GA4OXX Spectrum Analyzer

Programming Manual

Version A.1

GRATTEN

Content

This manual is intended to guide the user to use the remote control commands to control the measurement of GA40XX spectrum analyzer. This manual contains the following contents:

- **Section 1:**

  Overview on the remote command programming and SCPI-related specifications.

- **Section 2:**

  Detailed description of the GA40XX spectrum analyzer command system based on functions.

- **Section 3:**

  Specific programming steps and code implementation.

Contents

GA40XX

1 Overview

Spectrum Analyzer

1 Overview

1-1 Programming Overview

The following interfaces can be used for the communication between the spectrum analyzer and computer:

- COM: complying with RS232 standard; communication format: “8-N-1” ; default rate: 115200bps
- LAN: complying with IEEE802.3 standard, supporting SOCKET communication (the port number is fixed as 5025)

When commands are used for programming, all commands are sent in the form of ASCII character strings in order to facilitate users’ manipulation and secondary development.

The following operations can be made through programming:

- Setting the spectrum analyzer
- Measuring
- Acquiring data from the spectrum analyzer

1-2 SCPI Introduction

SCPI (Standard Commands for Programmable Instrument) is the standard instruction set for the programmable instruments in the IEEE 488.2. SCPI is divided into two parts: IEEE 488.2 common commands and SCPI specific control commands for instrument.

The common commands are the commands which must be supported by the instrument as specified in IEEE 488.2 and its syntax and should follow the specifications of IEEE 488.2.The common commands have nothing to do with the measurement and they are used to control the reset, self-test and status operations. For the introduction of SCPI common commands, please refer to the IEEE 488.2 protocol standard.

SCPI specific control commands for instrument are used for measurement and recording of data and switch change-over, including all measurement functions and some special performance functions.

1-2-1 Command Format

SCPI commands are of tree structure and include multiple subsystems. Each subsystem consists of a root keyword and one or several layer keywords. The command line usually begins with a colon ":" and between the keywords there is a colon ":" for separation. Following the keywords is the optional parameter setting. A question mark "?" is added at the end of the command line, indicating the query on this function. The command and the parameter are separated by "space".

For example:

:SENSe:FREQuency:STARt <freq_value>

:SENSe:FREQuency:STARt?

SENSe is the root keyword of the command. FREQuency and STARt are the second-level and third-level keywords respectively. The command line starts with a colon ":" and it is also used to separate different levels of keywords. <freq_value> means the parameters able to be set and question mark "?" indicates the query. “Space” is used to separate the command SENSe: FREQuency: STARt and parameter <freq_value>.

In the commands with parameters, usually the comma "," is used to separate

multiple parameters, For example: SYSTem: the DATE <year> <month>, the <day>.

1-2-2 Description of Symbols

The four symbols below are not the contents of the SCPI but usually used for supplementary description of the parameters in the commands.

1. Braces { }

The parameter in the braces is optional, which can be set or not set. It is can be set for one time or several times.

For example:

[:SENSe]:CORRection:CSET<n>:DATA<freq>,<rel_ampl>{,<freq>,<rel_ampl>}

In the command, the frequency and amplitude in {<freq>, <rel_ampl>} can be omitted. It is also possible to set one pair or several pairs of frequency and amplitude parameters.

2. Vertical bar |

The vertical bar is used to separate multiple parameter options and one parameter must be selected when sending commands.

For example:

In the command :

DISPlay:MENU:STATe OFF|ON|0|1,

the optional command parameter can be "OFF" , "ON","0" or "1."

3. Square brackets [ ]

The content (command keyword) in brackets is optional and it is implemented whether it is omitted or not.

For example:

[:SENSe]:FREQuency:STARt?
The results of sending the following two commands are the same:

:FREQuency:STARt?

:SENSe:FREQuency:STARt?

4. Triangular brackets < >

The parameters in the triangular brackets must be replaced with a valid value.

For example:

:SENSe:FREQuency:STARt <freq_value>

:SENSe:FREQuency:STARt 500MHz

1-2-3 Parameter type

The parameters contained in the commands which are described in this manual can be divided into the following types: Boolean, Keyword, Integer, Continuous real, discrete type, and ASCII character string.

1. Boolean

The parameter values can be "OFF", "ON", "0" or "1".

For example:

:INITiate:CONTinuous OFF|ON|0|1

2. Keywords

The values of parameters are the listed ones.

For example:

DETector[:FUNCtion] POSitive|NEGative|SAMPle|NORMal
The parameters can be "POSitive", "NEGative", "SAMPle" or "Normal".

3. Integer

Unless otherwise indicated, the parameter can be any integer value within the range of valid values. Please note that do not set the parameter to be in decimal format. Otherwise, there will be exception.

For example:

:SWEep:TIME < integer >

The parameter <integer> can be any integer from 0 to 65536.

4. Continuous real

The parameter can be any values within the range of valid value according to precision requirement (usually the default precision is to take six-digit valid value after the decimal point).

For example:

:SENSe:FREQuency:STARt <freq_value>

5. Discrete type

The parameter can only take specified values and these values are not continuous. For example:

:CALCulate:MARKer<n>:MAXimum:MAX

Parameter <n> can only be 1, 2, 3, or 4.

6. ASCII character string

The parameter value is the combination of ASCII characters.

For example:

:SYSTem:DATE <year>,<month>,<day>

The parameter is the set character string in date format.

1-3 Command Abbreviation

All commands are case insensitive and you can use uppercase or lowercase for all. But if you need abbreviations, all the capital letters in the command format should be entered. For example:

:SENSe:FREQuency:STARt? can be abbreviated as :SENS:FREQ:STAR?

2

GA40XX Command Set System

Spectrum Analyzer

2 GA40XX Command Set System

In this chapter, the subcommand system of GA40XX is introduced in alphabetical order (except the IEEE488.2 standard commands).

- IEEE 488.2
- :CALCulate
- :CONFigure
- :COUPle
- :DISPlay
- : FORMat
- :INITiate
- :OUTPut
- [:SENSe]
- SOURce
- :TRACe
- :TRIGer
- UNIT

2-1 IEEE 488.2

*IDN?
*RST

*IDN?	
Command format	*IDN?
Function description	Query the instrument ID character string.
Note	
Example	*IDN? Gratten Technologies, GA40XX, SN11120001, 1.0.0.0

*RST	
Command format	*RST
Function description	Reset the instrument to the default state value.
Note	
Example	

2-2 CALCulate

CALCulate:LLINe:ALL[:STATE]
CALCulate:LLINe:FAIL,
CALCulate:LLINe**<n>**:STATE
CALCulate:LLINe**<n>**DATA
CALCulate:LLINe**<n>**:DELete
CALCulate:NTData[:STATe]
CALCulate:MARKer**<n>**:X:STOP
CALCulate:MARKer**<n>**:FUNCtion
CALCulate:MARKer<n>:MODE
CALCulate:MARKer:AOFF
CALCulate:MARKer**<n>** [:SET] :CENT
CALCulate:MARKer**<n>** [:SET] : STARt
CALCulate:MARKer**<n>** [:SET] : STOP
CALCulate:MARKer**<n>**:PEAK:CONTinuous[:STATe]
CALCulate:MARKer**<n>**:MAXimum:MAX
CALCulate:MARKer**<n>**:MAXimum: LEFT
CALCulate:MARKer**<n>**:MAXimum: RIGHt
CALCulate:MARKer**<n>**:MAXimum: NEXT
CALCulate:MARKer**<n>**: MINimum
CALCulate:MARKer**<n>**: X?
CALCulate:MARKer**<n>**: Y?
CALCulate:MARKer**<n>**:FCOunt[:STATe]
CALCulate:MARKer**<n>**:FCOunt:X?

CALCulate:LLINe:ALL[:STATE]	
Command format	CALCulate:LLINe:ALL[:STATE] OFF|ON|0|1
Function description	Set up the pass/fail test function open or closed
Note	
Example	CALCulate:LLINe:ALL ON CALCulate:LLINe:ALL?

CALCulate:LLINe:FAIL?	
Command format	CALCulate:LLINe:FAIL?

Function description	Obtain the PASS/FAIL test results, return to PASS or FAIL
Note	
Example	CALCulate:LLINe:FAIL?

CALCulate:LLINe**<n>**:STATE	
Command format	CALCulate:LLINe**<n>**:STATE OFF|ON|0|1
Function description	Set the specified limit line function turned ON or OFF
Note	
Example	CALCulate:LLINe**1**:STATE ON CALCulate:LLINe**1**:STATE?

CALCulate:LLINe**<n>:**DATA	
Command format	CALCulate:LLINe**<n>:**DATA /* <x-axis>,<ampl>,1{,x-axis>,<ampl>,1}
Function description	Set the data for limit line
Note	
Example	CALCulate:LLINe**1:**DATA 1GHz, -20dBm,1

CALCulate:LLINe**<n>**:DELete	
Command format	CALCulate:LLINe**<n>**:DELete
Function description	Delete all the date of limit line
Note	
Example	CALCulate:LLINe**1**:DELete

CALCulate:NTData[:STATe]	
Command format	CALCulate:NTData[:STATe] OFF|ON|0|1
Function description	Set the trace source normalized open or closed
Note	
Example	CALCulate:NTData:STATe ON

CALCulate:MARKer**<n>**:X:STOP

Command format	CALCulate:MARKer**<n>**:X:STOP <freq>
Function description	Set the difference value frequency Marker of frequency
Note	
Example	CALCulate:MARKer**1**:X:STOP 1GHz

CALCulate:MARKer<n>:FUNCtion	
Command format	CALCulate:MARKer<n>:FUNCtion NOISe|OFF
Function description	Set the frequency marker testing function， NOISe is phase noise testing, OFF is closing the frequency marker testing function
Note	
Example	CALCulate:MARKer1:FUNCtion NOISe

CALCulate:MARKer<n>:MODE	
Command format	CALCulate:MARKer<n>:MODE POSition|DELTa|OFF
Function description	Set the mode of the current frequency marker
Note	
Example	CALCulate:MARKer1:MODE POSition CALC:MARK1:MODE POS

CALCulate:MARKer:AOFF	
Command format	CALCulate:MARKer:AOFF
Function description	Close all frequency markers
Note	
Example	CALCulate:MARKer:AOFF CALC:MARK:AOFF

CALCulate:MARKer**<n>** [:SET] :CENT	
Command format	CALCulate:MARKer**<n>** [:SET] :CENT
Function description	Set the frequency corresponding to the current frequency marker as the Center frequency.
Note	
Example	CALCulate:MARKer**1**:SET:CENT

	CALC:MARK**1**: CENT

CALCulate:MARKer**<n>** [:SET]:STARt	
Command format	CALCulate:MARKer**<n>**[:SET]:STARt
Function description	Set the frequency corresponding to the current frequency marker as the start frequency
Note	
Example	CALCulate:MARKer**1**:SET:STARt CALC:MARK**1**:STARt

CALCulate:MARKer**<n>** [:SET] : STOP	
Command format	CALCulate:MARKer**<n>** [:SET]:STOP
Function description	Set the frequency corresponding to the current frequency marker as the stop frequency.
Note	
Example	CALCulate:MARKer**1**:SET:STOP CALC:MARK**1**:STOP

CALCulate:MARKer**<n>**:PEAK:CONTinuous[:STATe]	
Command format	CALCulate:MARKer**<n>**:PEAK:CONTinuous[:STATe] OFF|ON|0|1
Function description	Set the continuous peak on / off
Note	
Example	CALCulate:MARKer**1**:PEAK:CONTinuous:STATe OFF CALC:MARK**1**:PEAK:CONT OFF

CALCulate:MARKer**<n>**:MAXimum:MAX	
Command format	CALCulate:MARKer**<n>**:MAXimum:MAX
Function description	Search the maximum peak
Note	
Example	CALCulate:MARKer**1**:MAXimum:MAX CALC:MARK**1**:MAX: MAX

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

Command format	CALCulate:MARKer**<n>**:MAXimum: LEFT
Function description	Left peak
Note	
Example	CALCulate:MARKer**1**:MAXimum: LEFT CALC:MARK**1**:MAX:LEFT

CALCulate:MARKer**<n>**:MAXimum: RIGHt	
Command format	CALCulate:MARKer**<n>**:MAXimum:RIGHt
Function description	Right peak
Note	
Example	CALCulate:MARKer**1**:MAXimum:RIGHt CALC:MARK**1**:MAX:RIGHt

CALCulate:MARKer**<n>**:MAXimum: NEXT	
Command format	CALCulate:MARKer**<n>**:MAXimum:NEXT
Function description	Search the next peak
Note	
Example	CALCulate:MARKer**1**:MAXimum:NEXT CALC:MARK**1**:MAX:NEXT

CALCulate:MARKer**<n>**: MINimum	
Command format	CALCulate:MARKer**<n>**:MINimum
Function description	Search the minimum peak
Note	
Example	CALCulate:MARKer**1**:MINimum CALC:MARK**1**:MIN

CALCulate:MARKer**<n>**: X?	
Command format	CALCulate:MARKer**<n>**:X?
Function description	Query the frequency value corresponding to the specified frequency marker.
Note	

Example	CALCulate:MARKer**1**:X? CALC:MARK1:X?

CALCulate:MARKer**<n>**: Y?	
Command format	CALCulate:MARKer**<n>**: Y?
Function description	Query the amplitude value corresponding to the specified frequency marker.
Note	
Example	CALCulate:MARKer**1**: Y? CALC:MARK**1**: Y?

CALCulate:MARKer**<n>**:FCOunt[:STATe]	
Command format	CALCulate:MARKer**<n>**:FCOunt[:STATe] OFF|ON|0|1
Function description	open/close the frequency tally function
Note	
Example	CALC:MARKer**1**:FCOunt:STATe ON

CALCulate:MARKer**<n>**:FCOunt:X?	
Command format	CALCulate:MARKer**<n>**:FCOunt:X?
Function description	readout the numerical value of frequency meter
Note	
Example	CALC:MARK**1**:FCO:X?

2.3 COUPle

COUPle

COUPle	
Command format	COUPle ALL|NONE
Function description	Use all the parameters of the coupling function according to the coupling function linkage
Note	
Example	CALC:MARK**1**:FCO:X?

2-4 DISPlay

DISPlay:ENABle
DISPlay:WINDow:TRACe:Y[:SCALe]:NRLevel
DISPlay:WINDow:TRACe:Y[:SCALe]:NRPosition
DISPlay:WINDow:TRACe:Y[:SCALe]:PDIVision
DISPlay:WINDow:TRACe:Y[:SCALe]:RLEVel
DISPlay:WINDow:TRACe:Y[:SCALe]:SPACing

DISPlay:ENABle	
Command format	DISPlay:ENABle OFF|ON|0|1
Function description	Open or freeze the screen display.
Note	This command is generally used when trace data transmission to improve the transmission speed , when lock the screen ,the trace of the screen won't refresh, reduce the instrument's dispose workload, avoid brush screen affect the trace data's transmission .
Example	DISPlay:ENABle OFF DISP:ENAB OFF

DISPlay:WINDow:TRACe:Y[:SCALe]:NRLevel	
Command format	DISPlay:WINDow:TRACe:Y[:SCALe]:NRLevel <float>
Function description	Set the value of tracking source normalized reference level
Note	
Example	DISPlay:WINDow:TRACe:Y:SCALe:NRLevel 0dB DISPlay:WINDow:TRACe:Y:SCALe:NRLevel?

DISPlay:WINDow:TRACe:Y[:SCALe]:NRPosition	
Command format	DISPlay:WINDow:TRACe:Y[:SCALe]:NRPosition <integer>
Function description	Set the location of tracking source normalized reference level, range from 1 to 11 . and 1 corresponding to the bottom of grid, 11 corresponding to the top of grid .10 is default.

Note	
Example	DISPlay:WINDow:TRACe:Y:SCALe:NRPosition 10

DISPlay:WINDow:TRACe:Y[:SCALe]:PDIVision	
Command format	DISPlay:WINDow:TRACe:Y[:SCALe]:PDIVision <level_value>
Function description	Set the size corresponding to the one grid of Y-axis.
Note	
Example	DISPlay:WINDow:TRACe:Y:SCALe:PDIVision 10dB DISP:WIND:TRAC:Y:PDIV 10dB

DISPlay:WINDow:TRACe:Y[:SCALe]:RLEVel	
Command format	DISPlay:WINDow:TRACe:Y[:SCALe]:RLEVel <level_value>
Function description	Set the reference level value
Note	
Example	DISPlay:WINDow:TRACe:Y:SCALe:RLEVel 0dbm DISP:WIND:TRAC:Y:RLEV 0dbm

DISPlay:WINDow:TRACe:Y[:SCALe]:SPACing	
Command format	DISPlay:WINDow:TRACe:Y[:SCALe]:SPACing LOG|LINear
Function description	Set Y axis calibration type. LOG type or LINear type can be selected
Note	
Example	DISPlay:WINDow:TRACe:Y:SCALe:SPACing LOG DISPlay:WINDow:TRACe:Y:SCALe:SPACing?

2-5 FORMat

FORMat[:TRACe][:DATA]

FORMat[:TRACe][:DATA]	
Command format	FORMat[:TRACe][:DATA] ASCii|INT,16|REAL,32
Function description	Set the output format of the trace data. ASCii is in the character format while INT16 and REAL32 are in binary format.
Note	When the INT16 format is selected, the data is magnified by 100 times. That is to say when the actual data is 1.01, the transmitted data is 101.
Example	FORMat:DATA REAL,32

2-6 INITiate

INITiate:CONTinuous
INITiate:CONTinuous?

INITiate:CONTinuous	
Command format	INITiate:CONTinuous OFF|ON|0|1
Function description	Set the scan mode as continuous or once
Note	
Example	INITiate:CONTinuous OFF INIT:CONT OFF

INITiate:CONTinuous?	
Command format	INITiate:CONTinuous?
Function description	Query the current scan mode
Note	
Example	INITiate:CONTinuous? INIT:CONT?

2.7 OUTPut

OUTPut[:STATe]

OUTPut[:STATe]	
Command format	OUTPut[:STATe] OFF|ON|0|1
Function description	Set the tracking source output open or close
Note	
Example	OUTPut:ON OUTPut?

2.8 SENSe

[SENSe:]FREQuency:CENTer
[SENSe:]FREQuency:CENTer?
[SENSe:]FREQuency:CENTer:STEP[:INCRement]
[SENSe:]FREQuency:CENTer:STEP:AUTO
[SENSe:]FREQuency:STARt
[SENSe:]FREQuency:STARt?
[SENSe:]FREQuency:STOP
[SENSe:]FREQuency:STOP?
[SENSe:]FREQuency:SPAN
[SENSe:]FREQuency:SPAN?
[SENSe:]FREQuency:SPAN:FULL
[SENSe:]FREQuency:SPAN:ZERO
[SENSe:]FREQuency:SPAN:PREVious
[SENSe:]FREQuency:SPAN:BANDwidth:RATio
[SENSe:]FREQuency:SPAN:BANDwidth:AUTO
[SENSe:]POWer[:RF]:ATTenuation
[SENSe:]POWer[:RF]:ATTenuation:AUTO
[SENSe:]DETector[:FUNCtion]
[SENSe:]DETector[:FUNCtion]?
[SENSe:]BANDwidth:RESolution
[SENSe:]BANDwidth:RESolution:AUTO
[SENSe:]BANDwidth:VIDeo
[SENSe:]BANDwidth:VIDeo:AUTO
[SENSe:]BANDwidth:VIDeo:RATio
[SENSe:]BANDwidth:VIDeo:RATio:AUTO
[SENSe:]POWer[:RF]:GAIN[:STATe]
[SENSe:]SWEep:TIME
[SENSe:]SWEep:TIME?
[SENSe:]SWEep:TIME:AUTO
[SENSe:]SWEep:POINts

[SENSe:]FREQuency:CENTer	
Command format	[SENSe:]FREQuency:CENTer <freq_value>
Function	Set the value of center frequency

description	
Note	
Example	SENSe:FREQuency:CENTer 500MHz FREQ:CENT 500MHz

[SENSe:]FREQuency:CENTer?	
Command format	[SENSe:]FREQuency:CENTer?
Function description	Query the value of center frequency
Note	
Example	SENSe:FREQuency:CENTer? FREQ:CENT?

[SENSe:]FREQuency:CENTer:STEP[:INCRement]	
Command format	[SENSe:]FREQuency:CENTer:STEP[:INCRement] <freq_value>
Function description	Set the step value of center frequency
Note	
Example	SENSe:FREQuency:CENTer:STEP:INCRement 50MHz FREQ:CENT:STEP 50MHz

[SENSe:]FREQuency:CENTer:STEP:AUTO	
Command format	[SENSe:]FREQuency:CENTer:STEP:AUTO OFF|ON|0|1
Function description	Set the step value of center frequency as autoadaptation.
Note	
Example	SENSe:FREQuency:CENTer:STEP:AUTO OFF FREQ:CENT:STEP:AUTO OFF

[SENSe:]FREQuency:STARt	
Command format	[SENSe:]FREQuency:STARt <freq_value>
Function description	Set the start frequency
Note	

Example	SENSe:FREQuency:STARt 100KHz FREQ:STAR 100KHz

[SENSe:]FREQuency:STARt?	
Command format	[SENSe:]FREQuency:STARt?
Function description	Query start frequency
Note	
Example	SENSe:FREQuency:STARt? FREQ:STAR?

[SENSe:]FREQuency: STOP	
Command format	[SENSe:]FREQuency:STOP <freq_value>
Function description	Set the stop frequency
Note	
Example	SENSe:FREQuency:STOP 1GHz FREQ:STOP 1GHz

[SENSe:]FREQuency:STOP?	
Command format	[SENSe:]FREQuency:STOP?
Function description	Query the stop frequency.
Note	
Example	SENSe:FREQuency:STOP? FREQ:TOP?

[SENSe:]FREQuency:SPAN	
Command format	[SENSe:]FREQuency:SPAN <freq_value>
Function description	Set the span value
Note	
Example	SENSe:FREQuency:SPAN 800MHz FREQ:SPAN 800MHz

[SENSe:]FREQuency:SPAN?

Command format	[SENSe:]FREQuency:SPAN?
Function description	Query the current span
Note	
Example	SENSe:FREQuency:SPAN? FREQ:SPAN?

[SENSe:]FREQuency:SPAN:FULL	
Command format	[SENSe:]FREQuency:SPAN:FULL
Function description	Full span
Note	
Example	SENSe:FREQuency:SPAN:FULL FREQ:SPAN:FULL

[SENSe:]FREQuency: SPAN:ZERO	
Command format	[SENSe:]FREQuency:SPAN:ZERO
Function description	Zero span
Note	
Example	SENSe:FREQuency:SPAN:ZERO FREQ:SPAN:ZERO

[SENSe:]FREQuency:SPAN:PREVious	
Command format	[SENSe:]FREQuency:SPAN:PREVious
Function description	Previous span
Note	
Example	SENSe:FREQuency:SPAN:PREVious FREQ:SPAN:PREV

[SENSe:]FREQuency:SPAN:BANDwidth:RATio	
Command format	[SENSe:]FREQuency:SPAN:BANDwidth:RATio < value>
Function description	Set the RBW/Span ratio

Note	
Example	[SENSe:]FREQuency:SPAN:BANDwidth:RATio 100 FREQ:SPAN:BAND:RAT 100

[SENSe:]FREQuency:SPAN:BANDwidth:AUTO	
Command format	[SENSe:]FREQuency:SPAN:BANDwidth:AUTO OFF|ON|0|1
Function description	Set the RBW/Span ratio as automatic
Note	
Example	SENSe:FREQuency:SPAN:BANDwidth:AUTO OFF SENS:FREQ:SPAN:BAND:AUTO OFF

[SENSe:]POWer[:RF]:ATTenuation	
Command format	[SENSe:]POWer[:RF]:ATTenuation <ampl_value>
Function description	Set the attenuation value
Note	
Example	SENSe:POWer:RF:ATTenuation 10dBm POW:ATT 10dBm

[SENSe:]POWer[:RF]:ATTenuation:AUTO	
Command format	[SENSe:]POWer[:RF]:ATTenuation:AUTO OFF|ON|0|1
Function description	Set the attenuation automatic function on / off
Note	
Example	SENSe:POWer:RF:ATTenuation:AUTO OFF POW: ATT:AUTO OFF

[SENSe:]DETector[:FUNCtion]	
Command format	[SENSe:]DETector[:FUNCtion] POSitive|NEGative|SAMPle|NORMal
Function description	Set the detection method of the current trace
Note	
Example	SENSe:DETector:FUNCtion POSitive

	DET POS

[SENSe:]DETector[:FUNCtion]?	
Command format	[SENSe:]DETector[:FUNCtion]?
Function description	Query the detection method of the current trace
Note	
Example	SENSe:DETector:FUNCtion? DET?

[SENSe:]BANDwidth:RESolution	
Command format	[SENSe:]BANDwidth:RESolution <freq_value>
Function description	Set the value of RBW
Note	
Example	SENSe:BANDwidth:RESolution 10MHz BAND:RES 1MHz

[SENSe:]BANDwidth:RESolution:AUTO	
Command format	[SENSe:]BANDwidth:RESolution:AUTO OFF|ON|0|1
Function description	Set RBW automatic function on / off
Note	
Example	SENSe:BANDwidth:RESolution :AUTO ON BAND:RES:AUTO ON

[SENSe:]BANDwidth:VIDeo	
Command format	[SENSe:]BANDwidth: VIDeo <freq_value>
Function description	Set the value of VBW
Note	
Example	SENSe:BANDwidth: VIDeo 3MHz BAND:VID 3MHz

[SENSe:]BANDwidth:VIDeo:AUTO	
Command format	[SENSe:]BANDwidth: VIDeo:AUTO OFF|ON|0|1

Function description	Set VBW automatic function On / Off
Note	
Example	SENSe:BANDwidth: VIDeo:AUTO ON BAND:VID:AUTO ON

[SENSe:]BANDwidth:VIDeo:RATio	
Command format	[SENSe:]BANDwidth: VIDeo: RATio <number>
Function description	Set the VBW/RBW ratio
Note	
Example	SENSe:BANDwidth: VIDeo: RATio 1 BAND:VID:RAT 1

[SENSe:]BANDwidth:VIDeo:RATio:AUTO	
Command format	[SENSe:]BANDwidth:VIDeo:RATio:AUTO OFF|ON|0|1
Function description	Set the VBW/RBW ratio automatic function on / off
Note	
Example	SENSe:BANDwidth:VIDeo:RATio:AUTO OFF BAND:VID:RAT:AUTO OFF

[SENSe:]POWer[:RF]:GAIN[:STATe]	
Command format	[SENSe:]POWer[:RF]:GAIN[:STATe]OFF|ON|0|1
Function description	Set the preamplifier on / off
Note	
Example	SENSe:POWer:RF:GAIN:STATe OFF POW:GAIN OFF

[SENSe:]POWer[:RF]:GAIN[:STATe]?	
Command format	[SENSe:]POWer[:RF]:GAIN[:STATe]?
Function description	Query the preamplifier state
Note	

Example	SENSe:POWer:RF:GAIN:STATe? POW:GAIN?

[SENSe:]SWEep:TIME	
Command format	[SENSe:]SWEep:TIME <time_value>
Function description	Set the scan time
Note	
Example	SENSe:SWEep:TIME 100ms SWE:TIME 100ms

[SENSe:]SWEep:TIME?	
Command format	[SENSe:]SWEep:TIME?
Function description	Query the scan time
Note	
Example	SENSe:SWEep:TIME? SWE:TIME?

[SENSe:]SWEep:TIME:AUTO	
Command format	[SENSe:]SWEep:TIME:AUTO OFF|ON|0|1
Function description	Set the automatic function of the scan time
Note	
Example	SENSe:SWEep:TIME:AUTO ON SWE:TIME:AUTO ON

[SENSe:]SWEep:POINts	
Command format	[SENSe:]SWEep:POINts FHUNdred|THOusand|DTHousand
Function description	Set the scan points, the three setting values correspond to 501/1001/2001 points respectively.
Note	
Example	SENSe:SWEep:POINts FHUNdred SWE:POIN FHUN

[SENSe:]SWEep:POINts?

Command format	[SENSe:]SWEep:POINts?
Function description	Query the current scan points
Note	
Example	SENSe:SWEep:POINts? SWE:POIN?

2.9 SOURce

SOURce:MEASure:SELect
SOURce:POWer[:LEVel][:IMMediate][:AMPLitude]

SOURce:MEASure:SELect	
Command format	SOURce:MEASure:SELect S21_S11|VSWR
Function description	Set the tracking source measurement pattern
Note	When select S21_S11 measurement pattern， S21 or S11 which is in testing it is depend on the external device connection type, if the connected external device is a directional coupler, then measurement is S11.
Example	SOURce:MEASure:SELect S21_S11 SOURce:MEASure:SELect?

SOURce:POWer[:LEVel][:IMMediate][:AMPLitude]	
Command format	SOURce:POWer[:LEVel][:IMMediate][:AMPLitude] <ampl>
Function description	Set the tracking generator output amplitude, allowable range is from 0dBm to -25dBm.
Note	
Example	SOURce:POWer -10dBm SOURce:POWer?

2-10 TRACe

TRACe<n>:MODE
TRACe<n>:MODE?
TRACe:CLEar:ALL
TRACe:DATA?

TRACe<n>:MODE	
Command format	TRACe<n>:MODE WRITe|MAXHold|MINHold|VIEW|BLANk|AVERage
Function description	Set the mode of the specified trace
Note	
Example	TRACe1:MODE WRITe TRAC1:MODE WRIT

TRACe<n>:MODE?	
Command format	TRACe<n>:MODE?
Function description	Query the mode of the specified trace
Note	
Example	TRACe1:MODE? TRAC1:MODE?

TRACe:CLEar:ALL	
Command format	TRACe:CLEar:ALL
Function description	Clear all traces
Note	
Example	TRACe:CLEar:ALL TRAC:CLE:ALL

TRACe:DATA?	
Command format	TRACe:DATA? TRACE1|TRACE2|TRACE3
Function description	Read the trace data.

Note	The number of data for each trace is equal to that of the scan points. If you want to improve the trace data transmission speed, you can use the command DISPlay:ENABle OFF to stop screen refreshing. In addition, the command FORMat:TRACe:DATA INT,16 can be used to set the data as the binary mode, which can also reduce the data amount transferred so as to improve the transmission rate .
Example	TRACe:DATA? TRACE1 TRAC:DATA? TRACE1

2-11 TRIGgle

TRIGgle:MODE
TRIGgle:LEVel
TRIGgle:SLOPe
TRIGgle:DELay
TRIGgle:DELay:STATe

TRIGgle:MODE	
Command format	TRIGgle:MODE FREE|VIDeo|EXTernal
Function description	Set the trigger type
Note	
Example	TRIGgle:MODE FREE TRIG:MODE FREE

TRIGgle:LEVel	
Command format	TRIGgle:LEVel <numeric_value>
Function description	Set the trigger level amplitude
Note	
Example	TRIGgle:LEVel -20 TRIG:LEV -20

TRIGgle:SLOPe	
Command format	TRIGgle:SLOPe POSitive|NEGitive
Function description	Set the trigger polarity
Note	
Example	TRIGgle:SLOPe POSitive TRIG:SLOP POS

TRIGgle:DELay	
Command format	TRIGgle:DELay <numeric_value>
Function description	Set the trigger delay time

Note	
Example	TRIGgle:DELay 2 TRIG:DEL 2

TRIGgle:DELay:STATe	
Command format	TRIGgle:DELay:STATe OFF|ON|0|1
Function description	Set the trigger delay OFF or ON.
Note	
Example	TRIGgle:DELay:STATe OFF TRIG:DEL:STAT OFF

2.12 UNIT

UNIT:POWer

UNIT:POWer	
Command format	UNIT:POWer DBM|DBMV|DBUV|WATTs|VOLTs
Function description	Set the Y axis unit
Note	
Example	UNIT:POWer DBM

GA40XX

3 Programming Guide

Spectrum Analyzer

3 Programming Guide

This chapter is to guide the user to use Microsoft Visual C++ for coding to realize programming control on the spectrum analyzer.

NI-VISA is also needed for programming. NI-VISA is one set of application programming interface (API) developed by NI for instrument control complying with the VISA standard.

VISA (Virtual Instrument Software Architecture) is an advanced application programming interface used for the communication with various kinds of instrument bus. It can achieve the bus driver for a variety of instruments communication within and it also provides unified interface for the upper application so that the users do not need to care about the lower communication implementation but focus on application development.

The following sections will use two detailed examples to demonstrate the specific realization process of programming.

3-1 The Programming Based on LAN interface

3-1-1 Preparations

1. Hardware environment preparation

In this section, the LAN interface of the spectrum analyzer is used for the communication with the computer. Please use one crossover cable to connect the LAN interface on the rear panel of the spectrum analyzer to the PC.

2. Software environment preparation

Make sure that NI VISA has been installed on the computer (it can be downloaded on the NI website http://www.ni.com/china). The default installation path is C: \ Program Files \ IVI Foundation \ VISA.

3. Configure the instrument network parameters

It is assumed here that the network parameters of the computer is as follows

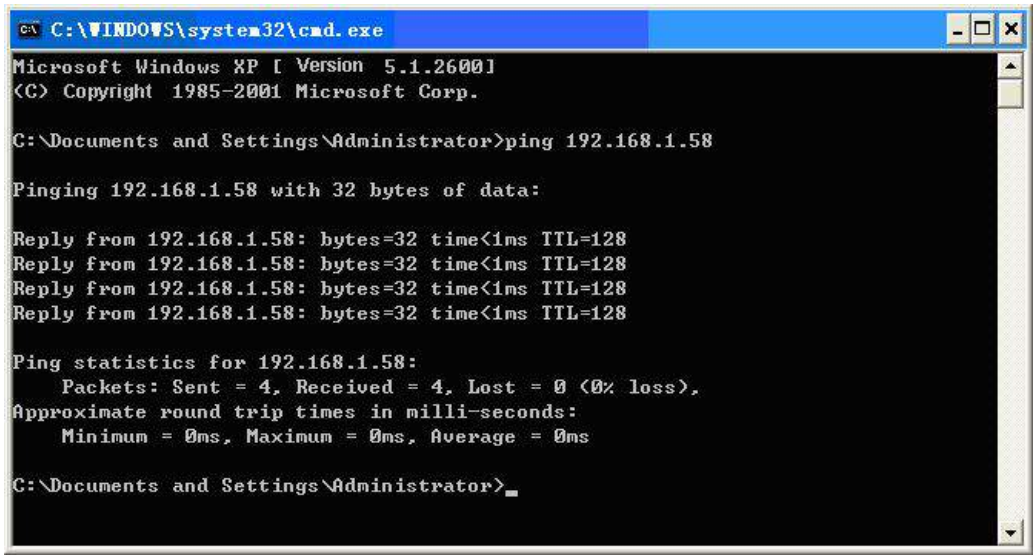
- IP Address: 192.168.1.100
- Subnet Mask: 255.255.255.0
- Gateway address: 192.168.1.1

Only when the spectrum analyzer is on the same network segment (192.168.1.x) as the computer can the communication between them be possible. The network parameters of the spectrum analyzer can be configured as follows:

- IP address: 192.168.1.58
- Subnet mask: 255.255.255.0
- Gateway address: 192.168.1.1

4. Check whether the network connection of the computer and the instrument is normal

Open the DOS command prompt window of the computer's Windows system and use command "Ping 192.168.1.58" to check the connection. If there appears the prompt "Packets: Sent = 4, Received = 4, Lost = 0 (0% loss)," the network connection of computer and the instrument is normal.



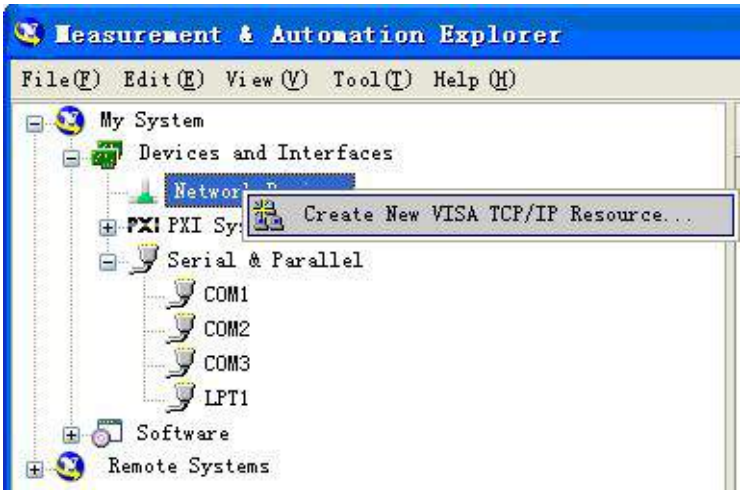
5. Use "Measurement & Automation Explorer" to configure NI-VISA

NI-VISA is provided with the tool "Measurement & Automation Explorer" which is used to configure the communication with the instrument. It can also be used to enter manually the SCPI command to control the instrument.

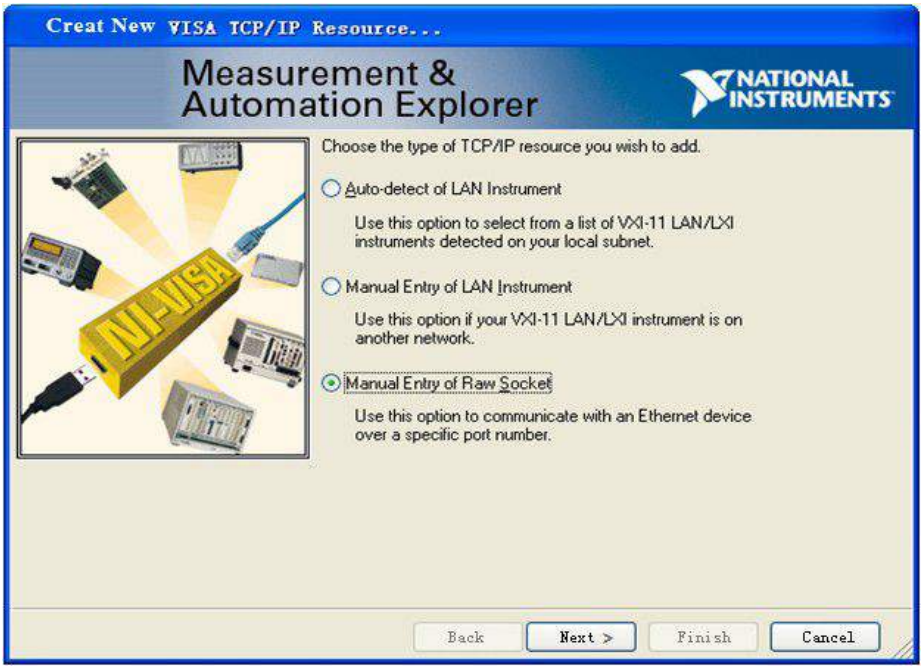
The configuration steps are as follows:

1) Run “Measurement & Automation Explorer”

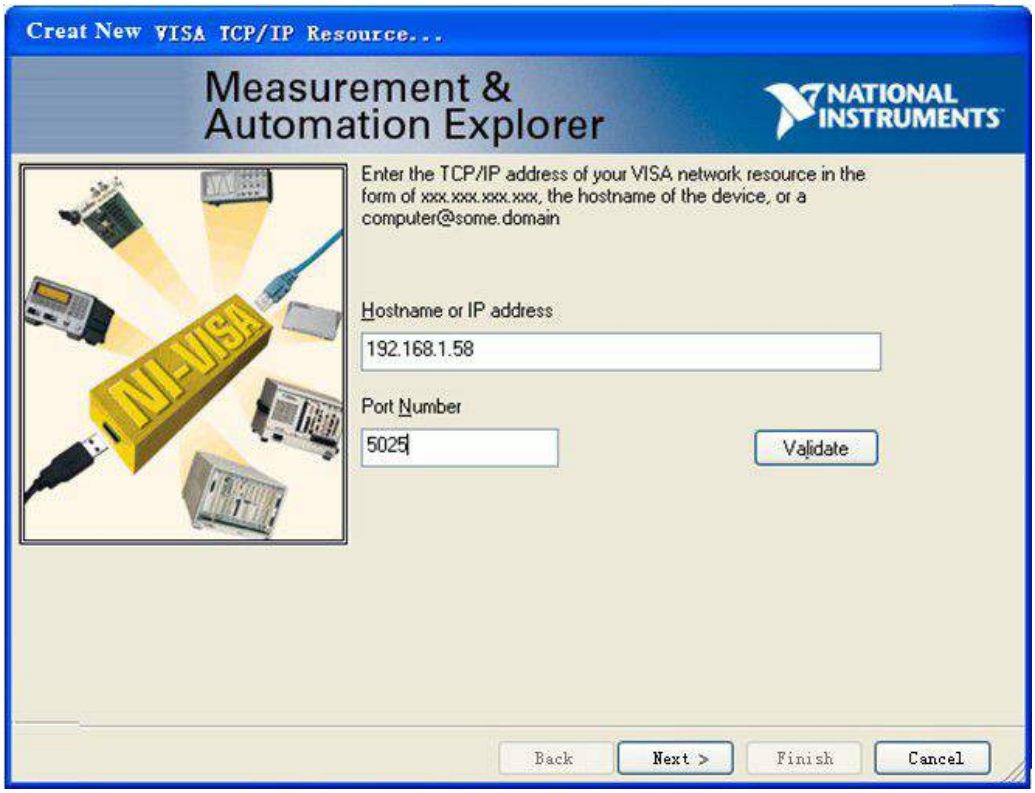
2) In the list on the left, right-click the network device, select the context menu to create a new VISA resource, as shown in the figure below



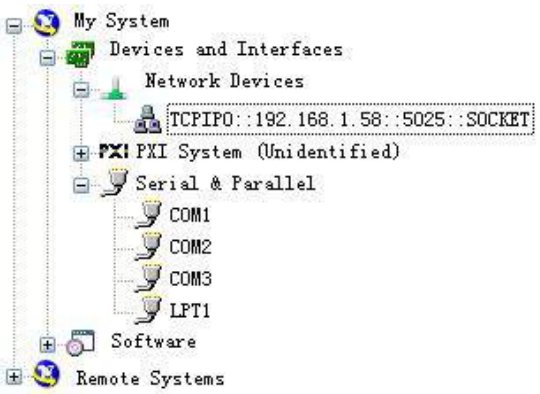
3) Select the resource type "Manual Entry of Raw Socket" in “type of TCP/IP resource” and then click "Next"



4) Set the IP address as "192.168.1.58" and the port number "5025", as shown in the figure below.



5) Click on "Finish". Now there is one more item "TCPIP0::192.168.1.58::5025::SOCKET" below the network devices on the left list, as shown in the following figure.

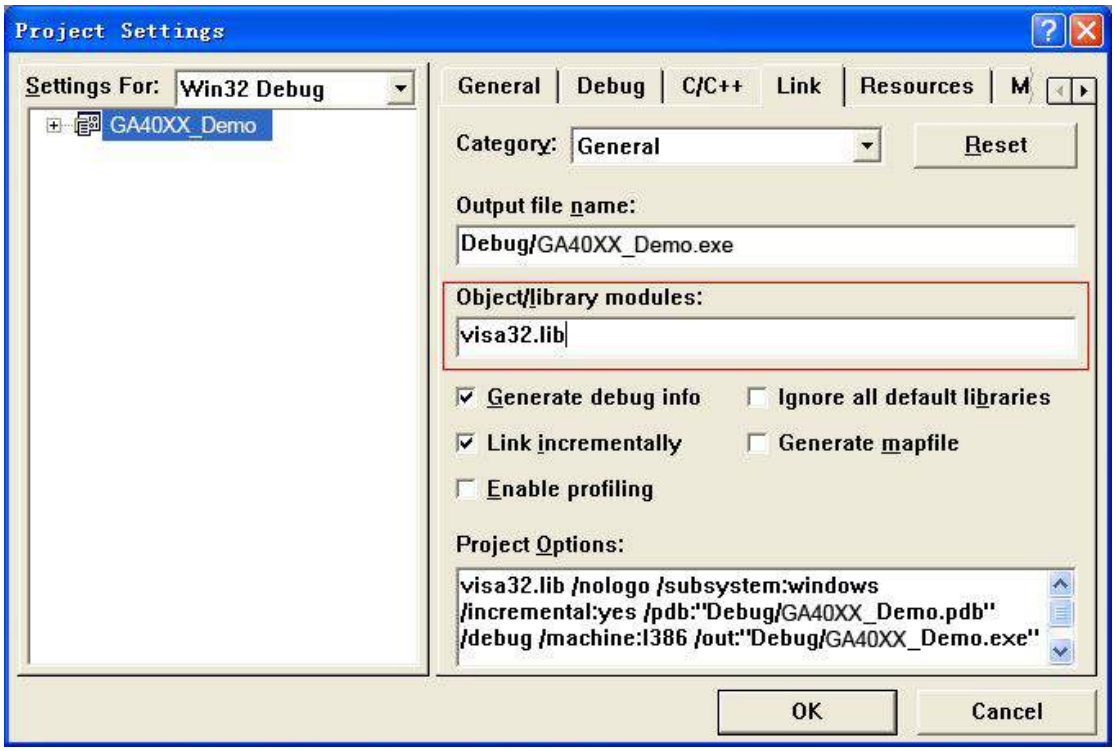


3-1-2 Use Visual C++ for Programming

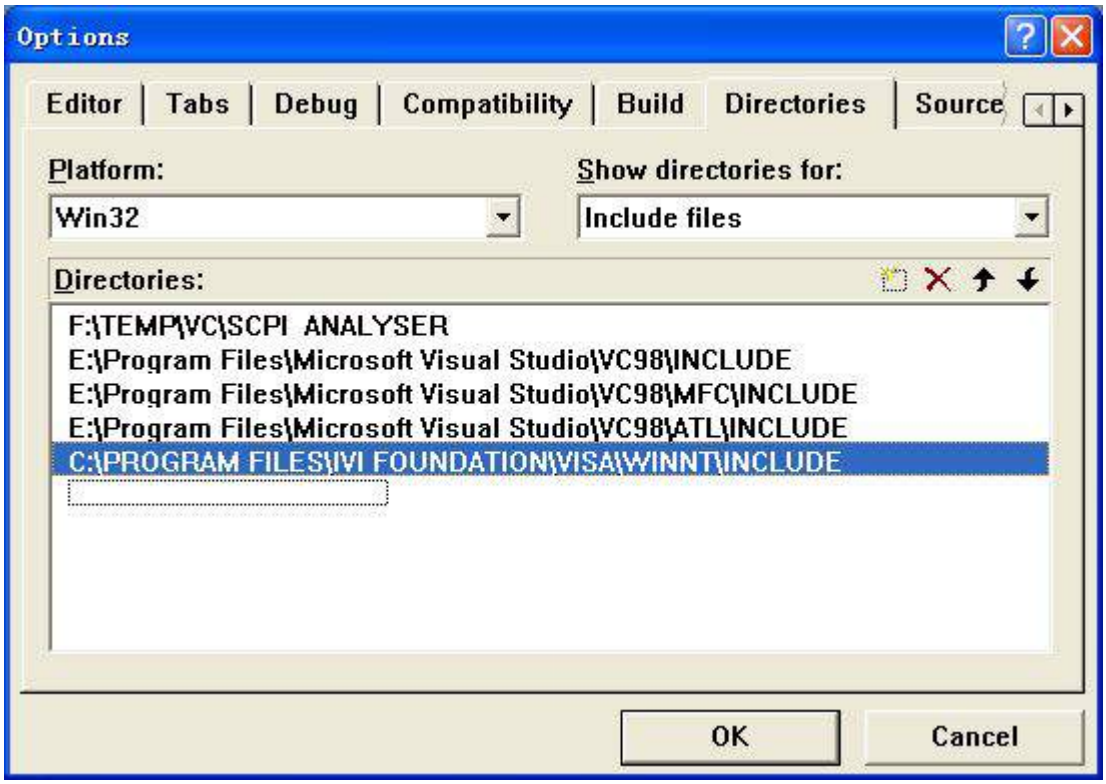
Get into the Visual C++ 6.0 programming environment. Follow these steps:

1. Create a dialog -based MFC project “GA40XX_Demo”.

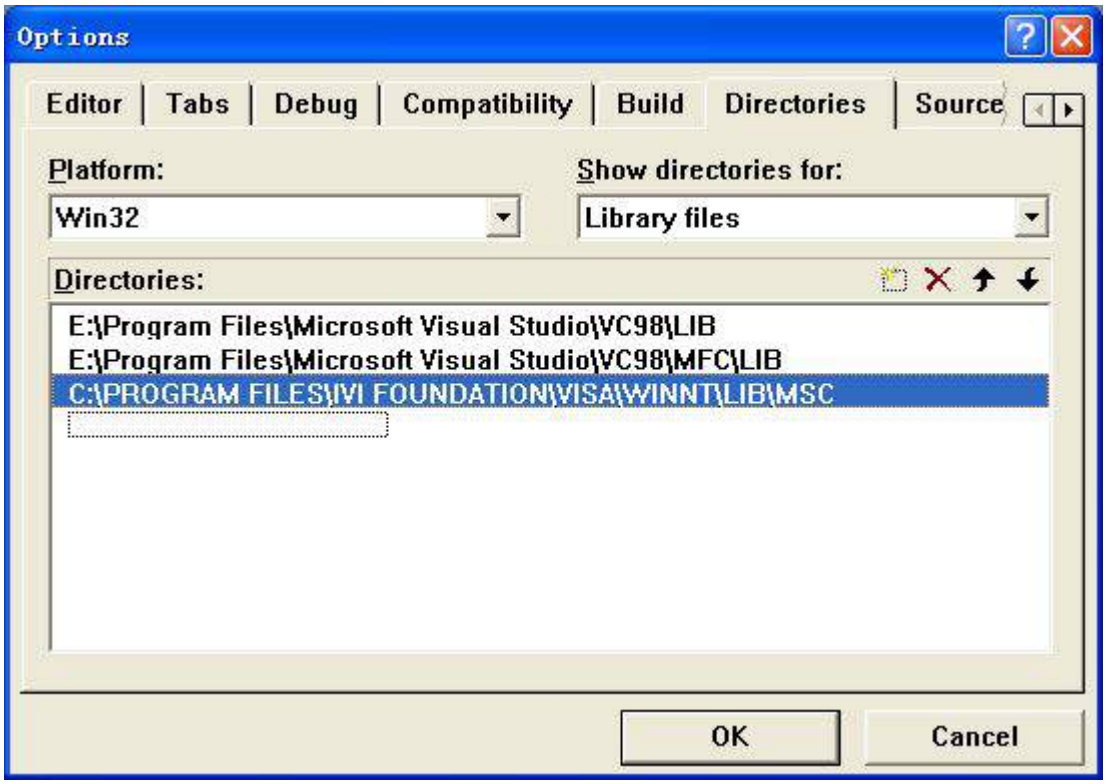
2. Open the Link tab in “Project → Settings”, manually add visa32.lib in the Object / library modules and click "OK" to confirm.



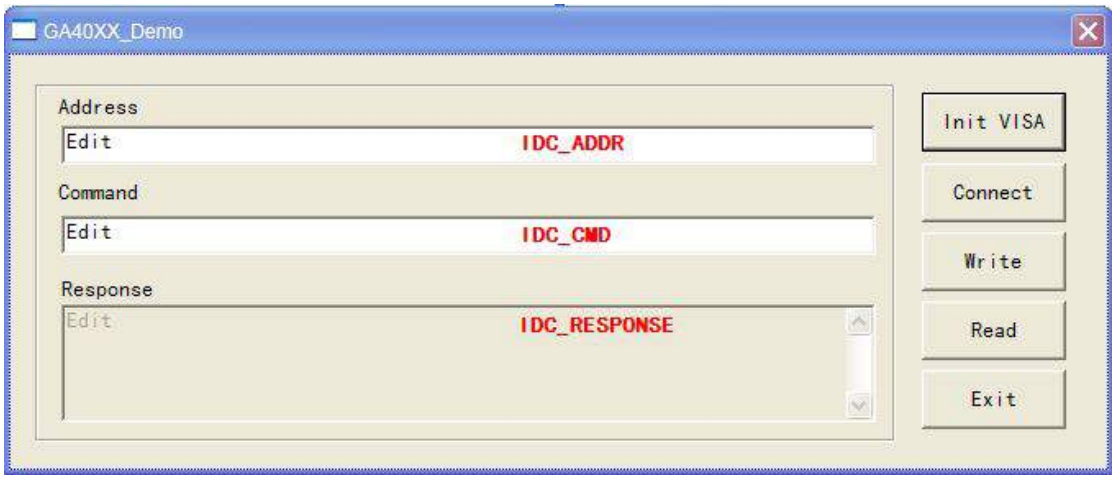
3. Open the Directories tab in “Tools → Options”.
Select “Include files” in “Show directories for”, add the path of “Include”: C: \ Program Files \ IVI Foundation \ VISA \ WinNT \ include.



Select “Library files” in “Show directories for”, add the path of “Include”: C:\Program Files\IVI Foundation\VISA\WinNT\lib\msc, then click “OK” to confirm.



4. Add control in the dialog box, as shown below. The red font parts are the IDs of 3 edit boxes.



5. Add code

After necessary codes are added, the final codes are as follows. The blue parts are the added codes.

(1) GA40XX_DemoDlg.h File

Note : GA40XX depict the model no of the unit which is being programmed., where XX can be 62, 63, 32, 33, 64. Use the correct GA40XX model no for the program to execute correctly

```
// GA40XX_DemoDlg.h : header file
//

#if   !defined(AFX_GA40XX_DEMODLG_H__52AEBB80_A1CF_4693_A974_9A0F
14B5762A__INCLUDED_)
#define
AFX_GA40XX_DEMODLG_H__52AEBB80_A1CF_4693_A974_9A0F14B5762A_
_INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#include "visa.h"

/////////////////////////////////////////////////////////////////////////////
// CGA40XX_DemoDlg dialog

class CGA40XX_DemoDlg : public CDialog
{
// Construction
public:
      ViSession defaultRM;
      ViSession instr;
      CGA40XX_DemoDlg(CWnd* pParent = NULL);    // standard constructor

// Dialog Data
```

```
	//{{AFX_DATA(CGA40XX_DemoDlg)
	enum { IDD = IDD_GA40XX_DEMO_DIALOG };
		// NOTE: the ClassWizard will add data members here
	//}}AFX_DATA

	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CGA40XX_DemoDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX); // DDX/DDV support
	//}}AFX_VIRTUAL

// Implementation
protected:
	HICON m_hIcon;

	// Generated message map functions
	//{{AFX_MSG(CGA40XX_DemoDlg)
	virtual BOOL OnInitDialog();
	afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
	afx_msg void OnPaint();
	afx_msg HCURSOR OnQueryDragIcon();
	afx_msg void OnInitVisa();
	afx_msg void OnConnect();
	afx_msg void OnWrite();
	afx_msg void OnRead();
	virtual void OnCancel();
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the
previous line.

#endif
//   !defined(AFX_GA40XX_DEMODLG_H__52AEBB80_A1CF_4693_A974_9A0F1
4B5762A__INCLUDED_)
```

(2) GA40XX_DemoDlg.cpp file

```
// GA40XX_DemoDlg.cpp : implementation file
//

#include "stdafx.h"
#include "GA40XX_Demo.h"
#include "GA40XX_DemoDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CGA40XX_DemoDlg dialog

CGA40XX_DemoDlg::CGA40XX_DemoDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CGA40XX_DemoDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CGA40XX_DemoDlg)
		// NOTE: the ClassWizard will add member initialization here
	//}}AFX_DATA_INIT
	// Note that LoadIcon does not require a subsequent DestroyIcon in Win32
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CGA40XX_DemoDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CGA40XX_DemoDlg)
		// NOTE: the ClassWizard will add DDX and DDV calls here
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CGA40XX_DemoDlg, CDialog)
```

```
	//{{AFX_MSG_MAP(CGA40XX_DemoDlg)
	ON_WM_PAINT()
	ON_BN_CLICKED(IDC_INIT_VISA, OnInitVisa)
	ON_BN_CLICKED(IDC_CONNECT, OnConnect)
	ON_BN_CLICKED(IDC_WRITE, OnWrite)
	ON_BN_CLICKED(IDC_READ, OnRead)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CGA40XX_DemoDlg message handlers

BOOL CGA40XX_DemoDlg::OnInitDialog()
{
	CDialog::OnInitDialog();

	// Set the icon for this dialog.  The framework does this automatically
	//  when the application's main window is not a dialog
	SetIcon(m_hIcon, TRUE);			// Set big icon
	SetIcon(m_hIcon, FALSE);		// Set small icon

	// TODO: Add extra initialization here
	defaultRM = 0;
	instr = 0;
	GetDlgItem(IDC_ADDR)->SetWindowText("TCPIP0::192.168.1.58::5025::SO
CKET");
	GetDlgItem(IDC_CMD)->SetWindowText("*IDN?");
	GetDlgItem(IDC_CONNECT)->EnableWindow(FALSE);
	GetDlgItem(IDC_WRITE)->EnableWindow(FALSE);
	GetDlgItem(IDC_READ)->EnableWindow(FALSE);

	return TRUE;  // return TRUE  unless you set the focus to a control
}

void CGA40XX_DemoDlg::OnInitVisa()
{
	// TODO: Add your control notification handler code here
```

```
    ViStatus status;

    status = viOpenDefaultRM(&defaultRM);
    if (status != VI_SUCCESS)
    {
        defaultRM = 0;
        MessageBox("Please confirm that “the VISA driver library has been
normally installed”, " VISA initialization failed ",MB_ICONERROR);
        return;
    }
    GetDlgItem(IDC_CONNECT)->EnableWindow(TRUE);
    GetDlgItem(IDC_INIT_VISA)->EnableWindow(FALSE);
}

void CGA40XX_DemoDlg::OnConnect()
{
    // TODO: Add your control notification handler code here
    ViStatus status;
    ViChar rsrcName[VI_FIND_BUFLEN];
    CString strAddr;

    GetDlgItem(IDC_ADDR)->GetWindowText(strAddr);
    LPTSTR p = strAddr.GetBuffer(strAddr.GetLength());
    strcpy(rsrcName,p);
    strAddr.ReleaseBuffer();

    status = viOpen(defaultRM, rsrcName, VI_NULL, VI_NULL, &instr);
    if (status < VI_SUCCESS)
    {
        instr = 0;
        MessageBox("please conform the hardware is normally connected
","connection failed!" ,MB_ICONERROR);
        return;
    }
    viSetAttribute(instr,VI_ATTR_TERMCHAR_EN,VI_TRUE);

    GetDlgItem(IDC_CONNECT)->EnableWindow(FALSE);
```

```
	GetDlgItem(IDC_WRITE)->EnableWindow(TRUE);
	GetDlgItem(IDC_READ)->EnableWindow(TRUE);
}

void CGA40XX_DemoDlg::OnWrite()
{
	// TODO: Add your control notification handler code here
	ViStatus status;
	ViUInt32 retCount;
	char * SendBuf = NULL;
	CString strCmd;

	GetDlgItem(IDC_CMD)->GetWindowText(strCmd);
	strCmd += "\n";
	SendBuf = strCmd.GetBuffer(strCmd.GetLength());
	strcpy(SendBuf,strCmd);
	strCmd.ReleaseBuffer();

	status = viWrite(instr, (unsigned char *)SendBuf, strlen(SendBuf), &retCount);

	if(status < VI_SUCCESS)
	{
		MessageBox("Writing of commands to the instrument failed!");
		return;
	}
}

void CGA40XX_DemoDlg::OnRead()
{
	// TODO: Add your control notification handler code here
	ViStatus status;
	ViUInt32 retCount;
	unsigned char RecBuf[1024];
	CString pstrResult;

	memset(RecBuf,0,1024);
	status = viRead(instr, RecBuf, 1024, &retCount);
```

```
    if(status < VI_SUCCESS)
    {
        pstrResult = "";
    }
    else
    {
        pstrResult.Format("%s",RecBuf);
    }
    GetDlgItem(IDC_REPONSE)->SetWindowText(pstrResult);
}

void CGA40XX_DemoDlg::OnCancel()
{
    // TODO: Add extra cleanup here
    if(instr != 0)
    {
        viClose(instr);
    }

    if(defaultRM != 0)
    {
        viClose(defaultRM);
    }

    CDialog::OnCancel();
}
```

6. Run the program

(1) Click "Init VISA" to initialize the VISA environment

(2) Click "Connect" to connect the instrument

(3) Click "Write" to write commands to the instrument

(4) Click "Read" to read the response information from the instrument.

The result is as shown below

3-2 The Programming Based on COM Interface

3-2-1 Preparations

1. Hardware environment preparation.

In this section, the COM interface of the spectrum analyzer is used for the communication with the computer. Please use a direct serial port cable to connect the COM interface on the rear panel of the spectrum analyzer with the PC's COM1 interface.

Note: The serial port cable should at least have three signal lines: RX, TX and GND.

2. Software environment preparation

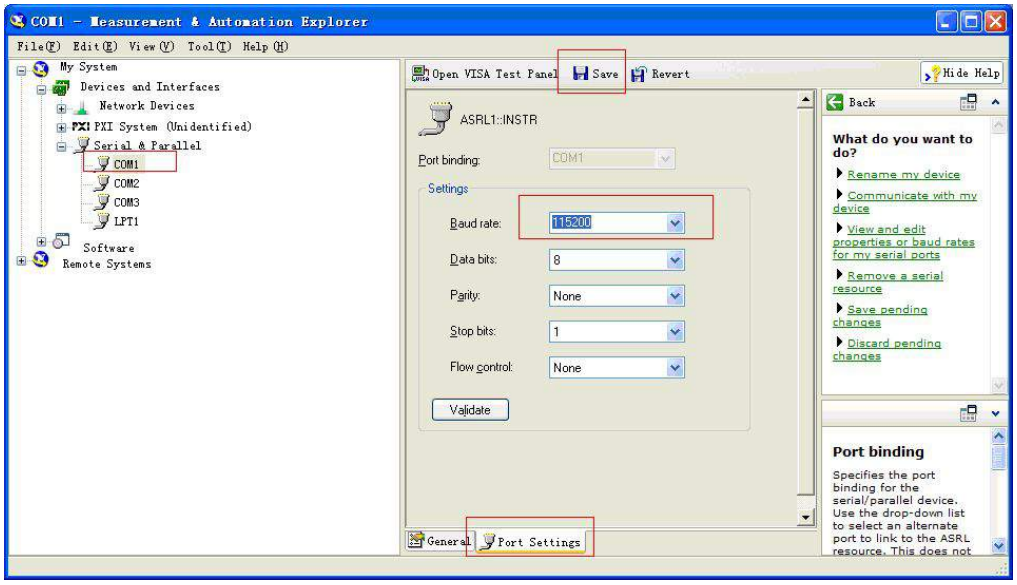
Confirm that NI VISA has been installed on the computer.

3. Use tool "Measurement & Automation Explorer" to configure NI-VISA.

The configuration steps are as follows:

1) Run “Measurement & Automation Explorer”.

2) Select "COM1" in the list on the left, then modify the baud rate to 115200 in the right property setting panel and finally click "Save". See the following figure.



3-2-2 Use Visual C + + for Programming

Directly use the program in the section 3.1.2 and there is no need to re-write it. Please note that before Connect, modify the Address into “ASRL1::INSTR”, as shown below.

Here we can see the advantages of using the VISA for programming. The same set of codes can be completely applicable to different interfaces without any modification. If a new communication interface is used, just make a simple configuration in Measurement & Automation Explorer.

www.gratten.cn

GLARUN-ATTEN TECHNOLOGY CO., LTD.
Add: Building A8, 2nd Floor,Tanglang Industrial Zone, Xili,
Nanshan, Shenzhen, 518055, P. R. China
Tel: +86-755-8602 1369
Fax: +86-755-8602-1299
E-mail: gratten@gratten.cn
Http://www.gratten.cn