SSG5000X Series

RF Signal Generator

Programming Guide

EN01D

SIGLENT TECHNOLOGIES CO.,LTD

Contents

1 Programming Overview

The SSG5000X Series Signal Generator supports USB, LAN and USB-GPIB interfaces. Through these interfaces, combined with NI-VISA and corresponding programming language, users can use the command set based on SCPI (Standard Commands for Programmable Instruments) to remotely program and control the instrument, and interact with other programmable instruments that support the SCPI command set.
Through the LAN interface, VXI-11, Sockets, and Telnet protocols can be used to communicate with the instrument.
This chapter describes how to establish communication between the signal generator and the computer, and how to remotely control the signal generator.

1.1 Build communication via VISA

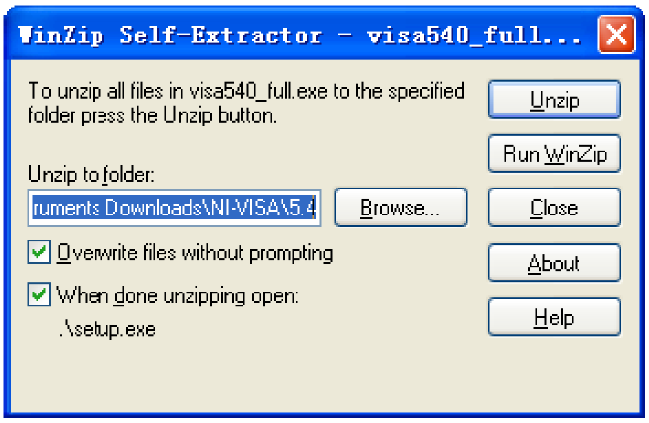
1.1.1 Install NI-VISA

Before programming, please make sure that you have properly installed the latest version of National Instruments NI-VISA Software.
NI-VISA is a communication library for communication between computers and devices. NI software has two valid VISA installation packages: full version and run-time engine version (Run-Time Engine). The runtime engine version provides NI device drivers such as USB-TMC, VXI and GPIB, etc. It is mainly used for remote control. The full version includes the runtime engine and NI MAX tools, where NI MAX is the user interface for controlling the device.

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

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

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


WinZip Self-Extractor - visa540_full...

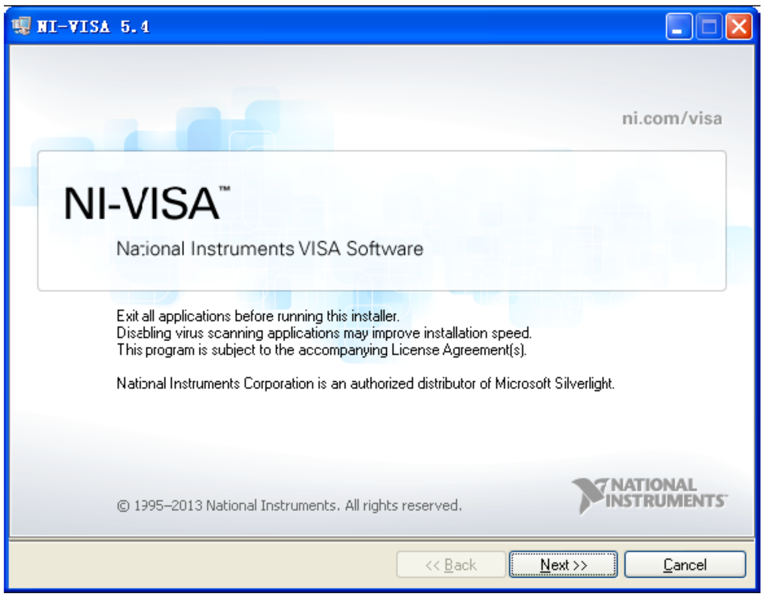
To unzip all files in visa540_full.exe to the specified folder press the Unzip button.

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

☑ Overwrite files without prompting

☑ When done unzipping open: .\setup.exe

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



NI-VISA 5.4

ni.com/visa

NI-VISA™

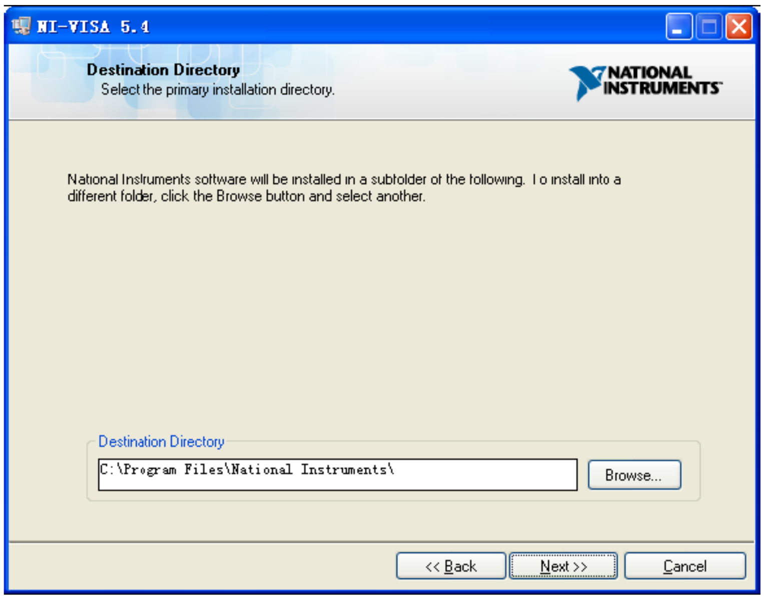
National Instruments VISA Software

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

National Instruments Corporation is an authorized distributor of Microsoft Silverlight.

© 1995–2013 National Instruments. All rights reserved.

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



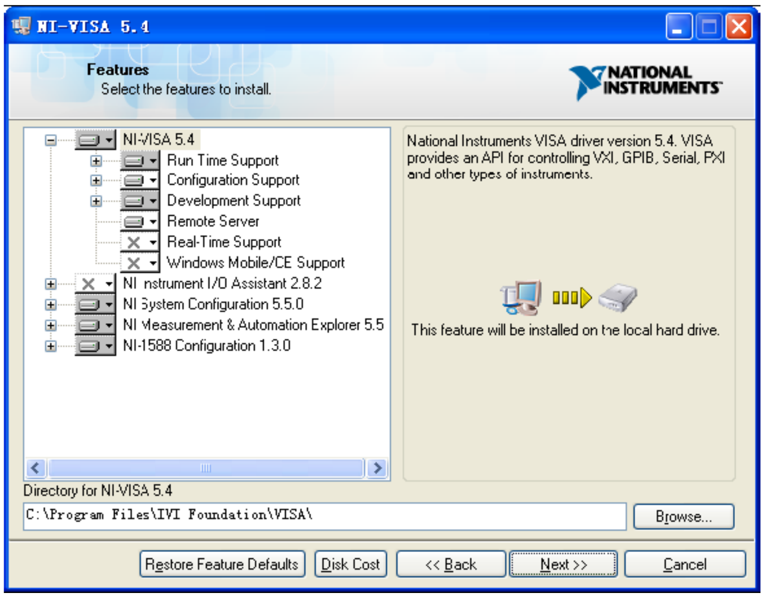
NI-VISA 5.4

Destination Directory
Select the primary installation directory.

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

Destination Directory
C:\Program Files\National Instruments\

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


NI-VISA 5.4

Features

Select the features to install.

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

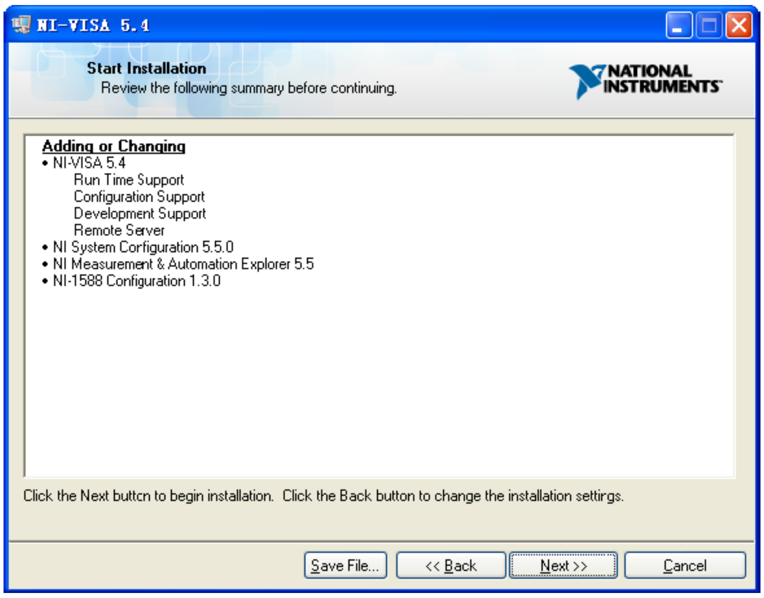
National Instruments VISA driver version 5.4. VISA provides an API for controlling VXI, GPIB, Serial, PXI and other types of instruments.

This feature will be installed on the local hard drive.

Directory for NI-VISA 5.4

C:\Program Files\IVI Foundation\VISA\

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


NI-VISA 5.4

Start Installation

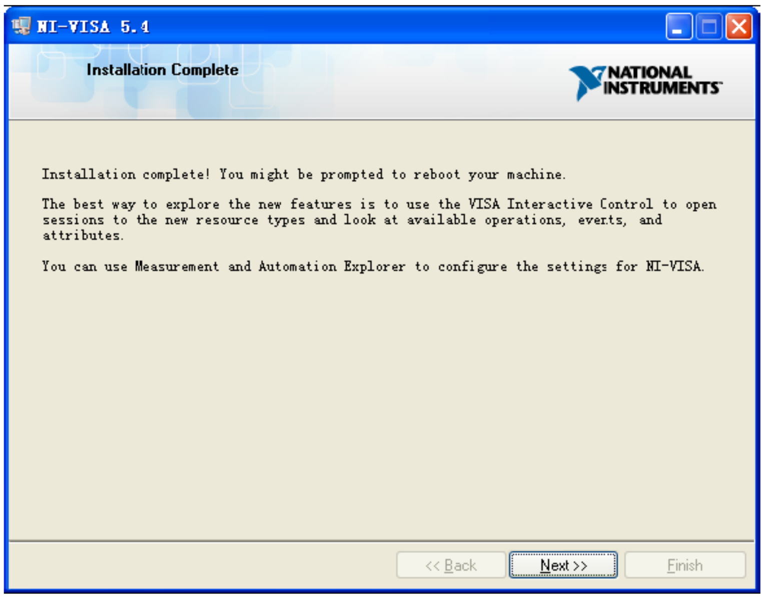
Review the following summary before continuing.

Adding or Changing

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

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

g. Click Next to begin installation.



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

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

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

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

1.1.2 Connect the instrument to computer

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

1.1.2.1 Connect using USB interface

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

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

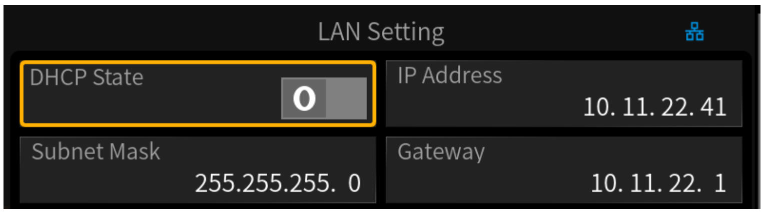
The signal generator will be automatically detected as a new USB device.

1.1.2.2 Connect using LAN interface

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

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

- Static: You can manually set the IP address, subnet mask, and gateway.


LAN Setting

DHCP State: O

IP Address: 10. 11. 22. 41

Subnet Mask: 255.255.255. 0

Gateway: 10. 11. 22. 1

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

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

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

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

3. Turn on the signal generator.
4. Press UTILITY → Interface and enter the GPIB number in GPIB Address.

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

1.2 Remote Control

1.2.1 User-defined Programming

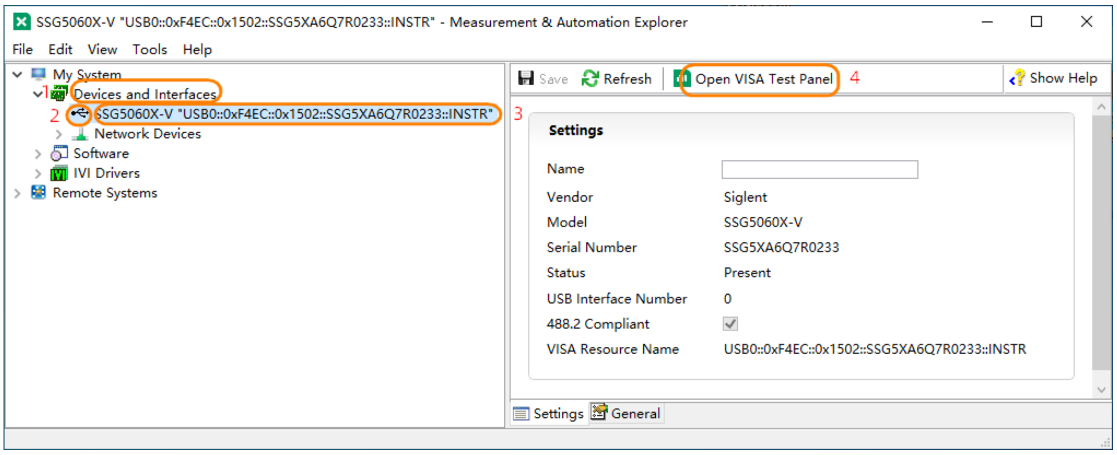
Users can send SCPI commands through the computer to program and control the signal generator. For details, please refer to the introduction in the “Programming Examples” chapter.

1.2.2 Send SCPI via NI-MAX

NI-MAX is a program created and maintained by National Instruments. It provides basic remote-control interfaces for VXI, LAN, USB, GPIB, and Serial communications. Users can send SCPI commands through NI-MAX to remotely control the signal generator.
The following describes the steps for connecting a device through a USB, LAN, or GPIB interface in NI-MAX and sending SCPI to the device.

1.2.2.1 Using USB

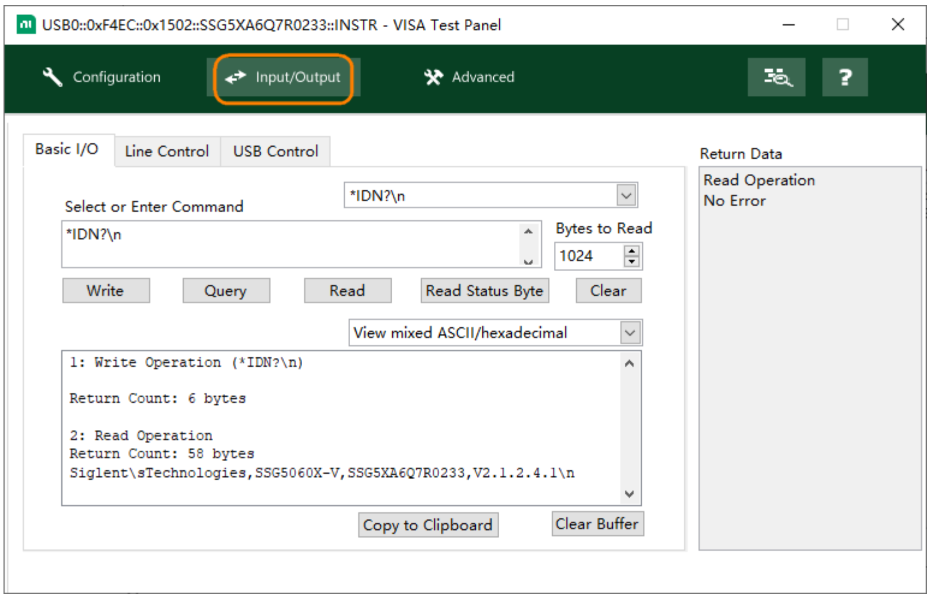
1. Run NI-MAX.
2. Click “Devices and Interfaces” in the upper left corner of the software.
3. Find the USBTMC device symbol: .
4. Select the signal generator device and click the “Open VISA Test Panel” button.



Settings

Name	
Vendor	Siglent
Model	SSG5060X-V
Serial Number	SSG5XA6Q7R0233
Status	Present
USB Interface Number	0
488.2 Compliant	✓
VISA Resource Name	USB0::0xF4EC::0x1502::SSG5XA6Q7R0233::INSTR

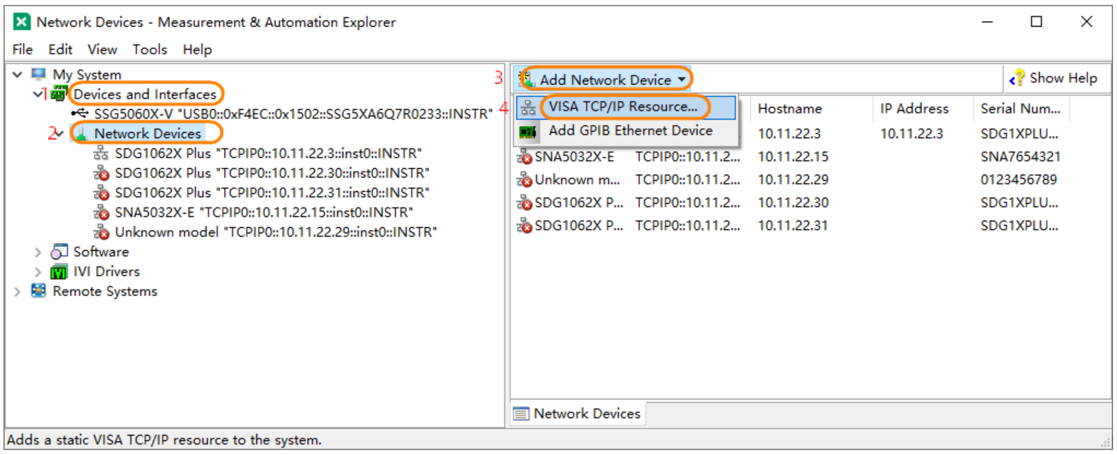
5. Select the “Input/Output” page in the VISA test panel. At this time, you can enter SCPI in the input field to write or query. Click the “Query” button as shown below to query the device’s IDN:



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

Return Count: 6 bytes

2: Read Operation
Return Count: 58 bytes
Siglent\sTechnologies,SSG5060X-V,SSG5XA6Q7R0233,V2.1.2.4.1\n
```

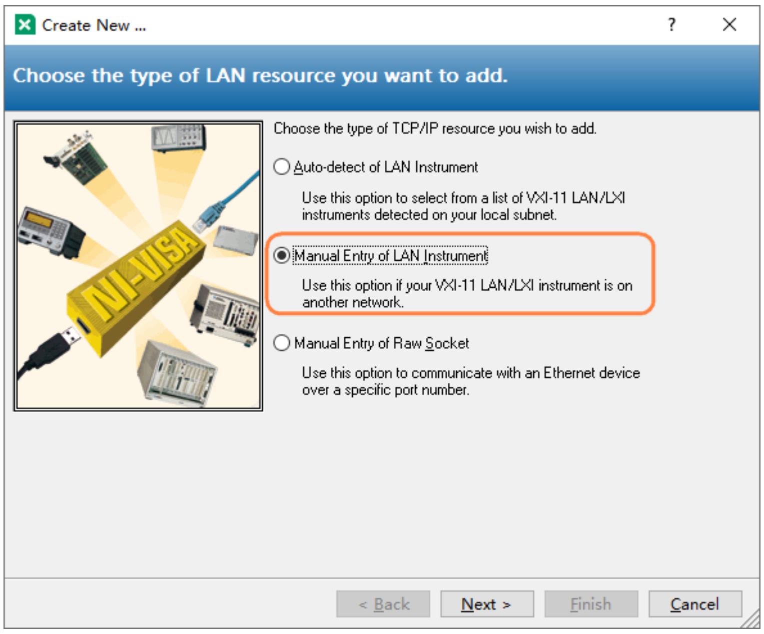
1.2.2.2 Using LAN

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


Network Devices - Measurement & Automation Explorer
File Edit View Tools Help
My System
1
Devices and Interfaces
SSG5060X-V "USB0::0xF4EC::0x1502::SSG5XA6Q7R0233::INSTR"
2
Network Devices
SDG1062X Plus "TCPIP0::10.11.22.3::inst0::INSTR"
SDG1062X Plus "TCPIP0::10.11.22.30::inst0::INSTR"
SDG1062X Plus "TCPIP0::10.11.22.31::inst0::INSTR"
SNA5032X-E "TCPIP0::10.11.22.15::inst0::INSTR"
Unknown model "TCPIP0::10.11.22.29::inst0::INSTR"
Software
IVI Drivers
Remote Systems
3
Add Network Device
Show Help
4
VISA TCP/IP Resource...
Add GPIB Ethernet Device
Network Devices
Adds a static VISA TCP/IP resource to the system.

	Hostname	IP Address	Serial Num...
	10.11.22.3	10.11.22.3	SDG1XPLU...
SNA5032X-E TCPIP0::10.11.2...	10.11.22.15		SNA7654321
Unknown m... TCPIP0::10.11.2...	10.11.22.29		0123456789
SDG1062X P... TCPIP0::10.11.2...	10.11.22.30		SDG1XPLU...
SDG1062X P... TCPIP0::10.11.2...	10.11.22.31		SDG1XPLU...

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


Create New ...

Choose the type of LAN resource you want to add.

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

( ) Auto-detect of LAN Instrument

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

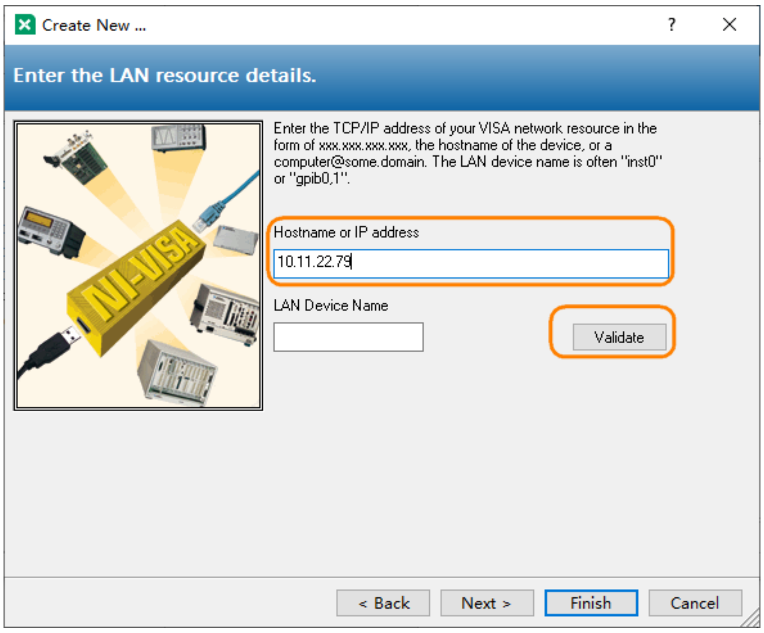
(●) Manual Entry of LAN Instrument

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

( ) Manual Entry of Raw Socket

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

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


Create New ...

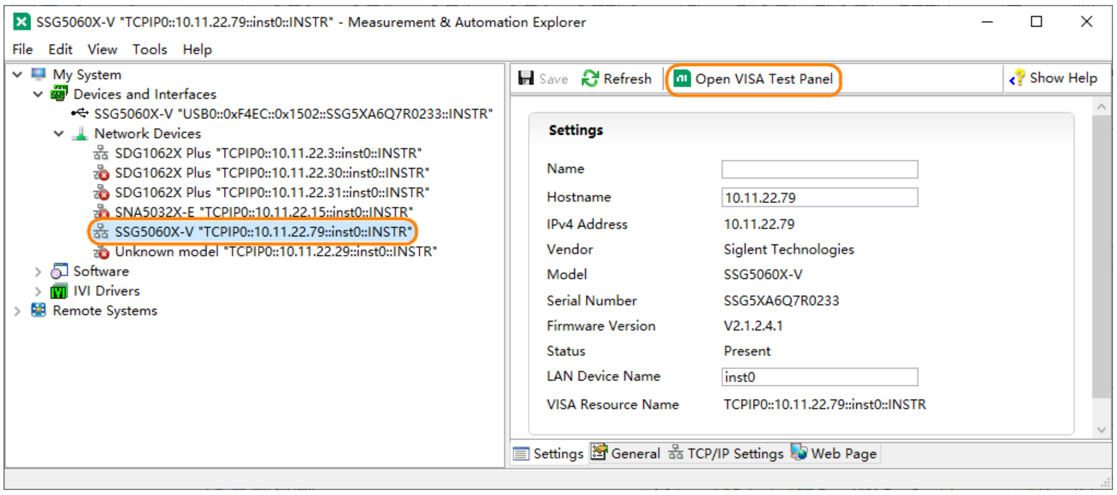
Enter the LAN resource details.

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

Hostname or IP address: 10.11.22.79

LAN Device Name:

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

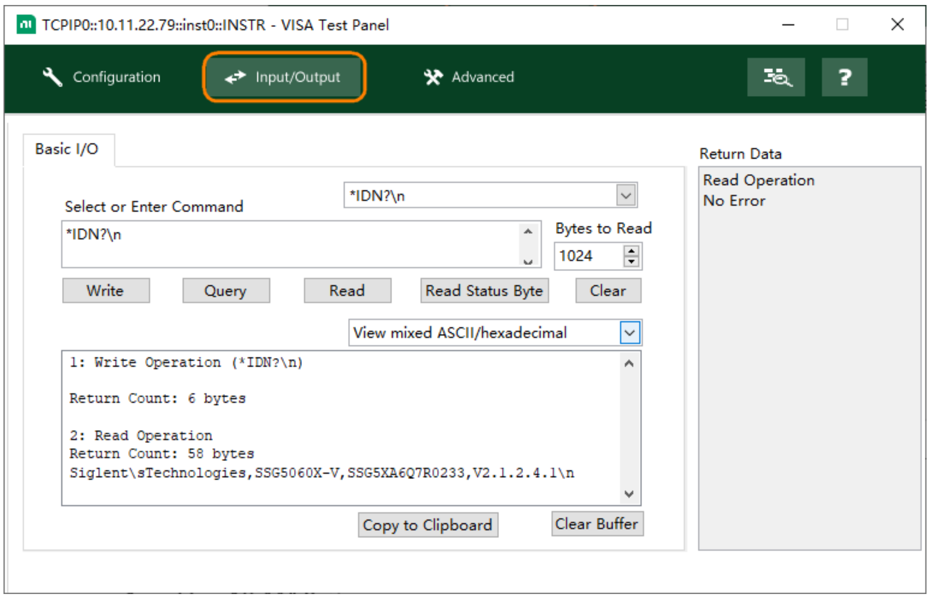

SSG5060X-V "TCPIP0::10.11.22.79::inst0::INSTR" - Measurement & Automation Explorer
File Edit View Tools Help
Save Refresh Open VISA Test Panel Show Help
Settings General TCP/IP Settings Web Page

- My System
  - Devices and Interfaces
    - SSG5060X-V "USB0::0xF4EC::0x1502::SSG5XA6Q7R0233::INSTR"
    - Network Devices
      - SDG1062X Plus "TCPIP0::10.11.22.3::inst0::INSTR"
      - SDG1062X Plus "TCPIP0::10.11.22.30::inst0::INSTR"
      - SDG1062X Plus "TCPIP0::10.11.22.31::inst0::INSTR"
      - SNA5032X-E "TCPIP0::10.11.22.15::inst0::INSTR"
      - SSG5060X-V "TCPIP0::10.11.22.79::inst0::INSTR"
      - Unknown model "TCPIP0::10.11.22.29::inst0::INSTR"
  - Software
  - IVI Drivers
- Remote Systems

Settings

Name	
Hostname	10.11.22.79
IPv4 Address	10.11.22.79
Vendor	Siglent Technologies
Model	SSG5060X-V
Serial Number	SSG5XA6Q7R0233
Firmware Version	V2.1.2.4.1
Status	Present
LAN Device Name	inst0
VISA Resource Name	TCPIP0::10.11.22.79::inst0::INSTR

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



Select or Enter Command: *IDN?\n

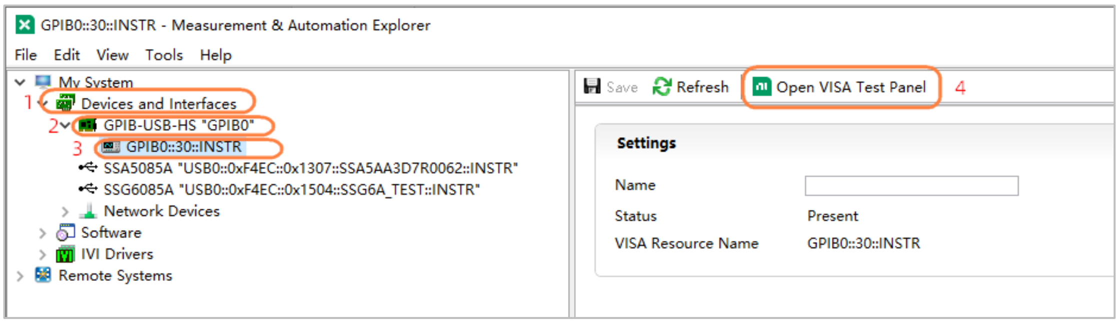
Bytes to Read: 1024

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

Return Count: 6 bytes

2: Read Operation
Return Count: 58 bytes
Siglent\sTechnologies,SSG5060X-V,SSG5XA6Q7R0233,V2.1.2.4.1\n
```

Return Data: Read Operation No Error

1.2.2.3 Using USB-GPIB

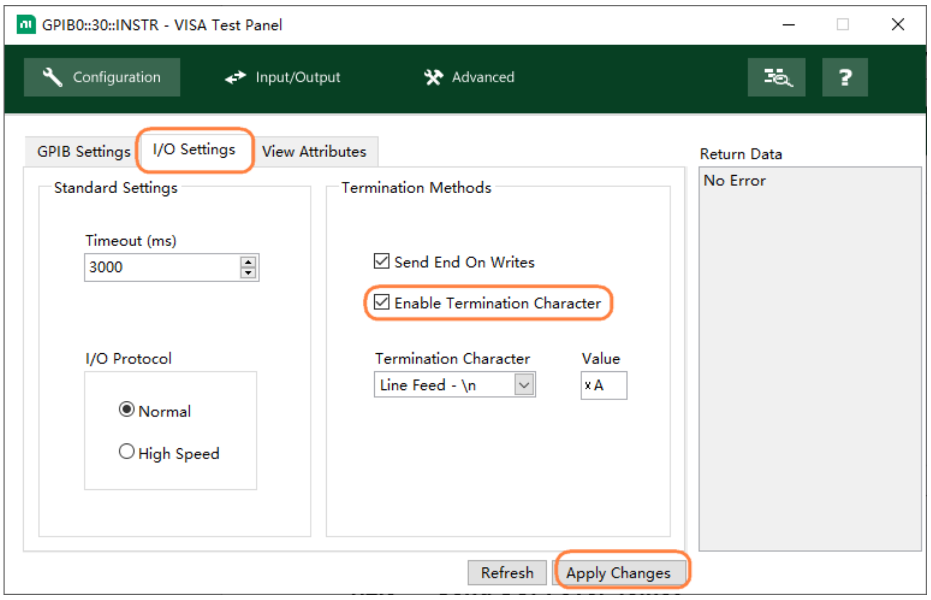
1. Run NI-MAX.
2. Click “Devices and Interfaces” in the upper left corner of the software.
3. Find the “GPIB-USB-HS” device symbol.
4. Select the signal generator device and click the “Open VISA Test Panel” button.


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

Settings

Name	
Status	Present
VISA Resource Name	GPIB0::30::INSTR

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

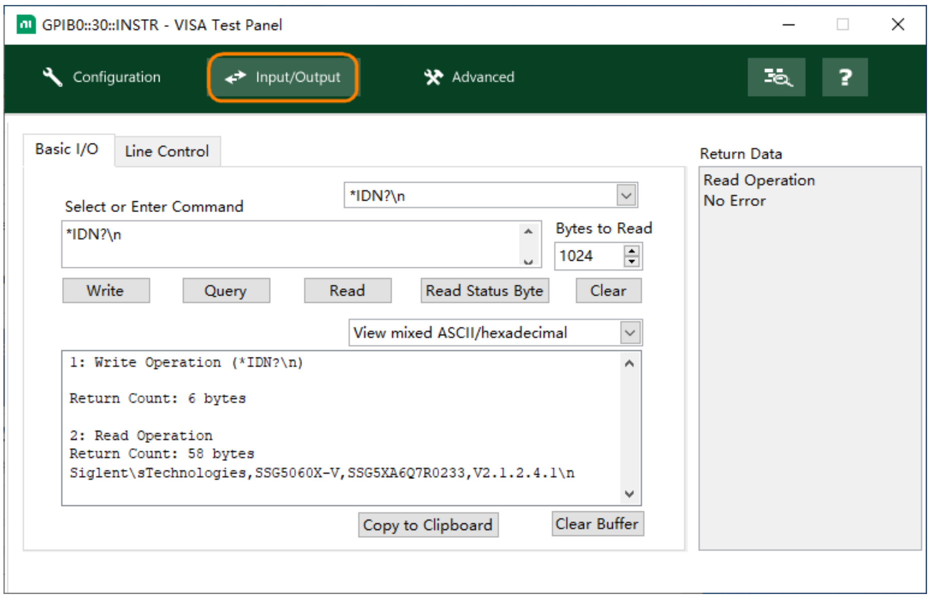


Standard Settings
Timeout (ms): 3000
I/O Protocol: [✓ Normal] High Speed

Termination Methods
☑ Send End On Writes
☑ Enable Termination Character
Termination Character: Line Feed - \n
Value: xA

Return Data
No Error

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



Select or Enter Command: *IDN?\n
Bytes to Read: 1024
View mixed ASCII/hexadecimal

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

Return Count: 6 bytes

2: Read Operation
Return Count: 58 bytes
Siglent\sTechnologies,SSG5060X-V,SSG5XA6Q7R0233,V2.1.2.4.1\n
```

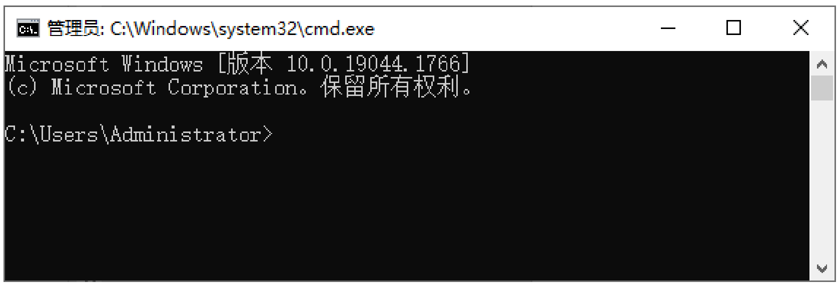
Return Data
Read Operation
No Error

1.2.3 Send SCPI over Telnet

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

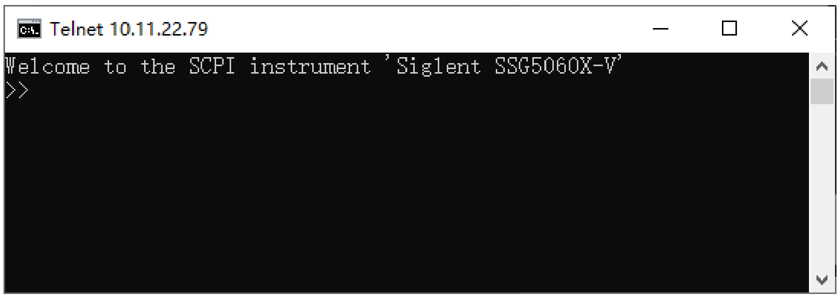
The steps are as follows:

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



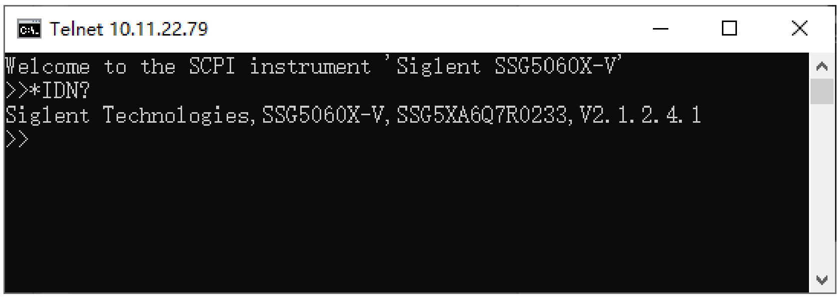
```
管理员: C:\Windows\system32\cmd.exe
Microsoft Windows [版本 10.0.19044.1766]
(c) Microsoft Corporation。保留所有权利。

C:\Users\Administrator>
```

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



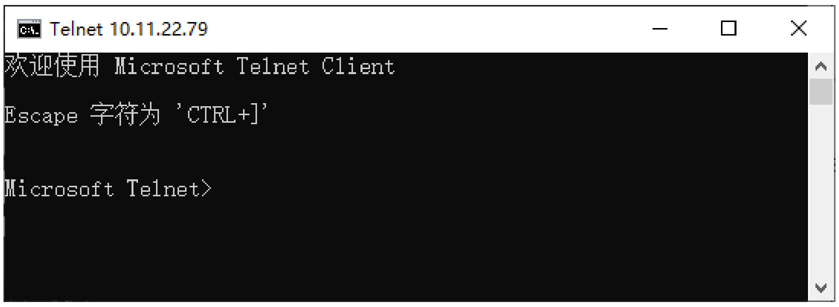
```
Telnet 10.11.22.79
Welcome to the SCPI instrument 'Siglent SSG5060X-V'
>>
```

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



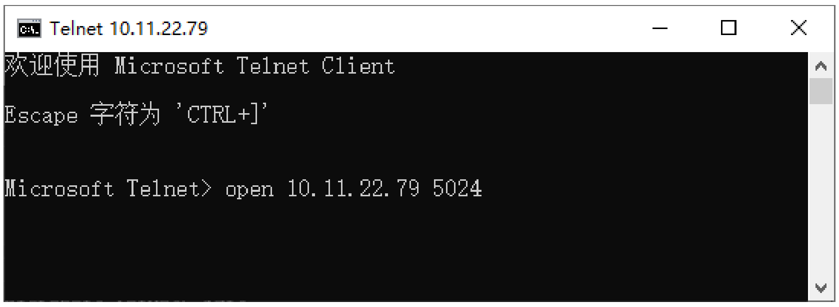
```
Telnet 10.11.22.79
Welcome to the SCPI instrument 'Siglent SSG5060X-V'
>>*IDN?
Siglent Technologies,SSG5060X-V,SSG5XA6Q7R0233,V2.1.2.4.1
>>
```

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



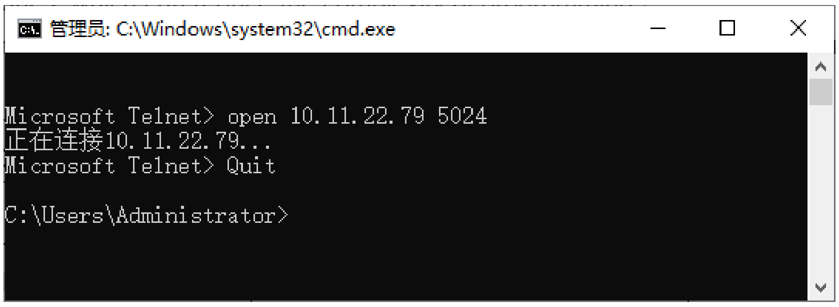
```
Telnet 10.11.22.79
欢迎使用 Microsoft Telnet Client

Escape 字符为 'CTRL+]'

Microsoft Telnet>
```

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



```
Telnet 10.11.22.79
欢迎使用 Microsoft Telnet Client

Escape 字符为 'CTRL+]'

Microsoft Telnet> open 10.11.22.79 5024
```

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



```
管理员: C:\Windows\system32\cmd.exe

Microsoft Telnet> open 10.11.22.79 5024
正在连接10.11.22.79...
Microsoft Telnet> Quit

C:\Users\Administrator>
```

1.2.4 Send SCPI over Socket

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

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

2 Introduction to the SCPI Language

2.1 Command Format

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

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

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

2.2 Symbol Instruction

The following are the symbols used in the SCPI commands:

1. Angular brackets < >

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

2. Square brackets [ ]

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

3. Vertical lines |

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

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

4. Braces { }

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

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

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

2.3 Parameter Type

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

1. Boolean

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

For example:

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

2. Enumeration

Parameter should be one of the listed values.

For example:

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

3. Integer

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

For example:

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

4. Float

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

For example:

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

5. String

The parameter should be the combination of ASCII characters.

For example:

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

6. Discrete

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

For example:

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

2.4 Command Abbreviation

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

For example:

```
:CORRection:FLATness:COUNt?
```

You can send in any of the following writing methods:

```
:CORRection:FLATness:COUNt?
:CORRECTION:FLATNESS:COUNT?
:correction:flatness:count?
```

You can also abbreviate it to:

```
:CORR:FLAT:COUN?
```

3 Commands

3.1 IEEE 488.2 Common Commands

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

3.1.1 Identification Query (*IDN?)

SYNTAX	*IDN?
DESCRIPTION	This command reads the product information (manufacturer, device model, serial number, and firmware revision number) of the device.
EQUIVALENT MENU	None
EXAMPLE	**IDN?* Return: *Siglent Technologies,SSG5060X-V,SSG5XA6Q7R0233,V2.1.2.4.1\n*

3.1.2 Reset (*RST)

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

3.1.3 Clear Status (*CLS)

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

3.1.4 Standard Event Status Enable (*ESE)

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

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

3.1.5 Standard Event Status Register Query (*ESR?)

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

3.1.6 Operation Complete Query (*OPC)

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

3.1.7 Service Request Enable (*SRE)

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

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

3.1.8 Status Byte Query (*STB?)

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

3.1.9 Wait-to-Continue (*WAI)

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

3.1.10 Self Test Query (*TST?)

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

3.2 SYSTem Commands

3.2.1 System Configuration

3.2.1.1 System Time (:SYSTem:TIME)

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

3.2.1.2 System Date (:SYSTem:DATE)

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

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

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

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

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

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

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

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

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

SYNTAX	:SYSTem:COMMunicate:LAN:TYPE STATIC|DHCP :SYSTem:COMMunicate:LAN:TYPE?
DESCRIPTION	This command sets/gets IP assignment type.
DATA TYPE	Enumeration
RANGE	STATIC|DHCP
RETURN	Enumeration

EQUIVALENT MENU	UTILITY > Interface > LAN Setting > DHCP State
EXAMPLE	*:SYSTem:COMMunicate:LAN:TYPE STATIC* *:SYSTem:COMMunicate:LAN:TYPE?* Return: *STATIC\n*

3.2.1.7 Language (SYSTem:LANGuage)

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

3.2.1.8 Screen Saver (SYSTem:SCReen:SAVer)

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

3.2.1.9 Beeper (SYSTem:ALARm)

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

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

3.2.1.10 Setup Type (:SYSTem:PON:TYPE)

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

3.2.1.11 Power On Line (SYSTem:POWeron:TYPE)

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

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

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

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

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

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

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

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

SYNTAX	:SYSTem:REF:DAC:LOAD <"file_name">
DESCRIPTION	This command loads the reference oscillator code from a DAC file.
DATA TYPE	String
RANGE	None
EQUIVALENT MENU	UTILITY > Setting > 10M Adjustment > Recall Ref Osc Setting
EXAMPLE	*:SYSTem:REF:DAC:LOAD "U-disk3/test.dac"*

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

SYNTAX	II
DESCRIPTION	This command resets the reference oscillator code to default value.
EQUIVALENT MENU	I
EXAMPLE	*:SYSTem:REF:DAC:DEFault*

3.2.1.17 GPIB Address (SYSTem:GPIB)

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

3.2.1.18 Quit Remote Control State (SYSTem:REMote 0)

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

3.2.1.19 Query Clock Reference Source (:SYStem:CLOCk?)

SYNTAX	:SYStem:CLOCk?
DESCRIPTION	Query whether the clock reference source used by the instrument is internal or external.
RETURN	EXTERNAL: The instrument uses an external clock reference, INTERNAL: The instrument uses an internal clock reference.
EQUIVALENT MENU	EXTERNAL: HOME > EXT REF logo, INTERNAL: No logo.
EXAMPLE	*:SYStem:CLOCk?* Return: *EXTERNAL\n*

3.2.2 System Reset/Preset

3.2.2.1 System Preset (:SYSTem:PRESet)

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

3.2.2.2 Preset Save (:SYSTem:PRESet:SAVE)

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

3.2.2.3 Preset Path (:SYSTem:PRESet:PATH)

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

3.2.2.4 Preset Type (:SYSTem:PRESet:TYPE)

SYNTAX	:SYSTem:PRESet:TYPE DFT|USER :SYSTem:PRESet:TYPE?

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

3.2.2.5 Factory Reset (:SYSTem:FDEFault)

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

3.2.2.6 Reset & Clear (SYSTem:RESet:CLEar)

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

3.3 OUTPut Commands

3.3.1 RF Output (:OUTPut[:STATe])

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

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

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

3.4 SOURce Commands

3.4.1 RF Output ([:SOURce]:OUTPut)

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

3.4.2 Software Trigger(*TRG)

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

3.4.3 Frequency

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

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

3.4.3.2 Frequency ([:SOURce]:FREQuency)

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

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

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

3.4.3.4 Phase Offset ([:SOURce]:PHASe)

SYNTAX	[:SOURce]:PHASe <phase> [:SOURce]:PHASe?
DESCRIPTION	This command sets/gets the phase of the RF output signal.
DATA TYPE	Float, unit: degree
RANGE	-360 ~ 360
RETURN	Float, unit: degree
DEFAULT VALUE	0
EQUIVALENT MENU	FREQ > Phase Offset
EXAMPLE	*PHASe 20*

	PHASe? Return: *20\n*

3.4.3.5 Phase Reset ([:SOURce]:PHASe:RESet)

SYNTAX	[:SOURce]:PHASe:RESet
DESCRIPTION	This command resets the phase offset display value of the RF signal to 0.
EQUIVALENT SCPI	[:SOURce]:PHASe:REF
EQUIVALENT MENU	FREQ > Reset phase delta display
EXAMPLE	*:PHASe:RESet*

3.4.3.6 Phase Reset ([:SOURce]:PHASe:REF)

SYNTAX	[:SOURce]:PHASe:REF
DESCRIPTION	This command resets the phase offset display value of the RF signal to 0.
EQUIVALENT SCPI	[:SOURce]:PHASe:RESet
EQUIVALENT MENU	FREQ > Reset phase delta display
EXAMPLE	*:PHASe:REF*

3.4.4 Level

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

SYNTAX	[:SOURce]:POWer:POWer <power> [:SOURce]:POWer:POWer?
DESCRIPTION	This command sets/gets the level display value in the parameter bar at the top of the screen.
DATA TYPE	Float, unit: dBm, dBuV, uV, mV, V, nW, uW, mW or W, default is dBm
RANGE	Level Offset + Full power range
RETURN	Float, unit: dBm
DEFAULT VALUE	-130 dBm
EQUIVALENT MENU	Level
EXAMPLE	*:POWer:POWer -2* *:POWer:POWer?* Return: *-2\n*

3.4.4.2 Level ([:SOURce]:POWer)

SYNTAX	[:SOURce]:POWer <power> [:SOURce]:POWer?
DESCRIPTION	This command sets/gets the level of the RF output signal.
DATA TYPE	Float, unit: dBm, dBuV, uV, mV, V, nW, uW, mW or W, default is dBm
RANGE	Please refer to SSG5000X datasheet
RETURN	Float, unit: dBm
DEFAULT VALUE	-130 dBm
EQUIVALENT MENU	LEVEL > Level
EXAMPLE	*:POWer 2* *:POWer?* Return: *2\n*

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

SYNTAX	[:SOURce]:POWer[:LEVel][:IMMediate][:AMPLitude] <power> [:SOURce]:POWer[:LEVel][:IMMediate][:AMPLitude]?
DESCRIPTION	This command sets/gets the level of the RF output signal.
DATA TYPE	Float, unit: dBm, dBuV, uV, mV, V, nW, uW, mW or W, default is dBm
RANGE	Please refer to SSG5000X datasheet
RETURN	Float, unit: dBm
DEFAULT VALUE	-130 dBm
EQUIVALENT MENU	LEVEL > Level
EXAMPLE	*:POWer:LEVel -5* *:POWer:LEVel?* Return: *-5\n*

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

SYNTAX	[:SOURce]:POWer:OFFSet <power> [:SOURce]:POWer:OFFSet?
DESCRIPTION	This command sets/gets the level offset of the RF output signal.
DATA TYPE	Float, unit: dB
RANGE	-100 dB to 100 dB
RETURN	Float, unit: dB
DEFAULT VALUE	0
EQUIVALENT MENU	LEVEL > Level Offset
EXAMPLE	*:POWer:OFFSet 2*

	:POWer:OFFSet? Return: *2\n*

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

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

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

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

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

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

EQUIVALENT MENU	LEVEL > Flatness > Add
EXAMPLE	*CORRection:FLATness:PAIR 3 MHz,-3*

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

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

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

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

3.4.4.10 Flatness List Content ([:SOURce]:CORRection:FLATness:LIST?)

SYNTAX	[:SOURce]:CORRection:FLATness:LIST? <start_row>,<stop_row>
DESCRIPTION	This command gets the flatness list content from *start_row* to *stop_row*. The index of flatness list starts from 1. Please note that when there is a frequency offset, the frequency offset value needs to be added to the correction frequency.
RETURN	String
DEFAULT VALUE	None
EQUIVALENT MENU	LEVEL > Flatness
EXAMPLE	*:CORRection:FLATness:LIST? 1,3* Return: *-1000000000,0.01 -500000000,-0.02 0,0.03 \n*

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

SYNTAX	[:SOURce]:CORRection:STORe <"file_name">

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

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

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

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

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

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

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

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

SYNTAX	[:SOURce]:CORRection:FLATness:STARtfreq <freq> [:SOURce]:CORRection:FLATness:STARtfreq?
DESCRIPTION	When the flatness list needs to be filled with a power sensor and the filling type is “Manual Step”, this command sets/queries the starting frequency of manual step filling.
DATA TYPE	Float, unit: Hz, kHz, MHz or GHz, default is Hz
RANGE	Full frequency range
RETURN	Float, unit: Hz
DEFAULT VALUE	Maximum frequency
EQUIVALENT MENU	LEVEL > Flatness > Setting > Fill Type (Manual Step) > Start Freq
EXAMPLE	*:CORRection:FLATness:STARtfreq 200 MHz* *:CORRection:FLATness:STARtfreq?* Return: *200000000\n*

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

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

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

SYNTAX	[:SOURce]:CORRection:FLATness:SPACe LINear|LOGarithmic [:SOURce]:CORRection:FLATness:SPACe?
DESCRIPTION	When the flatness list needs to be filled with a power sensor and the filling type is “Manual Step”, this command sets/queries the frequency step method of manual step filling.
DATA TYPE	Enumeration
RANGE	LINear|LOGarithmic
RETURN	Enumeration

DEFAULT VALUE	LINear
EQUIVALENT MENU	LEVEL > Flatness > Setting > Fill Type (Manual Step) > Fill Space
EXAMPLE	*:CORRection:FLATness:SPACe LINear* *:CORRection:FLATness:SPACe?* Return: *LINear\n*

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

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

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

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

3.4.4.20 Flatness List Fill Points ([:SOURce]:CORRection:FLATness:FILL:POINt)

SYNTAX	[:SOURce]:CORRection:FLATness:FILL:POINt <points> [:SOURce]:CORRection:FLATness:FILL:POINt?
DESCRIPTION	This command sets/queries the sweep points of manual step filling.
DATA TYPE	Integer

RANGE	2 to 500
RETURN	Integer
DEFAULT VALUE	2
EQUIVALENT MENU	LEVEL > Flatness > Setting > Fill Type (Manual Step) > Points
EXAMPLE	*:CORRection:FLATness:FILL:POINt 5* *:CORRection:FLATness:FILL:POINt?* Return: *5\n*

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

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

3.4.5 Sweep

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

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

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

SYNTAX	[:SOURce]:SWEep:TYPE LIST|STEP [:SOURce]:SWEep:TYPE?
DESCRIPTION	This command sets the sweep type to step sweep or list sweep. This command queries whether the sweep type is step sweep or list sweep.

DATA TYPE	Enumeration
RANGE	LIST|STEP
RETURN	Enumeration
DEFAULT VALUE	STEP
EQUIVALENT MENU	SWEEP > Step Sweep / List Sweep
EXAMPLE	*:SWEep:TYPE STEP* *:SWEep:TYPE?* Return: *STEP\n*

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

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

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

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

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

SYNTAX	[:SOURce]:SWEep:STEP:STARt:LEVel <level> [:SOURce]:SWEep:STEP:STARt:LEVel?
DESCRIPTION	This command sets/queries the starting level of step sweep.
DATA TYPE	Float, unit: dBm, dBuV, uV, mV, V, nW, uW, mW or W, default is dBm
RANGE	Please refer to SSG5000X datasheet
RETURN	Float, unit: dBm
DEFAULT VALUE	-130 dBm
EQUIVALENT MENU	SWEEP > Step Sweep > Start Level
EXAMPLE	*:SWEep:STEP:STARt:LEVel 0 dBm* *:SWEep:STEP:STARt:LEVel?* Return: *0\n*

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

SYNTAX	[:SOURce]:SWEep:STEP:STOP:LEVel <level> [:SOURce]:SWEep:STEP:STOP:LEVel?
DESCRIPTION	This command sets/queries the stop level of step sweep.
DATA TYPE	Float, unit: dBm, dBuV, uV, mV, V, nW, uW, mW or W, default is dBm
RANGE	Please refer to SSG5000X datasheet
RETURN	Float, unit: dBm
DEFAULT VALUE	-130 dBm
EQUIVALENT MENU	SWEEP > Step Sweep > Stop Level
EXAMPLE	*:SWEep:STEP:STOP:LEVel 0 dBm* *:SWEep:STEP:STOP:LEVel?* Return: *0\n*

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

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

	:SWEep:STEP:DWELl? Return: *0.02\n*

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

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

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

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

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

SYNTAX	[:SOURce]:SWEep:STEP:SPACe LINear|LOGarithmic [:SOURce]:SWEep:STEP:SPACe?
DESCRIPTION	This command sets/queries the frequency sweep space of step sweep.
DATA TYPE	Enumeration
RANGE	LINear|LOGarithmic

RETURN	Enumeration
DEFAULT VALUE	LINear
EQUIVALENT MENU	SWEEP > Step Sweep > Sweep Space
EXAMPLE	*:SWEep:STEP:SPACe LOGarithmic* *:SWEep:STEP:SPACe?* Return: *LOGarithmic\n*

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

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

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

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

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

SYNTAX	[:SOURce]:SWEep:LIST:ADDList <freq>,<level>,<time>
DESCRIPTION	This command adds a line to the sweep list.
DATA TYPE	Freq: float, unit: Hz, kHz, MHz or GHz, default is Hz, Level: float, unit: dBm, dBuV, uV, mV, V, nW, uW, mW or W, default is dBm, Time: float, unit: ns, us, ms or s, default is s
RANGE	Freq: Full frequency range, Level: Full level range, Time: 10.0 ms ~ 100.0 s
EQUIVALENT MENU	SWEEP > List Sweep > Add
EXAMPLE	*:SWEep:LIST:ADDList 1 GHz,0 dBm,1 s*

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

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

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

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

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

SYNTAX	[:SOURce]:SWEep:LIST:CPOint?
DESCRIPTION	This command queries the total number of rows of the sweep list.

RETURN	Integer
DEFAULT VALUE	1
EQUIVALENT MENU	SWEEP > List Sweep
EXAMPLE	*:SWEep:LIST:CPOint?* Return: *5\n*

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

SYNTAX	[:SOURce]:SWEep:LIST:LIST? <begin_row>,<end_row>
DESCRIPTION	This command queries the data from begin_row to end_row in the sweep list.
DATA TYPE	Integer, Integer
RANGE	1 ~ total number of rows in the sweep list, Start row ~ total number of rows in the sweep list
RETURN	String
DEFAULT VALUE	None
EQUIVALENT MENU	SWEEP > List Sweep
EXAMPLE	*:SWEep:LIST:LIST? 1,3* Return: *1000000000,0,0.03 2000000000,-2.4,0.03 3000000000,-4.8,0.03\n*

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

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

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

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

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

SYNTAX	[:SOURce]:SWEep:LOAD <"file_name">
DESCRIPTION	This command loads the sweep list from a LSW file.

DATA TYPE	String
RANGE	None
EQUIVALENT MENU	SWEEP > List Sweep > Open
EXAMPLE	*:SWEep:LOAD "U-disk3/test.lsw"*

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

SYNTAX	[:SOURce]:SWEep:STORe <"file_name">
DESCRIPTION	This command saves the sweep list into a LSW file.
DATA TYPE	String
RANGE	None
EQUIVALENT MENU	SWEEP > List Sweep > Save
EXAMPLE	*:SWEep:STORe "U-disk3/test.lsw"*

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

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

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

SYNTAX	[:SOURce]:SWEep:MODE CONTinue|SINGle [:SOURce]:SWEep:MODE?
DESCRIPTION	This command sets the sweep mode to continuous or single. This command queries whether the sweep mode is continuous or single.
DATA TYPE	Enumeration
RANGE	CONTinue|SINGle
RETURN	Enumeration
DEFAULT VALUE	CONTinue

EQUIVALENT MENU	SWEEP > Sweep Mode
EXAMPLE	*:SWEep:MODE SINGle* *:SWEep:MODE?* Return: *SINGle\n*

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

SYNTAX	[:SOURce]:SWEep:EXECute
DESCRIPTION	When the sweep mode is single, this command can perform a single sweep.
EQUIVALENT MENU	SWEEP > Execute single sweep
EXAMPLE	*:SWEep:EXECute*

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

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

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

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

	Return: *KEY\n*

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

SYNTAX	[:SOURce]:INPut:TRIGger:SLOPe POSitive|NEGative [:SOURce]:INPut:TRIGger:SLOPe?
DESCRIPTION	This command sets/queries the trigger edge of the external trigger signal for the sweep function.
DATA TYPE	Enumeration
RANGE	POSitive|NEGative
RETURN	Enumeration
DEFAULT VALUE	POSitive
EQUIVALENT MENU	SWEEP > Trigger Slope
EXAMPLE	*:INPut:TRIGger:SLOPe NEGative* *:INPut:TRIGger:SLOPe?* Return: *NEGative\n*

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

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

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

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

EXAMPLE	*:SWEep:CURRent:FREQuency?* Return: *1000000000.000000\n*

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

SYNTAX	[:SOURce]:SWEep:CURRent:LEVel?
DESCRIPTION	This command queries the level of the current sweep point.
RETURN	Float, unit: dBm
DEFAULT VALUE	None
EQUIVALENT MENU	Display level at the top of the screen
EXAMPLE	*:SWEep:CURRent:LEVel?* Return: *0.000000\n*

3.4.6 Sensor

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

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

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

SYNTAX	[:SOURce]:POWer:SPC:TARGet <power> [:SOURce]:POWer:SPC:TARGet?
DESCRIPTION	This command sets/queries the targrt level of the power sensor level control.
DATA TYPE	Float, unit: dBm, dBuV, uV, mV, V, nW, uW, mW or W, default is dBm
RANGE	-120 dBm ~ 20 dBm
RETURN	Float, unit: dBm

DEFAULT VALUE	0
EQUIVALENT SCPI	:SENSe[:POWer]:SPC:TARGet <power> :SENSe[:POWer]:SPC:TARGet?
EQUIVALENT MENU	HOME > POWER SENSOR > Level Control > Target Level
EXAMPLE	*:POWer:SPC:TARGet 0* *:POWer:SPC:TARGet?* Return: *0\n*

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

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

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

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

3.4.7 Anolog Modulation

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

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

3.4.7.2 AM

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

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

3.4.7.2.2 AM Shape ([:SOURce]:AM:WAVeform)

SYNTAX	[:SOURce]:AM:WAVeform SINE|SQUare [:SOURce]:AM:WAVeform?
DESCRIPTION	This command sets/queries the modulation waveform of AM.
DATA TYPE	Enumeration
RANGE	SINE|SQUare

RETURN	Enumeration
DEFAULT VALUE	SINE
EQUIVALENT MENU	MOD > AM > AM Shape
EXAMPLE	*:AM:WAVeform SQUare* *:AM:WAVeform?* Return: *SQUare\n*

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

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

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

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

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

SYNTAX	[:SOURce]:AM:FREQuency <value> [:SOURce]:AM:FREQuency?

DESCRIPTION	This command sets/queries the modulation rate of AM.
DATA TYPE	Float, unit: Hz, kHz, MHz or GHz, default is Hz
RANGE	Modulation wave is Sine: 0.01 Hz ~ 100 kHz, Modulation wave is Square: 0.01 Hz ~ 20 kHz.
RETURN	Float, unit: Hz
DEFAULT VALUE	1 kHz
EQUIVALENT MENU	MOD > AM > AM Rate
EXAMPLE	*:AM:FREQuency 10 kHz* *:AM:FREQuency?* Return: *10000\n*

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

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

3.4.7.3 FM

3.4.7.3.1 FM State ([:SOURce]:FM:STATe)

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

3.4.7.3.2 FM Source ([:SOURce]:FM:SOURce)

SYNTAX	[:SOURce]:FM:SOURce INT1|INT2|INT1,INT2|EXTernal|INT1,EXT|DUAL [:SOURce]:FM:SOURce?
DESCRIPTION	This command sets/queries the modulation source of FM.
DATA TYPE	Enumeration
RANGE	INT1|INT2|INT1,INT2|EXTernal|INT1,EXT|DUAL
RETURN	Enumeration
DEFAULT VALUE	INT1
EQUIVALENT MENU	MOD > FM > FM Source
EXAMPLE	*:FM:SOURce EXTernal* *:FM:SOURce?* Return: *EXTernal\n*

3.4.7.3.3 FM Shape1 ([:SOURce]:FM1:WAVeform)

SYNTAX	[:SOURce]:FM1:WAVeform SINE|SQUare|SAWTooth|TRIangle [:SOURce]:FM1:WAVeform?
DESCRIPTION	This command sets/queries the frequency modulation waveform of INT1 modulation source.
DATA TYPE	Enumeration
RANGE	SINE|SQUare|SAWTooth|TRIangle
RETURN	Enumeration
DEFAULT VALUE	SINE
EQUIVALENT MENU	MOD > FM > FM Shape1
EXAMPLE	*:FM1:WAVeform SQUare* *:FM1:WAVeform?* Return: *SQUare\n*

3.4.7.3.4 FM Deviation1 ([:SOURce]:FM1:DEViation)

SYNTAX	[:SOURce]:FM1:DEViation <value> [:SOURce]:FM1:DEViation?
DESCRIPTION	This command sets/queries the deviation value of the FM waveform of the INT1 modulation source.
DATA TYPE	Float, unit: Hz, kHz, MHz or GHz, default is Hz
RANGE	1 Hz ~ N*1 MHz, N is the frequency segment coefficient, please refer to the data sheet
RETURN	Float, unit: Hz
DEFAULT VALUE	100 kHz

EQUIVALENT MENU	MOD > FM > FM Deviation1
EXAMPLE	*:FM1:DEViation 200 kHz* *:FM1:DEViation?* Return: *200000\n*

3.4.7.3.5 FM Rate1 ([:SOURce]:FM1:FREQuency)

SYNTAX	[:SOURce]:FM1:FREQuency <value> [:SOURce]:FM1:FREQuency?
DESCRIPTION	This command sets/queries the FM rate of the INT1 modulation source.
DATA TYPE	Float, unit: Hz, kHz, MHz or GHz, default is Hz
RANGE	Modulation wave is Sine: 0.01 Hz ~ 100 kHz, Modulation wave is Square/Sawtooth/Triangle: 0.01 Hz ~ 20 kHz.
RETURN	Float, unit: Hz
DEFAULT VALUE	10 kHz
EQUIVALENT MENU	MOD > FM > FM Rate1
EXAMPLE	*:FM1:FREQuency 5 kHz* *:FM1:FREQuency?* Return: *5000\n*

3.4.7.3.6 FM Phase1 ([:SOURce]:FM1:PHASe)

SYNTAX	[:SOURce]:FM1:PHASe <value> [:SOURce]:FM1:PHASe?
DESCRIPTION	When the frequency modulation source is internal source 1 + internal source 2 (Int1 + Int2) or Dual waveform, this command sets/queries the initial phase of internal source 1.
DATA TYPE	Float, unit: degree (°)
RANGE	-360 ~ +360
RETURN	Float, unit: degree (°)
DEFAULT VALUE	0
EQUIVALENT MENU	MOD > FM > FM Phase1
EXAMPLE	*:FM1:PHASe -30* *:FM1:PHASe?* Return: *-30\n*

3.4.7.3.7 FM Shape2 ([:SOURce]:FM2:WAVeform)

SYNTAX	[:SOURce]:FM2:WAVeform SINE|SQUare|SAWTooth|TRIangle [:SOURce]:FM2:WAVeform?

DESCRIPTION	This command sets/queries the frequency modulation waveform of INT2 modulation source.
DATA TYPE	Enumeration
RANGE	SINE|SQUare|SAWTooth|TRIangle
RETURN	Enumeration
DEFAULT VALUE	SINE
EQUIVALENT MENU	MOD > FM > FM Shape2
EXAMPLE	*:FM2:WAVeform TRIangle* *:FM2:WAVeform?* Return: *TRIangle\n*

3.4.7.3.8 FM Deviation2 ([:SOURce]:FM2:DEViation)

SYNTAX	[:SOURce]:FM2:DEViation <value> [:SOURce]:FM2:DEViation?
DESCRIPTION	This command sets/queries the deviation value of the FM waveform of the INT2 modulation source.
DATA TYPE	Float, unit: Hz, kHz, MHz or GHz, default is Hz
RANGE	1 Hz ~ N*1 MHz, N is the frequency segment coefficient, please refer to the data sheet
RETURN	Float, unit: Hz
DEFAULT VALUE	100 kHz
EQUIVALENT MENU	MOD > FM > FM Deviation2
EXAMPLE	*:FM2:DEViation 200 kHz* *:FM2:DEViation?* Return: *200000\n*

3.4.7.3.9 FM Rate2 ([:SOURce]:FM2:FREQuency)

SYNTAX	[:SOURce]:FM2:FREQuency <value> [:SOURce]:FM2:FREQuency?
DESCRIPTION	This command sets/queries the FM rate of the INT2 modulation source.
DATA TYPE	Float, unit: Hz, kHz, MHz or GHz, default is Hz
RANGE	Modulation wave is Sine: 0.01 Hz ~ 100 kHz, Modulation wave is Square/Sawtooth/Triangle: 0.01 Hz ~ 20 kHz.
RETURN	Float, unit: Hz
DEFAULT VALUE	10 kHz
EQUIVALENT MENU	MOD > FM > FM Rate2
EXAMPLE	*:FM2:FREQuency 40 kHz* *:FM2:FREQuency?*

	Return: *40000\n*

3.4.7.3.10 FM Phase2 ([:SOURce]:FM2:PHASe)

SYNTAX	[:SOURce]:FM2:PHASe <value> [:SOURce]:FM2:PHASe?
DESCRIPTION	When the frequency modulation source is internal source 1 + internal source 2 (Int1 + Int2) or Dual waveform, this command sets/queries the initial phase of internal source 2.
DATA TYPE	Float, unit: degree (°)
RANGE	-360 ~ +360
RETURN	Float, unit: degree (°)
DEFAULT VALUE	0
EQUIVALENT MENU	MOD > FM > FM Phase2
EXAMPLE	*:FM2:PHASe -30* *:FM2:PHASe?* Return: *-30\n*

3.4.7.3.11 Int1 Proportion ([:SOURce]:FM1:PROPortion)

SYNTAX	[:SOURce]:FM1:PROPortion <value> [:SOURce]:FM1:PROPortion?
DESCRIPTION	This command sets/queries the proportion of waveform1 when the FM Source is Dual.
DATA TYPE	Float
RANGE	0 ~ 1
RETURN	Float
DEFAULT VALUE	0.5
EQUIVALENT MENU	MOD > FM > Int1 Proportion
EXAMPLE	*:FM1:PROPortion 0.6* *:FM1:PROPortion?* Return: *0.6\n*

3.4.7.3.12 FM Sensitivity1 ([:SOURce]:FM1:SENSitivity?)

SYNTAX	[:SOURce]:FM1:SENSitivity?
DESCRIPTION	When the frequency modulation source includes an external source, this command queries the sensitivity of the external source input.
RETURN	Float, unit: Hz/V

DEFAULT VALUE	None
EQUIVALENT MENU	MOD > FM > FM Sensitivity1
EXAMPLE	*:FM1:SENSitivity?* Return: *125000\n*

3.4.7.4 PM

3.4.7.4.1 PM State ([:SOURce]:PM:STATe)

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

3.4.7.4.2 PM Shape ([:SOURce]:PM:WAVeform)

SYNTAX	[:SOURce]:PM:WAVeform SINE|SQUare [:SOURce]:PM:WAVeform?
DESCRIPTION	This command sets/queries the modulation waveform of PM.
DATA TYPE	Enumeration
RANGE	SINE|SQUare
RETURN	Enumeration
DEFAULT VALUE	SINE
EQUIVALENT MENU	MOD > PM > PM Shape
EXAMPLE	*:PM:WAVeform SQUare* *:PM:WAVeform?* Return: *SQUare\n*

3.4.7.4.3 PM Source ([:SOURce]:PM:SOURce)

SYNTAX	[:SOURce]:PM:SOURce INTernal|EXTernal|INT,EXT [:SOURce]:PM:SOURce?
DESCRIPTION	This command sets/queries the modulation source of PM.

DATA TYPE	Enumeration
RANGE	INTernal|EXTernal|INT,EXT
RETURN	Enumeration
DEFAULT VALUE	INTernal
EQUIVALENT MENU	MOD > PM > PM Source
EXAMPLE	*:PM:SOURce EXTernal* *:PM:SOURce?* Return: *EXTernal\n*

3.4.7.4.4 PM Deviation ([:SOURce]:PM:DEViation)

SYNTAX	[:SOURce]:PM:DEViation <value> [:SOURce]:PM:DEViation?
DESCRIPTION	This command sets/queries the modulation deviation of the phase modulation.
DATA TYPE	Float, unit: rad
RANGE	0.01 rad ~ N*5 rad, N is the frequency segment coefficient, please refer to the data sheet
RETURN	Float, unit: rad
DEFAULT VALUE	1
EQUIVALENT MENU	MOD > PM > PM Deviation
EXAMPLE	*:PM:DEViation 2* *:PM:DEViation?* Return: *2\n*

3.4.7.4.5 PM Rate ([:SOURce]:PM:FREQuency)

SYNTAX	[:SOURce]:PM:FREQuency <value> [:SOURce]:PM:FREQuency?
DESCRIPTION	This command sets/queries the modulation rate of PM.
DATA TYPE	Float, unit: Hz, kHz, MHz or GHz, default is Hz
RANGE	Modulation wave is Sine: 0.01 Hz ~ 100 kHz, Modulation wave is Square: 0.01 Hz ~ 20 kHz.
RETURN	Float, unit: Hz
DEFAULT VALUE	10 kHz
EQUIVALENT MENU	MOD > PM > PM Rate
EXAMPLE	*:PM:FREQuency 1 kHz* *:PM:FREQuency?* Return: *1000\n*

3.4.7.4.6 PM Sensitivity ([:SOURce]:PM:SENSitivity?)

SYNTAX	[:SOURce]:PM:SENSitivity?
DESCRIPTION	This command queries the sensitivity of PM external modulation.
RETURN	Float, unit: rad/V
DEFAULT VALUE	None
EQUIVALENT MENU	MOD > PM > PM Sensitivity
EXAMPLE	*PM:SENSitivity?* Return: *0.25\n*

3.4.8 Pulse Modulation

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

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

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

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

EXAMPLE	*:PULM:SOURce INTernal* *:PULM:SOURce?* Return: *INTernal\n*

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

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

3.4.8.4 Pulse External Source Polarity ([:SOURce]:PULM:EXTernal:POLarity)

SYNTAX	[:SOURce]:PULM:EXTernal:POLarity NORMal|INVerted [:SOURce]:PULM:EXTernal:POLarity?
DESCRIPTION	This command sets/queries the polarity of the external modulation source of the pulse function.
DATA TYPE	Enumeration
RANGE	NORMal|INVerted
RETURN	Enumeration
DEFAULT VALUE	NORMal
EQUIVALENT MENU	MOD > PULSE > Pulse Source (Ext) > Ext Polarity
EXAMPLE	*:PULM:EXTernal:POLarity INVerted* *:PULM:EXTernal:POLarity?* Return: *INVerted\n*

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

SYNTAX	[:SOURce]:PULM:OUT:STATe ON|OFF|1|0 [:SOURce]:PULM:OUT:STATe?

DESCRIPTION	This command sets/queries the pulse output status of pulse modulation.
DATA TYPE	Boolean
RANGE	ON|OFF|1|0
RETURN	1|0
DEFAULT VALUE	0
EQUIVALENT MENU	MOD > PULSE > Pulse Out
EXAMPLE	*:PULM:OUT:STATe ON* *:PULM:OUT:STATe?* Return: *1\n*

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

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

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

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

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

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

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

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

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

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

RANGE	20 ns ~ 300 s
RETURN	Float, unit: s
DEFAULT VALUE	2 ms
EQUIVALENT SCPI	[:SOURce]:PULM:INT[1]:PWIDth <value> [:SOURce]:PULM:INT[1]:PWIDth?
EQUIVALENT MENU	MOD > PULSE > Pulse Width
EXAMPLE	*PULM:WIDT 33 us* *PULM:WIDT?* Return: *3.3e-05\n*

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

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

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

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

	:PULM:DOUBle:DELay? Return: *0.002\n*

3.4.8.13 #2 Width ([:SOURce]:PULM:DOUBle:WIDTh)

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

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

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

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

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

3.4.8.16 Pulse Train Edit On Time ([:SOURce]:PULM:TRAin:DATA:ONTime)

SYNTAX	[:SOURce]:PULM:TRAin:DATA:ONTime <raw>,<on_time>
DESCRIPTION	This command edits the positive pulse width of the specified row in the pulse train.
DATA TYPE	Row: Integer, On_time: float, unit: ns, us, ms or s, default is s
RANGE	Row: 1 ~ total number of rows in the pulse train, On_time: 20 ns ~ 300 s
EQUIVALENT MENU	MOD > PULSE > Pulse Train
EXAMPLE	*:PULM:TRAin:DATA:ONTime 1,10 ms*

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

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

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

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

3.4.8.19 Pulse Train Edit ([:SOURce]:PULM:TRAin:CHANge)

SYNTAX	[:SOURce]:PULM:TRAin:CHANge <row>,<on_time>,<off_time>,<count>
DESCRIPTION	This command edits the specified row of the pulse train.
DATA TYPE	Row: integer, On_time: float, unit: ns, us, ms or s, default is s,

	Off_time: float, unit: ns, us, ms or s, default is s, Count: integer
RANGE	Row: 1 ~ total number of rows in the pulse train, On_time: 20 ns ~ 300 s, Off_time: 20 ns ~ 300 s, Count: 1 ~ 65535
EQUIVALENT MENU	MOD > PULSE > Pulse Train
EXAMPLE	*:PULM:TRAin:CHANge 2,3 ms,500 ns,4*

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

SYNTAX	[:SOURce]:PULM:TRAin:LIST? <begin_row>,<end_row>
DESCRIPTION	This command queries the data from begin_row to end_row in the pulse train.
DATA TYPE	Integer, Integer
RANGE	1 ~ total number of rows in the pulse train, Start row ~ total number of rows in the pulse train
RETURN	String
DEFAULT VALUE	None
EQUIVALENT MENU	MOD > PULSE > Pulse Train
EXAMPLE	*:PULM:TRAin:LIST? 1,3* Return: *0.001,0.005,1 0.003,0.003,2 0.004,0.002,1\n*

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

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

3.4.8.22 Pulse Train Clear ([:SOURce]:PULM:TRAin:CLEar)

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

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

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

3.4.8.24 Pulse Train Store ([:SOURce]:PULM:TRAin:STORe)

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

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

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

3.4.8.26 Pulse Trigger Mode ([:SOURce]:PULM:TRIGger:MODE)

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

RANGE	AUTO|SINGle|BUS|EXTernal|EGATe
RETURN	Enumeration
DEFAULT VALUE	AUTO
EQUIVALENT MENU	MOD > PULSE > Pulse Trigger
EXAMPLE	*:PULM:TRIG:MODE EXTernal* *:PULM:TRIGger:MODE?* Return: *EXTernal\n*

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

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

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

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

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

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

3.4.9 LF Source

3.4.9.1 LF State ([:SOURce]:LFOutput)

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

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

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

RETURN	Float, unit: Vpp
DEFAULT VALUE	0.5 Vpp
EQUIVALENT MENU	LF > LF Source > LF Level
EXAMPLE	*LFOutput:VOLTage 2 Vpp* *LFOutput:VOLTage?* Return: *2\n*

3.4.9.3 LF Offset ([:SOURce]:LFOutput:OFFSet)

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

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

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

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

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

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

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

3.4.10 LF Sweep

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

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

EQUIVALENT MENU	LF > LF Sweep > Sweep State
EXAMPLE	*:LFOutput:SWEep 1* *:LFOutput:SWEep?* Return: *1\n*

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

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

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

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

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

SYNTAX	[:SOURce]:LFOutput:SWEep:STOP:FREQuency <freq> [:SOURce]:LFOutput:SWEep:STOP:FREQuency?
DESCRIPTION	This command sets/queries the stop frequency of LF sweep.

DATA TYPE	Float, unit: Hz, kHz, MHz or GHz, default is Hz
RANGE	LF wave is Sine: 0.01 Hz ~ 1 MHz, LF wave is Square/Triangle/Sawtooth: 0.01 Hz ~ 20 kHz.
RETURN	Float, unit: Hz
DEFAULT VALUE	1.5 kHz
EQUIVALENT MENU	LF > LF Sweep > Stop Freq
EXAMPLE	*:LFOutput:SWEep:STOP:FREQuency 1000* *:LFOutput:SWEep:STOP:FREQuency?* Return: *1000\n*

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

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

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

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

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

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

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

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

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

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

	Return: *POS\n*

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

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

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

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

3.4.11 System Preset

3.4.11.1 Preset (:SOURce:PRESet)

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

3.4.12 IQ Modulation

3.4.12.1 IQ Modulation Switch ([:SOURce]:FUNCtion:DM:STATe)

SYNTAX	[:SOURce]:FUNCtion:DM:STATe ON|OFF|1|0 [:SOURce]:FUNCtion:DM:STATe?
DESCRIPTION	This command sets/queries the overall switch status of IQ modulation. **Note:** When turning on the IQ modulation function, this switch must be turned on to turn on the modulation function.
DATA TYPE	Boolean
RANGE	ON|OFF|1|0
RETURN	1|0
DEFAULT VALUE	0
EQUIVALENT MENU	HOME > IQ MOD > On, or MOD ON/OFF
EXAMPLE	*:FUNCtion:DM:STATe ON* *:FUNCtion:DM:STATe?* Return: *1\n*

3.4.13 Custom

3.4.13.1 Custom State ([:SOURce]:RADio:CUSTom[:STATe])

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

3.4.13.2 Data Setup ([:SOURce]:RADio:CUSTom:DATA)

SYNTAX	[:SOURce]:RADio:CUSTom:DATA PN7|PN9|PN15|PN23|USER [:SOURce]:RADio:CUSTom:DATA?
DESCRIPTION	This command sets/queries the data pattern for unframed transmission of Custom modulation.
DATA TYPE	Enumeration
RANGE	PN7|PN9|PN15|PN23|USER

RETURN	Enumeration
DEFAULT VALUE	PN7
EQUIVALENT MENU	IQ > Custom > Data Source > Data Setup
EXAMPLE	*:RADio:CUSTom:DATA PN9* *:RADio:CUSTom:DATA?* Return: *PN9\n*

3.4.13.3 Symbol Rate ([:SOURce]:RADio:CUSTom:SRATe)

SYNTAX	[:SOURce]:RADio:CUSTom:SRATe <val> [:SOURce]:RADio:CUSTom:SRATe?
DESCRIPTION	This command sets/queries the transmission symbol rate of Custom modulation.
DATA TYPE	Float, unit: Sps
RANGE	500 Sps ~ 120 MSps
RETURN	Float, unit: Sps
DEFAULT VALUE	1 MSps
EQUIVALENT MENU	IQ > Custom > Data Source > Symbol Rate
EXAMPLE	*:RADio:CUSTom:SRATe 2000000* *:RADio:CUSTom:SRATe?* Return: *2000000\n*

3.4.13.4 Symbol Length ([:SOURce]:RADio:CUSTom:SLENgth)

SYNTAX	[:SOURce]:RADio:CUSTom:SLENgth <val> [:SOURce]:RADio:CUSTom:SLENgth?
DESCRIPTION	This command sets/queries the transmission symbol length of Custom modulation.
DATA TYPE	Integer
RANGE	100 ~ 100000
RETURN	Integer
DEFAULT VALUE	512
EQUIVALENT MENU	IQ > Custom > Data Source > Symbol Length
EXAMPLE	*:RADio:CUSTom:SLENgth 1024* *:RADio:CUSTom:SLENgth?* Return: *1024\n*

3.4.13.5 Bits/Symbol ([:SOURce]:RADio:CUSTom:SBIT?)

SYNTAX	[:SOURce]:RADio:CUSTom:SBIT?
DESCRIPTION	This command gets the transmission bits per symbol of Custom modulation. This value is determined by the modulation type.
RETURN	Integer
DEFAULT VALUE	4
EQUIVALENT MENU	IQ > Custom > Data Source > Bits/Symbol
EXAMPLE	*:RADio:CUSTom:SBIT?* Return: *4\n*

3.4.13.6 Mod Type ([:SOURce]:RADio:CUSTom:MODulation[:TYPE])

SYNTAX	[:SOURce]:RADio:CUSTom:MODulation[:TYPE] ASK2| ASK4| ASK8| ASK16| BPSK| QPSK| PSK8| DBPSK| DQPSK| DPSK8| PI4DQPSK| PI8DPSK8| OQPSK| QAM16| QAM32| QAM64| QAM128| QAM256| QAM512| FSK2| FSK4| FSK8| FSK16| MSK1| USER [:SOURce]:RADio:CUSTom:MODulation[:TYPE]?
DESCRIPTION	This command sets/queries the modulation type for the custom modulation.
DATA TYPE	Enumeration
RANGE	ASK2| ASK4| ASK8| ASK16| BPSK| QPSK| PSK8| DBPSK| DQPSK| DPSK8| PI4DQPSK| PI8DPSK8| OQPSK| QAM16| QAM32| QAM64| QAM128| QAM256| QAM512| FSK2| FSK4| FSK8| FSK16| MSK1| USER
RETURN	Enumeration
DEFAULT VALUE	QAM16
EQUIVALENT MENU	IQ > Custom > Modulation > Mod Type
EXAMPLE	*:RADio:CUSTom:MODulation ASK2* *:RADio:CUSTom:MODulation?* Return: *ASK2\n*

3.4.13.7 Gray ([:SOURce]:RADio:CUSTom:MODulation:GRAY)

SYNTAX	[:SOURce]:RADio:CUSTom:MODulation:GRAY ON|OFF|1|0 [:SOURce]:RADio:CUSTom:MODulation:GRAY?
DESCRIPTION	This command sets/queries whether the Custom modulation symbol uses Gray code encoding.
DATA TYPE	Boolean
RANGE	ON|OFF|1|0
RETURN	1|0
DEFAULT VALUE	0

EQUIVALENT MENU	IQ > Custom > Modulation > Gray
EXAMPLE	*:RADio:CUSTom:MODulation:GRAY 1* *:RADio:CUSTom:MODulation:GRAY?* Return: *1\n*

3.4.13.8 Store User-defined IQ Data ([:SOURce]:RADio:CUSTom:MODulation:STORe)

SYNTAX	[:SOURce]:RADio:CUSTom:MODulation:STORe <"file_name">
DESCRIPTION	This command saves the user-defined I/Q data to a MAP file.
DATA TYPE	String
RANGE	Please refer to the user manual for naming rules.
EQUIVALENT MENU	IQ > Custom > Modulation > Mod Type(User) > Custom > Save
EXAMPLE	*:RADio:CUSTom:MODulation:STORe "test.map"*

3.4.13.9 Load User-defined IQ Data ([:SOURce]:RADio:CUSTom:MODulation:LOAD)

SYNTAX	[:SOURce]:RADio:CUSTom:MODulation:LOAD <"file_name">
DESCRIPTION	This command loads the I/Q data from a MAP file.
DATA TYPE	String
RANGE	None
EQUIVALENT MENU	IQ > Custom > Modulation > Mod Type(User) > Custom > Load
EXAMPLE	*:RADio:CUSTom:MODulation:LOAD "test.map"*

3.4.13.10 Get User-defined IQ Data ([:SOURce]:RADio:CUSTom:MODulation:UIQ?)

SYNTAX	[:SOURce]:RADio:CUSTom:MODulation:UIQ?
DESCRIPTION	This command gets the user-defined I/Q data.
RETURN	String Format: I value Q value\nI value Q value\n…I value Q value\n\n
DEFAULT VALUE	0.632456 0.000000\n1.264911 0.000000\n\n
EQUIVALENT MENU	IQ > Custom > Modulation > Mod Type(User) > Custom
EXAMPLE	*:RADio:CUSTom:MODulation:UIQ?* Return: *0.632456 0.000000\n0.700000 -0.700000\n-0.700000 0.700000\n1.264910 0.000000\n\n*

3.4.13.11 Insert User-defined IQ Data ([:SOURce]:RADio:CUSTom:MODulation:INSert)

SYNTAX	[:SOURce]:RADio:CUSTom:MODulation:INSert <symbol>, <i data>,<q data>

DESCRIPTION	This command inserts a row into a user-defined IQ data list.
DATA TYPE	symbol: integer, i data: float, q data: float
RANGE	symbol: 0 ~ (current total number of symbols - 1), i data: -1 ~ 1, q data: -1 ~ 1
EQUIVALENT MENU	IQ > Custom > Modulation > Mod Type(User) > Custom > Insert
EXAMPLE	*:RADio:CUSTom:MODulation:INSert 0,0.5,0.5*

3.4.13.12 Edit User-defined IQ Data ([:SOURce]:RADio:CUSTom:MODulation:CHANge)

SYNTAX	[:SOURce]:RADio:CUSTom:MODulation:CHANge <symbol>, <i data>,<q data>
DESCRIPTION	This command edits the specified row in the user-defined IQ data list.
DATA TYPE	symbol: integer, i data: float, q data: float
RANGE	symbol: 0 ~ (current total number of symbols - 1), i data: -1 ~ 1, q data: -1 ~ 1
EQUIVALENT MENU	IQ > Custom > Modulation > Mod Type(User) > Custom
EXAMPLE	*:RADio:CUSTom:MODulation:CHANge 0,0.5,0.5*

3.4.13.13 Delete User-defined IQ Data ([:SOURce]:RADio:CUSTom:MODulation:DELete)

SYNTAX	[:SOURce]:RADio:CUSTom:MODulation:DELete <symbol>
DESCRIPTION	This command removes a specified row from the user-defined IQ data list.
DATA TYPE	Integer
RANGE	0 ~ (current total number of symbols - 1)
EQUIVALENT MENU	IQ > Custom > Modulation > Mod Type(User) > Custom > Delete
EXAMPLE	*:RADio:CUSTom:MODulation:DELete 0*

3.4.13.14 Clear User-defined IQ Data ([:SOURce]:RADio:CUSTom:MODulation:CLEar)

SYNTAX	[:SOURce]:RADio:CUSTom:MODulation:CLEar
DESCRIPTION	This command resets the user-defined IQ data list to default values.
EQUIVALENT MENU	IQ > Custom > Modulation > Mod Type(User) > Custom > Clear
EXAMPLE	*:RADio:CUSTom:MODulation:CLEar*

3.4.13.15 FSK Deviation ([:SOURce]:RADio:CUSTom:MODulation:FSK[:DEViation])

SYNTAX	[:SOURce]:RADio:CUSTom:MODulation:FSK[:DEViation] <val> [:SOURce]:RADio:CUSTom:MODulation:FSK[:DEViation]?
DESCRIPTION	This command sets/queries the symmetric FSK frequency deviation value.
DATA TYPE	Float, unit: Hz
RANGE	0 ~ 0.8*Symbol Rate*Oversampling
RETURN	Float, unit: Hz
DEFAULT VALUE	600000
EQUIVALENT MENU	IQ > Custom > Modulation > Mod Type(FSK) > FSK Deviation
EXAMPLE	*:RADio:CUSTom:MODulation:FSK:DEViation 450e3* *:RADio:CUSTom:MODulation:FSK:DEViation?* Return: *450000\n*

3.4.13.16 Filter Type ([:SOURce]:RADio:CUSTom:FILTer)

SYNTAX	[:SOURce]:RADio:CUSTom:FILTer NONE | RCOSine | RRCosine | GAUSsian | HSINe [:SOURce]:RADio:CUSTom:FILTer?
DESCRIPTION	This command sets/queries the filter type of the Custom modulation.
DATA TYPE	Enumeration
RANGE	NONE | RCOSine | RRCosine | GAUSsian | HSINe
RETURN	Enumeration
DEFAULT VALUE	RRCosine
EQUIVALENT MENU	IQ > Custom > Filter > Filter Type
EXAMPLE	*:RADio:CUSTom:FILTer GAUSsian* *:RADio:CUSTom:FILTer?* Return: *GAUSsian\n*

3.4.13.17 Filter Alpha/BT ([:SOURce]:RADio:CUSTom:ALPHa)

SYNTAX	[:SOURce]:RADio:CUSTom:ALPHa <val> [:SOURce]:RADio:CUSTom:ALPHa?
DESCRIPTION	This command sets/queries the alpha value of a Nyquist or root Nyquist filter or the BT value of a Gaussian filter.
DATA TYPE	Float
RANGE	0.010 ~ 1.000
RETURN	Float
DEFAULT VALUE	0.350

EQUIVALENT MENU	IQ > Custom > Filter > Filter Alpha/BT
EXAMPLE	*:RADio:CUSTom:ALPHa 0.22* *:RADio:CUSTom:ALPHa?* Return: *0.22\n*

3.4.13.18 Filter Length ([:SOURce]:RADio:CUSTom:FILTer:LENgth)

SYNTAX	[:SOURce]:RADio:CUSTom:FILTer:LENgth <length> [:SOURce]:RADio:CUSTom:FILTer:LENgth?
DESCRIPTION	This command sets/queries the filter length of Custom modulation.
DATA TYPE	Integer
RANGE	1 ~ 512
RETURN	Integer
DEFAULT VALUE	128
EQUIVALENT MENU	IQ > Custom > Filter > Filter Length
EXAMPLE	*:RADio:CUSTom:FILTer:LENgth 64* *:RADio:CUSTom:FILTer:LENgth?* Return: *64\n*

3.4.13.19 OverSampling ([:SOURce]:RADio:CUSTom:FILTer:OSAMple)

SYNTAX	[:SOURce]:RADio:CUSTom:FILTer:OSAMple <val> [:SOURce]:RADio:CUSTom:FILTer:OSAMple?
DESCRIPTION	This command sets/queries the oversampling value of the Custom modulation filter.
DATA TYPE	Integer
RANGE	2 ~ 32
RETURN	Integer
DEFAULT VALUE	2
EQUIVALENT MENU	IQ > Custom > Filter > OverSampling
EXAMPLE	*:RADio:CUSTom:FILTer:OSAMple 4* *:RADio:CUSTom:FILTer:OSAMple?* Return: *4\n*

3.4.13.20 Bit Rate ([:SOURce]:RADio:CUSTom:BRATe?)

SYNTAX	[:SOURce]:RADio:CUSTom:BRATe?
DESCRIPTION	This command queries the bit rate of Custom modulation in bits per second.

RETURN	Float, unit: bps
DEFAULT VALUE	4 MSps
EQUIVALENT MENU	None
EXAMPLE	*:RADio:CUSTom:BRATe?* Return: *4000000\n*

3.4.13.21 Save Waveform ([:SOURce]:RADio:CUSTom:SAVE)

SYNTAX	[:SOURce]:RADio:CUSTom:SAVE <"file_name">
DESCRIPTION	This command saves Custom modulated waveform data to an ARB file.
DATA TYPE	String
RANGE	Please refer to the user manual for naming rules.
EQUIVALENT MENU	IQ > Custom > Save Waveform
EXAMPLE	*:RADio:CUSTom:SAVE "test.arb"*

3.4.13.22 Update ([:SOURce]:RADio:CUSTom:DOWNload)

SYNTAX	[:SOURce]:RADio:CUSTom:DOWNload
DESCRIPTION	This command updates the settings for Custom modulation.
EQUIVALENT MENU	IQ > Custom > Update
EXAMPLE	*:RADio:CUSTom:DOWNload*

3.4.14 ARB

3.4.14.1 ARB State ([:SOURce]:RADio:ARB[:STATe])

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

3.4.14.2 Select Waveform ([:SOURce]:RADio:ARB:WAVeform)

SYNTAX	[:SOURce]:RADio:ARB:WAVeform <"WFM:segment"|"SEQ:sequence"> [:SOURce]:RADio:ARB:WAVeform?
DESCRIPTION	This command sets/queries the waveform segment or waveform sequence currently played by ARB.
DATA TYPE	String
RANGE	"WFM:segment": waveform name in IQ > ARB > Waveform Segment, "SEQ:sequence": waveform name in IQ > ARB > Waveform Sequence.
RETURN	String
DEFAULT VALUE	*NONE
EQUIVALENT MENU	IQ > ARB > Select Waveform
EXAMPLE	*:RADio:ARB:WAVeform "WFM:SINE_WAVE"* *:RADio:ARB:WAVeform?* Return: *WFM:SINE_WAVE\n* *:RADio:ARB:WAVeform "SEQ:test_seq"* *:RADio:ARB:WAVeform?* Return: *SEQ:test_seq\n*

3.4.14.3 Get Segment List ([:SOURce]:IQ:DUALarb:SEGMent:NAMEs?)

SYNTAX	[:SOURce]:IQ:DUALarb:SEGMent:NAMEs?
DESCRIPTION	This command queries the list of waveform segments in ARB.
RETURN	String Format: segment_name sample_points\nsegment_name sample_points\n... segment_name sample_points\n
DEFAULT VALUE	RAMP_WAVE 200\nSINE_WAVE 200\n
EQUIVALENT MENU	IQ > ARB > Waveform Segment
EXAMPLE	*:IQ:DUALarb:SEGMent:NAMEs?* Return: *RAMP_WAVE 200\nSINE_WAVE 200\n*

3.4.14.4 Load Segment ([:SOURce]:IQ:DUALarb:SEGMent:LOAD)

SYNTAX	[:SOURce]:IQ:DUALarb:SEGMent:LOAD <"file_name">
DESCRIPTION	This command loads a waveform segment file in the ARB segment list. The file types that can be loaded are: *.arb, *.wdbin, *.txt.
DATA TYPE	String
RANGE	None

EQUIVALENT SCPI	[:SOURce]:IQ:DUALarb:SEGMent:LOAD:DATA <"file_name"> [:SOURce]:IQ:DUALarb:SEGMent:LOAD:TEXT <"file_name">
EQUIVALENT MENU	IQ > ARB > Waveform Segment > Load
EXAMPLE	*:IQ:DUALarb:SEGMent:LOAD "Local/test.arb"* *:IQ:DUALarb:SEGMent:LOAD "Local/sine.wdbin"* *:IQ:DUALarb:SEGMent:LOAD "Local/16qam.txt"*

3.4.14.5 Load Segment ([:SOURce]:IQ:DUALarb:SEGMent:LOAD:DATA)

SYNTAX	[:SOURce]:IQ:DUALarb:SEGMent:LOAD:DATA <"file_name">
DESCRIPTION	This command loads a waveform segment file in the ARB segment list. The file types that can be loaded are: *.arb, *.wdbin, *.txt.
DATA TYPE	String
RANGE	None
EQUIVALENT SCPI	[:SOURce]:IQ:DUALarb:SEGMent:LOAD <"file_name"> [:SOURce]:IQ:DUALarb:SEGMent:LOAD:TEXT <"file_name">
EQUIVALENT MENU	IQ > ARB > Waveform Segment > Load
EXAMPLE	*:IQ:DUALarb:SEGMent:LOAD:DATA "Local/test.arb"* *:IQ:DUALarb:SEGMent:LOAD:DATA "Local/sine.wdbin"* *:IQ:DUALarb:SEGMent:LOAD:DATA "Local/16qam.txt"*

3.4.14.6 Load Segment ([:SOURce]:IQ:DUALarb:SEGMent:LOAD:TEXT)

SYNTAX	[:SOURce]:IQ:DUALarb:SEGMent:LOAD:TEXT <"file_name">
DESCRIPTION	This command loads a waveform segment file in the ARB segment list. The file types that can be loaded are: *.arb, *.wdbin, *.txt.
DATA TYPE	String
RANGE	None
EQUIVALENT SCPI	[:SOURce]:IQ:DUALarb:SEGMent:LOAD <"file_name"> [:SOURce]:IQ:DUALarb:SEGMent:LOAD:DATA <"file_name">
EQUIVALENT MENU	IQ > ARB > Waveform Segment > Load
EXAMPLE	*:IQ:DUALarb:SEGMent:LOAD:TEXT "Local/test.arb"* *:IQ:DUALarb:SEGMent:LOAD:TEXT "Local/sine.wdbin"* *:IQ:DUALarb:SEGMent:LOAD:TEXT "Local/16qam.txt"*

3.4.14.7 Delete Segment ([:SOURce]:IQ:DUALarb:SEGMent:DEL)

SYNTAX	[:SOURce]:IQ:DUALarb:SEGMent:DEL <"segment">
DESCRIPTION	This command deletes a waveform segment from the waveform segment list.
DATA TYPE	String
RANGE	Waveform name in IQ > ARB > Waveform Segment
EQUIVALENT MENU	IQ > ARB > Waveform Segment > Delete

EXAMPLE	*:IQ:DUALarb:SEGMent:DEL "sine"*

3.4.14.8 Rename Segment ([:SOURce]:IQ:DUALarb:SEGMent:REName)

SYNTAX	[:SOURce]:IQ:DUALarb:SEGMent:REName <"original_name">,<"new_name">
DESCRIPTION	This command renames the name of a waveform segment in the ARB segment list.
DATA TYPE	String, string
RANGE	original_name: waveform name in IQ > ARB > Waveform Segment, new_name: Please refer to the user manual for naming rules.
EQUIVALENT MENU	IQ > ARB > Waveform Segment > Rename
EXAMPLE	*:IQ:DUALarb:SEGMent:REName "test","Rename_Wave"*

3.4.14.9 Clear Segment ([:SOURce]:IQ:DUALarb:SEGMent:CLEar)

SYNTAX	[:SOURce]:IQ:DUALarb:SEGMent:CLEar
DESCRIPTION	This command clears the ARB segment list. **Note:** The waveform segment being played will be retained.
EQUIVALENT MENU	IQ > ARB > Waveform Segment > Clear
EXAMPLE	*:IQ:DUALarb:SEGMent:CLEar*

3.4.14.10 Create Sequence ([:SOURce]:RADio:ARB:SEQuence)

SYNTAX	[:SOURce]:RADio:ARB:SEQuence <"file_name">,<"waveform1">,<reps>,<marker>,{<"waveform2">, <reps>,<marker>, ...} [:SOURce]:RADio:ARB:SEQuence? <"file_name">
DESCRIPTION	This command creates a waveform sequence composed of segments and other sequences. Segments and sequences play in the same order as the commands are placed in the waveform sequence. The query command returns the contents of the waveform sequence.
DATA TYPE	file_name: string, the name of the waveform sequence to be created, waveform: string, waveform segment or sequence name, reps: integer, number of waveform repetitions, marker: enumeration, identification point of the waveform.
RANGE	file_name: please refer to the user manual for naming rules, waveform: waveform name in IQ > ARB > Waveform Segment or IQ > ARB > Waveform Sequence, reps: 1 ~ 65535, marker: NONE| M1| M2| M3| M4| M1M2| M1M3| M1M4| M2M3| M2M4| M3M4| M1M2M3| M1M2M4| M1M3M4| M2M3M4| ALL
RETURN	String
DEFAULT VALUE	None

EQUIVALENT MENU	IQ > ARB > Waveform Sequence > Build
EXAMPLE	*:RADio:ARB:SEQuence "SEQ:Test_Data","WFM:ramp_wave",25,M1M4,"WFM:sine_wave",100,ALL,"SEQ:seq1",3,NONE* *:RADio:ARB:SEQuence? "SEQ:Test_Data"* Return: *"WFM:ramp_wave",25,M1M4,"WFM:sine_wave",100,ALL,"SEQ:seq1",3,NONE\n*

3.4.14.11 Sample Clock ([:SOURce]:RADio:ARB:SCLock:RATE)

SYNTAX	[:SOURce]:RADio:ARB:SCLock:RATE <rate> [:SOURce]:RADio:ARB:SCLock:RATE?
DESCRIPTION	This command sets/queries the sample clock rate of ARB modulation.
DATA TYPE	Float, unit: Hz, kHz, MHz or GHz, default is Hz
RANGE	0.002 Hz ~ 240 MHz
RETURN	Float, unit: Hz
DEFAULT VALUE	2 MHz
EQUIVALENT MENU	IQ > ARB > ARB Setup > Sample Clock
EXAMPLE	*:RADio:ARB:SCLock:RATE 4 MHz* *:RADio:ARB:SCLock:RATE?* Return: *4000000\n*

3.4.14.12 Modulator Atten Type ([:SOURce]:RADio:ARB:IQ:MODulation:ATTen:AUTO)

SYNTAX	[:SOURce]:RADio:ARB:IQ:MODulation:ATTen:AUTO AUTO|MANUal [:SOURce]:RADio:ARB:IQ:MODulation:ATTen:AUTO?
DESCRIPTION	This command sets/queries the attenuation type of the ARB modulator.
DATA TYPE	Enumeration
RANGE	AUTO|MANUal
RETURN	Enumeration
DEFAULT VALUE	AUTO
EQUIVALENT MENU	IQ > ARB > ARB Setup > Modulator Atten Type
EXAMPLE	*:RADio:ARB:IQ:MODulation:ATTen:AUTO AUTO* *:RADio:ARB:IQ:MODulation:ATTen:AUTO?* Return: *AUTO\n*

3.4.14.13 Modulation Atten ([:SOURce]:RADio:ARB:IQ:MODulation:ATTen)

SYNTAX	[:SOURce]:RADio:ARB:IQ:MODulation:ATTen <val> [:SOURce]:RADio:ARB:IQ:MODulation:ATTen?

DESCRIPTION	This command sets/queries the attenuation value of the ARB modulator.
DATA TYPE	Float, unit: dB
RANGE	0 ~ 20
RETURN	Float, unit: dB
DEFAULT VALUE	3
EQUIVALENT MENU	IQ > ARB > ARB Setup > Modulation Atten
EXAMPLE	*:RADio:ARB:IQ:MODulation:ATTen 10* *:RADio:ARB:IQ:MODulation:ATTen?* Return: *10\n*

3.4.14.14 Real Time AWGN State ([:SOURce]:RADio:ARB:NOISe[:STATe])

SYNTAX	[:SOURce]:RADio:ARB:NOISe[:STATe] ON|OFF|1|0 [:SOURce]:RADio:ARB:NOISe[:STATe]?
DESCRIPTION	This command sets/queries the real-time AWGN switch status of ARB.
DATA TYPE	Boolean
RANGE	ON|OFF|1|0
RETURN	1|0
DEFAULT VALUE	0
EQUIVALENT MENU	IQ > ARB > ARB Setup > Real Time AWGN
EXAMPLE	*:RADio:ARB:NOISe 1* *:RADio:ARB:NOISe?* Return: *1\n*

3.4.14.15 Output Mux ([:SOURce]:RADio:ARB:NOISe:OUTPut)

SYNTAX	[:SOURce]:RADio:ARB:NOISe:OUTPut CARRier|NOISe|CN [:SOURce]:RADio:ARB:NOISe:OUTPut?
DESCRIPTION	This command sets/queries the output type of ARB real-time AWGN.
DATA TYPE	Enumeration
RANGE	CARRier|NOISe|CN
RETURN	Enumeration
DEFAULT VALUE	CN
EQUIVALENT MENU	IQ > ARB > ARB Setup > Real Time AWGN > Output Mux
EXAMPLE	*:RADio:ARB:NOISe:OUTPut CARRier* *:RADio:ARB:NOISe:OUTPut?* Return: *CARRier\n*

3.4.14.16 Power Control ([:SOURce]:RADio:ARB:NOISe:POWer:TYPE)

SYNTAX	[:SOURce]:RADio:ARB:NOISe:POWer:TYPE CARRier|CHNO|TONO|TOPO [:SOURce]:RADio:ARB:NOISe:POWer:TYPE?
DESCRIPTION	This command sets/queries the power control mode of ARB real-time AWGN.
DATA TYPE	Enumeration
RANGE	CARRier|CHNO|TONO|TOPO
RETURN	Enumeration
DEFAULT VALUE	TOPO
EQUIVALENT MENU	IQ > ARB > ARB Setup > Real Time AWGN > Power Control
EXAMPLE	*:RADio:ARB:NOISe:POWer:TYPE CARRier* *:RADio:ARB:NOISe:POWer:TYPE?* Return: *CARRier\n*

3.4.14.17 Total Power ([:SOURce]:RADio:ARB:NOISe:POWer:TOTal)

SYNTAX	[:SOURce]:RADio:ARB:NOISe:POWer:TOTal <power> [:SOURce]:RADio:ARB:NOISe:POWer:TOTal?
DESCRIPTION	This command sets/queries the total power value of ARB real-time AWGN.
DATA TYPE	Float, unit: dBm
RANGE	-140 ~ 10
RETURN	Float, unit: dBm
DEFAULT VALUE	0
EQUIVALENT MENU	IQ > ARB > ARB Setup > Real Time AWGN > Total Power
EXAMPLE	*:RADio:ARB:NOISe:POWer:TOTal 0 dBm* *:RADio:ARB:NOISe:POWer:TOTal?* Return: *0\n*

3.4.14.18 Carrier Power ([:SOURce]:RADio:ARB:NOISe:POWer:CARRier)

SYNTAX	[:SOURce]:RADio:ARB:NOISe:POWer:CARRier <power> [:SOURce]:RADio:ARB:NOISe:POWer:CARRier?
DESCRIPTION	This command sets/queries the carrier power value of ARB real-time AWGN.
DATA TYPE	Float, unit: dBm
RANGE	Related to the total power value of real-time AWGN
RETURN	Float, unit: dBm
DEFAULT VALUE	-3.27

EQUIVALENT MENU	IQ > ARB > ARB Setup > Real Time AWGN > Carrier Power
EXAMPLE	*:RADio:ARB:NOISe:POWer:CARRier 0 dBm* *:RADio:ARB:NOISe:POWer:CARRier?* Return: *0\n*

3.4.14.19 Total Noise Power ([:SOURce]:RADio:ARB:NOISe:POWer:TONOise)

SYNTAX	[:SOURce]:RADio:ARB:NOISe:POWer:TONOise <power> [:SOURce]:RADio:ARB:NOISe:POWer:TONOise?
DESCRIPTION	This command sets/queries the total noise power value of ARB real-time AWGN.
DATA TYPE	Float, unit: dBm
RANGE	Related to the total power value of real-time AWGN
RETURN	Float, unit: dBm
DEFAULT VALUE	-3.27
EQUIVALENT MENU	IQ > ARB > ARB Setup > Real Time AWGN > Total Noise Power
EXAMPLE	*:RADio:ARB:NOISe:POWer:TONOise 0 dBm* *:RADio:ARB:NOISe:POWer:TONOise?* Return: *0\n*

3.4.14.20 Channel Noise Power ([:SOURce]:RADio:ARB:NOISe:POWer:CHNOise)

SYNTAX	[:SOURce]:RADio:ARB:NOISe:POWer:CHNOise <power> [:SOURce]:RADio:ARB:NOISe:POWer:CHNOise?
DESCRIPTION	This command sets/queries the channel noise power value of ARB real-time AWGN.
DATA TYPE	Float, unit: dBm
RANGE	Related to the total power value of real-time AWGN
RETURN	Float, unit: dBm
DEFAULT VALUE	-3.27
EQUIVALENT MENU	IQ > ARB > ARB Setup > Real Time AWGN > Channel Noise Power
EXAMPLE	*:RADio:ARB:NOISe:POWer:CHNOise 0 dBm* *:RADio:ARB:NOISe:POWer:CHNOise?* Return: *0\n*

3.4.14.21 Carrier To Noise Ratio Format ([:SOURce]:RADio:ARB:NOISe:CN:FORMat)

SYNTAX	[:SOURce]:RADio:ARB:NOISe:CN:FORMat CARRier|BIT [:SOURce]:RADio:ARB:NOISe:CN:FORMat?
DESCRIPTION	This command sets/queries the carrier-to-noise ratio format of ARB

	real-time AWGN.
DATA TYPE	Enumeration
RANGE	CARRier|BIT
RETURN	Enumeration
DEFAULT VALUE	CARRier
EQUIVALENT MENU	IQ > ARB > ARB Setup > Real Time AWGN > Carrier To Noise Ratio Format
EXAMPLE	*:RADio:ARB:NOISe:CN:FORMat BIT* *:RADio:ARB:NOISe:CN:FORMat?* Return: *BIT\n*

3.4.14.22 Carrier To Noise Ratio ([:SOURce]:RADio:ARB:NOISe:CN)

SYNTAX	[:SOURce]:RADio:ARB:NOISe:CN <val> [:SOURce]:RADio:ARB:NOISe:CN?
DESCRIPTION	This command sets/queries the carrier-to-noise ratio of the ARB real-time AWGN.
DATA TYPE	Float, unit: dB
RANGE	-100 ~ 100
RETURN	Float, unit: dB
DEFAULT VALUE	0
EQUIVALENT MENU	IQ > ARB > ARB Setup > Real Time AWGN > Carrier To Noise Ratio
EXAMPLE	*:RADio:ARB:NOISe:CN -5* *:RADio:ARB:NOISe:CN?* Return: *-5\n*

3.4.14.23 Bit To Noise Ratio ([:SOURce]:RADio:ARB:NOISe:CBNO)

SYNTAX	[:SOURce]:RADio:ARB:NOISe:CBNO <val> [:SOURce]:RADio:ARB:NOISe:CBNO?
DESCRIPTION	This command sets/queries the bit signal-to-noise ratio of ARB real-time AWGN.
DATA TYPE	Float, unit: dB
RANGE	Relevant to the values of carrier signal-to-noise ratio and carrier bit rate.
RETURN	Float, unit: dB
DEFAULT VALUE	0
EQUIVALENT MENU	IQ > ARB > ARB Setup > Real Time AWGN > Eb/No
EXAMPLE	*:RADio:ARB:NOISe:CBNO -5* *:RADio:ARB:NOISe:CBNO?*

	Return: *-5\n*

3.4.14.24 Carrier Bit Rate ([:SOURce]:RADio:ARB:NOISe:BRATe)

SYNTAX	[:SOURce]:RADio:ARB:NOISe:BRATe <rate> [:SOURce]:RADio:ARB:NOISe:BRATe?
DESCRIPTION	This command sets/queries the carrier bit rate of ARB real-time AWGN.
DATA TYPE	Float, unit: Sps
RANGE	1 ~ 10*Carrier Bandwidth
RETURN	Float, unit: Sps
DEFAULT VALUE	1
EQUIVALENT MENU	IQ > ARB > ARB Setup > Real Time AWGN > Carrier Bit Rate
EXAMPLE	*:RADio:ARB:NOISe:BRATe 5* *:RADio:ARB:NOISe:BRATe?* Return: *5\n*

3.4.14.25 Carrier Bandwidth ([:SOURce]:RADio:ARB:NOISe:CBWidth)

SYNTAX	[:SOURce]:RADio:ARB:NOISe:CBWidth <bandwidth> [:SOURce]:RADio:ARB:NOISe:CBWidth?
DESCRIPTION	This command sets/queries the carrier bandwidth of ARB real-time AWGN.
DATA TYPE	Float, unit: Hz
RANGE	1 Hz ~ 120 MHz
RETURN	Float, unit: Hz
DEFAULT VALUE	1 Hz
EQUIVALENT MENU	IQ > ARB > ARB Setup > Real Time AWGN > Carrier Bandwidth
EXAMPLE	*:RADio:ARB:NOISe:CBWidth 5000000* *:RADio:ARB:NOISe:CBWidth?* Return: *5000000\n*

3.4.14.26 Flat Noise Bandwidth ([:SOURce]:RADio:ARB:NOISe:NBWidth)

SYNTAX	[:SOURce]:RADio:ARB:NOISe:NBWidth <bandwidth> [:SOURce]:RADio:ARB:NOISe:NBWidth?
DESCRIPTION	This command sets/queries the flat noise bandwidth of ARB real-time AWGN.
DATA TYPE	Float, unit: Hz
RANGE	Carrier Bandwidth ~ 120 MHz

RETURN	Float, unit: Hz
DEFAULT VALUE	1 Hz
EQUIVALENT MENU	IQ > ARB > ARB Setup > Real Time AWGN > Flat Noise Bandwidth
EXAMPLE	*:RADio:ARB:NOISe:NBWidth 5000000* *:RADio:ARB:NOISe:NBWidth?* Return: *5000000\n*

3.4.14.27 ARB Filter Type ([:SOURce]:IQ:DUALarb:FILTer:TYPE)

SYNTAX	[:SOURce]:IQ:DUALarb:FILTer:TYPE NONE | RCOSine | RRCosine | GAUSsian | HSINe [:SOURce]:IQ:DUALarb:FILTer:TYPE?
DESCRIPTION	This command sets/queries the filter type of ARB modulation.
DATA TYPE	Enumeration
RANGE	NONE | RCOSine | RRCosine | GAUSsian | HSINe
RETURN	Enumeration
DEFAULT VALUE	NONE
EQUIVALENT MENU	IQ > ARB > ARB Setup > Modulation Filter > Filter Type
EXAMPLE	*:IQ:DUALarb:FILTer:TYPE GAUSsian* *:IQ:DUALarb:FILTer:TYPE?* Return: *GAUSsian\n*

3.4.14.28 ARB Filter Alpha/BT ([:SOURce]:IQ:DUALarb:FILTer:ALPHa)

SYNTAX	[:SOURce]:IQ:DUALarb:FILTer:ALPHa <val> [:SOURce]:IQ:DUALarb:FILTer:ALPHa?
DESCRIPTION	This command sets/queries the alpha value of a Nyquist or root Nyquist filter or the BT value of a Gaussian filter of the ARB modulation.
DATA TYPE	Float
RANGE	0.010 ~ 1.000
RETURN	Float
DEFAULT VALUE	0.500
EQUIVALENT MENU	IQ > ARB > ARB Setup > Modulation Filter > Filter Alpha/BT
EXAMPLE	*:IQ:DUALarb:FILTer:ALPHa 0.22* *:IQ:DUALarb:FILTer:ALPHa?* Return: *0.22\n*

3.4.14.29 ARB Filter Length ([:SOURce]:IQ:DUALarb:FILTer:LENgth)

SYNTAX	[:SOURce]:IQ:DUALarb:FILTer:LENgth <len>

	[:SOURce]:IQ:DUALarb:FILTer:LENgth?
DESCRIPTION	This command sets/queries the filter length of ARB modulation.
DATA TYPE	Integer
RANGE	1 ~ 512
RETURN	Integer
DEFAULT VALUE	32
EQUIVALENT MENU	IQ > ARB > ARB Setup > Modulation Filter > Filter Length
EXAMPLE	*:IQ:DUALarb:FILTer:LENgth 64* *:IQ:DUALarb:FILTer:LENgth?* Return: *64\n*

3.4.14.30 ARB Filter OverSampling ([:SOURce]:IQ:DUALarb:OSAMple)

SYNTAX	[:SOURce]:IQ:DUALarb:OSAMple <val> [:SOURce]:IQ:DUALarb:OSAMple?
DESCRIPTION	This command sets/queries the oversampling value of the ARB modulation filter.
DATA TYPE	Integer
RANGE	2 ~ 32
RETURN	Integer
DEFAULT VALUE	2
EQUIVALENT MENU	IQ > ARB > ARB Setup > Modulation Filter > OverSampling
EXAMPLE	*:IQ:DUALarb:OSAMple 4* *:IQ:DUALarb:OSAMple?* Return: *4\n*

3.4.14.31 ARB Filter Update ([:SOURce]:IQ:DUALarb:FILTer:UPDate)

SYNTAX	[:SOURce]:IQ:DUALarb:FILTer:UPDate
DESCRIPTION	This command updates the settings of the ARB filter.
EQUIVALENT MENU	IQ > ARB > ARB Setup > Modulation Filter > Update
EXAMPLE	*:IQ:DUALarb:FILTer:UPDate*

3.4.14.32 Baseband Offset State ([:SOURce]:RADio:ARB:OFFSet:STATe)

SYNTAX	[:SOURce]:RADio:ARB:OFFSet:STATe ON|OFF|1|0 [:SOURce]:RADio:ARB:OFFSet:STATe?
DESCRIPTION	This command sets/queries the switch status of ARB baseband frequency offset.

DATA TYPE	Boolean
RANGE	ON|OFF|1|0
RETURN	1|0
DEFAULT VALUE	0
EQUIVALENT MENU	IQ > ARB > ARB Setup > Baseband Offset
EXAMPLE	*:RADio:ARB:OFFSet:STATe 1* *:RADio:ARB:OFFSet:STATe?* Return: *1\n*

3.4.14.33 Baseband Offset Freq ([:SOURce]:RADio:ARB:OFFSet:FREQuency)

SYNTAX	[:SOURce]:RADio:ARB:OFFSet:FREQuency <freq> [:SOURce]:RADio:ARB:OFFSet:FREQuency?
DESCRIPTION	This command sets/queries the baseband offset frequency of ARB modulation.
DATA TYPE	Float, unit: Hz
RANGE	0 Hz ~ 60 MHz
RETURN	Float, unit: Hz
DEFAULT VALUE	0 Hz
EQUIVALENT MENU	IQ > ARB > ARB Setup > Baseband Offset(On) > Offset Freq
EXAMPLE	*:RADio:ARB:OFFSet:FREQuency -1000000* *:RADio:ARB:OFFSet:FREQuency?* Return: *-1000000\n*

3.4.14.34 Set Active Marker ([:SOURce]:IQ:DUALarb:MARKer)

SYNTAX	[:SOURce]:IQ:DUALarb:MARKer <marker> [:SOURce]:IQ:DUALarb:MARKer?
DESCRIPTION	This command selects/queries the active marker to edit.
DATA TYPE	Integer
RANGE	1 ~ 4
EQUIVALENT MENU	IQ > ARB > Marker Utilities > Marker Number
EXAMPLE	*:IQ:DUALarb:MARKer 2* *:IQ:DUALarb:MARKer?* Return: *2\n*

3.4.14.35 Set Markers ([:SOURce]:RADio:ARB:MARKer[:SET])

SYNTAX	[:SOURce]:RADio:ARB:MARKer[:SET] <“segment”>,<index>,<first_point>,<last_point>,<skip_count>

DESCRIPTION	This command sets the identification point of an ARB waveform segment.
DATA TYPE	segment: string, waveform segment name, index: integer, first_point: integer, marking the first identification point of the waveform segment, last_point: integer, marking the last identification point of the waveform segment, skip_count: integer, marking the interval between identification points of the waveform segment.
RANGE	segment: waveform segment in the ARB segment list, index: 1 ~ 1024, first_point: 1 ~ the number of points in the waveform segment, last_point: <first_point> ~ the number of points in the waveform segment, skip_count: 0 ~ (<last_point> - <first_point>)
EQUIVALENT MENU	IQ > ARB > Marker Utilities > Set Markers
EXAMPLE	*:RADio:ARB:MARKer:CLEar:ALL “RAMP_WAVE”,2* *:IQ:DUALarb:MARKer 2* *:RADio:ARB:MARKer "RAMP_WAVE",1,10,20,5* *:RADio:ARB:MARKer "RAMP_WAVE",2,100,150,0* At this time, the identifier of RAMP_WAVE is: 10,20,5\n100,150,0

3.4.14.36 Clear Marker ([:SOURce]:RADio:ARB:MARKer:CLEar:ALL)

SYNTAX	[:SOURce]:RADio:ARB:MARKer:CLEar:ALL <“segment”>,<marker>
DESCRIPTION	This command clears the identification of the specified identification point of the waveform segment.
DATA TYPE	segment: string, waveform segment name, marker: integer, identification point.
RANGE	segment: waveform segment in the ARB segment list, marker: 1 ~ 4.
EQUIVALENT MENU	IQ > ARB > Marker Utilities > Set Markers > Clear
EXAMPLE	*:RADio:ARB:MARKer:CLEar:ALL “SINE_WAVE”,1*

3.4.14.37 Marker Polarity ([:SOURce]:RADio:ARB:MPOLarity:MARKer1|2|3|4)

SYNTAX	[:SOURce]:RADio:ARB:MPOLarity:MARKer1|2|3|4 NEG|POS [:SOURce]:RADio:ARB:MPOLarity:MARKer1|2|3|4?
DESCRIPTION	This command sets/queries the polarity of the specified identification point.
DATA TYPE	Enumeration
RANGE	NEG|POS
RETURN	Enumeration
DEFAULT VALUE	NEG

EQUIVALENT MENU	IQ > ARB > Marker Utilities > Marker Polarity
EXAMPLE	*:RADio:ARB:MPOLarity:MARKer1 NEG* *:RADio:ARB:MPOLarity:MARKer1?* Return: *NEG\n*

3.4.14.38 Marker Output ([:SOURce]:RADio:ARB:MARKer:OUTPut)

SYNTAX	[:SOURce]:RADio:ARB:MARKer:OUTPut NONE|M1|M2|M3|M4 [:SOURce]:RADio:ARB:MARKer:OUTPut?
DESCRIPTION	This command sets/queries the output identification point of the ARB waveform segment.
DATA TYPE	Enumeration
RANGE	NONE|M1|M2|M3|M4
RETURN	Enumeration
DEFAULT VALUE	M1
EQUIVALENT MENU	IQ > ARB > Marker Utilities > Marker Output
EXAMPLE	*:RADio:ARB:MARKer:OUTPut M2* *:RADio:ARB:MARKer:OUTPut?* Return: *M2\n*

3.4.14.39 Marker Delay ([:SOURce]:IQ:DUALarb:MARKer:DELay)

SYNTAX	[:SOURce]:IQ:DUALarb:MARKer:DELay <time> [:SOURce]:IQ:DUALarb:MARKer:DELay?
DESCRIPTION	This command sets/queries the identification delay time of the waveform segment.
DATA TYPE	Float, unit: ns, us, ms or s, default is s
RANGE	-4 us ~ 860 us
RETURN	Float, unit: s
DEFAULT VALUE	0
EQUIVALENT MENU	IQ > ARB > Marker Utilities > Marker Delay
EXAMPLE	*:IQ:DUALarb:MARKer:DELay 20 us* *:IQ:DUALarb:MARKer:DELay?* Return: *2e-05\n*

3.4.14.40 Pulse/RF Blank ([:SOURce]:RADio:ARB:MDEStination:PULSe)

SYNTAX	[:SOURce]:RADio:ARB:MDEStination:PULSe NONE|M1|M2|M3|M4 [:SOURce]:RADio:ARB:MDEStination:PULSe?
DESCRIPTION	This command sets/queries the marker pulse/RF blanking function of the waveform segment.

DATA TYPE	Enumeration
RANGE	NONE|M1|M2|M3|M4
RETURN	Enumeration
DEFAULT VALUE	NONE
EQUIVALENT MENU	IQ > ARB > Marker Utilities > Pulse/RF Blank
EXAMPLE	*:RADio:ARB:MDEStination:PULSe M2* *:RADio:ARB:MDEStination:PULSe?* Return: *M2\n*

3.4.14.41 Clipping ([:SOURce]:RADio:ARB:CLIPping)

SYNTAX	[:SOURce]:RADio:ARB:CLIPping <“segment”>,IJQ|IORQ,<val>[,<val>]
DESCRIPTION	This command sets the clipping level of the selected waveform segment to a percentage of its highest peak.
DATA TYPE	segment: string, waveform segment name, IJQ|IORQ: enumeration, reduction type, |I+jQ| or |I|,|Q|, val: float, reduction coefficient.
RANGE	segment: waveform segment in the ARB segment list, IJQ|IORQ: IJQ represents |I+jQ|, IORQ represents |I|, |Q|, val: 0.01~1.
EQUIVALENT MENU	IQ > ARB > Waveform Utilities > Select Segment & Clip |I+jQ| to / Clip |I| to / Clip |Q| to
EXAMPLE	*:RADio:ARB:CLIPping "SINE_WAVE",IJQ,0.75* *:RADio:ARB:CLIPping "RAMP_WAVE",IORQ,0.75,0.8*

3.4.14.42 Scaling ([:SOURce]:RADio:ARB:SCALing)

SYNTAX	[:SOURce]:RADio:ARB:SCALing <“segment”>,<val>
DESCRIPTION	This command scales the specified waveform segment.
DATA TYPE	segment: string, waveform segment name, val: float, scaling factor.
RANGE	segment: waveform segment in the ARB segment list, val: 0.01~1.
EQUIVALENT MENU	IQ > ARB > Waveform Utilities > Select Segment & Scaling
EXAMPLE	*:RADio:ARB:Scaling "RAMP_WAVE",0.75*

3.4.14.43 ARB Trigger Type ([:SOURce]:IQ:DUALarb:TRIGger:TYPE)

SYNTAX	[:SOURce]:IQ:DUALarb:TRIGger:TYPE CONTinous| SINGle| SADVance| GATE [:SOURce]:IQ:DUALarb:TRIGger:TYPE?
DESCRIPTION	This command sets/queries the trigger type of ARB.

DATA TYPE	Enumeration
RANGE	CONTinous| SINGle| SADVance| GATE
RETURN	Enumeration
DEFAULT VALUE	CONTinous
EQUIVALENT MENU	IQ > ARB > Trigger > Trigger Type
EXAMPLE	*:IQ:DUALarb:TRIGger:TYPE SINGle* *:IQ:DUALarb:TRIGger:TYPE?* Return: *SINGle\n*

3.4.14.44 ARB Trigger Continuous Mode ([:SOURce]:IQ:DUALarb:TRIGger:CONTinous)

SYNTAX	[:SOURce]:IQ:DUALarb:TRIGger:CONTinous FREErun | RUNIgnored | RUNRestart [:SOURce]:IQ:DUALarb:TRIGger:CONTinous?
DESCRIPTION	This command sets/queries the trigger mode of ARB continuous trigger.
DATA TYPE	Enumeration
RANGE	FREErun | RUNIgnored | RUNRestart
RETURN	Enumeration
DEFAULT VALUE	FREErun
EQUIVALENT MENU	IQ > ARB > Trigger > Trigger Type(Continuous) > Continuous Mode
EXAMPLE	*:IQ:DUALarb:TRIGger:CONTinous RUNIgnored* *:IQ:DUALarb:TRIGger:CONTinous?* Return: *RUNIgnored\n*

3.4.14.45 ARB Trigger Single Mode ([:SOURce]:IQ:DUALarb:TRIGger:SINGle)

SYNTAX	[:SOURce]:IQ:DUALarb:TRIGger:SINGle NOREtrigger | BUFFeredtrig | RESTartontrig [:SOURce]:IQ:DUALarb:TRIGger:SINGle?
DESCRIPTION	This command sets/queries the trigger mode of ARB single trigger.
DATA TYPE	Enumeration
RANGE	NOREtrigger | BUFFeredtrig | RESTartontrig
RETURN	Enumeration
DEFAULT VALUE	NOREtrigger
EQUIVALENT MENU	IQ > ARB > Trigger > Trigger Type(Single) > Single Mode
EXAMPLE	*:IQ:DUALarb:TRIGger:SINGle BUFFeredtrig* *:IQ:DUALarb:TRIGger:SINGle?* Return: *BUFFeredtrig\n*

3.4.14.46 ARB Trigger Segment Mode ([:SOURce]:IQ:DUALarb:TRIGger:SEGMent)

SYNTAX	[:SOURce]:IQ:DUALarb:TRIGger:SEGMent SINGle | CONTinous [:SOURce]:IQ:DUALarb:TRIGger:SEGMent?
DESCRIPTION	This command sets/queries the trigger mode of ARB segment triggered in advance.
DATA TYPE	Enumeration
RANGE	SINGle | CONTinous
RETURN	Enumeration
DEFAULT VALUE	CONTinous
EQUIVALENT MENU	IQ > ARB > Trigger > Trigger Type(Segment Advance) > Segment Mode
EXAMPLE	*:IQ:DUALarb:TRIGger:SEGMent SINGle* *:IQ:DUALarb:TRIGger:SEGMent?* Return: *SINGle\n*

3.4.14.47 ARB Trigger Gate Mode ([:SOURce]:IQ:DUALarb:TRIGger:GATE)

SYNTAX	[:SOURce]:IQ:DUALarb:TRIGger:GATE LOW | HIGH [:SOURce]:IQ:DUALarb:TRIGger:GATE?
DESCRIPTION	This command sets/queries the trigger mode of ARB external gating trigger.
DATA TYPE	Enumeration
RANGE	LOW | HIGH
RETURN	Enumeration
DEFAULT VALUE	HIGH
EQUIVALENT MENU	IQ > ARB > Trigger > Trigger Type(Ext Gated) > Gate Mode
EXAMPLE	*:IQ:DUALarb:TRIGger:GATE LOW* *:IQ:DUALarb:TRIGger:GATE?* Return: *LOW\n*

3.4.14.48 ARB Trigger Source ([:SOURce]:IQ:DUALarb:TRIGger:SOURce)

SYNTAX	[:SOURce]:IQ:DUALarb:TRIGger:SOURce KEY| BUS| EXT [:SOURce]:IQ:DUALarb:TRIGger:SOURce?
DESCRIPTION	This command sets/queries the trigger source of ARB trigger.
DATA TYPE	Enumeration
RANGE	KEY| BUS| EXT
RETURN	Enumeration
DEFAULT VALUE	KEY

EQUIVALENT MENU	IQ > ARB > Trigger > Trigger Source
EXAMPLE	*:IQ:DUALarb:TRIGger:SOURce EXT* *:IQ:DUALarb:TRIGger:SOURce?* Return: *EXT\n*

3.4.14.49 ARB Bus Trigger ([:SOURce]:IQ:DUALarb:TRG)

SYNTAX	[:SOURce]:IQ:DUALarb:TRG
DESCRIPTION	When the ARB trigger source is Bus, executing this command sends an ARB trigger signal.
EQUIVALENT MENU	None
EXAMPLE	*:IQ:DUALarb:TRG*

3.4.14.50 ARB Trigger Polarity ([:SOURce]:IQ:DUALarb:TRIGger:POL)

SYNTAX	[:SOURce]:IQ:DUALarb:TRIGger:POL POS | NEG [:SOURce]:IQ:DUALarb:TRIGger:POL?
DESCRIPTION	This command sets/queries the trigger polarity of ARB external trigger.
DATA TYPE	Enumeration
RANGE	POS | NEG
RETURN	Enumeration
DEFAULT VALUE	POS
EQUIVALENT MENU	IQ > ARB > Trigger > Trigger Source(Ext) > Ext Polarity
EXAMPLE	*:IQ:DUALarb:TRIGger:POL NEG* *:IQ:DUALarb:TRIGger:POL?* Return: *NEG\n*

3.4.14.51 ARB Trigger Delay Type ([:SOURce]:IQ:DUALarb:TRIGger:DELay:TYPE)

SYNTAX	[:SOURce]:IQ:DUALarb:TRIGger:DELay:TYPE OFF | TIME | SAMPle [:SOURce]:IQ:DUALarb:TRIGger:DELay:TYPE?
DESCRIPTION	This command sets/queries the trigger delay type of ARB external trigger.
DATA TYPE	Enumeration
RANGE	OFF | TIME | SAMPle
RETURN	Enumeration
DEFAULT VALUE	OFF
EQUIVALENT MENU	IQ > ARB > Trigger > Trigger Source(Ext) > Delay Type
EXAMPLE	*:IQ:DUALarb:TRIGger:DELay:TYPE SAMPle*

	:IQ:DUALarb:TRIGger:DELay:TYPE? Return: *SAMPle\n*

3.4.14.52 ARB Trigger Delay Time ([:SOURce]:IQ:DUALarb:TRIGger:DELay:TIME)

SYNTAX	[:SOURce]:IQ:DUALarb:TRIGger:DELay:TIME <value> [:SOURce]:IQ:DUALarb:TRIGger:DELay:TIME?
DESCRIPTION	This command sets/queries the trigger delay time of ARB external trigger.
DATA TYPE	Float, unit: s
RANGE	0 ~ 40s
RETURN	Float, unit: s
DEFAULT VALUE	0
EQUIVALENT MENU	IQ > ARB > Trigger > Trigger Source(Ext) > Delay Type(Time) > Delay Time
EXAMPLE	*:IQ:DUALarb:TRIGger:DELay:TIME 2* *:IQ:DUALarb:TRIGger:DELay:TIME?* Return: *2\n*

3.4.14.53 ARB Trigger Delay Samples ([:SOURce]:IQ:DUALarb:TRIGger:DELay:SAMPle)

SYNTAX	[:SOURce]:IQ:DUALarb:TRIGger:DELay:SAMPle <value> [:SOURce]:IQ:DUALarb:TRIGger:DELay:SAMPle?
DESCRIPTION	This command sets/queries the trigger delay samples of ARB external trigger.
DATA TYPE	Integer
RANGE	0 ~ 100,000,000
RETURN	Integer
DEFAULT VALUE	0
EQUIVALENT MENU	IQ > ARB > Trigger > Trigger Source(Ext) > Delay Type(Sample) > Delay Samples
EXAMPLE	*:IQ:DUALarb:TRIGger:DELay:SAMPle 500* *:IQ:DUALarb:TRIGger:DELay:SAMPle?* Return: *500\n*

3.4.14.54 Header Info ([:SOURce]:IQ:DUALarb:HEADer:INFO?)

SYNTAX	[:SOURce]:IQ:DUALarb:HEADer:INFO?
DESCRIPTION	This command is used to query the content of the waveform header file. Before querying the content of the header file, you need to select the waveform to be played.

RETURN	String
DEFAULT VALUE	None
EQUIVALENT MENU	IQ > ARB > Waveform Header
EXAMPLE	*:RADio:ARB:WAVeform "WFM:SINE_WAVE"* *:IQ:DUALarb:HEADer:INFO?* Return: *discript=* *sampling rate=Unspecified* *marker1 polary=Unspecified* *marker2 polary=Unspecified* *marker3 polary=Unspecified* *marker4 polary=Unspecified* *rf marker=Unspecified* *output marker=Unspecified* *atten type=Unspecified* *atten value=Unspecified* *noise state=Unspecified* *noise output=Unspecified* *noise power control=Unspecified* *noise total power=Unspecified* *noise carrier power=Unspecified* *noise noise power=Unspecified* *channel noise power=Unspecified* *carrier to noise ratio format=Unspecified* *carrier to noise ratio=Unspecified* *bit to noise ratio=Unspecified* *carrier bit ratio=Unspecified* *carrier bandwidth=Unspecified* *noise bandwidth=Unspecified* *baseband offset state=Unspecified* *baseband offset freq=Unspecified\n\n*

3.4.14.55 Clear Header ([:SOURce]:IQ:DUALarb:HEADer:CLEar)

SYNTAX	[:SOURce]:IQ:DUALarb:HEADer:CLEar
DESCRIPTION	This command clears the contents of the header file for a waveform segment or sequence.
EQUIVALENT MENU	IQ > ARB > Waveform Header > Clear Header
EXAMPLE	*:IQ:DUALarb:HEADer:CLEar*

3.4.14.56 Save To Header ([:SOURce]:IQ:DUALarb:HEADer:STORe)

SYNTAX	[:SOURce]:IQ:DUALarb:HEADer:STORe
DESCRIPTION	This command saves the header file contents of a waveform segment or sequence.
EQUIVALENT MENU	IQ > ARB > Waveform Header > Save To Header
EXAMPLE	*:IQ:DUALarb:HEADer:STORe*

3.4.14.57 Describe ([:SOURce]:IQ:DUALarb:HEADer:DESCript)

SYNTAX	[:SOURce]:IQ:DUALarb:HEADer:DESCript <“string”> [:SOURce]:IQ:DUALarb:HEADer:DESCript?
DESCRIPTION	This command sets/queries the header file description of a waveform segment or sequence.
DATA TYPE	String
RANGE	Please refer to the user manual for naming rules.
RETURN	String
DEFAULT VALUE	No description
EQUIVALENT MENU	IQ > ARB > Waveform Header > Describe
EXAMPLE	*:IQ:DUALarb:HEADer:DESCript "INFO"* *:IQ:DUALarb:HEADer:DESCript?* Return: *INFO\n*

3.4.14.58 Multicarrier Waveform Name ([:SOURce]:RADio:DMODulation:ARB:SETup:MCARrier:NAME)

SYNTAX	[:SOURce]:RADio:DMODulation:ARB:SETup:MCARrier:NAME <“waveform”> [:SOURce]:RADio:DMODulation:ARB:SETup:MCARrier:NAME?
DESCRIPTION	This command sets/queries the waveform name of ARB multi-carrier.
DATA TYPE	String
RANGE	Please refer to the user manual for naming rules.
RETURN	String
DEFAULT VALUE	MULTICARRIER
EQUIVALENT MENU	IQ > ARB > Multi Carrier > Waveform Name
EXAMPLE	*:RADio:DMODulation:ARB:SETup:MCARrier:NAME “MULTI_TEST”* *:RADio:DMODulation:ARB:SETup:MCARrier:NAME?* Return: *MULTI_TEST\n*

3.4.14.59 Multicarrier Create and Load ([:SOURce]:RADio:DMODulation:ARB:SETup)

SYNTAX	[:SOURce]:RADio:DMODulation:ARB:SETup
DESCRIPTION	This command can create a multicarrier based on the current settings, then load the multicarrier into an ARB segment and select to play the multicarrier.
EQUIVALENT MENU	IQ > ARB > Multi Carrier > Create and Load
EXAMPLE	*:RADio:DMODulation:ARB:SETup*

3.4.14.60 Multicarrier Assistant ([:SOURce]:RADio:DMODulation:ARB:SETup:MCARrier)

SYNTAX	[:SOURce]:RADio:DMODulation:ARB:SETup:MCARrier <"segment">,<num>,<freq_space> [:SOURce]:RADio:DMODulation:ARB:SETup:MCARrier?
DESCRIPTION	This command builds a carrier table using the specified number of identical subcarriers and frequency spacing. The query command returns the subcarrier name, subcarrier number and frequency interval of the multi-carrier.
DATA TYPE	segment: string, num: integer, number of carriers, freq_space: double, frequency interval between carriers, unit: Hz.
RANGE	segment: waveform segment in the ARB waveform segment list, num: 2 ~ 100, freq_space: 0 ~ maximum sampling bandwidth/(number of carriers-1).
RETURN	String Format: <carrier>,<num carriers>,<freq spacing>
DEFAULT VALUE	*NONE,2,1000000\n
EQUIVALENT MENU	IQ > ARB > Multi Carrier > Carrier Table > Assistant
EXAMPLE	*:RADio:DMODulation:ARB:SETup:MCARrier "SINE_WAVE",3,2e6* *:RADio:DMODulation:ARB:SETup:MCARrier?* Return: *SINE_WAVE,3,2000000\n*

3.4.14.61 Multicarrier Table ([:SOURce]:RADio:DMODulation:ARB:SETup:MCARrier:TABLe)

SYNTAX	[:SOURce]:RADio:DMODulation:ARB:SETup:MCARrier:TABLe INIT|APPend|<carrier_num>,<"segment">,<freq_offset>,<power>,<phase> [:SOURce]:RADio:DMODulation:ARB:SETup:MCARrier:TABLe? <carrier_num>
DESCRIPTION	This command edits the specified row of the ARB multi-carrier list. INIT: This option deletes all multi-carriers and then writes a line of specified carriers, APPend: This option adds a new row of specified carriers after the multi-carrier list. carrier_num: This option modifies the specified row of the multi-carrier list. The query command queries the specified row of the ARB multi-carrier list.
DATA TYPE	INIT|APPend: enumeration, carrier_num: integer, carrier number, segment: string, freq_offset: double, offset frequency of the carrier, unit: Hz, power: double, carrier gain, unit: dB, phase: double, carrier phase, unit: ° (degree).
RANGE	carrier_num: 1 ~ total number of carriers, segment: waveform segment in the ARB waveform segment list, freq_offset: -60 MHz ~ 60 MHz, power: -40 ~ 0, phase: -360 ~ 360.

RETURN	String Format: <carrier>,<freq_offset>,<power>,<phase>,<sample_clock>, <sample_points>
DEFAULT VALUE	SINE_WAVE,0,0,0,2e+06,200\n
EQUIVALENT MENU	IQ > ARB > Multi Carrier > Carrier Table
EXAMPLE	*:RADio:DMODulation:ARB:SETup:MCARrier:TABLe INIT,"RAMP_WAVE",1000000,-10,20* *:RADio:DMODulation:ARB:SETup:MCARrier:TABLe APPend,"RAMP_WAVE",2000000,-5,90* *:RADio:DMODulation:ARB:SETup:MCARrier:TABLe 2,"TEST",-5000000,-2,-30* *:RADio:DMODulation:ARB:SETup:MCARrier:TABLe? 1* Return: *RAMP_WAVE,1e+06,-10,20,2e+06,200\n*

3.4.14.62 Multicarrier Save ([:SOURce]:RADio:DMODulation:ARB:SETup:MCARrier:STORe)

SYNTAX	[:SOURce]:RADio:DMODulation:ARB:SETup:MCARrier:STORe <“file_name”>
DESCRIPTION	This command saves the multi carrier table to a ML file.
DATA TYPE	String
RANGE	Please refer to the user manual for naming rules.
EQUIVALENT MENU	IQ > ARB > Multi Carrier > Carrier Table > Save
EXAMPLE	*:RADio:DMODulation:ARB:SETup:MCARrier:STORe “Multi_Table.ml”*

3.4.14.63 Multicarrier Load ([:SOURce]:IQ:CARRier:LOAD)

SYNTAX	[:SOURce]:IQ:CARRier:LOAD <”file_name”>
DESCRIPTION	This command loads the multi carrier table from a ML file.
DATA TYPE	String
RANGE	None
EQUIVALENT MENU	IQ > ARB > Multi Carrier > Carrier Table > Load
EXAMPLE	*:IQ:CARRier:LOAD “Multi_Table.ml”*

3.4.14.64 Multicarrier Power Reference ([:SOURce]:IQ:CARRier:POWer:TYPE)

SYNTAX	[:SOURce]:IQ:CARRier:POWer:TYPE RMS | PEAK [:SOURce]:IQ:CARRier:POWer:TYPE?
DESCRIPTION	This command sets/queries the power reference type of ARB multi-carrier.
DATA TYPE	Enumeration

RANGE	RMS | PEAK
RETURN	Enumeration
DEFAULT VALUE	PEAK
EQUIVALENT MENU	IQ > ARB > Multi Carrier > Power Reference
EXAMPLE	*:IQ:CARRier:POWer:TYPE RMS* *:IQ:CARRier:POWer:TYPE?* Return: *RMS\n*

3.4.14.65 Multicarrier Signal Period Mode ([:SOURce]:IQ:CARRier:PERIod:MODE)

SYNTAX	[:SOURce]:IQ:CARRier:PERIod:MODE LONGest | SHORtest | USER | LCM [:SOURce]:IQ:CARRier:PERIod:MODE?
DESCRIPTION	This command sets/queries the signal period mode of ARB multi-carrier.
DATA TYPE	Enumeration
RANGE	LONGest | SHORtest | USER | LCM
RETURN	Enumeration
DEFAULT VALUE	LCM
EQUIVALENT MENU	IQ > ARB > Multi Carrier > Signal Period Mode
EXAMPLE	*:IQ:CARRier:PERIod:MODE LONGest* *:IQ:CARRier:PERIod:MODE?* Return: *LONGest\n*

3.4.14.66 Multicarrier Signal Period ([:SOURce]:IQ:CARRier:PERIod)

SYNTAX	[:SOURce]:IQ:CARRier:PERIod <value> [:SOURce]:IQ:CARRier:PERIod?
DESCRIPTION	When the multi-carrier signal period mode is custom, this command sets the multi-carrier signal period. The query command returns the signal period of the multi-carrier.
DATA TYPE	Float, unit: s
RANGE	200/multi-carrier sampling rate ~ min(10e6/multi-carrier sampling rate, (35e6 - total number of sub-carrier sampling points)/(3 * multi-carrier sampling rate))
RETURN	Float, unit: s
DEFAULT VALUE	None
EQUIVALENT MENU	IQ > ARB > Multi Carrier > Signal Period
EXAMPLE	*:IQ:CARRier:PERIod 10e-3* *:IQ:CARRier:PERIod?* Return: *0.01\n*

3.4.14.67 Multicarrier Sampling Rate ([:SOURce]:IQ:CARRier:SAMPlerate?)

SYNTAX	[:SOURce]:IQ:CARRier:SAMPlerate?
DESCRIPTION	This command queries the sampling rate of ARB multi-carrier.
RETURN	Float, unit: Hz
DEFAULT VALUE	None
EQUIVALENT MENU	IQ > ARB > Multi Carrier > Sampling Rate
EXAMPLE	*:IQ:CARRier:SAMPlerate?* Return: *21000000\n*

3.4.15 I/Q Control

3.4.15.1 I/Q Mod State ([:SOURce]:DM:STATe)

SYNTAX	[:SOURce]:DM:STATe ON|OFF|1|0 [:SOURce]:DM:STATe?
DESCRIPTION	This command sets/queries the switch status of the I/Q modulator. **Note:** When the Custom modulation, ARB modulation, Stream modulation, IoT modulation, multi-tone or AWGN function is turned on, the I/Q modulator will be turned on automatically. You also need to turn on the IQ modulation master switch to enable the modulation function. For related commands, please refer to "IQ Modulation Switch ([:SOURce]:FUNCtion:DM:STATe)".
DATA TYPE	Boolean
RANGE	ON|OFF|1|0
RETURN	1|0
DEFAULT VALUE	0
EQUIVALENT MENU	IQ > I/Q Control > I/Q Mod State
EXAMPLE	*:DM:STATe ON* *:DM:STATe?* Return: *1\n*

3.4.15.2 I/Q Source ([:SOURce]:DM:SOURce)

SYNTAX	[:SOURce]:DM:SOURce EXTernal|INTernal [:SOURce]:DM:SOURce?
DESCRIPTION	This command selects/queries the I/Q modulator source.
DATA TYPE	Enumeration
RANGE	EXTernal|INTernal
RETURN	Enumeration

DEFAULT VALUE	INTernal
EQUIVALENT MENU	IQ > I/Q Control > I/Q Source
EXAMPLE	*:DM:SOURce EXTernal* *:DM:SOURce?* Return: *EXTernal\n*

3.4.15.3 Compensation Channel ([:SOURce]:DM:BW:CAL:LINK)

SYNTAX	[:SOURce]:DM:BW:CAL:LINK RF|OUTPut [:SOURce]:DM:BW:CAL:LINK?
DESCRIPTION	This command sets/queries broadband compensation links.
DATA TYPE	Enumeration
RANGE	RF|OUTPut
RETURN	Enumeration
DEFAULT VALUE	RF
EQUIVALENT MENU	IQ > I/Q Control > Compensation Channel
EXAMPLE	*:DM:BW:CAL:LINK OUTPut* *:DM:BW:CAL:LINK?* Return: *OUTPut\n*

3.4.15.4 I/Q Adjustment State ([:SOURce]:DM:IQADjustment[:STATe])

SYNTAX	[:SOURce]:DM:IQADjustment[:STATe] ON|OFF|1|0 [:SOURce]:DM:IQADjustment[:STATe]?
DESCRIPTION	This command sets/queries the switch status of I/Q adjustment.
DATA TYPE	Boolean
RANGE	ON|OFF|1|0
RETURN	1|0
DEFAULT VALUE	1
EQUIVALENT MENU	IQ > I/Q Control > I/Q Adjustment
EXAMPLE	*:DM:IQADjustment ON* *:DM:IQADjustment?* Return: *1\n*

3.4.15.5 Gain Balance ([:SOURce]:DM:IQADjustment:GAIN)

SYNTAX	[:SOURce]:DM:IQADjustment:GAIN <val> [:SOURce]:DM:IQADjustment:GAIN?
DESCRIPTION	This command sets/queries the gain for the I signal relative to the Q signal.

DATA TYPE	Float, unit: dB
RANGE	-4 ~ 4
RETURN	Float, unit: dB
DEFAULT VALUE	0
EQUIVALENT MENU	IQ > I/Q Control > I/Q Adjustment > Gain Balance
EXAMPLE	*:DM:IQADjustment:GAIN -0.5* *:DM:IQADjustment:GAIN?* Return: *-0.5\n*

3.4.15.6 I Offset ([:SOURce]:DM:IQADjustment:IOFFset)

SYNTAX	[:SOURce]:DM:IQADjustment:IOFFset <val> [:SOURce]:DM:IQADjustment:IOFFset?
DESCRIPTION	This command adjusts the I channel offset value.
DATA TYPE	Float, unit: %
RANGE	-50 ~ 50
RETURN	Float, unit: %
DEFAULT VALUE	0
EQUIVALENT MENU	IQ > I/Q Control > I/Q Adjustment > I Offset
EXAMPLE	*:DM:IQADjustment:IOFFset 1.2* *:DM:IQADjustment:IOFFset?* Return: *1.2\n*

3.4.15.7 Q Offset ([:SOURce]:DM:IQADjustment:QOFFset)

SYNTAX	[:SOURce]:DM:IQADjustment:QOFFset <val> [:SOURce]:DM:IQADjustment:QOFFset?
DESCRIPTION	This command adjusts the Q channel offset value.
DATA TYPE	Float, unit: %
RANGE	-50 ~ 50
RETURN	Float, unit: %
DEFAULT VALUE	0
EQUIVALENT MENU	IQ > I/Q Control > I/Q Adjustment > Q Offset
EXAMPLE	*:DM:IQADjustment:QOFFset -0.35* *:DM:IQADjustment:QOFFset?* Return: *-0.35\n*

3.4.15.8 Quad Angle Adjustment ([:SOURce]:DM:IQADjustment:QSKew)

SYNTAX	[:SOURce]:DM:IQADjustment:QSKew <val> [:SOURce]:DM:IQADjustment:QSKew?
DESCRIPTION	This command adjusts the phase angle (quadrature skew) between the I and Q vectors by increasing or decreasing the Q phase angle. It only affects the RF output path. Positive skew causes the angle to increase from 90 degrees, while negative skew causes the angle to decrease from 90 degrees. When the quadrature skew is zero, the phase angle between the I and Q vectors is 90 degrees.
DATA TYPE	Float, unit: degree
RANGE	-10 ~ 10
RETURN	Float, unit: degree
DEFAULT VALUE	0
EQUIVALENT MENU	IQ > I/Q Control > I/Q Adjustment > Quad Angle Adjustment
EXAMPLE	*:DM:IQAD:QSK 1.52* *:DM:IQAD:QSK?* Return: *1.52\n*

3.4.15.9 I/Q Output State ([:SOURce]:DM:IQADjustment:EXTernal[:STATe])

SYNTAX	[:SOURce]:DM:IQADjustment:EXTernal[:STATe] ON|OFF|1|0 [:SOURce]:DM:IQADjustment:EXTernal[:STATe]?
DESCRIPTION	This command sets/queries the output status of the I/Q signal.
DATA TYPE	Boolean
RANGE	ON|OFF|1|0
RETURN	1|0
DEFAULT VALUE	0
EQUIVALENT MENU	IQ > I/Q Control > I/Q Output
EXAMPLE	*:DM:IQADjustment:EXTernal 1* *:DM:IQADjustment:EXTernal?* Return: *1\n*

3.4.15.10 I/Q Output Atten ([:SOURce]:DM:IQADjustment:EXTernal:IQATten)

SYNTAX	[:SOURce]:DM:IQADjustment:EXTernal:IQATten <val> [:SOURce]:DM:IQADjustment:EXTernal:IQATten?
DESCRIPTION	This command sets the attenuation value of the I/Q output.
DATA TYPE	Float, unit: dB
RANGE	0 ~ 20
RETURN	Float, unit: dB
DEFAULT VALUE	0

EQUIVALENT MENU	IQ > I/Q Control > I/Q Output > I/Q Output Atten
EXAMPLE	*:DM:IQADjustment:EXTernal:IQATten 2.13* *:DM:IQADjustment:EXTernal:IQATten?* Return: *2.13\n*

3.4.15.11 I/Q Output Gain Balance ([:SOURce]:DM:IQADjustment:EXTernal:GAIN)

SYNTAX	[:SOURce]:DM:IQADjustment:EXTernal:GAIN <val> [:SOURce]:DM:IQADjustment:EXTernal:GAIN?
DESCRIPTION	This command sets/queries the I/Q gain ratio for signals routed to the rear panel I and Q output connectors
DATA TYPE	Float, unit: dB
RANGE	-4 ~ 4
RETURN	Float, unit: dB
DEFAULT VALUE	0
EQUIVALENT MENU	IQ > I/Q Control > I/Q Output > I/Q Output Gain Balance
EXAMPLE	*:DM:IQADjustment:EXTernal:GAIN -1.31* *:DM:IQADjustment:EXTernal:GAIN?* Return: *-1.31\n*

3.4.15.12 I Output Offset ([:SOURce]:DM:IQADjustment:EXTernal:DIOFfset)

SYNTAX	[:SOURce]:DM:IQADjustment:EXTernal:DIOFfset <val> [:SOURce]:DM:IQADjustment:EXTernal:DIOFfset?
DESCRIPTION	This command sets/queries the differential offset voltage for an in-phase (I) signal routed to the I output connectors.
DATA TYPE	Float, unit: mV or V, default is V
RANGE	-3 V ~ 3 V
RETURN	Float, unit: V
DEFAULT VALUE	0
EQUIVALENT MENU	IQ > I/Q Control > I/Q Output > I Output Offset
EXAMPLE	*:DM:IQADjustment:EXTernal:DIOFfset 0.12* *:DM:IQADjustment:EXTernal:DIOFfset?* Return: *0.12\n*

3.4.15.13 Q Output Offset ([:SOURce]:DM:IQADjustment:DQOFFset)

SYNTAX	[:SOURce]:DM:IQADjustment:EXTernal:DQOFfset <val> [:SOURce]:DM:IQADjustment:EXTernal:DQOFfset?
DESCRIPTION	This command sets/queries the differential offset voltage of the quadrature phase (Q) signal routed to the Q output connector.

DATA TYPE	Float, unit: mV or V, default is V
RANGE	-3 V ~ 3 V
RETURN	Float, unit: V
DEFAULT VALUE	0
EQUIVALENT MENU	IQ > I/Q Control > I/Q Output > Q Output Offset
EXAMPLE	*:DM:IQADjustment:EXTernal:DQOFfset -0.12* *:DM:IQADjustment:EXTernal:DQOFfset?* Return: *-0.12\n*

3.4.15.14 I/Q Common Offset ([:SOURce]:DM:IQADjustment:EXTernal:COFFset)

SYNTAX	[:SOURce]:DM:IQADjustment:EXTernal:COFFset <val> [:SOURce]:DM:IQADjustment:EXTernal:COFFset?
DESCRIPTION	This command sets/queries the common mode offset voltage for both the in-phase (I) and quadrature-phase(Q) signals going to the rear panel I and Q output connectors.
DATA TYPE	Float, unit: mV or V, default is V
RANGE	-2.5 V ~ 2.5 V
RETURN	Float, unit: V
DEFAULT VALUE	0
EQUIVALENT MENU	IQ > I/Q Control > I/Q Output > I/Q Common Offset
EXAMPLE	*:DM:IQADjustment:EXTernal:COFFset 1 mV* *:DM:IQADjustment:EXTernal:COFFset?* Return: *0.001\n*

3.4.15.15 I/Q Swap ([:SOURce]:IQ:BW:SWAP)

SYNTAX	[:SOURce]:IQ:BW:SWAP ON|OFF|1|0 [:SOURce]:IQ:BW:SWAP?
DESCRIPTION	This command sets/queries the exchange status of I and Q signals. After enabling, the I signal remains unchanged, the Q signal will be inverted, and the spectrum is a mirror image of the normal mode.
DATA TYPE	Boolean
RANGE	ON|OFF|1|0
RETURN	1|0
DEFAULT VALUE	0
EQUIVALENT MENU	IQ > I/Q Control > I/Q Swap
EXAMPLE	*:IQ:BW:SWAP ON* *:IQ:BW:SWAP?* Return: *1\n*

3.4.16 IoT

3.4.16.1 Protocol Type ([:SOURce]:RADio:IOT:TYPE)

SYNTAX	[:SOURce]:RADio:IOT:TYPE ZIGBee | ZWAVe [:SOURce]:RADio:IOT:TYPE?
DESCRIPTION	This command sets/queries the protocol type of IoT modulation.
DATA TYPE	Enumeration
RANGE	ZIGBee | ZWAVe
RETURN	Enumeration
DEFAULT VALUE	ZIGBee
EQUIVALENT MENU	IQ > IoT > Protocol Type
EXAMPLE	*:RADio:IOT:TYPE ZWAVe* *:RADio:IOT:TYPE?* Return: *ZWAVe\n*

3.4.16.2 ZigBee

3.4.16.2.1 Save Waveform ([:SOURce]:RADio:ZIGBee:SAVE)

SYNTAX	[:SOURce]:RADio:ZIGBee:SAVE <”file_name”>
DESCRIPTION	This command saves ZigBee modulated waveform data to an ARB file.
DATA TYPE	String
RANGE	Please refer to the user manual for naming rules.
EQUIVALENT MENU	IQ > IoT > Protocol Type(ZigBee) > Save Waveform
EXAMPLE	*:RADio:ZIGBee:SAVE “test_zigbee.arb”*

3.4.16.2.2 ZigBee State ([:SOURce]:RADio:ZIGBee:STATe)

SYNTAX	[:SOURce]:RADio:ZIGBee:STATe ON|OFF|1|0 [:SOURce]:RADio:ZIGBee:STATe?
DESCRIPTION	This command sets/queries the switch status of ZigBee modulation.
DATA TYPE	Boolean
RANGE	ON|OFF|1|0
RETURN	1|0
DEFAULT VALUE	0
EQUIVALENT MENU	IQ > IoT > Protocol Type(ZigBee) > ZigBee State
EXAMPLE	*:RADio:ZIGBee:STATe ON* *:RADio:ZIGBee:STATe?* Return: *1\n*

3.4.16.2.3 OverSampling Ratio ([:SOURce]:RADio:ZIGBee:OSAMple:RATio)

SYNTAX	[:SOURce]:RADio:ZIGBee:OSAMple:RATio <val> [:SOURce]:RADio:ZIGBee:OSAMple:RATio?
DESCRIPTION	This command sets/queries the oversampling rate of ZigBee modulated wave.
DATA TYPE	Integer
RANGE	2 ~ 64
RETURN	Integer
DEFAULT VALUE	8
EQUIVALENT MENU	IQ > IoT > Protocol Type(ZigBee) > Basic > OverSampling Ratio
EXAMPLE	*:RADio:ZIGBee:OSAMple:RATio 10* *:RADio:ZIGBee:OSAMple:RATio?* Return: *10\n*

3.4.16.2.4 Number of Frames ([:SOURce]:RADio:ZIGBee:FRAMe:NUMBer)

SYNTAX	[:SOURce]:RADio:ZIGBee:FRAMe:NUMBer <val> [:SOURce]:RADio:ZIGBee:FRAMe:NUMBer?
DESCRIPTION	This command sets and gets the number of frames contained in the ZigBee modulated wave.
DATA TYPE	Integer
RANGE	1 ~ 2000
RETURN	Integer
DEFAULT VALUE	1
EQUIVALENT MENU	IQ > IoT > Protocol Type(ZigBee) > Basic > Number of Frames
EXAMPLE	*:RADio:ZIGBee:FRAMe:NUMBer 2* *:RADio:ZIGBee:FRAMe:NUMBer?* Return: *2\n*

3.4.16.2.5 Total Sample Points ([:SOURce]:RADio:ZIGBee:TOTal:SAMPle:POINts?)

SYNTAX	[:SOURce]:RADio:ZIGBee:TOTal:SAMPle:POINts?
DESCRIPTION	This command queries the total number of sampling points of the ZigBee modulated wave.
RETURN	Integer
DEFAULT VALUE	None
EQUIVALENT MENU	IQ > IoT > Protocol Type(ZigBee) > Basic > Total Sample Points
EXAMPLE	*:RADio:ZIGBee:TOTal:SAMPle:POINts?* Return: *20608\n*

3.4.16.2.6 Waveform Length ([:SOURce]:RADio:ZIGBee:WAVeform:LENgth?)

SYNTAX	[:SOURce]:RADio:ZIGBee:WAVeform:LENgth?
DESCRIPTION	This command queries the waveform length of the ZigBee modulated wave.
RETURN	Float, unit: s
DEFAULT VALUE	None
EQUIVALENT MENU	IQ > IoT > Protocol Type(ZigBee) > Basic > Waveform Length
EXAMPLE	*:RADio:ZIGBee:WAVeform:LENgth?* Return: *0.00644\n*

3.4.16.2.7 PHY SCHEme ([:SOURce]:RADio:ZIGBee:PHY:SCHEme)

SYNTAX	[:SOURce]:RADio:ZIGBee:PHY:SCHEme OQPSK| BPSK [:SOURce]:RADio:ZIGBee:PHY:SCHEme?
DESCRIPTION	This command sets/queries the modulation method of ZigBee modulation.
DATA TYPE	Enumeration
RANGE	OQPSK| BPSK
RETURN	Enumeration
DEFAULT VALUE	OQPSK
EQUIVALENT MENU	IQ > IoT > Protocol Type(ZigBee) > General Settings > PHY SCHEme
EXAMPLE	*:RADio:ZIGBee:PHY:SCHEme BPSK* *:RADio:ZIGBee:PHY:SCHEme?* Return: *BPSK\n*

3.4.16.2.8 Idle Interval ([:SOURce]:RADio:ZIGBee:IDLE)

SYNTAX	[:SOURce]:RADio:ZIGBee:IDLE <val> [:SOURce]:RADio:ZIGBee:IDLE?
DESCRIPTION	This command sets/queries the idle interval (in seconds) between frames contained in the ZigBee modulated wave.
DATA TYPE	Float, unit: ns, us, ms or s, default is s
RANGE	0 ~ 200 ms
RETURN	Float, unit: s
DEFAULT VALUE	100 us
EQUIVALENT MENU	IQ > IoT > Protocol Type(ZigBee) > General Settings > Idle Interval
EXAMPLE	*:RADio:ZIGBee:IDLE 10 ms* *:RADio:ZIGBee:IDLE?*

	Return: *0.01\n*

3.4.16.2.9 Frequency Band ([:SOURce]:RADio:ZIGBee:FREQ:BAND)

SYNTAX	[:SOURce]:RADio:ZIGBee:FREQ:BAND F868M | F915M| F2450M [:SOURce]:RADio:ZIGBee:FREQ:BAND?
DESCRIPTION	This command sets/queries the frequency band of ZigBee modulation.
DATA TYPE	Enumeration
RANGE	F868M | F915M| F2450M
RETURN	Enumeration
DEFAULT VALUE	F868M
EQUIVALENT MENU	IQ > IoT > Protocol Type(ZigBee) > General Settings > Frequency Band
EXAMPLE	*:RADio:ZIGBee:FREQ:BAND F915M* *:RADio:ZIGBee:FREQ:BAND?* Return: *F915M\n*

3.4.16.2.10 Data Rate ([:SOURce]:RADio:ZIGBee:RATE?)

SYNTAX	[:SOURce]:RADio:ZIGBee:RATE?
DESCRIPTION	This command queries the data rate of ZigBee modulated.
RETURN	Float, unit: b/s
DEFAULT VALUE	None
EQUIVALENT MENU	IQ > IoT > Protocol Type(ZigBee) > General Settings > Data Rate
EXAMPLE	*:RADio:ZIGBee:RATE?* Return: *40000\n*

3.4.16.2.11 Preamble ([:SOURce]:RADio:ZIGBee:PREAmble?)

SYNTAX	[:SOURce]:RADio:ZIGBee:PREAmble?
DESCRIPTION	This command gets the Preamble field (hexadecimal) of ZigBee PPDU.
RETURN	Integer
DEFAULT VALUE	00000000
EQUIVALENT MENU	IQ > IoT > Protocol Type(ZigBee) > PPDU Settings > Preamble(Hex)
EXAMPLE	*:RADio:ZIGBee:PREAmble?* Return: *00000000\n*

3.4.16.2.12 SFD ([:SOURce]:RADio:ZIGBee:SFD?)

SYNTAX	[:SOURce]:RADio:ZIGBee:SFD?
DESCRIPTION	This command gets the SFD field (hexadecimal) of ZigBee PPDU.
RETURN	Integer
DEFAULT VALUE	a7
EQUIVALENT MENU	IQ > IoT > Protocol Type(ZigBee) > PPDU Settings > SFD(Hex)
EXAMPLE	*:RADio:ZIGBee:SFD?* Return: *a7\n*

3.4.16.2.13 PHR ([:SOURce]:RADio:ZIGBee:PHR?)

SYNTAX	[:SOURce]:RADio:ZIGBee:PHR?
DESCRIPTION	This command gets the PHR field (hexadecimal) of ZigBee PPDU.
RETURN	Integer
DEFAULT VALUE	21
EQUIVALENT MENU	IQ > IoT > Protocol Type(ZigBee) > PPDU Settings > PHR (Hex)
EXAMPLE	*:RADio:ZIGBee:PHR?* Return: *21\n*

3.4.16.2.14 MAC Frame Type ([:SOURce]:RADio:ZIGBee:FRAMe:TYPE)

SYNTAX	[:SOURce]:RADio:ZIGBee:FRAMe:TYPE GENeral | BEACon | DATA | ACK | MAC [:SOURce]:RADio:ZIGBee:FRAMe:TYPE?
DESCRIPTION	This command sets/queries the frame type of ZigBee modulation.
DATA TYPE	Enumeration
RANGE	GENeral | BEACon | DATA | ACK | MAC
RETURN	Enumeration
DEFAULT VALUE	GENeral
EQUIVALENT MENU	IQ > IoT > Protocol Type(ZigBee) > PPDU Settings > MAC Frame Type
EXAMPLE	*:RADio:ZIGBee:FRAMe:TYPE DATA* *:RADio:ZIGBee:FRAMe:TYPE?* Return: *DATA\n*

3.4.16.2.15 MAC Header State ([:SOURce]:RADio:ZIGBee:MAC:HEADer:STATe)

SYNTAX	[:SOURce]:RADio:ZIGBee:MAC:HEADer:STATe ON|OFF|1|0 [:SOURce]:RADio:ZIGBee:MAC:HEADer:STATe?

DESCRIPTION	This command sets/queries the enable status of the MAC Header field of ZigBee PPDU.
DATA TYPE	Boolean
RANGE	ON|OFF|1|0
RETURN	1|0
DEFAULT VALUE	1
EQUIVALENT MENU	IQ > IoT > Protocol Type(ZigBee) > PPDU Settings > MAC Header
EXAMPLE	*:RADio:ZIGBee:MAC:HEADer:STATe OFF* *:RADio:ZIGBee:MAC:HEADer:STATe?* Return: *0\n*

3.4.16.2.16 General MAC Frame Header ([:SOURce]:RADio:ZIGBee:FRAMe:GENeral)

SYNTAX	[:SOURce]:RADio:ZIGBee:FRAMe:GENeral <"header"> [:SOURce]:RADio:ZIGBee:FRAMe:GENeral?
DESCRIPTION	This command sets/queries the MAC Header field of ZigBee General frame.
DATA TYPE	String
RANGE	None
RETURN	String
DEFAULT VALUE	8821,01,1234,5678,4321,8765,,,\n
EQUIVALENT MENU	IQ > IoT > Protocol Type(ZigBee) > PPDU Settings > MAC Frame Type(General) > MAC Header
EXAMPLE	*:RADio:ZIGBee:FRAMe:GENeral “8888,01,1234,5678,4321,8765,,,AABB”* *:RADio:ZIGBee:FRAMe:GENeral?* Return: *8888,01,1234,5678,4321,8765,,,AABB\n*

3.4.16.2.17 Beacon Frame Header ([:SOURce]:RADio:ZIGBee:FRAMe:BEACon)

SYNTAX	[:SOURce]:RADio:ZIGBee:FRAMe:BEACon <"header"> [:SOURce]:RADio:ZIGBee:FRAMe:BEACon?
DESCRIPTION	This command sets/queries the MAC Header field of ZigBee Beacon frame.
DATA TYPE	String
RANGE	None
RETURN	String
DEFAULT VALUE	8820,01,1234,5678,4321,8765,,0000,00,00\n
EQUIVALENT MENU	IQ > IoT > Protocol Type(ZigBee) > PPDU Settings > MAC Frame Type(Beacon) > MAC Header
EXAMPLE	*:RADio:ZIGBee:FRAMe:BEACon*

	"8821,01,1234,5678,4321,8765,,0000,BD,00" *:RADio:ZIGBee:FRAMe:BEACon?* Return: *8820,01,1234,5678,4321,8765,,0000,BD,00\n*

3.4.16.2.18 Data Frame Header ([:SOURce]:RADio:ZIGBee:FRAMe:DATA)

SYNTAX	[:SOURce]:RADio:ZIGBee:FRAMe:DATA <"header"> [:SOURce]:RADio:ZIGBee:FRAMe:DATA?
DESCRIPTION	This command sets/queries the MAC Header field of ZigBee Data frame.
DATA TYPE	String
RANGE	None
RETURN	String
DEFAULT VALUE	8821,01,1234,5678,4321,8765,,,\n
EQUIVALENT MENU	IQ > IoT > Protocol Type(ZigBee) > PPDU Settings > MAC Frame Type(Data) > MAC Header
EXAMPLE	*:RADio:ZIGBee:FRAMe:DATA "8821,01,1234,5678,4321,8765,,AABB,"* *:RADio:ZIGBee:FRAMe:DATA?* Return: *8821,01,1234,5678,4321,8765,,AABB,\n*

3.4.16.2.19 Ack Frame Header ([:SOURce]:RADio:ZIGBee:FRAMe:ACK)

SYNTAX	[:SOURce]:RADio:ZIGBee:FRAMe:ACK <"header"> [:SOURce]:RADio:ZIGBee:FRAMe:ACK?
DESCRIPTION	This command sets/queries the MAC Header field of ZigBee Ack frame.
DATA TYPE	String
RANGE	None
RETURN	String
DEFAULT VALUE	8822,01\n
EQUIVALENT MENU	IQ > IoT > Protocol Type(ZigBee) > PPDU Settings > MAC Frame Type(Ack) > MAC Header
EXAMPLE	*:RADio:ZIGBee:FRAMe:ACK "1212,56"* *:RADio:ZIGBee:FRAMe:ACK?* Return: *1212,56\n*

3.4.16.2.20 MAC Command Frame Header ([:SOURce]:RADio:ZIGBee:FRAMe:MAC)

SYNTAX	[:SOURce]:RADio:ZIGBee:FRAMe:MAC <"header"> [:SOURce]:RADio:ZIGBee:FRAMe:MAC?
DESCRIPTION	This command sets/queries the MAC Header field of ZigBee MAC Command frame.

DATA TYPE	String
RANGE	None
RETURN	String
DEFAULT VALUE	8823,01,1234,5678,4321,8765,,,,00\n
EQUIVALENT MENU	IQ > IoT > Protocol Type(ZigBee) > PPDU Settings > MAC Frame Type(MAC Command) > MAC Header
EXAMPLE	*:RADio:ZIGBee:FRAMe:MAC "8823,56,1234,5678,4321,8765,, AABB,,12"* *:RADio:ZIGBee:FRAMe:MAC?* Return: *8823,01,1234,5678,4321,8765,,,,00\n*

3.4.16.2.21 Data Type ([:SOURce]:RADio:ZIGBee:PNTYpe)

SYNTAX	[:SOURce]:RADio:ZIGBee:PNTYpe PN9 | PN15 | USER [:SOURce]:RADio:ZIGBee:PNTYpe?
DESCRIPTION	This command sets/queries the data type of ZigBee PPDU.
DATA TYPE	Enumeration
RANGE	PN9 | PN15 | USER
RETURN	Enumeration
DEFAULT VALUE	PN9
EQUIVALENT MENU	IQ > IoT > Protocol Type(ZigBee) > PPDU Settings > Data Type
EXAMPLE	*:RADio:ZIGBee:PNTYpe PN15* *:RADio:ZIGBee:PNTYpe?* Return: *PN15\n*

3.4.16.2.22 Seed ([:SOURce]:RADio:ZIGBee:SEED)

SYNTAX	[:SOURce]:RADio:ZIGBee:SEED <val> [:SOURce]:RADio:ZIGBee:SEED?
DESCRIPTION	This command sets the seed that generates the PN sequence.
DATA TYPE	Integer
RANGE	1 ~ 32767
RETURN	Integer
DEFAULT VALUE	511
EQUIVALENT MENU	IQ > IoT > Protocol Type(ZigBee) > PPDU Settings > Seed(Hex)
EXAMPLE	*:RADio:ZIGBee:SEED 974* *:RADio:ZIGBee:SEED?* Return: *974\n*

3.4.16.2.23 Custom PN Data ([:SOURce]:RADio:ZIGBee:PN:USER:DATA)

SYNTAX	[:SOURce]:RADio:ZIGBee:PN:USER:DATA <user_data>
DESCRIPTION	This command sets custom PN data for the ZigBee PPDU Payload field.
DATA TYPE	Binary string
RANGE	None
EQUIVALENT MENU	IQ > IoT > Protocol Type(ZigBee) > PPDU Settings > Data Type (User) > User Data
EXAMPLE	*:RADio:ZIGBee:PN:USER:DATA 0101001100111*

3.4.16.2.24 Get PN Data ([:SOURce]:RADio:ZIGBee:PN:DATA?)

SYNTAX	[:SOURce]:RADio:ZIGBee:PN:DATA? PN9 | PN15 | USER
DESCRIPTION	This command queries the PN data of the ZigBee PPDU Payload field.
DATA TYPE	Enumeration
RANGE	PN9 | PN15 | USER
RETURN	Binary string
DEFAULT VALUE	None
EQUIVALENT MENU	IQ > IoT > Protocol Type(ZigBee) > PPDU Settings > Data Type (User) > User Data
EXAMPLE	*:RADio:ZIGBee:PN:DATA? USER* Return: *0101001100111\n*

3.4.16.2.25 Store User PN Data ([:SOURce]:RADio:ZIGBee:PN:SAVE)

SYNTAX	[:SOURce]:RADio:ZIGBee:PN:SAVE <”file_name”>
DESCRIPTION	This command saves the customized PN data of the ZigBee PPDU Payload field into the UDATA file.
DATA TYPE	String
RANGE	Please refer to the user manual for naming rules.
EQUIVALENT MENU	IQ > IoT > Protocol Type(ZigBee) > PPDU Settings > Data Type (User) > User Data > Save
EXAMPLE	*:RADio:ZIGBee:PN:SAVE “test.udata”*

3.4.16.2.26 Load User PN Data ([:SOURce]:RADio:ZIGBee:PN:LOAD)

SYNTAX	[:SOURce]:RADio:ZIGBee:PN:LOAD <”file_name”>
DESCRIPTION	This command loads the custom PN data of the ZigBee PPDU Payload field from the UDATA file.

DATA TYPE	String
RANGE	None
EQUIVALENT MENU	IQ > IoT > Protocol Type(ZigBee) > PPDU Settings > Data Type (User) > User Data > Load
EXAMPLE	*:RADio:ZIGBee:PN:LOAD "test.udata"*

3.4.16.2.27 Data Length ([:SOURce]:RADio:ZIGBee:FRAMe:LEN)

SYNTAX	[:SOURce]:RADio:ZIGBee:FRAMe:LEN <val> [:SOURce]:RADio:ZIGBee:FRAMe:LEN?
DESCRIPTION	This command sets/queries the data length of the ZigBee PPDU Payload field.
DATA TYPE	Integer, unit: octets
RANGE	0 ~ 114
RETURN	Integer, unit: octets
DEFAULT VALUE	20
EQUIVALENT MENU	IQ > IoT > Protocol Type(ZigBee) > PPDU Settings > Data Length
EXAMPLE	*:RADio:ZIGBee:FRAMe:LEN 10* *:RADio:ZIGBee:FRAMe:LEN?* Return: *10\n*

3.4.16.2.28 Data Mode ([:SOURce]:RADio:ZIGBee:CONTinuous:STATe)

SYNTAX	[:SOURce]:RADio:ZIGBee:CONTinuous:STATe ON|OFF|1|0 [:SOURce]:RADio:ZIGBee:CONTinuous:STATe?
DESCRIPTION	This command enables or disables data continuity status. Continuous mode will have the data bits spread out continuously over multiple frames, truncated mode will have the same payload data bits for all frames.
DATA TYPE	Boolean
RANGE	ON|OFF|1|0
RETURN	1|0
DEFAULT VALUE	0
EQUIVALENT MENU	IQ > IoT > Protocol Type(ZigBee) > PPDU Settings > Data Mode
EXAMPLE	*:RADio:ZIGBee:CONTinuous:STATe 1* *:RADio:ZIGBee:CONTinuous:STATe?* Return: *1\n*

3.4.16.2.29 MAC FCS State ([:SOURce]:RADio:ZIGBee:FCS:STATe)

SYNTAX	[:SOURce]:RADio:ZIGBee:FCS:STATe ON|OFF|1|0 [:SOURce]:RADio:ZIGBee:FCS:STATe?
DESCRIPTION	This command enables or disables MAC FCS in the PSDU. When turned off, it can be used to simulate an invalid FCS condition since the FCS section is actually filled with user data bits.
DATA TYPE	Boolean
RANGE	ON|OFF|1|0
RETURN	1|0
DEFAULT VALUE	1
EQUIVALENT MENU	IQ > IoT > Protocol Type(ZigBee) > PPDU Settings > MAC FCS
EXAMPLE	*:RADio:ZIGBee:FCS:STATe 0* *:RADio:ZIGBee:FCS:STATe?* Return: *0\n*

3.4.16.2.30 Symbol Timing Error ([:SOURce]:RADio:ZIGBee:SYMBle:TIMing:ERRor)

SYNTAX	[:SOURce]:RADio:ZIGBee:SYMBle:TIMing:ERRor <val> [:SOURce]:RADio:ZIGBee:SYMBle:TIMing:ERRor?
DESCRIPTION	This command sets the shift to the standard symbol rate in ppm.
DATA TYPE	Integer, unit: ppm
RANGE	-300 ~ 300
RETURN	Integer, unit: ppm
DEFAULT VALUE	0
EQUIVALENT MENU	IQ > IoT > Protocol Type(ZigBee) > Impairments > Symbol Timing Error
EXAMPLE	*:RADio:ZIGBee:SYMBle:TIMing:ERRor -10* *:RADio:ZIGBee:SYMBle:TIMing:ERRor?* Return: *-10\n*

3.4.16.2.31 Frequency Offset ([:SOURce]:RADio:ZIGBee:FREQ:OFFSet)

SYNTAX	[:SOURce]:RADio:ZIGBee:FREQ:OFFSet <val> [:SOURce]:RADio:ZIGBee:FREQ:OFFSet?
DESCRIPTION	This command sets or gets the offset (in Hz) of the nominal carrier frequency for ZigBee modulation.
DATA TYPE	Float, unit: Hz
RANGE	-200000 ~ 200000
RETURN	Float, unit: Hz
DEFAULT VALUE	0

EQUIVALENT MENU	IQ > IoT > Protocol Type(ZigBee) > Impairments > Frequency Offset
EXAMPLE	*:RADio:ZIGBee:FREQ:OFFSet 1000* *:RADio:ZIGBee:FREQ:OFFSet?* Return: *1000\n*

3.4.16.2.32 Marker 1 Source ([:SOURce]:RADio:ZIGBee:MARKer:ONE:SOURce)

SYNTAX	[:SOURce]:RADio:ZIGBee:MARKer:ONE:SOURce WAVeform| FRAMe [:SOURce]:RADio:ZIGBee:MARKer:ONE:SOURce?
DESCRIPTION	This command sets/queries the source of Mark 1 of the ZigBee modulated wave. Waveform Start - Indicates the start of the waveform, Frame Start - Indicates the beginning of each frame in the waveform.
DATA TYPE	Enumeration
RANGE	WAVeform| FRAMe
RETURN	Enumeration
DEFAULT VALUE	FRAMe
EQUIVALENT MENU	IQ > IoT > Protocol Type(ZigBee) > Marker > Marker 1 Source
EXAMPLE	*:RADio:ZIGBee:MARKer:ONE:SOURce WAVeform* *:RADio:ZIGBee:MARKer:ONE:SOURce?* Return: *WAVeform\n*

3.4.16.2.33 Marker 2 Soruce ([:SOURce]:RADio:ZIGBee:MARKer:TWO:SOURce)

SYNTAX	[:SOURce]:RADio:ZIGBee:MARKer:TWO:SOURce WAVeform| FRAMe [:SOURce]:RADio:ZIGBee:MARKer:TWO:SOURce?
DESCRIPTION	This command sets/queries the source of Mark 2 of the ZigBee modulated wave. Waveform Start - Indicates the start of the waveform, Frame Start - Indicates the beginning of each frame in the waveform.
DATA TYPE	Enumeration
RANGE	WAVeform| FRAMe
RETURN	Enumeration
DEFAULT VALUE	FRAMe
EQUIVALENT MENU	IQ > IoT > Protocol Type(ZigBee) > Marker > Marker 2 Source
EXAMPLE	*:RADio:ZIGBee:MARKer:TWO:SOURce WAVeform* *:RADio:ZIGBee:MARKer:TWO:SOURce?* Return: *WAVeform\n*

3.4.16.2.34 ZigBee Trigger Type ([:SOURce]:RADio:ZIGBee:TRIGger:TYPE)

SYNTAX	[:SOURce]:RADio:ZIGBee:TRIGger:TYPE CONTinous| SINGle| GATE [:SOURce]:RADio:ZIGBee:TRIGger:TYPE?
DESCRIPTION	This command sets/queries the trigger type of ZigBee.
DATA TYPE	Enumeration
RANGE	CONTinous| SINGle| GATE
RETURN	Enumeration
DEFAULT VALUE	CONTinous
EQUIVALENT MENU	IQ > IoT > Protocol Type(ZigBee) > Trigger > Trigger Type
EXAMPLE	*:RADio:ZIGBee:TRIGger:TYPE SINGle* *:RADio:ZIGBee:TRIGger:TYPE?* Return: *SINGle\n*

3.4.16.2.35 ZigBee Trigger Continuous Mode ([:SOURce]:RADio:ZIGBee:TRIGger:CONTinous)

SYNTAX	[:SOURce]:RADio:ZIGBee:TRIGger:CONTinous FREErun | RUNIgnored [:SOURce]:RADio:ZIGBee:TRIGger:CONTinous?
DESCRIPTION	This command sets/queries the trigger mode of ZigBee continuous trigger.
DATA TYPE	Enumeration
RANGE	FREErun | RUNIgnored
RETURN	Enumeration
DEFAULT VALUE	FREErun
EQUIVALENT MENU	IQ > IoT > Protocol Type(ZigBee) > Trigger > Trigger Type(Continuous) > Continuous Mode
EXAMPLE	*:RADio:ZIGBee:TRIGger:CONTinous RUNIgnored* *:RADio:ZIGBee:TRIGger:CONTinous?* Return: *RUNIgnored\n*

3.4.16.2.36 ZigBee Trigger Gate Mode ([:SOURce]:RADio:ZIGBee:TRIGger:GATE)

SYNTAX	[:SOURce]:RADio:ZIGBee:TRIGger:GATE LOW | HIGH [:SOURce]:RADio:ZIGBee:TRIGger:GATE?
DESCRIPTION	This command sets/queries the trigger mode of ZigBee external gating trigger.
DATA TYPE	Enumeration
RANGE	LOW | HIGH

RETURN	Enumeration
DEFAULT VALUE	LOW
EQUIVALENT MENU	IQ > IoT > Protocol Type(ZigBee) > Trigger > Trigger Type(Ext Gated) > Gate Mode
EXAMPLE	*:RADio:ZIGBee:TRIGger:GATE HIGH* *:RADio:ZIGBee:TRIGger:GATE?* Return: *HIGH\n*

3.4.16.2.37 ZigBee Trigger Source ([:SOURce]:RADio:ZIGBee:TRIGger:SOURce)

SYNTAX	[:SOURce]:RADio:ZIGBee:TRIGger:SOURce KEY| BUS| EXT [:SOURce]:RADio:ZIGBee:TRIGger:SOURce?
DESCRIPTION	This command sets/queries the trigger source of ZigBee trigger.
DATA TYPE	Enumeration
RANGE	KEY| BUS| EXT
RETURN	Enumeration
DEFAULT VALUE	KEY
EQUIVALENT MENU	IQ > IoT > Protocol Type(ZigBee) > Trigger > Trigger Source
EXAMPLE	*:RADio:ZIGBee:TRIGger:SOURce BUS* *:RADio:ZIGBee:TRIGger:SOURce?* Return: *BUS\n*

3.4.16.2.38 ZigBee Trigger Polarity ([:SOURce]:RADio:ZIGBee:TRIGger:POL)

SYNTAX	[:SOURce]:RADio:ZIGBee:TRIGger:POL POS | NEG [:SOURce]:RADio:ZIGBee:TRIGger:POL?
DESCRIPTION	This command sets/queries the trigger polarity of ZigBee external trigger.
DATA TYPE	Enumeration
RANGE	POS | NEG
RETURN	Enumeration
DEFAULT VALUE	POS
EQUIVALENT MENU	IQ > IoT > Protocol Type(ZigBee) > Trigger > Trigger Source(Ext) > Ext Polarity
EXAMPLE	*:RADio:ZIGBee:TRIGger:POL NEG* *:RADio:ZIGBee:TRIGger:POL?* Return: *NEG\n*

3.4.16.2.39 ZigBee Trigger Delay Samples ([:SOURce]:RADio:ZIGBee:TRIGger:DELay:SAMPle)

SYNTAX	[:SOURce]:RADio:ZIGBee:TRIGger:DELay:SAMPle <samples> [:SOURce]:RADio:ZIGBee:TRIGger:DELay:SAMPle?
DESCRIPTION	This command sets/queries the trigger delay samples of ZigBee external trigger.
DATA TYPE	Integer
RANGE	0 ~ 100,000,000
RETURN	Integer
DEFAULT VALUE	0
EQUIVALENT MENU	IQ > IoT > Protocol Type(ZigBee) > Trigger > Trigger Source(Ext) > Delay Samples
EXAMPLE	*:RADio:ZIGBee:TRIGger:DELay:SAMPle 1000* *:RADio:ZIGBee:TRIGger:DELay:SAMPle?* Return: *1000\n*

3.4.16.2.40 ZigBee Bus Trigger ([:SOURce]:RADio:ZIGBee:TRG)

SYNTAX	[:SOURce]:RADio:ZIGBee:TRG
DESCRIPTION	This command sends a trigger signal of the ZigBee bus trigger source.
EQUIVALENT MENU	None
EXAMPLE	*:RADio:ZIGBee:TRG*

3.4.16.2.41 ZigBee Update ([:SOURce]:RADio:ZIGBee:WAVeform:UPDate)

SYNTAX	[:SOURce]:RADio:ZIGBee:WAVeform:UPDate
DESCRIPTION	This command updates the settings for ZigBee modulation.
EQUIVALENT MENU	IQ > IoT > Protocol Type (ZigBee) > Update
EXAMPLE	*:RADio:ZIGBee:WAVeform:UPDate*

3.4.16.3 Z-Wave

3.4.16.3.1 Save Waveform ([:SOURce]:RADio:ZWAVe:SAVE)

SYNTAX	[:SOURce]:RADio:ZWAVe:SAVE <"file_name">
DESCRIPTION	This command saves Z-Wave modulated waveform data to an ARB file.
DATA TYPE	String
RANGE	Please refer to the user manual for naming rules.

EQUIVALENT MENU	IQ > IoT > Protocol Type(Z-Wave) > Save Waveform
EXAMPLE	*:RADio:ZWAVe:SAVE “test_zwave.arb”*

3.4.16.3.2 Z-Wave State ([:SOURce]:RADio:ZWAVe:STATe)

SYNTAX	[:SOURce]:RADio:ZWAVe:STATe ON|OFF|1|0 [:SOURce]:RADio:ZWAVe:STATe?
DESCRIPTION	This command sets/queries the switch status of Z-Wave modulation.
DATA TYPE	Boolean
RANGE	ON|OFF|1|0
RETURN	1|0
DEFAULT VALUE	0
EQUIVALENT MENU	IQ > IoT > Protocol Type(Z-Wave) > Z-Wave State
EXAMPLE	*:RADio:ZWAVe:STATe ON* *:RADio:ZWAVe:STATe?* Return: *1\n*

3.4.16.3.3 OverSampling Ratio ([:SOURce]:RADio:ZWAVe:OSAMple:RATio)

SYNTAX	[:SOURce]:RADio:ZWAVe:OSAMple:RATio <val> [:SOURce]:RADio:ZWAVe:OSAMple:RATio?
DESCRIPTION	This command sets/queries the oversampling rate of Z-Wave modulated wave.
DATA TYPE	Integer
RANGE	2 ~ 64
RETURN	Integer
DEFAULT VALUE	8
EQUIVALENT MENU	IQ > IoT > Protocol Type(Z-Wave) > Basic > OverSampling Ratio
EXAMPLE	*:RADio:ZWAVe:OSAMple:RATio 10* *:RADio:ZWAVe:OSAMple:RATio?* Return: *10\n*

3.4.16.3.4 Number of Frames ([:SOURce]:RADio:ZWAVe:FRAMe:NUMBer)

SYNTAX	[:SOURce]:RADio:ZWAVe:FRAMe:NUMBer <val> [:SOURce]:RADio:ZWAVe:FRAMe:NUMBer?
DESCRIPTION	This command sets and gets the number of frames contained in the Z-Wave modulated wave.
DATA TYPE	Integer
RANGE	1 ~ 2000

RETURN	Integer
DEFAULT VALUE	1
EQUIVALENT MENU	IQ > IoT > Protocol Type(Z-Wave) > Basic > Number of Frames
EXAMPLE	*:RADio:ZWAVe:FRAMe:NUMBer 2* *:RADio:ZWAVe:FRAMe:NUMBer?* Return: *2\n*

3.4.16.3.5 Total Sample Points ([:SOURce]:RADio:ZWAVe:TOTal:SAMPle:POINts?)

SYNTAX	[:SOURce]:RADio:ZWAVe:TOTal:SAMPle:POINts?
DESCRIPTION	This command queries the total number of sampling points of the Z-Wave modulated wave.
RETURN	Integer
DEFAULT VALUE	None
EQUIVALENT MENU	IQ > IoT > Protocol Type(Z-Wave) > Basic > Total Sample Points
EXAMPLE	*:RADio:ZWAVe:TOTal:SAMPle:POINts?* Return: *17720\n*

3.4.16.3.6 Waveform Length ([:SOURce]:RADio:ZWAVe:WAVeform:LENgth?)

SYNTAX	[:SOURce]:RADio:ZWAVe:WAVeform:LENgth?
DESCRIPTION	This command queries the waveform length of the Z-Wave modulated wave.
RETURN	Float, unit: s
DEFAULT VALUE	None
EQUIVALENT MENU	IQ > IoT > Protocol Type(Z-Wave) > Basic > Waveform Length
EXAMPLE	*:RADio:ZWAVe:WAVeform:LENgth?* Return: *0.0922916666666667\n*

3.4.16.3.7 Data Rate ([:SOURce]:RADio:ZWAVe:RATE:TYPE)

SYNTAX	[:SOURce]:RADio:ZWAVe:RATE:TYPE R1|R2|R3 [:SOURce]:RADio:ZWAVe:RATE:TYPE?
DESCRIPTION	This command sets/queries the data rate of Z-Wave modulation.
DATA TYPE	Enumeration
RANGE	R1|R2|R3
RETURN	Enumeration
DEFAULT VALUE	R1
EQUIVALENT MENU	IQ > IoT > Protocol Type(Z-Wave) > General Settings > Data Rate

EXAMPLE	*:RADio:ZWAVe:RATE:TYPE R2* *:RADio:ZWAVe:RATE:TYPE?* Return: *R2\n*

3.4.16.3.8 Modulation Type ([:SOURce]:RADio:ZWAVe:MODUlation:TYPE)

SYNTAX	[:SOURce]:RADio:ZWAVe:MODUlation:TYPE FSK | GFSK [:SOURce]:RADio:ZWAVe:MODUlation:TYPE?
DESCRIPTION	This command sets/queries the modulation type of Z-Wave modulation.
DATA TYPE	Enumeration
RANGE	FSK | GFSK
RETURN	Enumeration
DEFAULT VALUE	R1/R2 is FSK and R3 is GFSK.
EQUIVALENT MENU	IQ > IoT > Protocol Type(Z-Wave) > General Settings > Modulation
EXAMPLE	*:RADio:ZWAVe:MODUlation:TYPE GFSK* *:RADio:ZWAVe:MODUlation:TYPE?* Return: *GFSK\n*

3.4.16.3.9 Idle Interval ([:SOURce]:RADio:ZWAVe:IDLE)

SYNTAX	[:SOURce]:RADio:ZWAVe:IDLE <val> [:SOURce]:RADio:ZWAVe:IDLE?
DESCRIPTION	This command sets/queries the idle interval (in seconds) between frames contained in the Z-Wave modulated wave.
DATA TYPE	Float, unit: ns, us, ms or s, default is s
RANGE	0 ~ 200 ms
RETURN	Float, unit: s
DEFAULT VALUE	100 us
EQUIVALENT MENU	IQ > IoT > Protocol Type(Z-Wave) > General Settings > Idle Interval
EXAMPLE	*:RADio:ZWAVe:IDLE 10 ms* *:RADio:ZWAVe:IDLE?* Return: *0.01\n*

3.4.16.3.10 Ramp Symbols ([:SOURce]:RADio:ZWAVe:RAMP:SYMBol)

SYNTAX	[:SOURce]:RADio:ZWAVe:RAMP:SYMBol <val> [:SOURce]:RADio:ZWAVe:RAMP:SYMBol?
DESCRIPTION	This command sets/queries the number of sustained symbols for the rising and falling slopes of the Z-Wave modulated waveform.

DATA TYPE	Integer
RANGE	1 ~ 10
RETURN	Integer
DEFAULT VALUE	10
EQUIVALENT MENU	IQ > IoT > Protocol Type(Z-Wave) > Basic > Ramp Symbols
EXAMPLE	*:RADio:ZWAVe:RAMP:SYMBol 6* *:RADio:ZWAVe:RAMP:SYMBol?* Return: *6\n*

3.4.16.3.11 Ramp Up/Down Symbol ([:SOURce]:RADio:ZWAVe:RAMP:MODE)

SYNTAX	[:SOURce]:RADio:ZWAVe:RAMP:MODE FIRLast|CENTer|ONE|ZERO [:SOURce]:RADio:ZWAVe:RAMP:MODE?
DESCRIPTION	This command selects/queries the symbol type of the rising and falling periods of the Z-Wave modulated waveform ramp.
DATA TYPE	Enumeration
RANGE	FIRLast|CENTer|ONE|ZERO
RETURN	Enumeration
DEFAULT VALUE	FIRLast
EQUIVALENT MENU	IQ > IoT > Protocol Type(Z-Wave) > General Settings > Ramp Up/Down Symbol
EXAMPLE	*:RADio:ZWAVe:RAMP:MODE CENTer* *:RADio:ZWAVe:RAMP:MODE?* Return: *CENTer\n*

3.4.16.3.12 PPDU Length ([:SOURce]:RADio:ZWAVe:PPDU:LENgth?)

SYNTAX	[:SOURce]:RADio:ZWAVe:PPDU:LENgth?
DESCRIPTION	This command queries the PPDU byte length of Z-Wave modulation.
RETURN	Integer, unit: octets
DEFAULT VALUE	54
EQUIVALENT MENU	IQ > IoT > Protocol Type(Z-Wave) > PPDU Settings > PPDU Length
EXAMPLE	*:RADio:ZWAVe:PPDU:LENgth?* Return: *53\n*

3.4.16.3.13 Preamble Length ([:SOURce]:RADio:ZWAVe:PREAmble:LENgth)

SYNTAX	[:SOURce]:RADio:ZWAVe:PREAmble:LENgth <val>

	[:SOURce]:RADio:ZWAVe:PREAmble:LENgth?
DESCRIPTION	This command sets/gets the preamble length of Z-Wave PPDU.
DATA TYPE	Integer, unit: octets
RANGE	10 ~ 100
RETURN	Integer, unit: octets
DEFAULT VALUE	10
EQUIVALENT MENU	IQ > IoT > Protocol Type(Z-Wave) > PPDU Settings > Preamble Length
EXAMPLE	*:RADio:ZWAVe:PREAmble:LENgth 12* *:RADio:ZWAVe:PREAmble:LENgth?* Return: *12\n*

3.4.16.3.14 Start of Frame Delimiter ([:SOURce]:RADio:ZWAVe:SOF)

SYNTAX	[:SOURce]:RADio:ZWAVe:SOF <val> [:SOURce]:RADio:ZWAVe:SOF?
DESCRIPTION	This command sets/gets the frame start field (decimal) of Z-Wave PPDU.
DATA TYPE	Integer
RANGE	0 ~ 255
RETURN	Integer
DEFAULT VALUE	240
EQUIVALENT MENU	IQ > IoT > Protocol Type(Z-Wave) > PPDU Settings > Start of Frame Delimiter(Hex)
EXAMPLE	*:RADio:ZWAVe:SOF 100* *:RADio:ZWAVe:SOF?* Return: *100\n*

3.4.16.3.15 MAC Header State ([:SOURce]:RADio:ZWAVe:MAC:HEADer:STATe)

SYNTAX	[:SOURce]:RADio:ZWAVe:MAC:HEADer:STATe ON|OFF|1|0 [:SOURce]:RADio:ZWAVe:MAC:HEADer:STATe?
DESCRIPTION	This command sets/queries the enable status of the MAC Header field of Z-Wave PPDU.
DATA TYPE	Boolean
RANGE	ON|OFF|1|0
RETURN	1|0
DEFAULT VALUE	1
EQUIVALENT MENU	IQ > IoT > Protocol Type(Z-Wave) > PPDU Settings > MAC Header

EXAMPLE	*:RADio:ZWAVe:MAC:HEADer:STATe OFF* *:RADio:ZWAVe:MAC:HEADer:STATe?* Return: *0\n*

3.4.16.3.16 MAC Header ([:SOURce]:RADio:ZWAVe:MAC:HEADer)

SYNTAX	[:SOURce]:RADio:ZWAVe:MAC:HEADer <"header"> [:SOURce]:RADio:ZWAVe:MAC:HEADer?
DESCRIPTION	This command sets/queries the MAC Header field of Z-Wave frame.
DATA TYPE	String
RANGE	None
RETURN	String
DEFAULT VALUE	12345678,01,3412,2A,01,01
EQUIVALENT MENU	IQ > IoT > Protocol Type(Z-Wave) > PPDU Settings > MAC Header
EXAMPLE	*:RADio:ZWAVe:MAC:HEADer "12345678,01,3412,2B,01,AB"* *:RADio:ZWAVe:MAC:HEADer?* Return: *12345678,01,3412,2A,01,AB\n*

3.4.16.3.17 MAC Header Sequence Number State ([:SOURce]:RADio:ZWAVe:MAC:HEADer:SEQUence:STATe)

SYNTAX	[:SOURce]:RADio:ZWAVe:MAC:HEADer:SEQUence:STATe ON|OFF|1|0 [:SOURce]:RADio:ZWAVe:MAC:HEADer:SEQUence:STATe?
DESCRIPTION	This command sets/queries the enable status of the Sequence Number field in the Z-Wave PPDU MAC Header field.
DATA TYPE	Boolean
RANGE	ON|OFF|1|0
RETURN	1|0
DEFAULT VALUE	0
EQUIVALENT MENU	IQ > IoT > Protocol Type(Z-Wave) > PPDU Settings > MAC Header > Sequence Number
EXAMPLE	*:RADio:ZWAVe:MAC:HEADer:SEQUence:STATe ON* *:RADio:ZWAVe:MAC:HEADer:SEQUence:STATe?* Return: *1\n*

3.4.16.3.18 Data Type ([:SOURce]:RADio:ZWAVe:DATA:TYPE)

SYNTAX	[:SOURce]:RADio:ZWAVe:DATA:TYPE PN9 | PN15 | USER [:SOURce]:RADio:ZWAVe:DATA:TYPE?
DESCRIPTION	This command sets/queries the data type of Z-Wave PPDU.

DATA TYPE	Enumeration
RANGE	PN9 | PN15 | USER
RETURN	Enumeration
DEFAULT VALUE	PN9
EQUIVALENT MENU	IQ > IoT > Protocol Type(Z-Wave) > PPDU Settings > Data Type
EXAMPLE	*:RADio:ZWAVe:DATA:TYPE PN15* *:RADio:ZWAVe:DATA:TYPE?* Return: *PN15\n*

3.4.16.3.19 Seed ([:SOURce]:RADio:ZWAVe:PN:SEED)

SYNTAX	[:SOURce]:RADio:ZWAVe:PN:SEED <val> [:SOURce]:RADio:ZWAVe:PN:SEED?
DESCRIPTION	This command sets the seed that generates the PN sequence.
DATA TYPE	Integer
RANGE	1 ~ 32767
RETURN	Integer
DEFAULT VALUE	511
EQUIVALENT MENU	IQ > IoT > Protocol Type(Z-Wave) > PPDU Settings > Seed(Hex)
EXAMPLE	*:RADio:ZWAVe:PN:SEED 974* *:RADio:ZWAVe:PN:SEED?* Return: *974\n*

3.4.16.3.20 Custom PN Data ([:SOURce]:RADio:ZWAVe:PN:USER:DATA)

SYNTAX	[:SOURce]:RADio:ZWAVe:PN:USER:DATA <user_data>
DESCRIPTION	This command sets custom PN data for the Z-Wave PPDU Payload field.
DATA TYPE	Binary string
RANGE	None
EQUIVALENT MENU	IQ > IoT > Protocol Type(Z-Wave) > PPDU Settings > Data Type (User) > User Data
EXAMPLE	*:RADio:ZWAVe:PN:USER:DATA 0101001100111*

3.4.16.3.21 Get PN Data ([:SOURce]:RADio:ZWAVe:PN:DATA?)

SYNTAX	[:SOURce]:RADio:ZWAVe:PN:DATA? PN9 | PN15 | USER
DESCRIPTION	This command queries the PN data of the Z-Wave PPDU Payload field.

DATA TYPE	Enumeration
RANGE	PN9 | PN15 | USER
RETURN	Binary string
DEFAULT VALUE	None
EQUIVALENT MENU	IQ > IoT > Protocol Type(Z-Wave) > PPDU Settings > Data Type (User) > User Data
EXAMPLE	*:RADio:ZIGBee:PN:DATA? PN9* Return: *111111111000001111011111000101110011001000001001010011101* *101000111100111110011011000101010010001110001101101010111* *0001001100010001000000000100001000110000100111001010101100* *001101111010011011100100010100001010110100111111011001001* *001011011111100100110101001100110000000110001100101000110* *100101111111010001011000111010110010110011110001111101110* *100000110101101101110110000010110101111101010101000000101* *001010111100101110111000000111001110100100111101011101010* *0010010000110011100001011110110110011010000111011110000\n*

3.4.16.3.22 Store User PN Data ([:SOURce]:RADio:ZWAVe:PN:SAVE)

SYNTAX	[:SOURce]:RADio:ZWAVe:PN:SAVE <"file_name">
DESCRIPTION	This command saves the customized PN data of the Z-Wave PPDU Payload field into the UDATA file.
DATA TYPE	String
RANGE	Please refer to the user manual for naming rules.
EQUIVALENT MENU	IQ > IoT > Protocol Type(Z-Wave) > PPDU Settings > Data Type (User) > User Data > Save
EXAMPLE	*:RADio:ZWAVe:PN:SAVE "user_pn.udata"*

3.4.16.3.23 Load User PN Data ([:SOURce]:RADio:ZWAVe:PN:LOAD)

SYNTAX	[:SOURce]:RADio:ZWAVe:PN:LOAD <"file_name">
DESCRIPTION	This command loads the custom PN data of the Z-Wave PPDU Payload field from the UDATA file.
DATA TYPE	String
RANGE	None
EQUIVALENT MENU	IQ > IoT > Protocol Type(Z-Wave) > PPDU Settings > Data Type (User) > User Data > Load
EXAMPLE	*:RADio:ZWAVe:PN:LOAD "user_pn.udata"*

3.4.16.3.24 Data Length ([:SOURce]:RADio:ZWAVe:DATA:LENgth)

SYNTAX	[:SOURce]:RADio:ZWAVe:DATA:LENgth <val>

	[:SOURce]:RADio:ZWAVe:DATA:LENgth?
DESCRIPTION	This command sets/queries the data length of the Z-Wave PPDU Payload field.
DATA TYPE	Integer, unit: octets
RANGE	Depends on the maximum length of the PSDU and MSDU fields
RETURN	Integer, unit: octets
DEFAULT VALUE	32
EQUIVALENT MENU	IQ > IoT > Protocol Type(Z-Wave) > PPDU Settings > Data Length
EXAMPLE	*:RADio:ZWAVe:DATA:LENgth 10* *:RADio:ZWAVe:DATA:LENgth?* Return: *10\n*

3.4.16.3.25 Data Mode ([:SOURce]:RADio:ZWAVe:CONTinuous:STATe)

SYNTAX	[:SOURce]:RADio:ZWAVe:CONTinuous:STATe ON|OFF|1|0 [:SOURce]:RADio:ZWAVe:CONTinuous:STATe?
DESCRIPTION	This command enables or disables data continuity status. Continuous mode will have the data bits spread out continuously over multiple frames, truncated mode will have the same payload data bits for all frames.
DATA TYPE	Boolean
RANGE	ON|OFF|1|0
RETURN	1|0
DEFAULT VALUE	0
EQUIVALENT MENU	IQ > IoT > Protocol Type(Z-Wave) > PPDU Settings > Data Mode
EXAMPLE	*:RADio:ZWAVe:CONTinuous:STATe 1* *:RADio:ZWAVe:CONTinuous:STATe?* Return: *1\n*

3.4.16.3.26 MAC FCS State ([:SOURce]:RADio:ZWAVe:FCS:STATe)

SYNTAX	[:SOURce]:RADio:ZWAVe:FCS:STATe ON|OFF|1|0 [:SOURce]:RADio:ZWAVe:FCS:STATe?
DESCRIPTION	This command enables or disables MAC FCS in the PSDU. When turned off, it can be used to simulate an invalid FCS condition since the FCS section is actually filled with user data bits.
DATA TYPE	Boolean
RANGE	ON|OFF|1|0
RETURN	1|0
DEFAULT VALUE	1
EQUIVALENT MENU	IQ > IoT > Protocol Type(Z-Wave) > PPDU Settings > MAC FCS

EXAMPLE	*:RADio:ZWAVe:FCS:STATe 0* *:RADio:ZWAVe:FCS:STATe?* Return: *0\n*

3.4.16.3.27 End of Frame Delimiter ([:SOURce]:RADio:ZWAVe:EOF:STATe)

SYNTAX	[:SOURce]:RADio:ZWAVe:EOF:STATe ON|OFF|1|0 [:SOURce]:RADio:ZWAVe:EOF:STATe?
DESCRIPTION	Enable or disable EHR in PPDU. For data rates R2 and R3, this setting is off and read-only.
DATA TYPE	Boolean
RANGE	ON|OFF|1|0
RETURN	1|0
DEFAULT VALUE	1
EQUIVALENT MENU	IQ > IoT > Protocol Type(Z-Wave) > PPDU Settings > End of Frame Delimiter
EXAMPLE	*:RADio:ZWAVe:EOF:STATe 0* *:RADio:ZWAVe:EOF:STATe?* Return: *0\n*

3.4.16.3.28 Symbol Timing Error ([:SOURce]:RADio:ZWAVe:SYMBle:TIMing:ERRor)

SYNTAX	[:SOURce]:RADio:ZWAVe:SYMBle:TIMing:ERRor <val> [:SOURce]:RADio:ZWAVe:SYMBle:TIMing:ERRor?
DESCRIPTION	This command sets the shift to the standard symbol rate in ppm.
DATA TYPE	Integer, unit: ppm
RANGE	-300 ~ 300
RETURN	Integer, unit: ppm
DEFAULT VALUE	0
EQUIVALENT MENU	IQ > IoT > Protocol Type(Z-Wave) > Impairments > Symbol Timing Error
EXAMPLE	*:RADio:ZWAVe:SYMBle:TIMing:ERRor -10* *:RADio:ZWAVe:SYMBle:TIMing:ERRor?* Return: *-10\n*

3.4.16.3.29 Frequency Offset ([:SOURce]:RADio:ZWAVe:FREQ:OFFSet)

SYNTAX	[:SOURce]:RADio:ZWAVe:FREQ:OFFSet <val> [:SOURce]:RADio:ZWAVe:FREQ:OFFSet?
DESCRIPTION	This command sets or gets the offset (in Hz) of the nominal carrier frequency for Z-Wave modulation.
DATA TYPE	Float, unit: Hz

RANGE	-200000 ~ 200000
RETURN	Float, unit: Hz
DEFAULT VALUE	0
EQUIVALENT MENU	IQ > IoT > Protocol Type(Z-Wave) > Impairments > Frequency Offset
EXAMPLE	*:RADio:ZWAVe:FREQ:OFFSet 1000* *:RADio:ZWAVe:FREQ:OFFSet?* Return: *1000\n*

3.4.16.3.30 Frequency Deviation Scaling ([:SOURce]:RADio:ZWAVe:FREQ:DEVIation:SCALe)

SYNTAX	[:SOURce]:RADio:ZWAVe:FREQ:DEVIation:SCALe <val> [:SOURce]:RADio:ZWAVe:FREQ:DEVIation:SCALe?
DESCRIPTION	Set the additional scale to the nominal FSK frequency deviation, which is equivalent to applying scaling to the FSK modulation index.
DATA TYPE	Float
RANGE	0.5 ~ 1.5
RETURN	Float
DEFAULT VALUE	1
EQUIVALENT MENU	IQ > IoT > Protocol Type(Z-Wave) > Impairments > Frequency Deviation Scaling
EXAMPLE	*:RADio:ZWAVe:FREQ:DEVIation:SCALe 0.8* *:RADio:ZWAVe:FREQ:DEVIation:SCALe?* Return: *0.8\n*

3.4.16.3.31 Gaussian BT ([:SOURce]:RADio:ZWAVe:GAUSsian:BT)

SYNTAX	[:SOURce]:RADio:ZWAVe:GAUSsian:BT <val> [:SOURce]:RADio:ZWAVe:GAUSsian:BT?
DESCRIPTION	This command sets or gets the BT product of the Gaussian filter applied to FSK modulation.
DATA TYPE	Float
RANGE	0.1 ~ 10
RETURN	Float
DEFAULT VALUE	0.6
EQUIVALENT MENU	IQ > IoT > Protocol Type(Z-Wave) > Impairments > Gaussian BT
EXAMPLE	*:RADio:ZWAVe:GAUSsian:BT 0.8* *:RADio:ZWAVe:GAUSsian:BT?* Return: *0.8\n*

3.4.16.3.32 Marker 1 Source ([:SOURce]:RADio:ZWAVe:MARKer:ONE:SOURce)

SYNTAX	[:SOURce]:RADio:ZWAVe:MARKer:ONE:SOURce WAVeform| FRAMe [:SOURce]:RADio:ZWAVe:MARKer:ONE:SOURce?
DESCRIPTION	This command sets/queries the source of Mark 1 of the Z-Wave modulated wave. Waveform Start - Indicates the start of the waveform, Frame Start - Indicates the beginning of each frame in the waveform.
DATA TYPE	Enumeration
RANGE	WAVeform| FRAMe
RETURN	Enumeration
DEFAULT VALUE	FRAMe
EQUIVALENT MENU	IQ > IoT > Protocol Type(Z-Wave) > Marker > Marker 1 Source
EXAMPLE	*:RADio:ZWAVe:MARKer:ONE:SOURce WAVeform* *:RADio:ZWAVe:MARKer:ONE:SOURce?* Return: *WAVeform\n*

3.4.16.3.33 Marker 2 Soruce ([:SOURce]:RADio:ZWAVe:MARKer:TWO:SOURce)

SYNTAX	[:SOURce]:RADio:ZWAVe:MARKer:TWO:SOURce WAVeform| FRAMe [:SOURce]:RADio:ZWAVe:MARKer:TWO:SOURce?
DESCRIPTION	This command sets/queries the source of Mark 2 of the Z-Wave modulated wave. Waveform Start - Indicates the start of the waveform, Frame Start - Indicates the beginning of each frame in the waveform.
DATA TYPE	Enumeration
RANGE	WAVeform| FRAMe
RETURN	Enumeration
DEFAULT VALUE	FRAMe
EQUIVALENT MENU	IQ > IoT > Protocol Type(Z-Wave) > Marker > Marker 2 Source
EXAMPLE	*:RADio:ZWAVe:MARKer:TWO:SOURce WAVeform* *:RADio:ZWAVe:MARKer:TWO:SOURce?* Return: *WAVeform\n*

3.4.16.3.34 Z-Wave Trigger Type ([:SOURce]:RADio:ZWAVe:TRIGger:TYPE)

SYNTAX	[:SOURce]:RADio:ZWAVe:TRIGger:TYPE CONTinous| SINGle| GATE [:SOURce]:RADio:ZWAVe:TRIGger:TYPE?
DESCRIPTION	This command sets/queries the trigger type of Z-Wave.
DATA TYPE	Enumeration
RANGE	CONTinous| SINGle| GATE

RETURN	Enumeration
DEFAULT VALUE	CONTinous
EQUIVALENT MENU	IQ > IoT > Protocol Type(Z-Wave) > Trigger > Trigger Type
EXAMPLE	*:RADio:ZWAVe:TRIGger:TYPE SINGle* *:RADio:ZWAVe:TRIGger:TYPE?* Return: *SINGle\n*

3.4.16.3.35 Z-Wave Trigger Continuous Mode ([:SOURce]:RADio:ZWAVe:TRIGger:CONTinous)

SYNTAX	[:SOURce]:RADio:ZWAVe:TRIGger:CONTinous FREErun | RUNIgnored [:SOURce]:RADio:ZWAVe:TRIGger:CONTinous?
DESCRIPTION	This command sets/queries the trigger mode of Z-Wave continuous trigger.
DATA TYPE	Enumeration
RANGE	FREErun | RUNIgnored
RETURN	Enumeration
DEFAULT VALUE	FREErun
EQUIVALENT MENU	IQ > IoT > Protocol Type(Z-Wave) > Trigger > Trigger Type(Continuous) > Continuous Mode
EXAMPLE	*:RADio:ZWAVe:TRIGger:CONTinous RUNIgnored* *:RADio:ZWAVe:TRIGger:CONTinous?* Return: *RUNIgnored\n*

3.4.16.3.36 Z-Wave Trigger Gate Mode ([:SOURce]:RADio:ZWAVe:TRIGger:GATE)

SYNTAX	[:SOURce]:RADio:ZWAVe:TRIGger:GATE LOW | HIGH [:SOURce]:RADio:ZWAVe:TRIGger:GATE?
DESCRIPTION	This command sets/queries the trigger mode of Z-Wave external gating trigger.
DATA TYPE	Enumeration
RANGE	LOW | HIGH
RETURN	Enumeration
DEFAULT VALUE	LOW
EQUIVALENT MENU	IQ > IoT > Protocol Type(Z-Wave) > Trigger > Trigger Type(Ext Gated) > Gate Mode
EXAMPLE	*:RADio:ZWAVe:TRIGger:GATE HIGH* *:RADio:ZWAVe:TRIGger:GATE?* Return: *HIGH\n*

3.4.16.3.37 Z-Wave Trigger Source ([:SOURce]:RADio:ZWAVe:TRIGger:SOURce)

SYNTAX	[:SOURce]:RADio:ZWAVe:TRIGger:SOURce KEY| BUS| EXT [:SOURce]:RADio:ZWAVe:TRIGger:SOURce?
DESCRIPTION	This command sets/queries the trigger source of Z-Wave trigger.
DATA TYPE	Enumeration
RANGE	KEY| BUS| EXT
RETURN	Enumeration
DEFAULT VALUE	KEY
EQUIVALENT MENU	IQ > IoT > Protocol Type(Z-Wave) > Trigger > Trigger Source
EXAMPLE	*:RADio:ZWAVe:TRIGger:SOURce BUS* *:RADio:ZWAVe:TRIGger:SOURce?* Return: *BUS\n*

3.4.16.3.38 Z-Wave Trigger Polarity ([:SOURce]:RADio:ZWAVe:TRIGger:POL)

SYNTAX	[:SOURce]:RADio:ZWAVe:TRIGger:POL POS | NEG [:SOURce]:RADio:ZWAVe:TRIGger:POL?
DESCRIPTION	This command sets/queries the trigger polarity of Z-Wave external trigger.
DATA TYPE	Enumeration
RANGE	POS | NEG
RETURN	Enumeration
DEFAULT VALUE	POS
EQUIVALENT MENU	IQ > IoT > Protocol Type(Z-Wave) > Trigger > Trigger Source(Ext) > Ext Polarity
EXAMPLE	*:RADio:ZWAVe:TRIGger:POL NEG* *:RADio:ZWAVe:TRIGger:POL?* Return: *NEG\n*

3.4.16.3.39 Z-Wave Trigger Delay Samples ([:SOURce]:RADio:ZWAVe:TRIGger:DELay:SAMPle)

SYNTAX	[:SOURce]:RADio:ZWAVe:TRIGger:DELay:SAMPle <samples> [:SOURce]:RADio:ZWAVe:TRIGger:DELay:SAMPle?
DESCRIPTION	This command sets/queries the trigger delay samples of Z-Wave external trigger.
DATA TYPE	Integer
RANGE	0 ~ 100,000,000
RETURN	Integer
DEFAULT VALUE	0

EQUIVALENT MENU	IQ > IoT > Protocol Type(Z-Wave) > Trigger > Trigger Source(Ext) > Delay Samples
EXAMPLE	*:RADio:ZWAVe:TRIGger:DELay:SAMPle 1000* *:RADio:ZWAVe:TRIGger:DELay:SAMPle?* Return: *1000\n*

3.4.16.3.40 Z-Wave Bus Trigger ([:SOURce]:RADio:ZWAVe:TRG)

SYNTAX	[:SOURce]:RADio:ZWAVe:TRG
DESCRIPTION	This command sends a trigger signal of the Z-Wave bus trigger source.
EQUIVALENT MENU	None
EXAMPLE	*:RADio:ZWAVe:TRG*

3.4.16.3.41 Z-Wave Update ([:SOURce]:RADio:ZWAVe:WAVeform:UPDate)

SYNTAX	[:SOURce]:RADio:ZWAVe:WAVeform:UPDate
DESCRIPTION	This command updates the settings for Z-Wave modulation.
EQUIVALENT MENU	IQ > IoT > Protocol Type (Z-Wave) > Update
EXAMPLE	*:RADio:ZWAVe:WAVeform:UPDate*

3.4.17 Multitone

3.4.17.1 Multitone State ([:SOURce]:RADio:MTONe:ARB[:STATe])

SYNTAX	[:SOURce]:RADio:MTONe:ARB[:STATe] ON|OFF|1|0 [:SOURce]:RADio:MTONe:ARB[:STATe]?
DESCRIPTION	This command sets/queries the switch status of multi-tone modulation.
DATA TYPE	Boolean
RANGE	ON|OFF|1|0
RETURN	1|0
DEFAULT VALUE	0
EQUIVALENT MENU	IQ > Multitone > Multitone State
EXAMPLE	*:RADio:MTONe:ARB ON* *:RADio:MTONe:ARB?* Return: *1\n*

3.4.17.2 Tone Number ([:SOURce]:RADio:MTONe:ARB:SETup:TABLe:NTONes)

SYNTAX	[:SOURce]:RADio:MTONe:ARB:SETup:TABLe:NTONes <num> [:SOURce]:RADio:MTONe:ARB:SETup:TABLe:NTONes?
DESCRIPTION	This command sets/queries the number of tones in the multi-tone waveform.
DATA TYPE	Integer
RANGE	1 ~ 20
RETURN	Integer
DEFAULT VALUE	2
EQUIVALENT MENU	IQ > Multitone > Tone Number
EXAMPLE	*:RADio:MTONe:ARB:SETup:TABLe:NTONes 5* *:RADio:MTONe:ARB:SETup:TABLe:NTONes?* Return: *5\n*

3.4.17.3 Single Side ([:SOURce]:RADio:MTONe:ARB:SETup:TABLe:SINGle)

SYNTAX	[:SOURce]:RADio:MTONe:ARB:SETup:TABLe:SINGle ON|OFF|1|0 [:SOURce]:RADio:MTONe:ARB:SETup:TABLe:SINGle?
DESCRIPTION	This command enables or disables multi-tone single-sided.
DATA TYPE	Boolean
RANGE	ON|OFF|1|0
RETURN	1|0
DEFAULT VALUE	0
EQUIVALENT MENU	IQ > Multitone > Single Side
EXAMPLE	*:RADio:MTONe:ARB:SETup:TABLe:SINGle ON* *:RADio:MTONe:ARB:SETup:TABLe:SINGle?* Return: *1\n*

3.4.17.4 Sample Rate ([:SOURce]:RADio:MTONe:ARB:SCLock:RATE)

SYNTAX	[:SOURce]:RADio:MTONe:ARB:SCLock:RATE <val> [:SOURce]:RADio:MTONe:ARB:SCLock:RATE?
DESCRIPTION	This command sets/queries the sampling clock rate for Multitone modulation.
DATA TYPE	Float, unit: Hz, kHz, MHz or GHz, default is Hz
RANGE	500 Hz ~ 240 MHz
RETURN	Float, unit: Hz
DEFAULT VALUE	2 MHz
EQUIVALENT MENU	IQ > Multitone > Sample Rate

EXAMPLE	*:RADio:MTONe:ARB:SCLock:RATE 4 MHz* *:RADio:MTONe:ARB:SCLock:RATE?* Return: *4000000\n*

3.4.17.5 Freq Spacing ([:SOURce]:RADio:MTONe:ARB:SETup:TABLe:FSPacing)

SYNTAX	[:SOURce]:RADio:MTONe:ARB:SETup:TABLe:FSPacing <val> [:SOURce]:RADio:MTONe:ARB:SETup:TABLe:FSPacing?
DESCRIPTION	This command sets/queries the frequency interval between total tones.
DATA TYPE	Float, unit: Hz, kHz, MHz or GHz, default is Hz
RANGE	Tone Number * Sample Rate /4096 ~ min(120MHz, Sample Rate /1.28)
RETURN	Float, unit: Hz
DEFAULT VALUE	1 MHz
EQUIVALENT MENU	IQ > Multitone > Freq Spacing
EXAMPLE	*:RADio:MTONe:ARB:SETup:TABLe:FSPacing 2 MHz* *:RADio:MTONe:ARB:SETup:TABLe:FSPacing?* Return: *2000000\n*

3.4.17.6 Save State ([:SOURce]:RADio:MTONe:ARB:SETup:STORe)

SYNTAX	[:SOURce]:RADio:MTONe:ARB:SETup:STORe <"file_name">
DESCRIPTION	This command stores the current multitone settings in a MULSTATE file.
DATA TYPE	String
RANGE	Please refer to the user manual for naming rules.
EQUIVALENT MENU	IQ > Multitone > Save State
EXAMPLE	*:RADio:MTONe:ARB:SETup:STORe "test.mulstate"*

3.4.17.7 Load State ([:SOURce]:RADio:MTONe:ARB:SETup)

SYNTAX	[:SOURce]:RADio:MTONe:ARB:SETup <"file_name">
DESCRIPTION	This command loads multitone waveform settings from a MULSTATE file.
DATA TYPE	String
RANGE	None
EQUIVALENT MENU	IQ > Multitone > Load State
EXAMPLE	*:RADio:MTONe:ARB:SETup "test.mulstate"*

3.4.18 AWGN

3.4.18.1 AWGN State ([:SOURce]:RADio:AWGN:RT[:STATe])

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

3.4.18.2 Bandwidth ([:SOURce]:RADio:AWGN:RT:BWIDth)

SYNTAX	[:SOURce]:RADio:AWGN:RT:BWIDth <bandwidth> [:SOURce]:RADio:AWGN:RT:BWIDth?
DESCRIPTION	This command sets/queries the bandwidth of the real-time AWGN.
DATA TYPE	Float, unit: Hz
RANGE	1 Hz ~ 150 MHz
RETURN	Float, unit: Hz
DEFAULT VALUE	10 MHz
EQUIVALENT MENU	IQ > AWGN > Bandwidth
EXAMPLE	*:RADio:AWGN:RT:BWIDth 5000000* *:RADio:AWGN:RT:BWIDth?* Return: *5000000\n*

3.5 SENSe Commands

3.5.1 Power Sensor

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3.5.1.10 Statistics Clear (:SENSe[:POWer]:STATistics:CLEar)

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

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

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

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

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

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

SYNTAX	:SENSe[:POWer]:LIMit <power> :SENSe[:POWer]:LIMit?
DESCRIPTION	This command sets/queries the Limit level of the power sensor level control.
DATA TYPE	Float, unit: dBm, dBuV, uV, mV, V, nW, uW, mW or W, default is dBm
RANGE	-120 dBm ~ 20 dBm

RETURN	Float, unit: dBm
DEFAULT VALUE	0
EQUIVALENT SCPI	[:SOURce]:POWer:LIMit <power> [:SOURce]:POWer:LIMit?
EQUIVALENT MENU	HOME > POWER SENSOR > Level Control > Limit Level
EXAMPLE	*SENSe:LIMit 2* *SENSe:LIMit?* Return: *2\n*

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

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

3.5.1.15 Auto Zero (:CALibration:ZERO:TYPE)

SYNTAX	:CALibration:ZERO:TYPE OFF|INTernal|EXTernal :CALibration:ZERO:TYPE?
DESCRIPTION	This command sets/queries the automatic zero adjustment type of the power sensor.
DATA TYPE	Enumeration
RANGE	OFF|INTernal|EXTernal
RETURN	Enumeration
DEFAULT VALUE	OFF
EQUIVALENT MENU	HOME > POWER SENSOR > Auto Zero
EXAMPLE	*:CALibration:ZERO:TYPE EXTernal* *:CALibration:ZERO:TYPE?* Return: *EXTernal\n*

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

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

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

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

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

SYNTAX	:SENSe[:POWer]:FREQuency <freq> :SENSe[:POWer]:FREQuency?
DESCRIPTION	This command sets/queries the manual measurement frequency of the power sensor.
DATA TYPE	Float, unit: Hz, kHz, MHz or GHz, default is Hz
RANGE	Power sensor measurement range
RETURN	Float, unit: Hz
DEFAULT VALUE	None
EQUIVALENT MENU	HOME > POWER SENSOR > Frequency
EXAMPLE	*SENSe:FREQuency 1 MHz* *SENSe:FREQuency?* Return: *1000000\n*

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

SYNTAX	:SENSe[:POWer]:OFFSet:STATe ON|OFF|1|0 :SENSe[:POWer]:OFFSet:STATe?

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

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

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

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

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

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

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

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

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

3.6 MEMory Commands

3.6.1 ARB

3.6.1.1 Save Segment (:MEMory:COPY[:NAME])

SYNTAX	:MEMory:COPY[:NAME] <“segment”>,<“file_name”>
DESCRIPTION	Copy a waveform segment file from volatile memory to non-volatile memory
DATA TYPE	segment: string. Name of the segment to be saved. file_name: string. This variable names the destination file and the directory path. The directory path is not needed. The path "Local/" is implied.
RANGE	segment: Waveform name in IQ > ARB > Waveform Segment, file_name: ARB file name to save. Please refer to the user manual for naming rules.
EQUIVALENT MENU	IQ > ARB > Waveform Segment > Save
EXAMPLE	*:MEMory:COPY “SINE_WAVE”,“SINE_test.arb”*

3.6.1.2 Create Segment (:MEMory:DATA)

SYNTAX	:MEMory:DATA <“segment”>,<data_block>
DESCRIPTION	This command loads waveform data into the volatile memory of the signal generator using the <data_block> parameter, stores it as a waveform segment named <"segment">, and saves the waveform data to the WDbin file specified by the <"segment"> variable, with the directory path "Local/".
DATA TYPE	segment: string. This variable names the waveform segment and the waveform segment file. data_block: string. This parameter represents the waveform data.
RANGE	segment: please refer to the user manual for naming rules. data_block: none.
EQUIVALENT MENU	None

EXAMPLE	*:MEM:DATA "IQ_Data.arb",#14Y9oL*
	
	Note:
	#14Y9oL - Data block
	# - This character indicates the beginning of the data block
	1 - Number of digits in the byte count
	4 - Byte count
	Y9oL - 4 bytes of data (The data are not valid and are shown for example purposes only. Typically, ascii characters representing data are unprintable.)
	
	You can program it as follows:
	`with open(filename, 'rb') as f:`
	`    wave_data = f.read()`
	`    datalen = len(wave_data)`

```
    datalenlen = len(str(datalen))
    write_iq_data_cmd = '%s"%s",#%d%d' % (cmd, seg_name,
datalenlen, datalen)
    self.ssg.write_raw(write_iq_data_cmd.encode(encoding='utf-8')
+ wave_data)
```

4 Programming Examples

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

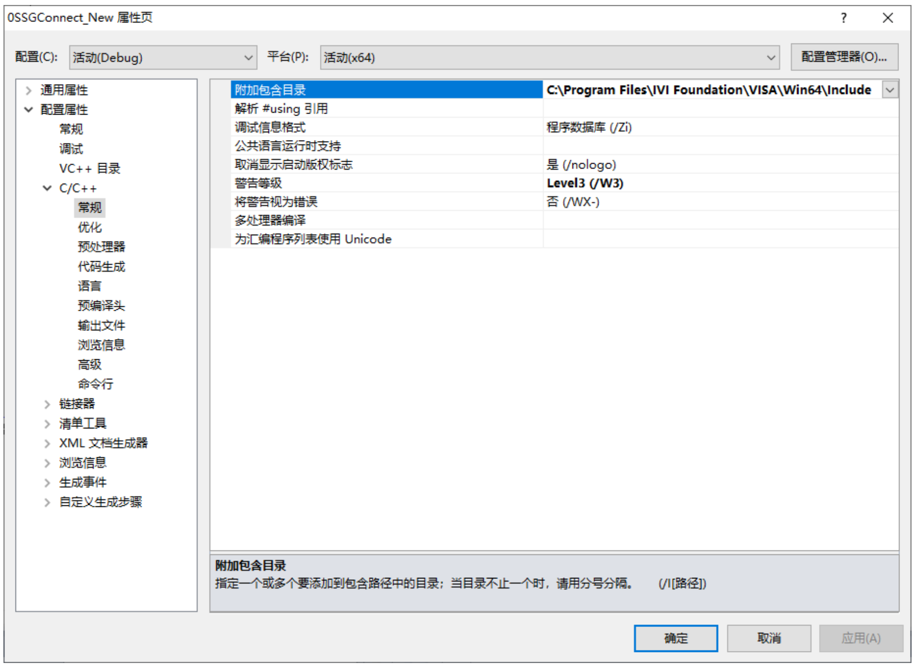
4.1 VISA Examples

4.1.1 VC++ Example

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

Please follow below steps to complete the example:

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


0SSGConnect_New 属性页

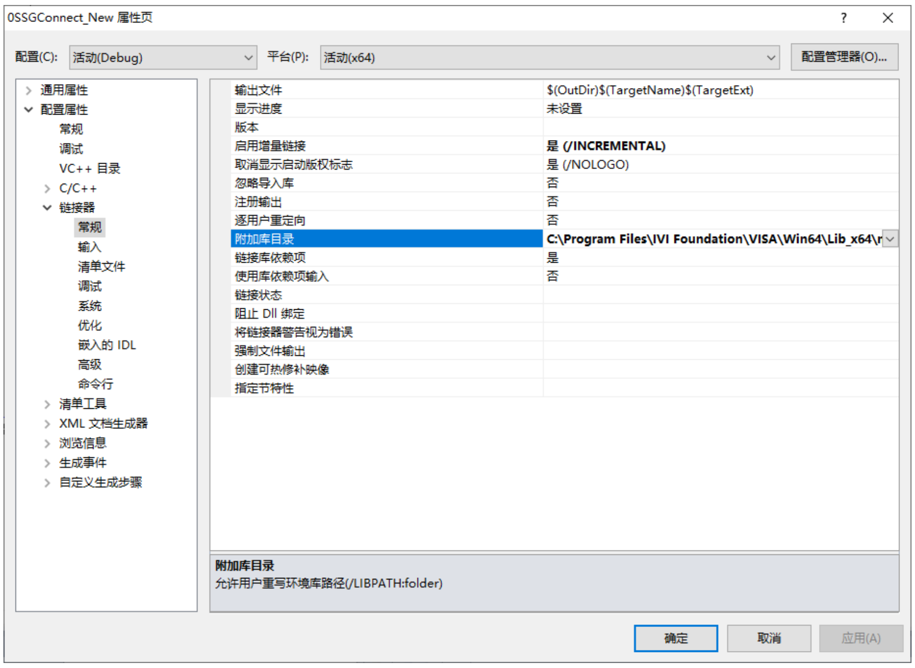
配置(C): 活动(Debug)　平台(P): 活动(x64)　配置管理器(O)...

- 通用属性
- 配置属性
  - 常规
  - 调试
  - VC++ 目录
  - C/C++
    - 常规
    - 优化
    - 预处理器
    - 代码生成
    - 语言
    - 预编译头
    - 输出文件
    - 浏览信息
    - 高级
    - 命令行
  - 链接器
  - 清单工具
  - XML 文档生成器
  - 浏览信息
  - 生成事件
  - 自定义生成步骤

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

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

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


0SSGConnect_New 属性页

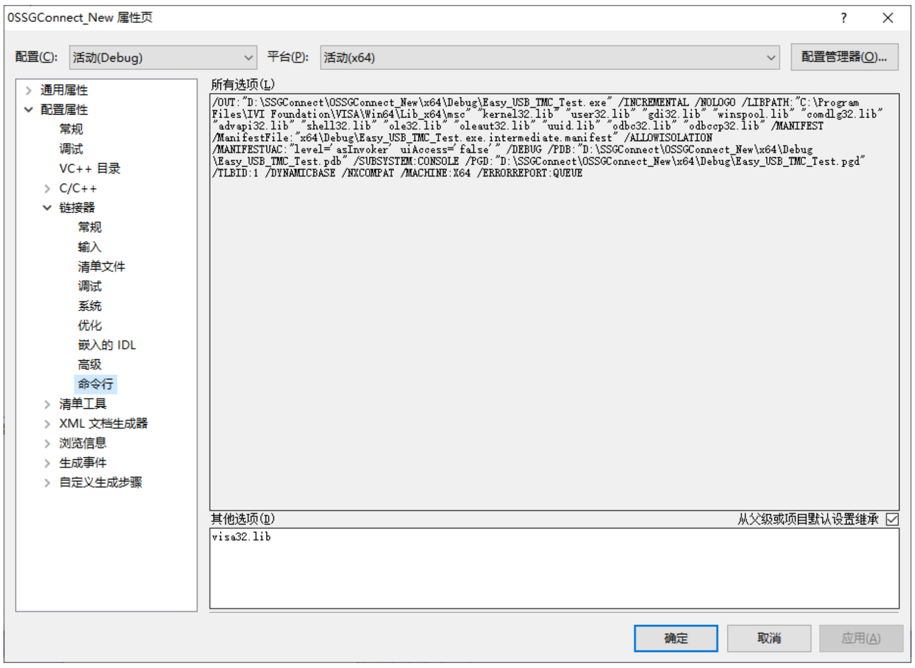
配置(C): 活动(Debug)　平台(P): 活动(x64)　配置管理器(O)...

- 通用属性
- 配置属性
  - 常规
  - 调试
  - VC++ 目录
  - C/C++
  - 链接器
    - 常规
    - 输入
    - 清单文件
    - 调试
    - 系统
    - 优化
    - 嵌入的 IDL
    - 高级
    - 命令行
  - 清单工具
  - XML 文档生成器
  - 浏览信息
  - 生成事件
  - 自定义生成步骤

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

附加库目录
允许用户重写环境库路径(/LIBPATH:folder)

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



```
/OUT:"D:\SSGConnect\0SSGConnect_New\x64\Debug\Easy_USB_TMC_Test.exe" /INCREMENTAL /NOLOGO /LIBPATH:"C:\Program
Files\IVI Foundation\VISA\Win64\Lib_x64\msc" "kernel32.lib" "user32.lib" "gdi32.lib" "winspool.lib" "comdlg32.lib"
"advapi32.lib" "shell32.lib" "ole32.lib" "oleaut32.lib" "uuid.lib" "odbc32.lib" "odbccp32.lib" /MANIFEST
/ManifestFile:"x64\Debug\Easy_USB_TMC_Test.exe.intermediate.manifest" /ALLOWISOLATION
/MANIFESTUAC:"level='asInvoker' uiAccess='false'" /DEBUG /PDB:"D:\SSGConnect\0SSGConnect_New\x64\Debug
\Easy_USB_TMC_Test.pdb" /SUBSYSTEM:CONSOLE /PGD:"D:\SSGConnect\0SSGConnect_New\x64\Debug\Easy_USB_TMC_Test.pgd"
/TLBID:1 /DYNAMICBASE /NXCOMPAT /MACHINE:X64 /ERRORREPORT:QUEUE
```

```
visa32.lib
```

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

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

3. Add code

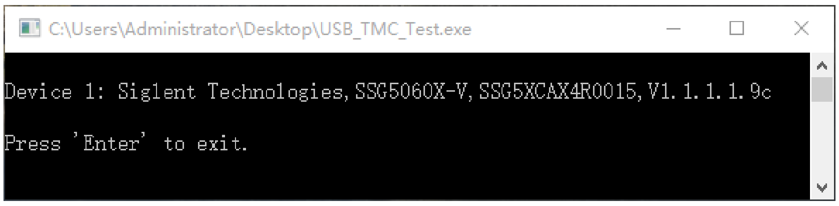
(1) USBTMC access code

Write a function Usbtmc_test:

```
int Usbtmc_test()
{
    /* This code demonstrates sending synchronous read & write commands */
    /* to an USB Test & Measurement Class (USBTMC) instrument using    */
    /* NI-VISA            */
    /* The example writes the "*IDN?\n" string to all the USBTMC    */
    /* devices connected to the system and attempts to read back    */
    /* results using the write and read functions.          */
    /* The general flow of the code is */
    /* Open Resource Manager     */
    /* Open VISA Session to an Instrument                              */
    /* Write the Identification Query Using viPrintf                   */
    /* Try to Read a Response With viScanf                             */
    /* Close the VISA Session    */
    /*************************************************************/
    ViSession defaultRM;
    ViSession instr;
    ViUInt32 numInstrs;
    ViFindList findList;
    ViStatus    status;
    char instrResourceString[VI_FIND_BUFLEN];
```

```
        unsigned char buffer[100];
        int i;
        /** First we must call viOpenDefaultRM to get the manager
        * handle. We will store this handle in defaultRM.*/
        status = viOpenDefaultRM (&defaultRM);
        if (status<VI_SUCCESS)
        {
            printf ("Could not open a session to the VISA Resource Manager!\n");
            return      status;
        }
    /* Find all the USB TMC VISA resources in our system and store the number of resources in the system in
    numInstrs.*/
        status = viFindRsrc (defaultRM, "USB?*INSTR", &findList, &numInstrs, instrResourceString);
        if (status<VI_SUCCESS)
        {
            printf ("An error occurred while finding resources.\nPress 'Enter' to continue.");
            fflush(stdin);
            getchar();
            viClose (defaultRM);
            return      status;
        }
        /** Now we will open VISA sessions to all USB TMC instruments.
        * We must use the handle from viOpenDefaultRM and we must
        * also use a string that indicates which instrument to open. This
        * is called the instrument descriptor. The format for this string
        * can be found in the function panel by right clicking on the
        * descriptor parameter. After opening a session to the
        * device, we will get a handle to the instrument which we
        * will use in later VISA functions. The AccessMode and Timeout
        * parameters in this function are reserved for future
        * functionality. These two parameters are given the value VI_NULL.*/
        for (i=0; i<int(numInstrs); i++)
        {
            if (i> 0)
            {
                viFindNext (findList, instrResourceString);
            }
            status = viOpen (defaultRM, instrResourceString, VI_NULL, VI_NULL, &instr);
            if (status<VI_SUCCESS)
            {
                printf ("Cannot open a session to the device %d.\n", i+1);
                continue ;
            }
            /* * At this point we now have a session open to the USB TMC instrument.
            * We will now use the viPrintf function to send the device the string "*IDN?\n",
            * asking for the device's identification. */
            char * cmmand ="*IDN?\n";
            status = viPrintf (instr, cmmand);
            if (status<VI_SUCCESS)
            {
                printf ("Error writing to the device %d.\n", i+1);
                status = viClose (instr);
                continue;
            }
            /** Now we will attempt to read back a response from the device to
            * the identification query that was sent. We will use the viScanf
            * function to acquire the data.
            * After the data has been read the response is displayed. */
            status = viScanf(instr, "%t", buffer);
            if (status<VI_SUCCESS)
            {
                printf ("Error reading a response from the device %d.\n", i+1);
            }
            else
            {
                printf ("\nDevice %d: %s\n", i+1, buffer);
            }
            status = viClose (instr);
        }
        /** Now we will close the session to the instrument using
        * viClose. This operation frees all system resources.          */
        status = viClose (defaultRM);
        printf("Press 'Enter' to exit.");
        fflush(stdin);
        getchar();
        return 0;
    }

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

The running result:



```
C:\Users\Administrator\Desktop\USB_TMC_Test.exe

Device 1: Siglent Technologies,SSG5060X-V,SSG5XCAX4R0015,V1.1.1.1.9c

Press 'Enter' to exit.
```

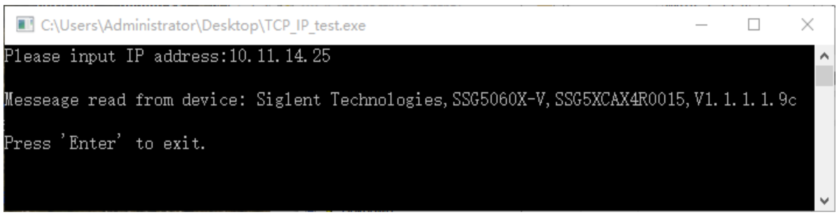
(2) TCP/IP access code

Write a function TCP_IP_Test:

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

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

    strcat(head,pIP);
    strcat(head,tail);
    status = viOpen (defaultRM, head, VI_LOAD_CONFIG, VI_NULL, &instr);
    if (status<VI_SUCCESS)
    {
        printf ("An error occurred opening the session\n");
        viClose(defaultRM);
    }
    status = viPrintf(instr, "*idn?\n");
    if (status<VI_SUCCESS)
    {
        printf("Error writing to the device.\n");
        viClose(defaultRM);
    }
    status = viScanf(instr, "%t", outputBuffer);
    if (status<VI_SUCCESS)
    {
        printf("viRead failed with error code: %x \n",status);
        viClose(defaultRM);
    }
    else
    {
        printf ("\nMesseage read from device: %*s\n", 0,outputBuffer);
    }
    status = viClose (instr);
    status = viClose (defaultRM);
    printf("Press 'Enter' to exit.");
```

```
        fflush(stdin);
        getchar();
        return 0;
}

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

The running result:



```
Please input IP address:10.11.14.25

Messeage read from device: Siglent Technologies,SSG5060X-V,SSG5XCAX4R0015,V1.1.1.1.9c

Press 'Enter' to exit.
```

4.1.2 VB Example

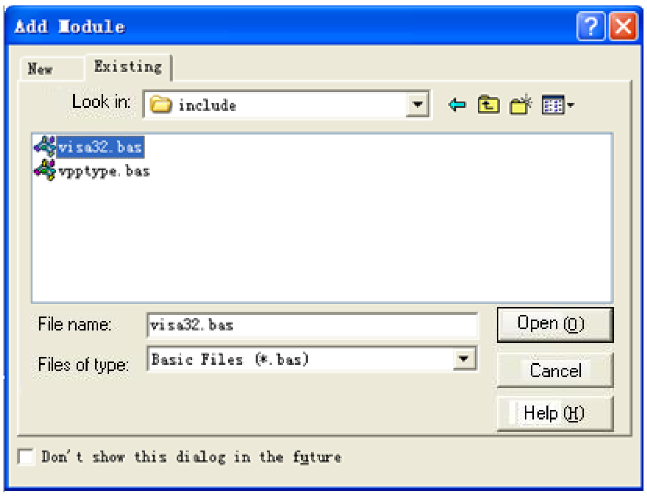
System environment: Windows 7

Programming software: Microsoft Visual Basic 6.0

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

Please follow below steps to complete the example:

1. Open Visual Basic and build a standard application program project (standard EXE).
2. Set up the project environment to use the NI-VISA library. Click the Existing tag of Project >> Add Existing Item, search for the visa32.bas file in the include folder under the NI-VISA installation path and add the file. This will enable VISA functions and VISA data types to be used in the program.



3. Add code

(1) USBTMC access code

Write a function Usbtmc_test:

```
Private Function Usbtmc_test() As Long
    ' This code demonstrates sending synchronous read & write commands
    ' to an USB Test & Measurement Class (USBTMC) instrument using
    ' NI-VISA
    ' The example writes the "*IDN?\n" string to all the USBTMC
    ' devices connected to the system and attempts to read back
    ' results using the write and read functions.
    ' The general flow of the code is
    ' Open Resource Manager
    ' Open VISA Session to an Instrument
    ' Write the Identification Query Using viWrite
    ' Try to Read a Response With viRead
    ' Close the VISA Session
```

```
Const MAX_CNT = 200

Dim defaultRM As Long
Dim instrsesn As Long
Dim numInstrs As Long
Dim findList As Long
Dim retCount As Long

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

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

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

```
            status = viOpen(defaultRM, instrResourceString, VI_NULL, VI_NULL, instrsesn)
            If (status < VI_SUCCESS) Then
                resultTxt.Text = "Cannot open a session to the device " + CStr(i + 1)
                GoTo NextFind
            End If

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

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

        Next i

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

    End Function
```

(2) TCP/IP access code

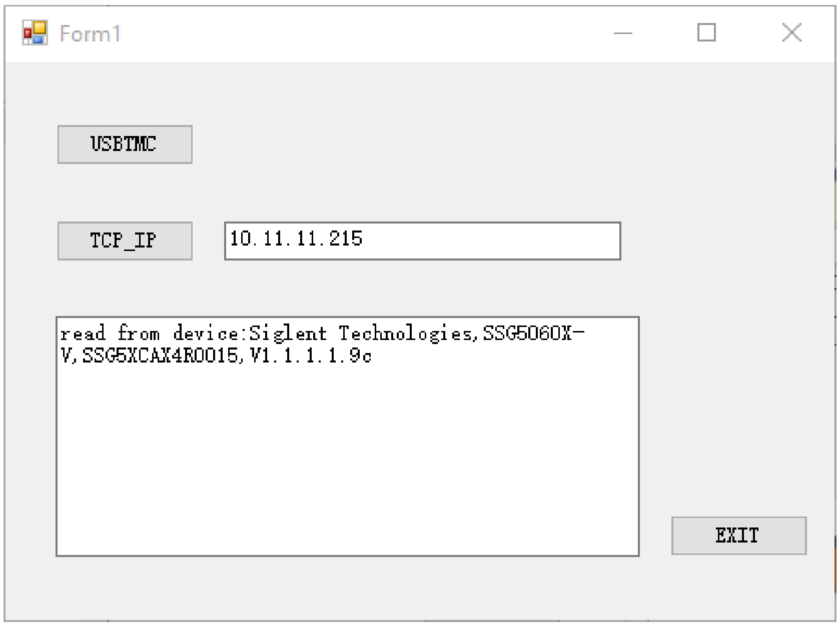
Write a function TCP_IP_Test:

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

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

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

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

        End Function
```

(3) Button control code:

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

The running result:



10.11.11.215

```
read from device:Siglent Technologies,SSG5060X-
V,SSG5XCAX4R0015,V1.1.1.9c
```

4.1.3 MATLAB Example

System environment: Windows 7

Programming software: MATLAB R2013a

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

Please follow below steps to complete the example:

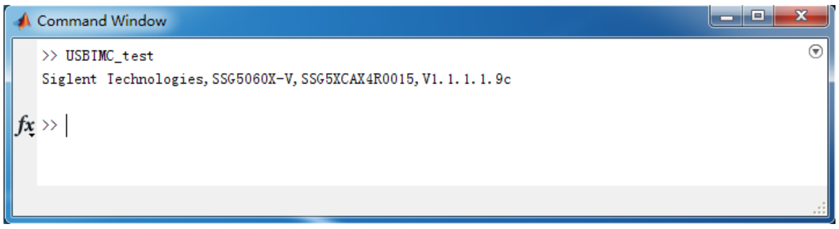
1. Open MATLAB and modify the current directory. In this example, change the current directory to D:\USBTMC_TCPIP_Demo.
2. Click File>>New>>Script in the MATLAB interface to create an empty M document.
3. Add code

   (1) USBTMC access code

   Write a function Usbtmc_test:

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

%Create a VISA-USB object connected to a USB instrument
vu = visa('ni','USB0::0xF4EC::0x1502::SSG5XCAX4R0015::INSTR');

%Open the VISA object created
fopen(vu);

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

%Request the data
outputbuffer = fscanf(vu);
disp(outputbuffer);

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

end
```

The running result:



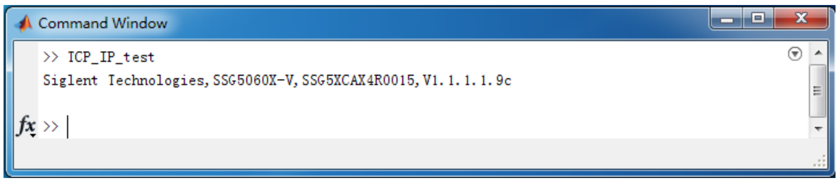
```
>> USBIMC_test
Siglent Technologies,SSG5060X-V,SSG5XCAX4R0015,V1.1.1.1.9c
>>
```

(2) TCP/IP access code

Write a function TCP_IP_Test:

```
function TCP_IP_test()
% This code demonstrates sending synchronous read & write commands
% to an TCP/IP instrument using NI-VISA

%Create a VISA-TCPIP object connected to an instrument
%configured with IP address.
vt = visa('ni',['TCPIP0::','10.11.11.215','::INSTR']);

%Open the VISA object created
fopen(vt);

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

%Request the data
outputbuffer = fscanf(vt);
disp(outputbuffer);

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

end
```

The running result:



```
>> TCP_IP_test
Siglent Technologies,SSG5060X-V,SSG5XCAX4R0015,V1.1.1.1.9c
>>
```

4.1.4 LabVIEW Example

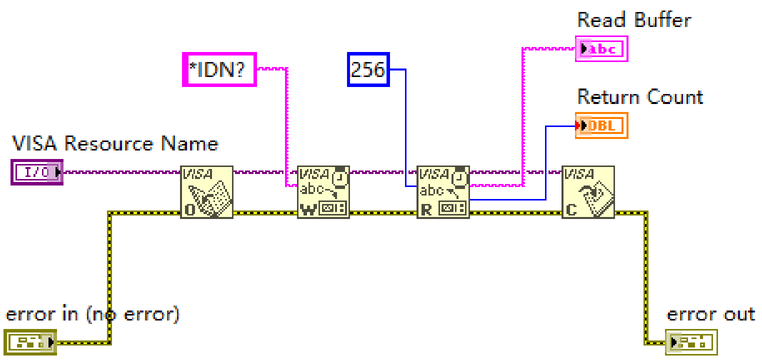
System environment: Windows 7

Programming software: LabVIEW 2011

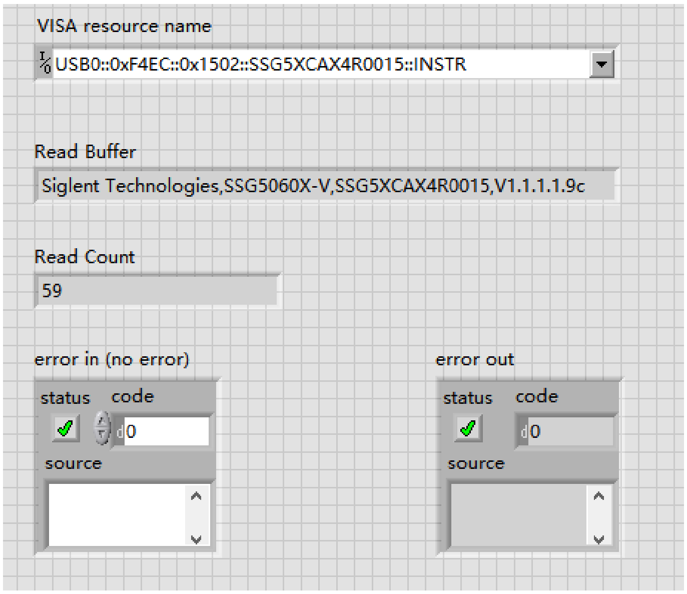
Example content: Use NI-VISA to access and control the device through USBTMC or TCP/IP and perform read and write operations.

Please follow below steps to complete the example:

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



5. Select the device resource from the VISA resource name list box and run the program.


VISA resource name

USB0::0xF4EC::0x1502::SSG5XCAX4R0015::INSTR

Read Buffer

Siglent Technologies,SSG5060X-V,SSG5XCAX4R0015,V1.1.1.1.9c

Read Count

59

error in (no error)

status ✓ code 0

source

error out

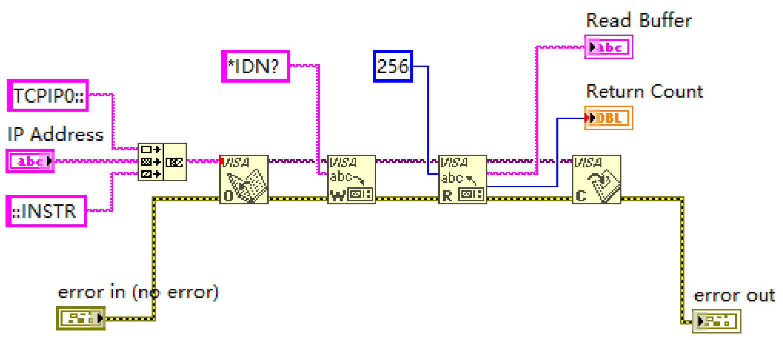
status ✓ code 0

source

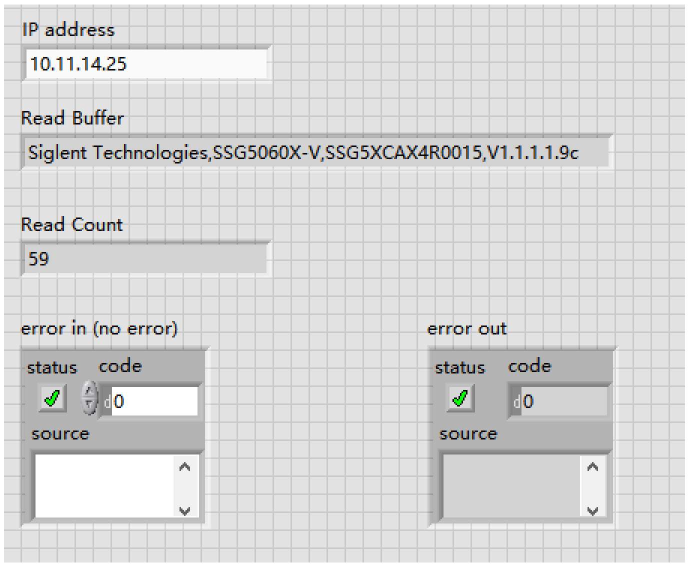
In this example, the VI opens a VISA session to the USBTMC device, writes a command to the device, and then reads back the response. In this example, the specific command sent is a device ID query. Please check the device command set with your device manufacturer. After all communication is complete,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

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

1. Connect them as shown in the figure below:



2. Enter the IP address and run the program:


IP address
10.11.14.25
Read Buffer
Siglent Technologies,SSG5060X-V,SSG5XCAX4R0015,V1.1.1.1.9c
Read Count
59
error in (no error)
status code
0
source
error out
status code
0
source

4.2 Socket Examples

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

4.2.1 Python Example

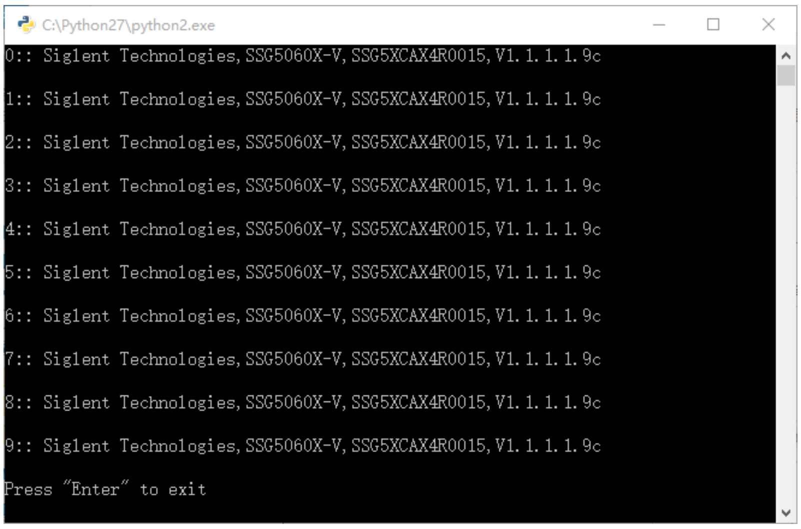
Python is an interpreted programming language that allows you to work quickly and is very portable. Python has an underlying network module that provides access to the Socket interface, port 5025. Python scripts can be written for the Socket interface to perform various test and measurement tasks.

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

Below is the code of the script:

```
#!/usr/bin/env python
#-*- coding:utf-8 –*-
#-----------------------------------------------------------------------------
# The short script is an example that open a socket, sends a query,
# print the return message and closes the socket.
#-----------------------------------------------------------------------------
import socket     # for sockets
import sys    # for exit
import time # for sleep
#-----------------------------------------------------------------------------
remote_ip = "10.11.22.27"   # should match the instrument’s IP address
port = 5025 # the port number of the instrument service

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

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

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

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

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

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

The running result:



```
0:: Siglent Technologies,SSG5060X-V,SSG5XCAX4R0015,V1.1.1.1.9c

1:: Siglent Technologies,SSG5060X-V,SSG5XCAX4R0015,V1.1.1.1.9c

2:: Siglent Technologies,SSG5060X-V,SSG5XCAX4R0015,V1.1.1.1.9c

3:: Siglent Technologies,SSG5060X-V,SSG5XCAX4R0015,V1.1.1.1.9c

4:: Siglent Technologies,SSG5060X-V,SSG5XCAX4R0015,V1.1.1.1.9c

5:: Siglent Technologies,SSG5060X-V,SSG5XCAX4R0015,V1.1.1.1.9c

6:: Siglent Technologies,SSG5060X-V,SSG5XCAX4R0015,V1.1.1.1.9c

7:: Siglent Technologies,SSG5060X-V,SSG5XCAX4R0015,V1.1.1.1.9c

8:: Siglent Technologies,SSG5060X-V,SSG5XCAX4R0015,V1.1.1.1.9c

9:: Siglent Technologies,SSG5060X-V,SSG5XCAX4R0015,V1.1.1.1.9c

Press "Enter" to exit
```

About SIGLENT

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

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

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

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

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

Follow us on
Facebook: SiglentTech