USG系列信号发生器

编程手册

V 1.0

2024.12

UNI-T

保证和声明

版权

商标信息

UNI-T是优利德科技（中国）股份有限公司的注册商标

文档编号

2024.12.20

软件版本

软件升级可能更改或增加产品功能，请关注UNI-T网站获取最新版本手册或联系UNI-T升级软件。

声明

- 本公司产品受中国及其它国家和地区的专利（包括已取得的和正在申请的专利）保护。
- 本公司保留改变规格及价格的权利。
- 本手册提供的信息取代以往出版的所有资料。
- 本手册提供的信息如有变更，恕不另行通知。
- 对于本手册可能包含的错误，或因手册所提供的信息及演绎的功能以及因使用本手册而导致的任何偶然或继发的损失，UNI-T概不负责。
- 未经UNI-T事先书面许可，不得影印、复制或改编本手册的任何部分。

产品认证

UNI-T认证本产品符合中国国家产品标准和行业产品标准及ISO9001：2015标准和ISO14001：2015标准，并进一步认证本产品符合其它国际标准组织成员的相关标准。

联系我们

如您在使用此产品或本手册的过程中有任何问题或需求，可与UNI-T联系：

电子邮箱：infosh@uni-trend.com.cn

网址：http://www.uni-trend.com.cn

SCPI 命令简介

SCPI（Standard Commands for Programmable Instruments，即可编程仪器标准命令集）是一种建立在现有标准 IEEE 488.1 和 IEEE 488.2 基础上，并遵循了 IEEE754 标准中浮点运算规则、ISO646 信息交换 7 位编码符号（相当于 ASCII 编程）等多种标准的标准化仪器编程语言。本节简介 SCPI 命令的格式、符号、参数和缩写规则。

命令格式

SCPI 命令为树状层次结构，包括多个子系统，每个子系统由一个根关键字和一个或数个层次关键字构成。命令行通常以冒号“：”开始；关键字之间用冒号“：”分隔，关键字后面跟随可选的参数设置。命令关键字和第一个参数之间以空格分开。命令字符串必须以一个 <换行>（<NL>）字符结尾。命令行后面添加问号“?”通常表示对此功能进行查询。

符号说明

下面四种符号不是 SCPI 命令中的内容，不随命令发送，但是通常用于辅助说明命令中的参数。

- **大括号｛｝**

  大括号中通常包含多个可选参数，发送命令时必须选择其中一个参数。如:DISPlay:GRID:MODE ｛ FULL | GRID | CROSS | NONE}命令。

- **竖线 |**

  竖线用于分隔多个参数选项，发送命令时必须选择其中一个参数。

  如:DISPlay:GRID:MODE ｛ FULL | GRID | CROSS | NONE}命令。

- **方括号［］**

  方括号中的内容（命令关键字）是可省略的。如果省略参数，仪器将该参数设置为默认值。例如：对于:MEASure:NDUTy? [<source>]命令，[<source>]表示当前通道。

- **三角括号 < >**

  三角括号中的参数必须用一个有效值来替换。例如：以DISPlay:GRID:BRIGhtness 30的形式发送 DISPlay:GRID:BRIGhtness <count>命令。

参数说明

本手册介绍的命令中所含的参数可以分为以下 5 种类型：布尔型、整型、实型、离散型、ASCII 字符串。

- **布尔型**

  参数取值为“ON”（1）或“OFF”（0）。例如：:SYSTem:LOCK {{1 | ON} | {0 | OFF}}。

- **整型**

除非另有说明，参数在有效值范围内可以取任意整数值。注意：此时，请不要设置参数为小数格式，否则将出现异常。例如：:DISPlay:GRID:BRIGhtness <count>命令中的参数<count>可取 0 到 100 范围内的任一整数。

- **实型**

  除非另有说明，参数在有效值范围内可以取任意值。

  例如：对于 CH1，CHANnel1:OFFSet <offset>命令中的参数<offset>的取值为实型。

- **离散型**

  参数只能取指定的几个数值或字符。例如：:DISPlay:GRID:MODE { FULL | GRID | CROSS | NONE} 命令的参数只能为 FULL、GRID、 CROSS、NONE。

- **ASCII字符串**

  字符串参数实际上可包含所有 ASCII 字符集。字符串必须以配对的引号开始和结尾；可以用单引号或双引号。引号分隔符也可以作为字符串的一部分，只需键入两次并且不在中间添加任何字符，例如设置 IP：SYST:COMM:LAN:IPAD "192.168.1.10"。

简写规则

所有命令对大小写都能识别，可以全部采用大写或小写。如果要缩写，必须输完命令格式中的所有大写字母。

数据返回

数据返回分为单个数据和批量数据返回，单个数据返回相对应的参数类型，其中实型返回用科学计数法表示，e前部分小数点后面保留三位数据，e部分保留三位数据；批量数据返回必须符合IEEE 488.2 #格式的字符串数据，其格式：‘#’+ 长度所占的字符位数[固定为一个字符] + 有效数据长度的ASCII值 + 有效数据 + 结束符[‘\n’]，例如#3123xxxxxxxxxxxxxxxxxxx\n表示的具有123个字节有效批量数据返回格式，其中‘3’表示“123”占3个字符位。

SCPI 命令详解

IEEE488.2 通用命令

*CLS

- **命令格式：**

  *CLS

- **功能描述：**

  将所有事件寄存器和状态字节寄存器的值清零。

*ESE

- **命令格式：**

  *ESE <integer>

  *ESE?

- **功能描述：**

  设置标准事件状态寄存器的使能值。寄存器位的十进制总和，默认为0。例如为了启用位2（值4）、位3（值8）和位7（值128），十进制和要为140(4+8+128)。标准事件状态寄存器的位1和位6未使用。

  <integer>：整数值

- **返回格式：**

  查询返回标准事件状态寄存器的使能值，返回整数值。

- **举例：**

  *ESE 16　　启用标准事件状态寄存器的第4位

  *ESE?　　查询返回16

*ESR?

- **命令格式：**

  *ESR?

- **功能描述：**

  标准事件状态寄存器查询。事件寄存器是只读寄存器，从条件寄存器锁存事件。

- **返回格式：**

  查询返回标准事件状态寄存器的事件值，返回整数值。

- **举例：**

  *ESR?

  查询返回24，位3和位4启用

***IDN?**

- **命令格式：**

  *IDN?

- **功能描述：**

  用于查询制造商名称、产品型号、产品序列号和软件版本号。

- **返回格式：**

  制造商名称，产品型号，产品序列号，由点号分隔的软件版本号。

  注意：返回的型号要与铭牌信息一致。

- **举例：**

  UNI-TREND,USG5000M,ASA322214A002,SW V1.03.0027

***OPC**

- **命令格式：**

  *OPC

  *OPC?

- **功能描述：**

  用于在当前操作完成后将标准事件状态寄存器OPC位置1。

- **返回格式：**

  查询返回当前操作是否完成，0表示未完成，1表示完成。

- **举例：**

  *OPC　　将标准事件状态寄存器置 1

  *OPC?　　查询返回1，表示当前操作已完成，否则返回0

***RST**

- **命令格式：**

  *RST

- **功能描述：**

  用于恢复出厂设置并清空所有的错误信息及发送接收队列缓冲。

***SRE**

- **命令格式：**

  *SRE <integer>

  *SRE?

- **功能描述：**

  设置状态字节寄存器的使能值。服务请求启用为状态字节寄存器组启用使能寄存器中的位。使能寄存

器可定义事件寄存器中的哪些位将被报告给“状态字节”寄存器组。使能寄存器是可读写的。

<integer>：整数值

- **返回格式：**

  查询返回状态字节寄存器的使能值，返回整数值。

- **举例：**

  *SRE 24　　启用使能寄存器中的位3和位4

  *SRE?　　查询返回24

***STB?**

- **命令格式：**

  *STB?

- **功能描述：**

  查询状态字节寄存器的状态值。

- **返回格式：**

  查询返回状态字节寄存器的状态值，返回整数值。

- **举例：**

  *STB?

  40，状态字节寄存器位3和位5设置

SYSTem 命令

用于对射频源进行最基本的操作，主要包括全键盘锁定、系统设置数据的操作。

:SYSTem:NETAdpt

- **命令格式：**

  :SYSTem:NETAdpt {{1|ON} | {0|OFF}}

  :SYSTem:NETAdpt?

- **功能描述：**

  网络设置的适配器开关。

  1|ON：打开

  0|OFF：关闭

- **返回格式：**

  查询返回适配器开关状态，0或1。

- **举例：**

  :SYSTem:NETAdpt ON　　打开网络适配器

:SYSTem:NETAdpt? 查询返回1

:SYSTem:IPV4

- **命令格式：**

  :SYSTem:IPV4 <ip>

  :SYSTem:IPV4?

- **功能描述：**

  配置网络设置的IP地址。

  <ip>：四段点分十进制数据字符，xxx.xxx.xxx.xxx

- **返回格式：**

  查询返回当前IP地址，格式xxx.xxx.xxx.xxx。

- **举例：**

  :SYSTem:IPV4 “192.168.20.111”设置IP

  :SYSTem:IPV4? 返回192.168.20.111

:SYSTem:DHCP

- 命令格式：

  :SYSTem:DHCP {{1|ON} | {0|OFF}}

  :SYSTem:IPV4:DHCP?

- **功能描述：**

  网络设置的DHCP开关。

  1|ON：打开

  0|OFF：关闭

- **返回格式：**

  查询返回DHCP开关状态，0或1。

- **举例：**

  :SYSTem:DHCP ON 打开DHCP

  :SYSTem:DHCP? 查询返回1

:SYSTem:GATEway

- **命令格式：**

  :SYSTem:GATEway <gateway>

  :SYSTem:GATEway?

- **功能描述：**

  配置网络设置的网关。

<gateway>：四段点分十进制数据，xxx.xxx.xxx.xxx

- **返回格式：**

  查询返回网关数据，格式xxx.xxx.xxx.xxx。

- **举例：**

  :SYSTem:GATEway“192.168.20.1”　　设置网关为192.168.20.1

  :SYSTem::GATEway?　　返回192.168.20.1

:SYSTem:IPMAsk

- **命令格式：**

  :SYSTem:IPMAsk <mask>

  :SYSTem:IPMAsk?

- **功能描述：**

  配置网络设置的掩码。

  <mask>：四段点分十进制数据，xxx.xxx.xxx.xxx

- **返回格式：**

  查询返回掩码数据，格式xxx.xxx.xxx.xxx。

- **举例：**

  :SYSTem:IPMAsk“255.255.255.0”　　设置掩码为255.255.255.0

  :SYSTem:IPMAsk?　　返回255.255.255.0

:SYSTem:MAC?

- **命令格式：**

  :SYSTem:MAC?

- **功能描述：**

  查询设备MAC地址。

- **返回格式：**

  查询返回设备MAC地址，返回字符。

- **举例：**

  :SYSTem:MAC?　　返回3E:A4:20:3D:82:83

:SYSTem:DATE

- **命令格式：**

  :SYSTem:DATE <yyyymmdd>

  :SYSTem:DATE?

- **功能描述：**

设置系统日期，格式为年月日，中间没有分割字符。例如设置时间2000年1月1日，参数为20000101

- **返回格式：**

  查询返回日期。

- **举例：**

  :SYSTem:DATE?　　返回20000101 (2000年1月1日)

:SYSTem:BACKlight

- **命令格式：**

  :SYSTem:BACKlight <integer>

  :SYSTem:BACKlight?

- **功能描述：**

  调整系统背光亮度。

  <integer>：连续整数，默认单位%，取值范围0-100

- **返回格式：**

  查询返回系统背光亮度等级，单位为%。

- **举例：**

  :SYSTem:BACKlight 80　　设置系统背光亮度80%

  :SYSTem:BACKlight?　　查询返回80

:SYSTem:CFORmat

- **命令格式：**

  :SYSTem:CFORmat {HR12|HR24}

  :SYSTem:CFORmat?

- **功能描述：**

  设置系统时间显示格式。

  HR12：12小时制

  HR24：24小时制

- **返回格式：**

  查询返回系统时间显示格式，HR12或HR24。

- **举例：**

  :SYSTem:CFORmat HR24　　设置24小时显示格式

  :SYSTem:CFORmat?　　查询返回HR24

:SYSTem:LANGuage

- **命令格式：**

:SYSTem:LANGuage {CHINese|ENGLish|GERMan}

:SYSTem:LANGuage?

- **功能描述：**

  设置系统显示语言。

  CHINese：中文

  ENGLish：英文

  GERMan：德文

- **返回格式：**

  查询返回系统显示语言，CHINese、ENGLish或GERMan。

- **举例：**

:SYSTem:LANGuage CHINese	设置中文为系统显示语言
:SYSTem:LANGuage?	查询返回CHINese

:SYSTem:LOCK

- **命令格式：**

  :SYSTem:LOCK {{1|ON} | {0|OFF}}

  :SYSTem:LOCK?

- **功能描述：**

  用于锁定或者解锁键盘按键和触屏输入。

  1|ON：锁定

  0|OFF：解锁

- **返回格式：**

  查询返回键盘按键和触屏输入锁定状态，0表示解锁，1表示锁定。

- **举例：**

:SYSTem:LOCK ON	键盘按键和触屏输入锁定
:SYSTem:LOCK OFF	键盘按键和触屏输入解锁
:SYSTem:LOCK?	查询返回0，表示解锁

:SYSTem:FORMat

- **命令格式：**

  :SYSTem:FORMat {BMP|PNG}

  :SYSTem:FORMat?

- **功能描述：**

  设置系统截图图片保存格式。

- **返回格式：**

查询系统图片格式，BMP、PNG。

- **举例：**

:SYSTem:FORMat BMP　　系统图片格式BMP

:SYSTem:FORMat?　　查询返回BMP

:SYSTem:PICInvert

- **命令格式：**

:SYSTem:PICInvert {{1|ON} | {0|OFF}}

:SYSTem:PICInvert?

- **功能描述：**

截屏反色开关。

1|ON：打开

0|OFF：关闭

- **返回格式：**

查询返回截屏反色开关状态，0或1。

- **举例：**

:SYSTem:PICInvert ON　　打开截屏反色

:SYSTem:PICInvert?　　查询返回1

:SYSTem:TIME

- **命令格式：**

:SYSTem:TIME <hhmmss>

:SYSTem:TIME?

- **功能描述：**

设置系统时间，格式为24小时制。例如设置时间12点12分12秒，参数为121212

- **返回格式：**

查询返回时间。

- **举例：**

:SYSTem:TIME?　　返回130101 (13点1分1秒)

:SYSTem:TEMPerature?

- **命令格式：**

:SYSTem:TEMPerature?

- **功能描述：**

查询设备温度。

- **返回格式：**

  查询返回设备温度单位°C。

- **举例：**

  :SYSTem:TEMPerature? 返回30°C。

:SYSTem:GPIB

- **命令格式：**

  :SYSTem:GPIB<value>

- **功能描述：**

  设置GPIB地址，无单位。

- **返回格式：**

  查询返回GPIB地址。

- **举例：**

  :SYSTem:GPIB 2 设置GPIB地址为2。

  :SYSTem:GPIB? 返回2。

KEY 命令

:KEY:<key>

- **命令格式：**

  :KEY:<key>

  :KEY:<key>:LOCK { {1 | ON} | {0 | OFF} }

  :KEY:<key>:LOCK?

  :KEY:LED?

- **功能描述：**

  实现按键功能以及按键的锁定/解锁。按键<key>的定义和描述，详见附录1：<key>列表。

- **返回格式：**

  查询返回按键锁定状态或者具有LED按键灯状态。

  LED灯状态：0表示不亮，1表示亮（绿灯）

- **举例：**

  :KEY:FREQ 进入频率设置菜单

  :KEY:LED? 查询返回LED灯状态，0表示不亮

:KEY:LED?

- **命令格式：**

:KEY:LED?

- **功能描述：**

  查询所有带灯按键灯状态。

- **返回格式：**

  查询返回所有按键的灯状态，返回字符序列，每个字符代表一个按键灯状态，亮为ASCII'1',不亮为ASCII'0'。一共4个按键带灯，按顺序是Touch/Lock按键，LF按键，MOD按键，RF按键。返回4位由'1'或'0'组成的字符串。

- **举例：**

  Touch/Lock按键亮，LF按键，MOD按键和RF按键不亮，返回ASCII字符串1000。

:KEY:LOCK?

- **命令格式：**

  :KEY:LOCK?

- **功能描述：**

  查询所有按键的锁定状态。

- **返回格式：**

  查询返回所有按键的锁定状态，返回字符序列，每个字符代表一个按键的锁定状态，锁定为ASCII'1'，未锁定为ASCII'0'，按照附录1：<key>列表顺序返回锁定状态。

- **举例：**

  一共41个按键，只有第4、5两个按键锁定，返回ASCII字符串

  00011000000000000000000000000000000000000

各功能模块命令

调制源，模拟调制，射频，函数生成，调制输入。

SOURce命令

调制源

:MOGEn:STATe

- **命令格式：**

  [:SOURce]:MOGEn:STATe {{0 | OFF | {1 | ON}}

  [:SOURce]:MOGEn:STATe?

- **功能描述：**

  调制源开关。

- **返回格式：**

  查询返回当前状态0或1。

- **举例：**

  :SOURce:MOGEn:STATe ON　　打开调制源

  :SOURce:MOGEn:STATe?　　查询返回1

:MOGEn:TYPE

- **命令格式：**

  [:SOURce]:MOGEn:TYPE {{0 | SINE} | {1 | SQUAre} | {2 | RAMP}}

  [:SOURce]:MOGEn:TYPE?

- **功能描述：**

  调制源波形选择。

- **返回格式：**

  查询返回当前波形SINE、SQUAre或RAMP。

- **举例：**

  :SOURce:MOGEn:TYPE SQUAre　　调制源波形选择方波

  :SOURce:MOGEn:TYPE?　　查询返回SQUAre

:MOGEn:FREQ

- **命令格式：**

  [:SOURce]:MOGEn:FREQ <value><unit>

  [:SOURce]:MOGEn:FREQ?

- **功能描述：**

  调制源频率。

- **返回格式：**

  查询返回频率值，单位为Hz。

- **举例：**

  :SOURce:MOGEn:FREQ 1kHz　　设置调制源频率1kHz

  :SOURce:MOGEn:FREQ?　　查询返回1000

:MOGEn:AMPT

- **命令格式：**

  [:SOURce]:MOGEn:AMPT <value><unit>

  [:SOURce]:MOGEn:AMPT?

- **功能描述：**

调制源幅度，单位包括Vpp,mVpp,Vrms,mVrms。

- **返回格式：**

  查询返回当前幅度值和单位。

- **举例：**

  :SOURce:MOGEn:AMPT 100mVpp　　设置调制源幅度100mVpp

  :SOURce:MOGEn:AMPT?　　查询返回100mVpp

:MOGEn:PHASe

- **命令格式：**

  [:SOURce]:MOGEn:PHASe <value><unit>

  [:SOURce]:MOGEn:PHASe?

- **功能描述：**

  调制源相位，单位deg,rad。

- **返回格式：**

  查询返回当前相位值和单位。

- **举例：**

  :SOURce:MOGEn:PHASe 30deg　　设置调制源相位30deg

  :SOURce:MOGEn:PHASe ?　　查询返回30deg

MOGEn:DCOFst

- **命令格式：**

  [:SOURce]:MOGEn:DCOFst <value><unit>

  [:SOURce]:MOGEn:DCOFst?

- **功能描述：**

  调制源直流偏移，单位V,mV。

- **返回格式：**

  查询返回当前直流偏移值和单位。

- **举例：**

  :SOURce:MOGEn:DCOFst 500mv　　设置调制源直流偏移500mv

  :SOURce:MOGEn:DCOFst?　　查询返回500mv

模拟调制

:MOD:STATe

- **命令格式：**

  [:SOURce]:MOD:STATe {{1|ON} | {0|OFF}}

[:SOURce]:MOD:STATe?

- **功能描述：**

  模拟调制开关。

- **返回格式：**

  查询返回当前状态0或1。

- **举例：**

  :SOURce:MOD:STATe ON 打开模拟调制

  :SOURce:MOD:STATe? 查询返回1

:MOD:AM:EN

- **命令格式：**

  [:SOURce]:MOD:AM:EN {{1|ON} | {0|OFF}}

  [:SOURce]:MOD:AM:EN?

- **功能描述：**

  开启模拟调幅。

- **返回格式：**

  查询返回当前状态0或1。

- **举例：**

  [:SOURce]:MOD:AM:EN ON 打开模拟调幅

  [:SOURce]:MOD:AM:EN? 查询返回1

:MOD:AM:SOURce

- **命令格式：**

  [:SOURce]:MOD:AM:SOURce {{0 | INT} | {1 | EXT} | {2 | INT+EXT}}

  [:SOURce]:MOD:AM:SOURce?

- **功能描述：**

  模拟调幅中的调制源选择，内部，外部或内部加外部。

- **返回格式：**

  查询返回调制源，INT，EXT或INT+EXT。

- **举例：**

  :SOURce:MOD:AM:SOURce INT 调制源选择内部

  :SOURce:MOD:AM:SOURce? 查询返回INT

:MOD:AM:DEPTh

- **命令格式：**

[:SOURce]:MOD:AM:DEPTh <value>

[:SOURce]:MOD:AM:DEPTh?

- **功能描述：**

  设置调制深度，value是浮点数。

- **返回格式：**

  查询返回调制深度数值单位是%。

- **举例：**

  :SOURce:MOD:AM:DEPTh 60.5　　设置调制深度60.5%

  :SOURce:MOD:AM:DEPTh?　　查询返回值60.5

:MOD:FM:EN

- **命令格式：**

  [:SOURce]:MOD:FM:EN {{1|ON} | {0|OFF}}

  [:SOURce]:MOD:FM:EN?

- **功能描述：**

  设置模拟调频开关。

- **返回格式：**

  查询返回模拟调频开关状态。

- **举例：**

  :SOURce:MOD:FM:EN ON　　打开模拟调频

  :SOURce:MOD:FM:EN?　　查询返回1

:MOD:FM:SOURce

- **命令格式：**

  [:SOURce]:MOD:FM:SOURce {{0 | INT} | {1 | EXT} | {2 | INT+EXT}}

  [:SOURce]:MOD:FM:SOURce?

- **功能描述：**

  模拟调频中的调制源选择，内部，外部或内部加外部。

- **返回格式：**

  查询返回模拟调频的调制源类型，返回值为INT，EXT或INT+EXT。

- **举例：**

  :SOURce:MOD:FM:SOURce INT　　调制源选择内部

  :SOURce:MOD:FM:SOURce?　　查询返回INT

:MOD:FM:DEV

- **命令格式：**

  [:SOURce]:MOD:FM:DEV <value><unit>

  [:SOURce]:MOD:FM:DEV?

- **功能描述：**

  设置模拟调频的频率偏移。

- **返回格式：**

  查询返回当前频率偏移值，单位Hz。

- **举例：**

  :SOURce:MOD:FM:DEV 1kHz　　设置频率偏移1kHz

  :SOURce:MOD:FM:DEV?　　查询返回值1.000000e+03

:MOD:PM:EN

- **命令格式：**

  [:SOURce]:MOD:PM:EN {{1|ON} | {0|OFF}}

  [:SOURce]:MOD:PM:EN?

- **功能描述：**

  打开关闭模拟调相。

- **返回格式：**

  查询返回模拟调相开关状态。

- **举例：**

  :SOURce:MOD:PM:EN ON　　打开模拟调相

  :SOURce:MOD:PM:EN?　　查询返回1

:MOD:PM:SOURce

- **命令格式：**

  [:SOURce]:MOD:PM:SOURce {{0 | INT} | {1 | EXT} | {2 | INT+EXT}}

  [:SOURce]:MOD:PM:SOURce?

- **功能描述：**

  模拟调相中的调制源选择，内部，外部或内部加外部。

- **返回格式：**

  返回调制源模式INT，EXT或 INT+EXT。

- **举例：**

  :SOURce:MOD:PM:SOURce INT　　设置模拟调相调制源为内部

  :SOURce:MOD:PM:SOURce?　　查询返回INT

:MOD:PM:PHASe

- **命令格式：**

  [:SOURce]:MOD:PM:PHASe <value><unit>

  [:SOURce]:MOD:PM:PHASe?

- **功能描述：**

  设定相位偏移，单位度(deg)和弧度(reg)。

- **返回格式：**

  查询返回相位偏移值和单位。

- **举例：**

  :SOURce:MOD:PM:PHASe 50deg　　设置相位偏移50度

  :SOURce:MOD:PM:PHASe?　　查询返回50deg

:MOD:PULSe:EN

- **命令格式：**

  [:SOURce]:MOD:PULSe:EN {{1|ON} | {0|OFF}}

  [:SOURce]:MOD:PULSe:EN?

- **功能描述：**

  设定脉冲调制开关。

- **返回格式：**

  查询返回脉冲开关状态，返回值为0或1。

- **举例：**

  :SOURce:MOD:PULSe:EN ON　　打开脉冲调制

  :SOURce:MOD:PULSe:EN?　　查询返回1

:MOD:PULSe:TYPE

- **命令格式：**

  [:SOURce]:MOD:PULSe:TYPE {{FREErun | 0} | {SQUAre | 1} | {TRIGgered | 2} | {ADJDoublet | 3} | {TRIDoublet | 4} | {GATEd | 5} | {EXTPulse | 6} | {PULSetrain | 7}}

  [:SOURce]:MOD:PULSe:TYPE?

- **功能描述：**

  设定脉冲调制模式。

- **返回格式：**

  查询返回FREErun ,SQUAre ,TRIGgered ,ADJDoublet ,TRIDoublet ,GATEd ,EXTPulse ,PULSetrain。

- **举例：**

  :SOURce:MOD:PULSe:TYPE FREErun　　脉冲调制模式为自由运行

:SOURce:MOD:PULSe:TYPE? 查询返回FREErun

:MOD:PULSe:PERIod

- **命令格式：**

  [:SOURce]:MOD:PULSe:PERIod <value><unit>

  [:SOURce]:MOD:PULSe:PERIod?

- **功能描述：**

  设定脉冲调制周期,，单位包括s,ms,us,ns。

- **返回格式：**

  查询返回脉冲调制周期值，单位为秒。

- **举例：**

  :SOURce:MOD:PULSe:PERIod 60ns 设置脉冲周期60ns

  :SOURce:MOD:PULSe:PERIod? 查询返回6.000000e-08

:MOD:PULSe:RATE

- **命令格式：**

  [:SOURce]:MOD:PULSe:RATE <value><unit>

  [:SOURce]:MOD:PULSe:RATE?

- **功能描述：**

  设定脉冲调制方波模式下的速率。

- **返回格式：**

  查询返回速率的值，单位Hz。

- **举例：**

  :SOURce:MOD:PULSe:RATE 1kHz 设置方波速率1kHz

  :SOURce:MOD:PULSe:RATE? 查询返回值1000

:MOD:PULSe:DELAy

- **命令格式：**

  [:SOURce]:MOD:PULSe:DELAy <value><unit>

  [:SOURce]:MOD:PULSe:DELAy?

- **功能描述：**

  设定脉冲延迟时间，单位包括s,ms,us,ns。

- **返回格式：**

  查询返回脉冲延迟时间，单位秒。

- **举例：**

:SOURce:MOD:PULSe:DELAy 20ns　　设置延迟时间20

:SOURce:MOD:PULSe:DELAy?　　查询返回2.000000e-08

:MOD:PULSe:WIDTh

- **命令格式：**

  [:SOURce]:MOD:PULSe:WIDTh <value><unit>

  [:SOURce]:MOD:PULSe:WIDTh?

- **功能描述：**

  设置脉冲脉宽，单位包括s,ms,us,ns。

- **返回格式：**

  返回脉冲值，单位秒。

- **举例：**

  :SOURce:MOD:PULSe:WIDTh 50ns　　设置脉宽50ns

  :SOURce:MOD:PULSe:WIDTh?　　查询返回5.000000e-08

:MOD:PULSe:DELAys

- **命令格式：**

  [:SOURce]:MOD:PULSe:DELAys <value><unit>

  [:SOURce]:MOD:PULSe:DELAys?

- **功能描述：**

  设定脉冲延迟2的时间，单位包括s,ms,us,ns。

- **返回格式：**

  查询返回脉冲延迟2的时间，单位秒。

- **举例：**

  :SOURce:MOD:PULSe:DELAys 20ns　　设置延迟2的时间为20ns

  :SOURce:MOD:PULSe:DELAys?　　查询返回2.000000e-08

:MOD:PULSe:WIDThs

- **命令格式：**

  [:SOURce]:MOD:PULSe:WIDThs <value><unit>

  [:SOURce]:MOD:PULSe:WIDThs?

- **功能描述：**

  设置脉冲脉宽2，单位包括s,ms,us,ns。

- **返回格式：**

  返回脉冲脉宽2的值，单位秒。

- **举例：**

  :SOURce:MOD:PULSe:WIDThs 50ns　　设置脉宽2的值为50ns

  :SOURce:MOD:PULSe:WIDThs?　　查询返回5.000000e-08

:MOD:PULSe:SYNCwidth

- **命令格式：**

  [:SOURce]:MOD:PULSe:SYNCwidth <value><unit>

  [:SOURce]:MOD:PULSe:SYNCwidth?

- **功能描述：**

  设置同步脉宽，单位包括s,ms,us,ns。

- **返回格式：**

  查询返回同步脉宽，单位秒。

- **举例：**

  :SOURce:MOD:PULSe:SYNCwidth 20ns　　设置同步脉宽20ns

  :SOURce:MOD:PULSe:SYNCwidth?　　查询返回值2.000000e-08

:MOD:PULSe:EXTPolarity

- **命令格式：**

  [:SOURce]:MOD:PULSe:EXTPolarity {{1|INVErt} | {0|NORMal}}

  [:SOURce]:MOD:PULSe:EXTPolarity?

- **功能描述：**

  设置外部极性为正常或反转。

- **返回格式：**

  查询返回值0或1。

- **举例：**

  :SOURce:MOD:PULSe:EXTPolarity INVErt　　设置外部极性为反转

  :SOURce:MOD:PULSe:EXTPolarity?　　查询返回值1

:MOD:PULSe:TRIGgermode

- **命令格式：**

  [:SOURce]:MOD:PULSe:TRIGgermode {FREErun | TRIGgered | GATEd}

  [:SOURce]:MOD:PULSe:TRIGgermode?

- **功能描述：**

  在脉冲类型是脉冲串时，设置触发模式，自由运行，外部触发，选通。

- **返回格式：**

查询返回触发模式，返回值0,1或2。

- **举例：**

:SOURce:MOD:PULSe:TRIGgermode FREErun　　设置触发模式为自由运行

:SOURce:MOD:PULSe:TRIGgermode?　　查询返回值为0

:MOD:PULSe:LIST

- **命令格式：**

[:SOURce]:MOD:PULSe:LIST {{1|ON} | {0|OFF}}

[:SOURce]:MOD:PULSe:LIST?

- **功能描述：**

脉冲串编辑列表数据发送使能，关闭状态编辑列表不会发送数据，开启状态编辑列表或发送数据。

- **返回格式：**

查询返回脉冲串编辑列表开关状态，0或1。

- **举例：**

:SOURce:MOD:PULSe:LIST ON　　打开脉冲串编辑列表使能

:SOURce:MOD:PULSe:LIST?　　查询返回值1

:MOD:PULSe:LIST:ONTIme

- **命令格式：**

[:SOURce]:MOD:PULSe:LIST:ONTIme <value><unit>

[:SOURce]:MOD:PULSe:LIST:ONTIme?

- **功能描述：**

设置脉冲串列表当前行高电平时间，单位包括s,ms,us,ns。

- **返回格式：**

查询返回当前行高电平时间，单位秒。

- **举例：**

:SOURce:MOD:PULSe:LIST:ONTIme 20ns　　设置高电平时间20ns

:SOURce:MOD:PULSe:LIST:ONTIme?　　查询返回值2.000000e-08

:MOD:PULSe:LIST:OFFTime

- **命令格式：**

[:SOURce]:MOD:PULSe:LIST:OFFTime <value><unit>

[:SOURce]:MOD:PULSe:LIST:OFFTime?

- **功能描述：**

设置脉冲串列表当前行低电平时间，单位包括s,ms,us,ns。

- **返回格式：**

  查询返回当前行低电平时间，单位秒。

- **举例：**

  :SOURce:MOD:PULSe:LIST:OFFTime 20ns　　设置低电平时间20ns

  :SOURce:MOD:PULSe:LIST:OFFTime?　　查询返回值2.000000e-08

:MOD:PULSe:LIST:REPEat

- **命令格式：**

  [:SOURce]:MOD:PULSe:LIST:REPEat <value>

  [:SOURce]:MOD:PULSe:LIST:REPEat?

- **功能描述：**

  设置脉冲串列表当前行重复次数，没有单位。

- **返回格式：**

  查询返回脉冲串列表当前行重复次数。

- **举例：**

  :SOURce:MOD:PULSe:LIST:REPEat 10　　设置重复次数10次

  :SOURce:MOD:PULSe:LIST:REPEat?　　查询返回值10

:MOD:PULSe:LIST:CURRow

- **命令格式：**

  [:SOURce]:MOD:PULSe:LIST:CURRow <value>

- **功能描述：**

  设置脉冲串列表当前可以编辑的第几行。

- **返回格式：**

  无返回。

- **举例：**

  :SOURce:MOD:PULSe:LIST:CURRow 1　　设置当前可编辑的第1行

:MOD:PULSe:LIST:ADDRow

- **命令格式：**

  [:SOURce]:MOD:PULSe:LIST:ADDRow <value>

- **功能描述：**

  脉冲串编辑列表增加几行，无单位。

- **返回格式：**

  无返回。

- **举例：**

  :SOURce:MOD:PULSe:LIST:ADDRow 2　　设置脉冲串编辑列表增加2行

:MOD:PULSe:LIST:SUBRow

- **命令格式：**

  [:SOURce]:MOD:PULSe:LIST:SUBRow <value>

- **功能描述：**

  脉冲串编辑列表减少几行，无单位。

- **返回格式：**

  无返回。

- **举例：**

  :SOURce:MOD:PULSe:LIST:SUBRow 2　　设置脉冲串编辑列表减少2行

射频

:OUTPut

- **命令格式：**

  [:SOURce]:OUTPut[:STATe] {{1|ON} | {0|OFF}}

  [:SOURce]:OUTPut[:STATe]?

- **功能描述：**

  射频输出开关使能。

- **返回格式：**

  查询返回开关状态，返回值为0,或1。

- **举例：**

  :SOURce:OUTPut:STATe ON　　设置射频输出开关使能打开

  :SOURce:OUTPut:STATe?　　查询返回值1

:FREQuency

- **命令格式：**

  [:SOURce]:FREQuency[:CW] <value><unit>

  [:SOURce]:FREQuency[:CW]?

- **功能描述：**

  设置射频输出频率。

- **返回格式：**

  查询返回频率值，单位Hz。

- **举例：**

:SOURce:FREQuency:CW 1GHz　　设置输出1GHz

:SOURce:FREQuency:CW?　　查询返回值1e+9

:FREQuency:OFFSet

- **命令格式：**

  [:SOURce]:FREQuency:OFFSet <value><unit>

  [:SOURce]:FREQuency:OFFSet?

- **功能描述：**

  设置偏移频率。

- **返回格式：**

  查询返回频率偏移值，单位Hz。

- **举例：**

  :SOURce:FREQuency:OFFSet 1kHz　　设置频率偏移为1kHz

  :SOURce:FREQuency:OFFSet?　　查询返回值1000

:FREQuency:OFFSet:STATe

- **命令格式：**

  [:SOURce]:FREQuency:OFFSet:STATe {{1|ON} | {0|OFF}}

  [:SOURce]:FREQuency:OFFSet:STATe?

- **功能描述：**

  设置偏移频率使能开关。

- **返回格式：**

  查询返回频率偏移使能状态，返回值1或0。

- **举例：**

  :SOURce:FREQuency:OFFSet:STATe ON　　打开频率偏移使能

  :SOURce:FREQuency:OFFSet:STATe?　　查询返回值1

:RADIo:PHASeofst

- **命令格式：**

  [:SOURce]:RADIo:PHASeofst <value><unit>

  [:SOURce]:RADIo:PHASeofst?

- **功能描述：**

  设置相位偏移，单位包括deg,rad。

- **返回格式：**

  查询返回相位偏移和单位。

- **举例：**

  :SOURce:RADIo:PHASeofst 30deg　　设置相位偏移30度

  :SOURce:RADIo:PHASeofst?　　查询返回值30deg

:RADIo:INTBcal

- **命令格式：**

  [:SOURce]:RADIo:INTBcal <value><unit>

  [:SOURce]:RADIo:INTBcal?

- **功能描述：**

  设置内部TB校准，单位为ppb,ppm。

- **返回格式：**

  查询返回内部TB校准值和单位。

- **举例：**

  :SOURce:RADIo:INTBcal 10ppb　　设置内部TB校准值10ppb

  :SOURce:RADIo:INTBcal?　　查询返回值10ppb

:RADIo:OSREf

- **命令格式：**

  [:SOURce]:RADIo:OSREf {{0 | INT} | {1 | EXT} | {2 | AUTO}}

  [:SOURce]:RADIo:OSREf?

- **功能描述：**

  设置参考源选项。

- **返回格式：**

  查询返回值为0,1或2。

- **举例：**

  :SOURce:RADIo:OSREf EXT　　设置参考源为外部

  :SOURce:RADIo:OSREf?　　查询返回值1

:FREQuency:REFerence:STATe

- **命令格式：**

  [:SOURce]:FREQuency:REFerence:STATe {{1|ON} | {0|OFF}}

  [:SOURce]:FREQuency:REFerence:STATe?

- **功能描述：**

  参考频率使能。

- **返回格式：**

查询返回参考频率使能状态，返回值0或1。

- **举例：**

:SOURce:FREQuency:REFerence:STATe ON　　设置参考频率打开

:SOURce:FREQuency:REFerence:STATe?　　查询返回值为1

:PHASe:REFerence

- **命令格式：**

[:SOURce]:PHASe:REFerence {{1|ON} | {0|OFF}}

[:SOURce]:PHASe:REFerence?

- **功能描述：**

使能射频相位参考。

- **返回格式：**

查询返回值为0或1。

- **举例：**

:SOURce:PHASe:REFerence ON　　打开相位参考使能

:SOURce:PHASe:REFerence?　　查询返回值为1

:ROSCillator:OVEN:STATe

- **命令格式：**

[:SOURce]:ROSCillator:OVEN:STATe

[:SOURce]:ROSCillator:OVEN:STATe?

- **功能描述：**

10MHz输出使能。

- **返回格式：**

查询返回使能状态，返回值为0或1。

- **举例：**

:SOURce:ROSCillator:OVEN:STATe ON　　打开10MHz输出使能

:SOURce:ROSCillator:OVEN:STATe?　　查询返回值为1

:POWer

- **命令格式：**

[:SOURce]:POWer[:LEVel][:IMMediate][:AMPLitude] <value><unit>

[:SOURce]:POWer[:LEVel][:IMMediate][:AMPLitude]?

- **功能描述：**

设置输出幅度，单位为dBm,dBuV,uV,mV,V,nW,uW,mW。

- **返回格式：**

  查询返回幅度值，单位dBm。

- **举例：**

  :SOURce:POWer:LEVel:IMMediate:AMPLitude 1V　　设置幅度值1V

  :SOURce:POWer:LEVel:IMMediate:AMPLitude?　　查询返回值13

:POWer:OFFSet

- **命令格式：**

  [:SOURce]:POWer[:LEVel][:IMMediate]:OFFSet <value><unit>

  [:SOURce]:POWer[:LEVel][:IMMediate]:OFFSet?

- **功能描述：**

  设置幅度偏移值，单位为dB。

- **返回格式：**

  返回幅度偏移值和单位。

- **举例：**

  :SOURce:POWer:LEVel:IMMediate:OFFSet 1dB　　设置幅度偏移1dB

  :SOURce:POWer:LEVel:IMMediate:OFFSet?　　查询返回值为1dB

:RADIo:MAXPower

- **命令格式：**

  [:SOURce]:RADIo:MAXPower <value><unit>

  [:SOURce]:RADIo:MAXPower?

- **功能描述：**

  设置用户最大功率，单位为dBm,dBuV,uV,mV,V,nW,uM,mW。

- **返回格式：**

  查询返回当前设定值，单位为dBm。

- **举例：**

  :SOURce:RADIo:MAXPower 1V　　设置用户最大功率1V

  :SOURce:RADIo:MAXPower?　　查询返回13

:POWer:ATTenuation

- **命令格式：**

  [:SOURce]:POWer:ATTenuation <value><unit>

  [:SOURce]:POWer:ATTenuation?

- **功能描述：**

设置衰减值，单位dB。

- **返回格式：**

  返回当前设定值和单位。

- **举例：**

  :SOURce:POWer:ATTenuation 10dB　　设置衰减10dB

  :SOURce:POWer:ATTenuation?　　查询返回值10dB

:POWer:ALC:LEVel

- **命令格式：**

  [:SOURce]:POWer:ALC:LEVel <value><unit>

  [:SOURce]:POWer:ALC:LEVel?

- **功能描述：**

  设置ALC幅度，单位为dBm,dBuV,uV,mV,V,nW,uW,mW。

- **返回格式：**

  返回ALC幅度值，单位为dBm。

- **举例：**

  :SOURce:POWer:ALC:LEVel 10dBm　　设置ALC幅度为10dBm

  :SOURce:POWer:ALC:LEVel ？　　查询返回值10

:POWer:ALC

- **命令格式：**

  [:SOURce]:POWer:ALC[:STATe] {{0 | ON} | {1 | OFF} | {2 | AUTO}}

  [:SOURce]:POWer:ALC[:STATe]?

- **功能描述：**

  设置ALC状态。

- **返回格式：**

  返回ALC状态值，ON,OFF,AUTO。

- **举例：**

  :SOURce:POWer:ALC:STATe AUTO　　设置ALC为AUTO模式

  :SOURce:POWer:ALC:STATe?　　查询返回值AUTO

:RADIo:LEVElofst:EN

- **命令格式：**

  [:SOURce]:RADIo:LEVElofst:EN {{1|ON} | {0|OFF}}

  [:SOURce]:RADIo:LEVElofst:EN?

- **功能描述：**

  设置幅度偏移使能。

- **返回格式：**

  查询返回偏移使能状态，返回值0或1。

- **举例：**

  :SOURce:RADIo:LEVElofst:EN ON　　打开幅度偏移使能

  :SOURce:RADIo:LEVElofst:EN?　　查询返回1

:POWer:REFerence:STATe

- **命令格式：**

  [:SOURce]:POWer:REFerence:STATe {{1|ON} | {0|OFF}}

  [:SOURce]:POWer:REFerence:STATe?

- **功能描述：**

  幅度参考使能。

- **返回格式：**

  查询返回使能状态，返回值为0或1。

- **举例：**

  :SOURce:POWer:REFerence:STATe ON　　打开幅度参考使能

  :SOURce:POWer:REFerence:STATe?　　查询返回值1

:RADIo:MAXPower:EN

- **命令格式：**

  [:SOURce]:RADIo:MAXPower:EN {{1|ON} | {0|OFF}}

  [:SOURce]:RADIo:MAXPower:EN?

- **功能描述：**

  设置用户最大功率使能。

- **返回格式：**

  查询使能状态，返回值为0或1。

- **举例：**

  :SOURce:RADIo:MAXPower:EN ON　　打开用户最大功率使能

  :SOURce:RADIo:MAXPower:EN?　　查询返回值1

:RADIo:ATTEnhold:EN

- **命令格式：**

  [:SOURce]:RADIo:ATTEnhold:EN {{1|ON} | {0|OFF}}

[:SOURce]:RADIo:ATTEnhold:EN?

- **功能描述：**

  手动衰减使能。

- **返回格式：**

  查询返回手动衰减使能状态，返回值为0或1。

- **举例：**

  :SOURce:RADIo:ATTEnhold:EN ON　　打开手动衰减使能

  :SOURce:RADIo:ATTEnhold:EN?　　查询返回值1

:RADIo:LEVElflat:EN

- **命令格式：**

  [:SOURce]:RADIo:LEVElflat:EN {{1|ON} | {0|OFF}}

  [:SOURce]:RADIo:LEVElflat:EN?

- **功能描述：**

  平坦度编辑表格使能。

- **返回格式：**

  返回使能状态，返回值为1或0。

- **举例：**

  :SOURce:RADIo:LEVElflat:EN ON　　平坦度编辑表格使能打开

  :SOURce:RADIo:LEVElflat:EN?　　查询返回值1

:RADIo:FLATness:FREQ

- **命令格式：**

  [:SOURce]:RADIo:FLATness:FREQ <value><unit>

  [:SOURce]:RADIo:FLATness:FREQ?

- **功能描述：**

  设置当前编辑行的频率。

- **返回格式：**

  查询返回当前编辑行的频率，返回单位Hz。

- **举例：**

  :SOURce:RADIo:FLATness:FREQ 10kHz　　设置当前编辑行的频率10kHz

  :SOURce:RADIo:FLATness:FREQ?　　查询返回值为10000

:RADIo:FLATness:CORR

- **命令格式：**

[:SOURce]:RADIo:FLATness:CORR <value><unit>

[:SOURce]:RADIo:FLATness:CORR?

- **功能描述：**

  设置当前编辑行修正值。

- **返回格式：**

  返回值单位为dB。

- **举例：**

  :SOURce:RADIo:FLATness:CORR 10dB　　设置当前编辑行修正值为10dB

  :SOURce:RADIo:FLATness:CORR?　　查询返回值10

:RADIo:FLATness:CURRow

- **命令格式：**

  [:SOURce]:RADIo:FLATness:CURRow <value>

- **功能描述：**

  设置编辑当前表格显示的第几行。

- **返回格式：**

  无返回。

- **举例：**

  :SOURce:RADIo:FLATness:CURRow 2　　设置编辑当前表格显示的第二行

:RADIo:FLATness:ADDRow

- **命令格式：**

  [:SOURce]:RADIo:FLATness:ADDRow <value>

- **功能描述：**

  从表格最后一行开始，增加一行。

- **返回格式：**

  无返回值。

- **举例：**

  :SOURce:RADIo:FLATness:ADDRow 2　　表格从最后一行增加2行

:RADIo:FLATness:SUBRow

- **命令格式：**

  [:SOURce]:RADIo:FLATness:SUBRow <value>

- **功能描述：**

  从表格最后一行开始，减少2行。

- **返回格式：**

  无返回值。

- **举例：**

  :SOURce:RADIo:FLATness:SUBRow 2　　表格从最后一行减少2行

:RADIo:SWEEp:FREQ:EN

- **命令格式：**

  [:SOURce]:RADIo:SWEEp:FREQ:EN {{1|ON} | {0|OFF}}

  [:SOURce]:RADIo:SWEEp:FREQ:EN?

- **功能描述：**

  设置扫频使能开关。

- **返回格式：**

  查询返回扫频使能开关状态。

- **举例：**

  :SOURce:RADIo:SWEEp:FREQ:EN ON　　打开扫频使能

  :SOURce:RADIo:SWEEp:FREQ:EN?　　查询返回值1

:RADIo:SWEEp:LEVEl:EN

- **命令格式：**

  [:SOURce]:RADIo:SWEEp:LEVEl:EN {{1|ON} | {0|OFF}}

  [:SOURce]:RADIo:SWEEp:LEVEl:EN?

- **功能描述：**

  设置扫幅使能开关。

- **返回格式：**

  查询返回扫幅使能开关状态，返回值为1或0。

- **举例：**

  :SOURce:RADIo:SWEEp:LEVEl:EN ON　　打开扫幅使能

  :SOURce:RADIo:SWEEp:LEVEl:EN?　　查询返回值1

:RADIo:SWEEp:STEP:EN

- **命令格式：**

  [:SOURce]:RADIo:SWEEp:STEP:EN {{1|ON} | {0|OFF}}

  [:SOURce]:RADIo:SWEEp:STEP:EN?

- **功能描述：**

  设置步进扫描使能。

- **返回格式：**

  查询返回步进扫描开关状态，返回值0或1。

- **举例：**

  :SOURce:RADIo:SWEEp:STEP:EN ON　　打开步进扫描使能

  :SOURce:RADIo:SWEEp:STEP:EN?　　查询返回值1

:RADIo:SWEEp:LIST:EN

- **命令格式：**

  [:SOURce]:RADIo:SWEEp:LIST:EN {{1|ON} | {0|OFF}}

  [:SOURce]:RADIo:SWEEp:LIST:EN?

- **功能描述：**

  设置列表扫描使能。

- **返回格式：**

  查询返回列表扫描使能状态，返回值为1或0。

- **举例：**

  :SOURce:RADIo:SWEEp:LIST:EN ON　　打开列表扫描使能

  :SOURce:RADIo:SWEEp:LIST:EN?　　查询返回值1

:LIST:DIRection

- **命令格式：**

  [:SOURce]:LIST:DIRection {{1|DOWN} | {0|UP}}

  [:SOURce]:LIST:DIRection?

- **功能描述：**

  设置扫描方向，向上或向下。

- **返回格式：**

  返回当前扫描方向，返回值0或1。

- **举例：**

  :SOURce:LIST:DIRection DOWN　　设置向下扫描

  :SOURce:LIST:DIRection?　　查询返回值1

:RADIo:SWEEp:MODE

- **命令格式：**

  [:SOURce]:RADIo:SWEEp:MODE {{1|SINGle} | {0|COUNt}}

  [:SOURce]:RADIo:SWEEp:MODE?

- **功能描述：**

设置扫描模式，连续或单次。

- **返回格式：**

  查询返回扫描模式状态，返回值为1或0。

- **举例：**

  :SOURce:RADIo:SWEEp:MODE SINGle　　设置扫描模式单次

  :SOURce:RADIo:SWEEp:MODE?　　查询返回值1

:RADIo:SWEEp:TRIGedge

- **命令格式：**

  [:SOURce]:RADIo:SWEEp:TRIGedge {{0| RISEedge} | {1 | FALLedge}}

  [:SOURce]:RADIo:SWEEp:TRIGedge?

- **功能描述：**

  设置扫描触发输出边沿，上升沿，下降沿。

- **返回格式：**

  查询返回值为0,1,。

- **举例：**

  :SOURce:RADIo:SWEEp:TRIGedge RISEedge　　设置触发边沿为上升沿

  :SOURce:RADIo:SWEEp:TRIGedge ?　　查询返回值0

:RADIo:POINt:TRIGger

- **命令格式：**

  [:SOURce]:RADIo:POINt:TRIGger {AUTO | NKEY | BUS | EXT}

  [:SOURce]:RADIo:POINt:TRIGger?

- **功能描述：**

  设置点触发方式，自动，按键，总线，外部。

- **返回格式：**

  查询返回值为0,1,2,3。

- **举例：**

  :SOURce:RADIo:POINt:TRIGger NKEY　　设置点触发方式为按键

  :SOURce:RADIo:POINt:TRIGger?　　查询返回值1

:RADIo:TRIGger:MODE

- **命令格式：**

  [:SOURce]:RADIo:TRIGger:MODE {AUTO | NKEY | BUS | EXT}

  [:SOURce]:RADIo:TRIGger:MODE?

- **功能描述：**

  设置触发方式，自动，按键，总线，外部。

- **返回格式：**

  查询返回值为0,1,2,3。

- **举例：**

  :SOURce:RADIo:TRIGger:MODE NKEY　　设置触发方式为按键

  :SOURce:RADIo:TRIGger:MODE?　　查询返回值1

:FREQuency:STARt

- **命令格式：**

  [:SOURce]:FREQuency:STARt <value><unit>

  [:SOURce]:FREQuency:STARt?

- **功能描述：**

  设置扫描起始频率。

- **返回格式：**

  查询返回起始频率值，单位Hz。

- **举例：**

  :SOURce:FREQuency:STARt 1kHz　　设置扫描起始频率1kHz

  :SOURce:FREQuency:STARt?　　查询返回值1.000000e+03

:FREQuency:STOP

- **命令格式：**

  [:SOURce]:FREQuency:STOP <value><unit>

  [:SOURce]:FREQuency:STOP?

- **功能描述：**

  设置扫描停止频率。

- **返回格式：**

  查询返回停止频率值，单位Hz。

- **举例：**

  :SOURce:FREQuency:STOP 1kHz　　设置扫描停止频率

  :SOURce:FREQuency:STOP?　　查询返回值1.000000e+03

:POWer:STARt

- **命令格式：**

  [:SOURce]:POWer:STARt <value><unit>

[:SOURce]:POWer:STARt?

- **功能描述：**

  设置扫描起始幅度,单位包括dBm,dBuV,uV,mV,V,nW,uW,mW。

- **返回格式：**

  查询返回扫描起始幅度，单位dBm。

- **举例：**

  :SOURce:POWer:STARt 1V　　设置扫描起始幅度1V

  :SOURce:POWer:STARt?　　查询返回值13

:POWer:STOP

- **命令格式：**

  [:SOURce]:POWer:STOP <value><unit>

  [:SOURce]:POWer:STOP?

- **功能描述：**

  设置扫描停止幅度，单位包括dBm,dBuV,uV,mV,V,nW,uW,mW。

- **返回格式：**

  查询返回扫描停止幅度，单位dBm。

- **举例：**

  :SOURce:POWer:STOP 1V　　设置扫描停止幅度1V

  :SOURce:POWer:STOP?　　查询返回值13

:SWEep:DWELl

- **命令格式：**

  [:SOURce]:SWEep:DWELl <value><unit>

  [:SOURce]:SWEep:DWELl?

- **功能描述：**

  步进扫描驻留时间，单位包括s,ms,us,ns。

- **返回格式：**

  返回当前设置驻留时间，返回单位s。

- **举例：**

  :SOURce:SWEep:DWELl 50ms　　设置驻留时间50ms

  :SOURce:SWEep:DWELl?　　查询返回值5.000000e-02

:SWEep:POINts

- **命令格式：**

[:SOURce]:SWEep:POINts <value>

[:SOURce]:SWEep:POINts?

- **功能描述：**

  设置扫描点数，无单位。

- **返回格式：**

  返回设置点数。

- **举例：**

  :SOURce:SWEep:POINts 5　　设置扫描点数5个

  :SOURce:SWEep:POINts?　　查询返回值5

:RADIo:SWEEp:SHAPe

- **命令格式：**

  [:SOURce]:RADIo:SWEEp:SHAPe {SAWTooth | TRIAngel}

  [:SOURce]:RADIo:SWEEp:SHAPe?

- **功能描述：**

  设置扫描形状，锯齿，三角。

- **返回格式：**

  返回值为0或1。

- **举例：**

  :SOURce:RADIo:SWEEp:SHAPe SAWTooth　　设置扫描形状为锯齿

  :SOURce:RADIo:SWEEp:SHAPe?　　查询返回值0

:SWEep:SPACing

- **命令格式：**

  [:SOURce]:SWEep:SPACing {LINEar | LOG}

  [:SOURce]:SWEep:SPACing?

- **功能描述：**

  设置扫描方式，对数，线性。

- **返回格式：**

  返回值为0或1。

- **举例：**

  :SOURce:SWEep:SPACing LOG　　设置对数扫描

  :SOURce:SWEep:SPACing?　　查询返回值1

:LIST:FREQuency

- **命令格式：**

  [:SOURce]:LIST:FREQuency　　<value><unit>

  [:SOURce]:LIST:FREQuency?

- **功能描述：**

  设置列表扫描频率。

- **返回格式：**

  返回频率值，单位Hz。

- **举例：**

  :SOURce:LIST:FREQuency 1kHz　　设置列表扫描频率为1kHz

  :SOURce:LIST:FREQuency?　　查询返回值1.000000e+03

:LIST:POWer

- **命令格式：**

  [:SOURce]:LIST:POWer <value><unit>

  [:SOURce]:LIST:POWer?

- **功能描述：**

  设置列表扫描幅度，单位dBm。

- **返回格式：**

  返回扫描幅度，单位为dBm。

- **举例：**

  :SOURce:LIST:POWer 10dBm　　设置列表扫描幅度10dBm

  :SOURce:LIST:POWer?　　查询返回值10

:LIST:DWELl

- **命令格式：**

  [:SOURce]:LIST:DWELl <value><unit>

  [:SOURce]:LIST:DWELl?

- **功能描述：**

  设置列表扫描当前行驻留时间。

- **返回格式：**

  返回当前行的驻留时间，单位秒(S)。

- **举例：**

  :SOURce:LIST:DWELl 500ms　　设置驻留时间500ms

  :SOURce:LIST:DWELl?　　查询返回值0.5

:RADIo:SWEEp:CURRow

- **命令格式：**

  [:SOURce]:RADIo:SWEEp:CURRow <value>

- **功能描述：**

  设置当前列表扫描编辑第几行。

- **返回格式：**

  无返回。

- **举例：**

  :SOURce:RADIo:SWEEp:CURRow 2　　设置当前编辑第二行

:RADIo:SWEEp:ADDRow

- **命令格式：**

  [:SOURce]:RADIo:SWEEp:ADDRow <value>

- **功能描述：**

  设置列表扫描编辑表格增加几行。

- **返回格式：**

  无返回值。

- **举例：**

  :SOURce:RADIo:SWEEp:ADDRow 2　　表格增加两行

:RADIo:SWEEp:SUBRow

- **命令格式：**

  [:SOURce]:RADIo:SWEEp:SUBRow <value>

- **功能描述：**

  从列表扫描表格最后一行开始，减少设置的行数。

- **返回格式：**

  无返回值。

- **举例：**

  :SOURce:RADIo:SWEEp:SUBRow 2　　表格从最后一行减少两行

:RADIo:SENSor:DEVIce

- **命令格式：**

  [:SOURce]:RADIo:SENSor:DEVIce <value>

  [:SOURce]:RADIo:SENSor:DEVIce?

- **功能描述：**

选择功率计已经连接的设备，序号从0开始。

- **返回格式：**

  返回当前连接的设备的序号，无单位。

- **举例：**

  :SOURce:RADIo:SENSor:DEVIce 1　　选择已经连接的第二个设备

  :SOURce:RADIo:SENSor:DEVIce?　　查询返回值1

:RADIo:SENSor:FREQ

- **命令格式：**

  [:SOURce]:RADIo:SENSor:FREQ <value><unit>

  [:SOURce]:RADIo:SENSor:FREQ?

- **功能描述：**

  设置功率计测量频率。

- **返回格式：**

  返回测量频率值，单位Hz。

- **举例：**

  :SOURce:RADIo:SENSor:FREQ 10kHz　　设置10kHz测量频率

  :SOURce:RADIo:SENSor:FREQ?　　查询返回值10000

:RADIo:SWEEp:BUS:TRIGger

- **命令格式：**

  [:SOURce]:RADIo:SWEEp:BUS:TRIGger

- **功能描述：**

  设置射频扫描总线触发，无参数。

- **返回格式：**

  无返回。

- **举例：**

  :SOURce:RADIo:SWEEp:BUS:TRIGger　　射频扫描总线触发一次

:RADIo:SENSor:AMPTofst

- **命令格式：**

  [:SOURce]:RADIo:SENSor:AMPTofst <value><unit>

  [:SOURce]:RADIo:SENSor:AMPTofst?

- **功能描述：**

  设置功率计幅度偏移，单位dB。

- **返回格式：**

  返回当前幅度偏移值和单位。

- **举例：**

  :SOURce:RADIo:SENSor:AMPTofst 10dB　　设置幅度偏移10dB

  :SOURce:RADIo:SENSor:AMPTofst?　　查询返回10dB

:RADIo:SENSor:FREQmul

- **命令格式：**

  [:SOURce]:RADIo:SENSor:FREQmul <value>

  [:SOURce]:RADIo:SENSor:FREQmul?

- **功能描述：**

  设置频率倍乘系数，无单位。

- **返回格式：**

  返回频率倍乘系数值。

- **举例：**

  :SOURce:RADIo:SENSor:FREQmul 2　　设置频率倍乘系数2倍

  :SOURce:RADIo:SENSor:FREQmul?　　查询返回值2

:RADIo:SENSor:FREQofst

- **命令格式：**

  [:SOURce]:RADIo:SENSor:FREQofst <value><unit>

  [:SOURce]:RADIo:SENSor:FREQofst?

- **功能描述：**

  设置功率计频率偏移值。

- **返回格式：**

  返回频率偏移值，单位Hz。

- **举例：**

  :SOURce:RADIo:SENSor:FREQofst 100Hz　　设置频率偏移100Hz

  :SOURce:RADIo:SENSor:FREQofst?　　查询返回值100

:RADIo:SENSor:AVECount

- **命令格式：**

  [:SOURce]:RADIo:SENSor:AVECount <value>

  [:SOURce]:RADIo:SENSor:AVECount?

- **功能描述：**

设置功率计平均次数。

- **返回格式：**

  返回平均次数，无单位。

- **举例：**

  :SOURce:RADIo:SENSor:AVECount 10　　设置功率计平均10次

  :SOURce:RADIo:SENSor:AVECount?　　查询返回值10

:RADIo:SENSor:FREQcouple

- **命令格式：**

  [:SOURce]:RADIo:SENSor:FREQcouple {{1|ON} | {0|OFF}}

  [:SOURce]:RADIo:SENSor:FREQcouple?

- **功能描述：**

  设置功率计频率耦合使能。

- **返回格式：**

  返回值为1或0。

- **举例：**

  :SOURce:RADIo:SENSor:FREQcouple ON　　打开功率计频率耦合

  :SOURce:RADIo:SENSor:FREQcouple?　　查询返回值1

:RADIo:SENSor:LEVEl

- **命令格式：**

  [:SOURce]:RADIo:SENSor:LEVEl?

- **功能描述：**

  查询功率计幅度测量值。

- **返回格式：**

  返回值为幅度测量值和单位。

- **举例：**

  :SOURce:RADIo:SENSor:LEVEl?　　查询返回值0.00dBm

函数生成

:LFOutput:STATe

- **命令格式：**

  [:SOURce]:LFOutput:STATe {{1|ON} | {0|OFF}}

  [:SOURce]:LFOutput:STATe?

- **功能描述：**

设置低频输出使能状态。

- **返回格式：**

  返回低频输出使能状态，返回值为0或1。

- **举例：**

  :SOURce:LFOutput:STATe ON　　打开低频输出使能

  :SOURce:LFOutput:STATe?　　查询返回值1

:LOWF:NOISesum

- **命令格式：**

  [:SOURce]:LOWF:NOISesum {{0|OFF} | {1|ON}}

  [:SOURce]:LOWF:NOISesum?

- **功能描述：**

  设置低频噪声叠加使能。

- **返回格式：**

  返回值为1或0。

- **举例：**

  :SOURce:LOWF:NOISesum ON　　打开低频噪声叠加使能

  :SOURce:LOWF:NOISesum?　　查询返回值1

:LOWF:TYPE

- **命令格式：**

  [:SOURce]:LOWF:TYPE {{0 | SINE} | {1 | SQUAre} | {2 | PULSe} | {3 | RAMP} | {4 | ARB} | {5 | DC} | {6 | NOISe}}

  [:SOURce]:LOWF:TYPE?

- **功能描述：**

  设置基础波类型。

- **返回格式：**

  返回当波形名称。

- **举例：**

  :SOURce:LOWF:TYPE SINE　　设置基础波形为正弦波

  :SOURce:LOWF:TYPE?　　查询返回SINE

:LFOutput:LOAD:IMPedance

- **命令格式：**

  [:SOURce]:LFOutput:LOAD:IMPedance {50 | 600 | 1000000}

[:SOURce]:LFOutput:LOAD:IMPedance?

- **功能描述：**

  设置低频负载50欧，600欧，高阻。

- **返回格式：**

  返回值为50，600，1000000。

- **举例：**

  :SOURce:LFOutput:LOAD:IMPedance 50　　设置低频负载50欧

  :SOURce:LFOutput:LOAD:IMPedance?　　查询返回值50

:LOWF:FREQ

- **命令格式：**

  [:SOURce]:LOWF:FREQ <value><unit>

  [:SOURce]:LOWF:FREQ?

- **功能描述：**

  设置低频输出频率。

- **返回格式：**

  返回低频输出频率值，单位Hz。

- **举例：**

  :SOURce:LOWF:FREQ 10kHz　　设置输出10kHz

  :SOURce:LOWF:FREQ?　　查询返回值1.000000e+04

:LFOutput:AMPLitude

- **命令格式：**

  [:SOURce]:LFOutput:AMPLitude <value><unit>

  [:SOURce]:LFOutput:AMPLitude?

- **功能描述：**

  设置低频幅度值，单位包括Vpp,mVpp,Vrms,mVrms。

- **返回格式：**

  返回当前设置的幅度值和单位，单位为Vpp,mVpp,Vrms,mVrms。

- **举例：**

  :SOURce:LFOutput:AMPLitude 1Vpp　　设置低频输出1Vpp

  :SOURce:LFOutput:AMPLitude?　　查询返回值1Vpp

:LOWF:PHASe

- **命令格式：**

[:SOURce]:LOWF:PHASe <value><unit>

[:SOURce]:LOWF:PHASe?

- **功能描述：**

  设置低频输出相位值，单位度(deg),弧度(rad)。

- **返回格式：**

  返回相位值和单位为度。

- **举例：**

:SOURce:LOWF:PHASe 30deg	设置低频相位30度
:SOURce:LOWF:PHASe?	查询返回值30deg

:LFOutput:OFFset

- **命令格式：**

  [:SOURce]:LFOutput:OFFset <value><unit>

  [:SOURce]:LFOutput:OFFset?

- **功能描述：**

  设置直流偏移幅度，单位为V,mV。

- **返回格式：**

  查询返回直流偏移幅度值和单位为。

- **举例：**

:SOURce:LFOutput:OFFset 100mV	设置直流偏移100mV
:SOURce:LFOutput:OFFset?	查询返回值100mV

:LOWF:DUTY

- **命令格式：**

  [:SOURce]:LOWF:DUTY <value>

  [:SOURce]:LOWF:DUTY?

- **功能描述：**

  设置方波或脉冲波占空比。

- **返回格式：**

  查询返回占空比值，单位为%。

- **举例：**

[:SOURce]:LOWF:DUTY 20	设置方波或脉冲波占空比为20%
[:SOURce]:LOWF:DUTY?	查询返回值20

:LOWF:SYMMetry

- **命令格式：**

  [:SOURce]:LOWF:SYMMetry <value>

  [:SOURce]:LOWF:SYMMetry?

- **功能描述：**

  设置三角波对称性。

- **返回格式：**

  查询返回三角波对称性值，单位为%。

- **举例：**

  [:SOURce]:LOWF:SYMMetry 20　　设置三角波对称性为20%

  [:SOURce]:LOWF:SYMMetry?　　查询返回值20

:LOWF:RISEedge

- **命令格式：**

  [:SOURce]:LOWF:RISEedge<value><unit>

  [:SOURce]:LOWF:RISEedge?

- **功能描述：**

  设置脉冲波边沿上升时间，单位为s,ms,us,ns。

- **返回格式：**

  查询返回边沿上升时间，单位为s。

- **举例：**

  :SOURce:LOWF:RISEedge20ns　　设置边沿时间为20ns

  :SOURce:LOWF:RISEedge?　　查询返回值2.000000e-08

:LOWF:FALLedge

- **命令格式：**

  [:SOURce]:LOWF:FALLedge <value><unit>

  [:SOURce]:LOWF:FALLedge?

- **功能描述：**

  设置脉冲波边沿下降时间，单位为s,ms,us,ns。

- **返回格式：**

  查询返回边沿下降时间，单位为s。

- **举例：**

  :SOURce:LOWF:FALLedge 20ns　　设置边沿时间为20ns

  :SOURce:LOWF:FALLedge?　　查询返回值2.000000e-08

:LOWF:NOISebw

- **命令格式：**

  [:SOURce]:LOWF:NOISebw <value><unit>

  [:SOURce]:LOWF:NOISebw?

- **功能描述：**

  设置噪声带宽，单位包括Hz,kHz,MHz,GHz。

- **返回格式：**

  查询返回噪声带宽值，返回单位为Hz。

- **举例：**

  :SOURce:LOWF:NOISebw 10kHz　　设置噪声带宽为10kHz

  :SOURce:LOWF:NOISebw?　　查询返回值为1.000000e+04

:LOWF:NOISeamp

- **命令格式：**

  [:SOURce]:LOWF:NOISeamp <value><unit>

  [:SOURce]:LOWF:NOISeamp?

- **功能描述：**

  设置噪声幅度值，单位为mV，V。

- **返回格式：**

  查询返回幅度值和单位。

- **举例：**

  :SOURce:LOWF:NOISeamp 10mV　　设置噪声幅度为10mV

  :SOURce:LOWF:NOISeamp?　　查询返回值10mV

:LOWF:MOD:EN

- **命令格式：**

  [:SOURce]:LOWF:MOD:EN {{1|ON} | {0|OFF}}

  [:SOURce]:LOWF:MOD:EN?

- **功能描述：**

  设置低频调制使能。

- **返回格式：**

  返回值为1或0。

- **举例：**

  :SOURce:LOWF:MOD:EN ON　　打开低频调制使能

  :SOURce:LOWF:MOD:EN?　　查询返回值1

:LOWF:MOD:TYPE

- **命令格式：**

  [:SOURce]:LOWF:MOD:TYPE {{0 | AM} | {1 | FM} | {2 | PHM} | {3 | PM} | {4 | ASK} | {5 | FSK} | {6 | PSK} | {7 | QAM}}

  [:SOURce]:LOWF:MOD:TYPE?

- **功能描述：**

  设置低频调制模式。

- **返回格式：**

  查询返回值为0,1,2,3,4,5,6,7。

- **举例：**

  :SOURce:LOWF:MOD:TYPE AM　　设置调制模式为调幅

  :SOURce:LOWF:MOD:TYPE?　　查询返回值0

:LOWF:MOD:WAVE

- **命令格式：**

  [:SOURce]:LOWF:MOD:WAVE {{0 | SINE} | {1 | SQUAre} | {2 | RAMP} | {3 | ARB}}

  [:SOURce]:LOWF:MOD:WAVE?

- **功能描述：**

  设置调制波形，正弦波，方波，三角波，任意波。

- **返回格式：**

  查询返回值为0,1,2,3。

- **举例：**

  :SOURce:LOWF:MOD:WAVE SINE　　设置调制波行为正弦波

  :SOURce:LOWF:MOD:WAVE?　　查询返回值为0

:LOWF:MOD:FREQ

- **命令格式：**

  [:SOURce]:LOWF:MOD:FREQ <value><unit>

  [:SOURce]:LOWF:MOD:FREQ?

- **功能描述：**

  设置调制频率。

- **返回格式：**

  查询返回单位是Hz。

- **举例：**

  :SOURce:LOWF:MOD:FREQ 10kHz　　设置调制频率为10kHz

:SOURce:LOWF:MOD:FREQ? 查询返回值1.000000e+04

:LOWF:MOD:DEPTh

- **命令格式：**

  [:SOURce]:LOWF:MOD:DEPTh <value>

  [:SOURce]:LOWF:MOD:DEPTh?

- **功能描述：**

  设置调制深度，范围0到120%。

- **返回格式：**

  查询返回调制深度，无单位。

- **举例：**

  :SOURce:LOWF:MOD:DEPTh 50 设置调试深度50%

  :SOURce:LOWF:MOD:DEPTh? 查询返回值50

:LOWF:MOD:FREQdev

- **命令格式：**

  [:SOURce]:LOWF:MOD:FREQdev <value><unit>

  [:SOURce]:LOWF:MOD:FREQdev?

- **功能描述：**

  设置调频的频率偏移。

- **返回格式：**

  查询返回频率偏移值，返回单位Hz。

- **举例：**

  :SOURce:LOWF:MOD:FREQdev 10kHz 设置频率偏移10kHz

  :SOURce:LOWF:MOD:FREQdev? 查询返回值1.000000e+04

:LOWF:MOD:PHASe1

- **命令格式：**

  [:SOURce]:LOWF:MOD:PHASe1 <value><unit>

  [:SOURce]:LOWF:MOD:PHASe1?

- **功能描述：**

  设置调相相位值或相移键控相位1的值，单位为deg,rad。

- **返回格式：**

  查询返回相位值和单位。

- **举例：**

:SOURce:LOWF:MOD:PHASe1 30deg　　设置相位30度

:SOURce:LOWF:MOD:PHASe1?　　查询返回值30deg

:LOWF:MOD:PHASe2

- **命令格式：**

  [:SOURce]:LOWF:MOD:PHASe2 <value><unit>

  [:SOURce]:LOWF:MOD:PHASe2?

- **功能描述：**

  设置相移键控相位2的值，单位为deg,rad。

- **返回格式：**

  查询返回相位值和单位。

- **举例：**

  :SOURce:LOWF:MOD:PHASe2 30deg　　设置相位2的值为30度

  :SOURce:LOWF:MOD:PHASe2?　　查询返回值30deg

:LOWF:MOD:PHASe3

- **命令格式：**

  [:SOURce]:LOWF:MOD:PHASe3 <value><unit>

  [:SOURce]:LOWF:MOD:PHASe3?

- **功能描述：**

  设置相移键控相位3的值，单位为deg,rad。

- **返回格式：**

  查询返回相位值和单位。

- **举例：**

  :SOURce:LOWF:MOD:PHASe3 30deg　　设置相位30度

  :SOURce:LOWF:MOD:PHASe3?　　查询返回值3030deg

:LOWF:MOD:PHASe4

- **命令格式：**

  [:SOURce]:LOWF:MOD:PHASe4 <value><unit>

  [:SOURce]:LOWF:MOD:PHASe4?

- **功能描述：**

  设置相移键控相位4的值，单位为deg,rad。

- **返回格式：**

  查询返回相位值和单位。

- **举例：**

  :SOURce:LOWF:MOD:PHASe4 30deg　　设置相位30度

  :SOURce:LOWF:MOD:PHASe4?　　查询返回值30deg

:LOWF:MOD:PMFReq

- **命令格式：**

  [:SOURce]:LOWF:MOD:PMFReq <value><unit>

  [:SOURce]:LOWF:MOD:PMFReq?

- **功能描述：**

  设置脉冲调制脉冲频率。

- **返回格式：**

  查询返回脉冲频率值，返回单位Hz。

- **举例：**

  :SOURce:LOWF:MOD:PMFReq 10kHz　　设置脉冲频率10kHz

  :SOURce:LOWF:MOD:PMFReq?　　查询返回值1.000000e+04

:LOWF:MOD:PMDUty

- **命令格式：**

  [:SOURce]:LOWF:MOD:PMDUty <value>

  [:SOURce]:LOWF:MOD:PMDUty?

- **功能描述：**

  设置脉冲调制脉冲的占空比。

- **返回格式：**

  查询返回单位%。

- **举例：**

  :SOURce:LOWF:MOD:PMDUty 50　　设置脉冲的占空比50%

  :SOURce:LOWF:MOD:PMDUty?　　查询返回值50

::LOWF:MOD:RATE

- **命令格式：**

  [:SOURce]:LOWF:MOD:RATE <value><unit>

  [:SOURce]:LOWF:MOD:RATE?

- **功能描述：**

  设置数字调制的调制速率。

- **返回格式：**

查询返回调制速率值，返回单位Hz。

- **举例：**

:SOURce:LOWF:MOD:RATE 10kHz　　设置调制速率为10kHz

:SOURce:LOWF:MOD:RATE?　　查询返回值1.000000e+04

:LOWF:HOP:FREQ1

- **命令格式：**

[:SOURce]:LOWF:HOP:FREQ1 <value><unit>

[:SOURce]:LOWF:HOP:FREQ1?

- **功能描述：**

设置频移键控跳跃频率1的值。

- **返回格式：**

查询返回单位为Hz。

- **举例：**

:SOURce:LOWF:HOP:FREQ1 10kHz　　设置跳跃频率1的10kHz

:SOURce:LOWF:HOP:FREQ1?　　查询返回值1.000000e+04

:LOWF:HOP:FREQ2

- **命令格式：**

[:SOURce]:LOWF:HOP:FREQ2 <value><unit>

[:SOURce]:LOWF:HOP:FREQ2?

- **功能描述：**

设置频移键控跳跃频率2的值。

- **返回格式：**

查询返回单位为Hz。

- **举例：**

:SOURce:LOWF:HOP:FREQ2 10kHz　　设置跳跃频率2的10kHz

:SOURce:LOWF:HOP:FREQ2?　　查询返回值1.000000e+04

:LOWF:HOP:FREQ3

- **命令格式：**

[:SOURce]:LOWF:HOP:FREQ3 <value><unit>

[:SOURce]:LOWF:HOP:FREQ3?

- **功能描述：**

设置频移键控跳跃频率3的值。

- **返回格式：**

  查询返回单位为Hz。

- **举例：**

  :SOURce:LOWF:HOP:FREQ3 10kHz　　设置跳跃频率3的10kHz

  :SOURce:LOWF:HOP:FREQ3?　　查询返回值1.000000e+04

:LOWF:MOD:FSKMode

- **命令格式：**

  [:SOURce]:LOWF:MOD:FSKMode {{0 | 2FSK} | {1 | 4FSK}}

  [:SOURce]:LOWF:MOD:FSKMode?

- **功能描述：**

  设置频移键控模式。

- **返回格式：**

  查询返回为2FSK或4FSK，无单位。

- **举例：**

  :SOURce:LOWF:MOD:FSKMode 2FSK　　设置2FSK频移键控

  :SOURce:LOWF:MOD:FSKMode?　　查询返回值2FSK

:LOWF:MOD:PSKMode

- **命令格式：**

  [:SOURce]:LOWF:MOD:PSKMode {{0 | 2PSK} | {1 | 4PSK}}

  [:SOURce]:LOWF:MOD:PSKMode?

- **功能描述：**

  设置相移键控模式。

- **返回格式：**

  查询返回为2PSK或4PSK，无单位。

- **举例：**

  :SOURce:LOWF:MOD:PSKMode 2PSK　　设置2PSK相移键控

  :SOURce:LOWF:MOD:PSKMode?　　查询返回值2PSK

:LOWF:MOD:SOURce

- **命令格式：**

  [:SOURce]:LOWF:MOD:SOURce {{0 | PN7} | {1 | PN9} | {2 | PN11} | {3 | PN15} | {4 | PN17} | {5 | PN21} | {6 | PN23} | {7 | PN25}}

  [:SOURce]:LOWF:MOD:SOURce?

- **功能描述：**

  设置码元模式。

- **返回格式：**

  查询返回码元模式值，无单位。

- **举例：**

  :SOURce:LOWF:MOD:SOURce PN7　　设置码元模式以为PN7

  :SOURce:LOWF:MOD:SOURce?　　查询返回值0

:LOWF:MOD:QAMMode

- **命令格式：**

  [:SOURce]:LOWF:MOD:QAMMode {{0 | QAM4} | {1 | QAM8} | {2 | QAM16} | {3 | QAM32} | {4 | QAM64} | {5 | QAM128} | {6 | QAM256}}

  [:SOURce]:LOWF:MOD:QAMMode?

- **功能描述：**

  设置QAM模式。

- **返回格式：**

  查询返回值为0,1,2,3,4,5,6，返回值无单位。

- **举例：**

  :SOURce:LOWF:MOD:QAMMode QAM4　　设置QAM模式为QAM4

  :SOURce:LOWF:MOD:QAMMode?　　查询返回值0

:LOWF:SWEEp:EN

- **命令格式：**

  [:SOURce]:LOWF:SWEEp:EN {{1|ON} | {0|OFF}}

  [:SOURce]:LOWF:SWEEp:EN?

- **功能描述：**

  设置扫描使能。

- **返回格式：**

  查询返回值为0或1。

- **举例：**

  :SOURce:LOWF:SWEEp:EN ON　　设置扫描使能打开

  :SOURce:LOWF:SWEEp:EN?　　查询返回值1

:LOWF:SWEEp:MODE

- **命令格式：**

[:SOURce]:LOWF:SWEEp:MODE {{0 | LINEar} | {1 | LOG} | {2 | STEP}}

[:SOURce]:LOWF:SWEEp:MODE?

- **功能描述：**

  设置扫描模式。

- **返回格式：**

  查询返回值为0,1,2。

- **举例：**

  :SOURce:LOWF:SWEEp:MODE LINEar　　设置线性扫描

  :SOURce:LOWF:SWEEp:MODE?　　查询返回值0

:LOWF:SWEEp:SHAP

- **命令格式：**

  [:SOURce]:LOWF:SWEEp:SHAP {{0 | POSSAW} | {1 | NEGSAW} | {2 | POSTRI} | {3 | NEGTRI}}

  [:SOURce]:LOWF:SWEEp:SHAP?

- **功能描述：**

  设置扫描形状，正锯齿，负锯齿，正三角，负三角。

- **返回格式：**

  查询返回为POSSAW,NEGSAW,POSTRI,NEGTRI 。

- **举例：**

  :SOURce:LOWF:SWEEp:SHAP POSSAW　　设置扫描形状为正锯齿

  :SOURce:LOWF:SWEEp:SHAP?　　查询返回POSSAW

:LOWF:SWEEp:TRIGout

- **命令格式：**

  [:SOURce]:LOWF:SWEEp:TRIGout {{0 | CLOSe} | {1 | RISEedge} | {2 | FALLedge}}

  [:SOURce]:LOWF:SWEEp:TRIGout?

- **功能描述：**

  设置扫描触发输出，关闭，上升沿，下降沿。

- **返回格式：**

  查询返回值为CLOSe,RISEedge,FALLedge。

- **举例：**

  :SOURce:LOWF:SWEEp:TRIGout CLOSe　　设置触发输出关闭

  :SOURce:LOWF:SWEEp:TRIGout?　　查询返回CLOSe

:LOWF:SWEEp:TRIGin

- **命令格式：**

  [:SOURce]:LOWF:SWEEp:TRIGin {{0 | AUTO} | {1 | NKEY} | {2 | BUS} | {3 | EXT}}

  [:SOURce]:LOWF:SWEEp:TRIGin?

- **功能描述：**

  设置扫描触发输入，自动，按键，总线，外部。

- **返回格式：**

  查询返回为AUTO,NKEY,BUS,EXT。

- **举例：**

  :SOURce:LOWF:SWEEp:TRIGin AUTO　　设置触发输入为自动

  :SOURce:LOWF:SWEEp:TRIGin?　　查询返回AUTO

:LOWF:SWEEp:TRIGedge

- **命令格式：**

  [:SOURce]:LOWF:SWEEp:TRIGedge {{0| RISEedge} | {1 | FALLedge}}

  [:SOURce]:LOWF:SWEEp:TRIGedge?

- **功能描述：**

  设置扫描外部触发输出边沿，上升沿，下降沿。

- **返回格式：**

  查询返回值为RISEedge，FALLedge,。

- **举例：**

  :SOURce:LOWF:SWEEp:TRIGedge RISEedge　　设置外部触发边沿为上升沿

  :SOURce:LOWF:SWEEp:TRIGedge ?　　查询返回RISEedge

:LOWF:FREQ:STAR

- **命令格式：**

  [:SOURce]:LOWF:FREQ:STARt <value><unit>

  [:SOURce]:LOWF:FREQ:STARt?

- **功能描述：**

  设置扫描起始频率。

- **返回格式：**

  查询返回起始频率值，返回单位为Hz。

- **举例：**

  :SOURce:LOWF:FREQ:STARt 10kHz　　设置扫描起始频率为10kHz

  :SOURce:LOWF:FREQ:STARt?　　查询返回值1.000000e+04

:LOWF:FREQ:STOP

- **命令格式：**

  [:SOURce]:LOWF:FREQ:STOP <value><unit>

  [:SOURce]:LOWF:FREQ:STOP?

- **功能描述：**

  设置扫描停止频率。

- **返回格式：**

  查询返回停止频率值，返回单位为Hz。

- **举例：**

  :SOURce:LOWF:FREQ:STOP 10kHz　　设置扫描停止频率为10kHz

  :SOURce:LOWF:FREQ:STOP?　　查询返回值1.000000e+04

:LOWF:SWEEp:TIME

- **命令格式：**

  [:SOURce]:LOWF:SWEEp:TIME <value><unit>

  [:SOURce]:LOWF:SWEEp:TIME?

- **功能描述：**

  设置扫描时间。

- **返回格式：**

  查询返回值单位为s。

- **举例：**

  :SOURce:LOWF:SWEEp:TIME 2s　　设置扫描时间为2s

  :SOURce:LOWF:SWEEp:TIME?　　查询返回值2.000000e+00

:LOWF:DWELl:TIME

- **命令格式：**

  [:SOURce]:LOWF:DWELl:TIME <value><unit>

  [:SOURce]:LOWF:DWELl:TIME?

- **功能描述：**

  设置扫描驻留时间。

- **返回格式：**

  查询返回值单位为s。

- **举例：**

  :SOURce:LOWF:DWELl:TIME 2s　　设置扫描驻留时间为2s

  :SOURce:LOWF:DWELl:TIME?　　查询返回值2.000000e+00

:LOWF:STEP:COUNt

- **命令格式：**

  [:SOURce]:LOWF:STEP:COUNt <value>

  [:SOURce]:LOWF:STEP:COUNt?

- **功能描述：**

  设置扫描点数，无单位。

- **返回格式：**

  查询返回扫描点数值。

- **举例：**

  :SOURce:LOWF:STEP:COUNt 5　　设置扫描点数为5

  :SOURce:LOWF:STEP:COUNt?　　查询返回值5

:LOWF:SWEEp:BUS:TRIGger

- **命令格式：**

  [:SOURce]:LOWF:SWEEp:BUS:TRIGger

- **功能描述：**

  设置低频扫描总线触发，无参数。

- **返回格式：**

  无返回。

- **举例：**

  :SOURce:LOWF:SWEEp:BUS:TRIGger　　低频扫描总线触发一次

WARB 命令

:WARB:CARRier

- **命令格式：**

  :WARB:CARRier <arb file>

- **功能描述：**

  用于写基波任意波形，先发送该指令，然后发送任意波形文件数据到信号源。

  <arb file>表示任意波形文件名称，只支持bsv文件格式。

- **返回格式：**

  无返回。

- **举例：**

  :WARB:CARRier test.bsv　　写低频基波任意波形文件

:WARB:MODulate

- **命令格式：**

  :WARB:MODulate <arb file>

- **功能描述：**

  用于写调制任意波形，先发送该指令，然后发送任意波形文件数据到信号源。

  <arb file>表示任意波形文件名称，只支持bsv文件格式。

- **返回格式：**

  无返回。

- **举例：**

  :WARB:MODulate test.bsv　　写低频调制任意波形文件

调制输入

:MODIn:STATe

- **命令格式：**

  [:SOURce]:MODIn:STATe {{1|ON} | {0|OFF}}

  [:SOURce]:MODIn:STATe?

- **功能描述：**

  调制输入使能。

- **返回格式：**

  查询返回状态，0或1。

- **举例：**

  :SOURce:MODIn:STATe ON　　打开调制输入

  :SOURce:MODIn:STATe?　　查询返回值1

:MODIn:LOAD

- **命令格式：**

  [:SOURce]:MODIn:LOAD {50 | 600 | 1000000}

  [:SOURce]:MODIn:LOAD?

- **功能描述：**

  设置负载等级，50欧，600欧，高阻。

- **返回格式：**

  查询返回值为50,600,1000000。

- **举例：**

  :SOURce:MODIn:LOAD 50　　设置负载为50欧

  :SOURce:MODIn:LOAD?　　查询返回值50

DISPlay 命令

:DISPlay:DATA?

- **命令格式：**

  :DISPlay:DATA?

- **功能描述：**

  获取屏幕图像。

- **返回格式：**

  查询返回屏幕图像数据。

- **举例：**

  :DISPlay:DATA?　　获取屏幕图像

编程说明

描述在编程操作过程中可能出现的一些问题及解决方法。当您遇到如下这些问题时，请按照相应的说明进行处理。

编程准备

用户可以通过使用射频信号源的 USB 或 LAN 端口，并结合 NI-VISA 和程序语言，远程控制射频信号源。编程工作仅适用于在 Windows 操作系统下使用 Visual Studio 和 LabVIEW 开发工具进行编程。

1. 建立通信

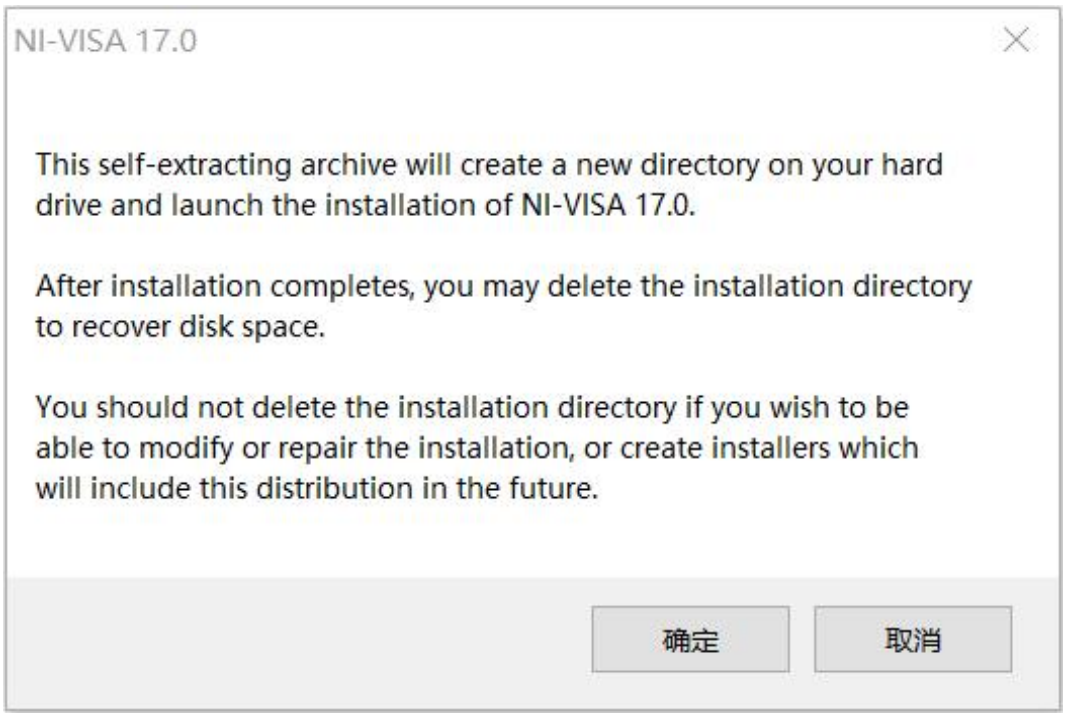
NI-VISA 是用于计算机与设备之间通信的通信库。 NI 软件有两种有效 VISA 安装包：完整版和运行引擎版（ Run-Time Engine ）。完整版包括NI设备驱动和 NI MAX工具，其中NI MAX是用于控制设备的用户界面。虽然驱动和 NI MAX 都很有用，但是它们不用于远程控制。运行引擎版（ Run-Time Engine ）是一个比完整版更小的文件，它主要用于远程控制。

你可以在NI官网上下载最新的NI-VISA运行引擎或完整版。它们的安装步骤基本相同。

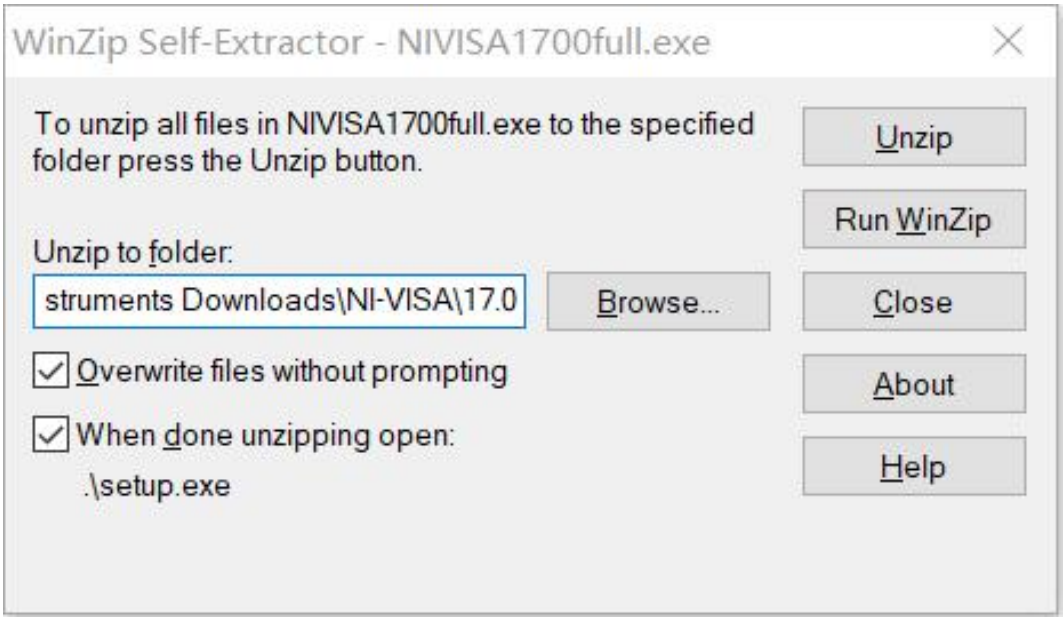
按照下列步骤安装 NI-VISA （示例使用 NI-VISA17.0 完整版）：

a. 下载合适版本的 NI-VISA

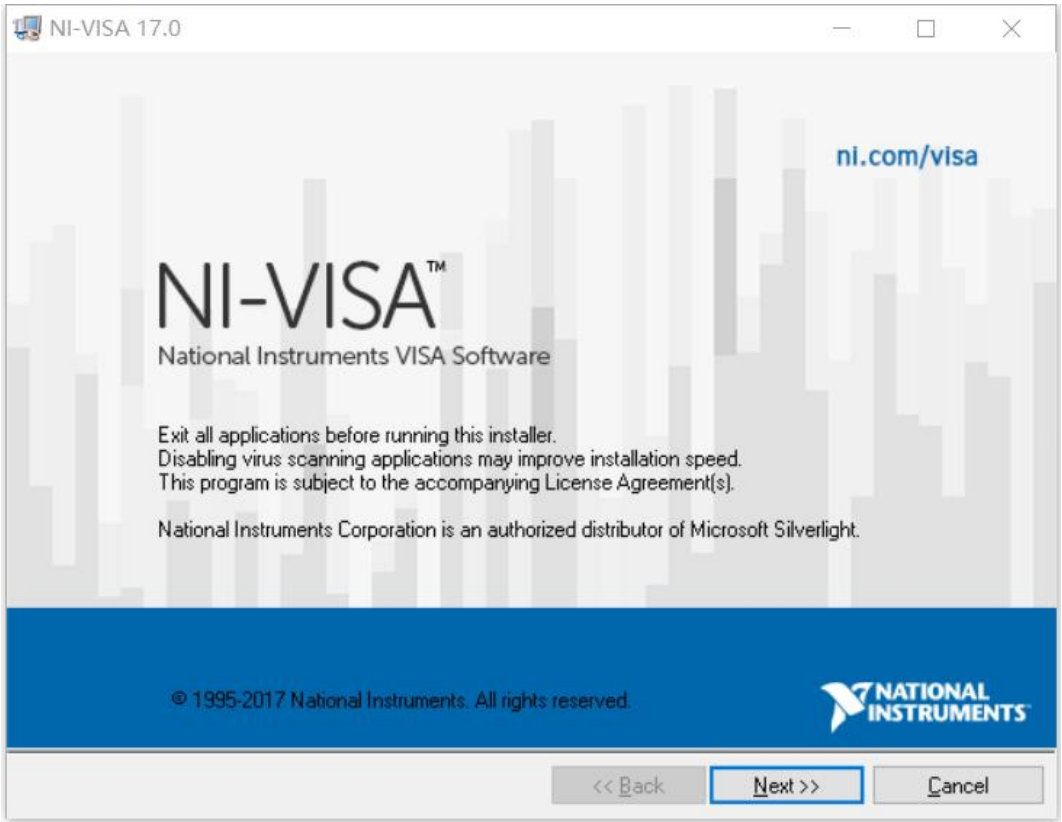
b. 双击NIVISA1700full.exe ，弹出对话框如下：



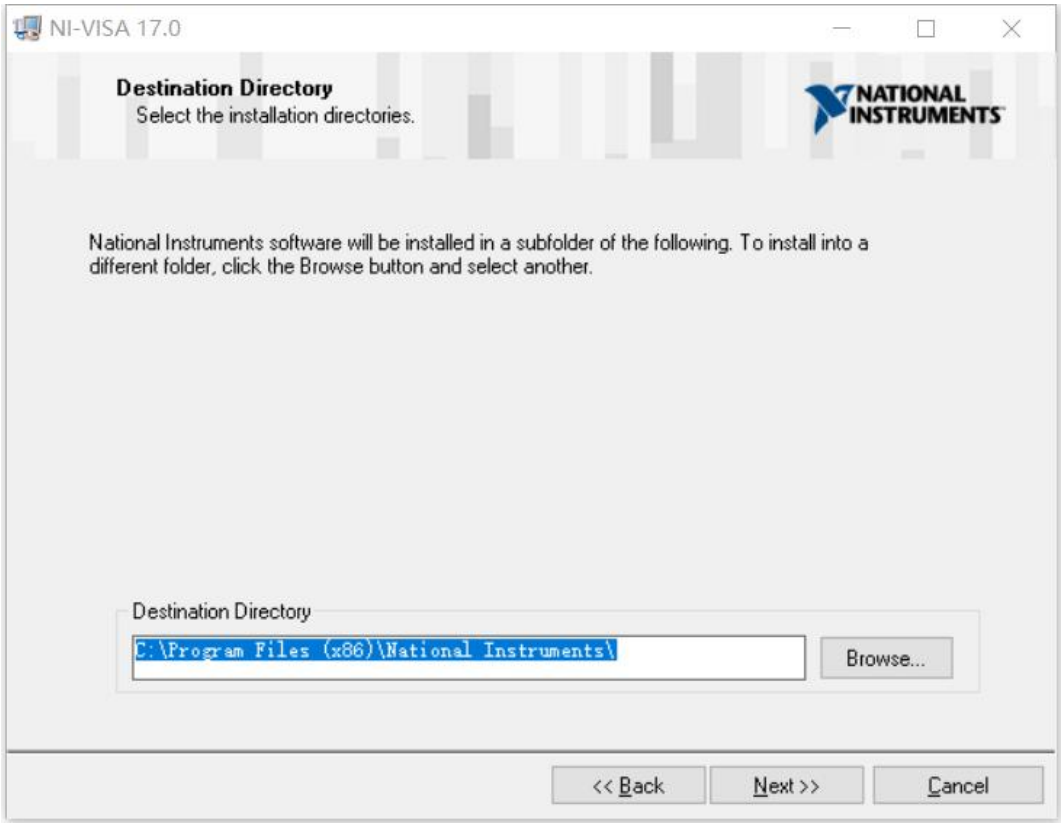
c.点击确认弹出对话框如下：



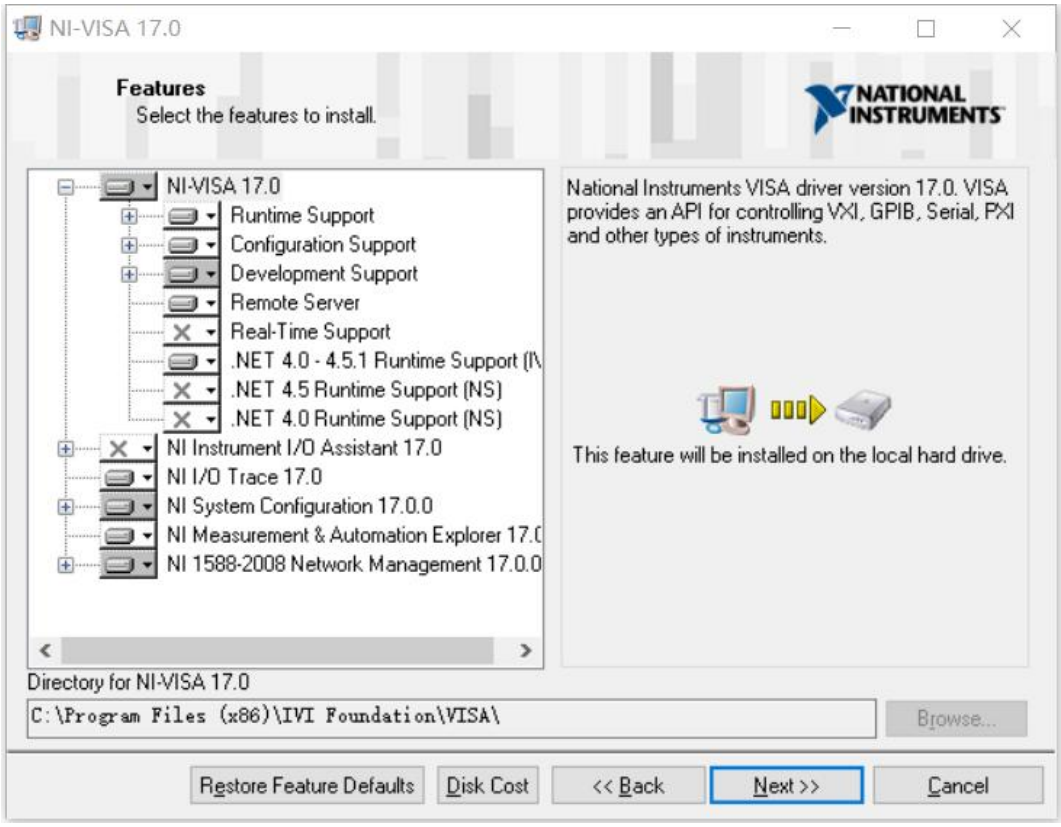
d. 点击 Unzip 解压文件，当解压完成后，安装程序将自动执行。若你的计算机需要安装 .NET Framework4 ，则在安装过程会自动安装 .NET Framework4 。



e. NI-VISA 安装对话框如上图所示，点击 Next 开始安装过程。

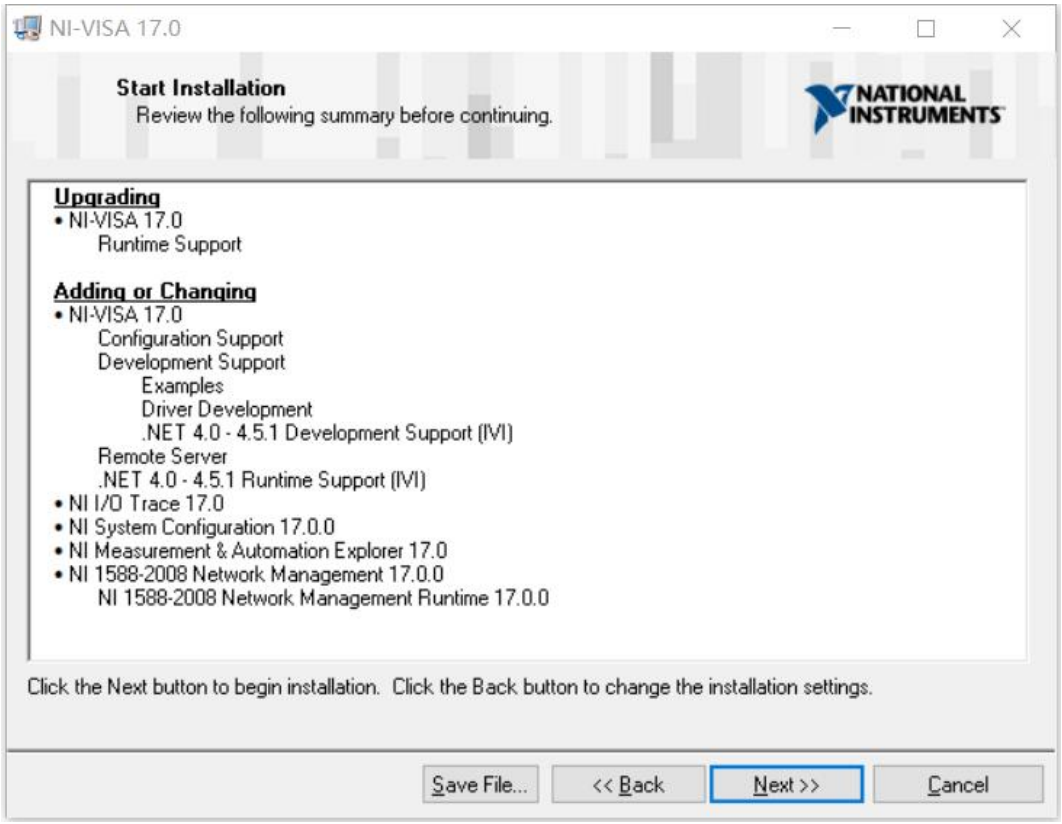


f. 设置安装路径，默认路径为“C:\Program Files (x86)\National Instruments\”。你也可以修改安装路径。点击 Next ，对话框如下图所示：

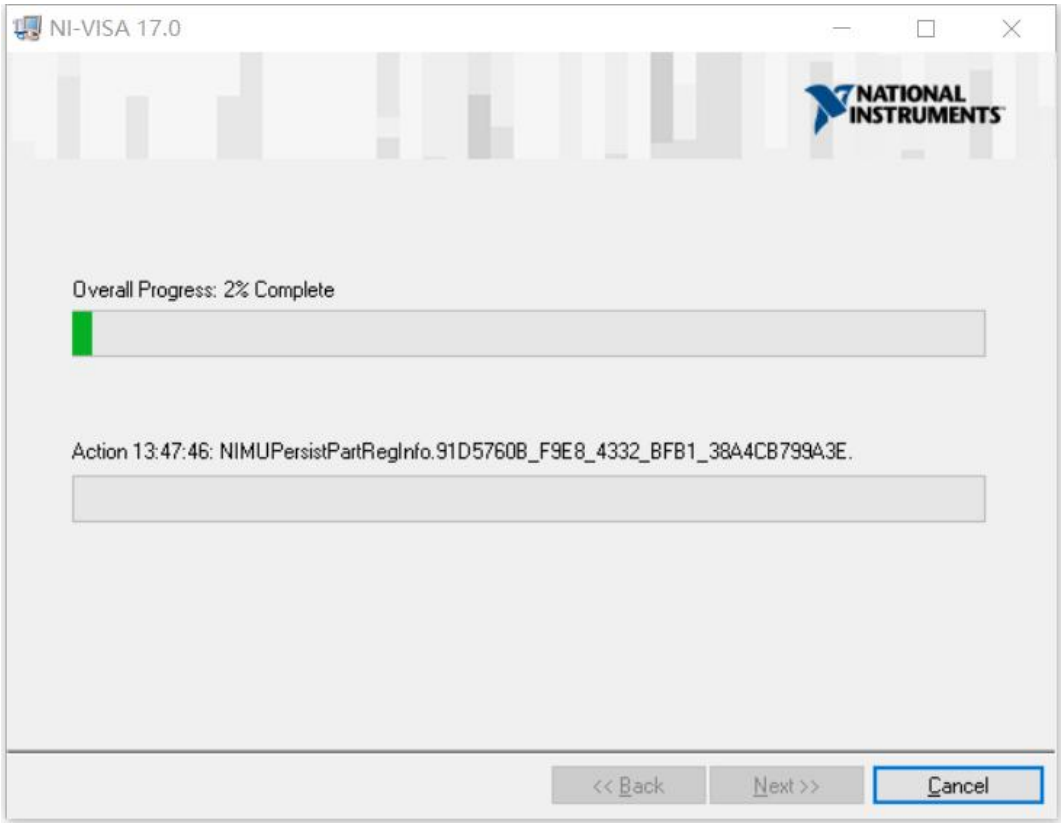


g. 点击Next两次，在许可协议对话框下，选择“I accept the above 2 License Agreement(s).”并点击Next ，

对话框如下图显示：



h. 点击 Next 开始安装：



j. 安装完成后，重启电脑

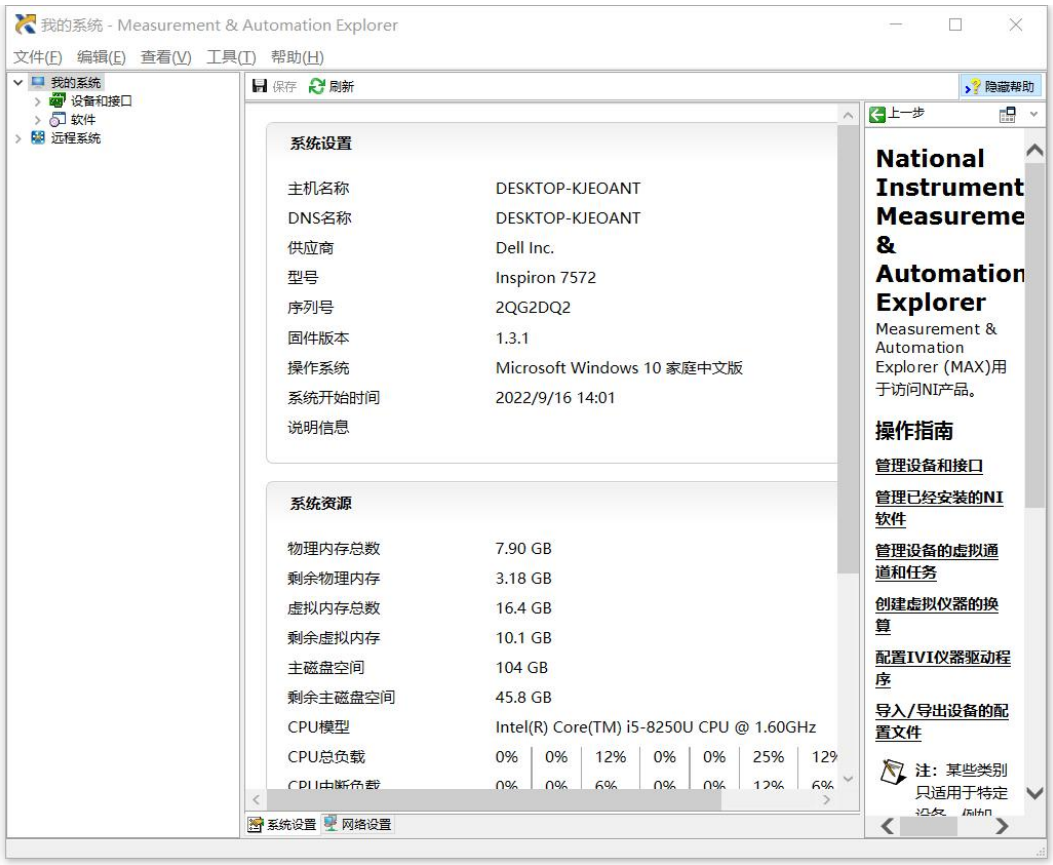
2. 连接仪器

以下通过USB方式连接进行介绍。

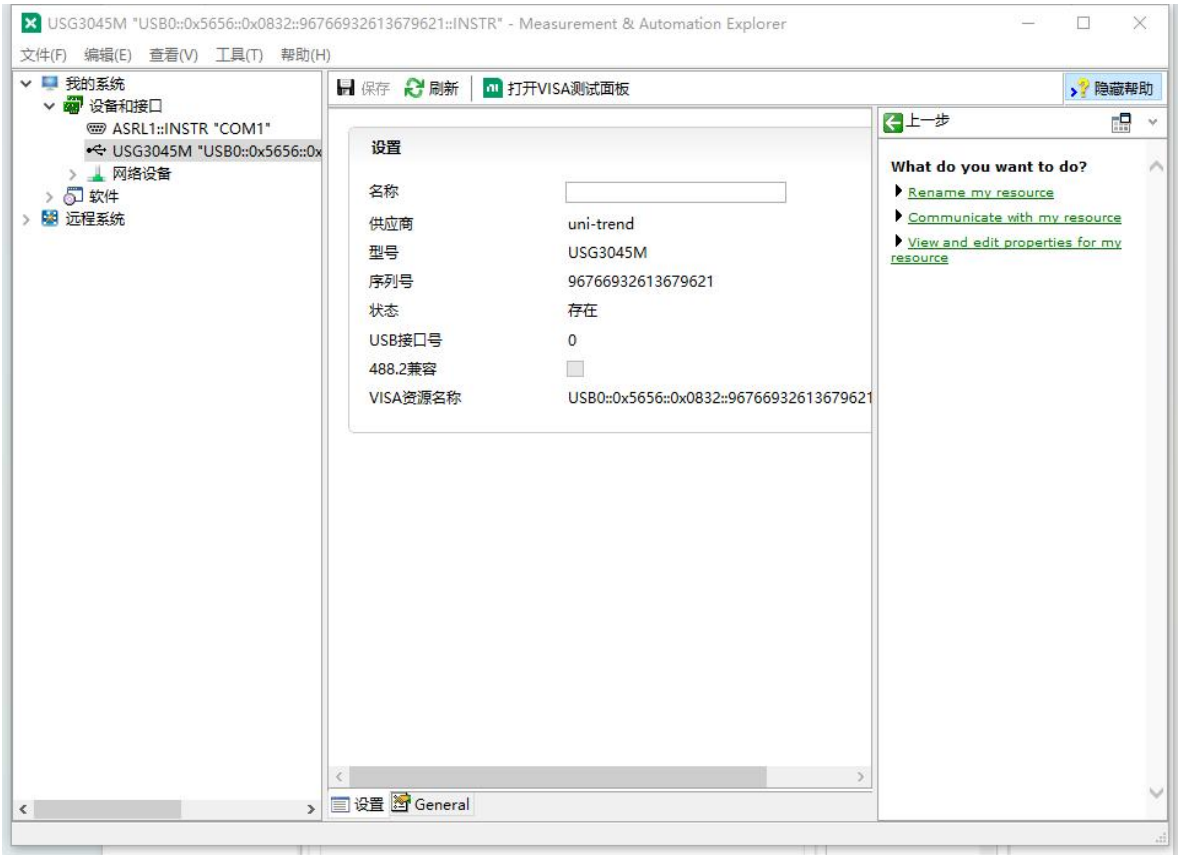
a. 打开射频信号源；

b. 使用USB线将射频信号源的USB Device端口和计算机的USB Host端口连接起来；

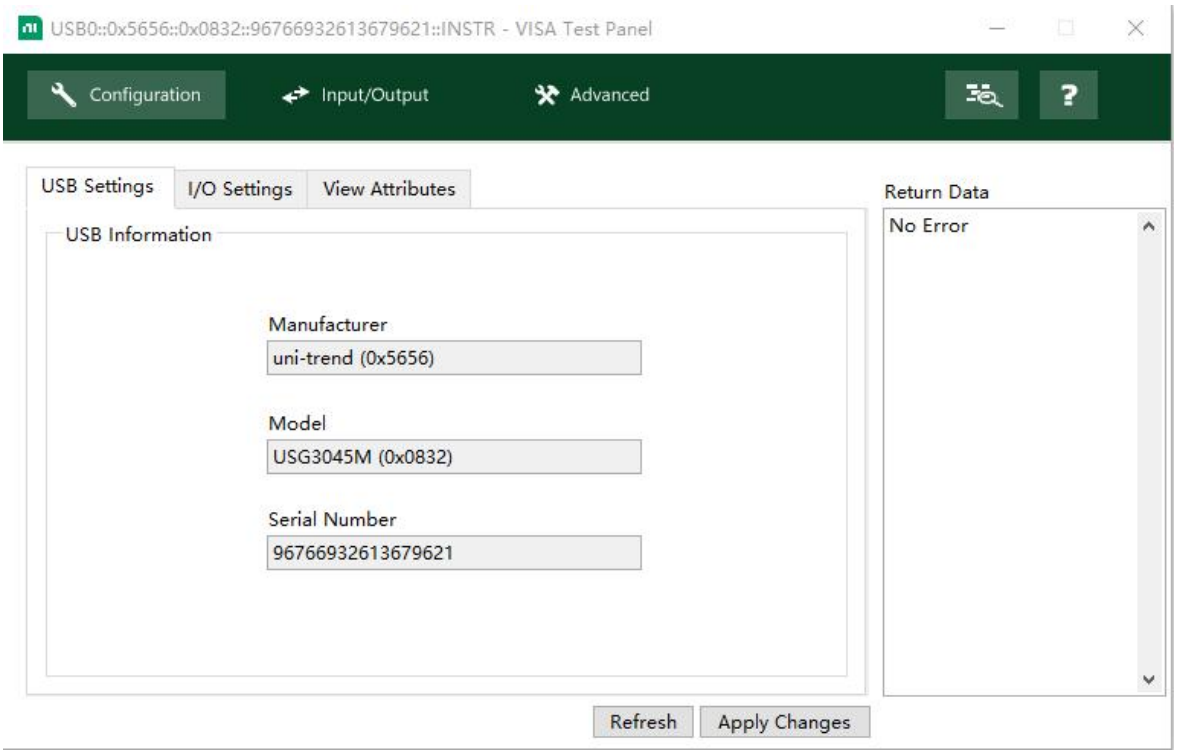
c. 在计算机上打开NI MAX，弹出如下对话框：



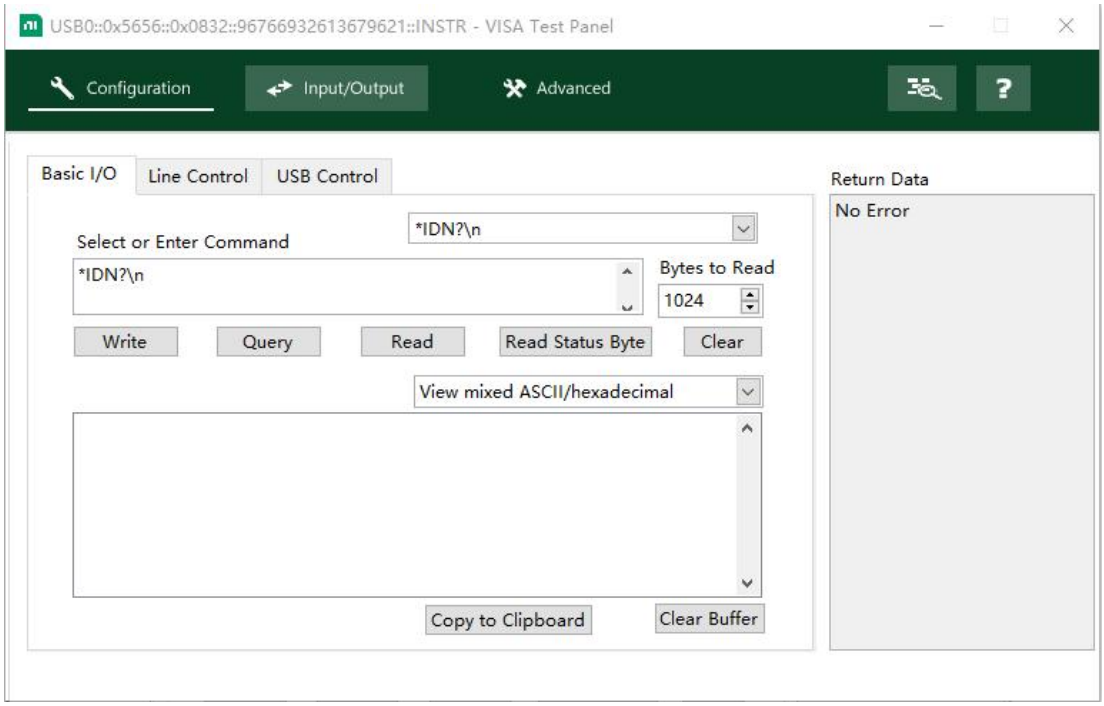
d. 打开设备和接口下拉选项，并选中射频信号源驱动，如下图所示：



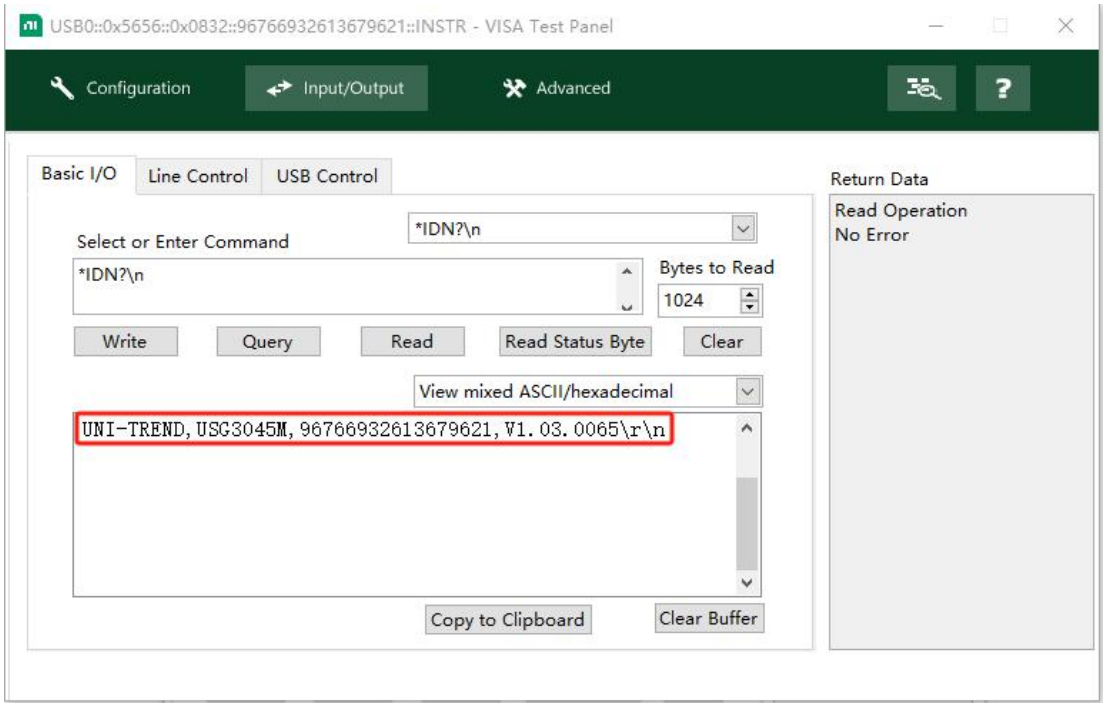
e. 鼠标点击VISA测试面板，弹出如下对话框：



f. 鼠标点击Input/Output选项，如图下图所示：



g. 鼠标点击Query按钮，查询射频信号源的IDN，查询结果如下图红色区域所示：



h. 如果能够查询到射频信号源相关信息，就表示射频信号源已经可以与计算机通信。

VISA 编程示例

本节给出了一些编程示例。通过这些例子，可以了解如何使用VISA，并结合编程手册的命令实现对仪器设备的控制。通过下面的例子，你可以开发更多应用。

VC++示例

- 环境：Window系统, Visual Studio。
- 描述：通过USBTMC 和TCP/IP访问仪器设备，并在NI-VISA上发送"*IDN?"命令来查询设备信息。
- 步骤：

1. 打开 Visual Studio 软件，新建一个 VC++ win32 console project。
2. 设置调用NI-VISA库的项目环境，分别为静态库和动态库。

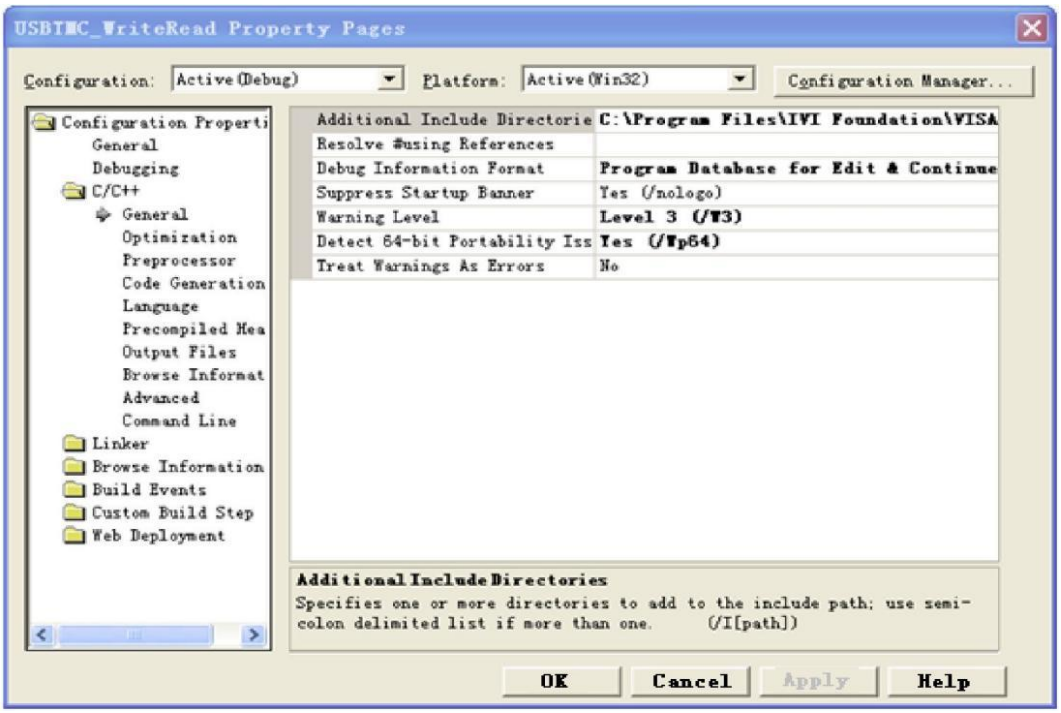
a) 静态库：

在NI-VISA安装路径找:visa.h、visatype.h、visa32.lib文件，将它们复制到VC++项目的根路径下并添加到项目中。在projectname.cpp文件上添加下列两行代码：

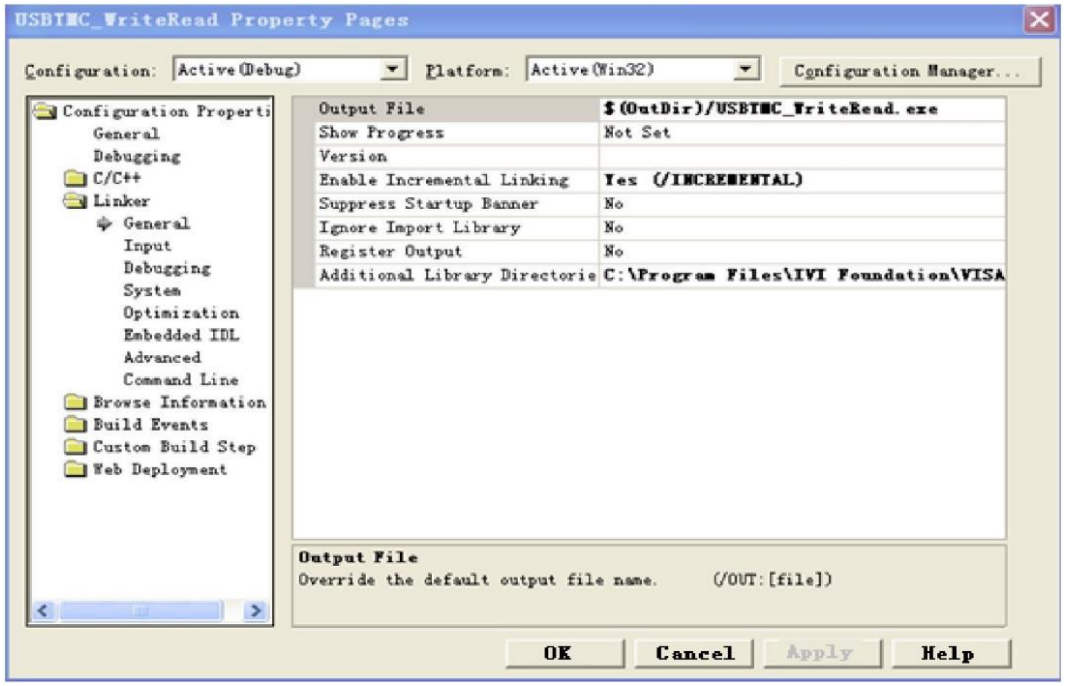
```
#include "visa.h"
#pragma comment(lib,"visa32.lib")
```

b) 动态库：

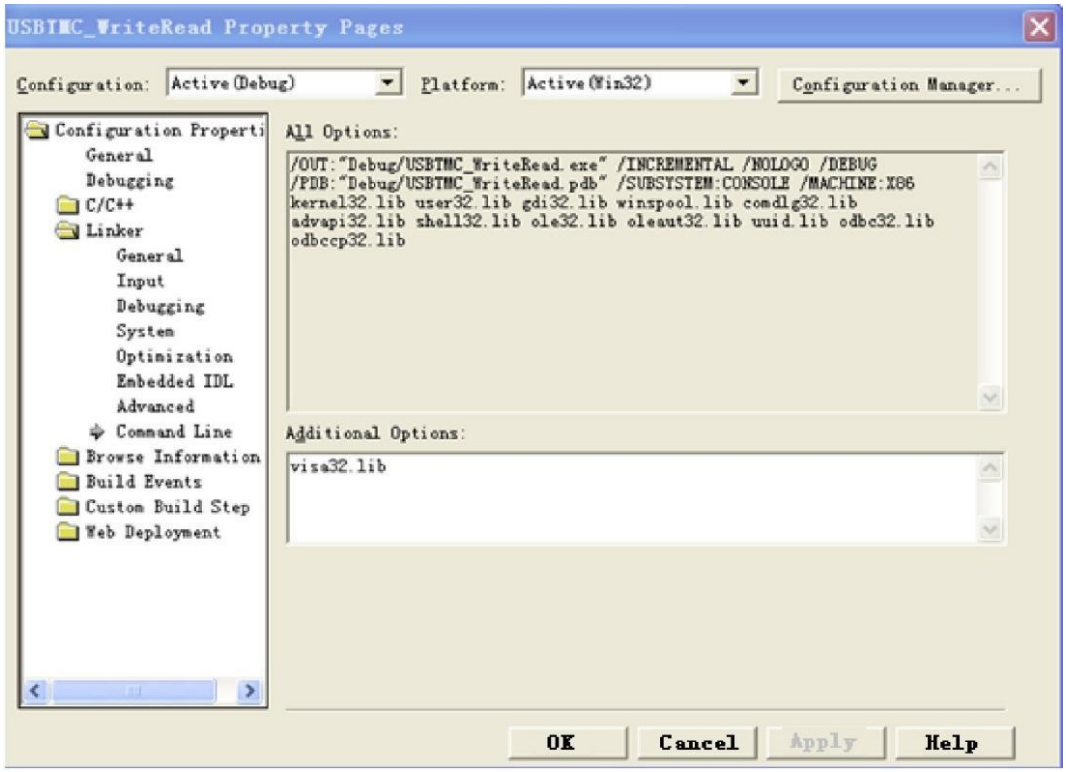
点击"project>>properties"，在属性对话框左侧选择"c/c++---General"中，将 "Additional Include Directories"项的值设置为NI-VISA的安装路径，(例如：C:\ProgramFiles\IVI Foundation\VISA\WinNT\include),如下图所示:



在属性对话框左侧选择"Linker-General",并将"Additional Library Directories"项的值设置为NI-VISA的安装路径，(例如：C:\Program Files\IVI Foundation\VISA\WinNT\include)，如下图所示：



在属性对话框左侧选择"Linker-Command Line",将"Additional"项的值设置为visa32.lib，如下图所示：



在projectname.cpp文件上添加visa.h文件:

#include <visa.h>

1. 源码：

a) USBTMC示例

```
int usbtmc_test()
{ /** This code demonstrates sending synchronous read & write commands
    * to an USB Test & Measurement Class (USBTMC) instrument using NI-VISA
     * The example writes the "*IDN?\n" string to all the USBTMC
     * devices connected to the system and attempts to read back
     * results using the write and read functions.
     * Open Resource Manager
     * Open VISA Session to an Instrument
     * Write the Identification Query Using viPrintf
     * Try to Read a Response With viScanf
     * Close the VISA Session*/
     ViSession defaultRM;
     ViSession instr;
     ViUInt32 numInstrs;
     ViFindList findList;
     ViStatus status;
     char instrResourceString[VI_FIND_BUFLEN];
     unsigned char buffer[100];
     int i;
     status = viOpenDefaultRM(&defaultRM);
     if (status < VI_SUCCESS)
     {
          printf("Could not open a session to the VISA Resource Manager!\n");
          return status;
     }
    /*Find all the USB TMC VISA resources in our system and store the number of resources in the system in
    numInstrs.*/
     status = viFindRsrc(defaultRM, "USB?*INSTR", &findList, &numInstrs, instrResourceString);
     if (status<VI_SUCCESS)
     {
          printf("An error occurred while finding resources. \nPress Enter to continue.");
          fflush(stdin);
          getchar();
          viClose(defaultRM);
          return status;
     }
     /** Now we will open VISA sessions to all USB TMC instruments.
     *     We must use the handle from viOpenDefaultRM and we must
     *     also use a string that indicates which instrument to open. This
```

```
*      is called the instrument descriptor. The format for this string
*      can be found in the function panel by right clicking on the
*      descriptor parameter. After opening a session to the
*      device, we will get a handle to the instrument which we
*      will use in later VISA functions. The AccessMode and Timeout
*      parameters in this function are reserved for future
*      functionality. These two parameters are given the value VI_NULL. */
for (i = 0; i < int(numInstrs); i++)
{
      if (i > 0)
      {
            viFindNext(findList, instrResourceString);
      }
      status = viOpen(defaultRM, instrResourceString, VI_NULL, VI_NULL, &instr);
      if (status < VI_SUCCESS)
      {
            printf("Cannot open a session to the device %d. \n", i + 1);
            continue;
      }
      /** At this point we now have a session open to the USB TMC instrument.
      *We will now use the viPrintf function to send the device the string "*IDN?\n",
      *asking for the device's identification. */
      char * cmmand = "*IDN?\n";
      status = viPrintf(instr, cmmand);
      if (status < VI_SUCCESS)
      {
            printf("Error writing to the device %d. \n", i + 1);
            status = viClose(instr);
            continue;
      }
      /** Now we will attempt to read back a response from the device to
      *the identification query that was sent. We will use the viScanf
      *function to acquire the data.
      *After the data has been read the response is displayed. */
      status = viScanf(instr, "%t", buffer);
      if (status < VI_SUCCESS)
      {
            printf("Error reading a response from the device %d. \n", i + 1);
      }
      else
```

```
        {
            printf("\nDevice %d: %s\n", i + 1, buffer);
        }
        status = viClose(instr);
    }
    /*Now we will close the session to the instrument using viClose. This operation frees all
    system resources.*/
    status = viClose(defaultRM);
    printf("Press Enter to exit.");
    fflush(stdin);
    getchar();
    return 0;
}
int _tmain(int argc, _TCHAR* argv[])
{
    usbtmc_test();
    return 0;
}
```

b) TCP/IP示例

```
int tcp_ip_test(char *pIP)
{
    char outputBuffer[VI_FIND_BUFLEN];
    ViSession defaultRM, instr;
    ViStatus status;
    /* First we will need to open the default resource manager. */
    status = viOpenDefaultRM(&defaultRM);
    if (status < VI_SUCCESS)
    {
        printf("Could not open a session to the VISA Resource Manager!\n");
    }
    /* Now we will open a session via TCP/IP device */
    char head[256] = "TCPIP0::";
    char tail[] = "::inst0::INSTR";
    strcat(head, pIP);
    strcat(head, tail);
    status = viOpen(defaultRM, head, VI_LOAD_CONFIG, VI_NULL, &instr);
    if (status < VI_SUCCESS)
    {
        printf("An error occurred opening the session\n");
```

```
            viClose(defaultRM);
        }
        status = viPrintf(instr, "*idn?\n");
        status = viScanf(instr, "%t", outputBuffer);
        if (status < VI_SUCCESS)
        {
            printf("viRead failed with error code: %x \n", status);
            viClose(defaultRM);
        }
        else
        {
            printf("\nMesseage read from device: %*s\n", 0, outputBuffer);
        }
        status = viClose(instr);
        status = viClose(defaultRM);
        printf("Press Enter to exit.");
        fflush(stdin);
        getchar();
        return 0;
}

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

C#示例

- 环境：Window系统, Visual Studio。
- 描述：通过USBTMC 和TCP/IP访问仪器设备，并在NI-VISA上发送"*IDN?"命令来查询设备信息。
- 步骤：

1. 打开 Visual Studio 软件，新建一个 C# console project。
2. 添加VISA的C#引用Ivi.Visa.dll和NationalInstruments.Visa.dll。
3. 源码：

a) USBTMC示例

```
class Program
{
    void usbtmc_test()
    {
        using (var rmSession = new ResourceManager())
        {
            var resources = rmSession.Find("USB?*INSTR");
            foreach (string s in resources)
            {
                try
                {
                    var mbSession = (MessageBasedSession)rmSession.Open(s);
                    mbSession.RawIO.Write("*IDN?\n");
                    System.Console.WriteLine(mbSession.RawIO.ReadString());
                }
                catch (Exception ex)
                {
                    System.Console.WriteLine(ex.Message);
                }
            }
        }
    }

    void Main(string[] args)
    {
        usbtmc_test();
    }
}
```

b) TCP/IP示例

```
class Program
  {
    void tcp_ip_test(string ip)
    {
        using (var rmSession = new ResourceManager())
        {
            try
            {
                var resource = string.Format("TCPIP0::{0}::inst0::INSTR", ip);
```

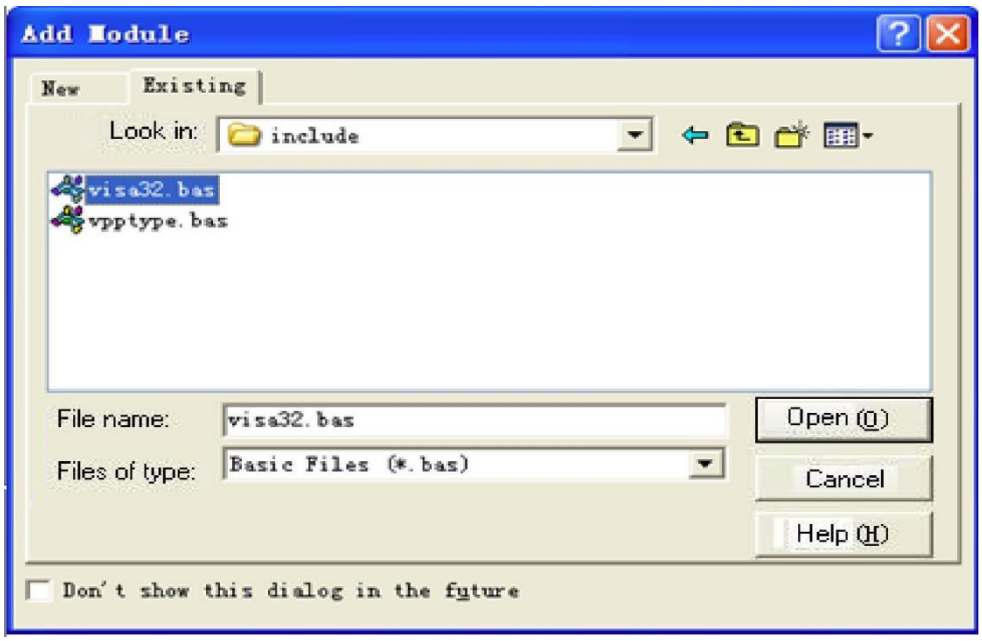
```
                var mbSession = (MessageBasedSession)rmSession.Open(resource);
                mbSession.RawIO.Write("*IDN?\n");
                System.Console.WriteLine(mbSession.RawIO.ReadString());
            }
            catch (Exception ex)
            {
                System.Console.WriteLine(ex.Message);
            }
        }
    }

    void Main(string[] args)
    {
        tcp_ip_test("192.168.20.11");
    }
}
```

VB 示例

- 环境：Window系统, Microsoft Visual Basic 6.0。
- 描述：通过USBTMC 和TCP/IP访问仪器设备，并在NI-VISA上发送"*IDN?"命令来查询设备信息。
- 步骤：

1. 打开Visual Basic软件，并新建一个标准的应用程序项目。
2. 设置调用NI-VISA库项目环境：点击 Existing tab of Project>>Add Existing Item，在 NI-VISA 安装路径下的"include"文件夹中查找visa32.bas文件并添加该文件。如下图所示：



3. 源码：

a) USBTMC示例

```
PrivateFunction usbtmc_test() AsLong
' This code demonstrates sending synchronous read & write commands
' to an USB Test & Measurement Class (USBTMC) instrument using NI-VISA
' The example writes the "*IDN?\n" string to all the USBTMC
' devices connected to the system and attempts to read back
' results using the write and read functions.
' The general flow of the code is
' Open Resource Manager
' Open VISA Session to an Instrument
' Write the Identification Query Using viWrite
' Try to Read a Response With viRead
' Close the VISA Session


Const MAX_CNT = 200
Dim defaultRM AsLong
Dim instrsesn AsLong
Dim numInstrs AsLong
Dim findList AsLong
Dim retCount AsLong
Dim status AsLong
Dim instrResourceString AsString *VI_FIND_BUFLEN
Dim Buffer AsString * MAX_CNT
Dim i AsInteger


' First we must call viOpenDefaultRM to get the manager
```

```
' handle. We will store this handle in defaultRM.
status = viOpenDefaultRM(defaultRM)
If(status < VI_SUCCESS) Then
    resultTxt.Text = "Could not open a session to the VISA Resource Manager!"
    usbtmc_test = status
ExitFunction
EndIf

' Find all the USB TMC VISA resources in our system and store the
' number of resources in the system in numInstrs.
status = viFindRsrc(defaultRM, "USB?*INSTR", findList, numInstrs, instrResourceString)
If (status < VI_SUCCESS) Then
    resultTxt.Text = "An error occurred while finding resources."
    viClose(defaultRM)
    usbtmc_test = status
ExitFunction
EndIf

' Now we will open VISA sessions to all USB TMC instruments.
' We must use the handle from viOpenDefaultRM and we must
' also use a string that indicates which instrument to open.  This
' is called the instrument descriptor.  The format for this string
' can be found in the function panel by right clicking on the
' descriptor parameter. After opening a session to the
' device, we will get a handle to the instrument which we
' will use in later VISA functions.  The AccessMode and Timeout
' parameters in this function are reserved for future
' functionality.  These two parameters are given the value VI_NULL.
For i = 0 To numInstrs
If (i > 0) Then
    status = viFindNext(findList, instrResourceString)
EndIf
    status = viOpen(defaultRM, instrResourceString, VI_NULL, VI_NULL, instrsesn)
If (status < VI_SUCCESS) Then
    resultTxt.Text = "Cannot open a session to the device " + CStr(i + 1)
GoTo NextFind
EndIf

' At this point we now have a session open to the USB TMC instrument.
' We will now use the viWrite function to send the device the string "*IDN?",
```

```
' asking for the device's identification.
status = viWrite(instrsesn, "*IDN?", 5, retCount)
If (status < VI_SUCCESS) Then
    resultTxt.Text = "Error writing to the device."
    status = viClose(instrsesn)
GoTo NextFind
EndIf

' Now we will attempt to read back a response from the device to
' the identification query that was sent.  We will use the viRead
' function to acquire the data.
' After the data has been read the response is displayed.
status = viRead(instrsesn, Buffer, MAX_CNT, retCount)
If (status < VI_SUCCESS) Then
    resultTxt.Text = "Error reading a response from the device." + CStr(i + 1)
Else
    resultTxt.Text = "Read from device: " + CStr(i + 1) + " " + Buffer
EndIf
    status = viClose(instrsesn)
Next i

' Now we will close the session to the instrument using
' viClose. This operation frees all system resources.
status = viClose(defaultRM)
usbtmc_test = 0
EndFunction
```

b) TCP/IP示例

```
PrivateFunction tcp_ip_test(ByVal ip AsString) AsLong
Dim outputBuffer AsString * VI_FIND_BUFLEN
Dim defaultRM AsLong
Dim instrsesn AsLong
Dim status AsLong
Dim count AsLong

' First we will need to open the default resource manager.
status = viOpenDefaultRM(defaultRM)
If (status < VI_SUCCESS) Then
    resultTxt.Text = "Could not open a session to the VISA Resource Manager!"
    tcp_ip_test = status
```

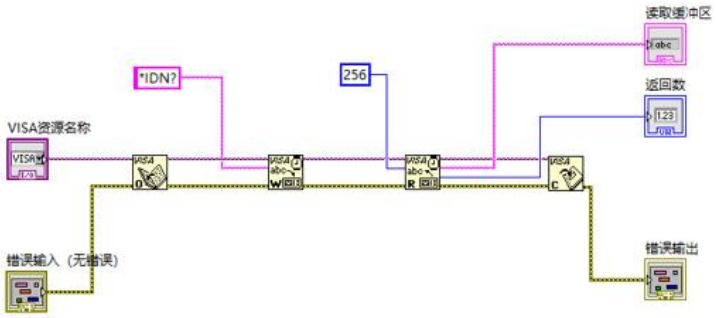
```
ExitFunction
EndIf

' Now we will open a session via TCP/IP device
status = viOpen(defaultRM, "TCPIP0::" + ip + "::inst0::INSTR", VI_LOAD_CONFIG, VI_NULL, instrsesn)
If (status < VI_SUCCESS) Then
    resultTxt.Text = "An error occurred opening the session"
    viClose(defaultRM)
    tcp_ip_test = status
ExitFunction
EndIf
status = viWrite(instrsesn, "*IDN?", 5, count)
If (status < VI_SUCCESS) Then
    resultTxt.Text = "Error writing to the device."
EndIf
    status = viRead(instrsesn, outputBuffer, VI_FIND_BUFLEN, count)
If (status < VI_SUCCESS) Then
    resultTxt.Text = "Error reading a response from the device." + CStr(i + 1)
Else
    resultTxt.Text = "read from device:" + outputBuffer
EndIf
    status = viClose(instrsesn)
    status = viClose(defaultRM)
    tcp_ip_test = 0
EndFunction
```

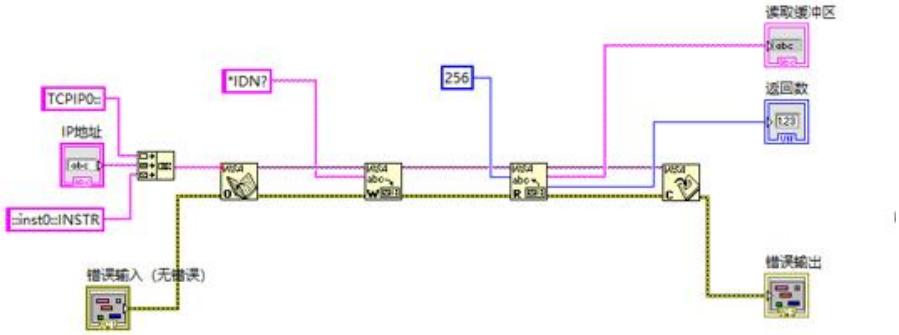
LabIEW 示例

- 环境：Window系统, LabVIEW。
- 描述：通过USBTMC和TCP/IP访问仪器设备，并在NI-VISA上发送"*IDN?"命令来查询设备信息。
- 步骤：

1. 打开LabVIEW 软件，并创建一个VI文件。
2. 添加控件，右击前面板界面，从控制列中选择并添加 VISA资源名、错误输入、错误输出以及部分的指示符。
3. 打开框图界面，右击 VISA资源名称，并在弹出菜单的VISA面板中选择和添加下列功能：VISA Write、VISA Read、VISA Open和 VISA Close。
4. VI打开了一个USBTMC设备的VISA会话，并向设备写*IDN?命令并回读的响应值。当所有通信完成时，VI将关闭VISA会话，如下图所示：



5. 通过TCP/IP与设备通信类似于USBTMC,但是你需要将VISA写函数和VISA 读函数设置为同步I/O,LabVIEW 默认设置为异步IO。右键单击节点，然后从快捷菜单中选择,"Synchronous I/O Mode>>Synchronous"以实现同步写入或读取数据，如下图所示：



MATLAB 示例

- 环境：Window系统, MATLAB。
- 描述：通过USBTMC 和TCP/IP访问仪器设备，并在NI-VISA上发送"*IDN?"命令来查询设备信息。
- 步骤：

1. 打开MATLAB软件，点击在 Matlab界面的 File>>New>>Script 创建一个空的M文件。
2. 源码：

a) USBTMC示例

```
function usbtmc_test()
% This code demonstrates sending synchronous read & write commands
% to an USB Test & Measurement Class (USBTMC) instrument using
% NI-VISA

%Create a VISA-USB object connected to a USB instrument
vu = visa('ni','USB0::0x5345::0x1234::SN20220718::INSTR');

%Open the VISA object created
fopen(vu);

%Send the string "*IDN?",asking for the device's identification.
```

```
fprintf(vu,'*IDN?');

%Request the data

outputbuffer = fscanf(vu);
disp(outputbuffer);

%Close the VISA object
fclose(vu);
delete(vu);
clear vu;

end
```

b) TCP/IP示例

```
function tcp_ip_test()
% This code demonstrates sending synchronous read & write commands
% to an TCP/IP instrument using NI-VISA
%Create a VISA-TCPIP object connected to an instrument

%configured with IP address.
vt = visa('ni',['TCPIP0::','192.168.20.11','::inst0::INSTR']);

%Open the VISA object created

fopen(vt);

%Send the string "*IDN?",asking for the device's identification.
fprintf(vt,'*IDN?');

%Request the data
outputbuffer = fscanf(vt);
disp(outputbuffer);

%Close the VISA object
fclose(vt);
delete(vt);
clear vt;

end
```

Python 示例

- 环境：Window系统, Python3.8, PyVISA 1.11.0。
- 描述：通过USBTMC 和TCP/IP访问仪器设备，并在NI-VISA上发送"*IDN?"命令来查询设备信息。
- 步骤：

1. 首先安装python，然后打开Python脚本编译软件，创建一个空的test.py文件。
2. 使用pip install PyVISA指令安装PyVISA，如无法安装，请参考此链接使用说明 (https://pyvisa.readthedocs.io/en/latest/)
3. 源码：

a) USBTMC示例

```
import pyvisa
rm = pyvisa.ResourceManager()
rm.list_resources()
my_instrument = rm.open_resource('USB0::0x5345::0x1234::SN20220718::INSTR')
print(my_instrument.query('*IDN?'))
```

b) TCP/TP示例

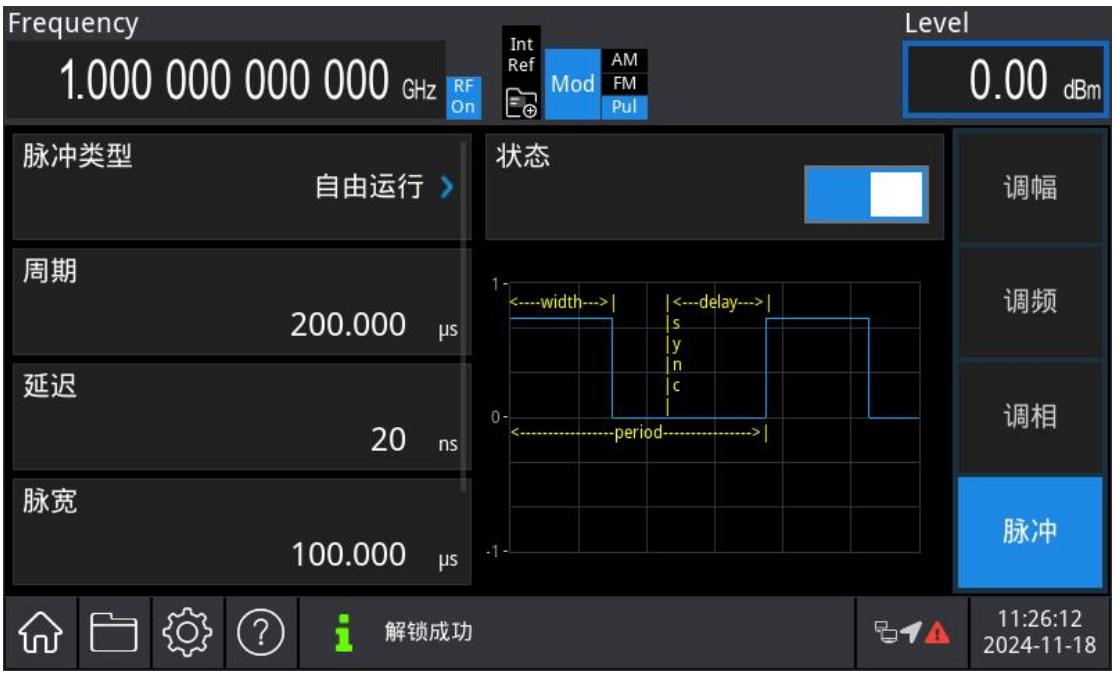
```
import pyvisa
rm = pyvisa.ResourceManager()
rm.list_resources()
my_instrument = rm.open_resource('TCPIP0::192.168.20.11::inst0::INSTR')
print(my_instrument.query('*IDN?'))
```

编程应用实例

本部分将介绍运用 SCPI 命令让射频信号源输出脉冲调制信号。

设置射频信号源参数：

先 default，让参数恢复到默认设置；打开射频输出，设置射频参数，频率、幅度；打开模拟调制，选择脉冲调制，选择脉冲类型 FREErun，设置脉冲参数，周期、延迟、脉宽、同步脉宽，打开脉冲输出。



以下命令可进行如上所属的操作，获得上图所示输出信号。

```
:KEY:DEFault        //恢复默认设置
:OUTPut ON          //打开射频输出
:FREQuency 1GHz     //设置输出频率为1GHz
:POWer -10dBm       //设置幅度-10dBm
:MOD:STATe ON       //打开模拟调制
:MOD:PULSe:TYPE FREErun      //设置脉冲类型FREErun
:MOD:PULSe:PERIod 200us      //设置脉冲周期200微秒
:MOD:PULSe:DELAy 20ns        //设置脉冲延时20纳秒
:MOD:PULSe:WIDTh 100us       //设置脉冲脉宽100微秒
:MOD:PULSe:SYNCwidth 1us     //设置脉冲同步脉宽1微秒
:MOD:PULSe:EN ON             //打开脉冲调制输出
```

附录 1：<key>列表

按键命令关键字	功能描述	LED灯
HOMe	主页	
UTILity	系统设置	
DEFAult	恢复出厂设置	
TLock	触屏锁	√
FREQ	频率	
AMPT	幅度	
SWEep	扫描	
MODe	模块	
AM	调幅	
FM	调频	
PULSe	脉冲调制	
IQ	矢量源	
NUM7	数字键7	
NUM4	数字键4	
NUM1	数字键1	
DOT	数字键小数点	
ESC	退出	
NUM8	数字键8	
NUM5	数字键5	
NUM2	数字键2	
NUM0	数字键0	
BACKspace	退格	
NUM9	数字键9	
NUM6	数字键6	
NUM3	数字键3	
SYMBOL	数字键符号	
ENTER	回车	
GN	单位	
MU	单位	
KM	单位	
DBM	单位	
FKNob	确认	
UP	方向键上	

LEFT	方向键左	
TRIGger	触发	
LF	低频	√
MODS	调制	√
RF	射频	√
RIGHT	方向键右	
DOWN	方向键下	
FKNLeft	旋钮左旋	
FKNRight	旋钮右旋	

附录 2：按键锁定状态

位序	按键	状态
0	HOMe	0表示未锁定，1表示锁定
1	UTILity	0表示未锁定，1表示锁定
2	DEFAult	0表示未锁定，1表示锁定
3	TLock	0表示未锁定，1表示锁定
4	FREQ	0表示未锁定，1表示锁定
5	AMPT	0表示未锁定，1表示锁定
6	SWEep	0表示未锁定，1表示锁定
7	MODe	0表示未锁定，1表示锁定
8	AM	0表示未锁定，1表示锁定
9	FM	0表示未锁定，1表示锁定
10	PULSe	0表示未锁定，1表示锁定
11	IQ	0表示未锁定，1表示锁定
12	NUM7	0表示未锁定，1表示锁定
13	NUM4	0表示未锁定，1表示锁定
14	NUM1	0表示未锁定，1表示锁定
15	DOT	0表示未锁定，1表示锁定
16	ESC	0表示未锁定，1表示锁定
17	NUM8	0表示未锁定，1表示锁定
18	NUM5	0表示未锁定，1表示锁定
19	NUM2	0表示未锁定，1表示锁定
20	NUM0	0表示未锁定，1表示锁定
21	BACKspace	0表示未锁定，1表示锁定
22	NUM9	0表示未锁定，1表示锁定

23	NUM6	0表示未锁定，1表示锁定
24	NUM3	0表示未锁定，1表示锁定
25	SYMBOL	0表示未锁定，1表示锁定
26	ENTER	0表示未锁定，1表示锁定
27	GN	0表示未锁定，1表示锁定
28	MU	0表示未锁定，1表示锁定
29	KM	0表示未锁定，1表示锁定
30	DBM	0表示未锁定，1表示锁定
31	FKNob	0表示未锁定，1表示锁定
32	UP	0表示未锁定，1表示锁定
33	LEFT	0表示未锁定，1表示锁定
34	TRIGger	0表示未锁定，1表示锁定
35	LF	0表示未锁定，1表示锁定
36	MODS	0表示未锁定，1表示锁定
37	RF	0表示未锁定，1表示锁定
38	RIGHT	0表示未锁定，1表示锁定
39	DOWN	0表示未锁定，1表示锁定
40	FKNLeft	0表示未锁定，1表示锁定
41	FKNRight	0表示未锁定，1表示锁定