Remote Control Manual

SDG Series
Function/Arbitrary
Waveform Generator

Contents

1 INTRODUCTION OF THE SCPI LANGUAGE 1

1.1 ABOUT COMMANDS & QUERIES 1
1.1.1 How They are Listed 1
1.1.2 How They are Described 1
1.1.3 When can They be Used 1
1.1.4 Command Notation 1
1.2 TABLE OF COMMANDS & QUERIES 2
1.3 IEEE 488.2 COMMON COMMAND INTRODUCTION 3
1.3.1 IDN 3
1.3.2 OPC 4
1.3.3 CLS 5
1.3.4 ESE 5
1.3.5 ESR 6
1.3.6 RST 6
1.3.7 SRE 6
1.3.8 STB 7
1.3.9 TST 8
1.3.10 WAI 8
1.3.11 DDR 9
1.3.12 CMR 9
1.4 COMM_HEADER COMMAND 10
1.5 OUTPUT COMMAND 11
1.6 BASIC WAVE COMMAND 12
1.7 MODULATE WAVE COMMAND 14
1.8 SWEEP WAVE COMMAND 18
1.9 BURST WAVE COMMAND 20
1.10 PARAMETER COPY COMMAND 23
1.11 ARBITRARY WAVE COMMAND 23
1.12 SYNC COMMAND 25
1.13 NUMBER FORMAT COMMEND 25
1.14 LANGUAGE COMMAND 26
1.15 CONFIGURATION COMMAND 26
1.16 BUZZER COMMAND 27
1.17 SCREEN SAVE COMMAND 27
1.18 CLOCK SOURCE COMMAND 28
1.19 FREQUENCY COUNTER COMMAND 28
1.20 INVERT COMMAND 29
1.21 COUPLING COMMAND 30
1.22 VOLTAGE OVERLOAD COMMAND 30
1.23 STORE LIST COMMAND 31
1.24 VIRTUAL KEY COMMAND 32

1.25 IP COMMAND 33
1.26 SUBNET MASK COMMAND 34
1.27 GATEWAY COMMAND 34
1.28 SAMPLING RATE COMMAND 35
1.29 INDEX 36

2 PROGRAMMING DEMOS 37

2.1 VISUAL C++ PROGRAMMING DEMO 37

1 Introduction of the SCPI Language

1.1 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 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 the subject is indicated as a command or query or both. Queries perform actions such as obtaining information, and are recognized by the question mark (?) following the header.

1.1.1 How They are Listed

The descriptions are listed in alphabetical order according to their short format.

1.1.2 How They are Described

In the descriptions themselves, 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.

1.1.3 When can They be Used

The commands and queries listed here can be used for SDGxxxx Series Function/Arbitrary Waveform Generators.

1.1.4 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.

… An ellipsis indicates that the items both to its left and right may be repeated for a number of times.

1.2 Table of Commands & Queries

Short	Long Form	Subsystem	What Command/Query does
*IDN	*IDN	SYSTEM	Gets identification from device.
*OPC	*OPC	SYSTEM	Gets or sets the OPC bit (0) in the Event Status Register (ESR).
*CLS	*CLS	SYSTEM	Clears all the status data registers.
*ESE	*ESE	SYSTEM	Sets or gets the Standard Event Status Enable register (ESE).
*ESR	*ESR	SYSTEM	Reads and clears the contents of the Event Status Register (ESR).
*RST	*RST	SYSTEM	Initiates a device reset.
*SRE	*SRE	SYSTEM	Sets the Service Request Enable register (SRE).
*STB	*STB	SYSTEM	Gets the contents of the IEEE 488.2 defined status register.
*TST	*TST	SYSTEM	Performs an internal self-test.
*WAI	*WAI	SYSTEM	Wait to continue command.
DDR	DDR	SYSTEM	Reads and clears the Device Dependent Register (DDR).
CMR	CMR	SYSTEM	Reads and clears the command error register.
CHDR	COMM_HEADER	SIGNAL	Sets or gets the command returned format
OUTP	OUTPUT	SIGNAL	Sets or gets output state.
BSWV	BASIC_WAVE	SIGNAL	Sets or gets basic wave parameters.
MDWV	MODULATEWAVE	SIGNAL	Sets or gets modulation parameters.
SWWV	SWEEPWAVE	SIGNAL	Sets or gets sweep parameters.
BTWV	BURSTWAVE	SIGNAL	Sets or gets burst parameters.
PACP	PARACOPY	SIGNAL	Copies parameters from one channel to the other.
ARWV	ARBWAVE	DATA	Changes arbitrary wave type.
SYNC	SYNC	SIGNAL	Sets or gets synchronization signal.
NBFM	NUMBER_FORMAT	SYSTEM	Sets or gets data format.
LAGG	LANGUAGE	SYSTEM	Sets or gets language.

SCFG	SYS_CFG	SYSTEM	Sets or gets the power-on system setting. way.
BUZZ	BUZZER	SYSTEM	Sets or gets buzzer state.
SCSV	SCREEN_SAVE	SYSTEM	Sets or gets screen save state.
ROSC	ROSCILLATOR	SIGNAL	Sets or gets state of clock source.
FCNT	FREQCOUNTER	SIGNAL	Sets or gets frequency counter parameters.
INVT	INVERT	SIGNAL	Sets or gets polarity of current channel.
COUP	COUPLING	SIGNAL	Sets or gets coupling parameters.
VOLTPRT	VOLTPRT	SYSTEM	Sets or gets state of over-voltage protection.
STL	STORELIST	SIGNAL	Lists all stored waveforms.
VKEY	VIRTUALKEY	SYSTEM	Sets the virtual keys.
SYST:COMM:LAN:IPAD	SYSTEM:COMMUNICATE:LAN:IPADDRESS	SYSTEM	The Command can set and get system IP address.
SYST:COMM:LAN:SMAS	SYSTEM:COMMUNICATE:LAN:SMASK	SYSTEM	The Command can set and get system subnet mask.
SYST:COMM:LAN:GAT	SYSTEM:COMMUNICATE:LAN:GATEWAY	SYSTEM	The Command can set and get system Gateway.
SRATE	SAMPLERATE	SIGNAL	Sets or gets sampling rate. You can only use it in TrueArb mode

1.3 IEEE 488.2 Common Command Introduction

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.

1.3.1 IDN

DESCRIPTION The *IDN? query causes the instrument to identify itself. The response comprises manufacturer, model, serial number, software version and firmware version.

QUERY SYNTAX *IDN?

RESPONSE FORMAT *IDN, <device id>,<model>,<serial number>, <software version>, <firmware version>.

<device id>:="SDG" is used to identify instrument.
<model>:= A model identifier less than 14 characters, should not contain the word "MODEL".
<serial number>:Each product has its own number, the serial number can labeled product uniqueness.
<software version>:= A serial numbers about software version.
<firmware version>:=The firmware level field, should contain information about all separately revisable subsystems. This information can be contained in single or multiple revision codes.

EXAMPLE Reads version information.
*IDN?
Return:
*IDN SDG, SDG5162, 120465, 5.01.02.05, 02-00-00-21
-24(It may differ from each version)

Notes:
1)

Parameter/command	SDG800	SDG1000	SDG2000X	SDG5000
<firmware version>	yes	yes	yes	no

2) Explain for <firmware version>:value1- value2- value3- value4- value5.
value1: PCB version.
value2: Hardware version.
value3: Hardware subversion.
value4: FPGA version.
value5: CPLD version.

1.3.2 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 *OPC 1

RELATED COMMANDS *WAI

1.3.3 CLS

DESCRIPTION The *CLS command clears all the status data registers.

COMMAND SYNTAX *CLS

EXAMPLE The following command causes all the status data registers to be cleared:
*CLS

RELATED COMMANDS CMR, DDR, *ESR, *STB

1.3.4 ESE

DESCRIPTION The *ESE command sets the Standard Event Status Enable register (ESE). This command allows one or more events in the ESR register to be reflected in the ESB summary message bit (bit 5) of the STB register.
The *ESE? query reads the contents of the ESE register.

COMMAND SYNTAX *ESE <value>
<value> : = 0 to 255.

QUERY SYNTAX *ESE?

RESPONSE FORMAT *ESE <value>

EXAMPLE The following instruction allows the ESB bit to be set if a user request (URQ bit 6, i.e. decimal 64) and/or a device dependent error (DDE bit 3, i.e. decimal 8) occurs. Summing these values yields the ESE register mask 64+8=72.
*ESE?
Return:

*ESE 72

RELATED COMMANDS *ESR

1.3.5 ESR

DESCRIPTION The *ESR? query reads and clears the contents of the Event Status Register (ESR). The response represents the sum of the binary values of the register bits 0 to 7.

QUERY SYNTAX *ESR?

RESPONSE FORMAT *ESR <value>
<value> : = 0 to 255

EXAMPLE The following instruction reads and clears the content of the ESR register:
*ESR?
Return：
*ESR 0

RELATED COMMANDS *CLS, *ESE

1.3.6 RST

DESCRIPTION The *RST command initiates a device reset. The *RST sets all of traces to the GND line and recalls the default setup.

COMMAND SYNTAX * RST

EXAMPLE This example resets the signal generator:
*RST

1.3.7 SRE

DESCRIPTION The *SRE command sets the Service Request Enable register (SRE). This command allows the user to specify which summary message bit(s) in the STB register will generate a service request.

A summary message bit is enabled by writing a '1' into the corresponding bit location. Conversely, writing a '0' into a given bit location prevents the associated event from generating a service request (SRQ). Clearing the SRE register disables SRQ interrupts.
The *SRE? query returns a value that, when converted to a binary number represents the bit settings of the SRE register. Note that bit 6 (MSS) cannot be set and it's returned value is always zero.

COMMAND SYNTAX *SRE <value>
<value> : = 0 to 255

QUERY SYNTAX *SRE?

RESPONSE FORMAT *SRE <value>

EXAMPLE The following instruction allows a SRQ to be generated as soon as the MAV summary bit (bit 4, i.e. decimal 16) or the INB summary bit (bit 0, i.e. decimal 1) in the STB register, or both are set. Summing these two values yields the SRE mask 16+1 = 17.
*SRE?
Return:
*SRE 17

1.3.8 STB

DESCRIPTION The *STB? query reads the contents of the 488.2 defined status register (STB), and the Master Summary Status (MSS).
The response represents the values of bits 0 to 5 and 7 of the Status Byte register and the MSS summary message.
The response to a *STB? query is identical to the response of a serial poll except that the MSS summary message appears in bit 6 in place of the RQS message.

UERY SYNTAX *STB?

RESPONSE FORMAT *STB <value>
<value> : = 0 to 255

EXAMPLE The following reads the status byte register:
*STB?
Return:

*STB 0

RELATED COMMANDS *CLS, *SRE

1.3.9 TST

DESCRIPTION The *TST? query performs an internal self-test and the response indicates whether the self-test has detected any errors. The self-test includes testing the hardware of all channels, the time-base and the trigger circuits.
Hardware failures are identified by a unique binary code in the returned <status> number. A “0” response indicates that no failures occurred.

QUERY SYNTAX *TST?

RESPONSE FORMAT *TST <status>
<status> : = 0 self-test successful

EXAMPLE The following causes a self-test to be performed:
TST?
Return(if no failure):
*TST 0

RELATED COMMANDS *CAL

Note:

Parameter/command	SDG800	SDG1000	SDG2000X	SDG5000
TST	no	yes	yes	yes

1.3.10 WAI

DESCRIPTION The *WAI (WAIT to continue) command, requires by the IEEE 488.2 standard, has no effect on the instrument, as the signal generator only starts processing a command when the previous command has been entirely executed.

COMMAND SYNTAX *WAI

RELATED COMMANDS *OPC

1.3.11DDR

DESCRIPTION The DDR? query reads and clears the contents of the device dependent or device specific error register (DDR). In case of a hardware failure, the DDR register specifies the origin of the failure.

QUERY SYNTAX DDR?

RESPONSE FORMAT DDR <value>
<value> : = 0 to 65535

EXAMPLE DDR?
Return:
DDR 0

The following table gives details：

Bit	Bit Value	Description
15...14		Reserved
13	8192	Time-base hardware failure detected
12	4096	Trigger hardware failure detected
11		Reserved
10		Reserved
9	512	Channel 2 hardware failure detected
8	256	Channel 1 hardware failure detected
7	128	External input overload condition detected
6…4		Reserved
3		Reserved
2		Reserved
1	2	Channel 2 overload condition detected
0	1	Channel 1 overload condition detected

Notes:

Parameter/command	SDG800	SDG1000	SDG2000X	SDG5000
DDR	yes	yes	no	yes

1.3.12CMR

DESCRIPTION The CMR? query reads and clears the contents of the command error register (CMR) .See the table below which specifies the last syntax error type detected by the instrument.

QUERY SYNTAX CMR?

RESPONSE FORMAT CMR <value>
<value> : = 0 to 14

EXAMPLE CMR?
Return:
CMR 0

Value	Description
0	
1	Unrecognized command/query header
2	Invalid character
3	Invalid separator
4	Missing parameter
5	Unrecognized keyword
6	String error
7	Parameter can't allowed
8	Command String Too Long
9	Query cannot allowed
10	Missing Query mask
11	Invalid parameter
12	Parameter syntax error
13	Filename too long
14	Directory not exist

Note:

Parameter/command	SDG800	SDG1000	SDG2000X	SDG5000
CMR	yes	yes	no	yes

1.4 Comm_Header Command

DESCRIPTION This command is used to change the query command returned format. "SHORT" parameter returns short format. "LONG" parameter returns long format. "OFF" returns nothing.

COMMAND SYNTAX CHDR (Comm_HeaDeR) <parameter>
<parameter>:= {SHORT,LONG,OFF}

QUERY SYNTAX CHDR (Comm_HeaDeR)?

RESPONSE FORMAT CHDR <parameter>

EXAMPLE

Set query command format to long.
CHDR LONG

Read query command format.
CHDR?
Return:
COMM_HEADER LONG

Note:

Parameter/command	SDG800	SDG1000	SDG2000X	SDG5000
CHDR	yes	yes	no	yes

1.5 Output Command

DESCRIPTION

Enable or disable the output of the [Output] connector at the front panel corresponding to the channel.
The query returns the output state, load and polarity parameters value.

COMMAND SYNTAX

<channel>:OUTP (OUTPut) <parameter>
<channel>:={C1, C2}
<parameter >:= {a parameter from the table below}

Parameters	Value	Description
ON	---	Turn on
OFF	---	Turn off
LOAD	<load>	Value of load
PLRT	<NOR, INVT>	Value of polarity parameter

< load>:= {please see the note below.}

QUERY SYNTAX

<channel>: OUTP(OUTPut)?

RESPONSE FORMAT

<channel>: OUTP <load>

EXAMPLE

Turn on channel one.
C1: OUTP ON

Read channel one output state.
C1: OUTP?
Return:
C1: OUTP ON, LOAD, HZ, PLRT, NOR

Set the load to 50.
C1: OUTP LOAD, 50

Set the load to HZ.
C1: OUTP LOAD, HZ

Set the polarity normal.
C1: OUTP PLRT, NOR

Set the polarity inverted.
C1: OUTP PLRT, INVT

Note:

Parameter/command	SDG800	SDG1000	SDG2000X	SDG5000
<channel>	no	yes	yes	yes
<load>(default unit is ohm)	50, HZ	50~10000, HZ	50~100000, HZ	50, HZ

1.6 Basic Wave Command

DESCRIPTION Sets or gets basic wave parameters.

COMMAND SYNTAX <channel>:BSWV(BaSic_WaVe) <parameter>
<channel>:={C1, C2}
<parameter>:= {a parameter from the table below}

Parameters	Value	Description
WVTP	<type>	Type of wave
FRQ	<frequency>	Value of frequency. If wave type is Noise or DC, you can't set this parameter.
PERI	<period>	Value of period. If wave type is Noise or DC, you can't set this parameter.
AMP	<amplitude>	Value of amplitude. If wave type is Noise or DC, you can't set this parameter.
OFST	<offset>	Value of offset. If wave type is Noise or DC, you can't set this parameter.
SYM	<symmetry>	Value of symmetry. Only when wave type is Ramp, you can set this parameter.
DUTY	<duty>	Value of duty cycle. Only when wave type is Square and Pulse, you can set this parameter.
PHSE	<phase>	Value of phase. If wave type is Noise or Pulse or DC, you can't set this parameter.
STDEV	<stdev>	Value of Noise wave stdev. Only when wave type is

		Noise, you can set this parameter.
MEAN	<mean>	Value of Noise wave mean. Only when wave type is Noise, you can set this parameter.
WIDTH	<width>	Value of width. Only when wave type is Pulse, you can set this parameter.
RISE	<rise>	Value of rise time. Only when wave type is Pulse, you can set this parameter.
FALL	<fall>	Value of fall time. Only when wave type is Pulse, you can set this parameter.
DLY	<delay>	Value of delay. Only when wave type is Pulse, you can set this parameter.
HLEV	<high level>	Value of high level. If wave type is Noise or DC, you can't set this parameter.
LLEV	<low level>	Value of low level. If wave type is Noise or DC, you can't set this parameter.

Note: if the command doesn't set basic wave type, WVPT parameter will be set to current wave type.

where: <type>:={SINE, SQUARE, RAMP, PULSE, NOISE, ARB ,DC}
<frequency>:= {Default unit is "Hz". Value depends on the model.}
<amplitude>:= {Default unit is "V". Value depends on the model.}
<offset>:= {Default unit is "V". Value depends on the model.}
<duty>:= {0% to 100%. Value depends on frequency.}
<symmetry> :={ 0% to 100%}
<phase>:= { Value depends on the model.}
<stdev>:= {Default unit is "V". Value depends on the model.}
<mean>:= {Default unit is "V". Value depends on the model.}
<width>:= {Max_width < (Max_duty * 0.01) * period and Min_width > (Min_duty * 0.01) * period.}
<rise>:= {Value depends on the model.}
<fall>:= {Value depends on the model.}
<delay>:= {Unit is S. Maximal is Pulse period, minimum value is 0.}

QUERY SYNTAX <channel>: BSWV (BaSic_WaVe)?
<channel>:={C1, C2}

RESPONSE FORMAT <channel>:BSWV<type>,<frequency>,<amplitude>,<offset>, <duty>,<symmetry>, <phase>,<variance>,<mean>,<width>, <rise>, <fall>, <delay>.

EXAMPLE Change channel one wave type to ramp.
C1: BSWV WVTP, RAMP

Change frequency of channel one to 2000 Hz.
C1: BSWV FRQ, 2000

Set amplitude of channel one to 3Vpp.
C1: BSWV AMP, 3

Read channel basic wave parameters from device.
C1: BSWV?
Return:
C1: BSWV WVTP, SINE,FRQ,100HZ,PERI,0.01S,AMP,2V,OFST,0V,HLEV,1V,LLEV,-1V,PHSE,0

RELATED COMMANDS ARWV, BTWV, MDWV, SWWV

Note:

Parameter/command	SDG800	SDG1000	SDG2000X	SDG5000
<channel>	no(single channel)	yes	yes	yes
RISE	yes	no	yes	yes
FAL	yes	no	yes	yes
DLY	no	yes	yes	yes

1.7 Modulate Wave Command

DESCRIPTION Sets or gets modulation parameters.

COMMAND SYNTAX <channel>:MDWV(MoDulateWaVe)<parameter>
<channel>:={C1, C2}
<parameter>:= {a parameter from the table below}

Parameters	Value	Description
STATE	<state>	Turn on or off modulation. Note: if you want to set or read other parameters of modulation, you must set STATE to ON at first.
AM, SRC	<src>	AM signal source.
AM, MDSP	<mod wave shape>	AM modulation wave. Only when AM signal source is set to INT, you can set the parameter.
AM, FRQ	<AM frequency>	AM frequency. Only when AM signal source is set to INT, you can set the parameter.
AM, DEPTH	<depth>	AM depth. Only when AM signal source is set to INT, you can set the parameter.

DSBAM, SRC	<src>	DSBAM signal source.
DSBAM, MDSP	<mod wave shape>	DSBAM modulation wave. Only when AM signal source is set to INT, you can set the parameter.
DSBAM, FRQ	<DSB-AM frequency>	DSBAM frequency. Only when AM signal source is set to INT, you can set the parameter.
FM, SRC	<src>	FM signal source.
FM, MDSP	<mod wave shape>	FM modulation wave. Only when FM signal source is set to INT, you can set the parameter.
FM, FRQ	<FM frequency>	FM frequency. Only when FM signal source is set to INT, you can set the parameter.
FM, DEVI	<FM frequency deviation >	FM frequency deviation. Only when FM signal source is set to INT. you can set the parameter.
PM, SRC,	<src>	PM signal source.
PM, MDSP	<mod wave shape>	PM modulation wave. Only when PM signal source is set to INT, you can set the parameter.
PM, FRQ	<PM frequency>	PM frequency. Only when PM signal source is set to INT, you can set the parameter.
PWM, FRQ	<PWM frequency>	PWM frequency. Only when carrier wave is PULSE wave, you can set the parameter.
PWM, DEVI	<PWM dev>	Duty cycle deviation. Only when carrier wave is PULSE wave, you can set the parameter.
PWM, MDSP	<mod wave shape>	PWM modulation wave. Only when carrier wave is PULSE wave, you can set the parameter.
PWM, SRC	<src>	PWM signal source.
PM, DEVI	<PM phase offset>	PM phase deviation. Only when PM signal source is set to INT, you can set the parameter.
ASK, SRC	<src>	ASK signal source.
ASK, KFRQ	<ASK key frequency>	ASK key frequency. Only when ASK signal source is set to INT, you can set the parameter.
FSK, KFRQ	<FSK frequency>	FSK frequency. Only when FSK signal source is set to INT, you can set the parameter.
FSK, HFRQ	<FSK hop frequency>	FSK hop frequency.
FSK, SRC	<src>	FSK signal source.

CARR, WVTP	<wave type>	Carrier wave type.
CARR, FRQ	<frequency>	Value of carrier frequency.
CARR, AMP	<amplitude>	Value of carrier amplitude.
CARR, OFST	<offset>	Value of carrier offset.
CARR, SYM	<symmetry>	Value of carrier symmetry. Only ramp can set this parameter.
CARR, DUTY	<duty>	Value of duty cycle. Only square and pulse can set this parameter.
CARR, PHSE	<phase>	Value of carrier phase.
CARR, RISE	<rise>	Value of rise time. Only Pulse can set this parameter.
CARR, FALL	<fall>	Value of fall time. Only Pulse can set this parameter.
CARR, DLY	<delay>	Value of carrier delay. Only PULSE can set this parameter.

Note: If carrier wave is Noise you can't set to turn on modulation.
If you want to set AM, FM, PM, CARR and STATE the first parameter have to be one of them.

where: <state>:={ON, OFF}
<src>:= {INT, EXT}
<mod wave shape>:={SINE, SQUARE, TRIANGLE, UP RAMP, DNRAMP, NOISE, ARB}
<am frequency>:= {Default unit is "Hz". Value depends on the model.}
<depth>:= {0% to 120%}
<fm frequency>:= {Default unit is "Hz". Value depends on the model.}
<fm frequency deviation > :={ 0 to carrier frequency, Value depends on the difference between carrier frequency and bandwidth frequency.}
<pm frequency> :={ Default unit is "Hz", Value depends on the model.}
<pm phase deviation >:= {0 to 360.}
<pwm frequency>:= {Default unit is "Hz", Value depends on the model. }
<pwm dev>:= { Default unit is "%",value depends on carrier duty cycle}
<ask key frequency>:= {Default unit is "Hz", Value depends on the model.}
<fsk frequency>:= { Default unit is "Hz", Value depends on the version.}
<fsk jump frequency>:= { the same with basic wave frequency}
<wave type>:={SINE ,SQUARE, RAMP, ARB, PULSE }
<frequency> :={ Default unit is "Hz", Value depends on the model.}
<amplitude> :={ Default unit is "V", Value depends on the model.}
<offset> :={ Default unit is "V", Value depends on the model.}
<duty>:= {0% to 100 %.}
<symmetry>:={ 0% to 100%}
<phase>:={ Value depends on the model.}
<rise>:= {Value depends on the model.}
<fall>:= {Value depends on the model.}

<delay>:= {Default unit is "S".}

Note:
There are some parameters Value depends on the model, You can read version datasheet to get specific parameters

QUERY SYNTAX

<channel>: MDWV (MoDulateWaVe)?
<channel>:={C1, C2}

RESPONSE FORMAT

<channel>:MDWV <parameter>
<parameter> :={ Return all parameter of the current modulation parameters.}

EXAMPLE

Set channel one modulation type to AM.
C1: MDWV AM

Set modulation shape to AM, and set AM modulating wave type to sine wave.
C1: MDWV AM, MDSP, SINE

Read channel one modulation parameters of which STATE is ON.
C1: MDWV?
Return:
C1:MDWV STATE,ON,AM,MDSP,SINE,SRC,INT,FRQ,100HZ,
DEPTH,100,CARR,WVTP,RAMP,FRQ,1000HZ,AMP,4V,OFST,0V,PHSE,
0, SYM, 50

Read channel one modulate wave parameters of which STATE is OFF.
C1: MDWV?
Return:
C1: MDWV STATE, OFF

Set channel one FM frequency to 1000Hz
C1: MDWV FM, FRQ, 1000

Set channel one carrier shape to SINE.
C1: MDWV CARR, WVTP, SINE

Set channel one carrier frequency to 1000 Hz.
C1: MDWV CARR, FRQ,1000

RELATED COMMANDS

ARWV, BTWV, SWWV, BSWV

Note:

Parameter/command	SDG800	SDG1000	SDG2000X	SDG5000
<channel>	No(single channel)	yes	yes	yes
[type], SRC	no(only internal source)	yes	yes	yes
CARR, DLY	no	yes	yes	yes
CARR, RISE	yes	no	yes	yes
CARR, FALL	yes	n o	yes	yes

[type]:={AM, FM, PM, FSK, ASK, DSBAM, PWM}

1.8 Sweep Wave Command

DESCRIPTION Sets or gets sweep parameters.

COMMAND SYNTAX <channel>SWWV(SweepWaVe) <parameter>
<channel>:={C1, C2}
<parameter>:= {a parameter from the table below}

Parameters	Value	Description
STATE	<state>	Turn on or off sweep. Note: if you want to set or read other parameters you must set STATE to ON at first.
TIME	<time>	Value of sweep time.
STOP	<stop frequency>	Value of stop frequency.
START	<start frequency>	Value of start frequency.
TRSR	<trigger src>	Trigger source.
TRMD	<trigger mode>	State of trigger output. If TRSR is EXT, the parameter is invalid.
SWMD	<sweep mode>	Sweep style.
DIR	<direction>	Sweep direction.
EDGE	<edge>	Value of edge. Only when TRSR is EXT, the parameter is valid.
MTRIG	<manual trigger>	Make a manual trigger. Only when TRSR is MAN, the parameter is valid.
CARR, WVTP	<wave type>	Carrier type.
CARR, FRQ	<frequency>	Value of carrier frequency.
CARR, AMP	<amplitude>	Value of carrier amplitude.
CARR, OFST	<offset>	Value of carrier offset.
CARR, SYM	<symmetry>	Value of carrier symmetry, Only Ramp can set this parameter.
CARR,	<duty>	Value of carrier duty cycle. Only Square can set

DUTY		this parameter.
CARR, PHSE	<phase>	Value of carrier phase.

Note: If carrier is Pulse or Noise you can't turn on sweep.
If you want to set CARR and STATE, the first parameter has to be one of them.

where:

<state>:= {ON, OFF}
<time>:= { Default unit is "S". Value depends on the model.}
<stop frequency> :={ the same with basic wave frequency}
<start frequency> :={ the same with basic wave frequency}
<trigger src>:= {EXT, INT, MAN}
<trigger mode>:= {ON, OFF}
<sweep mod>:= {LINE, LOG}
<direction>:= {UP, DOWN}
<edge>:={RISE, FALL}
<wave type>:={SINE ,SQUARE, RAMP, ARB}
<frequency> :={ Default unit is "Hz". Value depends on the model.}
<amplitude> :={ Default unit is "V". Value depends on the model.}
<offset> :={ Default unit is "V", Value depends on the model.}
<duty>:= {0% to 100 %.}
<symmetry>:={ 0% to 100%}
<phase>:={ Value depends on the model.}

Note:
There are some parameters Value depends on the model,
You can read version datasheet.

QUERY SYNTAX

<channel>: SWWV (SWeepWaVe)? <channel>:={C1, C2}

RESPONSE FORMAT

<parameter> :={ Return all parameters of the current sweep wave.}

EXAMPLE

Set channel one sweep time to 1 S.
C1: SWWV TIME, 1

Set channel one sweep stop frequency to 1000 Hz.
C1: SWWV STOP, 1000

Read channel one sweep parameters of which STATE is ON.
C2: SWWV?
Return:

C2: SWWV STATE, ON, TIME, 1S, STOP, 100HZ, START, 100HZ, TRSR, MAN,TRMD, OFF, SWMD, LINE, DIR, UP, CARR, WVTP, SQUARE,
FRQ, 1000HZ, AMP, 4V, OFST, 0V, DUTY, 50, PHSE, 0

Read channel two sweep parameters of which STATE is OFF.
C2: SWWV?
Return:
C2: SWWV STATE, OFF

Notes:

Parameter/command	SDG800	SDG1000	SDG2000X	SDG5000
<channel>	no(single channel)	yes	yes	yes
TRMD	no	yes	yes	yes
EDGE	no	yes	yes	yes

1.9 Burst Wave Command

DESCRIPTION Sets or gets burst wave parameters.

COMMAND SYNTAX <channel>BTWV(BursTWaVe) <parameter>
<channel>:={C1, C2}
<parameter>:= {a parameter from the table below}

Parameters	Value	Description
STATE	<state>	Turn on or off burst. Note: If you want to set or read other parameters of burst, you must set state to ON at first.
PRD	<period>	Value of burst period. When carrier is NOISE wave, you can't set it. When GATE was chosen, you can't set it (but in SDG2000X, you can).And when trigger source is EXT, you can't set it.
STPS	<start phase>	Start phase of carrier. When carrier is NOISE or PULSE wave, you can't set it.
GATE_NCYC	<gate Ncycle>	Set the burst mode to GATE or NCYC. When carrier is NOISE, you can't set it.
TRSR	<trigger source>	Set the trigger source.
DLAY	<delay>	Value of delay. When carrier is NOISE wave, you can't set it. When NCYC is chosen you can set it.
PLRT	<polarity>	Value of polarity. When GATE is chosen you can set it. When carrier is NOISE, it is the only parameter.

TRMD	<trig mode>	Value of trigger mode. When carrier is NOISE wave, you can't set it. When NCYC is chosen you can set it. When TRSR is set to EXT, you can't set it.
EDGE	<edge>	Value of edge. When carrier is NOISE wave, you can't set it. When NCYC is chosen and TRSR is set to EXT, you can set it.
TIME	<circle time>	Value of Ncycle number. When carrier is NOISE wave, you can't set it. When NCYC is chosen you can set it.
MTRIG	<manual trig>	Manual trigger. When TRSR is set to MAN, it can be set.
CARR, WVTP	<wave type>	Value of carrier type.
CARR, FRQ	<frequency>	Value of carrier frequency
CARR, AMP	<amplitude>	Value of carrier amplitude.
CARR, OFST	<offset>	Value of carrier offset.
CARR, SYM	<symmetry>	Value of symmetry. Only Ramp can set this parameter.
CARR, DUTY	<duty>	Value of duty cycle. Only Square or Pulse can set this parameter.
CARR, PHSE	<phase>	Value of carrier phase.
CARR, RISE	<rise>	Value of rise edge. Only when carrier is Pulse, the Value is valid.
CARR, FALL	<fall>	Value of fall edge. Only when carrier is Pulse, the Value is valid.
CARR, STDEV	<stdev>	Value of stdev. Only when carrier is Noise, the Value is valid.
CARR, MEAN	<mean>	Value of mean. Only when carrier wave is Noise, the Value is valid.
CARR, DLY	<delay>	Value of delay. Only when carrier is Pulse, the parameter is valid

Note: If you want to set CARR and STATE, the first parameter has to one of them

where:

<state>:= {ON, OFF}
<period>:= {Default unit is "S". Value depends on the model.}
<start phase>:= {0 to 360}
<gate ncycle>:= {GATE, NCYC}
<trigger source>:= {EXT, INT, MAN}
<delay>:= {Default unit is "S", Value depends on the model.}
<polarity>:= {NEG, POS}
<trig mode >:= {RISE, FALL, OFF}

<edge>:= { RISE, FALL}
<circle time> :={ Value depends on the Model (“INF” means infinite).}
<wave type>:={SINE ,SQUARE, RAMP, PULSE, NOISE, ARB}
<frequency> :={ Default unit is "HZ". Value depends on the model.}
<amplitude>:= {Default unit is "V". Value depends on the model.}
<offset>:= {Default unit is "V". Value depends on the model.}
<duty>:= {0% to 100%.}
<symmetry> :={ 0% to 100%}
<phase>:={ Value depends on the model.}
<stdev>:= {Default unit is "V". Value depends on the model.}
<mean>:= {Default unit is "V". Value depends on the model.}
<width> :={ Max_width < (Max_duty * 0.01) * period and Min_width > (Min_duty * 0.01) * period.}
<rise>:= {Value depends on the model.}
<fall>:= {Value depends on the model.}
<delay>:= {Default unit is “S”.}

Note:
There are some parameters Value depends on the model,
You can read version datasheet to get specific parameters.

QUERY SYNTAX

<channel>: BTWV (BursTWaVe)? <parameter>
<channel>:={C1, C2}
<parameter>:=<period>……

RESPONSE FORMAT

<channel>:BTWV <type>,<state>,<period>……

EXAMPLE

Set channel one burst period to 1S.
C1: BTWV PRD, 1

Set channel one burst delay to 1s
C1: BTWV DLAY, 1

Set channel one burst to infinite
C1: BTWV TIME, INF

Read channel two burst parameters of which 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 channel two burst parameters of which STATE is OFF.
C2: BTWV?
Return:
C2: BTWV STATE, OFF

Notes:

Parameter/command	SDG800	SDG1000	SDG2000X	SDG5000
<channel>	no(single channel)	yes	yes	yes
TRMD	no	yes	yes	yes
EDGE	no	yes	yes	yes
CARR, DLY	yes	yes	yes	yes
CARR, RISE	yes	no	yes	yes
CARR, FALL	yes	no	yes	yes

1.10 Parameter Copy Command

DESCRIPTION Copies parameters from one channel to the other.

COMMAND SYNTAX PACP(ParaCoPy) <destination channel>, <src channel>
< destination channel>:= {C1, C2}
<src channel>:= {C1, C2}
Note: the parameters C1 and C2 must be set to the device together.

EXAMPLE Copy parameters from channel one to channel two.
PACP C2, C1

RELATED COMMANDS ARWV, BTWV, MDWV, SWWV, BSWV

Note:

Parameter/command	SDG800	SDG1000	SDG2000X	SDG5000
PACP	no	yes	yes	yes

1.11 Arbitrary Wave Command

DESCRIPTION Sets and gets arbitrary wave type.

COMMAND SYNTAX <channel> ARWV(ArbWaVe) INDEX,<value1>, NAME,<value2>
<channel>:={C1, C2}
< value1>: the table below shows what the index number mean.)
< value2>: see table below.

QUERY SYNTAX <channel>: ARWV (ARbWaVe)?
<channel>:={C1, C2}

RESPONSE FORMAT <channel>:ARWV <index>

EXAMPLE Set StairUp arbitrary wave output by index.
C1:ARWV INDEX, 2

Read system current wave.
ARWV?
Return:
ARWV INDEX,2,NAME,StairUp

Set Cardiac arbitrary wave output by name.
ARWV NAME, Cardiac

RELATED COMMANDS BSWV

<table>:

Index	Name	Index	Name	Index	Name	Index	Name
0	StairUp	12	Sqrt	24	Cardiac	36	Bartlett
1	StairDn	13	Root3	25	Quake	37	Tan
2	Stairud	14	X^2	26	Chirp	38	Cot
3	Ppulse	15	X^3	27	Twotone	39	Sec
4	Npulse	16	Sinc	28	Snr	40	Csc
5	Trapezia	17	Gaussian	29	Hamming	41	Asin
6	Upramp	18	Dlorentz	30	Hanning	42	Acos
7	Dnramp	19	Haversine	31	Kaiser	43	Atan
8	Exp_fall	20	Lorentz	32	Blackman	44	Acot
9	Exp_rise	21	Gauspuls	33	Gausswin		
10	Logfall	22	Gmonopuls	34	Triang		
11	Logrise	23	Tripuls	35	Harris		

About the table: This table is just an example, the index may depend on the model. You can execute “STL?” command to get them accurately.

Notes:

Parameter/command	SDG800	SDG1000	SDG2000X	SDG5000
<channel>	no(single channel)	yes	yes	yes

INDEX	yes	yes	yes(only built-in wave)	yes
NAME	yes	yes	yes(user define wave)	yes

1.12 Sync Command

DESCRIPTION Sets synchronization signal.

COMMAND SYNTAX <channel>: SYNC <parameter>
<channel>:={C1, C2}
<parameter>:= {ON, OFF}

QUERY SYNTAX <channel>: SYNC?
<channel>:={C1, C2}

RESPONSE FORMAT <channel>:SYNC <parameter>

EXAMPLE Turn on sync function of channel one.
C1: SYNC ON

Read state of channel one sync.
C1: SYNC?
Return:
C1: SYNC OFF

Note:

Parameter/command	SDG800	SDG1000	SDG2000X	SDG5000
SYNC	no	yes	yes	yes

1.13 Number Format Commend

DESCRIPTION Sets or gets number format.

COMMAND SYNTAX NBFM(NumBer_ForMat) <parameter>
<parameter> :={ a parameter from the table below.}

Parameters	Value	Description
PNT	<pnt>	Point format
SEPT	<sept>	Separator format

Where:
<pnt>:= {Dot, Comma}.
<sept> :={ Space, Off, On}.

QUERY SYNTAX NBFM (NumBer_ForMat)?

RESPONSE FORMAT NBFM <parameter>

EXAMPLE Set point format to DOT.
NBFM PNT, DOT

Set Separator format to ON.
NBFM SEPT,ON

Read number format.
NBFM?
Return:
NBFM PNT, DOT, SEPT, ON

1.14 Language Command

DESCRIPTION Sets or gets system language.

COMMAND SYNTAX LAGG(LAnGuaGe) <parameter>
<parameter>:={EN, CH, RU}

QUERY SYNTAX LAGG (LAnGuaGe)?

RESPONSE FORMAT LAGG <parameter>

EXAMPLE Set language to English.
LAGG EN

Read language
LAGG?
Return:
LAGG EN

Note:

Parameter/command	SDG800	SDG1000	SDG2000X	SDG5000
RU	no	yes	no	no

1.15 Configuration Command

DESCRIPTION Sets or gets the power-on system setting..

COMMAND SYNTAX	SCFG(Sys_CFG)<parameter> <parameter>:= {DEFAULT, LAST}
QUERY SYNTAX	SCFG (Sys_CFG)?
RESPONSE FORMAT	SCFG <parameter>
EXAMPLE	Set the power-on system setting to LAST. SCFG LAST

1.16 Buzzer Command

DESCRIPTION	Turns on or off the buzzer.
COMMAND SYNTAX	BUZZ(BUZZer) <parameter> <parameter>:= {ON, OFF}
QUERY SYNTAX	BUZZ (BUZZer)?
RESPONSE FORMAT	BUZZ <parameter>
EXAMPLE	Turn on the buzzer. BUZZ ON

1.17 Screen Save Command

DESCRIPTION	Turns off or sets screen save time (default unit is minutes).
COMMAND SYNTAX	SCSV (SCreen_SaVe) <parameter> <parameter>:= {OFF, 1, 5, 15, 30, 60, 120, 300 }
QUERY SYNTAX	SCSV (SCreen_SaVe)?
RESPONSE FORMAT	SCreen_SaVe <parameter>
EXAMPLE	Set screen save time to 5 minutes. SCSV 5 Read the current screen save time. SCreen_SaVe?

Return:
SCSV 5MIN

1.18 Clock Source Command

DESCRIPTION Sets or gets the clock source.

COMMAND SYNTAX ROSC (ROSCillator) <parameter>
<parameter>:= {INT, EXT}

QUERY SYNTAX ROSC (ROSCillator)?

RESPONSE FORMAT ROSC <parameter>

EXAMPLE Set internal time base as the clock source.
ROSC INT

Note:

Parameter/command	SDG800	SDG1000	SDG2000X	SDG5000
ROSC	no	yes	yes	yes

1.19 Frequency Counter Command

DESCRIPTION Sets or gets frequency counter parameters.

COMMAND SYNTAX FCNT(FreqCouNTer) <parameter>
<parameter>:= {a parameter from the table below}

Parameters	Value	Description
STATE	<state>	State of frequency counter.
FRQ	<frequency>	Value of frequency. Can't be set.
PW	<position width>	Value of positive width. Can't be set.
NW	<negative width>	Value of negative width. Can't be set.
DUTY	<duty>	Value of duty cycle. Can't be set.
FRQDEV	<freq deviation>	Value of freq deviation. Can't be set.
REFQ	<ref freq>	Value of reference freq.
TRG	<triglev>	Value of trigger level.
MODE	<mode>	Value of mode.
HFR	<HFR>	State of HFR.

where: < state >:={ON, OFF}
<frequency>:= {Default unit is "Hz". Value range depends on the model.}

< mode >:={AC, DC}
<HFR>:={ON, OFF}

QUERY SYNTAX FCNT (FreqCouNTer)?

RESPONSE FORMAT FCNT < state ><frequency><duty>
<ref freq><triglev><position width><negative width>
<freq deviation><mode><HFR>

EXAMPLE

Turn on frequency counter:
FCNT STATE,ON

Query frequency counter information:
FCNT?
Return:
FCNT
STATE,ON,FRQ,0HZ,DUTY,0,REFQ,1e+07HZ,TRG,0V,PW,0
S,NW,0S,FRQDEV,-1e+06ppm,MODE,AC,HFR,OFF

Note:

Parameter/command	SDG800	SDG1000	SDG2000X	SDG5000
FCNT	no	yes	yes	yes

1.20 Invert Command

DESCRIPTION Sets or gets polarity of current channel.

COMMAND SYNTAX <channel>:INVT(INVerT) <parameter>
<channel>:={C1, C2}
<parameter>:= {ON, OFF}

QUERY SYNTAX <channel>: INVT (INVerT)?
<channel>:={C1, C2}

RESPONSE FORMAT <channel>:INVerT <parameter>

EXAMPLE

Set C1 ON:
C1: INVT ON

Read the polarity of channel one.
C1: INVT?

Return:
C1: INVT ON

Note:
1.

Parameter/command	SDG800	SDG1000	SDG2000X	SDG5000
<channel>	no(single channel)	yes	yes	yes

2. The <channel> is a selectable parameter. If channel is not set, default is current channel.

1.21 Coupling Command

DESCRIPTION Sets or gets channel coupling parameters.

COMMAND SYNTAX COUP (COUPling)<parameter>
<parameter>:= {a parameter from the table below}

Parameters	Value	Description
STATE	<state>	State of channel coupling.
BSCH	<bsch>	Value of bsch.
FDEV	<frq_dev>	Value of frq_dev.
PDEV	<pha_dev>	Value of position pha_dev.

where: < state >:={ON, OFF}
< bsch >:= {CH1, CH2}
< frq_dev >:={ Default unit is “Hz”}
< pha_dev >:={ Default unit is “° ”}

QUERY SYNTAX COUP (COUPling)?

RESPONSE FORMAT COUP < state >,< bsch >,< frq_dev >< pha_dev >

Note:

Parameter/command	SDG800	SDG1000	SDG2000X	SDG5000
Coupling	no	yes	no	no

1.22 Voltage Overload Command

DESCRIPTION Sets or gets state of over-voltage protection.

COMMAND SYNTAX VOLTPRT<parameter>

<parameter>:= {ON, OFF}

QUERY SYNTAX VOLTPRT?

RESPONSE FORMAT VOLTPRT<parameter>

Note:

Parameter/command	SDG800	SDG1000	SDG2000X	SDG5000
VOLTPRT／ command	no	yes	yes	no

1.23 Store list command

DESCRIPTION This command is used to read the stored wave data names. If the store unit is empty, the command will return “EMPTY” string.

Note: M50~ M59 is user defined memory. The name will return what you defined. If you do not define an arbitrary name, it will return “EMPTY”(It depends on the model).

QUERY SYNTAX STL (StoreList)? BUILDIN, USER

EXAMPLE Read all arbitrary data saved in the device.
STL?
Return:
STL M0, StairUp, M1, StairDn, M2, StairUD, M3, Trapezia, M4, ExpFall, M5, ExpRise, M6, LogFall, M7, LogRise, M8, Sqrt, M9, X^2, M10, Sinc, M11, Gaussian, M12, Dlorentz, M13, Haversine, M14, Lorentz, M15, Gauspuls, M16, Gmonopuls, M17, Cardiac, M18, Quake, M19, TwoTone, M20, SNR, M21, Hamming, M22, Hanning, M23, Kaiser, M24, Blackman, M25, GaussiWin, M26, Harris, M27, Bartlett, M28, Tan, M29, Cot, M30, Sec, M31, Csc, M32, Asin, M33, Acos, M34, Atan, M35, ACot, M36, EMPTY, M37 …..

Read built-in wave data.
STL? BUILDIN
Return:
STL M0, Sine, M1, Noise, M10, ExpFal, M11, ExpRise, M12, LogFall, M13, LogRise, M14, Sqrt, M15, Root3, M16, X^2, M17, X^3, M18, Sinc, M19, Gussian, M2, StairUp, M20, Dlorentz, M21, Haversine, M22, Lorentz, M23, Gauspuls, M24, Gmonopuls, M25, Tripuls, M26, Cardiac, M27, Quake,

M28, Chirp, M29, Twotone, M3, StairDn, M30, SNR, M31, Hamming, M32, Hanning, M33, kaiser, M34, Blackman, M35, Gausswin, M36, Triang, M37, Harris, M38, Bartlett, M39, Tan, M4, StairUD, M40, Cot, M41, Sec, M42, Csc, M43, Asin, M44, Acos, M45, Atan, M46, Acot, M47, Square, M5, Ppulse, M6, Npulse, M7, Trapezia, M8, Upramp, M9, Dnramp

Read wave data defined by user.
STL? USER
Return:
STL
WVNM,sinec_8M,sinec_3000000,sinec_1664000,ramp_8M, sinec_2000000,sinec_50000,square_8M,sinec_5000,wave1, square_1M

Notes:

Parameter/command	SDG800	SDG1000	SDG2000X	SDG5000
BUILDIN	no	no	yes(get built-in wave)	no
USER	No	no	yes(get user defined wave)	no

1.24 Virtual key command

DESCRIPTION

The Command is used to send a simulate operation of pressing key on front panel.

COMMAND SYNTAX

VKEY (VirtualKEY) VALUE,<value>,STATE,<sate>
<value>:= {a parameter from the table below.}
<state>:=<0,1>(“1” is effective to virtual value, and “0” is useless)

EXAMPLE

VKEY VALUE,15, STATE,1
VKEY VALUE,KB_SWEEP, STATE,1

Note:
<table>

KB_FUNC1	28	KB_NUMBER_4	52
KB_FUNC2	23	KB_NUMBER_5	53
KB_FUNC3	18	KB_NUMBER_6	54
KB_FUNC4	13	KB_NUMBER_7	55
KB_FUNC5	8	KB_NUMBER_8	56
KB_FUNC6	3	KB_NUMBER_9	57
KB_MOD	15	KB_POINT	46
KB_SWEEP	16	KB_NEGATIVE	43

KB_BURST	17	KB_LEFT	44
KB_WAVES	4	KB_RIGHT	40
KB_UTILITY	11	KB_OUTPUT1	153
KB_PARAMETER	5	KB_OUTPUT2	152
KB_STORE_RECALL	70	KB_KNOB_RIGHT	175
KB_NUMBER_0	48	KB_KNOB_LEFT	177
KB_NUMBER_1	49	KB_KNOB_DOWN	176
KB_NUMBER_2	50	KB_HELP	12
KB_NUMBER_3	51	KB_CHANNEL	72

Note:

Parameter/command	SDG800	SDG1000	SDG2000X	SDG5000
KB_STORE_RECALL	yes	yes	yes	no
KB_HELP	yes	yes	no	no
KB_CHANNEL	no(single channel)	yes	yes	no

1.25 IP Command

DESCRIPTION The Command can set and get system IP address.

COMMAND SYNTAX SYST:COMM:LAN:IPAD
(SYSTem:COMMunicate:LAN:IPADdress)
<parameter1>.<parameter2>.<parameter3>.<parameter4>

<parameter1>:={a integer value between 1 and 223}
<parameter2>:={a integer value between 0 and 255}
<parameter3>:={a integer value between 0 and 255}
<parameter4>:={a integer value between 0 and 255}

QUERY SYNTAX SYST:COMM:LAN:IPAD
(SYSTem:COMMunicate:LAN:IPADdress)?

EXAMPLES Set IP address to 10.11.13.203
SYSTem: COMMunicate: LAN:IPADdress 10.11.13.203

Get IP address.
SYST:COMM:LAN:IPAD?
Return:
“10.11.13.203”

Note:

Parameter/command	SDG800	SDG1000	SDG2000X	SDG5000
SYST:COMM:LAN:IPAD	no	no	yes	no

1.26 Subnet Mask Command

DESCRIPTION The Command can set and get system subnet mask.

COMMAND SYNTAX SYST:COMM:LAN:SMAS (SYSTem:COMMunicate:LAN:SMASk)
<parameter1>.<parameter2>.<parameter3>.<parameter4>

<parameter1>:={a integer value between 0 and 255}
<parameter2>:={a integer value between 0 and 255}
<parameter3>:={a integer value between 0 and 255}
<parameter4>:={a integer value between 0 and 255}

QUERY SYNTAX SYSTem:COMMunicate:LAN:SMASk?

EXAMPLES Set subnet mask to 255.0.0.0
SYSTem:COMMunicate:LAN:SMASk 255.0.0.0

Get subnet mask
SYSTem:COMMunicate:LAN:SMASk?
Return:
"255.0.0.0"

Note:

Parameter/command	SDG800	SDG1000	SDG2000X	SDG5000
SYST:COMM:LAN:SMAS	no	no	yes	no

1.27 Gateway Command

DESCRIPTION The Command can set and get system Gateway.

COMMAND SYNTAX SYST:COMM:LAN:GAT(SYSTem:COMMunicate:LAN:GATeway)
<parameter1>.<parameter2>.<parameter3>.<parameter4>

<parameter1>:={a integer value between 0 and 223}
<parameter2>:={a integer value between 0 and 255}
<parameter3>:={a integer value between 0 and 255}
<parameter4>:={a integer value between 0 and 255}

QUERY SYNTAX SYSTem:COMMunicate:LAN:GATeway?

EXAMPLES Set Gateway to 10.11.13.5:
SYSTem:COMMunicate:LAN:GATeway 10.11.13.5

Get gateway:
SYSTem:COMMunicate:LAN:GATeway?
Return:
"10.11.13.5"

Note:

Parameter/command	SDG800	SDG1000	SDG2000X	SDG5000
SYST:COMM:LAN:GAT	no	no	yes	no

1.28 Sampling Rate Command

DESCRIPTION Sets or gets arb mode and sampling rate.
Note:
You can set or get sampling rate only in TrueArb mode.

COMMAND SYNTAX

<channel>:SRATE(SampleRATE) MODE <parameter1>, VALUE, <parameter2>
<channel> :=<C1, C2>
<parameter1> :=< DDS, TARB>
<parameter2> :={ a integer value between 1e-6 and 75000000, (default unit is Sa/s)}

QUERY SYNTAX <channel>: SRATE?

EXAMPLES Get the channel one sample rate value
C1: SRATE?
Return:
C1: SRATE MODE, DDS
Set channel one to TureArb mode.
C1: SRATE MODE, TARB

Set channel one sample rate value to 1000000Sa/s.
C1: SRATE VALUE, 1000000

Note:

Parameter/command	SDG800	SDG1000	SDG2000X	SDG5000
<channel>	No(single channel)	yes	yes	yes
SRATE	no	no	yes	no

1.29 Index

*IDN
OPC
*CLS
*ESE
*ESR
*RST
*SRE
*STB
*TST
*WAI
DDR
CMR

A
ARWV ARBWAVE

B
BSWV BASIC_WAVE
BTWV BURSTWAVE
BUZZ BUZZER

C
CHDR COMM_HEADER
COUP COUPLING

F
FCNT FREQCOUNTER

I
IVNT INVERT

L
LAGG LANGUAGE

M
MDWV MODULATEWAVE

N
NBFM NUMBER_FORMAT

O

OUTP OUTPUT

P

PACP PARACOPY

R

ROSC ROSCILLATOR

S

SCFG Sys_CFG
SCSV SCREEN_SAVE
SWWV SWEEPWAVE
SYNC SYNC
STL STORELIST
SYST:COMM:LAN:IPAD SYSTEM:COMMUNICATE:LAN:IPADDRESS
SYST:COMM:LAN:SMAS SYSTem:COMMunicate:LAN:SMASk
SYST: COMM: LAN:GAT SYSTem:COMMunicate:LAN:GATeway
SRATE SAMPLERATE

V

VKEY VIRTUALKEY

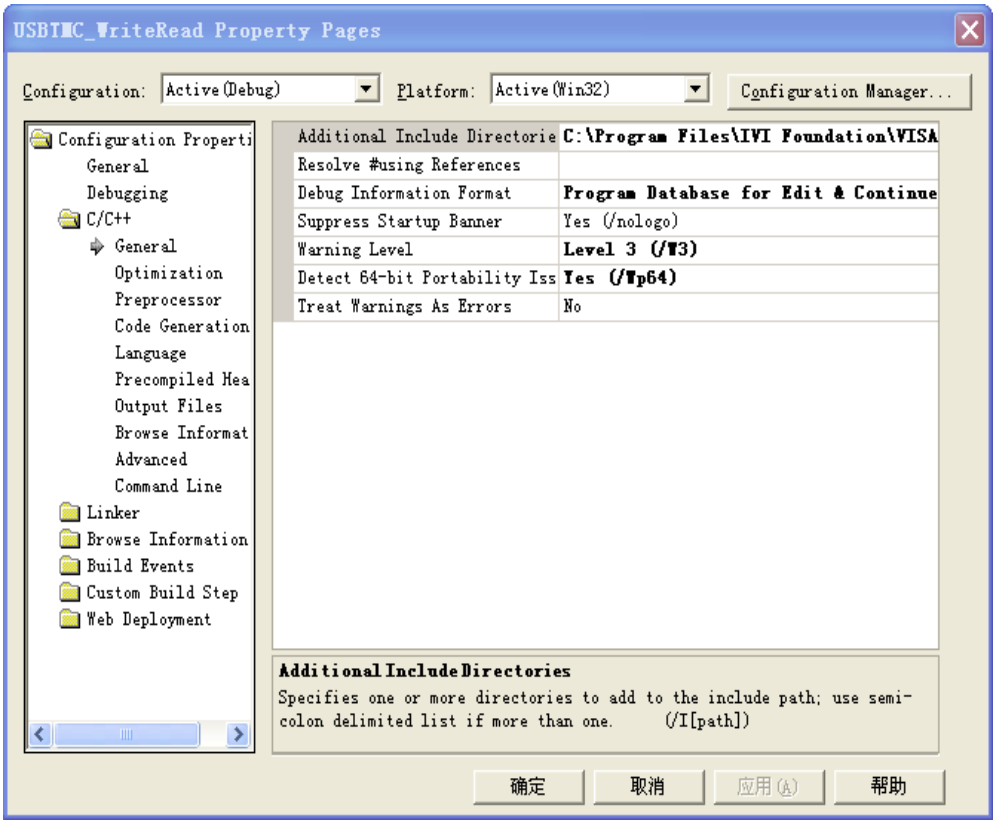
2 Programming Demos

2.1 Visual C++ Programming Demo

The program used in this demo: Microsoft Visual Studio 2003
The functions realized in this demo: use the NI-VISA to control the device with USBTMC or TCP/IP access to do a write or read operation.

1. Open Visual Studio: create a new VC++ win32 console project.
2. Set the project environment to use the NI-VISA lib, there are two ways to use the NI-VISA: static way and automatic way.
   2.1 static way:
   Find visa.h, visatype.h and visa32.lib files in NI-VISA install path. Copy them to your project, and add them into the project. In C++ source file，add the following two lines:
   #include "visa.h"
   #pragma comment (lib,"visa32.lib")
   2.2 automatic way:
   Let the head file include the NI-VISA install path. We can set the path

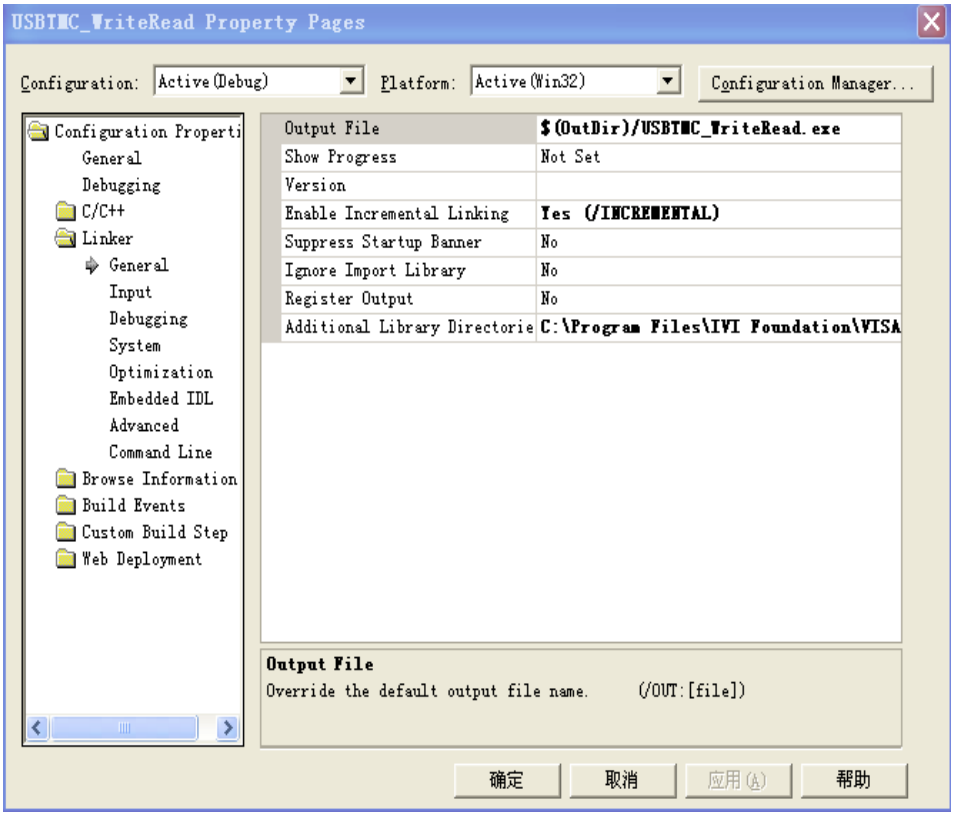
like: C:\Program Files\IVI Foundation\VISA\WinNT\include. Set this path to Project---Properties---C/C++---General---Additional Include Directories. See the picture below.



Set the lib path and file:

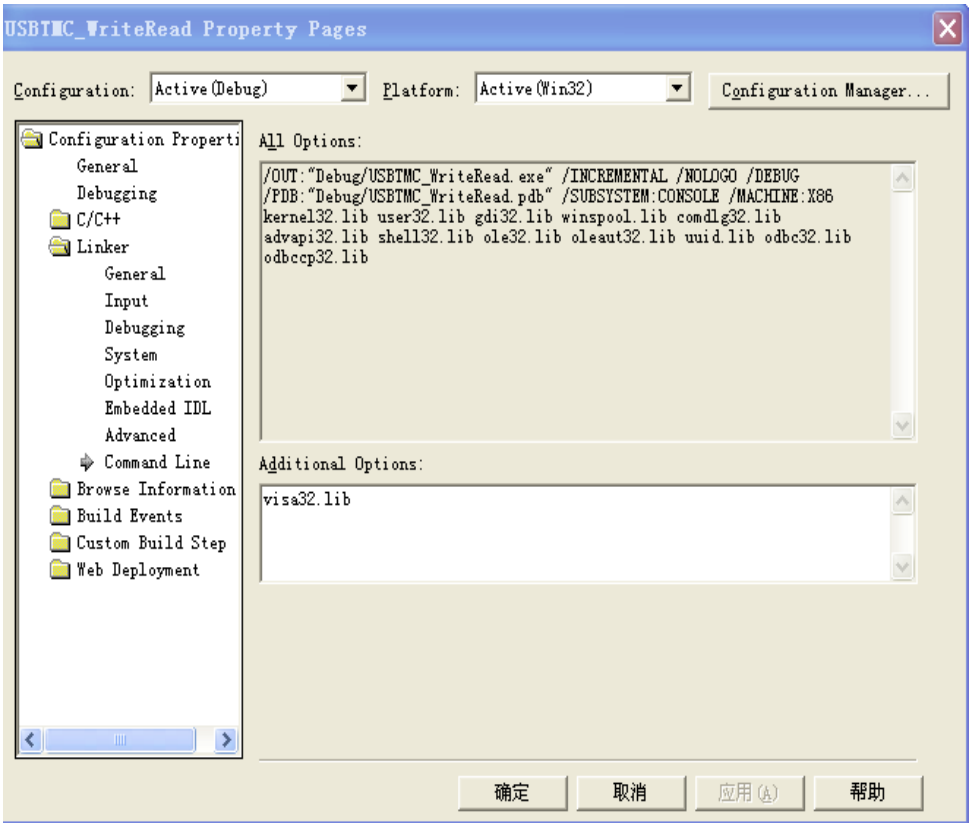
1) Set the lib path:

We can set the path: C:\Program Files\IVI Foundation\VISA\WinNT\lib\msc. The path is your NI-VISA install path. Set this path to Project---Properties---linker---general---Additional Library Directories. See the picture below.



2) Set the lib file:

Project---Properties---Linker---Command Line---Additional Options: visa32.lib. See the picture below.



Add the visa.h file into the project source file (xxx.cpp):

#include <visa.h>

3. Add your codes:

3.1 Include the header files:

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

//#include <visa.h>      // automatic way

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
```

3.2 USBTMC access code:

Write a function Usbtmc_test().

```
int Usbtmc_test()
{
        ViSession defaultRM;
        ViSession instr;
        ViUInt32 numInstrs;
        ViFindList findList;
        ViUInt32 retCount;
        ViUInt32 writeCount;
        ViStatus status;
        char instrResourceString[VI_FIND_BUFLEN];
        unsigned char buffer[100];
        char stringinput[512];
        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 USBTMC 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.\nHit enter to
        continue.");
        fflush(stdin);
        getchar();
        viClose (defaultRM);
        return status;
}

for (i=0; i<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 have opened a session to the USBTMC
***instrument. We will use the viPrintf function to send the ***device
the string "*IDN?", 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,retCount, buffer);
                }
            status = viClose (instr);
        }
        /** Now we will close the session to the instrument using
        * viClose. This operation frees all the system resources.     */
        status = viClose (defaultRM);
        return 0;
}
```

3.3 TCP/IP access code:

Write a function TCP_IP_Test(char* pIP).

```
int TCP_IP_Test (char*  pIP)
{
        char outputBuffer[VI_FIND_BUFLEN];
        ViSession defaultRM, instr;
        ViStatus status;
        ViUInt32 count;
        ViUInt16 portNo;
        /* 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";
        char resource [256];
        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 ("\ndata read from device: %*s\n", 0,outputBuffer);
        status = viClose (instr);
        status = viClose (defaultRM);
        return 0;
    }
```

3.4 main access code:

```
    int _tmain(int argc, _TCHAR* argv[])
    {
        printf ("please select the connect type ,input 1 for usbtmc,input 2 for
        tcp/ip: ");
        fflush (stdin);
        int inputchar =getchar();

          if(inputchar ==49)
          {
                Usbtmc_test();
          }else if(inputchar ==50)
          {
                printf("please input tcp/ip address:");
                char ip[256];
                fflush(stdin);
                gets(ip);
                TCP_IP_Test(ip);
          }
          printf("Hit enter to continue.");
          fflush(stdin);
          getchar();
          return 0;
     }
```

4 Run the demo and get result:

Save, compile and run the project. When the instrument connects to the PC correctly, you can choose USBTMC(1) or TCP/IP(2) to connect the instrument.