USB interface for LabView

1. SCPI command language

SCPI (Standard Commands for Programmable Instruments) is a standardized set of commands and be based on ASCII code，through the remote interface to programmer control the instruments.

1.1 Command format: The SCPI adopt the following format:

[SOURce{1|2}]:FREQuency{<Value>[MHz|kHz|Hz|mHz]|MINimum|MAXimum}

Brace { } contains the parameter options of command string.

Separating character|: separate several parameter options, only one parameter option could be chosen at one time.

Angular bracket < > indicates this option is a parameter value

Square bracket [ ]: parameters contains in this character is optional and could be omitted.

The sign { }, [ ], < > and | are shown for convenient expression in example, but not sent with command string and not allowed in practical applications.

1.2 Command abbreviation: command for instrument is either can be used in abbreviation format which is simple and brief for program writing, or be used in full format which is clear for program meaning and easy to read. The commands given by this guide is full format, among which written by uppercase stands for abbreviation format, abbreviation format is shorter than 4 letters. The other format besides that will result in mistake.

The command will not distinguish uppercase and lowercase, and it allows both uppercase and lowercase, or mixing uppercase and lowercase is also allowed. For example, “FREQ 1KHZ”, “freq 1khz” and “Freq 1kHz”are all acceptable with same running result. But for unit, letter M is different from letter m, and users can’t mix-use. For example, 1 mHz and 1MHz are both acceptable but with different running result.

1.3 Command separator: with hierarchical structure, SCPI command can be divided into Root, Subnode and Endnode commands. Using colon “:” to separate the keywords, and separate command keyword and parameters with space, if the command contains

many parameters, you can use comma “,” to separate.

For example, Apply:Sin 1 kHz, 5.2 Vpp, -0.2Vdc

Use semicolon “;” to link several commands with same scale and under one subsystem, in this case the higher scale command could be omitted and the program becomes simpler.

For example: AM:INTernal:FREQuency 3kHz

AM:INTernal:FUNCtion SINusoid

Above two commands could be linked with semicolon “;” and are simplified as one command as:

AM:INTernal:FREQuency 3kHz;FUNCtion SINusoid

Use a semicolon and a colon “;:” to link several commands under different subsystems, every command should start with root command.

For example: AM:STATe ON;:FREQuency 100kHz;:AM:DEPTh?

1.4 Parameter type: parameter type has 4 formats as following.

1.4.1 Numeric value parameter: numeric value parameter is presented by decimal number, composed by digits, minus and decimal point, such as -8.253. Floating-point number could be used to indicate either, such as 1.0E+06. You also can use two special values Minimum and Maximum to instead parameter value of command. Min set the parameter to be the allowed minimum value, and Max set the parameter to be the allowed maximum value. You can add the unit in the end of parameter value, such as kHz,Vrms and so on. If not adding, you can use the basic unit Hz,V,s and so on. The unit omitting parameter can make the program simple, but sometimes adding unit can be more convenient, for example, Freq 10MHz is more simple than 10000000.

Command of value parameter, for example, Frequency 1000, or Amplitude Max.

1.4.2 Discrete parameter: discrete parameter only has a few of value, and be same as commands, you can use full or abbreviation format, or mixing uppercase and lowercase is also allowed.

The command of discrete parameter such as FSKey:Source Internal.

1.4.3 Boolean parameter: a Boolean parameter specifies a single binary condition which is either true or false. For “True”, the parameter value is “ON” or “1”, and for “False”, the parameter value is “OFF” or “0”.

The command of Boolean parameter such as FM: State On.

1.4.4 Character string parameter: this parameter is constituted by ASCII characters, and enclosed by a pair of quotation marks. Such as, “No error”.

1.5 Parameter query: users can add interrogation “?” in the end of commands, so you can query the current value of most parameters. For example, the present CHA is sine wave, Freq 1 kHz, output port is open.

Sending query Source: Function?, can return discrete parameter “SIN”, it indicates that the current function is sine wave. For discrete parameter, both the query and return are abbreviation format with the uppercase.

Sending query Source: Frequency?, can return value parameter “1.000000E+03”, it indicates that the current frequency is 1kHz. For value parameter, basic unit is default when query or return a value without a unit with it, the format of the value is floating-point.

Sending query ‘Output?’, can return Boolean parameter “1”. It indicates that the current state of output port is open. For Boolean parameter, query or return “1”or“0”.

Sending query System: Error?, can return the character string parameter “No error”, it indicates that no error happened now. For character string parameter, query or return a character string enclosed in a pair of quotation marks.

1.6 Universal command: universal commands start with *, with length of three characters, and could have parameters.

Example of universal command:*RST

1.7 Command end character: total characters in a command string should not be more than 60. Each end of character string should be added an end character (shift character of ASCII code 10), indicates an end of character string in order to avoid a mistake. It is suggested that the end character is written in the sending function when programmable so that it is not necessary for adding it at the end of each command and it will never be lost carelessly.

2. Command set

The instrument set the SCPI command of most functions, but since the parameter calibration function is complex and remote control is not needed, so the programmer commands are not set.

The command keyword in the command set is written in full format, the uppercase is the abbreviation of this command, the unit of value could not be omitted, and the names of waveforms have only abbreviation format.

2.1 Direct configuration command

[SOURce]:APPLy:SINusoid [<frequency>[,<amplitude>[,<offset>]]]

[SOURce]:APPLy:SQUare [<frequency >[,< amplitude >[,< offset >]]]

[SOURce]:APPLy:RAMP [<frequency >[,< amplitude >[,< offset >]]]

[SOURce]:APPLy:NOISE [<frequency >[,< amplitude >[,< offset >]]]

[SOURce]:APPLy:PPULS [<frequency >[,< amplitude >[,< offset >]]]

[SOURce]:APPLy:NPULS [<frequency >[,< amplitude >[,< offset >]]]

[SOURce]:APPLy:STAIR [<frequency >[,< amplitude >[,< offset >]]]

[SOURce]:APPLy:HSINE [<frequency >[,< amplitude >[,< offset >]]]

[SOURce]:APPLy:LSINE [<frequency >[,< amplitude >[,< offset >]]]

[SOURce]:APPLy:REXP [<frequency >[,< amplitude >[,< offset >]]]

[SOURce]:APPLy:RLOG [<frequency >[,< amplitude >[,< offset >]]]

[SOURce]:APPLy:TANG [<frequency >[,< amplitude >[,< offset >]]]

[SOURce]:APPLy:SINC [<frequency >[,< amplitude >[,< offset >]]]

[SOURce]:APPLy:ROUND[<frequency >[,< amplitude >[,< offset >]]]

[SOURce]:APPLy:CARD [<frequency >[,< amplitude >[,< offset >]]]

[SOURce]:APPLy:QUAKE [<frequency >[,< amplitude >[,< offset >]]]

[SOURce]:APPLy?

Users may use Apply command to configure directly the four parameters of the function generator: function, frequency, amplitude and offset, the sequence of which could not be changed. For example:

Configuration continuously outputs sine, frequency 10kHz, amplitude 1.2Vpp, offset 0.5Vdc, the command of which is as follows:

SOURce:Apply:Sin 10kHz, 1.2, 0.5

Where frequency, amplitude and offset three parameters could be omitted from the last to the first, the omitted parameters keeps as the current value. Such as:

SOURce:Apply:Sin 10kHz, 1.2 set frequency and amplitude, offset is omitted.

SOURce:Apply:Sin 10kHz set frequency amplitude and offset are omitted.

SOURce:Apply:Sin frequency, amplitude and offset are omitted.

Use query command Source:Apply? to return current configured function, frequency, amplitude and offset value. Such as: “SIN,1.000000E+04, 1.200000E+00, 0.500000E+00”.

Use Apply command to configure directly and conveniently four parameters. Single command is more applicable when changing specifically parameters.

2.2 Output configuration command

[SOURce]:FUNCtion{SINusoid|SQUare|RAMP|NOISe|PPULS|NPULS|STAIR
|HSINE|LSINE|REXP|RLOG|TANG|SINC|ROUND|CARD|QUAKE}

[SOURce]:FUNCtion? /* Output function

[SOURce]:FUNCtion:SQUare:DCYCle{<Value %>|MINimum|MAXimum}

[SOURce]:FUNCtion:SQUare:DCYCle? /* Square duty cycle

[SOURce]:FUNCtion:RAMP:SYMMetry{<Value %>|MINimum|MAXimum}

[SOURce]:FUNCtion:RAMP:SYMMetry? /* Ramp symmetry

[SOURce]:FREQuency[:CW]{<Value MHz|kHz|Hz|mHz>|MINimum|MAXimum}

[SOURce]:FREQuency[:CW]? /* Frequency(CW means continuous waveform)

[SOURce]:PERiod[:CW]{<Value s|ms>|MINimum|MAXimum}

[SOURce]:PERiod[:CW]? /* Period

[SOURce]:VOLTage[:AMPLitude]{<Value Vrms|mVrms|Vpp|mVpp>
|MINimum|MAXimum}

[SOURce]:VOLTage[:AMPLitude]? /* Amplitude

[SOURce]:VOLTage:OFFSet{<Value Vdc|mVdc>|MINimum|MAXimum}

[SOURce]:VOLTage:OFFSet? /* DC offset

[SOURce]:VOLTage:ATTenuation:{<Value dB>|MINimum|MAXimum|AUTO}

[SOURce]:VOLTage:ATTenuation? /* Amplitude attenuation

[SOURce]:VOLTage:UNIT{Vpp|Vrms}

[SOURce]:VOLTage:UNIT? /* Amplitude unit

OUTPut:POLarity{NORMal|INVerted}

OUTPut:POLarity? /* Output polarity

OUTPut[:STATe]{ON|OFF}

OUTPut[:STATe]? /* Output state

2.3 Frequency sweeping command

FREQuency:STARt{<Value MHz|kHz|Hz|mHz>|MINimum|MAXimum}

FREQuency:STARt? /* start frequency

FREQuency:STOP{<Value MHz|kHz|Hz|mHz>|MINimum|MAXimum}

FREQuency:STOP? /* end frequency

SWEep:SPACing{LINear|LOGarithmic}

SWEep:SPACing? /* sweeping mode（step interval）

SWEep:TIMe{<Value s|ms>|MINimum|MAXimum}

SWEep:TIMe? /* sweeping time

SWEep:STATe{ON|OFF}

SWEep:STATe? /* sweeping state

2.4 System command

RST / system reset

CLS / clear error queue

SYSTem:ERRor? /* query error queue

SYSTem:LOCal /* return to local

2. Interface Operation

Here, we assume that you have installed LabVIEW in your computer and you have had some knowledge about it.

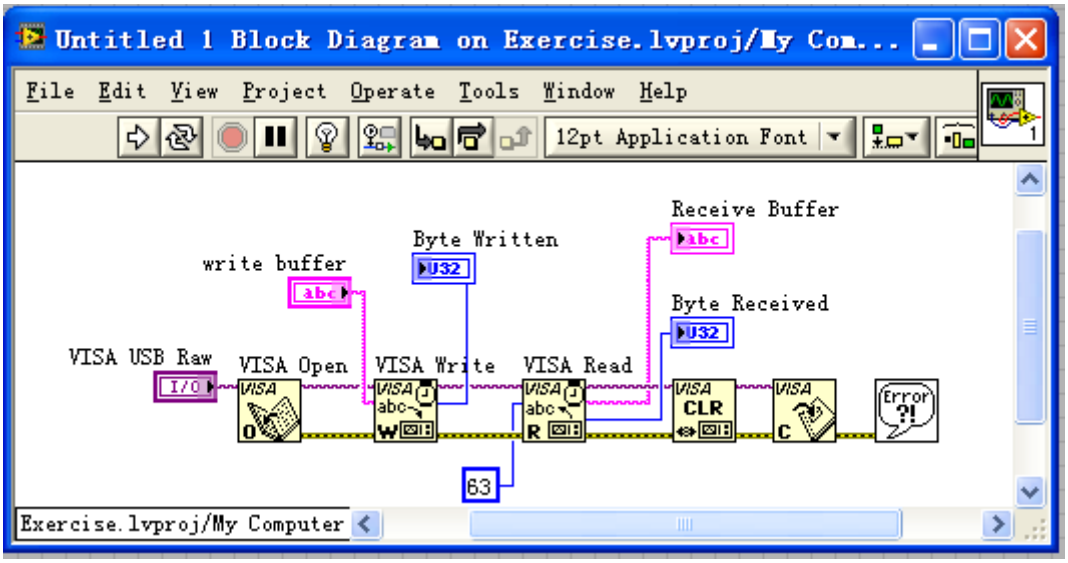
2.1 VISA Background

VISA is a high-level API used to communicate with instrumentation buses. It is platform independent, bus independent, and environment independent. In other words, the same API is used regardless of whether a program is created to communicate with a USB device with LabVIEW on a machine running Windows 2000/XP or with a GPIB device with C on a machine running Mac OS X.

2.2 Using NI-VISA to Communicate with Your USB Instrument

This instrument uses the NI-VISA USB RAW class. It uses 488.2 style communication and you can simply use the VISA Open, VISA Close, VISA Read, and VISA Write functions in the same way you would if you were communicating with GPIB instruments.

Figure 1 illustrates a LabVIEW VI that communicates with our instrument. In this example, a VISA session is opened to our USB instrument. A command is written to the device, and the response is read back. After all communication is complete, the buffer is cleared and the VISA session is closed.



To help you build your own system with our instrument in LabVIEW,we have an simple example for you. You can copy the TFG1900B directory to your instr.lib directory,for example C:\Program Files\National Instruments\LabVIEW 2009\instr.lib, and open the project in the TFG1900B_Demo directory which we have given to you for reference.

2.3 Configuring NI-VISA to Control Your USB Instrument

At this point, NI-VISA already should be installed on your computer, and your USB device should not be connected. Furthermore, you should not have a driver for your USB device installed. There are three steps to configure your USB device to use NI-VISA:

1. Create the INF file using the Driver Development Wizard.
2. Install the INF file and the USB device using the INF file.
3. Test the device with NI-VISA Interactive Control.

For more detail information, please visit the <<USB Instrument Control Tutorial>> in http://zone.ni.com/devzone/cda/tut/p/id/4478.

Note: Here, we have provided the INF file for your convenience, and you need not to create it yourself now. But, of course, you need to install it all the same.

If you are not very sure about how to use the VISA to configure your USB instrument, we strongly suggest that you visit the NI website above to configure your USB instrument step by step.

2.4 Enter program

2.4.1 Install the INF files and the USB device: The installation of the INF files is different for each version of Windows. For detailed information about installing the INF file, open the INF file in a text editor and follow the instructions at the top of the file. This tutorial assumes you are using Windows XP.

1. Copy the INF file to the INF folder. On Windows XP, this folder is usually at C:\WINDOWS\INF. This folder may be hidden, so you may need to change your folder options to view hidden files.
2. Right-click on the INF file in C:\WINDOWS\INF and click Install. This process creates a PNF file for your device. You are now ready to install your USB device.

Connect your USB device. Because USB is hot pluggable, Windows should be able to

detect your USB device, and the Add New Hardware Wizard should open automatically as soon as you connect it to the USB port. Follow the onscreen instructions for the wizard. **When you are prompted to select a driver for this device, browse to the INF folder and select the INF file we provided**.

Note: In some cases, Windows may already have a default driver associated with your USB device. If this is the case, Windows will look to install that driver first. Once you've plugged in your USB device and Windows has installed the default driver, you need to update your USB driver to the INF file we provided.

The INF file just need to be installed for the first time to use.

2.4.2 Open the demo software: You just need to go into the TFG1900B_Demo directory and open the project in it to operate the software.

2.4.3 Operating sequence: User should take notice of that when you have open the instrument and software, you should first click the VISA USB Raw box on the front panel of the software to select your instrument.

2.4.4 Operating the Demo software: Turn on the instrument. After initialization, the instrument enters into the local state acquiescently and can be operated by keyboard. Then you can open the demonstration software to operate it.

On the front panel of the software, there are three parts and some buttons combining to configure the instrument. And you are responsible for checking the validity of the parameters.

Parameters are firstly set through controls on the tab, then you can press the 【Validate】 button to make it work. The effects of other buttons are clearly when you press it to check.

2.5 Exit program: press【Local】, computer send the "SYSTem:LOCal" command. The instrument exit programmable and return "local" state. The sign "Remote" in the up-right corner of screen disappears. All press-keys in panel resume their function. If user sends the programmable command again, the instrument will return to the programmable state.

Close the interface and turn off the power switch, and then you can remove the USB

cable.

2.6 Reset command: press 【Reset】, computer send the '*RST' command. The instrument resumes the initial state.

3. Application

The software in CD is just a simple USB demo program. The PC can programmable control the instrument through USB interface. Instrument is able to make the right response for each command within practical command set. Command response and working parameters loaded from instrument to computer are able to be shown in demo interface exactly. If users need to set the auto measuring system by PC for more complicated work, you can program the application by yourself. This demonstration is programmed by LabView 2009(V9.0) software, and the application can be programmed with the following data format. If user wants to program the application by himself, you can make reference to the following data.

3.1 Download data format: Computer send a programmable command string. The first byte is data length (including end character), then programmable command ASCII code string, last for end character (ASCII code value 10). Instrument receives the specified amount characters along with data length, nothing with end character, but end character must be needed. Because a string may conclude several commands, after receiving command character string, the instrument will check the correctness of the string, and instrument will execute continuously until meeting with end character, even when instrument executes next commands. If miss the end character, instrument may be out of work.

3.2 Upload data format: After receiving and executing the programmable command, the instrument send a command-responding character string, "Receive ok" or "Receive error". If meeting the query command with "?" in the executing process, the instrument firstly send the query-responding character string to computer, then send command-responding character string "Receive ok". The first byte of send character string is character string 63. The length of character string is fixed to 63, and it should be filled up with space if not long enough for 63.

Make sure that after receiving the upload data then the computer can send the next programmable character string.