TELONIC
TEST INSTRUMENTS & POWER SUPPLIES
ESTABLISHED OVER 50 YEARS SINCE 1966
01189 786911 TELONIC.CO.UK

SDG8000A

Arbitrary Waveform Generator

Programming Guide

EN01A

SIGLENT TECHNOLOGIES CO., LTD.

Content

1 Programming Overview

By using USB and LAN interfaces, in combination with NI-VISA and programming languages, users can remotely control the waveform generator. Through the LAN interface, VXI-11, Sockets, and Telnet protocols can be used to communicate with the instruments. This chapter introduces how to build communication between the instrument and the PC. It also introduces how to configure a system for remote instrument control.

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.

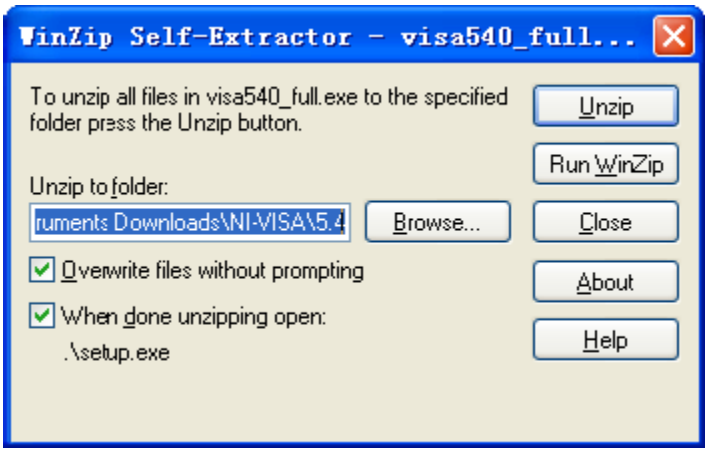
NI-VISA is a communication library that enables computer communications to instrumentation. There are two available VISA packages: A full version and the Run-Time Engine. The full version includes NI device drivers and a tool named NI MAX; a user interface to control the device. While the drivers and NI MAX can be useful, they are not required for remote control. The Run-Time Engine is a much smaller file and is recommanded for remote control.

For convenience, you can obtain the latest version of the NI-VISA run-time engine or the full version from the National Instruments website. The installation process is similar for both versions.

Follow these steps to install NI-VISA (The full version of NI-VISA 5.4 is used in this example):

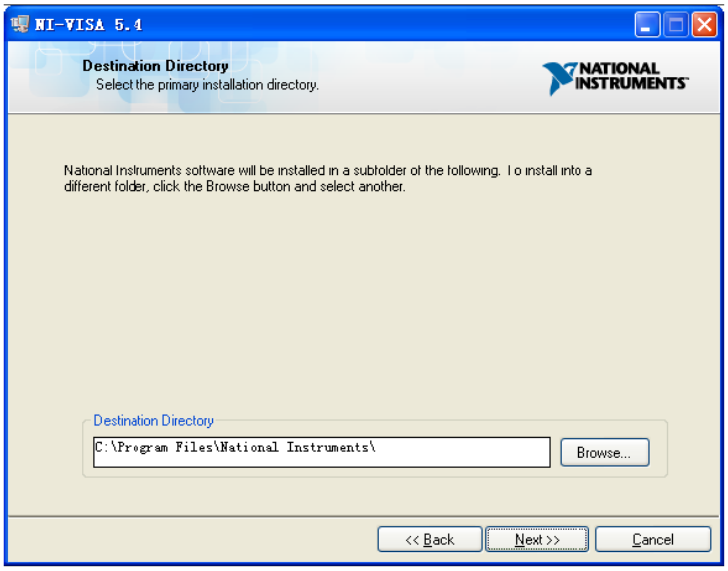
a. Download the appropriate version of NI-VISA (the Run-time engine is recommanded)

b. Double click the visa540_full.exe and observe the dialog box as shown below:

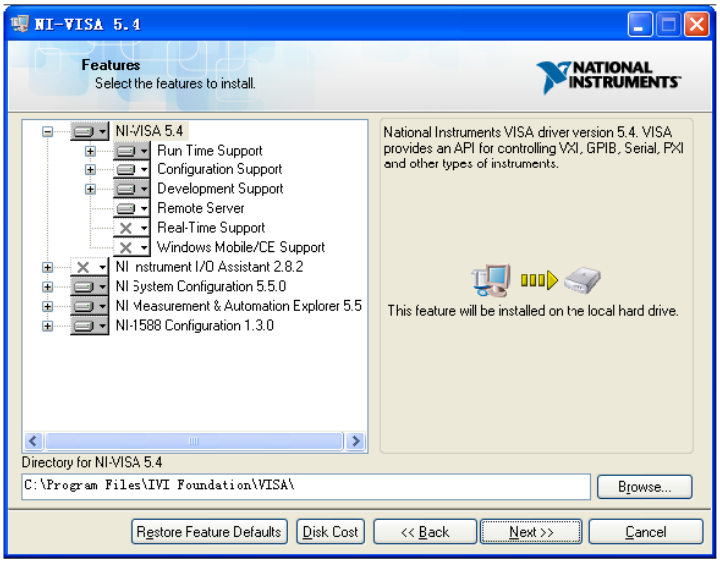


c. Click Unzip, the install process will launch after unzipping files. If your computer needs to install the .NET Framework 4, it may auto-start.

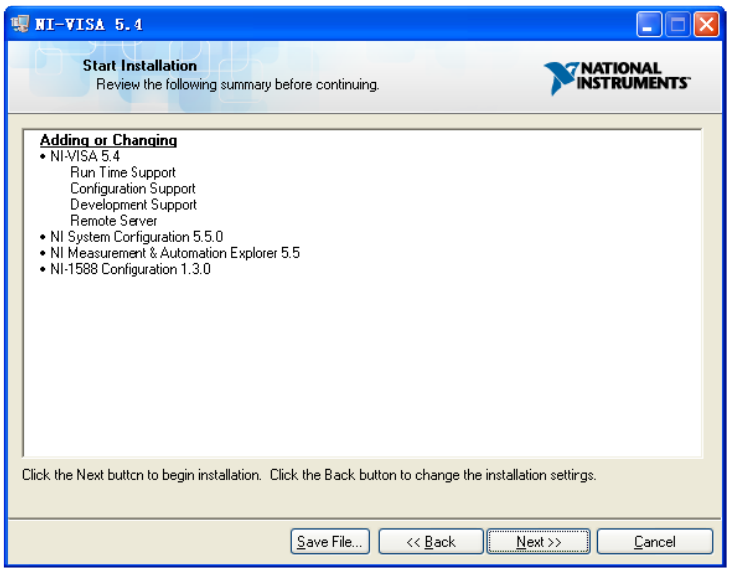
d. The NI-VISA install dialog is shown above. Click Next to start the installation process.



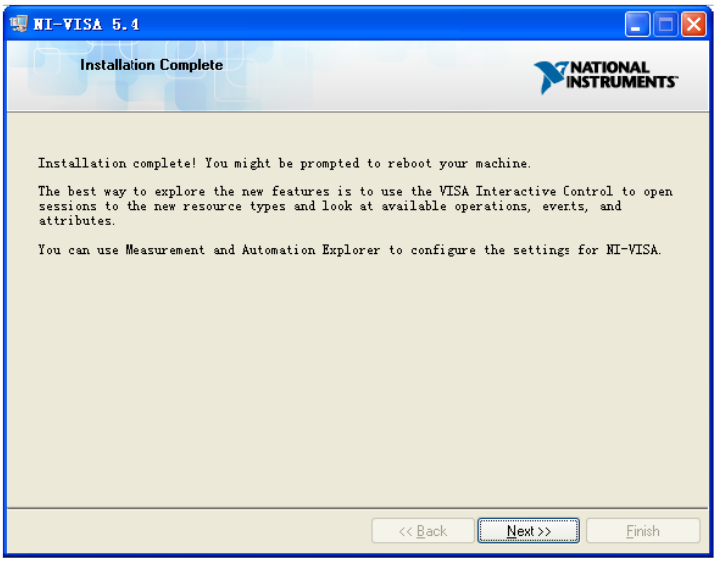
e. Set the install path, the default path is “C:\Program Files\National Instruments\”, you can change it if you prefer. Click Next, dialog as shown above.



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



g. Click Next to begin installation.

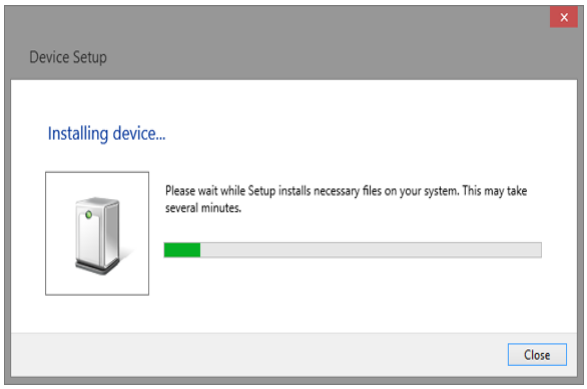


h. Now the installation is complete. Reboot your PC.

1.1.2 Connect the instrument

Depending on the specific model, the arbitrary waveform generator may be able to communicate with a PC through the USB or LAN interface.

Connect the arbitrary waveform generator and the USB Host interface of the PC using a USB cable. Assuming your PC is already turned on, turn on the SDG, and then the PC will display the “Device Setup” screen as it automatically installs the device driver as shown below.



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

1.2 Remote Control

1.2.1 User-defined Programming

Users can send SCPI commands via a computer to program and control the arbitrary waveform generator. For details, refer to the introductions in "Programming Examples **错误!未找到引用源。**".

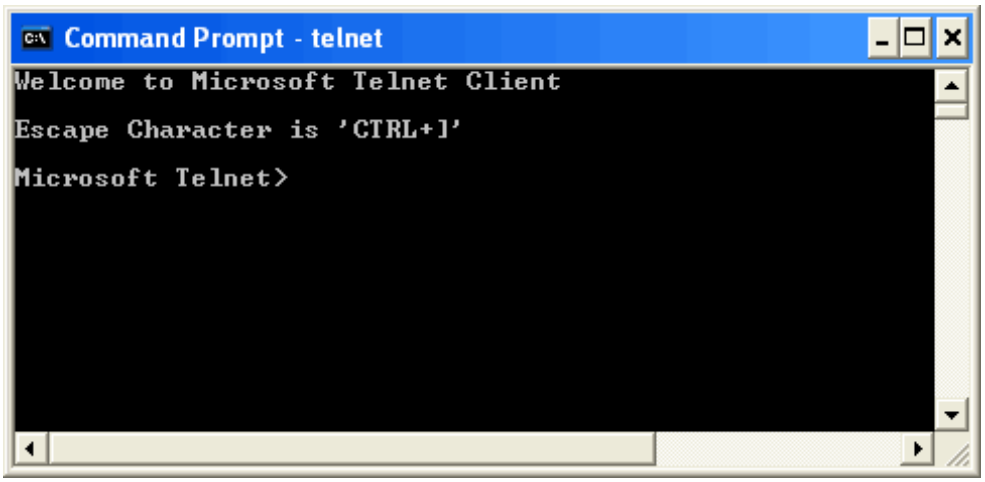
1.2.2 Using SCPI via NI-MAX

NI-MAX is a program created and maintained by National Instruments. It provides a basic remote control interface for VXI, LAN, USB, GPIB, and Serial communications. The SDG can be controlled remotely by sending SCPI commands via NI-MAX.

1.2.3 Using SCPI over Telnet

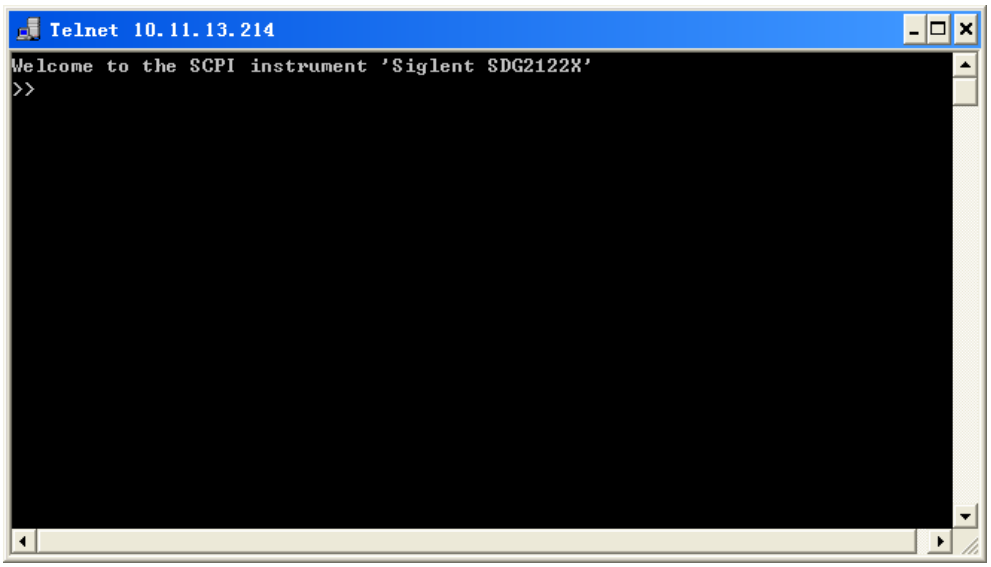
Telnet provides a means of communicating with the SDG over the LAN. The Telnet protocol sends SCPI commands to the SDG from a PC and is similar to communicating with the SDG over USB. It sends and receives information interactively: one command at a time. The Windows operating systems use a command prompt style interface for the Telnet client. The steps are as follows:

1. On your PC, click Start > All Programs > Accessories > Command Prompt.
2. At the command prompt, type in *telnet*.
3. Press the Enter key. The Telnet display screen will be displayed.

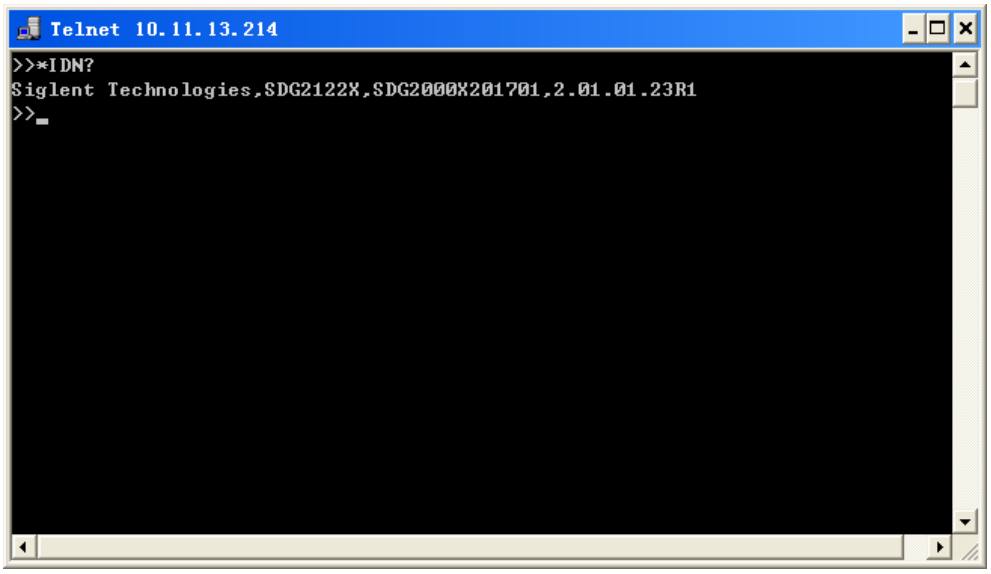


4. At the Telnet command line, type:open XXX.XXX.XXX.XXX 5024
   Where *XXX.XXX.XXX.XXX* is the instrument's IP address and 5024 is the port. You should see a

response similar to the following:



5. At the SCPI> prompt, input the SCPI commands such as **IDN?* to return the company name, model number, serial number, and firmware version number.



6. To exit the SCPI> session, press the Ctrl+] keys simultaneously.
7. Type *quit* at the prompt or close the Telnet window to close the connection to the instrument and exit Telnet.

1.2.4 Using SCPI over Socket

Socket API can be used to control the SDG series by LAN without installing any other libraries. This can reduce the complexity of programming.

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

Please see section 4.2 "Examples of Using Sockets" for the details.

2 Introduction to the SCPI Language

2.1 About Commands & Queries

This section lists and describes the remote control commands and queries recognized by the instrument. All commands and queries can be executed in either the local or remote state.

Each command or query, with syntax and other information, has some examples listed. The commands are given in both long and short format at "**COMMAND SYNTAX**" and "**QUERY SYNTAX**", and the subject is indicated as a command or query or both. Queries perform actions such as obtaining information from the instrument and are identified by a question mark (?) following the header.

2.2 Description

In the description, a brief explanation of the function performed is given. This is followed by a presentation of the formal syntax, with the header given in Upper-and-Lower-Case characters and the short form derived from it in ALL UPPER-CASE characters. Where applicable, the syntax of the query is given with the format of its response.

2.3 Usage

The commands and queries listed here can be used for SDG8000A Series Arbitrary Waveform Generators.

2.4 Command Notation

The following notations are used in the commands:

< > Angular brackets enclose words that are used as placeholders, of which there are two types: the header path and the data parameter of a command.

:= A colon followed by an equals sign separates a placeholder, from the description of the type and range of values that may be used in a command instead of the placeholder.

{ } Braces enclose a list of choices, one of which must be made.

[ ] Square brackets enclose optional items.

… Ellipsis (trailing dots) indicate that the preceding element may be repeated one or more times.

3 Commands and Queries

3.1 IEEE 488.2 Common Command Introduction

The IEEE standard defines the common commands used for querying the basic information of the instrument or executing basic operations. These commands usually start with "*" and the length of the keywords of the command is usually 3 characters.

3.1.1 *IDN

DESCRIPTION	The *IDN? query causes the instrument to identify itself. The response is comprised of the manufacturer, model, serial number, and firmware version.
QUERY SYNTAX	*IDN?
RESPONSE FORMAT	Format 1: *IDN, <device id>,<model>,<serial number>,<firmware version>, <hardware version> Format 2: <manufacturer>,<model>,<serial number>,<firmware version> <device id>:= “SDG”. <manufacturer>:= “Siglent Technologies”. <model>:= A model identifier less than 14 characters, should not contain the word “MODEL”. <serial number>:= The serial number. <firmware version>:= The firmware version number. <hardware version>:= The hardware level field, containing information about all separately revisable subsystems.
EXAMPLE	Reads version information: **IDN?* Return: *Siglent\sTechnologies,SDG8000A,* *SDG8000ATEST01,1.1.1.7R6_0526\n* (It may differ from each version)

3.1.2 *OPC

DESCRIPTION	The *OPC (Operation Complete) command sets the OPC bit (bit 0) in the standard Event Status Register (ESR). This command has no other effect on the operation of the device because the instrument starts parsing a command or query only after it has completely processed the previous command or query. The *OPC? query always responds with the ASCII character 1 because the device only responds to the query when the previous command has been entirely executed.
COMMAND SYNTAX	*OPC
QUERY SYNTAX	*OPC?
RESPONSE FORMAT	Format 1: *OPC 1 Format 2: 1

3.1.3 *RST

DESCRIPTION	The *RST command initiates a device reset and recalls the default setup.
COMMAND SYNTAX	*RST
EXAMPLE	This example resets the signal generator to the default setup: **RST*

3.2 Output Command

3.2.1 Noise Sum Command

DESCRIPTION	Enable noise sum,Add noise to the channel.
COMMAND SYNTAX	< channel >:NOISE_ADD < Parameters >,< Value > < channel >:= { C1, C2, C3, C4}。 < Parameters >:= {a parameter from the table below}. < Value >:={value of the corresponding parameter}.
QUERY SYNTAX	< channel >:NOISE_ADD?
RESPONSE FORMAT	<channel>:NOISE_ADDSTATE,ON|OFF,RATIO,<value> ,RATIO_DB,<value(dB)>
EXAMPLE	Set ch1 for noise sum and set the signal-to-noise ratio to 120: *C1:NOISE_ADD STATE,ON,RATIO,120*

Parameters	Value	Description
STATE	<state>	:={ON,OFF}.Noise sum switch.
RATIO_DB	<value>	:= signal-to-noise ratio（dB）, Please refer to the data manual for detailed values
RATIO	<value>	:= Signal to noise ratio, values detailed in the data manual

3.2.2 Channel polarity, switch, load setting commands

DESCRIPTION	Set the switch for channel output, channel load, and channel polarity.
COMMAND SYNTAX	< channel >:OUTPut ON|OFF,LOAD,<load>,PLRT,< polarity > < channel >:= {C1, C2, C3, C4}. <load>:= {50, HZ}. The default unit is ohm. <polarity>:={NOR, INVT}, NOR represents normal (off), and INVT represents reverse (on).
QUERY SYNTAX	<channel>:OUTPut?
RESPONSE FORMAT	< channel >:OUTP ON|OFF,LOAD,<load>,PLRT,< polarity >
EXAMPLE	Set ch1 output to open: *C1:OUTP ON*

	Query CH1 output status: *C1:OUTP?* Return: *C1:OUTP ON,LOAD,HZ,PLRT,NOR* Set the load of CH1 to 50 ohms: *C1:OUTP LOAD,50* Set CH1 polarity to normal (off): *C1:OUTP PLRT,NOR* Set the load of CH1 to a custom resistance 80 ohms: *C1:OUTP LOAD,80*

3.2.3 Amplitude limit command

DESCRIPTION	Set the maximum amplitude limit for the channel
COMMAND SYNTAX	<channel>:BaSic_WaVe MAX_OUTPUT_AMP,<value> <channel>:= {C1, C2, C3, C4}. <value>:= { The range of values can be found in the data manual }.
QUERY SYNTAX	<chanenel>:BaSic_WaVe?
EXAMPLE	Set the maximum output amplitude of ch1 to 1V: *C1:BSWV MAX_OUTPUT_AMP,1*

3.2.4 Digital Filter Command

DESCRIPTION	This command is used to set the switch and cutoff frequency of a digital filter.
COMMAND SYNTAX	<channel>:FILTer< parameter >,<value> <channel>:= {C1, C2, C3, C4} < parameter >:= { The parameters in the table below } <value>:= { The values of relevant parameters }

Parameter	Value	Description
STATE	<state>	:= {ON, OFF}. Digital filter status
COFF_FRQ	<cutoff_freq>	:= {1 uHz~2 GHz}.The unit is Hz, used to set the cutoff frequency of the digital filter

QUERY SYNTAX	<channel>:FILTer? <channel>:= {C1, C2, C3, C4}
RESPONSE FORMAT	<channel>:FILT <parameter>,<value>
EXAMPLE	Enable the digital filter for ch1 : *C1:FILT STATE,ON* Set the cutoff frequency of the digital filter for ch1 to 200 MHz : *C1:FILT COFF_FRQ,200000000* Query the digital filter information of ch1 : *C1:FILT?* Return : *STATE,OFF,COFF_FRQ,200000000HZ*

3.2.5 Analog filter command

DESCRIPTION	This command is used to set the switching of analog filters.
COMMAND SYNTAX	<channel>:FILT5G <parameter> <channel>:= {C1, C2, C3, C4} <parameter>:= {ON, OFF}. ON: Use 5G analog filters; OFF: Use a 2G analog filter.
QUERY SYNTAX	<channel>:FILT5G? <channel>:= {C1, C2, C3, C4}。
RESPONSE FORMAT	<parameter>
EXAMPLE	Enable the 5G analog filter for ch1 : *C1:FILT5G ON* Query the status of the 5G analog filter for ch1 : *C1:FILT5G?* Return : *ON\n*

3.2.6 Overvoltage protection command

DESCRIPTION	This command is used to set or query the overvoltage protection status.
COMMAND SYNTAX	<channel>:VOLTPRT<state> <channel>:= {C1, C2, C3, C4}。

	<state>:= {ON, OFF}
QUERY SYNTAX	<channel>:VOLTPRT? <channel>:= { C1, C2, C3, C4}
EXAMPLE	Turn on the overvoltage protection switch of ch1 : *C1:VOLTPRT ON* Query the overvoltage status of ch1 : *C1:VOLTPRT?* Return : *ON\n*

3.2.7 OutPut Skew Command

DESCRIPTION	This command is used to set or query the output skew.
COMMAND SYNTAX	<channel>:OUTPut:SKEW <value> <channel>:= {C1, C2, C3, C4} <value>:= {-500 ~ 500}，unit is “ns”
QUERY SYNTAX	<channel>:OUTPut:SKEW? <channel>:= {C1, C2, C3, C4}
RESPONSE FORMAT	<value> <value>:= { The current channel setting value }
EXAMPLE	Set the output delay of C1 channel to 0.2 ns : *C1:OUTPut:SKEW 0.2e-9* Query C1 channel output delay : *C1:OUTPut:SKEW?* Return : *2e-10*

3.2.8 Output Mode Command

DESCRIPTION	Set or query output mode.
COMMAND SYNTAX	<channel>:BaSic_WaVe PATHMODE,<value> <channel>:= {C1, C2, C3, C4} <value>:= {DC_DIRECT, DC_AMPLIFY, AC_DIRECT, AC_AMPLIFY}
QUERY SYNTAX	<channel>:BaSic_WaVe?
RESPONSE FORMAT	<channel>:BSWV<parameter> <parameter>:= { All parameters of the current waveform }
EXAMPLE	Set C1 output mode to AC_DIRECT :

	C1:BSWV PATHMODE,AC_DIRECT Query the parameters of C1: *C1:BSWV?* Return: *C1:BSWV\sWVTP,SINE,FRQ,1000HZ,PERI,0.001S,AMP,0.7112V,AMPVRMS,0.251447Vrms,MAX_OUTPUT_AMP,1.5V,PATHMODE,AC_DIRECT,OFST,0V,DLY,-0S,DIFF_OFST,0V,HLEV,0.3556V,LLEV,-0.3556V,PHSE,0\n*

3.2.9 Marker Setting Command

DESCRIPTION	This command is used to set or query Marker parameters and status.
COMMAND SYNTAX	<channel>:MARKer<parameter1> <parameter>,<value> <channel>:= {C1, C2, C3, C4} <parameter1>:= {1, 2},1 is MarKer1, 2 is MarKer2. <value>:= { The values of relevant parameters }

Parameter	Value	Description
STATE	<state>	:= {ON, OFF} Turn on or off Marker.
VOLTE	<value>	:= {0.2, 0.4, 0.6, 0.8, 1, 1.2, 1.4, 1.6, 1.8, 2}. Unit is "Vpp".
OFFSET	<value>	:= {Floating point numbers from 0 to 2 }. Unit is "Vpp".
SKEW	<value>	:= {-500ns~500ns}.Unit is "s".

QUERY SYNTAX	<channel>: MARKer<parameter1>? <channel>:= { C1, C2, C3, C4} <parameter1>:= {1, 2}.1 is MarKer1, 2 is MarKer2.
EXAMPLE	Set the Marker1 switch status of ch1 to open: *C1:MARKer1 STATE,ON* Set the amplitude of Marker1 for ch1 to 0.6 V: *C1:MARKer1 VOLTE,0.6* Query the parameters of Marker1 for ch1: *C1:MARKer1?* Return: *CH1:MARKERA,STATE,ON,MARKERA,HLEVEL,0.300000,LLEVEL,-0.300000,SKEW,0S\n*

3.3 Multi channel setting

3.3.1 Channel Mode Command

DESCRIPTION	Set or query the relationship between channels, where channels are grouped in pairs or four channels are grouped together.
COMMAND SYNTAX	GMOD <parameter> <parameter>:= {PAIR, ALL}
QUERY SYNTAX	GMOD?
RESPONSE FORMAT	MODE<parameter>
EXAMPLE	Set the pair mode: *GMOD PAIR* Set the all mode: *GMOD ALL*

Note: Only SDG8004A are available.

3.3.2 Phase Mode Command

DESCRIPTION	Set or query phase mode.
COMMAND SYNTAX	<group>:MODE <parameter> <group>:= {G1, G2, G3} <parameter>:= {PHASELOCKED, INDEPENDENT}
QUERY SYNTAX	<group>:MODE?
RESPONSE FORMAT	MODE<parameter>
EXAMPLE	Set ch1-ch2 phase mode to phase independent mode: *G1:MODE INDEPENDENT*

Note:

G1: When selecting PAIR channel mode, set CH1/CH2 channels

G2: When selecting PAIR channel mode, set the CH3/CH4 channels

G3:When selecting ALL channel mode, set all channels.

3.3.3 Channels Combine Command

DESCRIPTION	Set or query channel combine status。
COMMAND SYNTAX	<channel>:CoMBiNe <state> <channel>:= {C1, C2, C3, C4} <state>:= {ON, OFF}
QUERY SYNTAX	<channel>:CoMBiNe? <channel>:= {C1, C2, C3, C4}
RESPONSE FORMAT	<channel>:CMBN <satte>
EXAMPLE	Set ch1 channel combine to on: *C1:CoMBiNe ON* Query the channel combine status of ch2: *C2:CMBN?* Return: *C2:CMBN OFF*

3.3.4 Channel Copy Command

DESCRIPTION	This command copies the parameters of one channel to another channel.
COMMAND SYNTAX	ParaCoPy < Target channel >,< Source channel > < Target channel >:= {C1, C2, C3, C4} < Source channel >:= {C1, C2, C3, C4}
EXAMPLE	Copy the parameters of ch1 to ch2: *PACP C2,C1*

3.3.5 In-phase Command

DESCRIPTION	Used to set phase synchronization for channels
COMMAND SYNTAX	<group>:EQPHASE <group>:= {G1, G2, G3}
EXAMPLE	Set ch1 and ch2 phase synchronization: *G1:EQPHASE*

Note:

G1: When selecting PAIR channel mode, set ch1/ch2 channels

G2: When selecting PAIR channel mode, set the ch3/ch4 channels

G3: When selecting ALL channel mode, set all channels

3.3.6 Tracking And Coupling Command

DESCRIPTION	Set or query channel tracking and coupling parameters. The coupling value can only be set when the tracking function is turned off.
COMMAND SYNTAX	<group>:COUPling <parameter>,<value> <group>:= {G1, G2, G3} <parameter>:= { The parameters in the table below } <value>:= { The values of relevant parameters }

Parameter	Value	Description
TRACE	<track_enble>	:= {ON, OFF}. Channel tracking state
TPHASE	<track_pha_dev>	:= Track phase difference. The unit is '°'
STATE	<state>	:= {ON, OFF}. Channel coupling state.
BSCH	<bsch>	:= {CH1, CH2, CH3, CH4}.
FCOUP	<fcoup>	:= {ON, OFF}. Frequency coupling state.
FDEV	<frq_dev>	:=The frequency difference between two channels. The unit is 'Hz'.
FRAT	<frat>	:=The frequency ratio between two channels.
PCOUP	<pcoup>	:= {ON, OFF}. Phase coupling state.
PDEV	<pha_dev>	:= The phase difference between two channels. The unit is '°'.
PRAT	<prat>	:= Phase ratio between two channels.
ACOUP	<acoup>	:={ON, OFF}.State of amplitude coupling.
ARAT	<arat>	:=The amplitude ratio between

			two channels.
	ADEV	<adev>	:= The amplitude deviation between two channels. Unit is 'Vpp'.
QUERY SYNTAX	<group>:COUPling?		
RESPONSE FORMAT	COUP <parameter> <parameter>:= { All coupling parameter values }		
EXAMPLE	Set CH1-CH2 group coupling status to open: *G1:COUP STATE,ON* Set the frequency deviation to 5Hz: *G1:COUP FDEV,5* Set the amplitude ratio to 2: *G1:COUP ARAT,2* Query CH1-CH2 coupling information: *G1:COUP?* Return: *COUPTRACE,OFF,FCOUP,ON,PCOUP,ON,ACOUP,ON,FDEV,5HZ,PRAT,1,ARAT,2*		

Note:

G1: When selecting PAIR channel mode, set ch1/ch2 channels

G2: When selecting PAIR channel mode, set the ch3/ch4 channels

G3: When selecting ALL channel mode, set all channels

3.4 Channel Output Mode Command

DESCRIPTION	Used to set or query the current output mode of a channel.
COMMAND SYNTAX	<channel>:CPARam MODE,<parameter> <channel>:= {C1, C2, C3, C4} <parameter>:= {AFG, AWG, IQ}
QUERY SYNTAX	<channel>:CPARam? <channel>:= {C1, C2, C3, C4}
RESPONSE FORMAT	MODE,<parameter> <parameter>:= {AFG, AWG, IQ}
EXAMPLE	Set the output mode of channel 1 to AFG: *C1:CPARam MODE,AFG* Query the output mode of the ch1 : *C1:CPARam?* Return: *MODE,AFG*

3.5 AFG Command

3.5.1 Basic Waveform Command

DESCRIPTION	Set or query basic wave parameters.
COMMAND SYNTAX	<channel>:BaSic_WaVe <parameter>,<value> <channel>:= {C1, C2, C3, C4} <parameter>:= { The following table shows the parameters } <value>:= { The values of relevant parameters }

Parameter	Value	Description
WVTP	<type>	:= {SINE, SQUARE, RAMP, PULSE, NOISE, ARB, DC, PRBS}.
FRQ	<frequency>	: =Frequency. The unit is "Hz", and the effective range of parameter values can be found in the data manual. When WVTP is noise and DC, this value is invalid.
PERI	<period>	: = period. The unit is 's'. The effective range of parameter values can be found in the data manual. When WVTP is noise and DC, this value is invalid.
AMP	<amplitude>	:= Amplitude. The unit is "Vpp". The effective range of parameter values can be found in the data manual. When WVTP is noise and DC, this parameter is invalid.
OFST	<offset>	:= offset. The unit is "V". The effective range of parameter values can be found in the data manual. When WVTP is noise, this value is invalid.
DIFF_OFST	<common	Please refer to the data manual for details. The unit is

	offset>	"V". Only supports setting when the output mode is DC_AMPLIFY.
SYM	<symmetry>	:= {0 to 100}. The unit is '% '. This parameter can only be set when WVTP is RAMP.
DUTY	<duty>	:= {0 to 100}. Duty cycle. The unit is'% '. The value of this parameter depends on the frequency. This parameter can only be set when WVTP is SQUARE or PULSE.
PHSE	<phase>	:= {0 to 360}. The unit is "°", which is invalid when WVTP is NOISE, PULSE or DC.
STDEV	<stdev>	:= The standard deviation of noise. The unit is "V". The effective range of parameter values can be found in the data manual. It can only be set when WVTP is NOISE.
MEAN	<mean>	: =The mean of noise. The unit is "V". The effective range of parameter values can be found in the data manual. This parameter can only be set when WVTP is NOISE.
WIDTH	<width>	:= Positive pulse width. The unit is 's'. The effective range of parameter values can be found in the data manual. This parameter can only be set when WVTP is PULSE.
RISE	<rise>	:= Rise time (10%~90%). The unit is 's'. The effective range of parameter values can be found in the data manual. This parameter can only be set when WVTP is a PULSE.

FALL	<fall>	:= Fall time (10%~90%). The unit is 's'. The effective range of parameter values can be found in the data manual. This parameter can only be set when WVTP is a PULSE.
DLY	<delay>	:= Delay. The effective range of parameter values can be found in the data manual.
HLEV	<high level>	:= High level. The unit is "V". When WVTP is NOISE, this value is invalid.
LLEV	<low level>	:= Low level. The unit is "V". When WVTP is NOISE, this value is invalid.
BANDSTATE	<bandwidth switch>	:= {ON, OFF}. This parameter can only be set when WVTP is NOISE.
BANDWIDTH	<bandwidth value>	:= Noise bandwidth. The unit is “Hz”. The effective range of parameter values can be found in the data manual. This parameter can only be set when WVTP is NOISE.
LENGTH	<prbs length>	:= {3~32}. PRBS actual length=2 ^ length-1. This parameter can only be set when WVTP is PRBS.
EDGE	<prbs rise/fall>	:= The rise/fall time of PRBS. The unit is 's'. The effective range of parameter values can be found in the data manual. This parameter can only be set when WVTP is PRBS.
PATHMODE	<output pathmode>	:= {DC_DIRECT, DC_AMPLIFY, AC_DIRECT, AC_AMPLIFY}
BITRATE	<prbs bit rate>	:= PRBS bit rate. The unit is bits per second (bps). The effective range of parameter values can

			be found in the data manual. This parameter can only be set when WVTP is PRBS.
	AMPVRMS	<amplitude>	Set the amplitude unit to Vrm.
	AMPDBM	<amplitude>	Set the amplitude unit to dBm.
	MAX_OUTPUT_AMP	<amplitude>	:= The maximum amplitude limit of the channel can be found in the data manual for the effective range of parameter values.
QUERY SYNTAX	<channel>:BaSic_WaVe? <channel>:= {C1, C2, C3, C4}		
RESPONSE FORMAT	<channel>:BSWV<parameter> <parameter>:= { All parameters of the current waveform }		
EXAMPLE	Set ch1 waveform is RAMP: *C1:BSWV WVTP,RAMP* Set ch1 frequency to 2000 Hz: *C1:BSWV FRQ,2000* Set the amplitude of ch1 to 1 Vpp: *C1:BSWV AMP,1* Query the parameters of ch1: *C1:BSWV?* Return: *C1:BSWV WVTP,RAMP,FRQ,2000HZ,PERI,0.01S,AMP,1V,OFST,0V,HLEV,0.5V,LLEV,-0.5V,PHSE,0* Set the noise bandwidth of ch1 to 100MHz: *C1:BSWV BANDWIDTH,100E6* or *C1:BSWV BANDWIDTH,100000000*		

3.5.2 PRBS Polynomial Command

DESCRIPTION	This command is used to set or query PRBS custom polynomial parameters. This command is only valid when the basic waveform is PRBS.
COMMAND SYNTAX	<channel>:PRBS:<parameter> <value> <channel>:= {C1, C2, C3, C4} <value>:= { The values of relevant parameters }
QUERY SYNTAX	<channel>:PRBS:<parameter>? <channel>:= { C1, C2, C3, C4}
EXAMPLE	Set the PRBS custom polynomial function of ch1 to ON: *C1:PRBS:FSTAte ON* Set the PRBS polynomial of ch1 to X7+X5+1 : *C1:PRBS:FORMula "X7+X5"* Query the PRBS custom polynomial of ch1 : *C1:PRBS:FORMula?* Return : *X7+X5+1*

Parameter	Value	Description
FSTAte	<STATE>	:= {ON, OFF}. Turn on or off the PRBS custom polynomial switch.
FORMula	<value>	:= Set a PRBS polynomial, double quotation marks are required.
SEED	<value>	:= {1, 2, ..., M}. Set the number of PRBS seeds.

3.5.3 AFG's Arb Waveform Setting Command

DESCRIPTION	Set or query the Arb waveform of the channel.
COMMAND SYNTAX	Format 1 : <channel>:ArbWaVe INDEX,<index> Format 2 : <channel>:ArbWaVe NAME,<name> Format 3 : <channel>:ArbWaVe NAME,<path> <channel>:= { C1, C2, C3, C4} <index>:= The index number of arb wave in the table below. <name>:= The name of arb wave in the table below. <path>:= Waveform path.

QUERY SYNTAX	<channel>:ARbWaVe? <channel>:= {C1, C2, C3, C4}
RESPONSE FORMAT	<channel>:ARWV INDEX,<index>,NAME,<name>
EXAMPLE	Set the arb of ch1 to the waveform with index number 2 : *C1:ARWV INDEX,2* Query the waveform of ch1 : *C1:ARWV?* Return : *C1:ARWV INDEX,2,NAME,StairUp* Set the waveform of ch1 to ' Cardiac ' : *C1:ARWV NAME,"Cardiac"* Set the waveform of CH1 through the waveform path : *C1:ARWV NAME,"Local/wave1.bin"* *C1:ARWV NAME,"Local/wave2.mat"* *C1:ARWV NAME,"Local/wave3.csv"* *C1:ARWV NAME,"net_storage/wave4.bin"* *C1:ARWV NAME,"U-disk0/wave1.bin"*

index	name	index	name	index	name	index	name
0	Sine	51	AttALT	102	LFPulse	153	Duty18
1	Noise	52	RoundHalf	103	Tens1	154	Duty20
2	StairUp	53	RoundsPM	104	Tens2	155	Duty22
3	StairDn	54	BlaseiWave	105	Tens3	156	Duty24
4	Stairud	55	DampedOsc	106	Airy	157	Duty26
5	Ppulse	56	SwingOsc	107	Besselj	158	Duty28
6	Npulse	57	Discharge	108	Bessely	159	Duty30
7	Trapezia	58	Pahcur	109	Dirichlet	160	Duty32
8	Upramp	59	Combin	110	Erf	161	Duty34
9	Dnramp	60	SCR	111	Erfc	162	Duty36
10	ExpFal	61	Butterworth	112	ErfcInv	163	Duty38
11	ExpRise	62	Chebyshev1	113	ErfInv	164	Duty40
12	Logfall	63	Chebyshev2	114	Laguerre	165	Duty42
13	Logrise	64	TV	115	Legend	166	Duty44
14	Sqrt	65	Voice	116	Versiera	167	Duty46
15	Root3	66	Surge	117	Weibull	168	Duty48

16	X^2	67	NA	118	LogNormal	169	Duty50
17	X^3	68	Ripple	119	Laplace	170	Duty52
18	Sinc	69	Gamma	120	Maxwell	171	Duty54
19	Gaussian	70	StepResp	121	Rayleigh	172	Duty56
20	Dlorentz	71	BandLimited	122	Cauchy	173	Duty58
21	Haversine	72	CPulse	123	CosH	174	Duty60
22	Lorentz	73	CWPulse	124	CosInt	175	Duty62
23	Gauspuls	74	GateVibr	125	CotH	176	Duty64
24	Gmonopuls	75	LFMPulse	126	CscH	177	Duty66
25	Tripuls	76	MCNoise	127	SecH	178	Duty68
26	Cardiac	77	AM	128	SinH	179	Duty70
27	Quake	78	FM	129	SinInt	180	Duty72
28	Chirp	79	PFM	130	TanH	181	Duty74
29	Twotone	80	PM	131	ACosH	182	Duty76
30	SNR	81	PWM	132	ASecH	183	Duty78
31	Hamming	82	EOG	133	ASinH	184	Duty80
32	Hanning	83	EEG	134	ATanH	185	Duty82
33	Kaiser	84	EMG	135	ACsch	186	Duty84
34	Blackman	85	Pulseilogram	136	ACoth	187	Duty86
35	Gausswin	86	ResSpeed	137	Bartlett	188	Duty88
36	Triang	87	ECG1	138	BohmanWin	189	Duty90
37	BlackmanH	88	ECG2	139	ChebWin	190	Duty92
38	Bartlett-Hann	89	ECG3	140	FlattopWin	191	Duty94
39	Tan	90	ECG4	141	ParzenWin	192	Duty96
40	Cot	91	ECG5	142	TaylorWin	193	Duty98
41	Sec	92	ECG6	143	TukeyWin	194	Duty99
42	Csc	93	ECG7	144	Duty01	195	demo1_375
43	Asin	94	ECG8	145	Duty02	196	demo1_16k
44	Acos	95	ECG9	146	Duty04	197	demo2_3k
45	Atan	96	ECG10	147	Duty06	198	demo2_16k
46	Acot	97	ECG11	148	Duty08		
47	Square	98	ECG12	149	Duty10		
48	SineTra	99	ECG13	150	Duty12		
49	SineVer	100	ECG14	151	Duty14		
50	AmpALT	101	ECG15	152	Duty16		

3.5.4 Arbitrary Wave Small Point Waveform Command

DESCRIPTION	Set arb small point waveform.
COMMAND SYNTAX	<channel>:WVDT <parameter>,<value> <channel>:= {C1, C2, C3, C4} <parameter>:= { The parameters in the table below } <value>:= { The values of relevant parameters }

Parameter	Value	Description
WVNM	<name>	:= Waveform name.
FRQ	<frequency >	:= Frequency. The unit is Hertz (Hz).
AMP	<amplitude >	:= Amplitude. The unit is volts(Vpp).
OFST	< offset >	:= Offset. The unit is the volt "V".
PHASE	< phase >	:=Phase. The unit is '°'.
WAVEDATA	< data >	:= Waveform data. The data written into the waveform. The data is binary, with 16 bits representing a waveform point. The code word range for each waveform point is -32767~32767.

QUERY SYNTAX	Format 1: WVDT? Mn Format 2: WVDT? USER,< name> Format 3: WVDT? USER,<path>,< name > <path>:= { User defined waveform storage path.}. < name >:= { User defined waveform name }.
EXAMPLE	Write data 0x6000c0006000 into the 'wave1' waveform and send it to ch1 : *C1:WVDT WVNM,wave1,FRQ,2500,AMP,2.5,OFST,0.2,* *WAVEDATA, b'0x6000c0006000'*

Note: This instruction is recommended to be programmed using the method described in section 4.1.5

3.5.5 Arbitrary Wave Large Point Waveform Command

DESCRIPTION	Set arb large point waveform.Can send the waveform of big data in segments to the specified bin file in the specified path.

COMMAND SYNTAX

<channel>:WVDT:SEGMENT <parameter>,<data>
<channel>:= {C1, C2, C3, C4}
<parameter>:= { The parameters in the table below }.
{data}:= Waveform data.

Parameter	Value	Description
WVNM	<name>	:= Waveform name
FRQ	<ffrequency>	:= Frequency. The unit is Hertz (Hz). It only takes effect when writing the first waveform data.
AMP	<amplitude>	:= Amplitude. The unit is volts(Vpp). It only takes effect when writing the first waveform data.
OFST	<offset>	:= Offset. The unit is the volt(V). It only takes effect when writing the first waveform data.
PHASE	<phase>	:= Phase. The unit is '°'. It only takes effect when writing the first waveform data.
BEGIN,WAVEDATA	<data>	:= The first waveform data. The data is binary, with 16 bits representing a waveform point. The code word range for each waveform point is -32767 ~32767.
WAVEDATA	<data>	:= Intermediate waveform data. The data is binary, with 16 bits representing a waveform point. The code word range for each waveform point is -32767 ~32767. At least one segment of data needs to be written in the middle.
END,WAVEDATA	<data>	:= The last segment of waveform data. The data

			is binary, with 16 bits representing a waveform point. The code word range for each waveform point is -32767~32767.
EXAMPLE	Segmenting the waveform and writing it to the device: *C1:WVDT:SEGMENT WVNM,"wave1",FRQ,2000,AMP,2,OFST,0.2,BEGIN,WAVEDATA,b'0x6000c0006'* *C1:WVDT:SEGMENT WVNM,"wave1",WAVEDATA,b'0x0006000'* *C1:WVDT:SEGMENT WVNM,"wave1",END,WAVEDATA,b'0x6000'*		

Note: This command is suitable for situations where the waveform data length is large. This instruction is recommended to be programmed using the method described in section 4.1.6.

3.5.6 Nonlinear Compensation Command

DESCRIPTION	This command is used to set or query the status of the non-linear compensation switch. This command is only valid when the basic waveform is SINE.
COMMAND SYNTAX	<channel>:NONLINECOMP STATE,<value> <channel>:= {C1, C2, C3, C4} <value>:= {ON, OFF}
QUERY SYNTAX	<channel>:NONLINECOMP? <channel>:= { C1, C2, C3, C4}
EXAMPLE	Set the non-linear compensation switch of ch1 to ON: *C1:NONLINECOMP STATE,ON* Query the non-linear compensation switch status of ch1: *C1:NONLINECOMP?* Return: *C1:NonLineComp STATE,ON\n*

3.5.7 Harmonic Command

DESCRIPTION	Set or query harmonic parameters. This command is only valid when the basic waveform is SINE.
COMMAND SYNTAX	<channel>:HARMonic <parameter>,<value> <channel>:= {C1, C2, C3, C4}

<value>:= { The values of the relevant parameters in the table below }

Parameter	Value	Description
HARMSTATE	<state>	:= {ON, OFF} Turn on or off harmonics.
HARMTYPE	<type>	:= { EVEN, ODD, ALL} Harmonic type.
HARMORDER	<num>	:= {1, 2,... , M},M is the maximum number of supported levels.
HARMPHASE	<phase>	:= {0~360}.The unit is °.
HARMAMP	<value>	:= Specify the amplitude of the harmonic. The unit is volt "Vpp".
HARMDBC	<value>	:= Specify the amplitude of the harmonic. The unit is "dBc".

QUERY SYNTAX

<channel>:HARMonic?
<channel>:={ C1, C2, C3, C4}

EXAMPLE

Set the harmonic switch of ch1 to ON:
C1:HARM HARMSTATE,ON
Set the harmonic order of ch1 to 2 and the amplitude to -6dBc:
C1:HARM HARMORDER,2,HARMDBC,-6
Query the harmonic parameters of ch1:
C1:HARM?
Return:
C1:HARM ,HARMSTATE,ON,HARMTYPE,EVEN,HARMORDER,2,HARMAMP,2.004748935V,HARMDBC,-6dBc,HARMPHASE,0

3.6 Mod Command

DESCRIPTION	Set or query modulation parameters.
COMMAND SYNTAX	<channel>:MoDulateWaVe <type> <channel>:MoDulateWaVe<parameter>,<value> <channel>:= { C1, C2, C3, C4} <type>:= {AM, DSBSC, FM, PM, PWM, ASK, FSK, PSK} <parameter>:= { The parameters in the table below } <value>:= { The values of relevant parameters }

Parameter	Value	Description
STATE	<state>	:= {ON, OFF}. Enable and disable modulation.Other parameters of Mod can only be set after enabling Mod.
AM,SRC	<src>	:= {INT, EXT, CH}. AM modulation source.
AM,MDSP	<mod wave shape>	:= {SINE, SQUARE, TRIANGLE, UPRAMP, DNRAMP, NOISE, ARB}. AM modulation waveform. This parameter can only be set when the modulation source is set to INT.
AM,FRQ	<AM frequency>	:= AM modulation frequency. The unit is Hertz "Hz", and the effective range of parameter values can be found in the data manual. This parameter can only be set when the modulation source is set to INT.
AM,DEPTH	<depth>	:= {0 to 120}. AM depth. The unit is'%'. This parameter can only be set when the modulation source is set to INT.

DSBSC,SRC	<src>	:= {INT, EXT, CH}. DSBSC modulation source.
DSBSC,MDSP	<mod wave shape>	:= {SINE, SQUARE, TRIANGLE, UPRAMP, DNRAMP, NOISE, ARB}. DSB-SC modulation waveform. This parameter can only be set when the modulation source is set to INT.
DSBSC,FRQ	<DSB-SC frequency>	:= DSB SC modulation frequency. The unit is Hertz "Hz", and the effective range of parameter values can be found in the data manual. This parameter can only be set when the modulation source is set to INT.
FM,SRC	<src>	:= {INT, EXT, CH}. FM modulation source.
FM,MDSP	<mod wave shape>	:= {SINE, SQUARE, TRIANGLE, UPRAMP, DNRAMP, NOISE , ARB}. FM modulation waveform. This parameter can only be set when the modulation source is set to INT.
FM,FRQ	<FM frequency>	:= FM modulation frequency. The unit is Hertz "Hz", and the effective range of parameter values can be found in the data manual. This parameter can only be set when the modulation source is set to INT.
FM,DEVI	<FM frequency deviation>	:= From {0 to the current carrier frequency}. FM frequency deviation. This value depends on the

		difference between the carrier frequency and the bandwidth frequency. This parameter can only be set when the modulation source is set to INT.
PM,SRC	<src>	:= {INT，EXT，CH}. PM modulation source.
PM,MDSP	<mod wave shape>	:= {SINE, SQUARE, TRIANGLE, UPRAMP, DNRAMP, NOISE, ARB}. PM modulation waveform. This parameter can only be set when the modulation source is set to INT.
PM,FRQ	<PM frequency>	:= PM modulation frequency. The unit is Hertz "Hz", and the effective range of parameter values can be found in the data manual. This parameter can only be set when the modulation source is set to INT.
PM,DEVI	<PM phase offset>	:= {0 to 360}. PM phase deviation. The unit is "°", and this parameter can only be set when the modulation source is set to INT.
PWM,SRC	<src>	:= {INT, EXT, CH}. PWM modulation source.
PWM,FRQ	<PWM frequency>	:= PWM modulation frequency. The unit is Hertz "Hz", and the effective range of parameter values can be found in the data manual. This parameter can only be set when the modulation source is set to INT.
PWM,DEVI	<PWM dev>	:= Duty cycle deviation. The

		unit is's'. The value depends on the pulse width of the carrier wave.
PWM,MDSP	<mod wave shape>	:= {SINE, SQUARE, TRIANGLE, UPRAMP, DNRAMP, NOISE, ARB}. PWM modulation waveform. This parameter can only be set when the modulation source is set to INT.
ASK,SRC	<src>	:= {INT, EXT_A, EXT_B}. ASK modulation source.
ASK,KFRQ	< key frequency>	:= ASK key frequency. The unit is Hertz "Hz", and the effective range of parameter values can be found in the data manual. This parameter can only be set when the modulation source is set to INT.
FSK,SRC	<src>	:= {INT, EXT_A,EXT_B}. FSK modulation source.
FSK,KFRQ	< key frequency>	:= FSK key frequency. The unit is Hertz "Hz", and the effective range of parameter values can be found in the data manual. This parameter can only be set when the modulation source is set to INT.
FSK,HFRQ	<FSK_hop_freq>	:= FSK hopping frequency. The unit is Hertz "Hz", and the effective range of parameter values can be found in the data manual. This parameter can only be set when the modulation source is set to INT.
PSK,SRC	<src>	:= {INT, EXT_A, EXT_B}.

		PSK modulation source.
PSK,KFRQ	< key frequency>	:= PSK key frequency. The unit is Hertz "Hz", and the effective range of parameter values can be found in the data manual. This parameter can only be set when the modulation source is set to INT.
CARR,WVTP	<wave type>	:= {SINE, SQUARE, RAMP, ARB, PULSE}. Carrier frequency type.
CARR,FRQ	<frequency>	:= Carrier frequency. The unit is Hertz "Hz", and the effective range of parameter values can be found in the data manual.
CARR,PHSE	<phase>	:= {0 to 360}. Carrier phase. The unit is' °'.
CARR,AMP	<amplitude>	:= Carrier amplitude. The unit is "Vpp". The effective range of parameter values can be found in the data manual.
CARR,OFST	<offset>	:= Carrier offset. The unit is “V”. The effective range of parameter values can be found in the data manual.
CARR,SYM	<symmetry>	:= {0 to 100}. When the carrier is a RAMP, the carrier symmetry. The unit is'% '.
CARR,DUTY	<duty>	:= {0 to 100}. When the carrier wave is square, the carrier duty cycle. The unit is'% '.
CARR,RISE	<rise>	:= When the carrier wave is square or pulse, the rise time of the carrier wave. The unit is 's'. The effective range of parameter values can be found in the data manual.

	CARR,FALL	<fall>	:= When the carrier wave is square or pulse, the fall time of the carrier wave. The unit is 's'. The effective range of parameter values can be found in the data manual.
	CARR,DLY	<delay>	:= When the carrier wave is square or pulse, the carrier wave is delayed. The unit is 's'. The effective range of parameter values can be found in the data manual.
QUERY SYNTAX	<channel>:MoDulateWaVe? <channel>:= {C1, C2, C3, C4}		
RESPONSE FORMAT	<channel>:MDWV<parameter> <parameter>:= { All parameters of current modulation type}		
EXAMPLE	Set ch1 modulation status to ON: *C1:MDWV STATE,ON* Set ch1 modulation type to AM: *C1:MDWV AM* Set AM modulation and set the modulation waveform to SINE: *C1:MDWV AM,MDSP,SINE* Query the modulation parameters of ch1: *C1:MDWV?* Return: *C1:MDWVAM,STATE,ON,MDSP,SINE,SRC,INT,FRQ,100HZ,* *DEPTH,100,CARR,WVTP,RAMP,FRQ,1000HZ,AMP,4V,AMPVRMS,* *1.15473Vrms,OFST,0V,PHSE,0,SYM,50* Set the FM modulation frequency of ch1 to 1000Hz: *C1:MDWV FM,FRQ,1000* Set the carrier of ch1 to SINE: *C1:MDWV CARR,WVTP,SINE* Set the ch1 carrier frequency to 10000Hz: *C1:MDWV CARR,FRQ,10000*		

3.7 Sweep Command

DESCRIPTION	This command is used to set or query parameters for sweep.
COMMAND SYNTAX	<channel>:SWEep <state> <channel>:= {C1, C2, C3, C4} <state>:= {ON,OFF} <channel>:SWEep:<parameter> <value> <channel>:= {C1, C2, C3, C4} <value>:= { The values of relevant parameters }

Parameter	Value	Description
TYPE	<type>	:= {FREQ, AMP, BOTH}.Type ofsweep.
SOURce	<src>	:= {INT, EXT_A, EXT_B, MAN_A, MAN_B}. The trigger source for sweep.
FMODe	<mode>	:= {LINE, LOG, STEP}. The type of sweep frequency.
AMODe	<mode>	:= {LINE, STEP}. The type of amplitude sweep.
FSNumber	<value>	:= An integer between 2 and 1024. The number of steps in the step frequency sweep.
ASNumber	<value>	:= An integer between 2 and 1024. The number of steps in the step amplitude sweep.
TIME	<value>	:= Floating point type values. The unit is seconds. Scanning time for sweep.
SHTime	<value>	:= Floating point type values. The unit is seconds. The starting and holding time of the sweep.
EHTime	<value>	:= Floating point type values. The unit is seconds. The end holding time of the sweep.
RTIMe	<value>	:= Floating point type values. The unit is seconds. The end and return time of the sweep.
SFRequency	<value>	:= Floating point type values. The

		unit is Hz. The starting frequency of the sweep frequency.
EFRequency	<value>	:= Floating point type values. The unit is Hz. The ending frequency of the sweep frequency.
CFRequency	<value>	:= Floating point type values. The unit is Hz. The center frequency of the sweep frequency.
FSPan	<value>	:= Floating point type values. The unit is Hz. The frequency range of sweep.
SAMPlitude	<value>	:= Floating point type values. The unit is Vpp. The starting amplitude of the sweep.
EAMPlitude	<value>	:= Floating point type values. The unit is Vpp. The end amplitude of the sweep.
CAMPlitude	<value>	:= Floating point type values. The unit is Vpp. The central amplitude of the sweep.
ASPan	<value>	:= Floating point type values. The unit is Vpp. The amplitude range of the sweep.
FDIRection	<direction>	:= {UP, DOWN, UP_DOWN}. The scanning direction of frequency sweep. UP-DOWN is only supported under linear sweep (LINE).
ADIRection	<direction>	:= {UP, DOWN, UP_DOWN}. The scanning direction of amplitude sweep. UP-DOWN is only supported under linear sweep (LINE).
FSYMmetry	<value>	:= Floating point numbers from 0 to 100. Linear frequency sweep. Set the symmetry of the UP_DOWN sweep.
ASYMmetry	<value>	:= Floating point numbers from 0 to 100. Linear amplitude sweep. Set the symmetry of the UP_DOWN sweep.

	TOUT	<state>	:= {ON, OFF}. The trigger output switch for sweep. It can only be set when the trigger source is INT or MAN.
	EDGe	<polarity>	:= {RISE, FALL}. When the sweep trigger source is EXT, set the edge of the source.
	MTRigger		When the trigger source is MAN, manually trigger the sweep once.
	MARKer<num>:FMARker	<state>	num = {1,2}, It's markers 1 and 2. state = {ON, OFF}. Flag switch status for frequency sweep.
	MARKer<num>:MFRequency	<value>	num = {1,2}, It is marker 1 and marker 2. Value=Floating point type value, the unit is Hertz (Hz). The marked frequency of sweep frequency.
	MARKer<num>:MSNumber	<value>	num = {1,2}, It is marker 1 and marker 2. Value={1~1024}. The marked number of steps for sweep frequency.
QUERY SYNTAX	<channel>:SWEep<parameter>? <channel>:= {C1, C2, C3, C4}		
EXAMPLE	Set the sweep function switch of ch1 to ON: *C1:SWEep ON* Set ch1's sweep type to frequency sweep: *C1:SWEep:TYPE FREQ* Query the sweep type of ch1sweep: *C1:SWEep:TYPE?* Return: *FREQ\n*		

3.8 Burst Command

DESCRIPTION	Set or query the burst parameter.
COMMAND SYNTAX	<channel>:BursTWaVe <parameter>,<value> <channel>:= {C1, C2, C3, C4} <parameter>:= { The parameters in the table below } <value>:= { The values of relevant parameters }

Parameter	Value	Description
STATE	<state>	:= {ON, OFF}. Enable or disable burst. STATE must be set to ON before you set or read other parameters of the burst.
PRD	<period>	:= burst period. Refer to the datasheet for the range of valid values. The unit is seconds “s” Not valid when: Carrier is NOISE GATE_NCYC is GATE TRSR is EXT
STPS	<start_phase>	:= {0 to 360}. Start phase of the carrier. The unit is "degree". Not valid when the carrier is NOISE or PULSE.
GATE_NCYC	<burst_mode>	:= {GATE, NCYC}. Burst mode. Not valid when the carrier is NOISE.
TRSR	<trig_src>	:= {EXT_A, EXT_B, INT, MAN_A, MAN_B}. Trigger source. EXT_A and EXT_B refers to External, INT refers to Internal . MAN_A and MAN_B refers to Manual.
MTRIG		:= send a manual trigger. Only when TRSR is MAN, the parameter is valid.
DLAY	<delay>	:= trigger delay. The unit is seconds "s". Refer to the datasheet for the range of

		valid values. Available when GATE_NCYC is NCYC. Not valid when the carrier is NOISE.
PLRT	<polarity>	:= {NEG, POS}. Gate polarity. Negative or Positive.
TRMD	<trig_mode>	:= {RISE, FALL,OFF}. Trigger out mode. Available when GATE_NCYC is NCYC and TRSR is INT or MAN. Not valid when the carrier is NOISE.
EDGE	<edge>	:={RISE, FALL}. Available trigger edge. Only valid when TRSR is EXT or MAN.
TIME	<circle_ time>	:={INF, 1, 2,..., M}, where M is the maximum supported Ncycle number which depends on the model; INF sets the burst to Infinite mode. Available when GATE_NCYC is NCYC. Not valid when the carrier is NOISE.
COUNT	<counter>	:=Burst count, Only valid when TRSR is EXT or MAN.
HOLD	<type>	:= {MIDDLE, START, END}.Idle level.
CARR, WVTP	<wave type>	:= {SINE, SQUARE, RAMP, ARB, PULSE, NOISE}. Carrier waveform type.
CARR, FRQ	<frequency>	:=carrier frequency. The unit is Hertz "Hz". Refer to the datasheet for the range of valid values.
CARR, PHSE	<phase>	:= {0 to 360}. Carrier phase. The unit is "degree".
CARR, AMP	<amplitude>	:=carrier amplitude. The unit is volts, peak-to-peak "Vpp". Refer to the datasheet for the range of valid values.
CARR, OFST	<offset>	:=carrier offset. The unit is

		volts "V". Refer to the datasheet for the range of valid values.
CARR, SYM	<symmetry>	:={0 to 100}. Carrier symmetry when the carrier is RAMP. The unit is "%".
CARR, DUTY	<duty>	:={0 to 100}. Carrier duty cycle when the carrier is SQUARE or PULSE. The unit is "%".
CARR, RISE	<rise>	:=rise time when the carrier is PULSE. The unit is seconds "s". Refer to the datasheet for the range of valid values.
CARR, FALL	<fall>	:=fall time when the carrier is PULSE. The unit is seconds "s". Refer to the datasheet for the range of valid values.
CARR, DLY	<delay>	:=pulse delay when the carrier is PULSE. The unit is seconds “s”. Refer to the datasheet for the range of valid values.
CARR, STDEV	<stdev>	:= standard deviation of NOISE. The unit is volts “V”. Refer to the datasheet for the range of valid values.
CARR, MEAN	<mean>	:= mean of NOISE. The unit is volts “V”. Refer to the datasheet for the range of valid values.

QUERY SYNTAX	<channel>:BTWV(BursTWaVe)? <channel>:= {C1, C2, C3, C4}
RESPONSE FORMAT	<channel>:BTWV <parameter> < parameter>:={All parameters of the current burst wave}
EXAMPLE	Set CH1 burst state to ON *C1:BTWV STATE,ON* Set CH1 burst period to 1 s: *C1:BTWV PRD,1* Set CH1 burst delay to 1s: *C1:BTWV DLAY,1*

	Set CH1 burst to infinite: *C1:BTWV TIME,INF* Read CH2 burst parameters when the STATE is ON: *C2:BTWV?* Return: *C2:BTWV STATE,ON,PRD,0.01S,STPS,0,TRSR,INT,* *TRMD,OFF,TIME,1,DLAY,2.4e-07S,GATE_NCYC,NCYC,* *CARR,WVTP,SINE,FRQ,1000HZ,AMP,4V,OFST,0V,PHSE,0* Read CH2 burst parameters when the STATE is OFF: *C2:BTWV?* Return: *C2:BTWV STATE,OFF*

3.9 AWG Command

When the layer is single layer or single wave, the numbers of Scenario and Sequence are set to 1.

3.9.1 <channel>:AWG:STATE <state>

DESCRIPTION	This command is used to set or query the running status of AWG.
COMMAND SYNTAX	<channel>:AWG:STATe <state> <channel>:= {C1, C2, C3, C4} <state>:= {STOP, RUN}
QUERY SYNTAX	<channel>:AWG:STATe? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the RUNning state of channel 1 to run: C1:AWG:STATE RUN Query the running status of channel 1: *C1:AWG:STATE?* Return: *RUN\n*

Note: All parameters need to be modified when the running status is STOP.

3.9.2 <channel>:AWG:DEFAult

DESCRIPTION	This command is used to set the default settings of AWG.
COMMAND SYNTAX	<channel>:AWG:DEFAult <channel>:= {C1, C2, C3, C4}

3.9.3 <channel>:AWG:SRATE <value>

DESCRIPTION	This command is used to set or query the sampling rate of AWG.
COMMAND SYNTAX	<channel>:AWG:SRATE <value> <channel>:= {C1, C2, C3, C4} <value>:= Sampling rate
QUERY SYNTAX	<channel>:AWG:SRATe?

	<channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the sampling rate of channel 1 to 200M m: *C1:AWG:SRATE 2e8* Query the sampling rate of channel 1. *C1:AWG:SRATE?* Return: *200000000\n*

3.9.4 <channel>:AWG:SCALe <value>

DESCRIPTION	This command is used to set or query the output amplitude ratio of AWG.
COMMAND SYNTAX	<channel>:AWG:SCALe <value> <channel>:= {C1, C2, C3, C4} <value>:={0, 1}
QUERY SYNTAX	<channel>: AWG:SCALe? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the amplitude ratio of channel 1 to 88%: *C1:AWG:SCALe 0.88* Query the amplitude ratio of channel 1: *C1:AWG:SCALe?* Return: *0.88\n*

3.9.5 <channel>:AWG:OFFset <value>

DESCRIPTION	This command is used to set (query) the offset of AWG.
COMMAND SYNTAX	<channel>:AWG:OFFset <value> <channel>:= {C1, C2, C3, C4} <value>:= { The range of values is different under different output modes }See the data sheet for the parameter range.
QUERY SYNTAX	<channel>:AWG:OFFset? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the offset of channel 1 to 0.01: *C1:AWG:OFFset 0.01*

	Query the offset of channel 1: *C1:AWG:OFFset?* Return: *0.01\n*

3.9.6 <channel>:AWG:DIFFset <value>

DESCRIPTION	This command is used to set (query) the diff offset of AWG.
COMMAND SYNTAX	<channel>:AWG:DIFFset <value> <channel>:= {C1, C2, C3, C4} <value>:= { The range of values is different under different output modes}See the data sheet for the parameter range.
QUERY SYNTAX	<channel>:AWG:DIFFset? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the diff offset of channel 1 to 0.01: *C1:AWG:DIFFset 0.01* Query the diff offset of channel 1: *C1:AWG:DIFFset?* Return: *0.01\n*

3.9.7 <channel>:AWG:INTPtype <type>

DESCRIPTION	This command is used to set (query) the interpolation type of AWG.
COMMAND SYNTAX	<channel>:AWG:INTPtype <type> <channel>:= {C1, C2, C3, C4} <type>:= {ZERO, LINEAR, SINC, SINC13, SINC27}
QUERY SYNTAX	<channel>:AWG:INTPtype? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the interpolation type of channel 1 to SINC: *C1:AWG:INTPtype SINC* Query the interpolation type of channel 1: *C1:AWG:INTPtype?* Return: *SINC\n*

3.9.8 <channel>:AWG:INCReasing <type>

DESCRIPTION	This command is used to set (query) the interpolation mode of AWG.
COMMAND SYNTAX	<channel>:AWG:INCReasing <type> <channel>:= {C1, C2, C3, C4} <type>:= {INT, ZERo, HLAS, DUPL}
QUERY SYNTAX	<channel>: AWG:INCReasing? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the interpolation mode of channel 1 to hold: *C1:AWG:INCReasing HLAS* Query the interpolation mode of channel 1: *C1:AWG:INCReasing?* Return: *HLAS\n*

3.9.9 <channel>:AWG:DECReasing <type>

DESCRIPTION	This command is used to set (query) the decimation type of AWG.
COMMAND SYNTAX	<channel>:AWG:DECReasing <type> <channel>:= {C1, C2, C3, C4} < type >:={ DECi, CTAi, CHEa }
QUERY SYNTAX	<channel>: AWG:DECReasing? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the decimation type of channel 1 to CHEa: *C1:AWG:DECReasing CHEa* Query the decimation type of channel 1: *C1:AWG:DECReasing?* Return: *CHEa\n*

3.9.10 <channel>:AWG:TRIGger:TIMer <value>

DESCRIPTION	This command is used to set (query) the timer time of AWG.
COMMAND SYNTAX	<channel>:AWG:TRIGger:TIMer <value> <channel>:= {C1, C2, C3, C4}
QUERY SYNTAX	<channel>:AWG:TRIGger:TIMer? <channel>:= {C1, C2, C3, C4}

EXAMPLE	The time for setting the trigger timing of channel 1 is 1 second: *C1:AWG:TRIGger:TIMer 1* Query the time when channel 1 triggers the timing: *C1:AWG:TRIGger:TIMer?* Return: *1\n*

3.9.11 <channel>:AWG:TRIGAger:SLOPe <type >

DESCRIPTION	This command is used to set (query) the edge trigger mode of EXTA.
COMMAND SYNTAX	<channel>:AWG:TRIGAger:SLOPe <type> <channel>:= {C1, C2, C3, C4} <type>:={ RISe, FALL, BOTH}
QUERY SYNTAX	<channel>:AWG:TRIGAger:SLOPe? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the trigger edge of EXTA of channel 1 as RISe: *C1:AWG:TRIGAger:SLOPe RISe* Query the trigger edge of EXTA of channel 1: *C1:AWG:TRIGAger:SLOPe?* Return: *RISe\n*

3.9.12 <channel>:AWG:TRIGBger:SLOPe <type >

DESCRIPTION	This command is used to set (query) the edge trigger mode of EXTB.
COMMAND SYNTAX	<channel>:AWG:TRIGBger:SLOPe <type> <channel>:= {C1, C2, C3, C4} <type>:={ RISe, FALL, BOTH}
QUERY SYNTAX	<channel>: AWG: TRIGBger:SLOPe? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the trigger edge of EXTA of channel 1 as RISe: *C1:AWG:TRIGBger:SLOPe RISe* Query the trigger edge of EXTA of channel 1 : *C1:AWG:TRIGBger:SLOPe?* Return : *RISe\n*

3.9.13 <channel>:AWG:TRIGger:DELAy <value>

DESCRIPTION	This command is used to set (query) the trigger delay of AWG.
COMMAND SYNTAX	<channel>:AWG:TRIGger:DELAy <value> <channel>:= {C1, C2, C3, C4}
QUERY SYNTAX	<channel>:AWG:TRIGger:DELAy? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the trigger delay of channel 1 to 20 ns: *C1:AWG:TRIGger:DELAy 2e-08* Query the trigger mode of channel 1 trigger delay: *C1:AWG:TRIGger:DELAy?* Return: *2e-08\n*

3.9.14 <channel>:AWG:HOLDtype <type>

DESCRIPTION	This command is used to set (query) the type of AWG idle hold level.
COMMAND SYNTAX	<channel>:AWG:HOLDtype <type> <channel>:= {C1, C2, C3, C4} <type>:= {MID, START, END, USER}
QUERY SYNTAX	<channel>:AWG:INTPtype? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the hold level type of channel 1 to END: *C1:AWG:HOLDtype END* Query the idle hold level of channel 1: *C1:AWG:HOLDtype?* Return: *END\n*

3.9.15 <channel>:AWG:USRHOLD <value>

DESCRIPTION	This command is used to set (query) the level under AWG custom idle hold level type.
COMMAND SYNTAX	<channel>:AWG:USRHOLD <value> <channel>:= {C1, C2, C3, C4} <value>:= Customize the level value of the hold level.
QUERY SYNTAX	<channel>:AWG:USRHOLD? <channel>:= {C1, C2, C3, C4}

EXAMPLE	Set the custom hold level of channel 1 to 0.3: *C1:AWG:USRHOLD 0.3* Query the custom hold level of channel 1: *C1:AWG:USRHOLD?* Return: *0.3\n*

3.9.16 <channel>:AWG:DYNA:TYPE <type>

DESCRIPTION	This command is used to set (query) the type of AWG dynamic mode.
COMMAND SYNTAX	<channel>:AWG:DYNA:TYPE <type> <channel>:= {C1, C2, C3, C4} <type>:= {JUMP_TYPE, CONTROL_TYPE}
QUERY SYNTAX	<channel>:AWG:DYNA:TYPE? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the dynamic jump type of channel 1 to CONTROL_TYPE: *C1:AWG:DYNA:TYPE CONTROL_TYPE* Query the dynamic jump type of channel 1: *C1:AWG:DYNA:TYPE?* Return: *CONTROL_TYPE\n*

3.9.17 <channel>:AWG:GATELevel <type>

DESCRIPTION	This command is used to set (query) the gating effective level of AWG.
COMMAND SYNTAX	<channel>:AWG:GATELevel <type> <channel>:= {C1, C2, C3, C4} <type>:= {NEGATIVE, POSITIVE}
QUERY SYNTAX	<channel>: AWG:GATELevel? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the gating active level of channel 1 to positive: *C1:AWG:GATELevel POSITIVE* Query the gating effective level of channel 1: *C1:AWG:GATELevel?* Return: *POSITIVE\n*

3.9.18 <channel>:AWG:SCENario:TIMer <value>

DESCRIPTION	This command is used to set (query) the Scenario timing time of AWG.
COMMAND SYNTAX	<channel>:AWG:SCENario:TIMer <value> <channel>:= {C1, C2, C3, C4} <value>:= Timing time
QUERY SYNTAX	<channel>:AWG:SCENario:TIMer? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the Scenario timing time of channel 1 to 0.01: *C1:AWG:SCENario:TIMer 0.01* Query Scenario timing time of channel 1: *C1:AWG:SCENario:TIMer?* Return: *0.01\n*

3.9.19 <channel>:AWG:SEQUence:TIMer <value>

DESCRIPTION	This command is used to set (query) the sequence timing time of AWG.
COMMAND SYNTAX	<channel>:AWG:SEQUence:TIMer <value> <channel>:= {C1, C2, C3, C4} <value>:= timing time
QUERY SYNTAX	<channel>:AWG:SEQUence:TIMer? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the sequence timing time of channel 1 to 0.01: *C1:AWG:SEQUence:TIMer 0.01* Query sequence timing time of channel 1 : *C1:AWG:SEQUence:TIMer?* Return: *0.01\n*

3.9.20 <channel>:AWG:SEGMent:TIMer <value>

DESCRIPTION	This command is used to set (query) the segment timing time of AWG.
COMMAND SYNTAX	<channel>:AWG:SEGMent:TIMer <value> <channel>:= { C1,C2,C3,C4}

	<value>:= timing time
QUERY SYNTAX	<channel>:AWG:SEGMent:TIMer? <channel>:= { C1,C2,C3,C4}
EXAMPLE	Set the segment timing time of channel 1 to 0.01: *C1:AWG:SEGMent:TIMer 0.01* Query segment timing time of channel 1: *C1:AWG:SEGMent:TIMer?* Return: *0.01\n*

3.9.21 <channel>:AWG:COMpensation <state>

DESCRIPTION	This command is used to set (query) the compensation switch state of AWG.
COMMAND SYNTAX	<channel>:AWG:COMpensation <state> <channel>:= { C1,C2,C3,C4} <state>:= {ON, OFF}
QUERY SYNTAX	<channel>:AWG:COMpensation? <channel>:= { C1,C2,C3,C4}
EXAMPLE	Set the compensation status of channel 1 to ON: *C1:AWG:COMpensation ON* Query the compensation switch status of channel 1: *C1:AWG:COMpensation?* Return: *ON\n*

3.9.22 <channel>:AWG:MCABle:STATe <state>

DESCRIPTION	This command is used to set (query) the cable matching switch state of AWG.
COMMAND SYNTAX	<channel>:AWG:MCABle:STATe <state> <channel>:= {C1, C2, C3, C4} <state>:= {ON, OFF}
QUERY SYNTAX	<channel>:AWG:MCABle:STATe? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the cable matching switch state of channel 1 to ON: *C1:AWG:MCABle:STATe ON*

	Query the cable matching switch status of channel 1: *C1:AWG:MCABle:STATe?* Return: *ON\n*

3.9.23 <channel>:AWG:MCABle:FILe <path>

DESCRIPTION	This command is used to set (query) the transfer function of AWG cable matching.
COMMAND SYNTAX	<channel>:AWG:MCABle:FILe <path> <channel>:= {C1, C2, C3, C4} <path>:= The path of the transfer function, including the file name, has the suffix. stf2. It can be a local path of the device, a network storage path or a USB flash drive path.
QUERY SYNTAX	<channel>:AWG:MCABle:FILe? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the transfer function of channel 1 cable matching to test.stf2 under the local path: *C1:AWG:MCABle:FILe "Local/test.stf2"* Set the transfer function of channel 1 cable matching to test.stf2 under the U disk path: *C1:AWG:MCABle:FILe "U-disk0/test.stf2"* Set the transfer function of channel 1 cable matching to test.stf2 under the network storage path: *C1:AWG:MCABle:FILe "net_storage/test.stf2"* Query the transfer function of channel 1 cable matching: *C1:AWG:MCABle:FILe?* Return: *Local/test.stf2\n*

3.9.24 <channel>:CREAt:CTFunction <path1>,<path2>

DESCRIPTION	This command is used to set AWG cable matching to generate transfer function stf2 file according to s2p file of cable S parameters.
COMMAND SYNTAX	<channel>:CREAt:CTFunction <path1>,<path2> <channel>:= {C1, C2, C3, C4} <path1>:= The path of s2p file of cable S parameter, including file name and suffix. s2p.

	<path2>:= Generate the saving path of the transfer function stf2 file, including the file name and the suffix. stf2. Note: The path can be local to the device, network storage or USB flash drive.
EXAMPLE	Set the channel 1 to generate the transfer function according to the s2p file of the U disk path and save it in the local path: *C1:CREAt:CTFunction "U-disk0/test.s2p","Local/test.stf2"* Set the channel 1 to generate the transfer function according to the s2p file of the network storage path and save it in the local path: *C1:CREAt:CTFunction "net_storage/test.s2p","Local/test.stf2"*

3.9.25 <channel>:CREAt:ATFunction <path1>,<path2>,<path3>

DESCRIPTION	This command is used to set AWG cable matching to generate transfer function stf2 file according to s2p file of cable S parameters and s1p file of port reflection coefficient.
COMMAND SYNTAX	<channel>:CREAt:CTFunction <path1>,<path2>,<path3> <channel>:= {C1, C2, C3, C4} <path1>:= The path of s2p file of cable S parameter, including file name and suffix. s2p. <path2>:= The path of s1p file of port reflection coefficient, including file name and suffix .s1p. <path3>:= Generate the saving path of the transfer function stf2 file, including the file name and the suffix. stf2. Note: The path can be local to the device, network storage or USB flash drive.
EXAMPLE	Set the channel 1 to generate the transfer function according to the s2p and s1p files of the U disk path and save it in the local path: *C1:CREAt:CTFunction "U-disk0/test.s2p",* *U-disk0/test.s1p","Local/test.stf2"* Set the channel 1 to generate the transfer function according to the s2p and s1p files of the network storage path and save it in the local path: *C1:CREAt:CTFunction "net_storage/test.s2p",* *"net_storage/test.s1p","Local/test.stf2"*

3.9.26 <channel>:AWG:RMODe <type>

DESCRIPTION	This command is used to set (query) the operation mode of AWG.
COMMAND SYNTAX	<channel>:AWG:RMODe <type> <channel>:= {C1,C2,C3,C4} <type>:= {CONT, TCON, GATE_EXTA, GATE_EXTB, ADV}
QUERY SYNTAX	<channel>: AWG:RMODe? <channel>:= {C1,C2,C3,C4}
EXAMPLE	Set the operation mode of channel 1 to advanced: *C1:AWG:RMODe ADV* Query the operation mode of channel 1: *C1:AWG:RMODe?* Return: *ADV\n*

3.9.27 <channel>:AWG:WMODe <type>

DESCRIPTION	This command is used to set (query) the layer of AWG.
COMMAND SYNTAX	<channel>:AWG:WMODe <type> <channel>:= {C1,C2,C3,C4} <type>:= { SINGLE_LAYER, MULTILAYER, SINGLE_WAVE}
QUERY SYNTAX	<channel>:AWG:WMODe? <channel>:= {C1,C2,C3,C4}
EXAMPLE	Set the layer of channel 1 to MULTILAYER: *C1:AWG:WMODe MULTILAYER* Query the layer of channel 1: *C1:AWG:WMODe?* Return: *MULTILAYER\n*

3.9.28 <channel>:AWG:DYNA:TABLe:ADD PATT,<value1>,SCEN,<value2>, SEQU,<value3>,SEGM,<value4>

DESCRIPTION	This command is used to add items to the dynamic jump table of AWG.
COMMAND SYNTAX	<channel>:AWG:DYNA:TABLe:ADD

	PATT,<value1>,SCEN,<value2>,SEQU,<value3>,SEGM,<value4> <channel>:= {C1, C2, C3, C4} <value1>:= [0, 255] <value2>:= [1, The maximum number of scenario currently set] <value3>:= [1, The maximum number of sequence currently set] <value4>:= [1, The maximum number of segment currently set]
QUERY SYNTAX	<channel>: AWG:DYNA:TABLe? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Add an item to the dynamic jump table of channel 1: *C1:AWG:DYNA:TABLe:ADD PATT,99,SCEN,1,SEQU,1,SEGM,1* Query the dynamic jump table of channel 1: *C1:AWG:DYNA:TABLe?* Return: *pattern:99,go_secn:0,go_sequ:0,go_segm:0\n"\n*

3.9.29 <channel>:AWG:DYNA:TABLe:DELEte <value>

DESCRIPTION	This command is used to delete items in the dynamic jump table of AWG.
COMMAND SYNTAX	<channel>:AWG:DYNA:TABLe:DELEte <value> <channel>:= {C1, C2, C3, C4} <value>:= [0, 255]
EXAMPLE	Set the dynamic jump table of channel 1 to delete one item: *C1:AWG:DYNA:TABLe:DELEte 0*

3.9.30 <channel>:AWG:DYNA:TABLe:CLEAR

DESCRIPTION	This command is used for clearing the dynamic jump table of AWG.
COMMAND SYNTAX	<channel>:AWG:DYNA:TABLe:CLEAR <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the dynamic jump table of channel 1 to empty: *C1:AWG:DYNA:TABLe:CLEAR*

3.9.31 <channel>:AWG:DYNA:TABLe?

DESCRIPTION	This command is used to query the dynamic jump table of AWG.
QUERY SYNTAX	<channel>:AWG:DYNA:TABLe? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Query the dynamic jump table of channel 1: *C1:AWG:DYNA:TABLe?* Return: *pattern:99,go_secn:0,go_sequ:0,go_segm:0\n"\n*

3.9.32 <channel>:AWG:TRIGger:SOURce <type>

DESCRIPTION	This command is used to set (query) the trigger mode in AWG trigger mode.
COMMAND SYNTAX	<channel>:AWG:TRIGger:SOURce <type> <channel>:= {C1, C2, C3, C4} <type>:={MANA, MANB, TIMe, EXTA, EXTB}
QUERY SYNTAX	<channel>:AWG:TRIGger:SOURce? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the trigger mode of channel 1 to manual trigger B: *C1:AWG:TRIGger:SOURce MANB* Query the trigger mode of channel 1 trigger mode: *C1:AWG:TRIGger:SOURce?* Return: *MANB\n*

3.9.33 <channel>:AWG:TRIGgerA

DESCRIPTION	This command is used to trigger the manual trigger button A of AWG once.
COMMAND SYNTAX	<channel>:AWG:TRIGgerA <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set AWG manual trigger a once: *C1:AWG:TRIGgerA*

3.9.34 <channel>:AWG:TRIGgerB

DESCRIPTION	This command is used to trigger the manual trigger button B of AWG once.
COMMAND SYNTAX	<channel>:AWG:TRIGgerB <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set AWG manual trigger a once: *C1:AWG:TRIGgerB*

3.9.35 <channel>:AWG:SAVE PATH,<path>,SWFS,<swfs>

DESCRIPTION	This command is used to save the AWG configuration and waveform data of the channel.
COMMAND SYNTAX	<channel>:AWG:SAVE PATH,<path>,SWFS,<swfs> <channel>:= {C1, C2, C3, C4} <path>:= The saved path and file name, with the suffix ".awgx". <swfs>:= {"TRUE", "FALSE"}, Used to determine whether to save the waveform data.
EXAMPLE	Channel 3 stores AWG configuration and waveform data. : *C3:AWG:SAVE PATH,"Local/aaa.awgx",SWFS,"TRUE"*

3.9.36 <channel>:AWG:LOAD PATH,<path>

DESCRIPTION	This command is used to load the AWG configuration and waveform data of the channel.
COMMAND SYNTAX	<channel>:AWG:SAVE PATH,<path>,SWFS,<swfs> <channel>:= {C1, C2, C3, C4} <path>:= The path and file name of the load, and the file name should be suffixed with ".awgx".
EXAMPLE	Channel 3 loading configuration and waveform: *C3:AWG:LOAD "Local/aaa.awgx"*

3.9.37 <channel>:AWG:SCENario:CLEAR

DESCRIPTION	This command is used to clear all segments of scenario of AWG designated channel.
COMMAND SYNTAX	<channel>:AWG:SCENario:CLEAR <channel>:= {C1, C2, C3, C4}
EXAMPLE	Channel 3 clears all Scenario segments: *C3:AWG:SCENario:CLEAR*

3.9.38 <channel>:AWG:SCENario:COUNt?

DESCRIPTION	This command is used to query the channel scenario number of AWG.
QUERY SYNTAX	<channel>:AWG:SCENario:COUNt? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Query the number of Scenario segments of channel 3: *C3:AWG:SCENario:COUNt?* Return: *3\n*

3.9.39 <channel>:AWG:SCENario:INSErt <pos>

DESCRIPTION	This command is used to insert a Scenario before AWG specifies the segment.
COMMAND SYNTAX	<channel>:AWG:SCENario:INSErt <pos> <channel>:= {C1, C2, C3, C4} <pos>:= [1, The maximum number of scenario segments currently set]
EXAMPLE	Insert a Scenario before the first Scenario of channel 3: *C3:AWG:SCENario:INSErt 2*

3.9.40 <channel>:AWG:SCENario:DELEte <pos>

DESCRIPTION	This command is used for AWG to delete a specified Scenario segment.
COMMAND SYNTAX	<channel>:AWG:SCENario:DELEte <pos> <channel>:= {C1, C2, C3, C4} <pos>:= [1, The maximum number of scenario segments currently set]
EXAMPLE	Delete the second Scenario of channel 3: *C3:AWG:SCENario:DELEte 2*

3.9.41 <channel>:AWG:SCENario:MULTIDelete <pos1, pos2, pos3...>

DESCRIPTION	This command is used for AWG to delete multiple specified Scenario segments.
COMMAND SYNTAX	<channel>:AWG:SCENario:MULTIDelete <pos1, pos2, pos3...> <channel>:= {C1, C2, C3, C4} <pos1, pos2, pos3...>:= [1, The maximum number of scenario segments currently set]
EXAMPLE	Delete the 2nd, 3rd and 6th Scenario of Channel 3: *C3:AWG:SCENario:MULTIDelete 2,3,6*

3.9.42 <channel>:AWG:SCENario<x>:LOOP <value>

DESCRIPTION	This command is used to set or query the number of cycles of the specified Scenario in AWG.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:LOOP<value> <x>:= scenario number <channel>:= {C1, C2, C3, C4} <value>:= integer
QUERY SYNTAX	<channel>:AWG:SCENario<x>:LOOP? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the loop of the first Scenario of channel 1 to 5: *C1:AWG:SCENario1:LOOP 5* Query the loop of the first Scenario in channel 1: *C1:AWG:SCENario1:LOOP?*

	Return: *5\n*

3.9.43 <channel>:AWG:SCENario:STARTNumb <value>

DESCRIPTION	This command is used to set the first Scenario to be played.
COMMAND SYNTAX	<channel>:AWG:SCENario:STARTNumb <value> <channel>:= {C1, C2, C3, C4} <value>:= integer
QUERY SYNTAX	<channel>:AWG:SCENario:STARTNumb? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the Scenario played first in channel 1 as the second: *C1:AWG:SCENario:STARTNumb 2* Query the Scenario segment number played first in channel 1: *C1:AWG:SCENario:STARTNumb?* Return: *2\n*

3.9.44 <channel>:AWG:SCENario<x>:WAITEvent <type>

DESCRIPTION	This command is used to set or query the wait event of Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:WAITEvent <type> <x>:= scenario number <channel>:= {C1, C2, C3, C4} <type>:= {AUTO, MANA, MANB, EXTA, EXTB, TIMe}
QUERY SYNTAX	<channel>: AWG:SCENario<x>:WAITEvent? <x>:= scenario number
EXAMPLE	Set the first Scenario waiting event of channel 1 as manual trigger A: *C1:AWG:SCENario1:WAITEvent MANA* Query the wait event of the first Scenario in channel 1: *C1:AWG:SCENario1:WAITEvent?* Return: *MANA\n*

3.9.45 <channel>:AWG:SCENario<x>:GOTO <value>

DESCRIPTION	This command is used to set or query the next Scenario after the specified scenario is played.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:GOTO <value> <x>:= scenario number <channel>:= {C1, C2, C3, C4} <value>:= [1,The maximum number of scenario segments currently set]
QUERY SYNTAX	<channel>:AWG:SCENario<x>:GOTO? <x>:= scenario number
EXAMPLE	Set the next Scenario to be played after the first scenario is played in channel 1 as the third scenario: *C1:AWG:SCENario1:GOTO 3* Query the next Scenario to be played after the first scenario is played in channel 1 : *C1:AWG:SCENario1:GOTO?* Return: *3\n*

3.9.46 <channel>:AWG:SCENario<x>:PLAYBack <type>

DESCRIPTION	This command is used to set or query the playback mode of the specified Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:PLAYBack <type> <x>:= scenario number <channel>:= {C1, C2, C3, C4} <type>:= { AUTO, SINGLE, CONDITIONAL}
QUERY SYNTAX	<channel>:AWG:SCENario<x>:PLAYBack? <x>:= scenario number
EXAMPLE	Set the playback mode of the first Scenario in channel 1 to single : *C1:AWG:SCENario1:PLAYBack SINGLE* Query the next Scenario in the first scenario of Channel 1 : *C1:AWG:SCENario1:PLAYBack?* Return : *SINGLE\n*

3.9.47 <channel>:AWG:SCENario<x>:OUTEvent <type>

DESCRIPTION	This command is used to set or query the jump event of the specified Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:OUTEvent <type> <x>:= scenario number <channel>:= {C1, C2, C3, C4} <type>:= {MANA, MANB, EXTA, EXTB, TIMe}
QUERY SYNTAX	<channel>:AWG:SCENario<x>:OUTEvent? <x>:= scenario number
EXAMPLE	Set the jumping event of the first Scenario in channel 1 as the Scenario timer: *C1:AWG:SCENario1:OUTEvent TIMe* Query the jump event of the first Scenario in channel 1: *C1:AWG:SCENario1:OUTEven?* Return: *TIMe\n*

Note: It is effective when the playback mode is conditional jump.

3.9.48 <channel>:AWG:SCENario<x>:SEQUence:STARTNumb <value>

DESCRIPTION	This command is used to set or query the starting sequence number of the specified Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence:STARTNumb <value> <x>:= scenario number <channel>:= {C1, C2, C3, C4} <value>:= [1, The maximum sequence number set by the current scenario]
QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence:STARTNumb? <x>:= scenario number
EXAMPLE	Set the first sequence number of the first Scenario of channel 1 to 1: *C1:AWG:SCENario1:SEQUence:STARTNumb 1* Query the starting sequence number of the first Scenario in channel 1: *C1:AWG:SCENario1:SEQUence:STARTNumb?* Return: *1\n*

3.9.49 <channel>:AWG:SCENario<x>:SAVE PATH,<path>, SWFS,<swfs>

DESCRIPTION	This command is used to save the specified Scenario configuration and waveform data.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SAVE ATH,<path>,SWFS,<swfs> <channel>:= {C1, C2, C3, C4} <x>:= scenario number <path>:= Save the path and file name, the file name suffix is. scen. <swfs>:= {"TRUE", "FALSE"}, to determine whether to save the waveform data.
EXAMPLE	Channel 3 saves the configuration and waveform data of the first SCENario: *C3:AWG:SCENario1:SAVE PATH,"Local/aaa.scen",SWFS,"TRUE"*

3.9.50 <channel>:AWG:SCENario:LOAD,<path>

DESCRIPTION	This command is used to load Scenario configuration and waveform data.
COMMAND SYNTAX	<channel>:AWG:SCENario:LOAD <path> <channel>:= {C1, C2, C3, C4} <path>:= The path and file name of the load, and the file name suffix is. scen.
EXAMPLE	Channel 3 loads scenario configuration file: *C3:AWG:SCENario:LOAD "Local/aaa.scen"*

3.9.51 <channel>:AWG:SCENario<x>:SEQUence:CLEAR

DESCRIPTION	This command is used to empty the list of sequence of the specified Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence:CLEAR <x>:= scenario number <channel>:= {C1, C2, C3, C4}
EXAMPLE	Clear all sequence of the first Scenario in channel 1: *C1:AWG:SCENario1:SEQUence:CLEAR*

3.9.52 <channel>:AWG:SCENario<x>:SEQUence:COUNt?

DESCRIPTION	This command is used to query the sequence number of the specified Scenario.
QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence:COUNt? <x>:= scenario number <channel>:= {C1, C2, C3, C4}
EXAMPLE	Query the sequence number of the first Scenario in channel 1 : *C1:AWG:SCENario1:SEQUence:COUNT?* Return *3\n*

3.9.53 <channel>:AWG:SCENario<x>:SEQUence:INSErt <pos>

DESCRIPTION	This command is used to insert a sequence in the specified Scenario to achieve the specified order.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence:INSErt <pos> <x>:= scenario number <channel>:= {C1, C2, C3, C4} <pos>:= [0, Number of sequence set] , Inserts the tag number of the sequence.
EXAMPLE	Insert a sequence in the first Scenario of channel 1 into the second : *C1:AWG:SCENario1:SEQUence:INSErt 2*

3.9.54 <channel>:AWG:SCENario<x>:SEQUence:DELEte <pos>

DESCRIPTION	This command is used to delete the specified sequence in the specified Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>: SEQUence:DELEte <pos> <x>:=scenario number <channel>:= {C1, C2, C3, C4} <pos>:= [0, Number of sequence set]
EXAMPLE	Delete the second sequence in the first Scenario of channel 1 : *C1:AWG:SCENario1:SEQUence:DELEte 2*

3.9.55 <channel>:AWG:SCENario<x>:SEQUence:MULTIDelete <pos1,pos2,pos3...>

DESCRIPTION	This command is used for AWG to delete multiple sequence specified in the specified Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:MULTIDelete <pos1, pos2, pos3...> <x>:= scenario number <channel>:= {C1, C2, C3, C4} <pos1, pos2, pos3...>:= [1, Number of sequence set]Used to determine the sequence number that needs to be deleted.
EXAMPLE	Delete the 2nd, 3rd and 6th sequence of the 2nd Scenario of channel 3: *C3:AWG:SCENario2:SEQUence:MULTIDelete 2,3,6*

3.9.56 <channel>:AWG:SCENario<x>:SEQUence<y>:LOOP <value>

DESCRIPTION	This command is used to set or query the number of cycles of sequence in Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:LOOP <value> <x>:= scenario number <y>:= sequence number <channel>:= {C1, C2, C3, C4} <value>:= integer
QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:LOOP? <x>:=scenario number <y>:= sequence number <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the first sequence cycle number of the first Scenario of channel 1 to 3: *C1:AWG:SCENario1:SEQUence1:LOOP 3* Query the first sequence cycle number of the first Scenario in channel 1: *C1:AWG:SCENario1:SEQUence1:LOOP?* Return: *3\n*

3.9.57 <channel>:AWG:SCENario<x>:SEQUence<y>: WAITEvent <type>

DESCRIPTION	This command is used to set or query the wait event of sequence in Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>: SEQUence<y>: WAITEvent <type> <x>:= scenario number <y>:= sequence number <channel>:= {C1, C2, C3, C4} <type>:= {AUTO, MANA, MANB, EXTA, EXTB, TIMe}
QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:WAITEvent? <x>:= scenario number <y>:= sequence number <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the first sequence waiting event in the first Scenario of channel 1 as manual trigger B: *C1:AWG:SCENario1:SEQUence1:WAITEvent MANB* Query the first sequence cycle number of the first Scenario in channel 1: *C1:AWG:SCENario1:SEQUence1:WAITEvent?* Return: *MANB\n*

3.9.58 <channel>:AWG:SCENario<x>:GOTO <value>

DESCRIPTION	This command is used to set or query the next sequence of sequence in Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:GOTO <value> <x>:= scenario number <y>:= sequence number <channel>:= {C1, C2, C3, C4} <value>:= [0, Number of sequence set]
QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:GOTO? <x>:=scenario number <y>:=sequence number <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set that the next play of the first sequence in the first Scenario of channel 1 is the third sequence: *C1:AWG:SCENario1:SEQUence1:GOTO 3* Query the next play sequence number of the first sequence of the first Scenario of channel 1:

	C1:AWG:SCENario1:SEQUence1:GOTO? Return: *3\n*

3.9.59 <channel>:AWG:SCENario<x>: SEQUence<y>:PLAYBack <type>

DESCRIPTION	This command is used to set or query the playback mode of sequence in Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:PLAYBack <type> <x>:= scenario number <y>:= sequence number <channel>:= {C1, C2, C3, C4} <type>:= {AUTO, SINGLE, CONDITIONAL}
QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:PLAYBack? <x>:= scenario number <y>:= sequence number <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the playback mode of the first sequence of the first Scenario of channel 1 to single: *C1:AWG:SCENario1:SEQUence1:PLAYBack SINGLE* Query the playback mode of the first sequence of the first Scenario in channel 1: *C1:AWG:SCENario1:SEQUence1:PLAYBack?* Return: *SINGLE\n*

3.9.60 <channel>:AWG:SCENario<x>:SEQUence<y>:OUTEvent <type>

DESCRIPTION	This command is used to set or query the jump event of sequence in Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:OUTEvent <type> <x>:= scenario number <y>:= sequence number <channel>:= {C1, C2, C3, C4} <type>:= {MANA, MANB, EXTA, EXTB, TIMe}
QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:OUTEvent? <x>:= scenario number <y>:= sequence number <channel>:= {C1, C2, C3, C4}

EXAMPLE	Set the first sequence jump event of the first Scenario of channel 1 to manually trigger a : *C1:AWG:SCENario1:SEQUence1:OUTEvent MANA* Query the jump event of the first sequence of the first Scenario in channel 1 : *C1:AWG:SCENario1:SEQUence1:OUTEvent?* Return : *MANA\n*

3.9.61 <channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent:CLEAR

DESCRIPTION	This command is used to clear the segment list of a sequence in a Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent:CLEAR <x>:= scenario number <y>:= sequence number <channel>:= {C1, C2, C3, C4}
EXAMPLE	Empty the Segment in the first sequence in the first Scenario of channel 1: *C1:AWG:SCENario1:SEQUence1:SEGMent:CLEAR*

3.9.62 <channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent:STARTNumb <value>

DESCRIPTION	This command is used to set the starting segment number of sequence in Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent:STARTNumb <value> <x>:= scenario number <y>:= sequence number <channel>:= {C1, C2, C3, C4} <value>:= segment number
QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent:STARTNumb? <x>:= scenario number <y>:= sequence number <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the starting Segment number of the first sequence of the first

	Scenario of channel 1 to 3: *C1:AWG:SCENario1:SEQUence1:SEGMent:STARTNumb 3* Query the starting Segment number of the first sequence in the first Scenario of channel 1: *C1:AWG:SCENario1:SEQUence1:SEGMent:STARTNumb?* Return: *3\n*

3.9.63 <channel>:AWG:SCENario<x>:SEQUence<y>:SAVE PATH,<path>,SWFS,<swfs>

DESCRIPTION	This command is used to save the configuration and waveform data of the specified sequence in the specified Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SAVE PATH,<path>,SWFS,<swfs> <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <path>:= Save the path and file name, the file name needs to be suffixed. seq. <swfs>:= {"TRUE", "FALSE"}
EXAMPLE	Channel 3 saves the configuration of the first sequence in the first scenario and does not contain waveform data: *C3:AWG:SCENario1:SEQUence1:SAVE PATH,"Local/aaa.seq",SWFS,"FALSE"*

3.9.64 <channel>:AWG:SCENario<x>:SEQUence:LOAD PATH,<path>

DESCRIPTION	This command loads the sequence configuration and waveform data file under the specified Scenario.
COMMAND SYNTAX	<channel>:AWG: SCENario <x>: SEQUence:LOAD <path> <channel>:= {C1, C2, C3, C4} <x>:= scenario number <path>:= The path and file name of the load, including the suffix. seq.
EXAMPLE	Channel 3 First Scenario Load File: *C3:AWG: SCENario1:SEQUence:LOAD "Local/aaa.seq"*

3.9.65 <channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent:COUNt?

DESCRIPTION	This command is used to query the segment number of a sequence in a Scenario.
QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent:COUNt? <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number
EXAMPLE	Query the number of the first sequence in the first Scenario of channel 3: *C3:AWG:SCENario1:SEQUence1:SEGMent:COUNT?* Return: *3\n*

3.9.66 <channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent:INSErt <pos>

DESCRIPTION	This command is used to insert a segment after a segment of a sequence in a Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent: INSErt<pos> <channel>:= {C1, C2, C3, C4} <x>:=scenario number <y>:=sequence number <pos>:= segment number
EXAMPLE	The first sequence in the first Scenario of channel 3 inserts a segment to the fifth: *C3:AWG:SCENario1:SEQUence1:SEGMent:INSErt 5*

3.9.67 <channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent: DELEte <pos>

DESCRIPTION	This command is used to delete a segment of a sequence of a Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent: DELEte<pos> <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number

	<pos>:= segment number
EXAMPLE	Delete the fifth segment of the first sequence in the first Scenario of channel 3: *C3:AWG:SCENario1:SEQUence1:SEGMent:DELEte 5*

3.9.68 <channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent:MULTIDelete <pos1,pos2,pos3...>

DESCRIPTION	This command is used to delete multiple segment of the specified sequence of the specified Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent:MULTIDelete <pos1, pos2, pos3...> <x>:= scenario number <y>:= sequence number <channel>:= {C1, C2, C3, C4} <pos1, pos2, pos3...>:= [1, Number of segment set]
EXAMPLE	Delete the 2nd, 3rd and 6th segment of the 2nd sequence in the 2nd Scenario of channel 3: *C3:AWG:SCENario2:SEQUence2:SEGMent:MULTIDelete 2,3,6*

3.9.69 <channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>:LOOP <value>

DESCRIPTION	This command is used to set or query the number of cycles of a segment of a sequence in a Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>:LOOP <value> <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number <value>:= integer
QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>:LOOP? <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number

	<z>:= segment number
EXAMPLE	Set the number of first segment cycles of the first sequence of the first Scenario of channel 1 to 5: *C1:AWG:SCENario1:SEQUence1:SEGMent1:LOOP 5* Query the number of first segment cycles of the first sequence of the first Scenario of channel 1: *C1:AWG:SCENario1:SEQUence1:SEGMent1:LOOP?* Return: *5\n*

3.9.70 <channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>: WAITEvent <type>

DESCRIPTION	This command is used to set or query the waiting event of a segment of a sequence of a Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>: WAITEvent <type> <channel>:= {C1, C2, C3, C4} <x>:=scenario number <y>:=sequence number <z>:= segment number <type>:= {AUTO, MANA, MANB, EXTA, EXTB, TIMe}
QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>: SEGMent<z>:WAITEvent? <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number
EXAMPLE	Set the waiting event of the first segment of the first sequence of the first Scenario of channel 1 as manual trigger B: *C1:AWG:SCENario1:SEQUence1:SEGMent1:WAITEvent MANB* Query the wait event of the first segment of the first sequence of the first Scenario of channel 1: *C1:AWG:SCENario1:SEQUence1:SEGMent1:WAITEvent?* Return: *MANB\n*

3.9.71 <channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>: GOTO <value>

DESCRIPTION	This command is used to set or query the next segment of a sequence of a Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>: GOTO <value> <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number <value>:= { Integer, not exceeding the maximum number of segment}
QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>:GOTO? <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number
EXAMPLE	Set the first Segment of the first sequence of the first Scenario of channel 1 and the next segment of 3 : *C1:AWG:SCENario1:SEQUence1:SEGMent1:GOTO 3* Query the next of the first segment of the first sequence of the first Scenario of channel 1 : *C1:AWG:SCENario1:SEQUence1:SEGMent1:GOTO?* Return : *3\n*

3.9.72 <channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>: PLAYBack <type>

DESCRIPTION	This command is used to set or query the playback mode of a segment of a sequence of a Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>: PLAYBack <type> <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number <type>:= {AUTO, SINGLE, CONDITIONAL}

QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>:PLAYBack? <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number
EXAMPLE	Set the playback mode of the first segment of the first sequence of the first Scenario of channel 1 to single: *C1:AWG:SCENario1:SEQUence1:SEGMent1:PLAYBack SINGLE* Query the playback mode of the first segment of the first sequence of the first Scenario of channel 1: *C1:AWG:SCENario1:SEQUence1:SEGMent1:PLAYBack?* Return: *SINGLE\n*

3.9.73 <channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>: OUTEvent <type>

DESCRIPTION	This command is used to set or query the jump event of a segment of a sequence of a Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>:OUTEvent <type> <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number <type>:= {MANA, MANB, EXTA, EXTB, TIMe}
QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>:OUTEvent? <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number
EXAMPLE	Set the jump event of the first segment of the first sequence of the first Scenario of channel 1 as the segment timer: *C1:AWG:SCENario1:SEQUence1:SEGMent1:OUTEvent TIMe* Query the jump event of the first segment of the first sequence of the first Scenario of channel 1:

	C1:AWG:SCENario1:SEQUence1:SEGMent1:OUTEvent? Return: *TIMe\n*

3.9.74 <channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>: WAVeform <wavename>

DESCRIPTION	This command is used to set or query the waveform of a segment of a sequence of a Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z> :WAVeform <wavename> <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number <wavename >:= { For the name of the built-in wave table, see 3.5.3AFG's Arb Waveform Setting Command} or <wavename >:= { File path (including waveform name and suffix)} Note: The waveform name or waveform path needs double quotation marks.
QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>: WAVeform? <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number
EXAMPLE	Set the waveform of the first segment of the first sequence of the first Scenario of channel 1 as the built-in Noise: *C1:AWG:SCENario1:SEQUence1:SEGMent1:WAVeform "noise"* Set the waveform of the first segment of the first sequence of the first Scenario of channel 1 as wave.bin under the Local path: *C1:AWG:SCENario1:SEQUence1:SEGMent1:WAVeform "Local/wave1.bin"* Query the waveform name of the first segment of the first sequence of the first Scenario of channel 1: *C1:AWG:SCENario1:SEQUence1:SEGMent1:WAVeform?* Return: *wave1\n*

3.9.75 <channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>: AMPlitude <value>

DESCRIPTION	This command is used to set the amplitude of a segment waveform of a sequence of a Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z> :AMPlitude <value> <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number <value>:= Amplitude, unit Vpp
QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z> :AMPlitude? <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number
EXAMPLE	Set the waveform amplitude of the first segment of the first sequence of the first Scenario of channel 1 to 0.3 Vpp: *C1:AWG:SCENario1:SEQUence1:SEGMent1:AMPlitude 0.3* Query the waveform amplitude of the first segment of the first sequence of the first Scenario of channel 1.: *C1:AWG:SCENario1:SEQUence1:SEGMent1:AMPlitude?* Return: *0.3\n*

3.9.76 <channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>: OFFset <value>

DESCRIPTION	This command is used to set or query the offset of a segment waveform of a sequence of a Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z> :OFFset <value> <channel>:= {C1, C2, C3, C4} <x>:= scenario number

	<y>:= sequence number <z>:= segment number <value>:= Offset,The unit is Vdc.
QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>:OFFset? <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number
EXAMPLE	Set the waveform offset of the first segment of the first sequence of the first Scenario of channel 1 to 0.3 Vdc: *C1:AWG:SCENario1:SEQUence1:SEGMent1:OFFset 0.3* Query the waveform offset of the first segment of the first sequence of the first Scenario of channel 1: *C1:AWG:SCENario1:SEQUence1:SEGMent1:OFFset?* Return: *0.3\n*

3.9.77 <channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>: VOLTage:HIGH <value>

DESCRIPTION	This command is used to set or query the high level of a segment waveform of a sequence of a Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>:VOLTage:HIGH <value> <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number <value>:= High level, unit v.
QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>:VOLTage:HIGH? <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number
EXAMPLE	Set the waveform high level of the first segment of the first sequence

	of the first Scenario of channel 1 to 0.3 V: *C1:AWG:SCENario1:SEQUence1:SEGMent1:VOLTage:HIGH 0.3* Query the waveform high level of the first segment of the first sequence of the first Scenario of channel 1: *C1:AWG:SCENario1:SEQUence1:SEGMent1:VOLTage:HIGH?* Return: *0.3\n*

3.9.78 <channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>: VOLTage: LOW <value>

DESCRIPTION	This command is used to set or query the low level of a segment waveform of a sequence of a Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>:VOLTage: LOW <value> <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number <value>:= Low level, unit v
QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>:VOLTage: LOW? <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number
EXAMPLE	Set the waveform low level of the first segment of the first sequence of the first Scenario of channel 1 to -0.3 V: *C1:AWG:SCENario1:SEQUence1:SEGMent1:VOLTage:LOW -0.3* Query the waveform low level of the first segment of the first sequence of the first Scenario of channel 1: *C1:AWG:SCENario1:SEQUence1:SEGMent1:VOLTage:LOW?* Return: *-0.3\n*

3.9.79 <channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>:LENGth <value>

DESCRIPTION	This command is used to set or query the length of a segment waveform of a sequence of a Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z> :LENGth <value> <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number <value>:= Waveform length
QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z> :LENGth? <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number
EXAMPLE	Set the waveform length of the first segment of the first sequence of the first Scenario of channel 1 to 5000000. : *C1:AWG:SCENario1:SEQUence1:SEGMent1:LENGth 5000000* Query the waveform length of the first segment of the first sequence of the first Scenario of channel 1 : *C1:AWG:SCENario1:SEQUence1:SEGMent1:LENGth?* Return : *5000000\n*

3.9.80 <channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>:MARK1er:POS<K> <state>

DESCRIPTION	This command is used to set or query the marker1 switch state of segment.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z> :MARK1er:POS<K> <state> <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number

	<z>:= segment number <K>:= Tag number, waveform data length. < state >:= {ON, OFF}
QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z> :MARK1er? <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number <K>:= Tag number, waveform data length.
EXAMPLE	Set the data 1 of the mark 1 of the first segment of the first sequence of the first Scenario of channel 1 to open: *C1:AWG:SCENario1:SEQUence1:SEGMent1:MARK1er:POS1 ON* Query the marker1 status of the first segment of the first sequence of the first Scenario of channel 1: *C1:AWG:SCENario1:SEQUence1:SEGMent1:MARK1er?* Return: *"0,1,2,4"\n* # Data 0, 1, 2 and 4 data bits are marked on.

3.9.81 <channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>:MARK2er:POS<K> <state>

DESCRIPTION	This command is used to set or query the marker2 switch state of segment.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z> :MARK2er:POS<K> <state> <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number <K>:= Tag number, waveform data length. < state >:= {ON, OFF}
QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z> :MARK2er? <channel>:= {C1, C2, C3, C4} <x>:= scenario number

	<y>:= sequence number <z>:= segment number <K>:= Tag number, waveform data length.
EXAMPLE	Set the data 1 of the mark 2 of the first segment of the first sequence of the first Scenario of channel 1 to open: *C1:AWG:SCENario1:SEQUence1:SEGMent1:MARK2er:POS1 ON* Query the marker2 status of the first segment of the first sequence of the first Scenario of channel 1: *C1:AWG:SCENario1:SEQUence1:SEGMent1:MARK2er?* Return: *"0,1,2,4"\n* # Data 0, 1, 2 and 4 data bits are marked on.

Note: the numbers of scenario, sequence and segment in a single-wave command are all 1. For example, the length under a single wave is set as follows:

C1:AWG:SCENario1:SEQUence1:SEGMent1:LENGth 5000000

3.10 Hopping Command

DESCRIPTION	This command is used to set or query parameters of frequency hopping function.
COMMAND SYNTAX	<channel>:FHOP<parameter> <value> <channel>:= {C1, C2, C3, C4} <parameter>:={Parameters in the following table} <value>:= { Values of the following table parameters}

Parameter	Value	Description
SWITch	<state>	:= {ON,OFF}. Set frequency hopping output state.
TYPE	<type>	:= {MANUAL,RHOP,RLIST}. Set frequency hopping mode.
TIME	<time>	:= {Floating point numbers from 0 to 1000}. The unit is seconds. Set the frequency hopping time.
SFREquency	<value>	:= {Min_Freq ~ 2 GHz }, Set the starting frequency of random frequency hopping.
EFREquency	<value>	:= {Min_Freq ~ 2 GHz }, Set the end frequency of random frequency hopping.
FSTep	<value>	:= {Min_Freq ~ 2 GHz }, Set the frequency step of random frequency hopping.
RPATtern	<value>	:= {3~32}.Set prbs for random frequency hopping.
RLPATtern	<value>	:= {3~32 }.Set prbs for random list frequency hopping.
ALSTate	<state>	:= {ON, OFF}. Set the enabling state of the frequency filter table for random frequency hopping.
AFLIst <index>	<value>	:<index>= {1~4096 }. :<value>= {Min_Freq ~ 2 GHz }.Insert a new item before the specified item in the frequency table and set the frequency value.

MFLIst <index>	<value>	:<index>= {1~4096} :<value>= {Min_Freq ~ 2 GHz }.Modify the frequency value of the specified item in the frequency table.
DFLIst <index>		:<index>= {1~4096}. Delete the specified item of frequency table.
CFLIst		Clear the frequency table and return to the default settings.
AOLIst <index>	<freq_num>	:<index>= {1~4096 }. :<freq_num>:= {1~4096 }.Insert a new item before the specified item in the frequency jump table and set the frequency value.
MOLIst <index>	<freq_num>	:<index> ={1~4096 }. :<freq_num>:= {1~4096 }.Modify the frequency value of the specified item in the frequency jump table.
DOLIst <index>		:<index>= {1~4096 }. Delete the specified item of frequency jump table.
COLIst		Clear the jump frequency table and return to the default settings.
AALIst	<start_freq, end_freq>	:<start_freq>= {Min_Freq ~ 2 GHz } :<end_freq>:= {Min_Freq ~ 2 GHz }. Insert a new item in the frequency filter table and set the frequency value.
MALIst <index>	<start_freq, end_freq>	:<index>= {1~4096}. :<start_freq>={Min_Freq ~ 2 GHz }. :<end_freq>:= {Min_Freq ~ 2 GHz }. Modify the frequency value of the specified item in the frequency filter table.
DALIst <index>		:<index>= {1~4096 }.

		Delete the specified item in the frequency filter table.
CALIst		Empty the frequency filter table and return to the default settings.
LFLIst	<file>	:= File name (including path). Save the configuration of the frequency table. The path and file name need double quotation marks.
SFLIst	<file>	:= File name (including path). Load the configuration of frequency table. The path and file name need double quotation marks, and the path needs to exist in the file system.
LOLIst	<file>	:= File name (including path). Save the configuration of frequency hopping sequence table. The path and file name need double quotation marks.
SOLIst	<file>	:= File name (including path). Configuration of loading frequency hopping sequence table. The path and file name need double quotation marks, and the path needs to exist in the file system.
LALIst	<file>	:= File name (including path). Save the configuration of frequency filter table. The path and file name need double quotation marks.
SALIst	<file>	:= File name (including path). Configuration of loading frequency filter table. The path and file name need double quotation marks, and the path needs to exist in the file system.

QUERY SYNTAX <channel>:FHOP:<parameter>?

	<channel>:= {C1, C2, C3, C4}
EXAMPLE	The frequency filtering table of channel 1 is saved to the file avoid.hop: *C1:FHOP:SOLIst "order.hop"* Modify the frequency filter table of channel 1, and the first item frequency is 1K to 10K: *C1:FHOP:MALIst 1,1000,10000* Query the status of channel 1 frequency hopping switch: *C1:FHOP:SWITch?* Return: *"ON"*

3.11 Multitone Command

DESCRIPTION	This command is used to set or query the parameters of multi-tone function.
COMMAND SYNTAX	<channel>:MTONE:<parameter> <value> <channel>:= {C1, C2, C3, C4} <parameter>:={Parameters in the following table} <value>:= { Values of related parameters in the following table }

Paramerter	Value	Description
STAte	<state>	:= {ON, OFF}.Set to switch to multi-tone function. The premise is to set AWG mode, and all parameters of multitone can be set only when this command is "ON".
RSTAte	<state>	:= {ON, OFF}. Set the running state of multi-tone function. The premise is to set the multi-tone function, and all parameters of multi-tone can only be set in the "OFF" state.
STYPe	<type>	:={MANUAL,AUTO}. Set the sampling rate type.
SRATe	<value>	:= {100 Sa/s~5 GSa/s}.Set the multi-tone output sampling rate.
AMPlitude	<value>	:= Amplitude. The unit is volts, and the peak-to-peak value is "Vpp". Different output modes have different setting ranges. Please refer to the effective range of parameter values in the data sheet.
FTYPe	<type>	:={START,CENTER}.Frequency type switching (starting frequency, center frequency).
SFREquency	<value>	:= {Min~2 GHz}. Set the starting frequency.
FSTep	<value>	:= {Min~2 GHz }. Set the frequency step.
TNUMber	<value>	:= {2~1000 }. Set the number of multitone.
CFREquency	<value>	:= {Min~2 GHz }.Set the center frequency.

FINTerval	<value>	: = {Min to 2 GHz}. Set the frequency interval.
STONes	<value>	: = {1 to 500}. Set the number of single-sided tones.
LTYPe	<type>	:={MANUAL,AUTO}. Set the multi-tone waveform length type.
WLENgth	<value>	: ={262144 ~ 33554432}. Set the customized multi-tone waveform length when the multi-tone waveform length is MANUAL. The setting value can only be the square value of 2 in the range. When the setting value is not the square value of 2, the square value of 2 closest to the setting value will be set.
AMTList		According to the starting frequency, ending frequency, tone number and notch table, add frequency points to multi-tone table.
ATLIst	<value>	:= {Min_Freq ~ 2 GHz},The unit is Hz. Add a frequency point to the multi-tone table.
DTLIst	<value>	:= {integer}.Delete the frequency point with the specified serial number in the multitone table.
CTLIst		Clear all items of the multitone table.
STLIst		The frequency points of all items in the multi-tone table are merged with duplicates, and sorted from small to large.
MTLIst	<index, value>	:<index>={integer}, Frequency point number of multi-tone table. :<value>= {Min_Freq ~2 GHz }, Frequency value of multi-tone table, The unit is Hz. Set the frequency of multi-tone table according to the serial number.
TLISt		Query all items of multi-tone table.
NSTAte	<state>	:= {ON,OFF}. Set the on and off state of Notch table.
ANLIst	<value1,	Value1/2:= {Min_Freq ~ 2 GHz}, The unit

	value2>	is Hz. Add an item to the Notch table.
DNLIst	<value>	:= {integer}.Delete the specified serial number item in Notch table.
CNLIst		Empty the Notch table and restore the default values.
MNLIst	<index, value1, value2>	:<index>= {integer}. Sequence number of notch table. :<value1>= {Min_Freq~2GHz }. Frequency value of Notch table starting value, the unit is Hz. :<value2>= {Min_Freq~2GHz }. Frequency value of the end value of Notch table, the unit is Hz. Specifies the sequence number to set the frequency of the Notch table.
NLIst		Query all items in Notch table.

QUERY SYNTAX	<channel>:MTONE:<parameter>?
EXAMPLE	Channel 1 sets the multitone number to 3: *C1:MTONE:TNUMber 3* Channel 1 sets the frequency of multi-tone table No.1 to 6 MHz: *C1:MTONE:MTLIst 1,6e6* Query the multitone number of channel 1: *C1:MTONE:TNUMber?* Return: *"3\n"* Query all items of channel 1 multi-tone table: *C1:MTONE:TLIst?* Return: *"1,6000000.000000;2,10000.000000\n"*

3.12 Chirp Command

DESCRIPTION	This command is used to set or query the parameters of chirp function.
COMMAND SYNTAX	<channel>:CHIRp:<parameter> <value> <channel>:= {C1, C2, C3, C4} <parameter>:={Parameters in the following table} <value>:= {Values of the following table parameters}

Parameter	Value	Description
STAte	<state>	:= {ON,OFF}.Set to switch to chirp function. All parameters of chirp can be set only when this instruction is "ON".
RSTAte	<state>	:= {ON, OFF}. Set the operation status of chirp function. The premise is that chirp is set first, and all parameters of chirp can be set only when this instruction is in the "OFF" state.
SRATe	<value>	:= {100 Sa/s~5 GSa/s}.Set the chirp output sampling rate.
AMPlitude	<value>	:= Amplitude. The unit is volts, and the peak-to-peak value is "Vpp". Different output modes have different setting ranges. Please refer to the effective range of parameter values in the data sheet.
OFFSet	<value>	:= Offset. The unit is "Vdc". Different output modes have different setting ranges. Please refer to the effective range of parameter values in the data sheet.
HVTYpe	<type>	Set the idle level when no chirp is output. := {LOW, MIDDLE, HIGH }
TRIGger:SOURce	<type>	Set the trigger source of chirp. := {INT, MANA, MANB, EXTA, EXTB, TIMER}
TRIGger:TIMer	<value>	When the trigger source is timer, set the timer time. The unit is seconds.
TRIGger:DELAy	<value>	When the trigger source is manual or external trigger, set the trigger delay time.The unit is seconds.
TRIGger:SLOPe	<type>	When the trigger source is external trigger, set the trigger

		edge. = {RISe，FALL}
ATList	<start_freq, end_freq, duration>	Add a row at the end of chirp table. Start_freq: the starting frequency; End_freq: end frequency; Duration: scanning time. The units of start_freq and end_freq are Hz, and the units of duration are seconds.
ILIst	<index, start_freq, end_freq, duration>	Insert a row before the selected row (index) in the chirp table. Index: the serial number of the selected line; Start_freq: the starting frequency; End_freq: end frequency; Duration: scanning time. The index starts from 1, the units of start_freq and end_freq are Hz, and the unit of duration is seconds.
DLIst	<index>	:= { integer }.Index starts from 1. Delete the row with the specified serial number in the chirp table.
CLIst		Clear all rows of the chirp table and set them as default values.
MLIst	<index, start_freq, end_freq, duration>	Modify the parameters of the selected row in chirp table. Index: the serial number of the selected line; Start_freq: the starting frequency; End_freq: end frequency; Duration: scanning time. The index starts from 1, the units of start_freq and end_freq are Hz, and the unit of duration is seconds.
LIst		Query all items of chirp table.
QUERY SYNTAX	<channel>:CHIRp:<parameter>?	
EXAMPLE	Channel 1 sets the chirp sampling rate to 3 Gsa/s:	

	C1:CHIRp:SRATe 3e9 Channel 1 sets the starting frequency of chirp Table No.2 to 20 kHz, the ending frequency to 1 kHz, and the scanning time to 100 us: *C1:CHIRp:MLIst 2,2e4,1e3,1e-4* Query the sampling rate of chirp of channel 1: *C1:CHIRp:SRATe?* Return: *"3000000000\n"* Query all items in chirp table of channel 1: *C1:CHIRp:LIst?* Return: *"1,10000000.000000,100000000.000000,0.000100;2,20000.000000,1000.000000,0.000100\n"*

Note: manual triggering of scpi is the same as that of AWG.

See details <channel>:AWG:TRIGgerA3.9.33 and <channel>:AWG:TRIGgerB3.9.34

3.13 Multi Pulse Command

DESCRIPTION	This command is used to set or query the parameters of multi-pulse function.
COMMAND SYNTAX	<channel>:MPULse:<parameter> <value> <parameter>:={Parameters in the following table} <value>:= {Values of the following table parameters}

Parameter	Value	Description
STAte	<state>	:= {ON, OFF}.Set to switch to multi-pulse function. The premise is that AWG mode is set, and all parameters of multi-pulse can be set only in "ON" state.
SRate:TYPe	<type>	: = {AUTO, CUSTOM}. Set the sampling rate type of multi-pulse.
SRate	<value>	: ={100Sa/S~5GSa/s}. Set the sampling rate when the sampling rate type is CUSTOM.
RSTAte	<state>	:= {ON, OFF}.Set the running state of multi-pulse function. The premise is to set the multi-pulse function, and all parameters of multi-pulse can be set only when they are in the "OFF" state.
PNUMber	<value>	:= {2~30 }. Set the number of pulses.
HLEVel	<value>	:= { floating-point number }.Set the high level of the pulse. The setting range of different output modes is different. See the product data sheet for details.
LLEVel	<value>	:= { floating-point number }.Set the low level of the pulse. The setting range of different output modes is

			different. See the product data sheet for details.
	HVTYpe	<type>	:= {LOW, MIDDLE, HIGH, ZERO}, Set the idle level of multi-pulse.
	TRIGger:SOURce	<str>	:= {INT, MANA, MANB, EXTA, EXTB, TIMER}, Set the trigger source of multi-pulse.
	TRIGger:TIMer	<value>	:= {min~10s}, Set the timer time of multi-pulse.
	TRIGger:DELAy	<value>	:= {min~10s}, When the trigger source is manual or external, set the trigger delay.
	TRIGger:SLOPe	<value>	:= {RISe, FALL}. When the trigger source is external, set the trigger edge.
	MPULse	<num,width, dap>	:num= { Sequence number of pulse } :width= { Pulse width } :gap= { Pulse gap } Set the width and gap of each pulse according to the serial number of the pulse.
QUERY SYNTAX	<channel>:MPULse:<parameter>? Note: use the following scpi to query the width and gap of all pulses. <channel>:MPULse:PULse?		
EXAMPLE	Set the number of pulses in channel 1 to 3: *C1:MPULse:PNUMber 3* Set the width of pulse No.2 in channel 1 to 6 ms and the gap to 9 ms: *C1:MPULse MPULse 2,0.006,0.009* Query the number of pulses in channel 1: *C1:MPULse:PNUMber?* Return: *"3\n"* Query all pulse settings of channel 1:		

	C1:MPULse:PULse? Return: *"1,0.000001,0.000004;2,0.006,0.009\n"*

Note: manual triggering of scpi is the same as that of AWG.

See details <channel>:AWG:TRIGgerA3.9.33 and <channel>:AWG:TRIGgerB3.9.34

3.14 HSS Command

DESCRIPTION	This command is used to set or query the parameters of hssl function.
COMMAND SYNTAX	<channel>:HSS:<parameter> <value> <channel>:= {C1, C2, C3, C4} <parameter>:={Parameters in the following table} <value>:= {Values of the following table parameters}

Parameter	Value	Description
SSTAte	<state>	:= {ON,,OFF}.Set to switch to hss function. The premise is that AWG mode is set, and all parameters of hss can be set only in "ON" state.
SRAte	<value>	:={100Sa/S~5GSa/s}. Set the sampling rate of hss.
DRATe	<value>	:= {500Kbps~2.5Gbps }. Set the signal output data rate.
AMPL	<value>	:= Amplitude.
RSTAte	<state>	:= {ON,,OFF}.Set the running state of hss, and all other parameters can be set only when the running state is OFF.
PATTern	<type>	:={CLOCK,ZERO,ONE,PRBS,CUSTOM_PRBS,CUSTOM}.Set the data mode.
PATTern: CUSTom	<type,value>	Set the data under the data mode of CUSTOM. Type= {BIN,HEX}, value is data. BIN stands for binary data; HEX stands for hexadecimal data.
PRBS:LENGth	<value>	: = {3~32}.The PRBS length set when the data mode is PRBS.
PRBS:CUSTom	<"xa+xb",c>	Set the polynomial and seed of PRBS when the data mode is CUSTOM_PRBS, and the polynomial needs to be enclosed in double quotation marks. a is the high term coefficient of polynomial, b is

		the low term coefficient, and c is the seed.
BINVert	<state>	:= {ON,OFF}. bit flip switch.
B810:STATe	<state>	:= {ON,,OFF}.8b/10b enable switch.
B810:TYPE	<type>	:= {MINUS,PLUS}.Set the balance mode of 8b/10b, where MINUS is RD- and PLUS is RD+.
SCRAmbling: STATe	<state>	:= {ON,OFF}.Scrambling enable switch.
SCRAmbling: FORMula	<"xa+xb",c>	When scrambling is enabled, set the scrambled polynomial. Polynomials need double quotation marks. a is the high term coefficient of polynomial, b is the low term coefficient, and c is the seed.
SCRAmbling: TYPE	<type>	:= {ADD, MULTI}.Set the scrambling mode. ADD is addition and MULTI is multiplication.
SIGNal	<type>	:= {NRZ, NRZ_I, PAM4, PAM8, PAM16}. Set the encoding/modulation mode.
PAMData	<type,value>	Set the data in PAM modulation mode. Type= {PAM4, PAM8, PAM16}, value= 4 values or 8 values or 16 values, with values between 1, separated by commas, such as -1,-0.333,0.333,1.
PWM:STATe	<state>	:= {ON,OFF}.Set the PWM state.
PWM:WIDTh	<value>	Set the PWM pulse width. The unit is UI.
EDGE:STATe	<state>	:= {ON,OFF}. Set the edge switch state.
EDGE:RTIme	<value>	Sets the rise time of the edge. The unit is UI.
EDGE:FTIme	<value>	Sets the falling time of the edge. The unit is UI.

EDGE:RSHape	<type>	:= {LINEAR,FIRST_ORDER}. Sets the type of rising edge.
EDGE:FSHape	<type>	:= {LINEAR, FIRST_ORDER}.Sets the type of falling edge.
EDGE:RANGe	<type>	:= {RANGE_20_80,RANGE_10_90}. Set the type of edge.
DCD:STATe	<state>	:= {ON,OFF}. Set the switch with duty cycle distortion.
DCD:DATA	<value>	:= {0.01~0.5}. Set the duty cycle distortion value. The unit is UI.
PJITter:STATe	<state>	:= {ON,OFF}. Set the switch of periodic jitter.
PJITter:ADD		Add a period of jitter
PJITter:DEL	<value>	:= {1~4 }.Value is the ID number of periodic jitter. Deletes the specified period jitter.

PJITter:SET	ID,index,MAG, value1,PHASE ,value2,FREQ, value3	Index= {1~4}.Is the ID number of periodic jitter.
		value1= {0.001~50}, jitter range, the unit is UI;
		value2= {0~360}, phase, unit degree;
		value3= {10 KHz~500 MHz}, frequency, the unit is Hz.

RJITter:STATe	<state>	:= {ON,OFF}. Set the switch of random jitter.
RJITter:ADD		Add a random jitter.
RJITter:DEL	<value>	:= {1~4}. Value is the ID number of random jitter. Deletes a specified random jitter.
RJITter:SET	ID,index,MAG, value1,BW, value2,CF, value3	Index= {1~4}, Is the ID number of random jitter; value1={0.001~0.5}, Jitter range, the unit is UI; value2= {1 mHz~2 GHz}, bandwidth, the unit is Hz; value3= {0~12},

			Peak-to-average ratio, the unit is dB.
	NOISe:STATe	<state>	:= {ON,OFF}. Set the switch for noise superposition.
	NOISe:MAG	<value>	:= {0~1}. Value is the range of noise amplitude, and the percentage is set.
	NOISe:BW	<value>	:= {10 Hz~2 GHz}.Value is the noise bandwidth.the unit is Hz.
	NOISe:CF	<value>	:= {-100~100}.Value is the peak-to-average ratio of noise, and the unit is dB.
	EQUAlization: STATe	<state>	:= {ON,OFF}.Set the balanced switch.
	EQUAlization: TAP<value1>	<value2>	Value1= {1~5 }; value2= {-1~1}. The equilibrium coefficient is set, and the sum of the absolute value2 of the five Value1s cannot be greater than 1.
	SSC:STATe	<state>	:= {ON,OFF}. Set the switch of spread spectrum clock.
	SSC:SHAPe	<type>	:= {SINE, SQUARE, TRIANGLE}. Set the waveform type of spread spectrum clock.
	SSC:FREQ	<value>	:= {1 Hz~5 GHz}. Set the spread spectrum clock frequency.The unit is Hz.
	SSC:DEV	<value>	:= {0~1000000}. Set the deviation of spread spectrum clock. The unit is PPM.
QUERY SYNTAX	<channel>:HSS:<parameter>? <channel>:HSS:PAMData? <type> type= {PAM4, PAM8, PAM16} <channel>:HSS:PJITter:SET? <IDx> <channel>:HSS:RJITter:SET? <IDx> x= {1~4}		
EXAMPLE	Channel 1 sets the sampling rate of hss to 1 GSa/s:		

C1:HSS:SRAte 1e9

Query the sampling rate of hss in channel 1:

C1:HSS:SRAte?

Return:

"1000000000\n"

Channel 1 is set to binary data in CUSTOM data mode:

C1:HSS:PATTern:CUSTom BIN,01010011

Query data in custom data mode of channel 1:

C1:HSS:PATTern:CUSTom?

Return:

"BIN,01010011\n"

Channel 1 sets the polynomial in the data mode of CUSTOM_PRBS:

C1:HSS:PRBS:CUSTom "x5+x3",4

Querying Polynomials in Channel 1CUSTOM_PRBS Data Mode:

C1:HSS:PRBS:CUSTom?

Return:

"Formula=X5+X3+1,Seed=4\n"

Channel 1 sets data under PAM8 modulation:

C1:HSS:PAMData PAM8,-0.9,-0.6,-0.4,-0.1,0.1,0.4,0.6,0.9

Query the data of channel 1 PAM8.:

C1:HSS:PAMData? PAM8

Return:

"-0.900000;-0.600000;-0.400000;-0.100000;0.100000;0.400000;0.600000;0.900000;\n"

Channel 1 sets the periodic jitter of serial number 1:

C1:HSS:PJITter:SET ID,1,MAG,0.003,PHASE,66,FREQ,66e6

Query the periodic jitter of serial number 1 of channel 1:

C1:HSS:PJITter:SET? ID,1

Return:

"MAG:0.003000,PHASE:66.000000,FREQ:66000000.000000\n"

Channel 1 sets the balanced tap 2 coefficient:

C1:HSS:EQUAlization:TAP2 0.3

Query the coefficient of equalization tap 2 in channel 1:

C1:HSS:EQUAlization:TAP2?

Return:

"0.3\n"

3.15 IQ Command

3.15.1 IQ#:WAVeinfo?

DESCRIPTION	This command is used to query the waveform information of IQ.
COMMAND SYNTAX	IQ<index>:WAVeinfo? <index>:= {1, 2}
QUERY SYNTAX	IQ<index>:WAVeinfo? <index>:= {1, 2}
EXAMPLE	Query the waveform information of IQ1: *IQ1:WAVeinfo?* Return: *WAVE_INFO,SYMBOL_LENTH,1024,OVER_SAMPLING,4,MODULATION,2ASK,FILTER_TYPE,RootCosine,FILTER_ALPHA,0.35*

3.15.2 IQ#:CENTerfreq

DESCRIPTION	This command sets or queries the center frequency of I/Q.
COMMAND SYNTAX	<channel>:CENTerfreq <freq><unit> <channel>:= {IQ1, IQ2} <freq>:=center frequency. Please refer to the data sheet for the valid range of this parameter. <unit>:= {Hz, kHz, MHz, GHz}. The default unit is Hertz "Hz".
QUERY SYNTAX	<channel>:CENTerfreq? <channel>:={IQ1, IQ2}
RESPONSE FORMAT	< center frequency > (expressed in Hz)
EXAMPLE	Set the center frequency of the first IQ to 1kHz: *IQ1:CENTerfreq 1000Hz*

3.15.3 IQ#:SAMPlerate

DESCRIPTION	This command sets or queries the sampling rate of I/Q.
COMMAND SYNTAX	<channel>:SAMPlerate <samprate><unit> <channel>:= {IQ1, IQ2} <samprate>:= Sampling rate.Please refer to the data sheet for the valid range of this parameter.

	<unit>:= {Hz, kHz, MHz, GHz}. The default unit is Hertz "Hz".
QUERY SYNTAX	<channel>:SAMPlerate? <channel>:= {IQ1, IQ2}
EXAMPLE	Set the sampling rate of IQ1 to 100 kHz: *IQ1:SAMPlerate 100000* or: *IQ1:SAMP 100kHz*

3.15.4 IQ#:SYMBolrate

DESCRIPTION	This command is used to set the symbol rate of I/Q.
COMMAND SYNTAX	<channel>:SYMBolrate < symbolic rate ><unit> <channel>:= {IQ1, IQ2} < symbolic rate >:= symbolic rate.Please refer to the data sheet for the valid range of this parameter. <unit>:= {S/s, kS/s, MS/s}. The default unit is "S/s".
QUERY SYNTAX	<channel>:=SYMBolrate? <channel>:= {IQ1, IQ2}
EXAMPLE	Set the symbol rate of IQ1 to 1 ms/s: *IQ1:SYMB 1MS/s*

3.15.5 IQ#:AMPLitude

DESCRIPTION	This command sets the I/Q amplitude.
COMMAND SYNTAX	<channel>:AMPLitude <amp><unit> <channel>:= {IQ1, IQ2} <amp>:= Amplitude. Please refer to the data sheet for the valid range of this parameter. <unit>:= {Vrms, mVrms, dBm}.The default unit is root mean square "Vrms".
QUERY SYNTAX	<channel>:AMPLitude? <channel>:= {IQ1, IQ2}
EXAMPLE	Set the amplitude of the IQ1 to 0.2vrms: *IQ1:AMPL 0.2*

3.15.6 IQ#:IQADjustment:GAIN

DESCRIPTION	This command adjusts the ratio of I to Q while keeping IQ complex.
COMMAND SYNTAX	<channel>:IQADjustment:GAIN < gain ratio ><unit> <channel>:= {IQ1, IQ2} < gain ratio >:=Gain ratio of I and q. <unit>:= {dB}
QUERY SYNTAX	<channel>:IQADjustment:GAIN? <channel>:= {IQ1, IQ2}
EXAMPLE	Set the gain ratio of the IQ1 to 0.1dB: *IQ1:IQADjustment:GAIN 0.1*

3.15.7 IQ#:IQADjustment:IOFFset

DESCRIPTION	This command adjusts the offset of the I channel.
COMMAND SYNTAX	<channel>:IQADjustment:IOFFset <offset><unit> <channel>: ={IQ1, IQ2} <offset>:=The offset of I. <unit>:= {V, mV, uV}.The default unit is the volt "v".
QUERY SYNTAX	<channel>:IQADjustment:IOFFset? <channel>:= {IQ1, IQ2}
EXAMPLE	Set the I-channel bias of the IQ1 to 1mV: *IQ1:IQADjustment:IOFFset 1mV*

3.15.8 IQ#:IQADjustment:QOFFset

DESCRIPTION	This command adjusts the offset of the Q channel.
COMMAND SYNTAX	<channel>:IQADjustment:QOFFset <offset><unit> <channel>:={IQ1, IQ2} <offset>:= The offset of Q. <unit>:= {V, mV, uV}.The default unit is the volt "v".
QUERY SYNTAX	<channel>:IQADjustment:QOFFset? <channel>:= {IQ1, IQ2}
EXAMPLE	Set the Q-channel bias of the IQ1 to -1mV: *IQ1:IQAD:QOFF -0.001*

3.15.9 IQ#:IQADjustment:QSKew

DESCRIPTION	This command adjusts the phase angles of I and Q vectors by increasing or decreasing the phase angle of Q.
COMMAND SYNTAX	<channel>:IQADjustment:QSKew < angle > <channel>:= {IQ1, IQ2} < angle >:=angle. The unit is degrees.
QUERY SYNTAX	<channel>:IQADjustment:QSKew? <channel>:= {IQ1, IQ2}
EXAMPLE	Set the Q angle of the IQ1 to 1: *IQ1:IQADjustment:QSKew 1.0*

3.15.10 IQ#:TRIGger:SOURce

DESCRIPTION	This command sets the trigger source of I/Q.
COMMAND SYNTAX	<channel>:TRIGger:SOURce < source> <channel>:= {IQ1, IQ2} < source >:= {INT, EXTernalA, EXTernalB, MANualA, MANualB, TIMER}.
QUERY SYNTAX	<channel>:TRIGger:SOURce? <channel>:= {IQ1, IQ2}
EXAMPLE	Set the IQ1 trigger source as manual trigger A: *IQ1:TRIGger:SOURce MANA*

3.15.11 IQ#:TRIGger:SLOPe <type>

DESCRIPTION	This command sets or queries the edge of the external trigger source of I/Q.
COMMAND SYNTAX	<channel>:TRIGger:SLOPe <type> <channel>:= {IQ1, IQ2} <type>:= {RISE, FALL, BOTH}
QUERY SYNTAX	<channel>:TRIGger:SLOPe? <channel>:= {IQ1, IQ2}
EXAMPLE	Set the IQ1 external trigger source edge as the falling edge: *IQ1:TRIGger:SLOPe FALL*

	Query the edge of the IQ1 external trigger source: *IQ1:TRIGger:SLOPe?* Return: *FALL\n*

3.15.12 IQ#:MANTriger<type>

DESCRIPTION	This command is used to set the trigger when IQ is manually triggered.
COMMAND SYNTAX	<channel>:MANTriger<type> <channel>:= {IQ1, IQ2} <type>:= {A, B}
EXAMPLE	Set IQ1 to manually trigger A trigger: *IQ1:MANTrigerA*

3.15.13 IQ#:TRIGTimer <value>

DESCRIPTION	This command sets or queries the timing time when I/Q timing is triggered.
COMMAND SYNTAX	<channel>:TRIGTimer <value> <channel>:= {IQ1, IQ2} <value>:= Timing time.
QUERY SYNTAX	<channel>:TRIGTimer? <channel>:= {IQ1, IQ2}
EXAMPLE	Set the time for the IQ1 timing trigger to 0.01: *IQ1:TRIGTimer 0.01* Query the IQ1 timing trigger source time: *IQ1:TRIGTimer?* Return: *0.01\n*

3.15.14 IQ#:WAVEload:BUILtin

DESCRIPTION	This command is used to select an I/Q waveform from the list of built-in waveforms.

COMMAND SYNTAX	<channel>:WAVEload:BUILtin <wave> <channel>:= {IQ1, IQ2} <wave>:= { Waveform names in the following table }
QUERY SYNTAX	<channel>:WAVEload? <channel>:= {IQ1, IQ2}
EXAMPLE	Set the waveform of the IQ1 as the built-in waveform 2ASK: *IQ1:WAVE:BUIL "2ASK"*

2ASK	4ASK	8ASK	BPSK	4PSK
8PSK	DBPSK	4DPSK	8DPSK	8QAM
16QAM	32QAM	64QAM	128QAM	256QAM
16QAM_2	TEST_DC			

3.15.15 IQ#:WAVEload:USERstored

DESCRIPTION	This command is used to load the I/Q waveform of the storage disk.
COMMAND SYNTAX	<channel>:WAVEload:USERstored <path> <channel>:= {IQ1, IQ2} <path>:= { Waveform path from user storage (local, network storage, U disk), which needs to include file name and suffix. }
QUERY SYNTAX	<channel>:WAVEload? <channel>:= {IQ1, IQ2}
EXAMPLE1	Set the waveform of the IQ2 as the waveform wave1.arb stored locall by the user: *IQ2:WAVEload:USERstored "Local/EasyIQ_arb/DQPSK_UserIQ_1.arb"*
EXAMPLE2	Set the waveform of IQ1 as wave1.arb of the network storage path: *IQ1:WAVEload:USERstored "net_storage/wave/wave1.arb"*
EXAMPLE3	Set the waveform of IQ1 as wave1.arb of the storage path of the U disk: *IQ1:WAVEload:USERstored "U-disk0/wave/wave1.arb"*

3.15.16 IQ#:MARKer1:POS# <state>

DESCRIPTION	This command sets the marker switch state of I/Q.
COMMAND SYNTAX	<channel>:MARKer<value1>:POS<value2> <state> <channel>:= {IQ1, IQ2} <value1>:={1, 2} <value2>= Marker's serial number. <state>:= {ON, OFF}
QUERY SYNTAX	<channel>:MARKer<value1>? <channel>:= {IQ1, IQ2} <value1>:={1, 2}
EXAMPLE	Set the first data point of marker 1 to open: *IQ1:MARKer1:POS1 ON* Query the status of marker1: *IQ1:MARKer1?* Return: *"0,1,2,4"\n* # Data 0, 1, 2 and 4 data bits are marked on.

3.15.17 DACTYPe <type>

DESCRIPTION	This command sets the clock type of IQ.
COMMAND SYNTAX	DACTYPe <type> <type>:= {5G, 6G}. 5G, 6G represent 10G and 12G of DAC sample rate.
QUERY SYNTAX	DACType?
RESPONSE FORMAT	<type>
EXAMPLE	Set the DAC sampling rate type to 12G: *DACTYPe 6G*

Note: Four-channel models are all in IQ IF mode to switch to 12G DAC sampling rate.

3.15.18 IQ#:COMpensation <state>

DESCRIPTION	This command is used to set or query the status of IQ compensation.
COMMAND SYNTAX	<channel>:COMpensation <state>

	<channel>:= {IQ1, IQ2} <state>:= {ON, OFF}
QUERY SYNTAX	<channel>:COMpensation? <channel>:= { IQ1, IQ2}
EXAMPLE	Set the compensation status of IQ1 to ON: *IQ1:COMpensation ON* Query the compensation status of IQ1: *IQ1:COMpensation?* Return: *ON\n*

Note: When compensation is turned on, all parameter modifications need to be set after output is turned off.

3.15.19 IQ#:MCABle:STATe <state>

DESCRIPTION	This command is used to set (query) the cable matching switch state of IQ.
COMMAND SYNTAX	<channel>:MCABle:STATe <state> <channel>:= { IQ1, IQ2} <state>:= {ON, OFF}
QUERY SYNTAX	<channel>:MCABle:STATe? <channel>:= { IQ1, IQ2}
EXAMPLE	Set the state of IQ1' s cable matching switch to ON: *IQ1:MCABle:STATe ON* Query the cable matching switch status of IQ1: *IQ1: MCABle:STATe?* Return *ON\n*

3.15.20 IQ#:MCABle:IFILe <path>

DESCRIPTION	This command is used to set (query) the transfer function of I-way cable matching of IQ.
COMMAND SYNTAX	<channel>:MCABle:IFILe <path> <channel>:= { IQ1, IQ2} <path>:= The path of the transfer function, including the file name,

	has the suffix. stf2. It can be a local path of the device, a network storage path or a USB flash drive path.
QUERY SYNTAX	<channel>: MCABle:IFILe? <channel>:= {IQ1, IQ2}
EXAMPLE	Set the transfer function of I-way cable matching of IQ1 to test.stf2 under the local path: *IQ1:MCABle:IFILe "Local/test.stf2"* Set the transfer function of IQ1' s I-way cable matching to test.stf2 under the U disk path: *IQ1:MCABle:IFILe "U-disk0/test.stf2"* Set the transfer function of I-way cable matching of IQ1 as test.stf2 under the network storage path: *IQ1:MCABle:IFILe "net_storage/test.stf2"* Query the transfer function of I-channel cable matching of IQ1: *IQ1:MCABle:IFILe?* Return: *Local/test.stf2\n*

3.15.21 IQ#:MCABle:QFILe <path>

DESCRIPTION	This command is used to set (query) the transfer function of Q-way cable matching of IQ.
COMMAND SYNTAX	<channel>:MCABle:QFILe <path> <channel>:= {IQ1, IQ2} <path>:= The path of the transfer function, including the file name, has the suffix. stf2. It can be a local path of the device, a network storage path or a USB flash drive path.
QUERY SYNTAX	<channel>: MCABle:QFILe? <channel>:= {IQ1, IQ2}
EXAMPLE	Set the transfer function of Q-way cable matching of IQ1 to test.stf2 under the local path: *IQ1:MCABle:QFILe "Local/test.stf2"* Set the transfer function of IQ1' s Q-way cable matching to test.stf2 under the U disk path: *IQ1:MCABle:QFILe "U-disk0/test.stf2"*

	Set the transfer function of Q-way cable matching of IQ1 as test.stf2 under the network storage path: *IQ1:MCABle:QFILe "net_storage/test.stf2"* Query the transfer function of Q-channel cable matching of IQ1: *IQ1:MCABle:QFILe?* Return: *Local/test.stf2\n*

3.15.22 IQ#:CREAt:CTFunction <path1>,<path2>

DESCRIPTION	This command is used to set IQ cable matching and generate transfer function stf2 file according to s2p file of cable S parameters.
COMMAND SYNTAX	<channel>:CREAt:CTFunction <path1>,<path2> <channel>:= {IQ1, IQ2} <path1>:= The path of s2p file of cable S parameter, including file name and suffix. s2p. <path2>:= Generate the saving path of the transfer function stf2 file, including the file name and the suffix. stf2. Note: The path can be local to the device, network storage or USB flash drive.
EXAMPLE	Set IQ to generate transfer function according to s2p file of U disk path and save it in local path: *IQ1:CREAt:CTFunction "U-disk0/test.s2p","Local/test.stf2"* Set IQ to generate transfer function according to s2p file of network storage path and save it in local path: *IQ1:CREAt:CTFunction "net_storage/test.s2p","Local/test.stf2"*

3.15.23 IQ#:CREAt:ATFunction <path1>,<path2>,<path3>

DESCRIPTION	This command is used to set IQ cable matching to generate transfer function stf2 file according to s2p file of cable S parameters and s1p file of port reflection coefficient.
COMMAND SYNTAX	<channel>:CREAt:CTFunction <path1>,<path2>,<path3> <channel>:= {IQ1, IQ2} <path1>:= The path of s2p file of cable S parameter, including file

	name and suffix. s2p. <path2>:= The path of s1p file of port reflection coefficient, including file name and suffix .s1p. <path3>:= Generate the saving path of the transfer function stf2 file, including the file name and the suffix. stf2. Note: The path can be local to the device, network storage or USB flash drive.
EXAMPLE	Set IQ to generate a transfer function according to s2p and s1p files of the U disk path and save it in the local path: *IQ1:CREAt:CTFunction "U-disk0/test.s2p" , U-disk0/test.s1p","Local/test.stf2"* Set IQ to generate transfer function according to s2p and s1p files of network storage path and save it in local path: *IQ1:CREAt:CTFunction "net_storage/test.s2p", "net_storage/test.s1p","Local/test.stf2"*

3.15.24 IQ#:MODE <type>

DESCRIPTION	This command is used to set the switch between IQ and IQ Sequence and query the current mode.
COMMAND SYNTAX	<channel>:MODE <type> <channel>:= {IQ1, IQ2} <type>:= {NORMAL, AWG}
QUERY SYNTAX	<channel>:MODE? <channel>:= { IQ1, IQ2}
EXAMPLE	Set IQ1' s mode to NORMAL: *IQ1:MODE NORMAL* Query IQ1 mode: *IQ1:MODE?* Return: *IQ_NORMAL \n*

3.16 IQ Sequence

3.16.1 <channel>:AWG:STATE <state>

DESCRIPTION	This command is used to set or query the running status of IQ Sequence.
COMMAND SYNTAX	<channel>:AWG:STATe <state> <channel>:= {C1, C2, C3, C4} <state>:= {STOP, RUN}
QUERY SYNTAX	<channel>:AWG:STATe? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the RUNning state of channel 1 to run: *C1:AWG:STATE RUN* Query the running status of channel 1: *C1:AWG:STATE?* Return: *RUN\n*

Note: All parameters need to be modified when the running status is STOP.

3.16.2 <channel>:AWG:DEFAult

DESCRIPTION	This command is used to set the default settings of IQ Sequence.
COMMAND SYNTAX	<channel>:AWG:DEFAult <channel>:= {C1, C2, C3, C4}

3.16.3 <channel>:AWG:SRATE <value>

DESCRIPTION	This command is used to set or query the sampling rate of IQ Sequence.
COMMAND SYNTAX	<channel>:AWG:SRATE <value> <channel>:= {C1, C2, C3, C4} <value>:= Sampling rate
QUERY SYNTAX	<channel>:AWG:SRATe? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the sampling rate of channel 1 to 200M m: *C1:AWG:SRATE 2e8*

	Query the sampling rate of channel 1. *C1:AWG:SRATE?* Return: *200000000\n*

3.16.4 <channel>:AWG:SCALe <value>

DESCRIPTION	This command is used to set or query the output amplitude ratio of IQ Sequence.
COMMAND SYNTAX	<channel>:AWG:SCALe <value> <channel>:= {C1, C2, C3, C4} <value>:={0, 1}
QUERY SYNTAX	<channel>: AWG:SCALe? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the amplitude ratio of channel 1 to 88%: *C1:AWG:SCALe 0.88* Query the amplitude ratio of channel 1: *C1:AWG:SCALe?* Return: *0.88\n*

3.16.5 <channel>:AWG:OFFset <value>

DESCRIPTION	This command is used to set (query) the offset of IQ Sequence.
COMMAND SYNTAX	<channel>:AWG:OFFset <value> <channel>:= {C1, C2, C3, C4} <value>:= { The range of values is different under different output modes }See the data sheet for the parameter range.
QUERY SYNTAX	<channel>:AWG:OFFset? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the offset of channel 1 to 0.01: *C1:AWG:OFFset 0.01* Query the offset of channel 1: *C1:AWG:OFFset?* Return: *0.01\n*

3.16.6 <channel>:AWG:DIFFset <value>

DESCRIPTION	This command is used to set (query) the diff offset of IQ Sequence.
COMMAND SYNTAX	<channel>:AWG:DIFFset <value> <channel>:= {C1, C2, C3, C4} <value>:= { The range of values is different under different output modes}See the data sheet for the parameter range.
QUERY SYNTAX	<channel>:AWG:DIFFset? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the diff offset of channel 1 to 0.01: *C1:AWG:DIFFset 0.01* Query the diff offset of channel 1: *C1:AWG:DIFFset?* Return: *0.01\n*

3.16.7 <channel>:AWG:TRIGger:TIMer <value>

DESCRIPTION	This command is used to set (query) the timer time of IQ Sequence.
COMMAND SYNTAX	<channel>:AWG:TRIGger:TIMer <value> <channel>:= {C1, C2, C3, C4}
QUERY SYNTAX	<channel>:AWG:TRIGger:TIMer? <channel>:= {C1, C2, C3, C4}
EXAMPLE	The time for setting the trigger timing of channel 1 is 1 second: *C1:AWG:TRIGger:TIMer 1* Query the time when channel 1 triggers the timing: *C1:AWG:TRIGger:TIMer?* Return: *1\n*

3.16.8 <channel>:AWG:TRIGAger:SLOPe <type >

DESCRIPTION	This command is used to set (query) the edge trigger mode of EXTA.
COMMAND SYNTAX	<channel>:AWG:TRIGAger:SLOPe <type> <channel>:= {C1, C2, C3, C4} <type>:={ RISe, FALL, BOTH}

QUERY SYNTAX	<channel>:AWG:TRIGAger:SLOPe? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the trigger edge of EXTA of channel 1 as RISe: *C1:AWG:TRIGAger:SLOPe RISe* Query the trigger edge of EXTA of channel 1: *C1:AWG:TRIGAger:SLOPe?* Return: *RISe\n*

3.16.9 <channel>:AWG:TRIGBger:SLOPe <type >

DESCRIPTION	This command is used to set (query) the edge trigger mode of EXTB.
COMMAND SYNTAX	<channel>:AWG:TRIGBger:SLOPe <type> <channel>:= {C1, C2, C3, C4} <type>:={ RISe, FALL, BOTH}
QUERY SYNTAX	<channel>: AWG: TRIGBger:SLOPe? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the trigger edge of EXTA of channel 1 as RISe: *C1:AWG:TRIGBger:SLOPe RISe* Query the trigger edge of EXTA of channel 1 : *C1:AWG:TRIGBger:SLOPe?* Return : *RISe\n*

3.16.10 <channel>:AWG:TRIGger:DELAy <value>

DESCRIPTION	This command is used to set (query) the trigger delay of IQ Sequence.
COMMAND SYNTAX	<channel>:AWG:TRIGger:DELAy <value> <channel>:= {C1, C2, C3, C4}
QUERY SYNTAX	<channel>:AWG:TRIGger:DELAy? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the trigger delay of channel 1 to 20 ns: *C1:AWG:TRIGger:DELAy 2e-08* Query the trigger mode of channel 1 trigger delay:

	C1:AWG:TRIGger:DELAy? Return: *2e-08\n*

3.16.11 <channel>:AWG:HOLDtype <type>

DESCRIPTION	This command is used to set (query) the type of AWG idle hold level.
COMMAND SYNTAX	<channel>:AWG:HOLDtype <type> <channel>:= {C1, C2, C3, C4} <type>:= {MID, START, END, USER}
QUERY SYNTAX	<channel>:AWG:INTPtype? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the hold level type of channel 1 to END: *C1:AWG:HOLDtype END* Query the idle hold level of channel 1: *C1:AWG:HOLDtype?* Return: *END\n*

3.16.12 <channel>:AWG:USRHOLD <value>

DESCRIPTION	This command is used to set (query) the level under IQ Sequence custom idle hold level type.
COMMAND SYNTAX	<channel>:AWG:USRHOLD <value> <channel>:= {C1, C2, C3, C4} <value>:= Customize the level value of the hold level.
QUERY SYNTAX	<channel>:AWG:USRHOLD? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the custom hold level of channel 1 to 0.3: *C1:AWG:USRHOLD 0.3* Query the custom hold level of channel 1: *C1:AWG:USRHOLD?* Return: *0.3\n*

3.16.13 <channel>:AWG:DYNA:TYPE <type>

DESCRIPTION	This command is used to set (query) the type of IQ Sequence dynamic mode.
COMMAND SYNTAX	<channel>:AWG:DYNA:TYPE <type> <channel>:= {C1, C2, C3, C4} <type>:= {JUMP_TYPE, CONTROL_TYPE}
QUERY SYNTAX	<channel>:AWG:DYNA:TYPE? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the dynamic jump type of channel 1 to CONTROL_TYPE: *C1:AWG:DYNA:TYPE CONTROL_TYPE* Query the dynamic jump type of channel 1: *C1:AWG:DYNA:TYPE?* Return: *CONTROL_TYPE\n*

3.16.14 <channel>:AWG:GATELevel <type>

DESCRIPTION	This command is used to set (query) the gating effective level of IQ Sequence.
COMMAND SYNTAX	<channel>:AWG:GATELevel <type> <channel>:= {C1, C2, C3, C4} <type>:= {NEGATIVE, POSITIVE}
QUERY SYNTAX	<channel>: AWG:GATELevel? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the gating active level of channel 1 to positive: *C1:AWG:GATELevel POSITIVE* Query the gating effective level of channel 1: *C1:AWG:GATELevel?* Return: *POSITIVE\n*

3.16.15 <channel>:AWG:SCENario:TIMer <value>

DESCRIPTION	This command is used to set (query) the Scenario timing time of IQ Sequence.

COMMAND SYNTAX	<channel>:AWG:SCENario:TIMer <value> <channel>:= {C1, C2, C3, C4} <value>:= Timing time
QUERY SYNTAX	<channel>:AWG:SCENario:TIMer? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the Scenario timing time of channel 1 to 0.01: *C1:AWG:SCENario:TIMer 0.01* Query Scenario timing time of channel 1: *C1:AWG:SCENario:TIMer?* Return: *0.01\n*

3.16.16 <channel>:AWG:SEQUence:TIMer <value>

DESCRIPTION	This command is used to set (query) the sequence timing time of IQ Sequence.
COMMAND SYNTAX	<channel>:AWG:SEQUence:TIMer <value> <channel>:= {C1, C2, C3, C4} <value>:= timing time
QUERY SYNTAX	<channel>:AWG:SEQUence:TIMer? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the sequence timing time of channel 1 to 0.01: *C1:AWG:SEQUence:TIMer 0.01* Query sequence timing time of channel 1 : *C1:AWG:SEQUence:TIMer?* Return: *0.01\n*

3.16.17 <channel>:AWG:SEGMent:TIMer <value>

DESCRIPTION	This command is used to set (query) the segment timing time of IQ Sequence.
COMMAND SYNTAX	<channel>:AWG:SEGMent:TIMer <value> <channel>:= { C1,C2,C3,C4} <value>:= timing time
QUERY SYNTAX	<channel>:AWG:SEGMent:TIMer? <channel>:= { C1,C2,C3,C4}

EXAMPLE	Set the segment timing time of channel 1 to 0.01: *C1:AWG:SEGMent:TIMer 0.01* Query segment timing time of channel 1: *C1:AWG:SEGMent:TIMer?* Return: *0.01\n*

3.16.18 IQ#:COMpensation <state>

DESCRIPTION	This command is used to set or query the status of IQ compensation.
COMMAND SYNTAX	<channel>:COMpensation <state> <channel>:= {IQ1, IQ2} <state>:= {ON, OFF}
QUERY SYNTAX	<channel>:COMpensation? <channel>:= { IQ1, IQ2}
EXAMPLE	Set the compensation status of IQ1 to ON: *IQ1:COMpensation ON* Query the compensation status of IQ1: *IQ1:COMpensation?* Return: *ON\n*

3.16.19 IQ#:MCABle:STATe <state>

DESCRIPTION	This command is used to set (query) the cable matching switch state of IQ.
COMMAND SYNTAX	<channel>:MCABle:STATe <state> <channel>:= { IQ1, IQ2} <state>:= {ON, OFF}
QUERY SYNTAX	<channel>:MCABle:STATe? <channel>:= { IQ1, IQ2}
EXAMPLE	Set the state of IQ1' s cable matching switch to ON: *IQ1:MCABle:STATe ON* Query the cable matching switch status of IQ1: *IQ1: MCABle:STATe?* Return *ON\n*

3.16.20 IQ#:MCABle:IFILe <path>

DESCRIPTION	This command is used to set (query) the transfer function of I-way cable matching of IQ.
COMMAND SYNTAX	<channel>:MCABle:IFILe <path> <channel>:= {IQ1, IQ2} <path>:= The path of the transfer function, including the file name, has the suffix. stf2. It can be a local path of the device, a network storage path or a USB flash drive path.
QUERY SYNTAX	<channel>: MCABle:IFILe? <channel>:= {IQ1, IQ2}
EXAMPLE	Set the transfer function of I-way cable matching of IQ1 to test.stf2 under the local path: *IQ1:MCABle:IFILe "Local/test.stf2"* Set the transfer function of IQ1' s I-way cable matching to test.stf2 under the U disk path: *IQ1:MCABle:IFILe "U-disk0/test.stf2"* Set the transfer function of I-way cable matching of IQ1 as test.stf2 under the network storage path: *IQ1:MCABle:IFILe "net_storage/test.stf2"* Query the transfer function of I-channel cable matching of IQ1: *IQ1:MCABle:IFILe?* Return: *Local/test.stf2\n*

3.16.21 IQ#:MCABle:QFILe <path>

DESCRIPTION	This command is used to set (query) the transfer function of Q-way cable matching of IQ.
COMMAND SYNTAX	<channel>:MCABle:QFILe <path> <channel>:= {IQ1, IQ2} <path>:= The path of the transfer function, including the file name, has the suffix. stf2. It can be a local path of the device, a network storage path or a USB flash drive path.
QUERY SYNTAX	<channel>: MCABle:QFILe? <channel>:= {IQ1, IQ2}
EXAMPLE	Set the transfer function of Q-way cable matching of IQ1 to test.stf2 under the local path:

	IQ1:MCABle:QFILe "Local/test.stf2" Set the transfer function of IQ1' s Q-way cable matching to test.stf2 under the U disk path: *IQ1:MCABle:QFILe "U-disk0/test.stf2"* Set the transfer function of Q-way cable matching of IQ1 as test.stf2 under the network storage path: *IQ1:MCABle:QFILe "net_storage/test.stf2"* Query the transfer function of Q-channel cable matching of IQ1: *IQ1:MCABle:QFILe?* Return: *Local/test.stf2\n*

3.16.22 IQ#:CREAt:CTFunction <path1>,<path2>

DESCRIPTION	This command is used to set IQ cable matching and generate transfer function stf2 file according to s2p file of cable S parameters.
COMMAND SYNTAX	<channel>:CREAt:CTFunction <path1>,<path2> <channel>:= {IQ1, IQ2} <path1>:= The path of s2p file of cable S parameter, including file name and suffix. s2p. <path2>:= Generate the saving path of the transfer function stf2 file, including the file name and the suffix. stf2. Note: The path can be local to the device, network storage or USB flash drive.
EXAMPLE	Set IQ to generate transfer function according to s2p file of U disk path and save it in local path: *IQ1:CREAt:CTFunction "U-disk0/test.s2p","Local/test.stf2"* Set IQ to generate transfer function according to s2p file of network storage path and save it in local path: *IQ1:CREAt:CTFunction "net_storage/test.s2p","Local/test.stf2"*

3.16.23 IQ#:CREAt:ATFunction <path1>,<path2>,<path3>

DESCRIPTION	This command is used to set IQ cable matching to generate transfer function stf2 file according to s2p file of cable S parameters and s1p file of port reflection coefficient.
COMMAND SYNTAX	<channel>:CREAt:CTFunction <path1>,<path2>,<path3>

	<channel>:= { IQ1, IQ2} <path1>:= The path of s2p file of cable S parameter, including file name and suffix. s2p. <path2>:= The path of s1p file of port reflection coefficient, including file name and suffix .s1p. <path3>:= Generate the saving path of the transfer function stf2 file, including the file name and the suffix. stf2. Note: The path can be local to the device, network storage or USB flash drive.
EXAMPLE	Set IQ to generate a transfer function according to s2p and s1p files of the U disk path and save it in the local path: *IQ1:CREAt:CTFunction "U-disk0/test.s2p" , U-disk0/test.s1p","Local/test.stf2"* Set IQ to generate transfer function according to s2p and s1p files of network storage path and save it in local path: *IQ1:CREAt:CTFunction "net_storage/test.s2p", "net_storage/test.s1p","Local/test.stf2"*

3.16.24 <channel>:AWG:RMODe <type>

DESCRIPTION	This command is used to set (query) the operation mode of IQ Sequence.
COMMAND SYNTAX	<channel>:AWG:RMODe <type> <channel>:= {C1,C2,C3,C4} <type>:= {CONT, TCON, GATE_EXTA, GATE_EXTB, ADV}
QUERY SYNTAX	<channel>: AWG:RMODe? <channel>:= {C1,C2,C3,C4}
EXAMPLE	Set the operation mode of channel 1 to advanced: *C1:AWG:RMODe ADV* Query the operation mode of channel 1: *C1:AWG:RMODe?* Return: *ADV\n*

3.16.25 <channel>:AWG:WMODe <type>

DESCRIPTION	This command is used to set (query) the layer of IQ Sequence.
COMMAND SYNTAX	<channel>:AWG:WMODe <type> <channel>:= {C1,C2,C3,C4} <type>:= { SINGLE_LAYER, MULTILAYER }
QUERY SYNTAX	<channel>:AWG:WMODe? <channel>:= {C1,C2,C3,C4}
EXAMPLE	Set the layer of channel 1 to MULTILAYER: *C1:AWG:WMODe MULTILAYER* Query the layer of channel 1: *C1:AWG:WMODe?* Return: *MULTILAYER\n*

3.16.26 <channel>:AWG:DYNA:TABLe:ADD PATT,<value1>,SCEN,<value2>, SEQU,<value3>,SEGM,<value4>

DESCRIPTION	This command is used to add items to the dynamic jump table of IQ Sequence.
COMMAND SYNTAX	<channel>:AWG:DYNA:TABLe:ADD PATT,<value1>,SCEN,<value2>,SEQU,<value3>,SEGM,<value4> <channel>:= {C1, C2, C3, C4} <value1>:= [0, 255] <value2>:= [1, The maximum number of scenario currently set] <value3>:= [1, The maximum number of sequence currently set] <value4>:= [1, The maximum number of segment currently set]
QUERY SYNTAX	<channel>: AWG:DYNA:TABLe? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Add an item to the dynamic jump table of channel 1: *C1:AWG:DYNA:TABLe:ADD PATT,99,SCEN,1,SEQU,1,SEGM,1* Query the dynamic jump table of channel 1: *C1:AWG:DYNA:TABLe?* Return: *pattern:99,go_secn:0,go_sequ:0,go_segm:0\n"\n*

3.16.27 <channel>:AWG:DYNA:TABLe:DELEte <value>

DESCRIPTION	This command is used to delete items in the dynamic jump table of IQ Sequence.
COMMAND SYNTAX	<channel>:AWG:DYNA:TABLe:DELEte <value> <channel>:= {C1, C2, C3, C4} <value>:= [0, 255]
EXAMPLE	Set the dynamic jump table of channel 1 to delete one item: *C1:AWG:DYNA:TABLe:DELEte 0*

3.16.28 <channel>:AWG:DYNA:TABLe:CLEAR

DESCRIPTION	This command is used for clearing the dynamic jump table of IQ Sequence.
COMMAND SYNTAX	<channel>:AWG:DYNA:TABLe:CLEAR <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the dynamic jump table of channel 1 to empty: *C1:AWG:DYNA:TABLe:CLEAR*

3.16.29 <channel>:AWG:DYNA:TABLe?

DESCRIPTION	This command is used to query the dynamic jump table of IQ Sequence.
QUERY SYNTAX	<channel>:AWG:DYNA:TABLe? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Query the dynamic jump table of channel 1: *C1:AWG:DYNA:TABLe?* Return: *pattern:99,go_secn:0,go_sequ:0,go_segm:0\n"\n*

3.16.30 <channel>:AWG:TRIGger:SOURce <type>

DESCRIPTION	This command is used to set (query) the trigger mode in IQ Sequence trigger mode.
COMMAND SYNTAX	<channel>:AWG:TRIGger:SOURce <type>

	<channel>:= {C1, C2, C3, C4} <type>:={MANA, MANB, TIMe, EXTA, EXTB}
QUERY SYNTAX	<channel>:AWG:TRIGger:SOURce? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the trigger mode of channel 1 to manual trigger B: *C1:AWG:TRIGger:SOURce MANB* Query the trigger mode of channel 1 trigger mode: *C1:AWG:TRIGger:SOURce?* Return: *MANB\n*

3.16.31 <channel>:AWG:TRIGgerA

DESCRIPTION	This command is used to trigger the manual trigger button A of IQ Sequence once.
COMMAND SYNTAX	<channel>:AWG:TRIGgerA <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set IQ Sequence manual trigger a once: *C1:AWG:TRIGgerA*

3.16.32 <channel>:AWG:TRIGgerB

DESCRIPTION	This command is used to trigger the manual trigger button B of IQ Sequence once.
COMMAND SYNTAX	<channel>:AWG:TRIGgerB <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set IQ Sequence manual trigger a once: *C1:AWG:TRIGgerB*

3.16.33 <channel>:AWG:SAVE PATH,<path>,SWFS,<swfs>

DESCRIPTION	This command is used to save the IQ Sequence configuration and waveform data of the channel.
COMMAND SYNTAX	<channel>:AWG:SAVE PATH,<path>,SWFS,<swfs> <channel>:= {C1, C2, C3, C4} <path>:= The saved path and file name, with the suffix ".iqx".

	<swfs>:= {"TRUE", "FALSE"}, Used to determine whether to save the waveform data.
EXAMPLE	Channel 3 stores AWG configuration and waveform data. : *C3:AWG:SAVE PATH,"Local/aaa.iqx",SWFS,"TRUE"*

3.16.34 <channel>:AWG:LOAD PATH,<path>

DESCRIPTION	This command is used to load the IQ Sequence configuration and waveform data of the channel.
COMMAND SYNTAX	<channel>:AWG:SAVE PATH,<path>,SWFS,<swfs> <channel>:= {C1, C2, C3, C4} <path>:= The path and file name of the load, and the file name should be suffixed with ".iqx".
EXAMPLE	Channel 3 loading configuration and waveform: *C3:AWG:LOAD "Local/aaa.iqx"*

3.16.35 <channel>:AWG:SCENario:CLEAR

DESCRIPTION	This command is used to clear all segments of scenario of IQ Sequence designated channel.
COMMAND SYNTAX	<channel>:AWG:SCENario:CLEAR <channel>:= {C1, C2, C3, C4}
EXAMPLE	Channel 3 clears all Scenario segments: *C3:AWG:SCENario:CLEAR*

3.16.36 <channel>:AWG:SCENario:COUNt?

DESCRIPTION	This command is used to query the channel scenario number of IQ Sequence.
QUERY SYNTAX	<channel>:AWG:SCENario:COUNt? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Query the number of Scenario segments of channel 3: *C3:AWG:SCENario:COUNt?* Return: *3\n*

3.16.37 <channel>:AWG:SCENario:INSErt <pos>

DESCRIPTION	This command is used to insert a Scenario before IQ Sequence specifies the segment.
COMMAND SYNTAX	<channel>:AWG:SCENario:INSErt <pos> <channel>:= {C1, C2, C3, C4} <pos>:= [1, The maximum number of scenario segments currently set]
EXAMPLE	Insert a Scenario before the first Scenario of channel 3: *C3:AWG:SCENario:INSErt 2*

3.16.38 <channel>:AWG:SCENario:DELEte <pos>

DESCRIPTION	This command is used for IQ Sequence to delete a specified Scenario segment.
COMMAND SYNTAX	<channel>:AWG:SCENario:DELEte <pos> <channel>:= {C1, C2, C3, C4} <pos>:= [1, The maximum number of scenario segments currently set]
EXAMPLE	Delete the second Scenario of channel 3: *C3:AWG:SCENario:DELEte 2*

3.16.39 <channel>:AWG:SCENario:MULTIDelete <pos1, pos2, pos3...>

DESCRIPTION	This command is used for IQ Sequence to delete multiple specified Scenario segments.
COMMAND SYNTAX	<channel>:AWG:SCENario:MULTIDelete <pos1, pos2, pos3...> <channel>:= {C1, C2, C3, C4} <pos1, pos2, pos3...>:= [1, The maximum number of scenario segments currently set]
EXAMPLE	Delete the 2nd, 3rd and 6th Scenario of Channel 3: *C3:AWG:SCENario:MULTIDelete 2,3,6*

3.16.40 <channel>:AWG:SCENario<x>:LOOP <value>

DESCRIPTION	This command is used to set or query the number of cycles of the specified Scenario in IQ Sequence.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:LOOP<value> <x>:= scenario number <channel>:= {C1, C2, C3, C4} <value>:= integer
QUERY SYNTAX	<channel>:AWG:SCENario<x>:LOOP? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the loop of the first Scenario of channel 1 to 5: *C1:AWG:SCENario1:LOOP 5* Query the loop of the first Scenario in channel 1: *C1:AWG:SCENario1:LOOP?* Return: *5\n*

3.16.41 <channel>:AWG:SCENario:STARTNumb <value>

DESCRIPTION	This command is used to set the first Scenario to be played.
COMMAND SYNTAX	<channel>:AWG:SCENario:STARTNumb <value> <channel>:= {C1, C2, C3, C4} <value>:= integer
QUERY SYNTAX	<channel>:AWG:SCENario:STARTNumb? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the Scenario played first in channel 1 as the second: *C1:AWG:SCENario:STARTNumb 2* Query the Scenario segment number played first in channel 1: *C1:AWG:SCENario:STARTNumb?* Return: *2\n*

3.16.42 <channel>:AWG:SCENario<x>:WAITEvent <type>

DESCRIPTION	This command is used to set or query the wait event of Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:WAITEvent <type> <x>:= scenario number <channel>:= {C1, C2, C3, C4} <type>:= {AUTO, MANA, MANB, EXTA, EXTB, TIMe}
QUERY SYNTAX	<channel>: AWG:SCENario<x>:WAITEvent? <x>:= scenario number
EXAMPLE	Set the first Scenario waiting event of channel 1 as manual trigger A: *C1:AWG:SCENario1:WAITEvent MANA* Query the wait event of the first Scenario in channel 1: *C1:AWG:SCENario1:WAITEvent?* Return: *MANA\n*

3.16.43 <channel>:AWG:SCENario<x>:GOTO <value>

DESCRIPTION	This command is used to set or query the next Scenario after the specified scenario is played.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:GOTO <value> <x>:= scenario number <channel>:= {C1, C2, C3, C4} <value>:= [1,The maximum number of scenario segments currently set]
QUERY SYNTAX	<channel>:AWG:SCENario<x>:GOTO? <x>:= scenario number
EXAMPLE	Set the next Scenario to be played after the first scenario is played in channel 1 as the third scenario: *C1:AWG:SCENario1:GOTO 3* Query the next Scenario to be played after the first scenario is played in channel 1: *C1:AWG:SCENario1:GOTO?* Return: *3\n*

3.16.44 <channel>:AWG:SCENario<x>:PLAYBack <type>

DESCRIPTION	This command is used to set or query the playback mode of the specified Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:PLAYBack <type> <x>:= scenario number <channel>:= {C1, C2, C3, C4} <type>:= { AUTO, SINGLE, CONDITIONAL}
QUERY SYNTAX	<channel>:AWG:SCENario<x>:PLAYBack? <x>:= scenario number
EXAMPLE	Set the playback mode of the first Scenario in channel 1 to single: *C1:AWG:SCENario1:PLAYBack SINGLE* Query the next Scenario in the first scenario of Channel 1: *C1:AWG:SCENario1:PLAYBack?* Return: *SINGLE\n*

3.16.45 <channel>:AWG:SCENario<x>:OUTEvent <type>

DESCRIPTION	This command is used to set or query the jump event of the specified Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:OUTEvent <type> <x>:= scenario number <channel>:= {C1, C2, C3, C4} <type>:= {MANA, MANB, EXTA, EXTB, TIMe}
QUERY SYNTAX	<channel>:AWG:SCENario<x>:OUTEvent? <x>:= scenario number
EXAMPLE	Set the jumping event of the first Scenario in channel 1 as the Scenario timer: *C1:AWG:SCENario1:OUTEvent TIMe* Query the jump event of the first Scenario in channel 1: *C1:AWG:SCENario1:OUTEven?* Return: *TIMe\n*

Note: It is effective when the playback mode is conditional jump.

3.16.46 <channel>:AWG:SCENario<x>:SEQUence:STARTNumb <value>

DESCRIPTION	This command is used to set or query the starting sequence number of the specified Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence:STARTNumb <value> <x>:= scenario number <channel>:= {C1, C2, C3, C4} <value>:= [1, The maximum sequence number set by the current scenario]
QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence:STARTNumb? <x>:= scenario number
EXAMPLE	Set the first sequence number of the first Scenario of channel 1 to 1 : *C1:AWG:SCENario1:SEQUence:STARTNumb 1* Query the starting sequence number of the first Scenario in channel 1 : *C1:AWG:SCENario1:SEQUence:STARTNumb?* Return : *1\n*

3.16.47 <channel>:AWG:SCENario<x>:SAVE PATH,<path>, SWFS,<swfs>

DESCRIPTION	This command is used to save the specified Scenario configuration and waveform data.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SAVE ATH,<path>,SWFS,<swfs> <channel>:= {C1, C2, C3, C4} <x>:= scenario number <path>:= Save the path and file name, the file name suffix is. iqscen. <swfs>:= {"TRUE", "FALSE"}, to determine whether to save the waveform data.
EXAMPLE	Channel 3 saves the configuration and waveform data of the first SCENario : *C3:AWG:SCENario1:SAVE PATH,"Local/aaa.iqscen",SWFS,"TRUE"*

3.16.48 <channel>:AWG:SCENario:LOAD,<path>

DESCRIPTION	This command is used to load Scenario configuration and waveform data.

COMMAND SYNTAX	<channel>:AWG:SCENario:LOAD <path> <channel>:= {C1, C2, C3, C4} <path>:= The path and file name of the load, and the file name suffix is. iqscen.
EXAMPLE	Channel 3 loads scenario configuration file : *C3:AWG:SCENario:LOAD "Local/aaa.iqscen"*

3.16.49 <channel>:AWG:SCENario<x>:SEQUence:CLEAR

DESCRIPTION	This command is used to empty the list of sequence of the specified Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence:CLEAR <x>:= scenario number <channel>:= {C1, C2, C3, C4}
EXAMPLE	Clear all sequence of the first Scenario in channel 1 : *C1:AWG:SCENario1:SEQUence:CLEAR*

3.16.50 <channel>:AWG:SCENario<x>:SEQUence:COUNt?

DESCRIPTION	This command is used to query the sequence number of the specified Scenario.
QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence:COUNt? <x>:= scenario number <channel>:= {C1, C2, C3, C4}
EXAMPLE	Query the sequence number of the first Scenario in channel 1 : *C1:AWG:SCENario1:SEQUence:COUNT?* Return *3\n*

3.16.51 <channel>:AWG:SCENario<x>:SEQUence:INSErt <pos>

DESCRIPTION	This command is used to insert a sequence in the specified Scenario to achieve the specified order.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence:INSErt <pos> <x>:= scenario number

	<channel>:= {C1, C2, C3, C4} <pos>:= [0, Number of sequence set] , Inserts the tag number of the sequence.
EXAMPLE	Insert a sequence in the first Scenario of channel 1 into the second: *C1:AWG:SCENario1:SEQUence:INSErt 2*

3.16.52 <channel>:AWG:SCENario<x>:SEQUence:DELEte <pos>

DESCRIPTION	This command is used to delete the specified sequence in the specified Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>: SEQUence:DELEte <pos> <x>:=scenario number <channel>:= {C1, C2, C3, C4} <pos>:= [0, Number of sequence set]
EXAMPLE	Delete the second sequence in the first Scenario of channel 1: *C1:AWG:SCENario1:SEQUence:DELEte 2*

3.16.53 <channel>:AWG:SCENario<x>:SEQUence:MULTIDelete <pos1,pos2,pos3...>

DESCRIPTION	This command is used for IQ Sequence to delete multiple sequence specified in the specified Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:MULTIDelete <pos1, pos2, pos3...> <x>:= scenario number <channel>:= {C1, C2, C3, C4} <pos1, pos2, pos3...>:= [1, Number of sequence set]Used to determine the sequence number that needs to be deleted.
EXAMPLE	Delete the 2nd, 3rd and 6th sequence of the 2nd Scenario of channel 3: *C3:AWG:SCENario2:SEQUence:MULTIDelete 2,3,6*

3.16.54 <channel>:AWG:SCENario<x>:SEQUence<y>:LOOP <value>

DESCRIPTION	This command is used to set or query the number of cycles of sequence in Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:LOOP <value> <x>:= scenario number

	<y>:= sequence number <channel>:= {C1, C2, C3, C4} <value>:= integer
QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:LOOP? <x>:=scenario number <y>:= sequence number <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the first sequence cycle number of the first Scenario of channel 1 to 3: *C1:AWG:SCENario1:SEQUence1:LOOP 3* Query the first sequence cycle number of the first Scenario in channel 1: *C1:AWG:SCENario1:SEQUence1:LOOP?* Return: *3\n*

3.16.55 <channel>:AWG:SCENario<x>:SEQUence<y>: WAITEvent <type>

DESCRIPTION	This command is used to set or query the wait event of sequence in Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>: SEQUence<y>: WAITEvent <type> <x>:= scenario number <y>:= sequence number <channel>:= {C1, C2, C3, C4} <type>:= {AUTO, MANA, MANB, EXTA, EXTB, TIMe}
QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:WAITEvent? <x>:= scenario number <y>:= sequence number <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the first sequence waiting event in the first Scenario of channel 1 as manual trigger B: *C1:AWG:SCENario1:SEQUence1:WAITEvent MANB* Query the first sequence cycle number of the first Scenario in channel 1: *C1:AWG:SCENario1:SEQUence1:WAITEvent?* Return: *MANB\n*

3.16.56 <channel>:AWG:SCENario<x>:GOTO <value>

DESCRIPTION	This command is used to set or query the next sequence of sequence in Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:GOTO <value> <x>:= scenario number <y>:= sequence number <channel>:= {C1, C2, C3, C4} <value>:= [0, Number of sequence set]
QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:GOTO? <x>:=scenario number <y>:=sequence number <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set that the next play of the first sequence in the first Scenario of channel 1 is the third sequence: *C1:AWG:SCENario1:SEQUence1:GOTO 3* Query the next play sequence number of the first sequence of the first Scenario of channel 1: *C1:AWG:SCENario1:SEQUence1:GOTO?* Return: *3\n*

3.16.57 <channel>:AWG:SCENario<x>: SEQUence<y>:PLAYBack <type>

DESCRIPTION	This command is used to set or query the playback mode of sequence in Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:PLAYBack <type> <x>:= scenario number <y>:= sequence number <channel>:= {C1, C2, C3, C4} <type>:= {AUTO, SINGLE, CONDITIONAL}
QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:PLAYBack? <x>:= scenario number <y>:= sequence number <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the playback mode of the first sequence of the first Scenario of channel 1 to single: *C1:AWG:SCENario1:SEQUence1:PLAYBack SINGLE* Query the playback mode of the first sequence of the first Scenario in channel 1:

	C1:AWG:SCENario1:SEQUence1:PLAYBack? Return: *SINGLE\n*

3.16.58 <channel>:AWG:SCENario<x>:SEQUence<y>:OUTEvent <type>

DESCRIPTION	This command is used to set or query the jump event of sequence in Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:OUTEvent <type> <x>:= scenario number <y>:= sequence number <channel>:= {C1, C2, C3, C4} <type>:= {MANA, MANB, EXTA, EXTB, TIMe}
QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:OUTEvent? <x>:= scenario number <y>:= sequence number <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the first sequence jump event of the first Scenario of channel 1 to manually trigger a: *C1:AWG:SCENario1:SEQUence1:OUTEvent MANA* Query the jump event of the first sequence of the first Scenario in channel 1: *C1:AWG:SCENario1:SEQUence1:OUTEvent?* Return: *MANA\n*

3.16.59 <channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent:CLEAR

DESCRIPTION	This command is used to clear the segment list of a sequence in a Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent:CLEAR <x>:= scenario number <y>:= sequence number <channel>:= {C1, C2, C3, C4}
EXAMPLE	Empty the Segment in the first sequence in the first Scenario of channel 1: *C1:AWG:SCENario1:SEQUence1:SEGMent:CLEAR*

3.16.60 <channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent:STARTNumb <value>

DESCRIPTION	This command is used to set the starting segment number of sequence in Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent:STARTNumb <value> <x>:= scenario number <y>:= sequence number <channel>:= {C1, C2, C3, C4} <value>:= segment number
QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent: STARTNumb? <x>:= scenario number <y>:= sequence number <channel>:= {C1, C2, C3, C4}
EXAMPLE	Set the starting Segment number of the first sequence of the first Scenario of channel 1 to 3: *C1:AWG:SCENario1:SEQUence1:SEGMent:STARTNumb 3* Query the starting Segment number of the first sequence in the first Scenario of channel 1: *C1:AWG:SCENario1:SEQUence1:SEGMent:STARTNumb?* Return: *3\n*

3.16.61 <channel>:AWG:SCENario<x>:SEQUence<y>:SAVE PATH,<path>,SWFS,<swfs>

DESCRIPTION	This command is used to save the configuration and waveform data of the specified sequence in the specified Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SAVE PATH,<path>,SWFS,<swfs> <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <path>:= Save the path and file name, the file name needs to be suffixed. iqseq. <swfs>:= {"TRUE", "FALSE"}

EXAMPLE	Channel 3 saves the configuration of the first sequence in the first scenario and does not contain waveform data: *C3:AWG:SCENario1:SEQUence1:SAVE PATH,"Local/aaa.iqseq",SWFS,"FALSE"*

3.16.62 <channel>:AWG:SCENario<x>:SEQUence:LOAD PATH,<path>

DESCRIPTION	This command loads the sequence configuration and waveform data file under the specified Scenario.
COMMAND SYNTAX	<channel>:AWG: SCENario <x>: SEQUence:LOAD <path> <channel>:= {C1, C2, C3, C4} <x>:= scenario number <path>:= The path and file name of the load, including the suffix. iqseq.
EXAMPLE	Channel 3 First Scenario Load File: *C3:AWG: SCENario1:SEQUence:LOAD "Local/aaa.iqseq"*

3.16.63 <channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent:COUNt?

DESCRIPTION	This command is used to query the segment number of a sequence in a Scenario.
QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent:COUNt? <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number
EXAMPLE	Query the number of the first sequence in the first Scenario of channel 3: *C3:AWG:SCENario1:SEQUence1:SEGMent:COUNT?* Return: *3\n*

3.16.64 <channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent:INSErt <pos>

DESCRIPTION	This command is used to insert a segment after a segment of a sequence in a Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent:INSErt

	<pos> <channel>:= {C1, C2, C3, C4} <x>:=scenario number <y>:=sequence number <pos>:= segment number
EXAMPLE	The first sequence in the first Scenario of channel 3 inserts a segment to the fifth: *C3:AWG:SCENario1:SEQUence1:SEGMent:INSErt 5*

3.16.65 <channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent: DELEte <pos>

DESCRIPTION	This command is used to delete a segment of a sequence of a Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent:DELEte <pos> <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <pos>:= segment number
EXAMPLE	Delete the fifth segment of the first sequence in the first Scenario of channel 3: *C3:AWG:SCENario1:SEQUence1:SEGMent:DELEte 5*

3.16.66 <channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent:MULTIDelete <pos1,pos2,pos3...>

DESCRIPTION	This command is used to delete multiple segment of the specified sequence of the specified Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent:MULTIDelete <pos1, pos2, pos3...> <x>:= scenario number <y>:= sequence number <channel>:= {C1, C2, C3, C4} <pos1, pos2, pos3...>:= [1, Number of segment set]
EXAMPLE	Delete the 2nd, 3rd and 6th segment of the 2nd sequence in the 2nd Scenario of channel 3: *C3:AWG:SCENario2:SEQUence2:SEGMent:MULTIDelete 2,3,6*

3.16.67 <channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>:LOOP <value>

DESCRIPTION	This command is used to set or query the number of cycles of a segment of a sequence in a Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>: LOOP <value> <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number <value>:= integer
QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>: LOOP? <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number
EXAMPLE	Set the number of first segment cycles of the first sequence of the first Scenario of channel 1 to 5 : *C1:AWG:SCENario1:SEQUence1:SEGMent1:LOOP 5* Query the number of first segment cycles of the first sequence of the first Scenario of channel 1 : *C1:AWG:SCENario1:SEQUence1:SEGMent1:LOOP?* Return : *5\n*

3.16.68 <channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>: WAITEvent <type>

DESCRIPTION	This command is used to set or query the waiting event of a segment of a sequence of a Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>: WAITEvent <type> <channel>:= {C1, C2, C3, C4} <x>:=scenario number <y>:=sequence number <z>:= segment number <type>:= {AUTO, MANA, MANB, EXTA, EXTB, TIMe}
QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>: SEGMent<z>:WAITEvent?

	<channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number
EXAMPLE	Set the waiting event of the first segment of the first sequence of the first Scenario of channel 1 as manual trigger B: *C1:AWG:SCENario1:SEQUence1:SEGMent1:WAITEvent MANB* Query the wait event of the first segment of the first sequence of the first Scenario of channel 1: *C1:AWG:SCENario1:SEQUence1:SEGMent1:WAITEvent?* Return: *MANB\n*

3.16.69 <channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>: GOTO <value>

DESCRIPTION	This command is used to set or query the next segment of a sequence of a Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>: GOTO <value> <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number <value>:= { Integer, not exceeding the maximum number of segment}
QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>:GOTO? <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number
EXAMPLE	Set the first Segment of the first sequence of the first Scenario of channel 1 and the next segment of 3: *C1:AWG:SCENario1:SEQUence1:SEGMent1:GOTO 3* Query the next of the first segment of the first sequence of the first Scenario of channel 1: *C1:AWG:SCENario1:SEQUence1:SEGMent1:GOTO?* Return: *3\n*

3.16.70 <channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>: PLAYBack <type>

DESCRIPTION	This command is used to set or query the playback mode of a segment of a sequence of a Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>: PLAYBack <type> <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number <type>:= {AUTO, SINGLE, CONDITIONAL}
QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>:PLAYBack? <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number
EXAMPLE	Set the playback mode of the first segment of the first sequence of the first Scenario of channel 1 to single: *C1:AWG:SCENario1:SEQUence1:SEGMent1:PLAYBack SINGLE* Query the playback mode of the first segment of the first sequence of the first Scenario of channel 1: *C1:AWG:SCENario1:SEQUence1:SEGMent1:PLAYBack?* Return: *SINGLE\n*

3.16.71 <channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>: OUTEvent <type>

DESCRIPTION	This command is used to set or query the jump event of a segment of a sequence of a Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>:OUTEvent <type> <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number <type>:= {MANA, MANB, EXTA, EXTB, TIMe}
QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>:OUTEvent?

	<channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number
EXAMPLE	Set the jump event of the first segment of the first sequence of the first Scenario of channel 1 as the segment timer: *C1:AWG:SCENario1:SEQUence1:SEGMent1:OUTEvent TIMe* Query the jump event of the first segment of the first sequence of the first Scenario of channel 1: *C1:AWG:SCENario1:SEQUence1:SEGMent1:OUTEvent?* Return: *TIMe\n*

3.16.72 <channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>: IQBuildin <wavename>

DESCRIPTION	This command is used to set or query the waveform of a segment of a sequence of a Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>:IQBuildin <wavename> <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number <wavename >:= { For the name of the built-in wave table, See the table below}

2ASK	4ASK	8ASK	BPSK	4PSK
8PSK	DBPSK	4DPSK	8DPSK	8QAM
16QAM	32QAM	64QAM	128QAM	256QAM
16QAM_2	TEST_DC			

	Note: The waveform name or waveform path needs double quotation marks.
QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>: WAVeform? <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number

EXAMPLE	Set the waveform of the first segment of the first sequence of the first Scenario of channel 1 as the built-in 8PSK: *C1:AWG:SCENario1:SEQUence1:SEGMent1:IQBuildin "8PSK"* Query the waveform name of the first segment of the first sequence of the first Scenario of channel 1: *C1:AWG:SCENario1:SEQUence1:SEGMent1:IQBuildin?* Return: *8PSK\n*

3.16.73 <channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>: WAVeform <path>

DESCRIPTION	This command is used to set or query the waveform of a segment of a sequence of a Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>:WAVeform <path> <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number <path >:= { File path (including waveform name and suffix)} Note: The waveform path needs double quotation marks.
QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>: WAVeform? <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number
EXAMPLE	Set the waveform of the first segment of the first sequence of the first Scenario of channel 1 as 16QAM_UserIQ_1.ARB under the Local/ EasyIQ_arb path: *C1:AWG:SCENario1:SEQUence1:SEGMent1:WAVeform "Local/EasyIQ_arb/16QAM_UserIQ_1.ARB"* Query the waveform name of the first segment of the first sequence of the first Scenario of channel 1: *C1:AWG:SCENario1:SEQUence1:SEGMent1:WAVeform?* Return: *16QAM_UserIQ_1.ARB\n*

3.16.74 <channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>: AMPlitude <value>

DESCRIPTION	This command is used to set the amplitude of a segment waveform of a sequence of a Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z> :AMPlitude <value> <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number <value>:= Amplitude, unit Vpp
QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z> :AMPlitude? <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number
EXAMPLE	Set the waveform amplitude of the first segment of the first sequence of the first Scenario of channel 1 to 0.3 Vpp: *C1:AWG:SCENario1:SEQUence1:SEGMent1:AMPlitude 0.3* Query the waveform amplitude of the first segment of the first sequence of the first Scenario of channel 1.: *C1:AWG:SCENario1:SEQUence1:SEGMent1:AMPlitude?* Return: *0.3\n*

3.16.75 <channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>: OFFset <value>

DESCRIPTION	This command is used to set or query the offset of a segment waveform of a sequence of a Scenario.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z> :OFFset <value> <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number <value>:= Offset,The unit is Vdc.

QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z> :OFFset? <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number
EXAMPLE	Set the waveform offset of the first segment of the first sequence of the first Scenario of channel 1 to 0.3 Vdc: *C1:AWG:SCENario1:SEQUence1:SEGMent1:OFFset 0.3* Query the waveform offset of the first segment of the first sequence of the first Scenario of channel 1: *C1:AWG:SCENario1:SEQUence1:SEGMent1:OFFset?* Return: *0.3\n*

3.16.76 <channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>:MARK1er:POS<K> <state>

DESCRIPTION	This command is used to set or query the marker1 switch state of segment.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z> :MARK1er:POS<K> <state> <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number <K>:= Tag number, waveform data length. < state >:= {ON, OFF}
QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z> :MARK1er? <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number <K>:= Tag number, waveform data length.
EXAMPLE	Set the data 1 of the mark 1 of the first segment of the first sequence of the first Scenario of channel 1 to open: *C1:AWG:SCENario1:SEQUence1:SEGMent1:MARK1er:POS1 ON* Query the marker1 status of the first segment of the first sequence

	of the first Scenario of channel 1: *C1:AWG:SCENario1:SEQUence1:SEGMent1:MARK1er?* Return: *"0,1,2,4"\n* # Data 0, 1, 2 and 4 data bits are marked on.

3.16.77 <channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z>:MARK2er:POS<K> <state>

DESCRIPTION	This command is used to set or query the marker2 switch state of segment.
COMMAND SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z> :MARK2er:POS<K> <state> <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number <K>:= Tag number, waveform data length. < state >:= {ON, OFF}
QUERY SYNTAX	<channel>:AWG:SCENario<x>:SEQUence<y>:SEGMent<z> :MARK2er? <channel>:= {C1, C2, C3, C4} <x>:= scenario number <y>:= sequence number <z>:= segment number <K>:= Tag number, waveform data length.
EXAMPLE	Set the data 1 of the mark 2 of the first segment of the first sequence of the first Scenario of channel 1 to open: *C1:AWG:SCENario1:SEQUence1:SEGMent1:MARK2er:POS1 ON* Query the marker2 status of the first segment of the first sequence of the first Scenario of channel 1: *C1:AWG:SCENario1:SEQUence1:SEGMent1:MARK2er?* Return: *"0,1,2,4"\n* # Data 0, 1, 2 and 4 data bits are marked on.

Note: the numbers of scenario, sequence and segment in a single-wave command are all 1. For example, the length under a single wave is set as follows:

C1:AWG:SCENario1:SEQUence1:SEGMent1:LENGth 5000000

3.17 Clock Command

3.17.1 Clock Source Command

DESCRIPTION	This command sets or queries the clock source.
COMMAND SYNTAX	ROSCillator <src> <src>:= {INT, EXT,SAMPEXT}
QUERY SYNTAX	ROSCillator?
EXAMPLE	Set the clock source as internal clock: *ROSCillator INT* Query clock source configuration: *ROSCillator?* Return: *ROSC\sINT,10MOUT,OFF,AMPL,31dBm,SAMPOUT,OFF,AMPL,31dBm,SYNCOUT,OFF,AMPL,31dBm\n*

3.17.2 Clock Output Related Command

DESCRIPTION	This command is used to set or query the output and output amplitude of 10M clock, sampling clock and synchronous clock.
COMMAND SYNTAX	ROSCillator <parameter>,<value> <parameter>:= { Parameters in the following table } <value>:= { Values of related parameters in the following table }

Parameter	Value	Description
10MOUT	<state>	:= {ON,OFF}. Turn on or off the 10M clock output switch.
SAMPOUT	<state>	:= {ON,OFF}. Turn on or off the sampling clock output switch.
SYNCOUT	<state>	:= {ON,OFF}. Turn on or off the synchronous clock output switch.
10MAMPL	<value>	:= {3 ~ 10}. The unit is dBm. 10M clock output amplitude.
SAMPAMPL	<value>	:= {3 ~ 10}.

			The unit is dBm. Sampling clock output amplitude.
	SYNCAMPL	<value>	:= {3 ~ 10}. The unit is dBm. Synchronous clock output amplitude.
QUERY SYNTAX	ROSCillator?		
EXAMPLE	Turn on 10M clock output switch: *ROSCillator 10MOUT,ON* Set the output amplitude of 10M clock to 5 dBm: *ROSCillator 10MAMPL,5* Query clock output status: *ROSCillator?* Return: *ROSC\sINT,10MOUT,ON,AMPL,20dBm,SAMPOUT,OFF,AMPL,31dBm,SYNCOUT,OFF,AMPL,31dBm\n*		

3.18 Trigger Type Command

DESCRIPTION	This command sets or query the trigger type.
COMMAND SYNTAX	TRIGger:TYPE <type> < type>:= {ALONE, ALL}
QUERY SYNTAX	TRIGger:TYPE?
RESPONSE FORMAT	<type>
EXAMPLE	Set the trigger type to ALL: *TRIGger:TYPE ALL* Query trigger type: *TRIGger:TYPE?* Return: *ALL\n*

3.19 Buzzer Command

DESCRIPTION	This command is used to turn the buzzer on or off.
COMMAND SYNTAX	BUZZer <status> <status>:= {ON, OFF}
QUERY SYNTAX	BUZZer?

RESPONSE FORMAT	BUZZ <status>
EXAMPLE	Turn on the buzzer: *BUZZ ON*

3.20 Screen Saver Command

DESCRIPTION	This command is used to close or set the screen saver time .
COMMAND SYNTAX	SCreen_SaVe <parameter> <parameter>:= {OFF, 1, 5, 15, 30, 60, 120, 300}. The unit is minutes.
QUERY SYNTAX	SCreen_SaVe?
RESPONSE FORMAT	SCSV <parameter>
EXAMPLE	Set the screen saver time to 5 minutes: *SCSV 5* Query the current screen saver time: *SCreen_SaVe?* Return: *SCSV 5MIN*

3.21 Key Command

DESCRIPTION	This command is used to turn on or off the front panel keys.
COMMAND SYNTAX	KEY <state> <state>:= {ON,OFF}
QUERY SYNTAX	KEY?
RESPONSE FORMAT	KEY <state>
EXAMPLE	Set the enabling state of the front panel key to ON: *KEY ON*

3.22 Language Command

DESCRIPTION	This command is used to set or query the system language.
COMMAND SYNTAX	LAnGuaGe <type> <type>:= {EN,CH},EN is English and CH is Chinese.
QUERY SYNTAX	LAnGuaGe?
RESPONSE FORMAT	LAGG <type>

EXAMPLE	Set the language to English: *LAGG EN* Query the current system language: *LAGG?* Return: *LAGG EN\n*

3.23 Date And Time Command

DESCRIPTION	This command is used to set the date and time of the device.
COMMAND SYNTAX	SYST:DATE <date> <date>:= {Date to be set, format: yyyy/mm/dd} SYST:TIME <time> <time>:= { Time to be set, format: hh/mm/ss}
QUERY SYNTAX	SYST:DATE? SYST:TIME?
EXAMPLE	Set the date as January 10, 2021: *SYST:DATE 20210110* The setting time is 10:06:32: *SYST:TIME 100632*

3.24 Screenshot Command

DESCRIPTION	This command is used to set screen shots and format settings.
COMMAND SYNTAX1	SCREEN_SHOT
COMMAND SYNTAX2	SCREEN_SHOT_TYPE <type> <type>:= {BMP, PNG}
QUERY SYNTAX	SCREEN_SHOT_TYPE?
EXAMPLE	Screenshot: *SCREEN_SHOT* Set the screenshot format to BMP: *SCREEN_SHOT_TYPE BMP* Query screenshot format: *SCREEN_SHOT_TYPE?* Return: *BMP\n*

3.25 File Operation Commands

3.25.1 MMEMory:DELete

DESCRIPTION	This command deletes the file.
COMMAND SYNTAX	MMEMory:DELete <parameter> <parameter>:=”The path of the file”.
EXAMPLE	Delete a file whose path is “Local/1000pts.bin”: *MMEMory:DELete “Local/1000pts.bin”*

3.25.2 MMEMory:RDIRectory

DESCRIPTION	This command deletes the directory.
COMMAND SYNTAX	MMEMory:RDIRectory <parameter> <parameter>:=”The path of the directory”.
EXAMPLE	Delete a directory whose path is “Local/test”: *MMEMory:RDIRectory “Local/test”*

3.25.3 MMEMory:MDIRectory

DESCRIPTION	This command creates a new directory.
COMMAND SYNTAX	MMEMory:MDIRectory <parameter> <parameter>:=”The path of the directory”.
EXAMPLE	Create a directory whose path is “Local/test”: *MMEMory:MDIRectory “Local/test”*

3.25.4 MMEMory:CATalog

DESCRIPTION	This command checks the files and the directories from the path or checks the file with specific type.
COMMAND SYNTAX	MMEMory:CATalog? <parameter> <parameter>:=”The path of the directory”. MMEMory:CATalog:DATA:ARBitrary? <parameter> <parameter>:=”The path of the directory”.

	MMEMory:CATalog:STATe:XMLanguage? <parameter> <parameter>:=”The path of the directory”.
RESPONSE FORMAT	remain space, used space “File Name, File Type, File size”
EXAMPLE	Check the files and directories whose path is “Local/”: *MMEMory:CATalog? “Local”* Check the files with “.arb” or “.ARB” postfix whose path is “Local/”: *MMEMory:CATalog:DATA:ARBitrary? “Local”* Check the files with “.xml” or ”.XML” postfix whose path is “Local/”: *MMEMory:CATalog:STATe:XMLanguage? “Local”*

3.25.5 MMEMory:COPY

DESCRIPTION	This command copies a file or a directory .
COMMAND SYNTAX	MMEMory:COPY <parameter> <parameter>:= ”The path of the source”, “The path that the file is about to pasted to”.
EXAMPLE	Copy the file whose path is ”Local/test/1000pts.bin” and paste to “Local/1000pts.bin” *MMEMory:COPY “Local/test/1000pts.bin”,”Local/1000pts.bin”* Copy the directory whose path is ”Local/src” and paste to “Local/copy/” *MMEMory:COPY “Local/src”, ”Local/copy”*

3.25.6 MMEMory:MOVE

DESCRIPTION	This command movesthe file or the directory to a new location.
COMMAND SYNTAX	MMEMory:MOVE<parameter> <parameter>:= ”The path of the source”, “The path of the source that is about to move to”.
EXAMPLE	Move the file whose path is ”Local/test/1000pts.bin” to “Local/1000pts.bin” *MMEMory:MOVE “Local/test/1000pts.bin”,”Local/1000pts.bin”* Move the directory whose path is ”Local/src” to “Local/copy/paste” *MMEMory:MOVE “Local/src”, ”Local/copy/”*

3.25.7 MMEMory:SAVE:XML

DESCRIPTION	This command saves an xml configuration file to the default path or the specified path.
COMMAND SYNTAX	MMEMory:SAVE:XML <path> <path>:= { Saved path, including file name and suffix}
EXAMPLE	Save the test.xml file to the local path: *MMEMory:SAVE:XML "Local/test.xml"* or *MMEMory:SAVE:XML "test.xml"* Save the test.xml file to the network storage disk: *MMEMory:SAVE:XML "net_storage/test.xml"* Save the test.xml file to the U disk path: *MMEMory:SAVE:XML "U-disk0/test.xml"*

3.25.8 MMEMory:LOAD:XML

DESCRIPTION	This command loads an xml configuration file from the default path or the specified path.
COMMAND SYNTAX	MMEMory:LOAD:XML <path> <path>:= { Path, including file name and suffix }
EXAMPLE	Load the test.xml file from the local path: *MMEMory:LOAD:XML "Local/test.xml"* or *MMEMory:LOAD:XML "test.xml"* Load the test.xml file from the network storage disk: *MMEMory:LOAD:XML "net_storage/test.xml"* Load the test.xml file from the USB flash drive path: *MMEMory:LOAD:XML "U-disk0/test.xml"*

3.25.9 MMEMory:TRANsfer

DESCRIPTION	This command can send customized data to the specified bin file in the specified path.
COMMAND SYNTAX	MMEMory:TRANsfer <path>,#{data} <path>:= { Saved path, including path, file name and suffix} {data}:= Length of data length+data length+binary data
EXAMPLE	Send data with data length of 1 and data length of 4 to wave1.bin of

	local path: *MMEMory:TRANsfer "Local/wave1.bin",#14ABCD* Send data with data length of 1 and data length of 4 to wave1.bin of the U disk path: *MMEMory:TRANsfer "U-disk0/wave1.bin",#14ABCD* Send data with data length of 1 and data length of 4 to wave1.bin of network storage disk: *MMEMory:TRANsfer "net_storage/wave1.bin",#14ABCD*

3.26 IP Command

DESCRIPTION	This parameter is used to set or query the IP address of the device.
COMMAND SYNTAX	SYSTem:COMMunicate:LAN<num>:IPADdress <param1>.<param2>.<param3>.<param4> <num>:= [1, 2] <param1>:= {0~255} <param2>:= {0~255} <param3>:= {0~255} <param4>:= {0~255}
QUERY SYNTAX	SYSTem:COMMunicate:LAN1:IPADdress?
EXAMPLE	Set IP address to 10.11.13.203: *SYST:COMM:LAN1:IPAD "10.11.13.203"* Query IP address: *SYST:COMM:LAN1:IPAD?* Return: *"10.11.13.203"*

3.27 Subnet Mask Command

DESCRIPTION	This command is used to set or query the subnet mask of the device.
COMMAND SYNTAX	SYSTem:COMMunicate:LAN<num>:SMASk <param1>.<param2>.<param3>.<param4> <num>:= [1, 2] <param1>:= {0~255}

	<param2>:= {0~255} <param3>:= {0~255} <param4>:= {0~255}
QUERY SYNTAX	SYSTem:COMMunicate:LAN1:SMASk?
EXAMPLE	Set the subnet mask to 255.0.0.0: *SYST:COMM:LAN1:SMAS "255.0.0.0"* Query subnet mask: *SYST:COMM:LAN1:SMAS?* Return: *"255.0.0.0"*

3.28 Gateway Command

DESCRIPTION	This command sets or queries the gateway of the device.
COMMAND SYNTAX	SYSTem:COMMunicate:LAN<num>:GATeway <param1>.<param2>.<param3>.<param4> <num>:= [1, 2] <param1>:= {0~255} <param2>:= {0~255} <param3>:= {0~255} <param4>:= {0~255}
QUERY SYNTAX	SYSTem:COMMunicate:LAN1:GATeway?
EXAMPLE	Set the gateway to 10.11.13.5: *SYSTem:COMMunicate:LAN1:GATeway "10.11.13.1"* Query gateway: *SYSTem:COMMunicate:LAN1:GATeway?* Return: *"10.11.13.1"*

3.29 Synchronization Command

DESCRIPTION	This command is used to set or query the synchronous output of the channel.
COMMAND SYNTAX	<channel>:SYNC <state> <channel>:= {C1, C2, C3, C4}

	<state>:= {ON, OFF} <channel>:SYNC TYPE, <type> <channel>:= {C1, C2, C3, C4} <type>:= {CH, MOD_CH}
QUERY SYNTAX	<channel>:SYNC? <channel>:= {C1, C2, C3, C4}
EXAMPLE	Open channel 1 synchronous signal output function: *C1:SYNC ON* Channel 1 outputs CH1 synchronization signal: *C1:SYNC TYPE, CH* Query the status of synchronization function of channel 1: *C1:SYNC?* Return: *C1:SYNC ON,TYPE,CH*

3.30 Input Signal Setting Command

DESCRIPTION	This command is used to set or query the input signal configuration.
COMMAND SYNTAX	format 1: SYSTEM:<param1> <value> format 2: SYSTEM:< param1> <param2>,<value> <param1>:= {Parameter 1 in the following table} <param2>:= {Parameter 2 in the following table} <value>:= { Values of related parameters in the following table }

Parameter1	Parameter2	Value	Description
SYNCLEVEL		<value>	:= {-5 ~ 5}, the unit is volt "v". Synchronization signal level. Format 1
SYNCLOAD		<value>	:= {50, 10K}, the unit is ohm " Ω ". Synchronous signal load. Format 1
TRIGLEVEL	EXTA	<value>	:= {-5 ~ 5}, The unit is the volt "v". External trigger A signal level. Format 2
TRIGLEVEL	EXTB	<value>	:= {-5 ~ 5}, The unit is the volt "v". External trigger B signal level. Format 2

	TRIGLOAD	EXTA	<value>	:={50, 10K}, the unit is ohm " Ω ". External trigger A signal load. Format 2
	TRIGLOAD	EXTB	<value>	:={50, 10K}, the unit is ohm " Ω ". External trigger B signal load. Format 2
QUERY SYNTAX	SYSTEM:<param1>? <param1>:= { Parameter 1 in the above table}			
EXAMPLE	Set the synchronization signal load to 10KΩ: *SYSTEM:SYNCLOAD 10K* Set external trigger B level to 1V: *SYSTEM:TRIGLEVEL EXTB,1* Query synchronization signal level: *SYSTEM:SYNCLEVEL?* Return: *0\n* Query external trigger signal level: *SYSTEM:TRIGLEVEL?* Return: *EXTA_LEVEL:0.000000,EXTB_LEVEL:1.200000\n*			

3.31 Dynamic Jump Valid Edge Command

DESCRIPTION	This command is used to set or query the effective edge of dynamic jump.
COMMAND SYNTAX	SYSTEM:PATTERNEDGE <type> <type>:= {RISE, FALL}
QUERY SYNTAX	SYSTEM:PATTERNEDGE?
EXAMPLE	Set the effective edge of dynamic jump as the falling edge: *SYSTEM:PATTERNEDGE FALL* Query dynamic jump valid edge state: *SYSTEM:PATTERNEDGE?* Return: *FALL\n*

3.32 GPIB Command

DESCRIPTION	This command sets or queries the GPIB port number.
COMMAND SYNTAX	SYSTem:COMMunicate:GPIB:ADDRess <value> <value>:= {1-30}
QUERY SYNTAX	SYSTem:COMMunicate:GPIB:ADDRess?
EXAMPLE	Set the current GPIB port number: *SYSTem:COMMunicate:GPIB:ADDRess 20* Query the current GPIB port number: *SYSTem:COMMunicate:GPIB:ADDRess?* Return: *18*

3.33 Web Password Command

DESCRIPTION	This command sets or queries the web password.
COMMAND SYNTAX	WEB:PSW <value> <value>:= { Combination of numbers and letters }
QUERY SYNTAX	WEB:PSW?
EXAMPLE	Set the web password to 123456: *WEB:PSW 123456* Query the web password: *WEB:PSW?* Return: *123456\n*

3.34 Multi-device Synchronization Command

DESCRIPTION	This command sets the synchronization between two or more devices to realize in-phase output.
COMMAND SYNTAX	CASCADE STATE,ON|OFF,MODE,<mode>,DELAY,<delay> <mode>:= {MASTER, SLAVE} <delay>:= {0-0.000025},The unit is seconds, and this parameter can only be set in slave mode.
QUERY SYNTAX	CASCADE?

RESPONSE FORMAT	Slave mode return value: CASCADE STATE,ON,MODE,SLAVE,DELAY,<DELAY> Host mode return value: CASCADE STATE,ON,MODE,MASTER
EXAMPLE	Set the device to slave mode with a delay of 0.0000001s: *CASCADE STATE,ON,MODE,SLAVE,DELAY,0.0000001*

3.35 VNC Port Command

DESCRIPTION	This command set or query VNC port number.
COMMAND SYNTAX	VNC_PORT <value> <value>:= { 5900~5999 }
QUERY SYNTAX	VNC_PORT?
EXAMPLE	Set the VNC port to 5905: *VNC_PORT 5905* Query the VNC port: *VNC_PORT?* Return: *5905\n*

3.36 Preset Command

DESCRIPTION	Set or query the power-on mode.
COMMAND SYNTAX	Format 1: Sys_CFG <mode> <mode>:= {DEFAULT,LAST,USER} Format 2: Sys_CFG <type>,<path> <type>:= {USER,PRESET} <path>:= { The path of the configuration file stored by the user (local, network storage, U disk), including file name and suffix}
QUERY SYNTAX	Sys_CFG?
RESPONSE FORMAT	SCFG <type>
EXAMPLE1	Set power-on to last: *SCFG LAST*

EXAMPLE2	Set the boot recovery file: *SCFG USER,"net_storage/config/state.xml"* or:*SCFG USER,"U-disk0/config/state.xml"* or:*SCFG USER,"Local/state.xml"*
EXAMPLE3	Set recovery file: *SCFG PRESET," net_storage/config/state.xml"* or:*SCFG PRESET,"U-disk0/config/state.xml"* or:*SCFG PRESET,"Local/ state.xml"*

Note: The path must use double quotation marks, including and adding the suffix. xml.

3.37 Remote Mode Command

DESCRIPTION	This command sets the device to enter or exit remote mode.
COMMAND SYNTAX	SCREEN:LOCK <state> <state>:= { ON,OFF }
EXAMPLE	Set to enter remote mode : *SCREEN:LOCK ON*

4 Programming Examples

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

4.1 VISA application example

4.1.1 VC++ example

Environment: Win7 32-bit system, Visual Studio

Description: Access the signal source through USBTMC and TCP/IP respectively, and send the "*IDN?" command on NI-VISA to query device information.

Step:

1. Open the Visual Studio software and create a new VC++ win32 console project.
2. Set up the project environment that calls the NI-VISA library. Two setting methods are given here, namely static and dynamic:

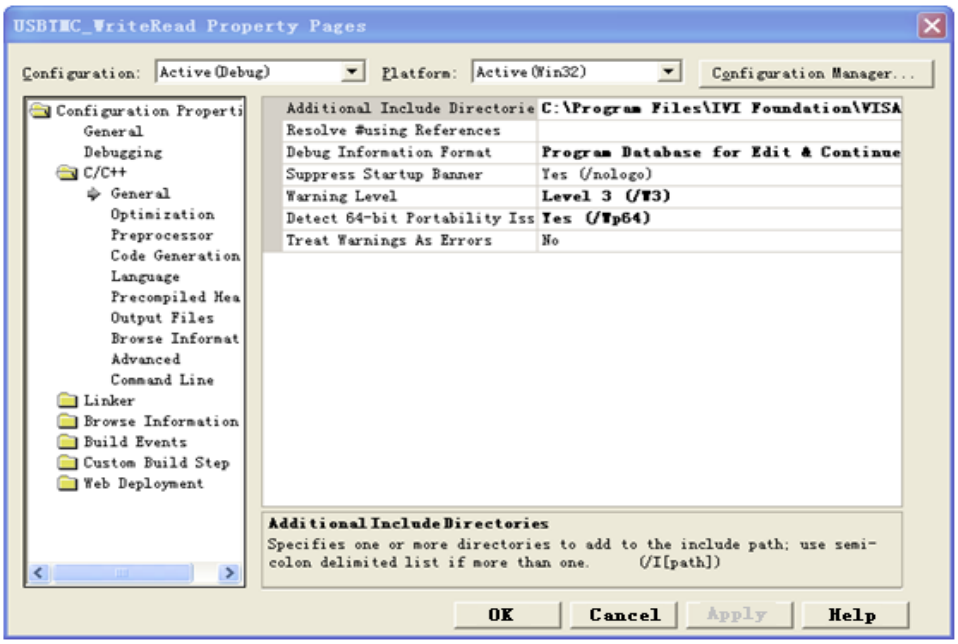
a) Static:

Find: visa.h, visatype.h, visa32.lib files in the NI-VISA installation path, copy them to the root path of the VC++ project and add them to the project. Add the following two lines of code to the projectname.cpp file:

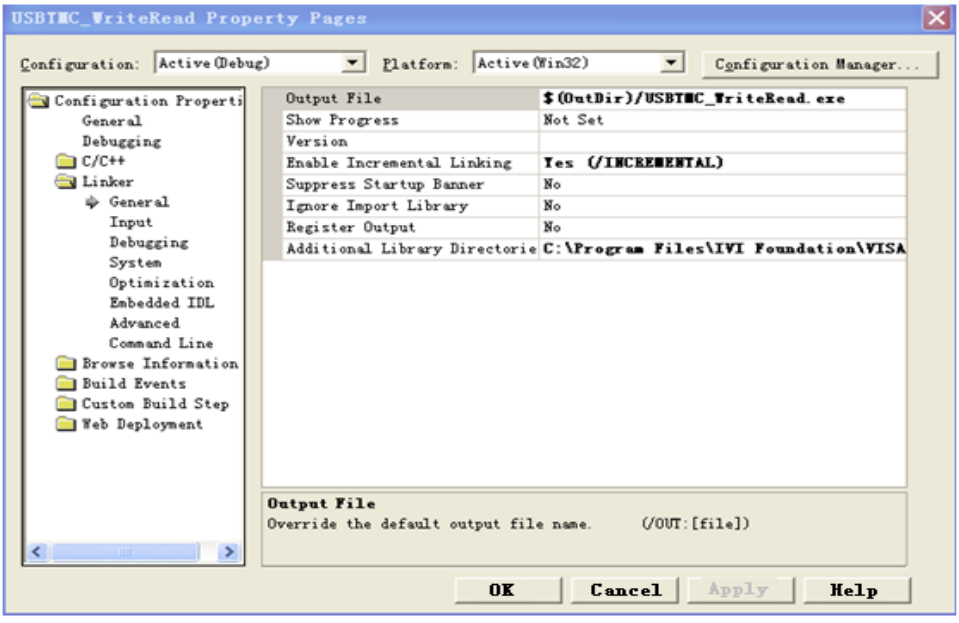
```
#include "visa.h"
#pragma comment(lib,"visa32.lib")
```

b) Dynamic:

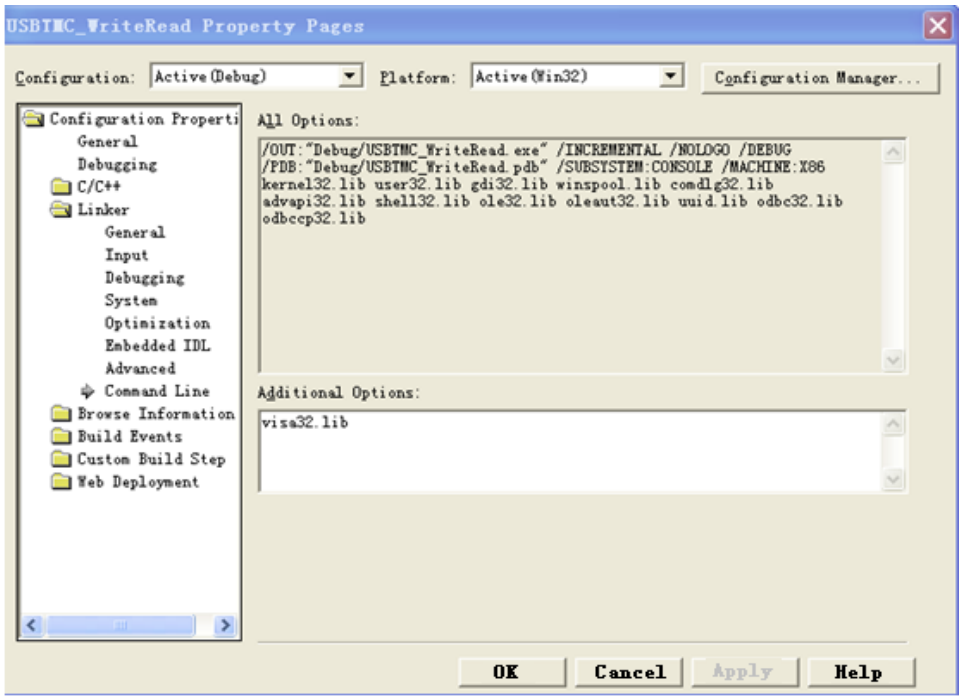
Click "project>>properties" and select "c/c++---General" on the left side of the properties dialog box. Set the value of the "Additional Include Directories" item to the installation path of NI-VISA (for example: C:\Program Files\IVI Foundation\VISA\WinNT\include), as shown in the following figure:



Select "Linker---General" on the left side of the properties dialog box, and set the value of the "Additional Library Directories" item to the installation path of NI-VISA. (For example: C:\Program Files\IVI Foundation\VISA\WinNT\include), as shown below:



Select "Linker---Command Line" on the left side of the properties dialog box, and set the value of the "Additional" item to visa32.lib, as shown in the following figure:



Add the visa.h file on the projectname.cpp file:

#include<visa.h>

3. Encoding:

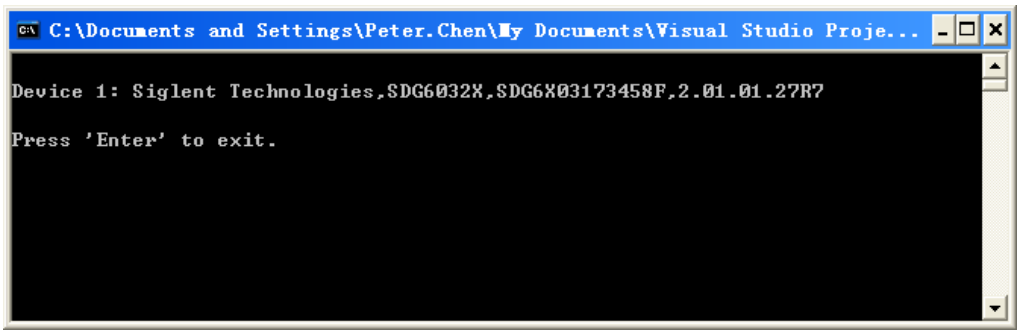
a) USBTMC:

```
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;
}
```

Running results:

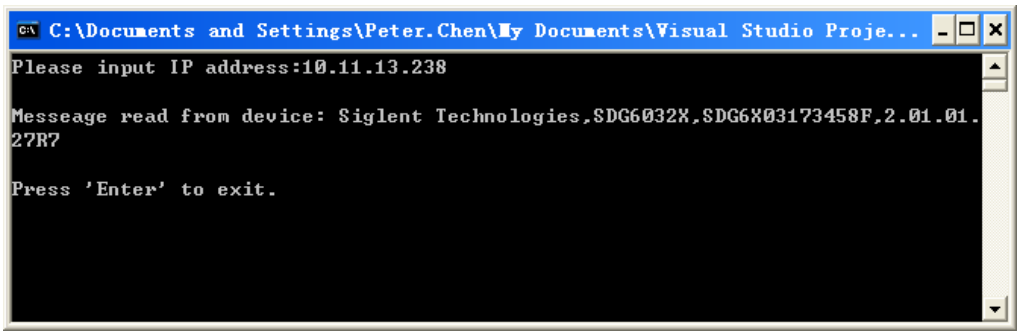


b) TCP/IP:

```
int TCP_IP_Test(char *pIP)
{
char outputBuffer[VI_FIND_BUFLEN];
ViSession defaultRM, instr;
ViStatus status;
/* First we will need to open the default resource manager. */
status = viOpenDefaultRM (&defaultRM);
if (status<VI_SUCCESS)
{
    printf("Could not open a session to the VISA Resource Manager!\n");
}
/* Now we will open a session via TCP/IP device */
char head[256] ="TCPIP0::";
char tail[] ="::INSTR";
strcat(head,pIP);
strcat(head,tail);
status = viOpen (defaultRM, head, VI_LOAD_CONFIG, VI_NULL, &instr);
if (status<VI_SUCCESS)
{
    printf ("An error occurred opening the session\n");
    viClose(defaultRM);
}
status = viPrintf(instr, "*idn?\n");
status = viScanf(instr, "%t", outputBuffer);
if (status<VI_SUCCESS)
{
    printf("viRead failed with error code: %x \n",status);
    viClose(defaultRM);
}
else
{
```

```
            printf ("\nMesseage read from device: %*s\n", 0,outputBuffer);
        }
        status = viClose (instr);
        status = viClose (defaultRM);
        printf("Press 'Enter' to exit.");
        fflush(stdin);
        getchar();
        return 0;
        }

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

Running results:



4.1.2 VB example

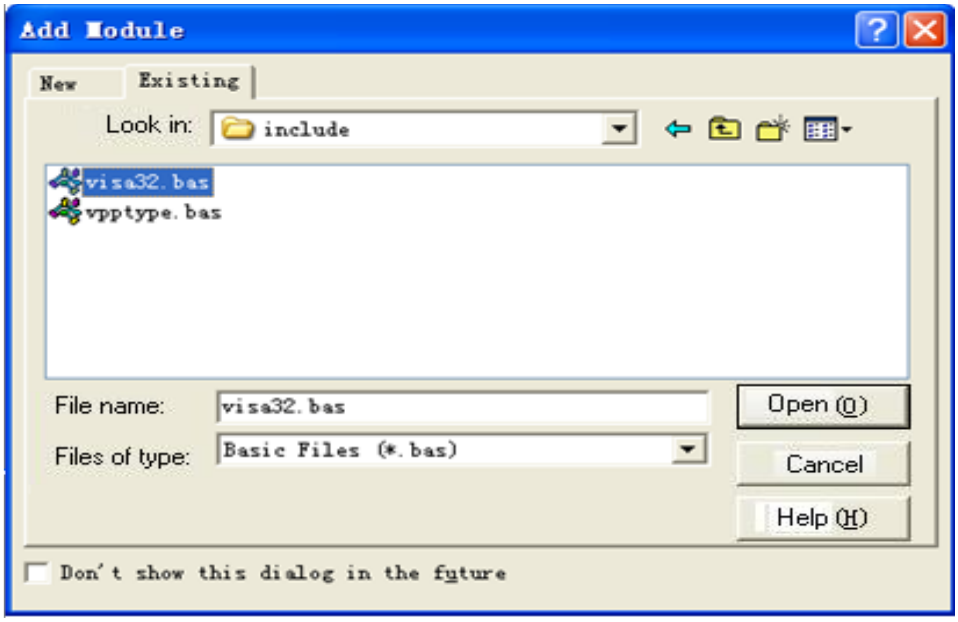
Environment: Windows7 32-bit system, Microsoft Visual Basic 6.0

Description: Access the signal source through USBTMC and TCP/IP respectively, and send the "*IDN?" command on NI-VISA to query device information.

Step:

1. Open the Visual Basic software and create a new standard application project.

Set the project environment for calling the NI-VISA library: click Existing tab of Project>>Add Existing Item, search for the visa32.bas file in the "include" folder under the NI-VISA installation path and add the file. As shown below:



2. Encoding:

a) USBTMC:

```
PrivateFunction Usbtmc_test() AsLong
' 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 AsLong
Dim instrsesn AsLong
Dim numInstrs AsLong
Dim findList AsLong
Dim retCount AsLong
Dim status AsLong
Dim instrResourceString AsString * VI_FIND_BUFLEN
Dim Buffer AsString * MAX_CNT
Dim i AsInteger

' 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
ExitFunction
EndIf

' 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
ExitFunction
EndIf

' 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)
    EndIf
            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
    EndIf

    ' 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
    EndIf

    ' 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
    EndIf
            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
EndFunction
```

b) TCP/IP:

```
PrivateFunction TCP_IP_Test(ByVal ip AsString) AsLong
Dim outputBuffer AsString * VI_FIND_BUFLEN
Dim defaultRM AsLong
```

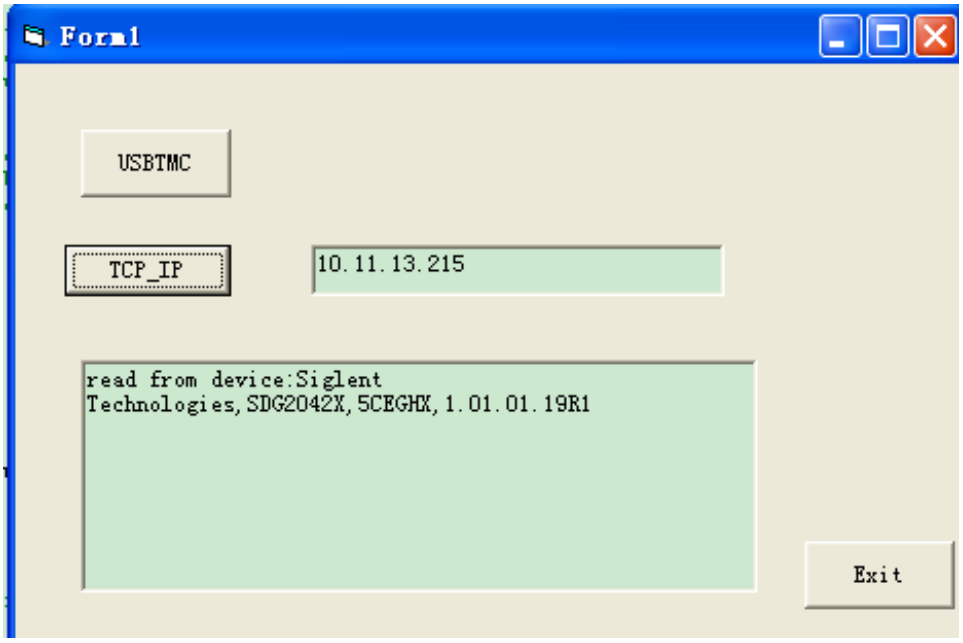
```
Dim instrsesn AsLong
Dim status AsLong
Dim count AsLong

' 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
ExitFunction
EndIf

' 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
ExitFunction
EndIf

    status = viWrite(instrsesn, "*IDN?", 5, count)
If (status < VI_SUCCESS) Then
        resultTxt.Text = "Error writing to the device."
EndIf
    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
EndIf
    status = viClose(instrsesn)
    status = viClose(defaultRM)
    TCP_IP_Test = 0
EndFunction
```

c) Button control code:

```
PrivateSub exitBtn_Click()
End
EndSub
PrivateSub tcpipBtn_Click()
Dim stat AsLong
```

```
        stat = TCP_IP_Test(ipTxt.Text)
    If (stat < VI_SUCCESS) Then
            resultTxt.Text = Hex(stat)
    EndIf
    EndSub
    PrivateSub usbBtn_Click()
    Dim stat AsLong
        stat = Usbtmc_test
    If (stat < VI_SUCCESS) Then
            resultTxt.Text = Hex(stat)
    EndIf
    EndSub
```

Running results:



4.1.3 MATLAB example

Environment: Windows 7 32-bit system, MATLAB R2013a

Description: Access the signal source through USBTMC and TCP/IP respectively, and send the "*IDN?" command on NI-VISA to query device information.

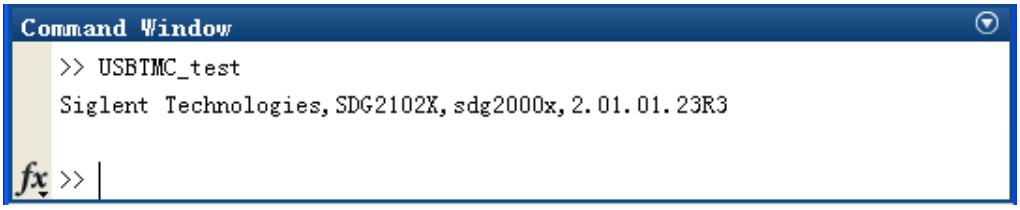
Step:

1. Open the MATLAB software and modify the current directory. In this example, the current directory is modified to: "D:\USBTMC_TCPIP_Demo".
2. Click File>>New>>Script in the Matlab interface to create an empty M file.
3. Encoding:

a) USBTMC:

```
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::0xF4ED::0xEE3A::sdg2000x::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
```

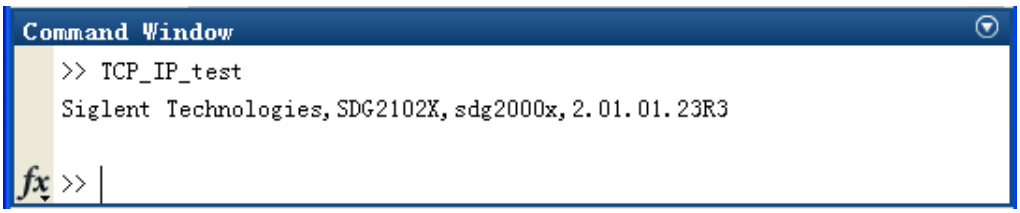
Running results:



b) TCP/IP:

```
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::','10.11.13.32','::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
```

Running results:



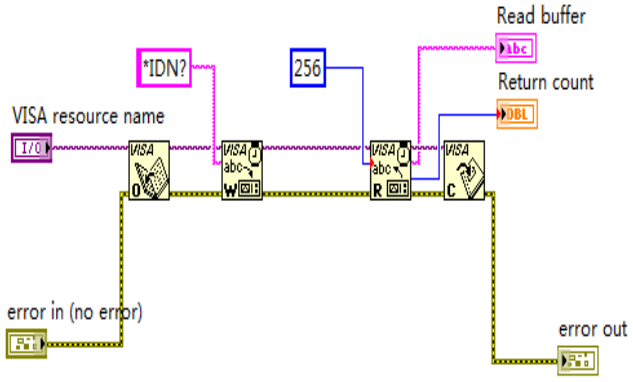
4.1.4 LabVIEW example

Environment: Windows 7 32-bit system, LabVIEW 2011

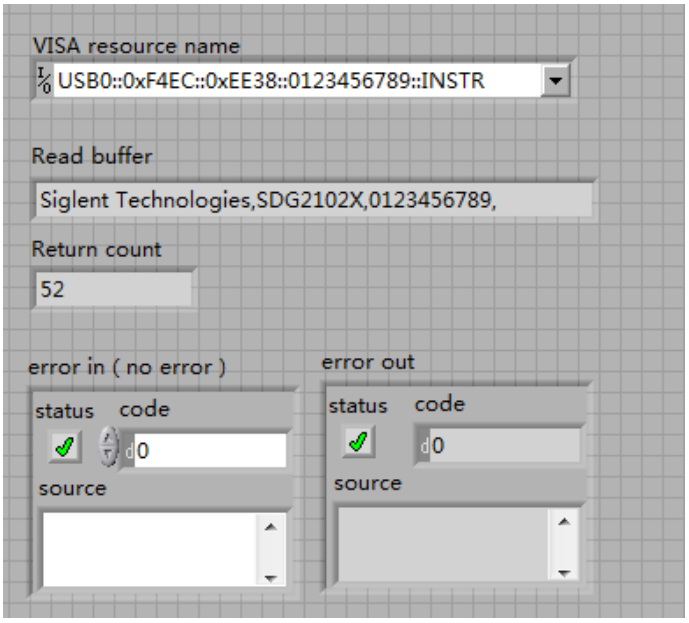
Description: Access the signal source through USBTMC and TCP/IP respectively, and send the "*IDN?" command on NI-VISA to query device information.

Step:

1. Open LabVIEW software and create a VI file.
2. Add controls. Right-click the front panel interface, select and add the VISA resource name, error input, error output, and partial indicators from the control column.
3. Open the block diagram interface. Right-click the VISA resource name, and select and add the following functions in the VISA Palette in the pop-up menu: VISA Write, VISA Read, VISA Open, and VISA Close.
4. Connect them as shown below:

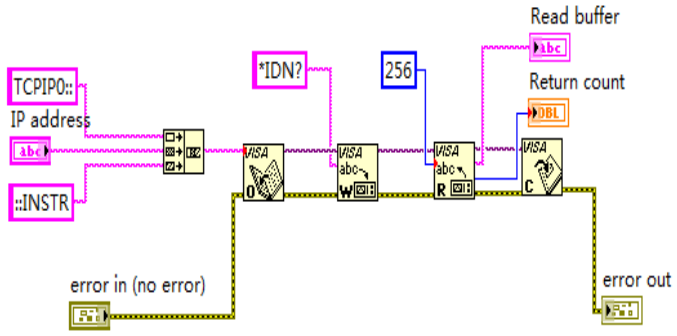


5. Select the device resource from the VISA resource name list and run the program.

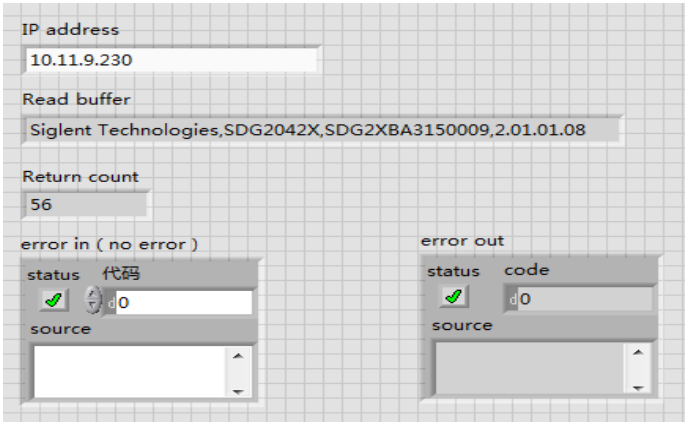


In this example, the VI opens a VISA session to a USBTMC device and writes *IDN? to the device. command, and the response value read back. When all communication is complete, the VI closes the VISA session.

6. Communicating with the device via TCP/IP is similar to USBTMC. But you need to set the VISA write function and VISA read function to synchronous I/O. LabVIEW is set to asynchronous IO by default. Right-click the node and select "Synchronous I/O Mod>>Synchronous" from the shortcut menu to write or read data synchronously.
7. Connect them as shown below:



8. Enter the IP address and run the program.



4.1.5 Python3 example1

Environment：Python3.6.5，PyVISA 1.9

Description: The number of waveform points created in the example. This value represents the codeword for each waveform point. Used to determine the level of the waveform point (the code word range of each waveform point is -32767~32767). Suitable for scenarios with a small number of waveform points. See Chapter 3.6.5.2 for details.

```
#!/usr/bin/env python3.6.5
# -*- coding: utf-8 -*-
import pyvisa as visa
import struct
#USB resource of Device
sdg = visa.ResourceManager().open_resource("USB0::0xF4EC::0x1105::SDG3000XABC002::INSTR")
data = [32767, 32767, 32767, 32767, 32767, 32767, 16384,16384,32767, 32767, 32767, 32767, 32767, 32767, -32767, -32767, -32767, -32767, -32767, -32767, -32767, -32767, -16384,-16384,-32767, -32767, -32767, -32767, -32767]

def int_to_two_byte_small_endian(n):
    return struct.pack('<h', n)

final_byte_data = b''.join(int_to_two_byte_small_endian(num) for num in data)
print('write bytes:', len(final_byte_data))
print(final_byte_data)
cmd = bytes('C1:WVDT WVNM,"wave1",FRQ,2500,AMP,2.5,OFST,0.2,WAVEDATA,', 'utf-8') + final_byte_data
sdg.write_raw(cmd)
# By reading the data of the bin file, the existing waveform can be written to the device.
f = open(r"C:\Users\Administrator\Desktop\wave1.bin", "rb")
data1 = f.read()
print ('write bytes:',len(data1))
cmd = bytes('C1:WVDT WVNM,"wave1",FRQ,2500,AMP,2.5,OFST,0.2,WAVEDATA,', 'utf-8') + data1
sdg.write_raw(cmd)
f.close()
```

4.1.6 Python3 example2

Environment：Python3.6.5，PyVISA 1.9

Description: The number of waveform points created in the example. This value represents the codeword for each waveform point. Used to determine the level of the waveform point (the code word range of each waveform point is -32767~32767). Suitable for scenarios with a small number of waveform points. See Chapter 3.6.5.3 for details.

```
#!/usr/bin/env python3.6.5
# -*- coding: utf-8 -*-
import pyvisa as visa
import struct
#USB resource of Device
sdg = visa.ResourceManager().open_resource("USB0::0xF4EC::0x1105::SDG3000XABC002::INSTR")
data = [32767, 32767, 32767, 32767, 32767, 32767, 16384,16384,32767, 32767, 32767, 32767, 32767, 32767, -32767, -32767, -32767, -32767, -32767, -32767, -32767, -32767, -16384,-16384,-32767, -32767, -32767, -32767, -32767]

def int_to_two_byte_small_endian(n):
    return struct.pack('<h', n)

byte_data1 = b''.join(int_to_two_byte_small_endian(num) for num in data[:8])
print('write bytes:', len(byte_data1))
print(byte_data1)
byte_data2 = b''.join(int_to_two_byte_small_endian(num) for num in data[8:16])
print('write bytes:', len(byte_data2))
print(byte_data2)
byte_data3 = b''.join(int_to_two_byte_small_endian(num) for num in data[16:])
print('write bytes:', len(byte_data3))
print(byte_data3)
cmd1 = bytes('C1:WVDT:SEGMENT WVNM,"wave6",FRQ,3500,AMP,1.5,OFST,0.01,BEGIN,WAVEDATA,', 'utf-8') + byte_data1
cmd2 = bytes('C1:WVDT:SEGMENT WVNM,"wave6",WAVEDATA,', 'utf-8') + byte_data2
cmd3 = bytes('C1:WVDT:SEGMENT WVNM,"wave6",END,WAVEDATA,', 'utf-8') + byte_data3
sdg.write_raw(cmd1)
sdg.write_raw(cmd2)
sdg.write_raw(cmd3)
```

```
# By reading the data of the bin file, the existing waveform can be written to the device.
f = open(r"C:\Users\Administrator\Desktop\wave1.bin", "rb")
data1 = f.read()
cmd1 = bytes('C1:WVDT:SEGMENT WVNM,"wave6",FRQ,3500,AMP,1.5,OFST,0.01,BEGIN,WAVEDATA,', 'utf-8') + data1[:8]
cmd2 = bytes('C1:WVDT:SEGMENT WVNM,"wave6",WAVEDATA,', 'utf-8') + data1[8:16]
cmd3 = bytes('C1:WVDT:SEGMENT WVNM,"wave6",END,WAVEDATA,', 'utf-8') + data1[16:]
sdg.write_raw(cmd1)
sdg.write_raw(cmd2)
sdg.write_raw(cmd3)
f.close()
```

4.2 Examples of Using Sockets

4.2.1 Python Example

Python has a low-level networking module that provides access to the socket interface. Python scripts can be written for sockets to do a variety of tests and measurement tasks.

Environment: Windows 7 32-bit, Python v2.7.5

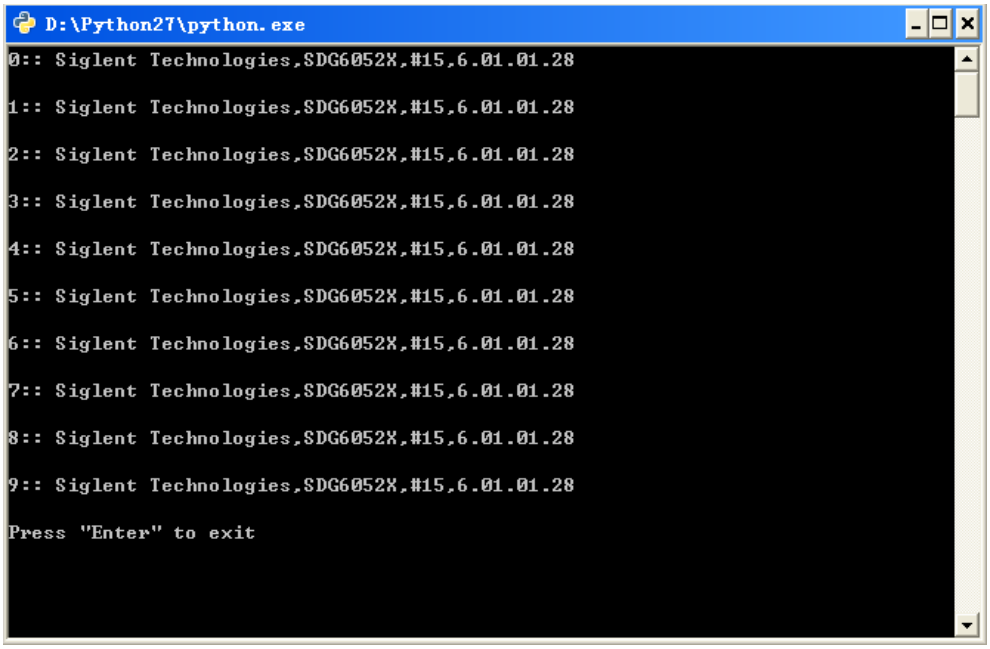
Description: Open a socket, send a query, and repeat this loop 10 times, finally close the socket. Note that SCPI command strings must be terminated with a "\n" (new line) character in programming.

Below is the code of the script:

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

import socket      # for sockets
import sys    # for exit
import time # for sleep

#------------------------------------------------------------------------------

remote_ip = "10.11.13.40"    # should match the instrument's IP address
port = 5025 # the port number of the instrument service
count = 0

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

```
            print ('failed to connect to ip ' + remote_ip)
        return s

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

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

def main():
        global remote_ip
        global port
        global count

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

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

Run result:


D:\Python27\python.exe
0:: Siglent Technologies,SDG6052X,#15,6.01.01.28
1:: Siglent Technologies,SDG6052X,#15,6.01.01.28
2:: Siglent Technologies,SDG6052X,#15,6.01.01.28
3:: Siglent Technologies,SDG6052X,#15,6.01.01.28
4:: Siglent Technologies,SDG6052X,#15,6.01.01.28
5:: Siglent Technologies,SDG6052X,#15,6.01.01.28
6:: Siglent Technologies,SDG6052X,#15,6.01.01.28
7:: Siglent Technologies,SDG6052X,#15,6.01.01.28
8:: Siglent Technologies,SDG6052X,#15,6.01.01.28
9:: Siglent Technologies,SDG6052X,#15,6.01.01.28
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, spectrum analyzers, function/arbitrary waveform generators, RF/MW signal generators, 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 NA, Inc
Add: 6557 Cochran Rd Solon, Ohio 44139
Tel: 440-398-5800
Toll Free:877-515-5551
Fax: 440-399-1211
Email: support@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

Malaysia:

SIGLENT Technologies (M) Sdn.Bhd
Add: NO.6 Lorong Jelawat 4
Kawasan Perusahaan Seberang Jaya
13700, Perai Pulau Pinang
Tel: 006-04-3998964
Email: sales@siglent.com
Website: int.siglent.com

Follow us on
Facebook: SiglentTech