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

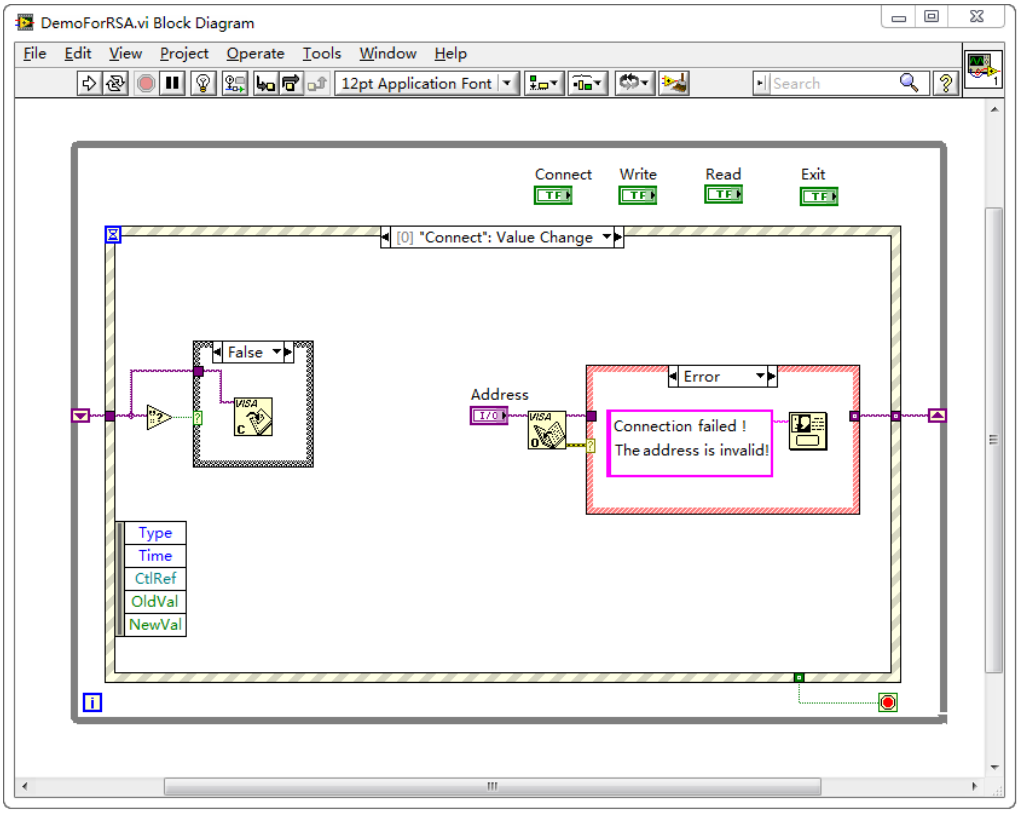
2. Add controls to the front panel interface, including the Address field, Command field, and Return field, the Connect button, the Write button, the Read button, and the Exit button.



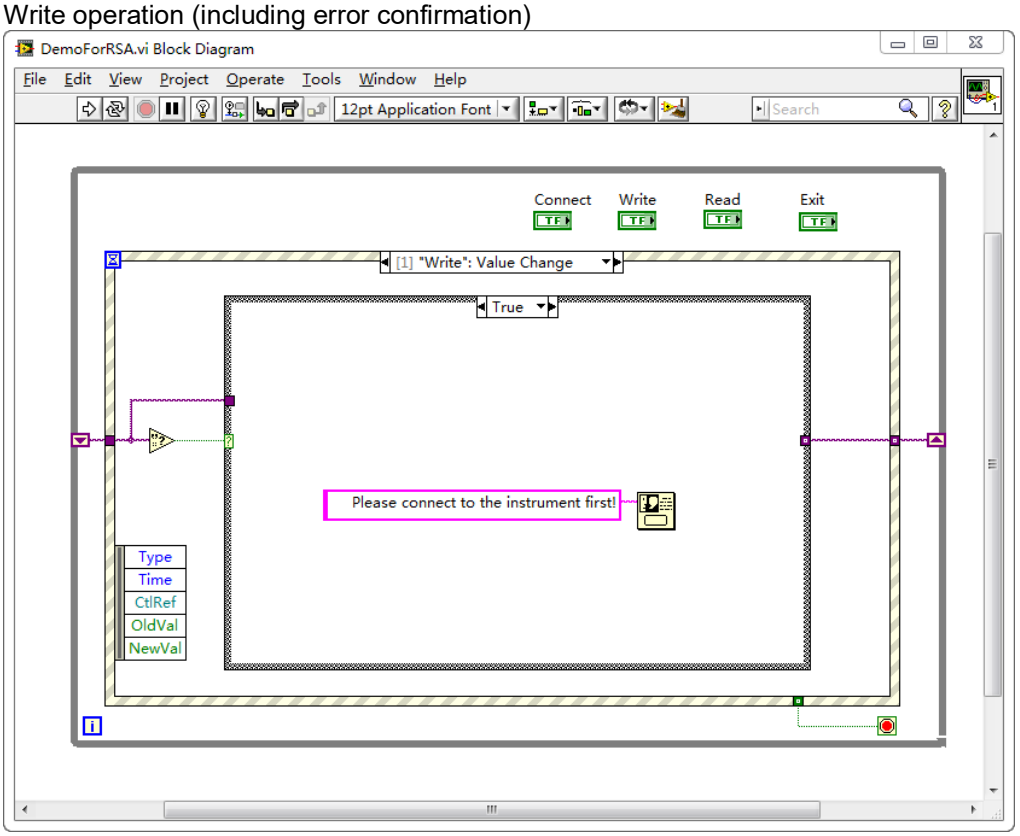

3. Click **Show Block Diagram** under the **Window** menu to create an event structure.

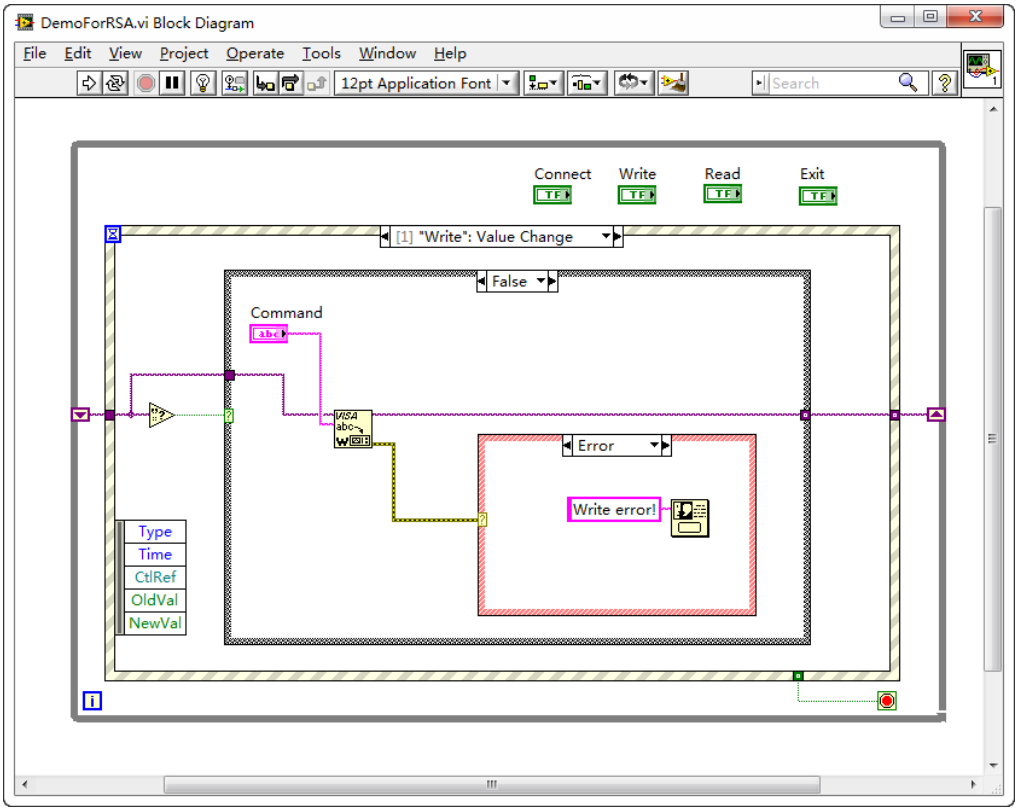



4. Add the events (including connecting to the instrument, write operation, read operation, and exit)
   1) Connect to the instrument

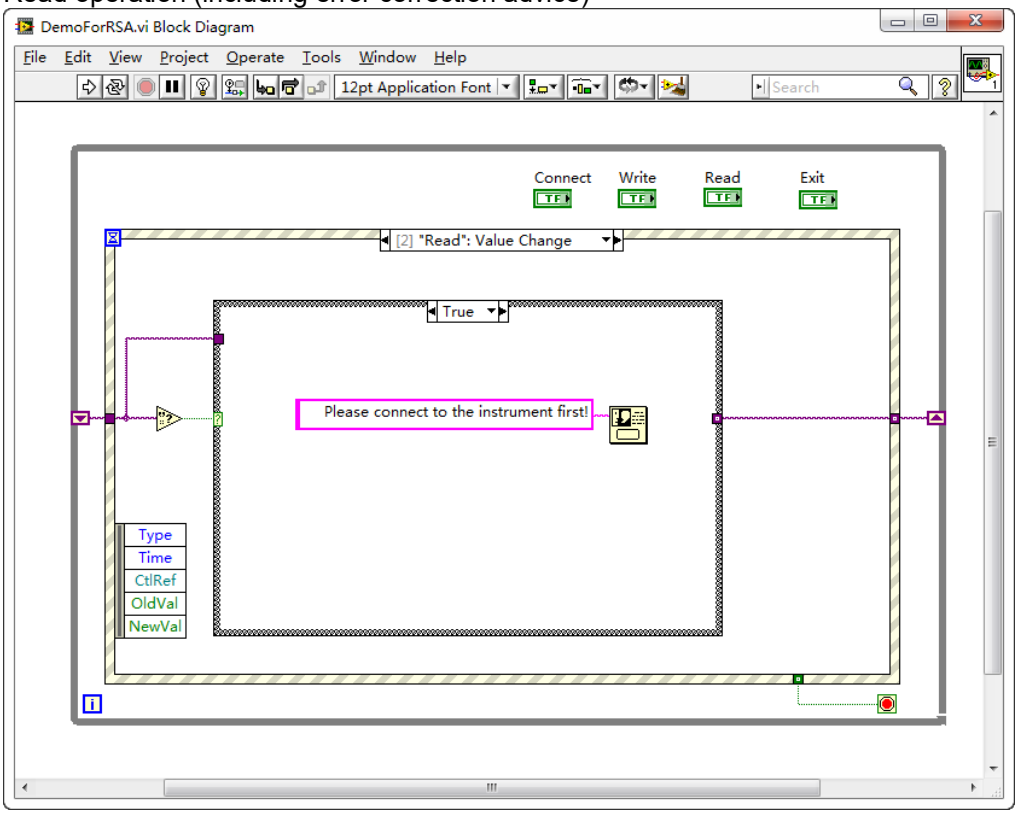


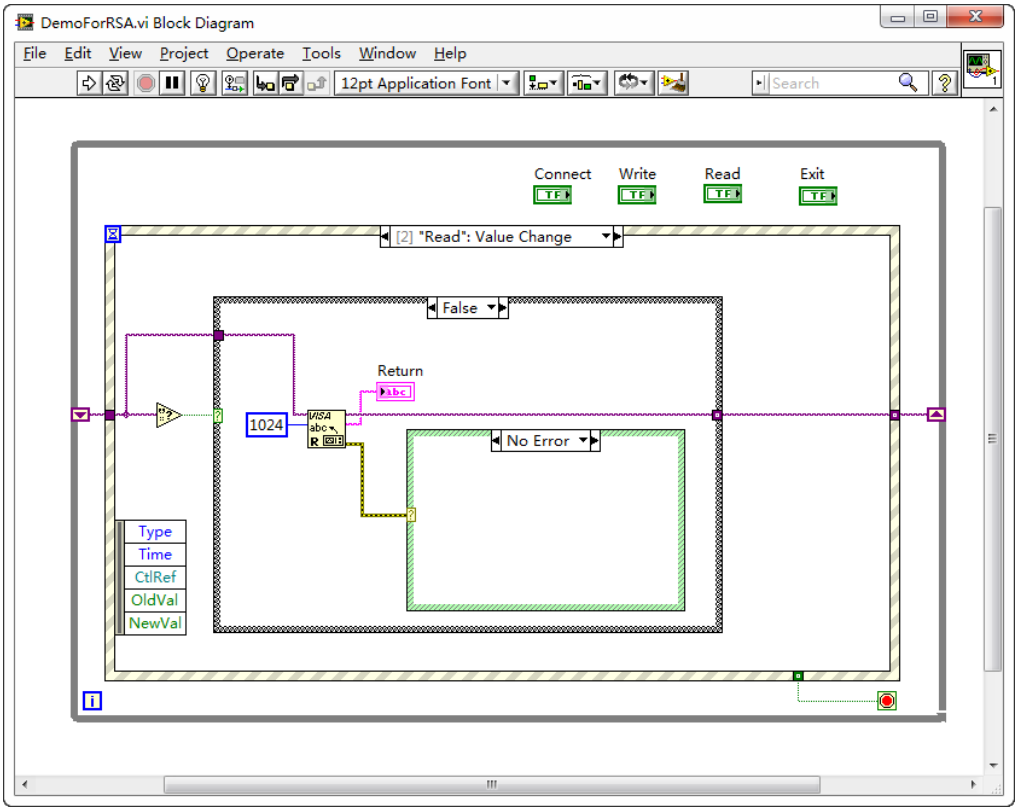

2) Write operation (including error confirmation)



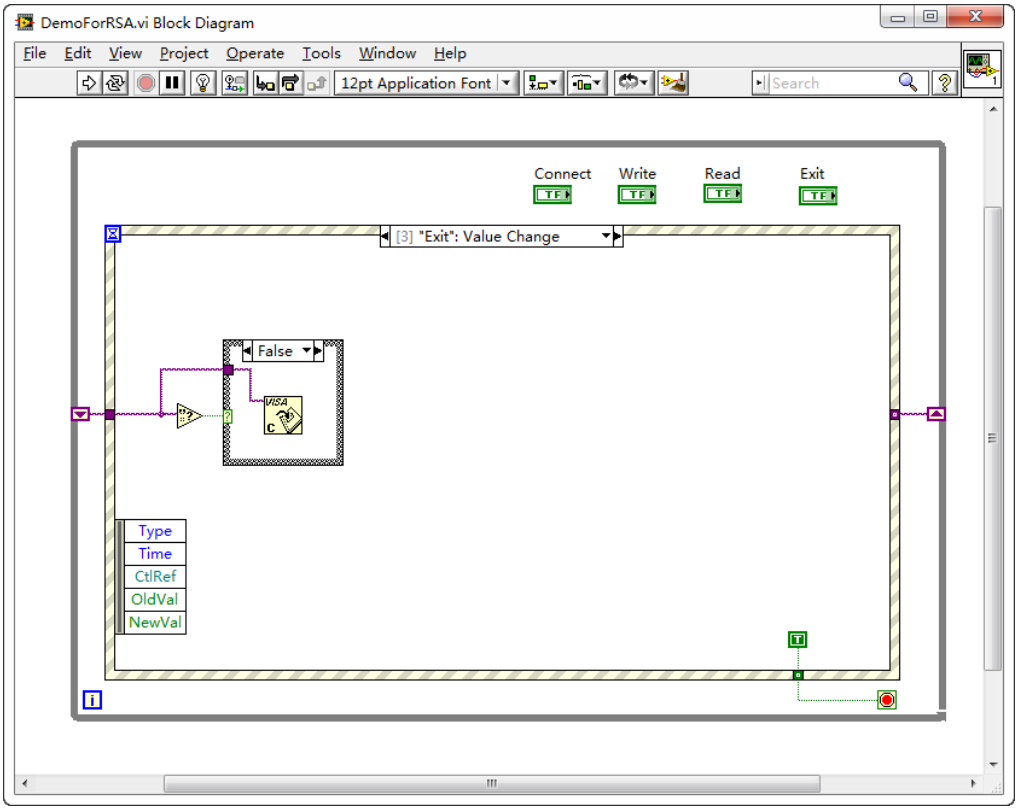



3) Read operation (including error correction advice)

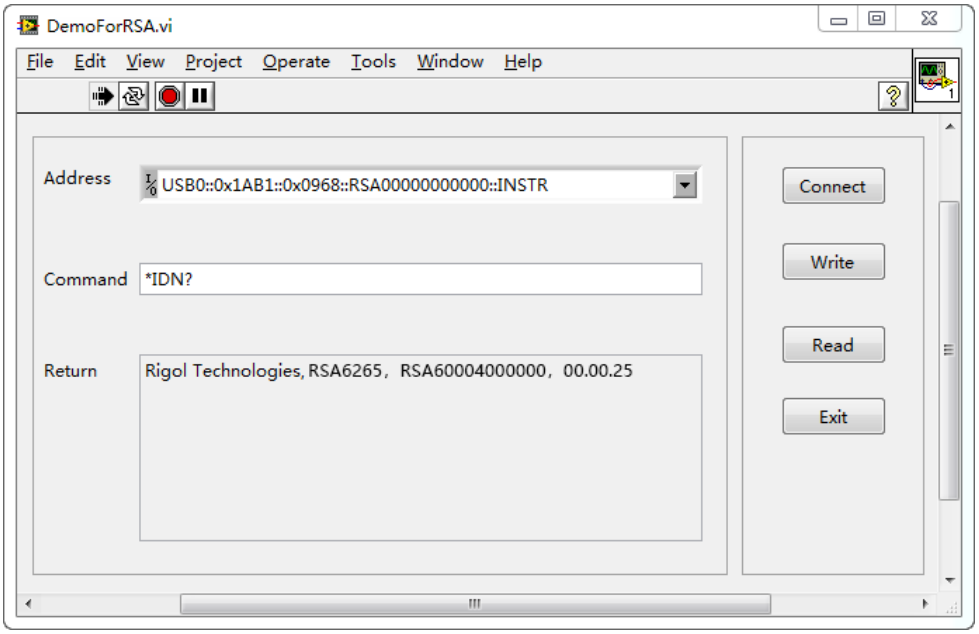





4) Exit




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

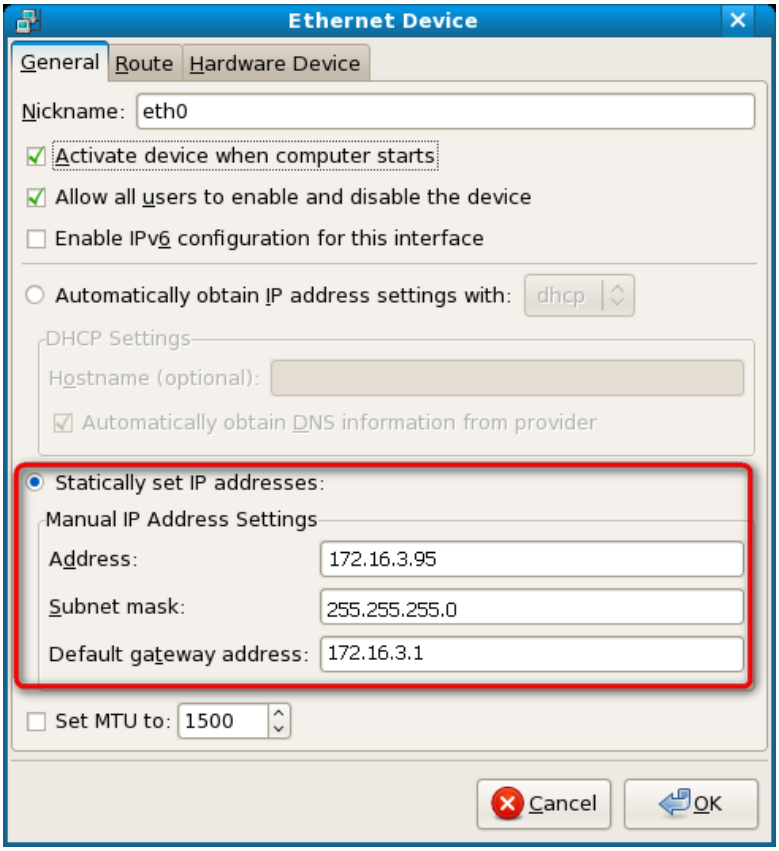
1. Programming environment:
   Operating system: Fedroa 8 (Linux-2.6.23)
   GCC version: gcc-4.1.2

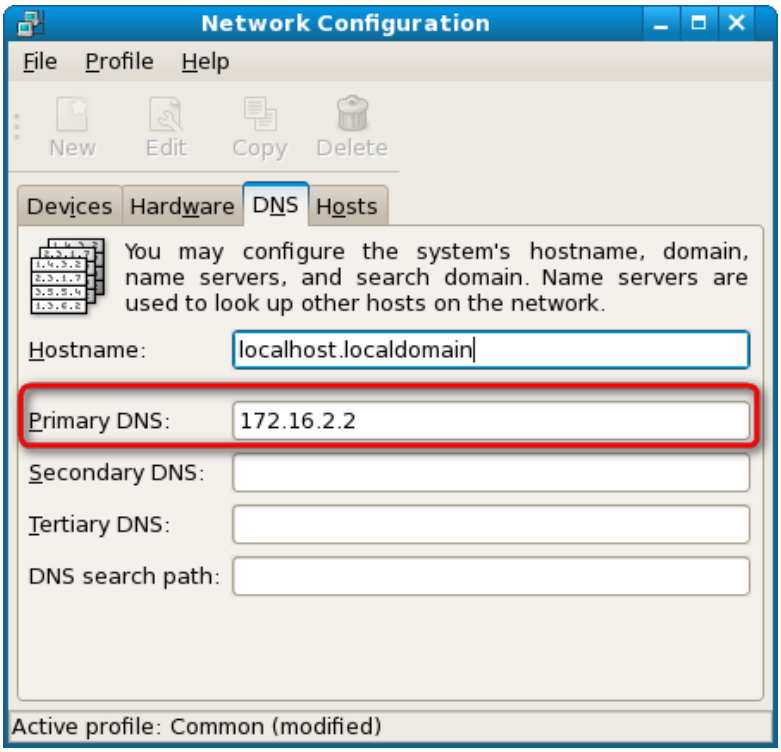
2. Install the VISA library. First, check whether your PC has installed NI's VISA library. If not, download it from NI website (*http://www.ni.com/visa/*). The installation procedures are as follows:
   Download the VISA library NI-VISA-4.4.0.ISO from the NI website.

   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.

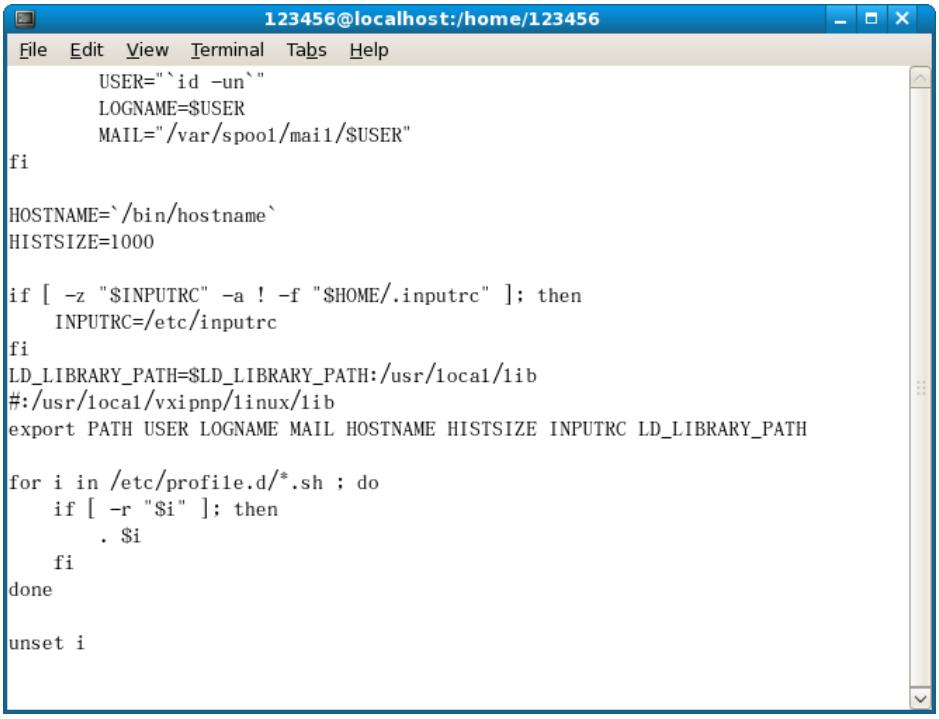
   After the spectrum analyzer is connected to the PC properly, configure the network address for the spectrum analyzer to make its address to be within the same network segment where the PC resides. For example, if the network address and DNS setting configured for the PC are as shown in the figures below, 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 ranging from 2 to 254.



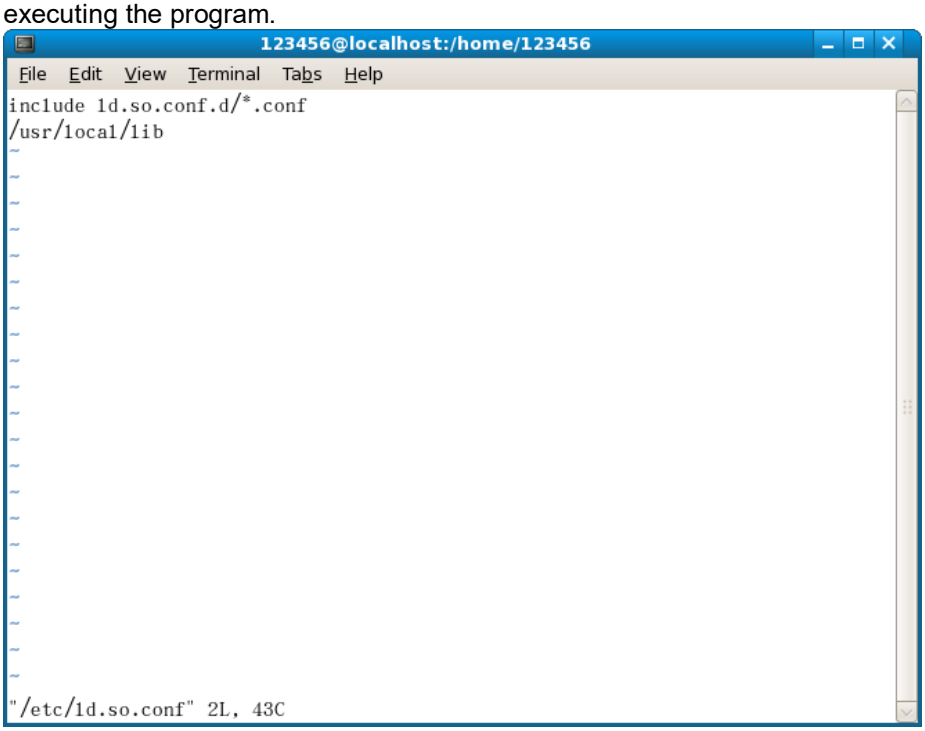




4. Use either of the following two methods to add the library location to the search path of the library, so that the program can load the installed library file automatically.

   **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

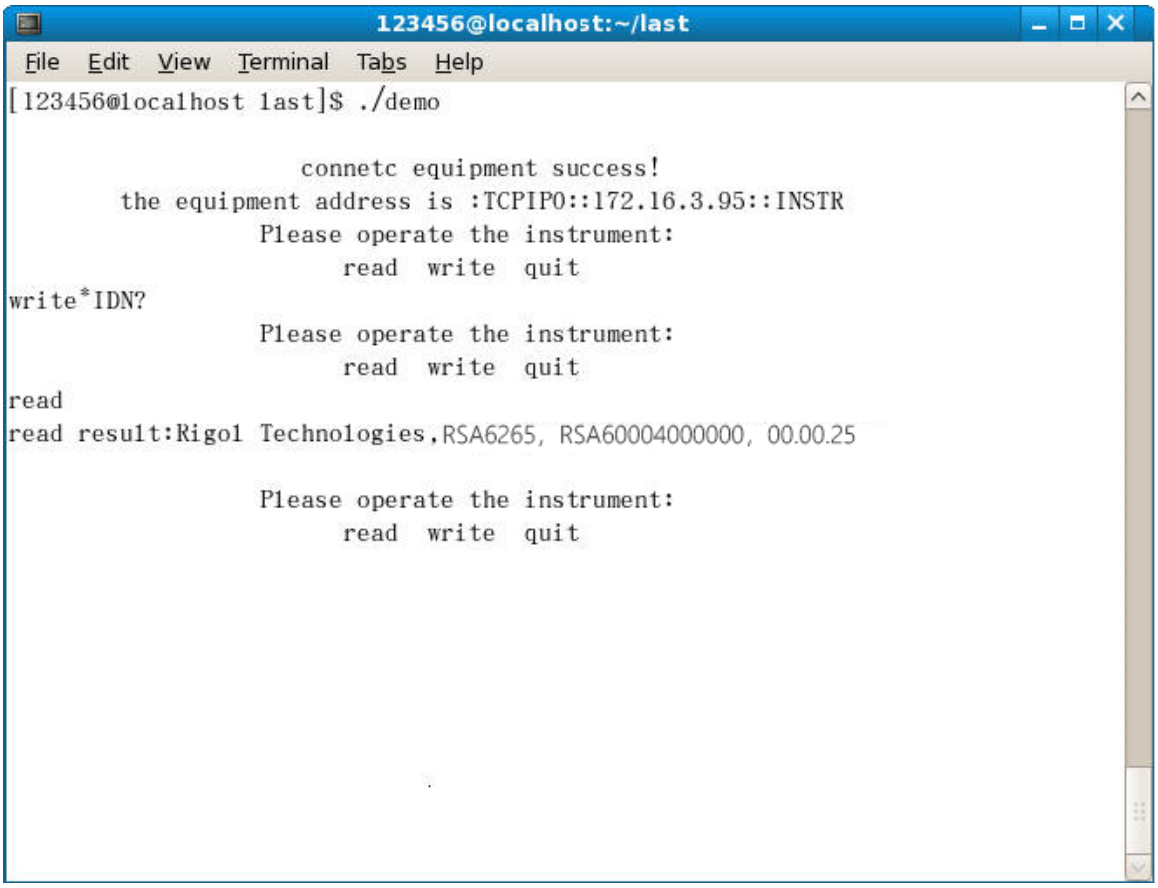
```
src = DemoForRSA.cpp mainloop.cpp DemoForRSA.h

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.



# 5 Appendix

## 5.1 Appendix A: Options and Accessories

| | Description | Order No. |
|---|---|---|
| Model | Real-time Spectrum Analyzer, 5 kHz to 8.5 GHz | RSA6085 |
| | Real-time Spectrum Analyzer, 5 kHz to 14 GHz | RSA6140 |
| | Real-time Spectrum Analyzer, 5 kHz to 26.5 GHz | RSA6265 |
| Standard Accessory | Power Cord | - |
| Options | Vector Signal Analysis Application Software | RSA6000-VSA |
| | EMI Measurement Application Software | RSA6000-EMI |
| | Analog Demodulation Application Software | RSA6000-ADM |
| | Preamplifier (PA), 8.5 GHz | RSA6000-P08 |
| | Preamplifier (PA), 14 GHz | RSA6000-P14 |
| | Preamplifier (PA), 26.5 GHz | RSA6000-P26 |
| | 200 MHz Analysis Bandwidth | RSA6000-B200 |
| | 200 MHz Real-time Bandwidth | RSA6000-RB200 |
| | Advanced Measurement Kit | RSA6000-AMK |
| | 8.5 GHz Tracking Generator Output | RSA6000-T08 |
| Optional Accessories | DSA utility kit. Refer to *Note[1]* for details. | DSA Utility Kit |
| | RF adaptor kit. Refer to *Note[2]* for details. | RF Adaptor Kit |
| | Includes: 50 Ω to 75 Ω adaptor (2pcs) | RF CATV Kit |
| | Includes: 6 dB attenuator (1pcs), 10 dB attenuator (2pcs) | RF Attenuator Kit |
| | 30 dB high-power attenuator, with the max. power of 100 W | ATT03301H |
| | N(M)-N(M) RF Cable | CB-NM-NM-75-L-12G |
| | N(M)-SMA(M) RF Cable | CB-NM-SMAM-75-L-12G |
| | Near-field Probe | NFP-3 |
| | USB Cable x1 | CB-USBA-USBB-FF-150 |

**NOTE**

- For all the mainframes, accessories, and options, please contact the local office of RIGOL.
- [1]: Includes N-SMA cable, BNC-BNC cable, N-BNC adaptor, N-SMA adaptor, 75 Ω-50 Ω adaptor, 900 MHz/1.8 GHz antenna (2pcs), 2.4 GHz antenna (2pcs)
- [2]: Includes: N(F)-N(F) adaptor (1pcs), N(M)-N(M) adaptor (1pcs), N(M)-SMA(F) adaptor (2pcs), N(M)-BNC(F) adaptor (2pcs), SMA(F)-SMA(F) adaptor (1pcs), SMA(M)-SMA(M) adaptor (1pcs), BNC T type adaptor (1pcs), 50 Ω SMA load (1pcs), 50 Ω BNC impedance adaptor (1pcs)

## 5.2 Appendix B: Warranty

RIGOL TECHNOLOGIES CO., LTD. (hereinafter referred to as RIGOL) warrants that the product mainframe and product accessories will be free from defects in materials and workmanship within the warranty period. If a product proves defective within the warranty period, RIGOL guarantees free replacement or repair for the defective product.

To get repair service, please contact your nearest RIGOL sales or service office.

There is no other warranty, expressed or implied, except such as is expressly set forth herein or other applicable warranty card. There is no implied warranty of merchantability or fitness for a particular purpose. Under no circumstances shall RIGOL be liable for any consequential, indirect, ensuing, or special damages for any breach of warranty in any case.

# Boost Smart World and Technology Innovation

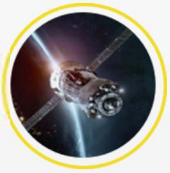
Industrial Intelligent Manufacturing

Semiconductors

Education& Research


Communication

System Integration



New Energy

- Cellular-5G/WIFI
- UWB/RFID/ ZIGBEE
- Digital Bus/Ethernet
- Optical Communication

- Digital/Analog/RF Chip
- Memory and MCU Chip
- Third-Generation Semiconductor
- Solar Photovoltaic Cells

- New Energy Automobile
- PV/Inverter
- Power Test
- Automotive Electronics

*Provide Testing and Measuring Products and Solutions for Industry Customers*


**HEADQUARTER**

**RIGOL** TECHNOLOGIES CO., LTD.
No.8 Keling Road, New District,
Suzhou,JiangSu, P.R.China
Tel: +86-400620002
Email: info-cn@rigol.com

**EUROPE**

**RIGOL** TECHNOLOGIES EU GmbH
Friedrichshafener Str. 5
82205 Gilching
Germany
Tel: +49(0)8105-27292-21
Email: info-europe@rigol.com

**NORTH AMERICA**

**RIGOL** TECHNOLOGIES, USA INC.
10220 SW Nimbus Ave.
Suite K-7
Portland, OR 97223
Tel: +1-877-4-**RIGOL**-1
Email: sales@rigol.com

**JAPAN**

**RIGOL** JAPAN CO., LTD.
5F,3-45-6,Minamiotsuka, Toshima-Ku,
Tokyo,170-0005,Japan
Tel: +81-3-6262-8932
Fax: +81-3-6262-8933
Email: info.jp@rigol.com

**KOREA**

**RIGOL** KOREA CO,. LTD.
5F, 222, Gonghang-daero,
Gangseo-gu, Seoul, Republic of Korea
Tel: +82-2-6953-4466
Fax: +82-2-6953-4422
Email: info.kr@rigol.com

**For Assistance in Other Countries**

Email: info.int@rigol.com