SDG6000X

函数/任意波形发生器

编程手册

CN03A

深圳市鼎阳科技股份有限公司
SIGLENT TECHNOLOGIES CO., LTD.

目录

1 编程概述

用户可以通过使用信号源的 USB、LAN 或 GPIB 端口，并结合 NI-VISA 和程序语言，远程控制信号源。基于 LAN 端口，SDG 支持 VXI-11、Sockets 和 Telnet 通信协议。本节介绍了如何建立 SDG 系列函数/任意波形发生器和计算机之间的通信，同时介绍如何远程控制信号源。

1.1 建立通信

1.1.1 NI-VISA 的安装

在编程之前，请确保正确安装 NI-VISA 软件的最新版本。

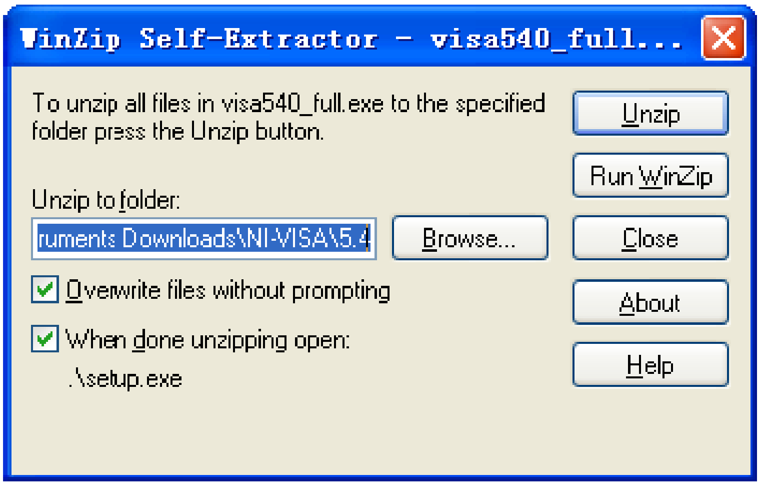
NI-VISA 是用于计算机与设备之间通信的通信库。NI 软件有两种有效 VISA 安装包：完整版和运行引擎版（Run-Time Engine）。完整版包括 NI 设备驱动和 NI MAX 工具，其中 NI MAX 是用于控制设备的用户界面。虽然驱动和 NI MAX 都很有用，但是它们不用于远程控制。运行引擎版（Run-Time Engine）是一个比完整版更小的文件，它主要用于远程控制。

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-VISA5.4 完整版）：

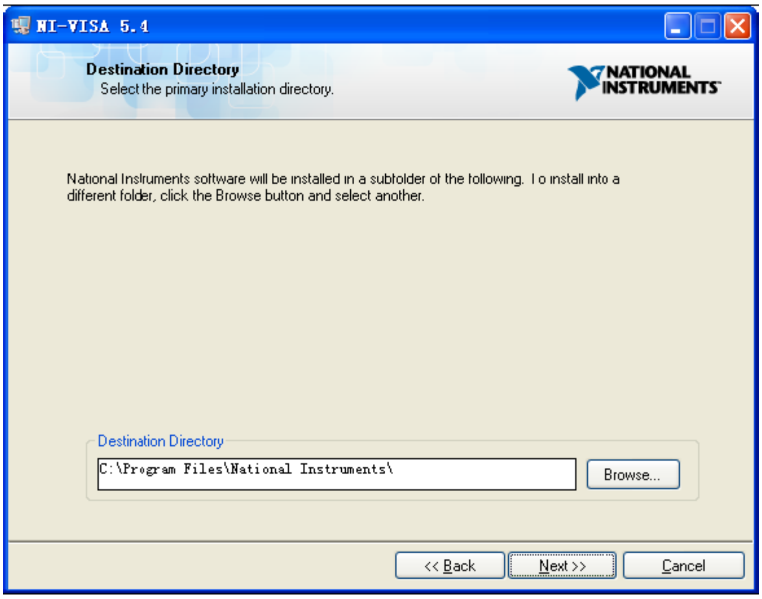
a. 下载合适版本的 NI-VISA（推荐运行引擎版）

b. 双击 visa540_full.exe，弹出对话框如下：

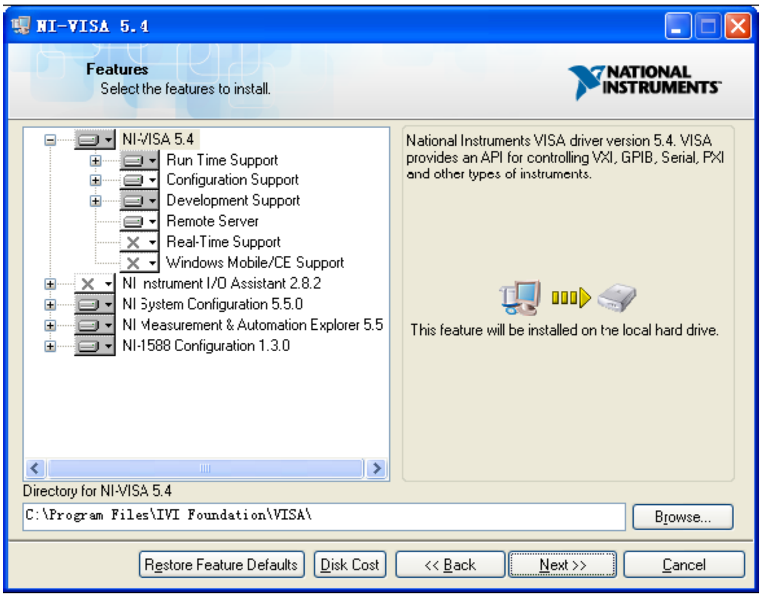


c. 点击 Unzip 解压文件，当解压完成后，安装程序将自动执行。若你的计算机需要安装.NET Framework4，则在安装过程会自动安装.NET Framework4。

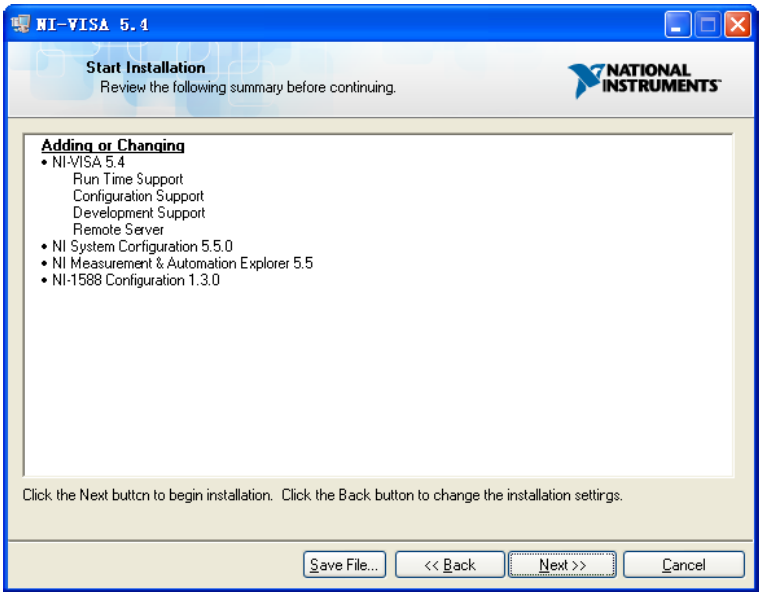
d. NI-VISA 安装对话框如上图所示，点击 Next 开始安装过程。



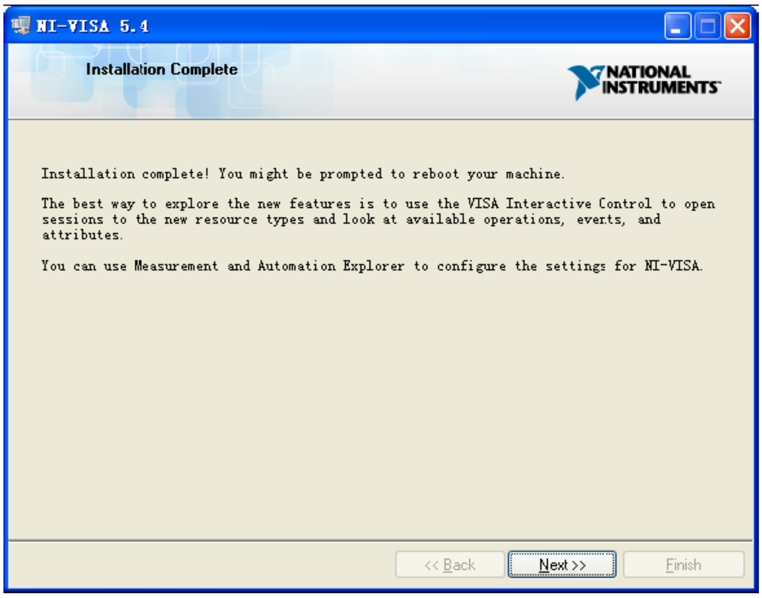
e. 设置安装路径，默认路径为“C:\Program Files\National Instruments\”。你也可以修改安装路径。点击 Next，对话框如下图所示。



f. 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：



g. 点击 Next 开始安装：

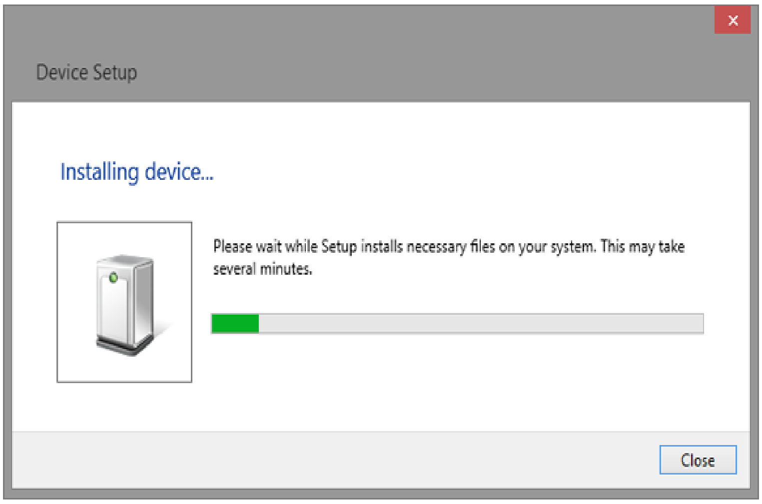


h. 安装完成后，重启电脑。

1.1.2 连接仪器

根据具体信号源机型，函数/任意波形发生器能够通过 USB、LAN 端口或 GPIB（选配）接口连接计算机。

使用 USB 线将函数/任意波形发生器的 USB Device 端口和计算机的 USBHost 端口连接起来。假设你的计算机已经启动，打开信号源后，桌面将弹出“设备安装”界面，并自动安装设备驱动，如下图所示：



等待安装完成，然后进行下一步。

1.2 远程控制的实现

1.2.1 用户自定义程序

用户可通过计算机发送 SCPI 命令实现编程和控制任意波形发生器。相关内容，请查阅“编程示例”中的介绍。

1.2.2 通过 NI-MAX 发送 SCPI 命令

NI-MAX 是由 NI 公司创建和维护的程序。它为 VXI、LAN、USB、GPIB 和串行通信提供基础的远程控制接口。用户可以通过 NI-MAX 发送 SCPI 命令远程控制信号源。

1.2.3 通过 Telnet 发送 SCPI 命令

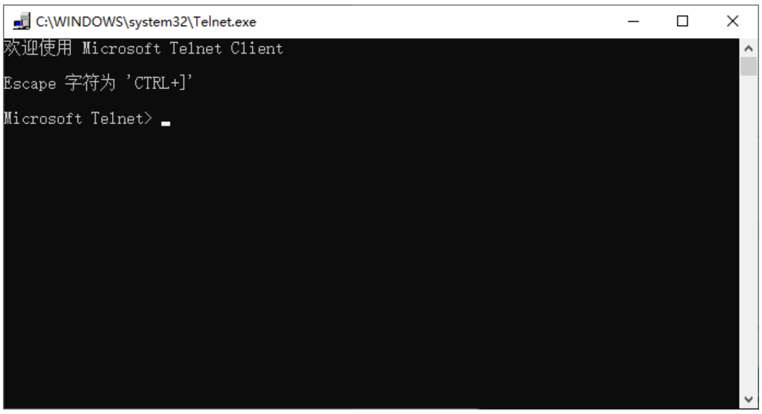
Telnet 提供一种通过 LAN 端口与信号源通信的方式。Telnet 协议支持从计算机向信号源发送 SCPI 命令，该方式类似于通过 USB 与信号源通信。发送和接受信息是交互的：一次只能发送一个命令。Windows 操作系统使用命令提示符样式接口作为 Telnet 客户端。

步骤如下：

1 在计算机桌面，点击开始>所有程序>附件>命令提示符

2 在命令提示符窗口，输入 Telnet

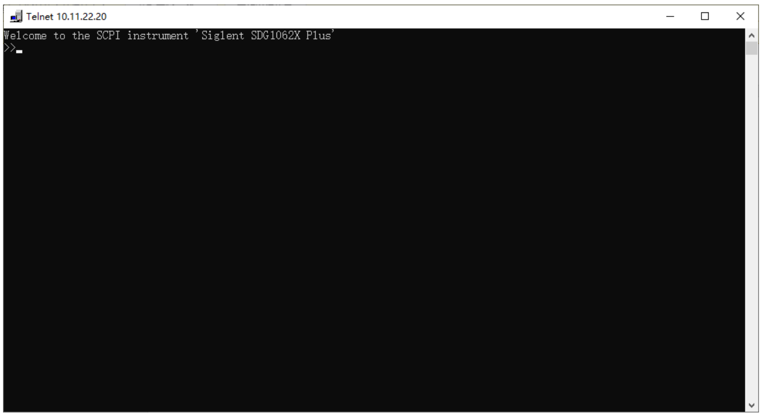
3 按下 Enter 键。将弹出 Telnet 显示窗



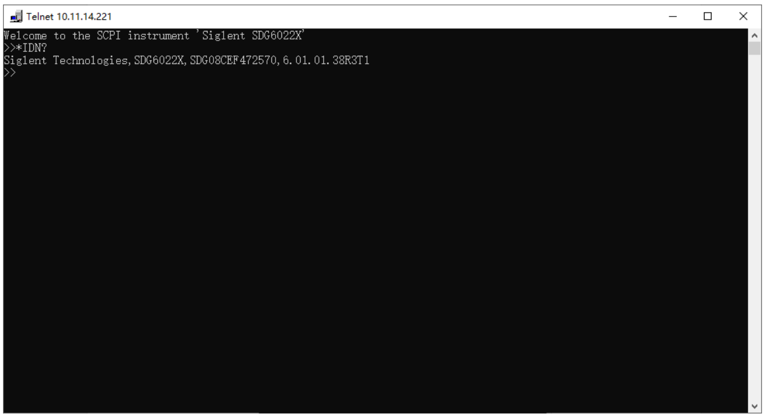
4 在 Telnet 命令行，键入：

open XXX.XXX.XXX.XXX 5024

其中 XXX.XXX.XXX.XXX 是指设备的 IP 地址；5024 是端口。通信成功后你会看到和下面类似的响应内容：



5　在“>>”提示符后，输入 SCPI 命令例如 **IDN?*。该命令将返回公司名、机器型号、序列号和固件版本号。



6　同时按下 Ctrl+] 键将退出 SCPI 会话。

7　通过在提示符出键入 *quit* 或关闭 Telnet 窗口来关闭设备的连接并退出 Telnet。

1.2.4　通过 Socket 发送 SCPI

Socket 接口可以在不安装其他库的情况下通过 LAN 端口控制 SDG 系列产品。它可以减少编程的复杂度。

SOCKET 地址	IP 地址+端口号
IP 地址	SDG IP 地址
端口号	5025

相关内容，请查阅第 5.2 节“Sockets 应用示例”。

2 SCPI 语言简介

2.1 有关命令和查询

本节列出并描述仪器识别的远程控制命令和查询语句。所有命令和查询都可以在本地或远程状态下执行。

本文对每个包含有语法以及其他信息的命令和查询都列举出一些例子。在“命令语法”和“查询语法”中，本文给出了该命令的长格式和短格式。SCPI 命令头部可分为表示命令、查询或命令和查询。可根据 SCPI 命令头部后面跟着问号（？）来识别查询操作，例如获取信息。

2.2 描述

在描述中给出了执行功能的简要说明。接下来是命令的正式语法的表示，其中命令头部使用大小写字符和从大写字符派生出来的缩写形式。在适用的情况下，查询的语法以其响应的格式给出。

2.3 用法

本手册列出的命令和查询可以用在 SDG6000X 系列的函数/任意波形发生器上。

2.4 命令符号

下面是用在命令中的符号：

<> 角括号包含用于占位的字符，其中分为两种类型：标题路径和命令参数。

:= 冒号后面跟着等号，它用于分隔占位符和占位符的描述，占位符的描述是指用于替换命令中占位符的参数类型和范围。

{ } 花括号列出了可供选择的参数，至少要选择一个参数。

[ ] 方括号包含可选项。

… 省略号表示其左侧和右侧可以重复多次。

2.5 命令&查询表

短格式	长格式	子系统	命令/查询的作用
*IDN	*IDN	系统	获取设备 id
*OPC	*OPC	系统	设置或获取事件状态寄存器（ESR）的 OPC 位。
*RST	*RST	系统	设为默认设置。
CHDR	COMM_HEADER	信号	设置或获取命令格式
OUTP	OUTPUT	信号	设置或获取输出状态。
BSWV	BASIC_WAVE	信号	设置或获取基本波形参数。
MDWV	MODULATEWAVE	信号	设置或获取调制参数。
SWWV	SWEEPWAVE	信号	设置或获取扫频参数。
BTWV	BURSTWAVE	信号	设置或获取脉冲串功能参数
PACP	PARACOPY	信号	将参数从通道复制到另一通道
ARWV	ARBWAVE	数据	设置任意波类型
SYNC	SYNC	信号	设置或获取同步信号
NBFM	NUMBER_FORMAT	系统	设置或获取数据格式
LAGG	LANGUAGE	系统	设置或获取语言
SCFG	SYS_CFG	系统	设置或获取上电开机模式
BUZZ	BUZZER	系统	设置或获取蜂鸣器状态
SCSV	SCREEN_SAVE	系统	设置或获取屏幕保护状态
ROSC	ROSCILLATOR	信号	设置或获取时钟源的状态
FCNT	FREQCOUNTER	信号	设置或获取频率计参数
INVT	INVERT	信号	设置或获取当前通道的极性
COUP	COUPLING	信号	设置或获取通道耦合参数
VOLTPRT	VOLTPRT	系统	设置或获取过压保护的状态
STL	STORELIST	信号	列出所有存储波形
WVDT	WVDT	信号	设置或获取任意波波形数据
SYST:COMM:LAN:IPAD	SYSTEM:COMMUNICATE:LAN:IPADDRESS	系统	设置或获取设备 IP 地址
SYST:COMM:LAN:SMAS	SYSTEM:COMMUNICATE:LAN:SMASK	系统	设置或获取设备子网掩码

短格式	长格式	子系统	命令/查询的作用
SYST:COMM:LAN:GAT	SYSTEM:COMMUNICATE:LAN:GATEWAY	系统	设置或获取设备网关
HARM	HARMonic	信号	设置或获取谐波信息
CMBN	CoMBiNe	信号	设置或获取波形合并信息
MODE	MODE	信号	设置或获取波形的相位模式

3 命令与查询

3.1 IEEE488.2 通用命令介绍

IEEE 标准定义的通用命令适用于查询设备的基本信息和执行基本操作。这些命令通常以“*”开头以及命令的关键字长度为 3 个字符。

3.1.1 *IDN

描述	*IND?是用于查询设备的 id。响应包含厂商、机型、序列号、软件版本和硬件版本。
查询语法	*IDN?
响应格式	格式 1：*IDN，<设备 id>，<机型>，<序列号>，<软件版本>，<硬件版本>。 格式 2：<设备商>，<机型>，<序列号>，<软件版本>，<硬件版本>。 <设备 id>：=“SDG”被用于识别仪器。 <设备商>：=“Siglent Technologies”。 <机型>：= 机器型号。 <序列号>：= 每个产品有自己的编号，此序列号标注产品的唯一性。 <软件版本>：= 与软件版本信息相关的编号。 <硬件版本>：= 固件级字段应该包含所有可单独修改子系统的信息。这些信息包含在单个或多个修订版本代码中。
示例	读取版本信息。 **IDN?* 返回值： *Siglent Technologies,SDG6022X,SDG08CEF472570,* *6.01.01.38R3* **注意：**每个版本可能不同

备注：

1. <硬件版本>格式：value1-value2-value3-value4-value5.

   value1：PCB 版本；

   value2：硬件版本；

   value3：硬件修订版；

   value4：FPGA 版本；

   value5：CPLD 版本。

3.1.2 *OPC

描述	*OPC（操作完成）命令设置在标准事件状态寄存器 [ESR] 的 OPC 位（位 0）。此命令对设备操作无其他影响。因为仪器是在处理完上一条设置或查询语句之后，才开始解析设置或查询命令。 *OPC?查询命令总是返回 ASCII 码的 1。因为在前一个命令已经完全执行完后设备才响应该查询命令。
命令语法	*OPC
查询语法	*OPC?
响应格式	1

3.1.3 *RST

描述	*RST 命令启动仪器重置并调用默认设置。
命令语法	*RST
示例	该例子将信号发生器重置成默认设置: **RST*

3.2 输出命令

描述	启用和禁用通道前面板[OUTPUT]接口的输出。 查询结果返回“ON”、“OFF”、LOAD 和 PLRT 参数。
命令语法	<通道>：OUTPutON|OFF，LOAD，<负载>，PLRT，<极性> <通道>：={C1，C2}. <负载>：={50, HZ }. <极性>：={NOR，INVT}，其中 NOR 代表正常，INVT 代表反相。
查询语法	<通道>：OUTPut?
响应格式	<通道>：OUTP ON|OFF，LOAD，<负载>，PLRT，<极性>
示例	打开 CH1： *C1:OUTP ON* 读取 CH1 输出状态： *C1:OUTP?* 返回值： *C1:OUTP ON,LOAD,HZ,PLRT,NOR* 设置 CH1 的负载为 50ohm： *C1:OUTP LOAD,50* 设置 CH1 的负载为高阻： *C1:OUTP LOAD,HZ* 设置 CH1 的极性为正常： *C1:OUTP PLRT,NOR*

备注：* "HZ"代表高阻态。

描述	同时设置两通道的输出状态
命令语法	OUT_BOTHCH <STATE> <STATE>：={ON，OFF}.
示例	两通道打开输出： *OUT_BOTHCH ON*

3.3 噪声叠加命令

描述	开启噪声叠加，并以指定信噪比向通道添加噪声
命令语法	<通道>：NOISE_ADD STATE，ON|OFF，RATIO，<值> 或 <通道>：NOISE_ADD STATE，ON|OFF，RATIO_DB，<值(dB)> <通道>：={C1，C2}. 信噪比的值详见数据手册
查询语法	<通道>：NOISE_ADD?
响应格式	<通道>：NOISE_ADD STATE，ON|OFF，RATIO，<值> 或 <通道>：NOISE_ADD STATE，ON|OFF，RATIO_DB，<值(dB)>
示例	打开通道一噪声叠加并设置信噪比为 120： *C1:NOISE_ADD STATE,ON,RATIO,120*

3.4 基本波形命令

描述	设置或者获取基本波参数。
命令语法	<通道>：BaSic_WaVe <参数>,<值> <通道>：={C1, C2}. <参数>：= MAX_OUTPUT_AMP <值>：={相关参数的值}。

参数	**值**	**描述**
WVTP	<type>	:={SINE，SQUARE，RAMP，PULSE，NOISE，ARB，DC，PRBS，MULTPULSE，SEQUENCE，IQ}。如果命令未设置基本波形类型，WVPT 默认设置为当前波形。
FRQ	<frequency>	:=频率。单位是赫兹“Hz”，可在数据手册查阅参数值的有效范围。当 WVTP 为噪声、直流和多脉冲，该值无效。
PERI	<period>	:=周期。单位是秒“s”。可在数据手册查阅参数值的有效范围。当 WVTP 为噪声、直流和多脉冲，该值无效。
AMP	<amplitude>	:=幅值。单位是伏特，峰峰值"Vpp"。可在数据手册查阅参数值的有效范围。当 WVTP 为噪声和直流，该参数无效。
OFST	<offset>	:=偏置。单位是伏特“V”。可在数据手册查阅参数值的有效范围。当 WVTP 为噪声，该值无效。
SYM	<symmetry>	:= {0 至 100}。三角波的对称度。单位是“%”。仅当 WVTP 为三角波才能设置该参数。
DUTY	<duty>	:={0 至 100}。占空比。单位是“%”。该参数的值取决于频率。仅当 WVTP 是方波和脉冲才能设置该参数。
PHSE	<phase>	:= {0 至 360}。单位是“°”，当 WVTP 是噪声、脉冲波、直流时，该参数无效。
STDEV	<stdev>	:=噪声的标准差。单位是伏特“V”。可在数据手册查阅参数值的有效范围。仅当 WVTP 是噪声时，才能设置。
MEAN	<mean>	:=噪声的均值。单位是伏特“V”。可在数据手册查阅参数值的有效范围。仅当 WVTP 是噪声时，才能设置该参数。

WIDTH	<width>	:=正脉宽。单位是秒“s”。可在数据手册查阅参数值的有效范围。仅当 WVTP 是脉冲波时，才能设置该参数。
RISE	<rise>	:=上升时间（10%~90%)。单位是秒“s”。可在数据手册查阅参数值的有效范围。仅当 WVTP 是脉冲波时，才能设置该参数。
FALL	<fall>	:=下降时间（10%~90%)。单位是秒“s”。可在数据手册查阅参数值的有效范围。仅当 WVTP 是脉冲波时，才能设置该参数。
DLY	<delay>	:=波形延迟。则可在数据手册查阅参数值的有效范围
HLEV	<high level>	:=高电平。单位是伏特“V”。当 WVTP 为噪声，该值无效。
LLEV	<low level>	:=低电平。单位是伏特“V”。当 WVTP 为噪声，该值无效。
BANDSTATE	<bandwidth switch>	:={ON（打开)，OFF（关闭）}。当 WVTP 为噪声，该参数才能设置
BANDWIDTH	<bandwidth value>	:=噪声带宽。单位为赫兹“Hz”。可在数据手册查阅参数值的有效范围。仅当 WVTP 是噪声时，才能设置该参数。
LENGTH	<prbs length>	:={3~32}。PRBS 实际长度=$2^{长度}-1$。仅当 WVTP 是 PRBS 时，才能设置该参数。
EDGE	<prbs rise/fall>	:= PRBS 的上升/下降时间。单位是秒“s”。可在数据手册查阅参数值的有效范围。仅当 WVTP 是 PRBS 时，才能设置该参数。
DIFFSTATE	<prbs differential switch>	:={ON（打开)，OFF（关闭）}。PRBS 差分输出模式。仅当 WVTP 是 PRBS 时，才能设置该参数。
BITRATE	<prbs bit rate>	:= PRBS 比特率。单位是位每秒“bps”。可在数据手册查阅参数值的有效范围。仅当 WVTP 是 PRBS 时，才能设置该参数。
LOGICLEVEL	<prbslogiclevel rate>	:= {TTL_CMOS，LVTTL_LVCMOS，ECL，LVPECL，LVDS，CUSTOM}。用于设置 PRBS 的逻辑电平，仅当 WVTP 是 PRBS 时，才能设置该参数。
AMPVRMS	<amplitude>	:=幅度。设置幅度单位为 Vrm。
AMPDBM	<amplitude>	:=幅度。设置幅度单位为 dBm。
MAX_OUTPUT_AMP	<amplitude>	:=通道最大幅度限制，可在数据手册查阅参数值的有效范围

查询语法	<通道>：BaSic_WaVe? <通道>：={C1, C2}.
响应格式	<通道>：BSWV<参数> <参数>：={当前波形的所有参数}.
示例	改变 C1 波形为三角波： *C1:BSWV WVTP,RAMP* 改变 C1 频率为 2000Hz： *C1:BSWV FRQ,2000* 设置 C1 的幅度为 3Vpp： *C1:BSWV AMP,3* 从设备读取 C1 的参数： *C1:BSWV?* 返回值： *C1:BSWV WVTP,SINE,FRQ,2000HZ,PERI,0.01S,AMP,3V,* *OFST,0V,HLEV,1V,LLEV,-1V,PHSE,0* 设置 C1 的噪声带宽为 100MHz： *C1:BSWV BANDWIDTH,100E6* 或 *C1:BSWV BANDWIDTH,100000000*

3.5 PRBS 多项式命令

描述	该命令用于设置或者获取 PRBS 自定义多项式参数。仅当基本波形为 PRBS 时，该命令有效。
命令语法	<通道>:PRBS:<参数> <值> <通道>:= {C1, C2}. <值>:= {相关参数的值}。

参数	值	描述
STAT	<STATE>	:={RUN,STOP} 打开或关闭 PRBS 播放状态。
TYPE	<type>	:={FORMULA,LENGTH}，设置多项式类型为默认或者自定义。
ITEM:Add	<value>	:= {1，2，...，32}，设置 PRBS 添加项。
ITEM:Delet	<value>	:= {1，2，...，32}，设置 PRBS 删除项。
ITEM:Clear	None	:=清空 PRBS 多项式。
SEED	<value>	:= {1，2，...，M}，设置 PRBS 种子数。

查询语法	<通道>:PRBS:<参数>? <通道>:={C1，C2}.
示例	启用 CH1 的 PRBS 播放状态： *C1:PRBS:STAT RUN* 设置 CH1 的 PRBS 多项式类型为自定义： *C1:PRBS:TYPE FORMULA* 获取 CH1 的 PRBS 多项式类型： *C1:PRBS:TYPE?* 返回值： *FORMULA*

3.6 DC 末级放大器命令

描述	此命令设置或者获取末级放大器状态。
命令语法	<通道>:DcUFAmplifier <状态> <通道>:= {C1，C2}. <状态>:= {ON，OFF}.
查询语法	<通道>:DcUFAmplifier? <通道>:= {C1，C2}.
响应格式	<通道>:DcUFAmplifier,<状态>
示例	打开 CH1 的末级放大器： *C1:DcUFAmplifier ON* 查询 CH2 的末级放大器状态： *C2:DcUFAmplifier?* 返回值： *C2:DcUFAmplifier,OFF*

3.7 谐波命令

描述	该命令用于设置或者获取谐波参数。仅当基本波形为正弦波时，该命令有效。
命令语法	<通道>:HARMonic <参数>,<值> <通道>: ={C1, C2}. <值>: ={相关参数的值}。

参数	值	描述
HARMSTATE	<STATE>	:={ON,OFF}打开或关闭谐波
HARMTYPE	<TYPE>	:={ EVEN（偶级数），ODD（奇级数），ALL（所有级数）} 谐波类型。
HARMORDER	<NUM>	: {1，2，...，M}，其中 M 是支持的最大级数。
HARMPHASE	<PHASE>	:={0~360}。单位是°。
HARMAMP	<VALUE>	:= 指定谐波的幅值。值的范围取决于机型。单位是伏特，峰峰值“Vpp”。
HARMDBC	<VALUE>	:= 指定谐波的幅值。值的范围取决于机型。单位是"dBc"。

查询语法	<通道>:HARMonic? <通道>:={C1，C2}.
示例	启用 CH1 的谐波功能： *C1:HARM HARMSTATE,ON* 设置 CH1 的谐波级数为 2，幅度为-6dBc： *C1:HARM HARMORDER,2,HARMDBC,-6* 获取 CH1 的谐波参数： *C1:HARM?* 返回值： *C1:HARM ,HARMSTATE,ON,HARMTYPE,EVEN,HARMORDER,2,HARMAMP,2.004748935V,HARMDBC,-6dBc,HARMPHASE,0*

3.8 双脉冲命令

描述	该命令用于设置或者获取双脉冲参数。
命令语法	格式 <通道>:MultiPulse:<参数> <值> <通道>:={C1, C2}. <值>:= {见表格中参数的值}

参数	**值**	**描述**
Count	<count>	:= 脉冲个数。可在数据手册查阅参数值的有效范围。
Source	<source>	:={ INT, EXT, MAN, TIMER}，触发源。INT 表示内部触发源，EXT 表示外部触发源，MAN 表示手动触发，TIMER 表示定时器触发。
TIMer	<timer>	:= 设置定时时间，当触发源为定时器时有效。
Dlay	<delay>	:= 设置触发延时，当触发源为手动或外部时有效。
Edge	<edge>	:={UP，DOWN} 设置外部触发沿，当触发源为外部时有效。
Item#:PosWidth	<width>	:= 设置脉冲的正脉宽，#表示脉冲的编号。
Item#:NegWidth	<width>	:= 设置脉冲的负脉宽，#表示脉冲的编号。
Trigger	None	:= 设置手动触发，当触发源为手动时生效。

查询语法	格式： <通道>:MultiPulse <参数>?
示例	设置通道 1 的脉冲数为 3： *C1:MultiPulse:Count 3* 获取通道 1 的脉冲个数： *C1:MultiPulse:Count?* 返回： *3* 设置通道 1 脉冲 1 的正脉宽为 20μs： *C1:MultiPulse:Item1:PosWidth 0.00002* 获取通道 1 脉冲 1 的正脉宽： *C1:MultiPulse:Item1:PosWidth?* 返回： *2e-05*

3.9 任意波形命令

3.9.1 任意波形切换波形命令

描述	该命令用于设置或者读取任意波的类型备注：命令语法和查询的响应格式中的索引号省略字符‘M’，直接使用数值代表索引号。
命令语法	格式 1：<通道>:ArbWaVe INDEX，<索引号> 格式 2：<通道>:ArbWaVe NAME，<名称> 格式 3：<通道>:ArbWaVe NAME，<路径> <通道>：={C1，C2}. <索引号>：下表中任意波的索引号 <名称>：下表的任意波名称 <路径>：波形路径，路径不需要加 Local
查询语法	<通道>：ARbWaVe? <通道>：={C1，C2}.
响应格式	<通道>：ARWV INDEX，<索引号>，NAME，<名称>
示例	通过索引号 2 设置 CH1 的当前波形： *C1:ARWV INDEX,2* 读取 CH1 的当前波形： *C1:ARWV?* 返回值： *C1:ARWV INDEX,2,NAME,StairUp* 通过波形名称设置 CH1 的波形为心波： *C1:ARWV NAME,Cardiac* 通过波形路径设置 CH1 的波形： *C1:ARWV NAME,“wave1.bin”*
注	具体路径参照文件管理器中路径
相关命令	STL

索引号	名称	索引号	名称	索引号	名称	索引号	名称
0	Sine	51	AttALT	102	LFPulse	153	Duty18
1	Noise	52	RoundHalf	103	Tens1	154	Duty20
2	StairUp	53	RoundsPM	104	Tens2	155	Duty22

3	StairDn	54	BlaseiWave	105	Tens3	156	Duty24
4	Stairud	55	DampedOsc	106	Airy	157	Duty26
5	Ppulse	56	SwingOsc	107	Besselj	158	Duty28
6	Npulse	57	Discharge	108	Bessely	159	Duty30
7	Trapezia	58	Pahcur	109	Dirichlet	160	Duty32
8	Upramp	59	Combin	110	Erf	161	Duty34
9	Dnramp	60	SCR	111	Erfc	162	Duty36
10	ExpFal	61	Butterworth	112	ErfcInv	163	Duty38
11	ExpRise	62	Chebyshev1	113	ErfInv	164	Duty40
12	Logfall	63	Chebyshev2	114	Laguerre	165	Duty42
13	Logrise	64	TV	115	Legend	166	Duty44
14	Sqrt	65	Voice	116	Versiera	167	Duty46
15	Root3	66	Surge	117	Weibull	168	Duty48
16	X^2	67	Radar	118	LogNormal	169	Duty50
17	X^3	68	Ripple	119	Laplace	170	Duty52
18	Sinc	69	Gamma	120	Maxwell	171	Duty54
19	Gaussian	70	StepResp	121	Rayleigh	172	Duty56
20	Dlorentz	71	BandLimited	122	Cauchy	173	Duty58
21	Haversine	72	CPulse	123	CosH	174	Duty60
22	Lorentz	73	CWPulse	124	CosInt	175	Duty62
23	Gauspuls	74	GateVibr	125	CotH	176	Duty64
24	Gmonopuls	75	LFMPulse	126	CscH	177	Duty66
25	Tripuls	76	MCNoise	127	SecH	178	Duty68
26	Cardiac	77	AM	128	SinH	179	Duty70
27	Quake	78	FM	129	SinInt	180	Duty72
28	Chirp	79	PFM	130	TanH	181	Duty74
29	Twotone	80	PM	131	ACosH	182	Duty76
30	SNR	81	PWM	132	ASecH	183	Duty78
31	Hamming	82	EOG	133	ASinH	184	Duty80
32	Hanning	83	EEG	134	ATanH	185	Duty82
33	Kaiser	84	EMG	135	ACsch	186	Duty84

34	Blackman	85	Pulseilogram	136	ACoth	187	Duty86
35	Gausswin	86	ResSpeed	137	Bartlett	188	Duty88
36	Triang	87	ECG1	138	BohmanWin	189	Duty90
37	BlackmanH	88	ECG2	139	ChebWin	190	Duty92
38	Bartlett-Hann	89	ECG3	140	FlattopWin	191	Duty94
39	Tan	90	ECG4	141	ParzenWin	192	Duty96
40	Cot	91	ECG5	142	TaylorWin	193	Duty98
41	Sec	92	ECG6	143	TukeyWin	194	Duty99
42	Csc	93	ECG7	144	Duty01	195	demo1_375
43	Asin	94	ECG8	145	Duty02	196	demo1_16k
44	Acos	95	ECG9	146	Duty04	197	demo2_3k
45	Atan	96	ECG10	147	Duty06	198	demo2_16k
46	Acot	97	ECG11	148	Duty08		
47	Square	98	ECG12	149	Duty10		
48	SineTra	99	ECG13	150	Duty12		
49	SineVer	100	ECG14	151	Duty14		
50	AmpALT	101	ECG15	152	Duty16		

3.9.2　任意波数据命令

描述	此命令用于设置或者获取任意波形数据。
命令语法	<通道>：WVDT，<参数>，<值> <通道>：={C1，C2} <参数>：={下表的参数} <值>：={相关参数的值}

参数	值	描述
WVNM	<波形名>	:= 波形名称
LENGTH	<长度>	:= 波形的字节数，有效范围取决于机型。详细信息查阅备注 1。“X”系列无需该参数。
FREQ	<频率>	:= 频率。单位为赫兹“Hz”。
AMPL	<幅值>	:= 幅值。单位是伏特，峰峰值“Vpp”。
OFST	<偏置>	:= 偏置。单位是伏特“V”。
PHASE	<相位>	:= 相位。单位是“度”。
WAVEDATA	<波形数据>	:= 波形数据。写入到波形中的数据。

查询语法	格式 1：WVDT? Mn 格式 2：WVDT? USER，<波形名称> 格式 3：WVDT? USER，<路径>，<波形名称> <路径>指定存储路径。 <波形名称>：={用户定义的波形名称}。
示例	将数据 0x6000c0006000 写到‘wave1’波形中并发送到通道 1 *C1:WVDT WVNM,wave1,WAVEDATA, b'0x6000c0006000'*

3.10 调制波形命令

描述	该命令可以设置或者获取调制波形参数
命令语法	<通道>：MoDulateWaVe <类型> <通道>：MoDulateWaVe<参数>，<值> <通道>：={C1，C2} <类型>：={AM，DSBAM，FM，PM，PWM，ASK，FSK，PSK} <参数>：={下表的值} <值>：={相关参数的值}

参数	**值**	**描述**
RSTAT	<state>	:={RUN，STOP}。设置 MOD 播放状态。具体用法请参考示例。
STATE	<state>	:={ON，OFF}。启用和禁用调制。如果你想设置或者读取调制的其他参数，必须先将 STATE 设置为“ON”。
AM，SRC	<src>	:={INT，EXT，CH1,CH2}。AM 调制源，其中‘CH1’只有在设置 CH2 调制源的时候才生效，‘CH2’只有在设置 CH1 调制源的时候才生效。
AM，MDSP	<mod wave shape>	:= {SINE，SQUARE，TRIANGLE，UPRAMP，DNRAMP，NOISE，ARB}. AM 调制波形。仅当调制源设置为内调制时，该参数才能设置。
AM，FRQ	<AM frequency>	:= AM 频率。单位是赫兹“Hz”，可在数据手册查阅参数值的有效范围。调制源设置为内调制时，该参数才能设置。
AM，DEPTH	<depth>	:={0 至 120}。AM 深度。单位是“%”。触发源设置为内触发时，该参数才能设置。
DSBAM，SRC	<src>	:= {INT，EXT，CH1, CH2}。DSBAM 调制源，其中‘CH1’只有在设置 CH2 调制源的时候才生效，‘CH2’只有在设置 CH1 调制源的时候才生效。
DSBAM，MDSP	<mod wave shape>	:= {SINE，SQUARE，TRIANGLE，UPRAMP，DNRAMP，NOISE，ARB}。 DSB AM 调制波形。调制源设置为内调制时，该参数才能设置。
DSBAM，FRQ	<DSB-AM frequency>	:= DSB AM 频率。单位是赫兹“Hz”，可在数据手册查阅参数值的有效范围。设置为内调制时，该参数才能设置。
FM，SRC	<src>	:={INT，EXT，CH1, CH2}。FM 调制源，其中‘CH1’

		只有在设置 CH2 调制源的时候才生效，‘CH2’只有在设置 CH1 调制源的时候才生效。
FM，MDSP	<mod wave shape>	:= {SINE， SQUARE， TRIANGLE， UPRAMP，DNRAMP，NOISE，ARB}. FM 调制波形。调制源设置为内调制时，该参数才能设置。
FM，FRQ	<FM frequency>	:= FM 频率。单位是赫兹“Hz”，可在数据手册查阅参数值的有效范围。调制源设置为内调制时，该参数才能设置。
FM，DEVI	<FM frequency deviation>	:= {0 至当前载波频率}。 FM 频率偏差。该值取决于载波频率和带宽频率的差值。调制源设置为内调制时，该参数才能设置。
PM，SRC	<src>	:= {INT，EXT，CH1, CH2}。PM 调制源，其中‘CH1’只有在设置 CH2 调制源的时候才生效，‘CH2’只有在设置 CH1 调制源的时候才生效。
PM，MDSP	<mod wave shape>	:= {SINE， SQUARE， TRIANGLE， UPRAMP，DNRAMP，NOISE，ARB}。 PM 调制波形。调制源设置为内调制时，该参数才能设置。
PM，FRQ	<PM frequency>	:= PM 频率。单位是赫兹“Hz”，可在数据手册查阅参数值的有效范围。调制源设置为内调制时，该参数才能设置。
PM，DEVI	<PM phase offset>	:= {0 至 360}。PM 相位偏差。单位是“°”， 调制源设置为内调制时，该参数才能设置。
PWM，SRC	<src>	:= {INT，EXT，CH1, CH2}。PWM 调制源，其中‘CH1’只有在设置 CH2 调制源的时候才生效，‘CH2’只有在设置 CH1 调制源的时候才生效。
PWM，FRQ	<PWM frequency>	:= PWM 频率。单位是赫兹“Hz”，可在数据手册查阅参数值的有效范围。调制源设置为内调制时，该参数才能设置。
PWM，DEVI	<PWM dev>	:=占空比偏差。单位是“s”。值取决于载波的脉宽。
PWM，MDSP	<mod wave shape>	:= {SINE， SQUARE， TRIANGLE， UPRAMP，DNRAMP，NOISE，ARB}。 PWM 调制波形。调制源设置为内调制时，该参数才能设置。
ASK，SRC	<src>	:= {INT（内调制），EXT（外调制）}。ASK 调制源。
ASK，KFRQ	< key frequency>	:= ASK 键控频率。单位是赫兹“Hz”，可在数据手册查阅参数值的有效范围。调制源设置为内调制时，该参数才能设置。

FSK，SRC	<src>	:= {INT（内调制），EXT（外调制）}。FSK 调制源。
FSK，KFRQ	< key frequency>	:= FSK 键控频率。单位是赫兹“Hz”，可在数据手册查阅参数值的有效范围。调制源设置为内调制时，该参数才能设置。
FSK，HFRQ	<FSK_hop_freq>	:= FSK 跳频频率。与基础波形频率相同。单位是赫兹“Hz”，可在数据手册查阅参数值的有效范围。
PSK，SRC	<src>	:= {INT, EXT}。PSK 调制源。
PSK，KFRQ	< key frequency>	:= PSK 键控频率。单位是赫兹“Hz”，可在数据手册查阅参数值的有效范围。调制源设置为内调制时，该参数才能设置。
PSK，PLRT	<polarity>	:= {POS,NEG}.
MDSP,ARB,INDEX	<index>	:= {0 至 198}，设置调制波形为内建波形，当调制波形为 Arb 时可设置。
MDSP,ARB,NAME	<name>	:= 波形名称，设置调制波形为已存波形，当调制波形为 Arb 时可设置。
CARR，WVTP	<wave type>	:={SINE，SQUARE，RAMP，ARB，PULSE}。载波频率类型。
CARR，FRQ	<frequency>	:= 载波频率。单位是赫兹“Hz”，可在数据手册查阅参数值的有效范围。
CARR，PHSE	<phase>	:= {0 至 360}。载波相位。单位为“度”。
CARR，AMP	<amplitude>	:=载波幅值。单位是伏特，峰峰值"Vpp"。可在数据手册查阅参数值的有效范围。
CARR，OFST	<offset>	:=载波偏置。单位是伏特。可在数据手册查阅参数值的有效范围。
CARR，SYM	<symmetry>	:= {0 至 100}。当载波为 RAMP 时，载波对称度。单位是“%”。
CARR，DUTY	<duty>	:= {0 至 100}。当载波为方波或者脉冲波时，载波占空比。单位是“%”。
CARR，RISE	<rise>	:=当载波为方波或者脉冲波时，载波上升时间。单位是秒“s”。可在数据手册查阅参数值的有效范围。
CARR，FALL	<fall>	:=当载波为方波或者脉冲波时，载波下降时间。单位是秒“s”。可在数据手册查阅参数值的有效范围。
CARR，DLY	<delay>	:=当载波为方波或者脉冲波时，载波延时。单位是秒“s”。可在数据手册查阅参数值的有效范围。

备注：

部分参数范围取决于机型。详细信息请查阅各机型的数据手册。

查询语法	<通道>：MoDulateWaVe?

	<通道>：＝{C1，C2}.
响应格式	<通道>：MDWV<参数> <参数>：＝{当前调制的所有参数}.
示例	设置 CH1 调制状态为打开： *C1:MDWV STATE,ON* 设置 CH1 播放状态为运行： *C1:MDWV:RSTAT RUN* 设置 CH1 调制类型为 AM： *C1:MDWV AM* 设置 AM 调制，且调制波形设为正弦波： *C1:MDWV AM,MDSP,SINE* 读取状态值为 ON 的 C1 的调制参数： *C1:MDWV?* 返回值： *C1:MDWVAM,STATE,ON,MDSP,SINE,SRC,INT,FRQ,100HZ,* *DEPTH,100,CARR,WVTP,RAMP,FRQ,1000HZ,AMP,4V,AMPVRMS,* *1.15473Vrms,OFST,0V,PHSE,0,SYM,50* 读取状态为 OFF 的 CH1 的调制参数： *C1:MDWV?* 返回值： *C1:MDWV STATE,OFF* 设置 CH1 的 FM 频率为 1000Hz： *C1:MDWV FM,FRQ,1000* 设置 CH1 的载波为正弦波： *C1:MDWV CARR,WVTP,SINE* 设置 CH1 载波频率为 1000Hz： *C1:MDWV CARR,FRQ,1000*

3.11 扫描波形命令

3.11.1 <channel>:SweepWaVe <para>,<value>

描述	该命令用于设置或者获取扫描波形的参数。
命令语法	< channel >：SweepWaVe <para>，< value > < channel >：＝{C1, C2} <param>：＝{下表的参数} < value >：＝{相关的值}

参数	**值**	**描述**
RSTAT	<state>	:={RUN，STOP}。设置 Sweep 播放状态。具体用法请参考示例。
STATE	<state>	:={ON，OFF}。启用和禁用扫描。 如果你想设置或者读取扫描的其他参数，必须先将 STATE 设置为“ON”。
TIME	<time>	:= 扫描时间。单位是秒“s”。可在数据手册查阅参数值的有效范围。
START	<start_freq>	:= 开始频率。与基本波形频率相同。单位是赫兹“Hz”。可在数据手册查阅参数值的有效范围。
STOP	<stop_freq>	:= 终止频率。与基本波形频率相同。单位是赫兹“Hz”。可在数据手册查阅参数值的有效范围。
CENTER	<center_freq>	:= 中心频率。单位是赫兹“Hz”。可在数据手册查阅参数值的有效范围。
SPAN	<span_freq>	:= 频率范围。单位是赫兹“Hz”。可在数据手册查阅参数值的有效范围。
SWMD	<sweep_mode>	:={LINE，LOG，STEP}，其中 LINE 代表线性扫频，LOG 代表对数扫频，STEP 代表步进扫频。
STEPNUM	<value>	:={2 至 1024}，步进数，当扫描类型为步进时有效。
DIR	<direction>	:={UP（向上），DOWN（向下）UP_DOWN（上下）}。扫描方向
SYM	<symmetry>	:={0% 至 100%}。当扫描方向为 UP_DOWN（上下）时的对称度。
TRSR	<trig_src>	:={EXT，INT，MAN}。触发源。EXT 代表外触发，INT 代表内触发，MAN 代表手动触发。
MTRIG		:= 发送一个手动触发信号。仅当 TRSR 是 MAN 时，该参数有效。
TRMD	<trig_mode>	:={ON，OFF}。触发输出状态。若 TRSR 为 EXT，该参数无效。

EDGE	<edge>	:={RISE，FALL}。可用触发沿。仅当 TRSR 设为 EXT 时有效。
MARK_STATE	<state>	:={ON，OFF}。设置频率标记状态打开或关闭。
MARK_FREQ	<value>	:= 标记频率，当扫描类型为线性或对数时有效，标记范围从起始频率至终止频率。
MARK_STEP	<value>	:={2 至 1024}，标记步进，当扫描类型为步进时有效。
IDLE_FREQ	<idle freq>	:={ZERO_FREQ,START_FREQ,STOP_FREQ}。空闲频率。ZERO_FREQ 表示零电平，START_FREQ 表示起始频率，STOP_FREQ 表示终止频率。仅当 TRSR 设为 EXT 或 MAN 时有效。
OUTPUT_MODE	<output mode>	:={ SINGLE, CONTINU}。输出模式。SINGLE 表示输出模式为单次扫描，CONTINU 表示输出模式为连续扫描。仅当 TRSR 设为 EXT 或 MAN 时有效。
CURFREQ	None	:= 该参数用于查询当前输出的频率,具体用法请参考示例。
CARR, WVTP	<wave type>	:={SINE，SQUARE，RAMP，ARB}。载波类型。若载波为脉冲波、噪声、直流。则无法扫描波形。
CARR, FRQ	<frequency>	:= 载波频率。单位是赫兹“Hz”，可在数据手册查阅参数值的有效范围。
CARR, PHSE	<phase>	:={0 至 360}。载波相位。单位是“度”
CARR, AMP	<amplitude>	:= 载波幅值。单位是伏特，峰峰值 "Vpp" 。可在数据手册查阅参数值的有效范围。
CARR, OFST	<offset>	:= 载波偏置。单位是伏特“V”。可在数据手册查阅参数值的有效范围。
CARR, SYM	<symmetry>	:={0 至 100}。当载波为 RAMP 时，载波对称度。单位是“%”。
CARR, DUTY	<duty>	:={0 至 100}。当载波为方波时，载波占空比。单位是“%”。

查询语法 < channel >:SWeepWaVe?

< channel >:= {C1，C2}.

响应格式 < channel >:SWWV <param>

< channel >:= {当前扫描波形的所有参数}

示例 设置 CH1 的扫描状态为打开：

C1:SWWV STATE,ON

设置 CH1 播放状态为运行：

C1:SWEep:RSTAT RUN

设置 CH1 的扫描时间为 1 秒：

	C1:SWWV TIME,1 设置 CH1 的终止频率为 1000Hz： C1:SWWV STOP,1000 设置 CH1 的触发源为手动： *C1:SWWV TRSR,MAN* 发送一个手动触发信号至 CH1 *C1:SWWV MTRIG* 读取状态为 ON 的 CH2 的扫描参数： *C2:SWWV?* 返回值： *C2:SWWVSTATE,ON,TIME,1S,STOP,1500HZ,START,500HZ,* *CENTER,1000HZ,SPAN,1000HZ,TRSR,INT,TRMD,OFF,SWMD,* *LINE,DIR,UP,SYM,0,MARK_STATE,OFF,MARK_FREQ,1000HZ,* *CARR,WVTP,SINE,FRQ,1000HZ,AMP,4V,AMPVRMS,1.41421Vrms,* *OFST,0V,PHSE,0* 读取 CH1 的输出频率： *C1:SWEep:CURFREQ?* 返回值： *505.05253904968*

3.11.2 <channel>:SWEep:TYPE <type>

描述	该命令用于设置（查询）扫描波形的类型。
命令语法	<channel>:SWEep:TYPE <type> <channel>:= {C1，C2}. <type>:= {FREQ，AMP }.
查询语法	<channel>: SWEep:TYPE? <channel>:= {C1，C2}.
示例	设置通道 1 的扫描类型为频率扫描 *:C1:SWEep:TYPE FREQ* 查询通道 1 的扫描类型 *:C1:SWEep:TYPE?* 返回值： *"FREQ"*

3.11.3 <channel>:SWEep:SAMPlitude <value>

描述	该命令用于设置（查询）幅度扫描的起始幅度。
命令语法	<channel>:SWEep:SAMPlitude <value> <channel>:= {C1，C2}. <value>:= 浮点类型的值，单位伏特（V）
查询语法	<channel>: SWEep:SAMPlitude? <channel>:= {C1，C2}.
示例	设置通道 1 幅度扫描的起始幅度为 100mV *:C1:SWEep:SAMPlitude 0.1* 查询通道 1 幅度扫描的起始幅度 *:C1:SWEep:SAMPlitude?* 返回值： *“0.1”*

3.11.4 <channel>:SWEep:EAMPlitude <value>

描述	该命令用于设置（查询）幅度扫描的终止幅度。
命令语法	<channel>:SWEep:EAMPlitude <value> <channel>:= {C1，C2}. <value>:= 浮点类型的值，单位伏特（V）
查询语法	<channel>: SWEep:EAMPlitude? <channel>:= {C1，C2}.
示例	设置通道 1 幅度扫描的终止幅度为 900mV *:C1:SWEep:EAMPlitude 0.9* 查询通道 1 幅度扫描的终止幅度 *:C1:SWEep:EAMPlitude?* 返回值： *“0.9”*

3.11.5 <channel>:SWEep:CAMPlitude <value>

描述	该命令用于设置（查询）幅度扫描的中心幅度。
命令语法	<channel>:SWEep:CAMPlitude <value> <channel>:= {C1，C2}. <value>:= 浮点类型的值，单位伏特（V）

查询语法	<channel>: SWEep:CAMPlitude? <channel>:= {C1，C2}.
示例	设置通道 1 幅度扫描的中心幅度为 500mV *:C1:SWEep:CAMPlitude 0.5* 查询通道 1 幅度扫描的中心幅度 *:C1:SWEep:CAMPlitude?* 返回值： *“0.5”*

3.11.6 <channel>:SWEep:ASPan <value>

描述	该命令用于设置（查询）幅度扫描的幅度范围。
命令语法	<channel>:SWEep:ASPan <value> <channel>:= {C1，C2}. <value>:= 浮点类型的值，单位伏特（V）
查询语法	<channel>: SWEep: ASPan? <channel>:= {C1，C2}.
示例	设置通道 1 幅度扫描的幅度范围为 800mV *:C1:SWEep:ASPan 0.8* 查询通道 1 幅度扫描的幅度范围 *:C1:SWEep: ASPan?* 返回值： *“0.8”*

3.12 脉冲串命令

描述	此命令用于设置或者读取脉冲串波形参数。
命令语法	<通道>：BursTWaVe <参数>，<值> <通道>：={C1，C2}。 <参数>：={下表的参数}。 <值>：={相关参数的值}。

参数	值	描述
RSTAT	<state>	:={RUN，STOP}。设置 Burst 播放状态。具体用法请参考示例。
STATE	<state>	:= {ON，OFF}。启用和禁用脉冲串。 如果你想设置或者读取脉冲串的其他参数，必须先将 STATE 设置为"ON"。
PRD	<period>	:=脉冲串周期。单位是秒"s"。可在数据手册查阅参数值的有效范围。
STPS	<start_phase>	:={0 至 360}。载波的起始相位。单位是"°"。当载波为噪声或者脉冲波时，该值无效。
GATE_NCYC	<burst_mode>	:={GATE（门控模式），NCYC（N 循环模式）}。脉冲串模式。当载波为噪声时，该值无效。
TRSR	<trig_src>	:= {EXT，INT，MAN}触发源。EXT 代表外部触发。INT 代表内部触发、MAN 代表手动触发。
MTRIG		:=发送一个手动触发信号。仅当 TRSR 是 MAN 时，该参数有效。
DLAY	<delay>	:=触发延时。单位是秒"s"。可在数据手册查阅参数值的有效范围。当 GATE_NCYC 为 N 循环时，该值有效。当在载波为噪声时该值无效。
PLRT	<polarity>	:={NEG（负极性），POS（正极性）}。门控极性，负极性或者正极性。
TRMD	<trig_mode>	:={RISE（向上），FALL（向下），OFF（关闭）}。触发输出模式。当 GATE_NCYC 为 N 循环且触发源为内触发或者手动触发时，该参数有效。当载波为噪声时该值无效。
EDGE	<edge>	:={RISE，FALL}。有效触发边沿。当 TRST 为手动触发或者外触发时，该值有效。当载波为噪声时该值无效。
TIME	<circle_ time>	:={INF，1，2，…，M}，M 值支持的最大 N 循环数取决于机型。INF 设置脉冲串为无限模式。当 GATE_NCYC

		为 N 循环时有效。当载波为噪声时，该值无效。
HVALue	<hold value>	:= {ZERo，SVALue，EVALue}。空闲电平。ZERo 表示中间值，SVALue 表示起始值，EVALue 表示终止值。具体用法请参考示例。
COUNTER	<counter>	:=脉冲串数，当触发源为外部或手动时有效，可在数据手册查阅参数的有效值范围。
CARR, WVTP	<wave type>	:={SINE，SQUARE，RAMP，ARB，PULSE，NOISE}。载波波形类型
CARR, FRQ	<frequency>	:=载波频率。单位是赫兹“Hz”，可在数据手册查阅参数值的有效范围。
CARR, PHSE	<phase>	:= {0 至 360}。载波相位。单位是“度”。
CARR, AMP	<amplitude>	:=载波幅值。单位是伏特，峰峰值“Vpp”。可在数据手册查阅参数值的有效范围。
CARR, OFST	<offset>	:=载波偏置。单位是幅度“V”。可在数据手册查阅参数值的有效范围。
CARR, SYM	<symmetry>	:={0 至 100}。载波对称度，当载波为三角波时，该值有效。单位是“%”。
CARR, DUTY	<duty>	:= {0 至 100}。载波占空比，当载波为方波和脉冲波，该值有效。单位是“%”。
CARR, RISE	<rise>	:=上升时间。当载波为脉冲波时该值有效。单位是“s”。可在数据手册查阅参数值的有效范围。
CARR, FALL	<fall>	:=下降时间。当载波为脉冲波时该值有效。单位是“s”。可在数据手册查阅参数值的有效范围。
CARR, DLY	<delay>	:=脉冲波延时。当载波为脉冲波时该值有效。单位是“s”。可在数据手册查阅参数值的有效范围。
CARR, STDEV	<stdev>	:=噪声的标准差。单位是伏特“V”。可在数据手册查阅参数值的有效范围。
CARR, MEAN	<mean>	:=噪声的均值。单位是伏特“V”。可在数据手册查阅参数值的有效范围。

查询语法 <通道>: BTWV(BursTWaVe)?

<通道>: = {C1, C2}

响应格式 <通道>: BTWV <参数>

<参数>: = {当前脉冲串的所有参数}

示例 设置 CH1 脉冲串状态为 ON:

C1:BTWV STATE,ON

设置 CH1 播放状态为运行：

C1:BURSt:RSTAT RUN

设置 CH1 脉冲串周期为 1s：

C1:BTWV PRD,1

设置脉冲串延时为 1s：

C1:BTWV DLAY,1

设置 CH1 脉冲串为无限：

C1:BTWV TIME,INF

设置 CH1 空闲电平为终止值：

C1:BURSt:HVALue EVALue

读取状态为 ON 的 CH2 脉冲串参数：

C2:BTWV?

返回值：

C2:BTWV STATE,ON,PRD,0.01S,STPS,0,TRSR,INT,
TRMD,OFF,TIME,1,DLAY,2.4e-07S,GATE_NCYC,NCYC,
CARR,WVTP,SINE,FRQ,1000HZ,AMP,4V,OFST,0V,PHSE,0

读取状态为 OFF 的 CH2 脉冲串参数：

C2:BTWV?

返回值：

C2:BTWV STATE,OFF

3.13 序列波命令

3.13.1 <channel>:SEQuence <switch>

描述	该命令用于设置（查询）序列的输出状态
命令语法	<channel>:SEQuence <switch> <channel>:= {C1，C2}. <switch>:= {ON，OFF}.
查询语法	<channel>:SEQuence? <channel>:= {C1，C2}.
示例	开启通道 1 的序列输出 *:C1:SEQuence ON* 读取通道 1 的序列输出状态 *:C1:SEQuence?* 返回值： *"ON"*

3.13.2 <channel>:SEQuence:SRATe

描述	该命令用于设置（查询）序列的采样率
命令语法	<channel>:SEQuence:SRATe <采样率> <channel>:= {C1，C2}. <采样率> := 采样率。单位是 Sa/s。
查询语法	<channel>: SEQuence:SRATe? <channel>:= {C1，C2}.
示例	设置通道 1 的序列采样率 16385000 *:C1:SEQuence:SRATe 16385000* 读取通道 1 的序列采样率 *:C1:SEQuence:SRATe?* 返回值： *16385000*

3.13.3 <channel>:SEQuence:STATe

描述	该命令用于设置（查询）序列的运行状态
命令语法	<channel>:SEQuence:STATe <state> <channel>:= {C1，C2}. <state>:= {RUN，STOP}
查询语法	<channel>:SEQuence:STATe? <channel>:= {C1，C2}.
示例	设置通道 1 序列的状态为”RUN” *:C1:SEQuence:STATe RUN* 查询通道 1 序列的状态 *:C1:SEQuence:STATe?* 返回值： *“RUN”*

3.13.4 <channel>:SEQuence:SCALe

描述	该命令用于设置（查询）序列的输出幅度
命令语法	<channel>:SEQuence:SCALe <scale> <channel>:= {C1，C2}. <scale>:= {1 到 100}
查询语法	<channel>:SEQuence:SCALe? <channel>:= {C1，C2}.
示例	设置通道 1 序列的输出幅度为 80% *:C1:SEQuence:SCALe 80* 查询通道 1 序列的输出幅度 *:C1:SEQuence:SCALe?* 返回值： *“80.000000%”*

3.13.5 <channel>:SEQuence: OFFset

描述	该命令用于设置（查询）序列波形的偏置
命令语法	<channel>:SEQuence:OFFset <offset> <channel>:= {C1，C2}. <offset>:= {-8V 到 8V}
查询语法	<channel>:SEQuence:OFFset? <channel>:={C1，C2}.
示例	设置通道 1 序列第三段波形的偏置为 2Vdc *:C1:SEQuence:OFFset 2* 查询通道 1 序列第三段波形的偏置 *:C1:SEQuence:OFFset?* 返回值： *“2”*

3.13.6 <channel>:SEQuence:SAVe

描述	该命令用于保存序列波形到文件
命令语法	<channel>:SEQuence:SAVe<path> <channel>:= {C1，C2}. <path>:= 保存波形的完整路径
示例	保存序列波形到 sequence. awg *:C1:SEQuence:SAVe "sequence.awgx"*

3.13.7 <channel>:SEQuence:RECall

描述	该命令用于从文件加载序列波形
命令语法	<channel>:SEQuence:RECall <path> <channel>:= {C1，C2}. <path>:=波形的完整路径
示例	从 Local 下的文件 sequence. awg 加载序列波形 *:C1:SEQuence:RECall "sequence.awgx"*

3.13.8 <channel>:SEQuence:RMODe

描述	该命令用于设置（查询）序列的运行模式
命令语法	<channel>:SEQuence:RMODe <mode> <channel>:= {C1，C2}. <mode>:= {CONT，STEP}.
查询语法	<channel>:SEQuence:RMODe? <channel>:= {C1，C2}.
示例	设置通道 1 序列的运行模式为连续运行 *:C1:SEQuence:RMODe CONT* 查询通道 1 序列的运行模式 *:C1:SEQuence:RMODe?* 返回值： *“CONT”*

3.13.9 <channel>:SEQuence:STARTNumb

描述	该命令用于设置（查询）序列的起始段。
命令语法	<channel>:SEQuence:STARTNumb <num> <channel>:= {C1，C2}. < num >:=段编号
查询语法	<channel>:SEQuence:STARTNumb? <channel>:= {C1，C2}.
示例	设置通道 1 序列的起始段为 2 *:C1:SEQuence:STARTNumb 2* 查询通道 1 序列的起始段编号 *:C1:SEQuence:STARTNumb?* 返回值： *2*

3.13.10 <channel>:SEQuence:INTPo

描述	该命令用于设置（查询）序列的插值方式
命令语法	<channel>:SEQuence:INTPo<mode> <channel>:= {C1，C2}. <mode>:= {LINE，HOLD，SINC，SINC13，SINC27}.
查询语法	<channel>:SEQuence:INTPo? <channel>:= {C1，C2}.
示例	设置通道 1 序列的插值方式为线性 *:C1:SEQuence:INTPo LINE* 查询通道 1 序列的插值方式 *:C1:SEQuence:INTPo?* 返回值： *"LINE"*

3.13.11 <channel>: SEQuence:TRIGger:SOURce

描述	该命令用于设置（查询）序列运行时的触发方式
命令语法	<channel>:SEQuence:TRIGger:SOURce <src> <channel>:= {C1，C2}. <src>:= {MAN，EXT}.
查询语法	<channel>:SEQuence:TRIGger:SOURce? <channel>:= {C1，C2}.
示例	设置通道 1 序列的触发方式为外部触发 *:C1:SEQuence:TRIGger:SOURce EXT* 查询通道 1 序列的触发方式 *:C1:SEQuence:TRIGger:SOURce?* 返回值： *"EXT"*

3.13.12 <channel>: SEQuence:TRIGger:SLOPe

描述	该命令用于设置（查询）序列外部触发的触发沿
命令语法	<channel>:SEQuence:TRIGger:SLOPe <slope> <channel>:= {C1，C2 }. <slope>:= {RISe，FALL}.
查询语法	<channel>:SEQuence:TRIGger:SLOPe? <channel>:= {C1，C2}.
示例	设置通道 1 序列外部触发的触发极性为上升沿 *:C1:SEQuence:TRIGger:SLOPe RISe* 查询通道 1 序列外部触发的触发极性 *:C1:SEQuence:TRIGger:SLOPe?* 返回值： *"RISe"*

3.13.13 <channel>:SEQuence:TRIGger:TRIG

描述	该命令立即触发一次序列输出
命令语法	<channel>: SEQuence:TRIGger:TRIG <channel>:= {C1，C2}.
示例	设置立即触发通道 1 的序列输出 *C1:SEQuence:TRIGger:TRIG*

3.13.14 <channel>:SEQuence:TRIGger:HOLD

描述	该命令用于设置（查询）序列的空闲电平
命令语法	<channel>:SEQuence:TRIGger:HOLD <type> <channel>:= {C1，C2}. <type>:= {END，MID，START}
查询语法	<channel>:SEQuence:TRIGger:HOLD?
示例	设置通道 1 序列的空闲电平为终止值 *:C1:SEQuence:TRIGger:HOLD END* 查询通道 1 序列的空闲电平

	:C1:SEQuence:TRIGger:HOLD? 返回值： *END*

3.13.15 <channel>:SEQuence:TRIGger:Delay

描述	该命令用于设置（查询）序列的触发延时
命令语法	<channel>:SEQuence:TRIGger:Delay<time> <channel>:= {C1，C2}. <time>:= 触发延时，单位 s。
查询语法	<channel>:SEQuence:TRIGger:Delay?
示例	设置通道 1 序列的触发延时为 10us *:C1:SEQuence:TRIGger:Delay 0.00001* 查询通道 1 序列的触发延时 *:C1:SEQuence:TRIGger:Delay?* 返回值： *1e-05*

3.13.16 <channel>:SEQuence:TRIGger:Out

描述	该命令用于设置（查询）序列的触发输出
命令语法	<channel>:SEQuence:TRIGger:Out<type> <channel>:= {C1，C2}. <type>:= {UP,DOWN,OFF}。
查询语法	<channel>:SEQuence:TRIGger:Out?
示例	设置通道 1 序列的触发输出为 UP *:C1:SEQuence:TRIGger:Out UP* 查询通道 1 序列的触发输出 *:C1:SEQuence:TRIGger:Out?* 返回值： *UP*

3.13.17 <channel>:SEQuence:INCReasing

描述	该命令用于设置（查询）序列的插值方式
命令语法	<channel>:SEQuence:INCReasing <mode> <channel>:= {C1，C2}. <mode>:= {INT，ZERo，HLAS，DUPL}
查询语法	<channel>:SEQuence:INCReasing?
示例	设置通道 1 序列的插值方式为线性插值 *:C1:SEQuence:INCReasing INT* 查询通道 1 序列的插值方式 *:C1:SEQuence:INCReasing?* 返回值： *"INT"*

3.13.18 <channel>:SEQuence:DECReasing

描述	该命令用于设置（查询）序列的抽值方式
命令语法	<channel>:SEQuence:DECReasing <mode> <channel>:= {C1，C2}. <mode>:= {DECi，CTAi，CHEa}
查询语法	<channel>:SEQuence:DECReasing?
示例	设置通道 1 序列的抽值方式为线性抽值 *:C1:SEQuence:DECReasing DECi* 查询通道 1 序列的抽值方式 *:C1:SEQuence:DECReasing?* 返回值： *"DECi"*

3.13.19 <channel>:SEQuence:SEGMent:Add

描述	该命令用于新增一段波形
命令语法	<channel>:SEQuence:SEGMent:Add <channel>:= {C1，C2}.
示例	通道 1 的序列 1 段波形 *C1:SEQuence:SEGMent:Add*

3.13.20 <channel>:SEQuence:SEGMent<x>:INSErt

描述	该命令用于在序列波形中的某一段之后插入一个新的段
命令语法	<channel>:SEQuence:SEGMent<x>:INSErt <channel>:= {C1，C2}. <x>:= 段编号
示例	在通道 1 序列的第三段之后插入一个新的段 *:C1:SEQuence:SEGMent3:INSErt*

3.13.21 <channel>:SEQuence:SEGMent<x>:DELEte

描述	该命令用于删除序列波形中的某一段
命令语法	<channel>:SEQuence:SEGMent<x>:DELEte <channel>:= {C1，C2}. <x>:= 段编号
示例	删除通道 1 序列中的第三段波形 *:C1:SEQuence:SEGMent3:DELEte*

3.13.22 <channel>:SEQuence:SEGMent:Clear

描述	该命令用于序列清空段
命令语法	<channel>:SEQuence:SEGMent:Clear <channel>:= {C1，C2}.
示例	清空通道 1 序列中的波形 *:C1:SEQuence:SEGMent:Clear*

3.13.23 <channel>:SEQuence:SEGMent<x>:GOTO

描述	该命令用于设置（查询）序列段的跳转
命令语法	<channel>:SEQuence:SEGMent<x>:GOTO <num> <channel>:= {C1，C2}. <x>:= 段编号 <num>:= 跳转的段编号
查询语法	<channel>:SEQuence:SEGMent<x>:GOTO? <channel>:= {C1，C2}. <x>:= 段编号
示例	设置通道 1 序列段 1 跳转到段 3 *:C1:SEQuence:SEGMent1:GOTO 3* 查询通道 1 序列第三段波形的长度 *:C1:SEQuence:SEGMent1:GOTO?* 返回值： *“3”*

3.13.24 <channel>:SEQuence:SEGMent<x>:LENGth

描述	该命令用于设置（查询）序列某一段波形的长度
命令语法	<channel>:SEQuence:SEGMent<x>:LENGth <len> <channel>:= {C1，C2}. <x>:= 段编号 <len>:= 会自动截断设置到 16 的整数倍
查询语法	<channel>:SEQuence:SEGMent<x>:LENGth?
示例	设置通道 1 序列第三段波形的长度为 16384 *:C1:SEQuence:SEGMent3:LENGth 16384* 查询通道 1 序列第三段波形的长度 *:C1:SEQuence:SEGMent3:LENGth?* 返回值： *“16384”*

3.13.25 <channel>:SEQuence:SEGMent<x>:REPeat:COUNt

描述	该命令用于设置（查询）序列某一段波形的重复次数
命令语法	<channel>:SEQuence:SEGMent<x>:REPeat:COUNt <count> <channel>:= {C1，C2}. <x>:= 段编号 <count>:= 最大值与本段波形长度，以及其他段波形的总长度相关。
查询语法	<channel>:SEQuence:SEGMent<x>:REPeat:COUNt?
示例	设置通道 1 序列的第三段波形重复 2 次. *:C1:SEQuence:SEGMent3:REPeat:COUNt 2* 查询通道 1 序列的第三段波形重复次数 *:C1:SEQuence:SEGMent3:REPeat:COUNt?* 返回值: *“2”*

3.13.26 <channel>:SEQuence:SEGMent<x>:WAVeform

描述	该命令用于通过波形名字设置（查询）序列某一段的波形
命令语法	<channel>:SEQuence:SEGMent<x>:WAVeform <name> <channel>:= {C1，C2}. <x>:= 段编号 <name>:= 波形名字。可用波形名见 3.9.1 节
查询语法	<channel>:SEQuence:SEGMent<x>:WAVeform? <channel>:= {C1，C2}. <x>:= 段编号
示例	设置通道 1 序列第三段的波形为 stairup *:C1:SEQuence:SEGMent3:WAVeform stairup* 查询通道 1 序列的第三段波形（返回波形名字） *:C1:SEQuence:SEGMent3:WAVeform?* 返回值: *“stairup”*

3.13.27 <channel>:SEQuence:SEGMent<x>:AMPlitude

描述	该命令用于设置（查询）序列某一段波形的幅度
命令语法	<channel>:SEQuence:SEGMent<x>:AMPlitude <amp> <channel>:= {C1，C2}. <x>:=段编号 <amp>:= {0 到 20Vpp}
查询语法	<channel>:SEQuence:SEGMent<x>:AMPlitude?
示例	设置通道 1 序列第三段波形的幅度为 2Vpp *:C1:SEQuence:SEGMent3:AMPlitude 2* 查询通道 1 序列第三段波形的幅度 *:C1:SEQuence:SEGMent3:AMPlitude?* 返回值： *“2”*

3.13.28 <channel>:SEQuence:SEGMent<x>:OFFset

描述	该命令用于设置（查询）序列某一段波形的偏置
命令语法	<channel>:SEQuence:SEGMent<x>:OFFset <offset> <channel>:= {C1，C2}. <x>:= 段编号 <offset>:= {-8V 到 8V}
查询语法	<channel>:SEQuence:SEGMent<x>:OFFset?
示例	设置通道 1 序列第三段波形的偏置为 2Vdc *:C1:SEQuence:SEGMent3:OFFset 2* 查询通道 1 序列第三段波形的偏置 *:C1:SEQuence:SEGMent3:OFFset?* 返回值： *“2”*

3.13.29 <channel>:SEQuence:SEGMent<x>:VOLTage:HIGH

描述	该命令用于设置（查询）序列某一段波形的高电平值
命令语法	<channel>:SEQuence:SEGMent<x>:VOLTage:HIGH <highLevel> <channel>:= {C1，C2}. <x>:= 段编号 <highLevel>:= {lowLevel 到 12V}
查询语法	<channel>:SEQuence:SEGMent<x>:VOLTage:HIGH?
示例	设置通道 1 序列第三段波形的高电平为 10V *:C1:SEQuence:SEGMent3:VOLTage:HIGH 10* 查询通道 1 序列第三段波形的高电平值 *:C1:SEQuence:SEGMent3:VOLTage:HIGH?* 返回值： *“10”*

3.13.30 <channel>:SEQuence:SEGMent<x>:VOLTage: LOW

描述	该命令用于设置（查询）序列某一段波形的低电平值
命令语法	<channel>:SEQuence:SEGMent<x>:VOLTage:LOW <lowLevel> <channel>:= {C1，C2}. <x>:= 段编号 <lowLevel>:= {-8V 到 highLevel}
查询语法	<channel>:SEQuence:SEGMent<x>:VOLTage:LOW?
示例	设置通道 1 序列第三段波形的低电平为 5V *:C1:SEQuence:SEGMent3:VOLTage:LOW 5* 查询通道 1 序列第三段波形的低电平值 *:C1:SEQuence:SEGMent3:VOLTage:LOW?* 返回值： *“5”*

3.14 IQ 命令

3.14.1 IQ:WAVeinfo?

描述	该命令查询 I/Q 的波形信息。
查询语法	IQ:WAVeinfo?
示例	查询当前 I/Q 波形信息： *IQ:WAVeinfo?* 返回值： *WAVE_INFO,SYMBOL_LENTH,1024,OVER_SAMPLING,4,MODULATION,2ASK,FILTER_TYPE,RootCosine,FILTER_ALPHA,0.35*

3.14.2 IQ:CENTerfreq

描述	该命令设置 I/Q 调制的中心频率。
命令语法	IQ:CENTerfreq <中心频率><单位> <中心频率>:= 中心频率。该参数的有效范围请查阅数据手册。 <单位> := {Hz，kHz，MHz，GHz}. 默认单位是赫兹“Hz”。
查询语法	IQ:CENTerfreq?
响应格式	<中心频率> (用 Hz 表示)
示例	设置中心频率为 1kHz： *IQ:CENTerfreq 1000Hz*

3.14.3 IQ:SAMPlerate

描述	该命令设置 I/Q 的采样率
命令语法	IQ:SAMPlerate<采样率><单位> <采样率> := 采样率。该参数的有效范围请查阅数据手册。 <单位> := {Hz，kHz，MHz，GHz}. 默认单位是赫兹“Hz”。
查询语法	IQ:SAMPlerate?
响应格式	<采样率> (用 Hz 表示)
示例	设置采样率为 100kHz：

	IQ:SAMPlerate 100000 或： *IQ:SAMP 100kHz*

3.14.4 IQ:SYMBolrate

描述	该命令用于设置 I/Q 的符号率
命令语法	IQ:SYMBolrate <符号率><单位> <符号率> := 符号率。该参数的有效范围请查阅数据手册。 <单位> := {S/s，kS/s，MS/s}。默认单位是符号每秒“S/s”
查询语法	IQ:SYMBolrate?
响应格式	<符号率> (使用 S/s 表示)
示例	设置符号率为 1MS/s： *IQ:SYMB 1MS/s*

3.14.5 IQ:AMPLitude

描述	该命令设置 I/Q 幅度。
命令语法	IQ:AMPLitude <幅值><单位> <幅值> := 幅度。该参数的有效范围请查阅数据手册。 <单位> := {Vrms，mVrms，dBm}。默认单位是伏特，均方根“Vrms”。
查询语法	IQ:AMPLitude?
响应格式	<幅值> (表示为 Vrms.)
示例	设置 IQ 的幅值(sqrt(I2+Q2))为 0.2Vrms： *IQ:AMPL 0.2*

3.14.6 IQ:IQADjustment:GAIN

描述	此命令在保持 IQ 复合状态下调节 I 与 Q 的比例。
命令语法	IQ:IQADjustment:GAIN <增益比><单位> <增益比> := I 与 Q 的增益比。 <单位> := {dB}.
查询语法	IQ:IQADjustment:GAIN?

响应格式	<增益比> (用单位 dB 表示)
示例	设置 I/Q 的增益比例为 0.1dB: *IQ:IQADjustment:GAIN 0.1*

3.14.7 IQ:IQADjustment:IOFFset

描述	此命令调节 I 通道的偏移量。
命令语法	IQ:IQADjustment:IOFFset <偏置><单位> <偏置> := I 的偏移量 <单位> := {V，mV，uV}。默认单位是伏特“V”。
查询语法	IQ:IQADjustment:IOFFset?
响应格式	<偏置>(表示为 V.)
示例	设置 I 的偏置为 1mV: *IQ:IQADjustment:IOFFset 1mV*

3.14.8 IQ:IQADjustment:QOFFset

描述	此命令调节 Q 通道的偏移量。
命令语法	IQ:IQADjustment:QOFFset <偏置><单位> <偏置> := Q 的偏移量。 <单位> := {V，mV，uV}。默认单位是伏特，“V”。
查询语法	IQ:IQADjustment:QOFFset?
响应格式	<偏置>(用 V 表示)
示例	设置 Q 的偏置为-1mV: *IQ:IQAD:QOFF -0.001*

3.14.9 IQ:IQADjustment:QSKew

描述	该命令通过增加或者减少 Q 的相位角来调节 I 和 Q 向量的相位角。
命令语法	IQ:IQADjustment:QSKew <角度> <角度> := 角度。单位是度。
查询语法	IQ:IQADjustment:QSKew?
响应格式	<角度>(使用度表示)

示例	设置 Q 角度为 1°： *IQ:IQADjustment:QSKew 1.0*

3.14.10 IQ:TRIGger:SOURce

描述	该命令设置 I/Q 的触发源。
命令语法	IQ:TRIGger:SOURce <触发源> <触发源>:={INTernal，EXTernal，MANual}， 其中 INTernal 是内部触发，EXTernal 为外部触发，MANual 为手动触发。
查询语法	IQ:TRIGger:SOURce?
响应格式	<触发源>
示例	设置触发源为内触发： *IQ:TRIGger:SOURce INTernal*

3.14.11 IQ:WAVEload:BUILtin

描述	此命令用于从内建波形列表中选择 I/Q 波形。
命令语法	IQ:WAVEload:BUILtin <波形名称> <波形名称>：={下表的波形名}.
查询语法	IQ:WAVEload?
响应格式	BUILtin|USERstored <波形名称>
示例	设置 I/Q 波形为内建波形的 2ASK： *IQ:WAVE:BUIL "2ASK"*

2ASK	4ASK	8ASK	BPSK	QPSK
8PSK	DBPSK	DQPSK	D8PSK	8QAM
16QAM	32QAM	64QAM	128QAM	256QAM

3.14.12 IQ:WAVEload:USERstored

描述	此命令用于在用户存储波形中选择 I/Q 波形。
命令语法	格式 1：IQ:WAVEload:USERstored <波形名称> <波形名称>:= {来自用户存储的波形}。

	格式 2：IQ:WAVEload:USERstored <路径> <路径> :={来自用户存储（本地，U 盘）中的波形路径，包括文件名和后缀}。
查询语法	IQ:WAVEload?
响应格式	BUILtin|USERstored <波形名称>
示例 1	设置 I/Q 波形为用户存储波形 wave1.arb： *IQ:WAVEload:USERstored wave1.arb*
示例 2	设置 I/Q 波形为 U 盘存储波形 wave1.arb： *IQ:WAVEload:USERstored "U-disk0/ wave/wave1.arb"*

备注 1：

1) 路径必须使用双引号把英文包括，例如："U-disk0/ wave/wave1.arb"。具体可用路径请参考文件管理器。

2) 不指定路径默认为本地路径，不添加后缀名默认为 .wav 后缀。

3.14.13 IQ:FrequencySampling

描述	此命令设置 I/Q 采样率。
命令语法	IQ: FrequencySampling <sampling> < sampling >:= {1000-200000000}. 单位为"Hz"。
查询语法	IQ:FrequencySampling? 查询当前采样率 IQ:FrequencySamplingLimit? 查询可设置采样率范围
响应格式	<sampling> MAX，<max_sampling>，MIN，< min_sampling >
示例	设置 I/Q 采样率为：2000000 Hz *IQ:FrequencySampling 2000000*

3.15 通道跟踪、耦合命令

描述

设置或者获取通道耦合参数。只有在追踪功能关闭时才能设置耦合值。

命令语法

COUPling<参数>，<值>

<参数>：={下表的参数}.

<值>：={相关参数的值}.

参数	值	描述
TRACE	<track_enble>	:= {ON，OFF}；通道跟踪状态
FCOUP	<fcoup>	:= {ON，OFF}；频率耦合状态
FDEV	<frq_dev>	:=两通道间的频率差值。单位是赫兹'Hz'
FRAT	<frat>	:=两通道间的频率比值
PCOUP	<pcoup>	:= {ON，OFF}；相位耦合状态
PDEV	<pha_dev>	:=两通道间的相位差值。单位是'度'。通道跟踪时的相位偏差设置也是用该参数。
PRAT	<prat>	:=两通道间的相位比值。
ACOUP	<acoup>	:= {ON，OFF}；幅度耦合的状态
ARAT	<arat>	:=两通道间的幅度比值
ADEV	<adev>	:=两通道间的幅度偏差。单位为伏特，峰峰值'Vpp'。
TRDUCH	<trigger_ch>	:={ON，OFF}；设置手动触发，ON 表示双通道触发，OFF 表示单通道触发。

查询语法

COUPling?

响应格式

COUP <参数>

<参数> := {所有耦合参数值}.

示例

设置耦合状态为打开：

COUP STATE,ON

设置频率偏差为 5Hz：

COUP FDEV,5

设置幅度比例为 2：

COUP ARAT,2

查询耦合信息：

COUP?

返回值：

COUPTRACE,OFF,FCOUP,ON,PCOUP,ON,ACOUP,ON,FDEV,5HZ,PRAT,1,ARAT,2

3.16 通道复制命令

描述	该命令将一个通道的参数复制到另一通道上。
命令语法	ParaCoPy <目标通道>，<通道源> <目标通道>：={C1，C2}. <通道源>：={C1，C2}. 备注：C1 和 C2 的参数必须一起设置到设备。
示例	通道 1 的参数复制到通道 2 上。 *PACP C2,C1*

3.17 通道极性命令

描述	此命令用于设置或者获取指定通道的极性。
命令语法	<通道>：INVerT <状态> <通道>：={C1，C2}. <状态>：={ON，OFF}.
查询语法	<通道>：INVerT? <通道>：={C1，C2}.
响应格式	<通道>:INVT <状态>
示例	设置 CH1 的极性为反相： *C1:INVT ON* 读取 CH1 的极性： *C1:INVT?* 返回值： *C1:INVT ON*

3.18 同相位命令

描述	此命令用于设置两个通道的相位同步
命令语法	EQPHASE
响应格式	EQPHASE <状态>
示例	设置同相位： *EQPHASE*

3.19 波形合并命令

描述	此命令设置或者获取波形合并。
命令语法	<通道>:CoMBiNe <状态> <通道>:= {C1，C2}. <状态>:= {ON，OFF}.
查询语法	<通道>:CoMBiNe? <通道>:= {C1，C2}.
响应格式	<通道>:CMBN<状态>
示例	打开 CH1 的波形合并： *C1:CoMBiNe ON* 查询 CH2 的波形合并状态： *C2:CMBN?* 返回值： *C2:CMBN OFF*

3.20 开机输出状态命令

描述	此命令设置或者获取上电后通道输出状态。
命令语法	<通道>:output POWERON_STATE,<状态> <通道>:= {C1，C2}. <状态>:= {ON，OFF}.
查询语法	<通道>:output? <通道>:= {C1，C2}.
响应格式	<通道>:output
示例	设置开机上电后通道 1 输出状态为开： *C1:output POWERON_STATE,ON* 查询通道 1 开机上电后的输出状态： *C1:output?* 返回值： *C1:OUTP ON,LOAD,HZ,POWERON_STATE,OFF,PLRT,NOR*

3.21 同步命令

描述	此命令用于设置同步信号。
命令语法	SYNC <状态> <状态>：＝{ON，OFF}.
查询语法	SYNC?
响应格式	SYNC <状态>
示例	打开同步功能： *SYNC ON* 读取同步功能状态： *SYNC?* 返回值： *C1:SYNC ON,TYPE,CH1*

描述	此命令用于设置同步信号类型。
命令语法	SYNC <参数>，<值> <参数>：＝TYPE <值>：＝{CH1，CH2，MOD_CH1，MOD_CH2}
示例	输出 CH1 同步信号： *SYNC TYPE, CH1*

描述	此命令用于设置同步信号输出方式。
命令语法	<通道>:SYNC:<参数> <值> <通道>:={C1，C2}. <参数>：＝OUT_MODE <值>：＝{Pulse, Square}
示例	设置 CH1 同步信号输出方式是 50%占空比的方波： *C1:SYNC:OUT_MODE Square*

3.22 时钟源命令

描述	此命令设置或者获取时钟源，或设置 10MHZ 时钟输出。
命令语法	格式 1：ROSCillator <src> <src>:={INT（内部时钟），EXT（外部时钟）} 格式 2：ROSCillator 10MOUT，<state> <state>:={ ON，OFF } 时钟输出状态

查询语法	ROSCillator?
响应格式	ROSC <src>,10MOUT,<state>
示例	设置内部时基作为时钟源： *ROSC INT* 打开 10MHz 时钟输出： *ROSC 10MOUT,ON*

3.23 相位模式设置命令

描述	该参数设置或者获取相位模式
命令语法	MODE<参数> <参数>:= {PHASE-LOCKED（相位锁定），INDEPENDENT（相位独立）}.
查询语法	MODE?
响应格式	MODE<参数>
示例	设置相位模式为相位独立模式： *MODE INDEPENDENT*

3.24 频率补偿开关命令

描述	该参数设置或者获取相位补偿开关状态
命令语法	Freq_com_switch <参数> <参数>:= {ON，OFF}.
查询语法	Freq_com_switch?
响应格式	Freq_com_switch <参数>
示例	设置频率补偿开关为 ON： *Freq_com_switch ON*

3.25 开机上电设置命令

描述	设置或者获取上电开机模式
命令语法	格式 1：Sys_CFG<模式> <模式>：={DEFAULT（默认），LAST（上次）} 格式 2：Sys_CFG <配置>，<路径>

	<配置>：＝USER（用户） <路径>：＝{用户存储（本地，网络存储，U 盘）的配置文件的路径，包括文件名和后缀}
查询语法	Sys_CFG?
响应格式	SCFG<模式> SCFG<配置>，<路径>
示例 1	设置上电开机系统为上次： *SCFG LAST*
示例 2	设置开机恢复文件： *SCFG USER,“state.xml”*

备注：路径必须使用双引号，英文包括且必须添加后缀名 .xml。具体可用路径请参考文件管理器。

3.26 多设备同步命令

描述	该命令设置两个或多个仪器之间的同步，并实现同相输出
命令语法	CASCADE <参数>,<值> <值>：＝{相关参数的值}。

参数	**值**	**描述**
STATE	<state>	:={ON,OFF}打开或关闭多设备同步
MODE	<type>	:= {MASTER，SLAVE}
DELAY	<time>	:={0-0.000025}，单位=s，此参数只能在从机模式时设置
SYNC_PHASE		设置一次，触发一次“同步设备”

查询语法	<CASCADE?
示例	设置设备为从机模式且延迟为 0.0000001s *CASCADE STATE,ON,MODE,SLAVE,DELAY,0.0000001*

3.27 数字格式命令

描述	此命令用于设置或者获取数字格式。
命令语法	NumBerForMat PNT，<pnt>，SEPT，<sept> <pnt>:= {Dot（点），Comma（逗号）}。小数点格式 <sept>:= {Space（空格），Off（关闭），On（打开）}。分隔符格式。

查询语法	NBFM?
响应格式	NBFM PNT，<pnt>，SEPT，<sept>
示例	设置小数点格式为点： *NBFM PNT,DOT* 设置分割符号为打开： *NBFM SEPT,ON* 读取数字格式： *NBFM?* 返回值 *NBFM PNT,DOT,SEPT,ON*

3.28 语言命令

描述	此命令设置或者获取系统语言。
命令语法	LanGuaGe <语言> <语言>：＝{EN，CH}，其中 EN 是英语，CH 是中文
查询语法	LAnGuaGe?
响应格式	LAGG <语言>
示例	设置语言为英语： *LAGG EN* 读取当前系统语言： *LAGG?* 返回值： *LAGG EN*

3.29 蜂鸣器命令

描述	此命令用于打开或者关闭蜂鸣器
命令语法	BUZZer<状态> <状态>：＝{ON，OFF}.
查询语法	BUZZer?
响应格式	BUZZ<状态>
示例	打开蜂鸣器： *BUZZ ON*

3.30 SCPI 更新 UI 命令

描述	此命令用于打开或者关闭 SCPI 更新 UI 的状态
命令语法	SCPI_UPDATE_UI <状态> <状态>：={ON，OFF}.
查询语法	SCPI_UPDATE_UI?
响应格式	<状态>
示例	设置 SCPI 更新 UI 状态关闭： *SCPI_UPDATE_UI OFF*

3.31 截屏格式命令

描述	此命令用于设置截屏的格式
命令语法	SCREEN_SHOT_TYPE <格式> <格式>：={PNG, BMP}.
查询语法	SCREEN_SHOT_TYPE?
响应格式	<格式>
示例	设置截屏格式为 BMP 格式： *SCREEN_SHOT_TYPE BMP*

描述	设置截屏
命令语法	SCREEN_SHOT

3.32 屏幕保护命令

描述	此命令用于关闭或者设置屏幕保护时间（单位为分钟）。
命令语法	SCreen_SaVe <参数> <参数>：={OFF，1，5，15，30，60，120，300}.
查询语法	SCreen_SaVe?
响应格式	SCSV<参数>
示例	设置屏幕保护时间为 5 分钟： *SCSV 5* 读取当前的屏幕保护时间： *SCreen_SaVe?*

	返回值： *SCSV 5MIN*

3.33 频率计命令

描述	此命令设置或者获取频率计参数。
命令语法	FreqCouNTer <参数>，<值> <参数>：＝{下表的参数}. <值>：＝{相关参数的值}.

参数	**值**	**描述**
STATE	<state>	:={ON，OFF}。频率计的状态
FRQ	<frequency>	测量频率。单位是赫兹“Hz”，该参数不支持设置。
PW	<pos_width>	测量正脉宽。单位是秒“s”，该参数不支持设置。
NW	<neg_width>	测量负脉宽。单位是秒“s”，该参数不支持设置。
DUTY	<duty>	测量占空比。单位是“%”，该参数不支持设置。
FRQDEV	<freq_dev>	测量频率偏差。单位是“ppm”，该参数不支持设置。
REFQ	<ref_freq>	参考频率，用于计算频率偏差。单位是赫兹“Hz”。
TRG	<triglev>	触发电平。有效值的范围取决于机型。单位是伏特“V”。
MODE	<mode>	:={AC，DC}。耦合模式
HFR	<HFR>	:={ON，OFF}。高频抑制状态
DEF	NA	恢复频率计默认设置

查询语法	FreqCouNTer?
响应格式	FCNT <参数> <参数>:={频率计的所有参数}
示例	打开频率计： *FCNT STATE,ON* 设置参考频率为 1000Hz *FCNT REFQ,1000* 查询频率计信息： *FCNT?* 返回值：

	FCNT STATE,ON,FRQ,10000000HZ,DUTY,59.8568,REFQ, 1e+07HZ,TRG,0V,PW,5.98568e-08S,NW,4.01432e-08S,FRQDEV, 0ppm,MODE,AC,HFR,OFF,TYPE,SLOW
描述	该命令清空频率计当前测量数据
命令语法	*Clear_fcnt*

3.34 过压保护命令

描述	此命令用于设置或者获取过压保护开关状态。
命令语法	VOLTPRT<状态> <状态>：＝{ON，OFF}
查询语法	VOLTPRT?
响应格式	VOLTPRT<状态>
示例	设置过压保护状态为开: *VOLTPRT ON* 读取当前过压保护开关状态: *VOLTPRT?* 返回值: *ON*

3.35 保存列表命令

描述	此命令用于读取存储波形数据的名称。若存储单位为空，该命令将返回"EMPTY"字段。
查询语法	SToreList? SToreList? <参数> SToreList? <USER>，<路径> <参数>=：{下表所示参数} <路径>=：{ }
响应格式	<波形名称>

参数		功能
BUILDIN	<参数>	查询内建波形名称及索引号
USER	<参数>	查询本地保存的波形名称

示例

（1）读取保存在设备中的所有任意波数据（不包括 U 盘中的数据）:

STL?

返回值:

STL M0, sine, M1, noise, M2, stairup, M3, stairdn, M4, stairud, M5, ppulse, M6, npulse, M7, trapezia, M8, upramp, M9, dnramp, M10, exp_fall, M11, exp_rise, M12, logfall, M13, logrise, M14, sqrt, M15, root3, M16, x^2, M17, x^3, M18, sinc, M19, gaussian, M20, dlorentz, M21, haversine, M22, lorentz, M23, gauspuls, M24, gmonopuls, M25, tripuls, M26, cardiac, M27, quake, M28, chirp, M29, twotone, M30, snr, M31, EMPTY, M32, EMPTY, M33, EMPTY, M34, hamming, M35, hanning, M36, kaiser, M37, blackman, M38, gaussiwin, M39, triangle, M40, blackmanharris, M41, bartlett, M42, tan, M43, cot, M44, sec, M45, csc, M46, asin, M47, acos, M48, atan, M49, acot, M50, EMPTY, M51, EMPTY, M52, EMPTY, M53, DDROPOUT, M54, FCLK1, M55, FSDA1, M56, EMPTY, M57, EMPTY, M58, EMPTY, M59, EMPTY

（2）读取保存在设备中的内建波形数据:

STL? BUILDIN

返回值:

STL M0, sine, M1, noise, M10, ExpFal, M100, ECG14, M101, ECG15, M102, LFPulse, M103, Tens1, M104, Tens2, M105, Tens3, M106, Airy, M107, Besselj, M108, Bessely, M109, Dirichlet, M11, ExpRise, M110, Erf, M111, Erfc, M112, ErfcInv, M113, ErfInv, M114, Laguerre, M115, Legend, M116, Versiera, M117, Weibull, M118, LogNormal, M119, Laplace, M12, LogFall, M120, Maxwell, M121, Rayleigh, M122, Cauchy, M123, CosH, M124, CosInt, M125, CotH, M126, CscH, M127, SecH, M128, SinH, M129, SinInt, M13, LogRise, M130, TanH, M131, ACosH, M132, ASecH, M133, ASinH, M134, ATanH, M135, ACsch, M136, ACoth, M137, Bartlett, M138, BohmanWin, M139, ChebWin, M14, Sqrt, M140, FlattopWin, M141, ParzenWin, M142, TaylorWin, M143, TukeyWin, M144, SquareDuty01, M145, SquareDuty02, M146, SquareDuty04, M147,SquareDuty06, M148, SquareDuty08, M149, SquareDuty10, M15, Root3, M150, SquareDuty12, M151, SquareDuty14, M152, SquareDuty16, M153, SquareDuty18, M154, SquareDuty20, M155, SquareDuty22, M156, SquareDuty24, M157, SquareDuty26, M158, SquareDuty28, M159, SquareDuty30, M16, X^2, M160, SquareDuty32, M161, SquareDuty34, M162, SquareDuty36, M163, SquareDuty38, M164, SquareDuty40, M165, SquareDuty42, M166, SquareDuty44, M167, SquareDuty46, M168, SquareDuty48, M169, SquareDuty50, M17, X^3, M170, SquareDuty52, M171, SquareDuty54, M172, SquareDuty56, M173, SquareDuty58, M174, SquareDuty60, M175, SquareDuty62, M176, SquareDuty64, M177, SquareDuty66, M178, SquareDuty68, M179, SquareDuty70, M18, Sinc, M180, SquareDuty72, M181, SquareDuty74, M182, SquareDuty76, M183, SquareDuty78, M184, SquareDuty80, M185, SquareDuty82, M186, SquareDuty84, M187, SquareDuty86, M188, SquareDuty88, M189, SquareDuty90, M19, Gaussian, M190, SquareDuty92, M191, SquareDuty94, M192, SquareDuty96, M193, SquareDuty98, M194, SquareDuty99, M195, demo1_375pts, M196, demo1_16kpts, M197, demo2_3kpts, M198, demo2_16kpts, M2, StairUp, M20, Dlorentz, M21, Haversine, M22, Lorentz, M23, Gauspuls, M24, Gmonopuls, M25, Tripuls, M26, Cardiac, M27, Quake, M28, Chirp, M29, Twotone, M3, StairDn, M30, SNR, M31, Hamming, M32, Hanning, M33, kaiser, M34, Blackman, M35, Gausswin, M36, Triangle, M37, Bartlett-Hann, M38, Bartlett, M39, Tan, M4, StairUD, M40, Cot, M41, Sec, M42, Csc, M43, Asin, M44, Acos, M45, Atan, M46, Acot, M47, Square, M48, SineTra, M49, SineVer, M5, Ppulse, M50, AmpALT, M51, AttALT, M52, RoundHalf, M53, RoundsPM, M54, BlaseiWave, M55, DampedOsc, M56, SwingOsc, M57, Discharge, M58, Pahcur, M59, Combin, M6, Npulse, M60, SCR, M61, Butterworth, M62, Chebyshev1, M63, Chebyshev2, M64, TV, M65, Voice, M66, Surge, M67, Radar, M68, Ripple, M69, Gamma, M7, Trapezia, M70, StepResp, M71, BandLimited, M72, CPulse, M73, CWPulse, M74, GateVibr, M75, LFMPulse, M76, MCNoise, M77, AM, M78, FM, M79, PFM, M8, Upramp, M80, PM, M81, PWM, M82, EOG, M83, EEG, M84, EMG, M85, Pulseilogram, M86, ResSpeed, M87, ECG1, M88, ECG2, M89, ECG3, M9, Dnramp, M90, ECG4, M91, ECG5, M92, ECG6, M93, ECG7, M94, ECG8, M95, ECG9, M96, ECG10, M97, ECG11, M98, ECG12, M99, ECG13

（3）读取来自设备中的用户定义的波形名称：

STL? USER

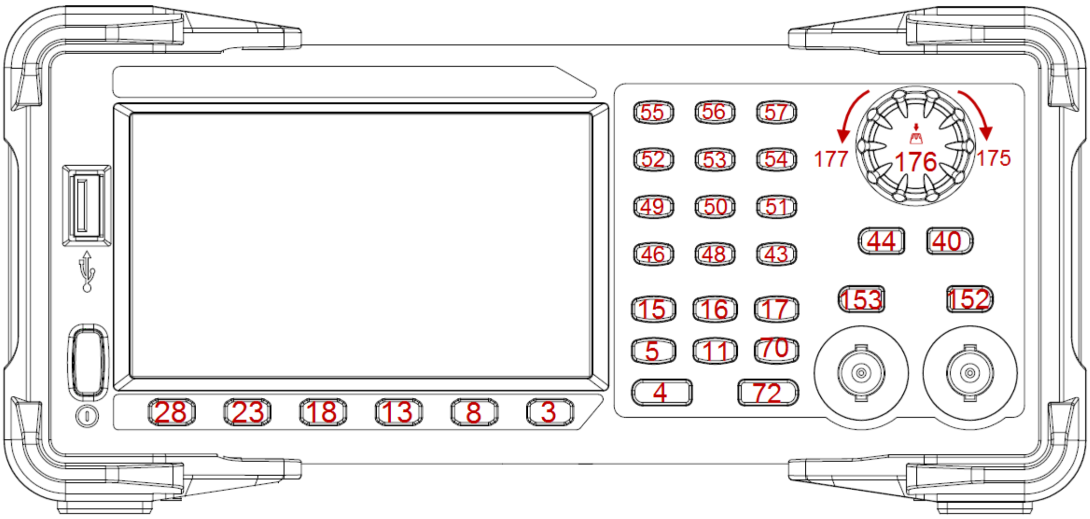
返回值：

STL WVNM,sinc_8M,sinc_3000000,sinc_1664000,
ramp_8M,sinc_2000000,sinc_50000,square_8M,sinc_5000,wave1,
square_1M

3.36 虚拟键命令

描述	此命令用于模拟前面板的按键的状态
命令语法	VirtualKEY VALUE,<值>,STATE,<状态> <值>:= {下表显示虚拟按键的索引号或者名称}. <状态>:= {0，1}，“1”是有效的虚拟值，即按下前面板的对应按键；而“0”是无效值。

名称	索引	名称	索引
KB_FUNC1	28	KB_NUMBER_4	52
KB_FUNC2	23	KB_NUMBER_5	53
KB_FUNC3	18	KB_NUMBER_6	54
KB_FUNC4	13	KB_NUMBER_7	55
KB_FUNC5	8	KB_NUMBER_8	56
KB_FUNC6	3	KB_NUMBER_9	57
KB_MOD	15	KB_POINT	46
KB_SWEEP	16	KB_NEGATIVE	43
KB_BURST	17	KB_LEFT	44
KB_UTILITY	11	KB_RIGHT	40
KB_PARAMETER	5	KB_OUTPUT1	153
KB_NUMBER_0	48	KB_OUTPUT2	152
KB_NUMBER_1	49	KB_KNOB_RIGHT	175
KB_NUMBER_2	50	KB_KNOB_LEFT	177
KB_NUMBER_3	51	KB_KNOB_DOWN	176



SDG6000X 上的按键和指示

示例	*VKEY VALUE,15,STATE,1* *VKEY VALUE,KB_SWEEP,STATE,1*

3.37 IP 命令

描述	该参数用于设置或者读取设备的 IP 地址。
命令语法	SYSTem:COMMunicate:LAN:IPADdress <参数 1>.<参数 2>.<参数 3>.<参数 4> <参数 1>：＝{在 1 至 223 之间的整数值}. <参数 2>：＝{在 0 至 255 之间的整数值}. <参数 3>：＝{在 0 至 255 之间的整数值}. <参数 4>：＝{在 0 至 255 之间的整数值}.
查询语法	SYSTem:COMMunicate:LAN:IPADdress?
示例	设置 IP 地址为 10.11.13.203： *SYST:COMM:LAN:IPAD "10.11.13.203"* 读取 IP 地址： *SYST:COMM:LAN:IPAD?* 返回值： *"10.11.13.203"*

3.38 子网掩码命令

描述	该命令用于设置或者获取设备的子网掩码。
命令语法	SYSTem:COMMunicate:LAN:SMASk <参数 1>.<参数 2>.<参数 3>.<参数 4> <参数 1>:= {在 0 至 255 之间的整数值}. <参数 2>:= {在 0 至 255 之间的整数值}. <参数 3>:= {在 0 至 255 之间的整数值}. <参数 4>:= {在 0 至 255 之间的整数值}.
查询语法	SYSTem:COMMunicate:LAN:SMASk?
示例	设置子网掩码为 255.0.0.0： *SYST:COMM:LAN:SMAS "255.0.0.0"* 获取子网掩码： *SYST:COMM:LAN:SMAS?* 返回值： *"255.0.0.0"*

3.39 网关命令

描述	该命令设置并获取设备的网关。
命令语法	SYSTem:COMMunicate:LAN:GATeway <参数 1>.<参数 2>.<参数 3>.<参数 4> <参数 1>:= {在 0 至 223 之间的整数值}. <参数 2>:= {在 0 至 255 之间的整数值}. <参数 3>:= {在 0 至 255 之间的整数值}. <参数 4>:= {在 0 至 255 之间的整数值}.
查询语法	SYSTem:COMMunicate:LAN:GATeway?
示例	设置网关为 10.11.13.1: *SYSTem:COMMunicate:LAN:GATeway "10.11.13.1"* 获取网关: *SYSTem:COMMunicate:LAN:GATeway?* 返回值: *"10.11.13.1"*

3.40 Gpib 命令

描述	该命令设置并获取 Gpib 的端口号。
命令语法	SYSTem:COMMunicate:GPIB:ADDRess <num> <num>.=Gpib 端口号
查询语法	SYSTem:COMMunicate:GPIB:ADDRess?
示例	设置 Gpib 端口号为 19: :SYSTem:COMMunicate:GPIB:ADDRess 19 获取 Gpib 端口号: *SYSTem:COMMunicate:GPIB:ADDRess?* 返回值: *19*

3.41 文件管理命令

3.41.1 MMEMory:DELete

描述	该命令用于删除指定文件
命令语法	MMEMory:DELete <参数> <参数>:=文件路径（包含操作文件名称）
示例	删除路径为”Local/1000pts.bin”的文件： *MMEMory:DELete “1000pts.bin”*

3.41.2 MMEMory:RDIRectory

描述	该命令用于删除指定文件夹
命令语法	MMEMory:RDIRectory <参数> <参数>:=文件路径（包含操作文件夹名称）
示例	删除文件夹路径为 “Local/”下名为 test 的文件夹： *MMEMory:RDIRectory “test”*

3.41.3 MMEMory:MDIRectory

描述	该命令用于新建指定路径的文件夹
命令语法	MMEMory:MDIRectory <参数> <参数>:=文件路径（包含操作文件夹名称）
示例	新建一个 “Local”路径下的名为 test 的文件夹： *MMEMory:MDIRectory “test”*

3.41.4 MMEMory:CATalog

描述	该命令用于查询指定路径下的全部文件和文件夹或查看指定格式文件
命令语法	
查询语法	查看路径下的所有文件和文件夹 MMEMory:CATalog? <参数> <参数>:=文件夹路径 MMEMory:CATalog: <参数 1>?<参数 2>

	<参数 1>:=STATe:XMLanguage <参数 2>:=文件夹路径
响应格式	剩余内存大小，已用内存大小 “文件名，文件类型，文件大小”
示例	查看“Local/”路径下的所有文件和文件夹: *MMEMory:CATalog? “”* 查看 “Local/”路径下的“.arb” 或 “.ARB” 后缀的文件: *MMEMory:CATalog:DATA:ARBitrary? “”* 查看“Local/”路径下的“.xml” 或 “.XML”后缀文件: *MMEMory:CATalog:STATe:XMLanguage? “”*

3.41.5 MMEMory:COPY

描述	此命令复制一个文件或文件夹
命令格式	MMEMory:COPY <参数> <参数>:=“源文件的路径”，“目标路径” 复制文件：源文件路径和目标路径均包含操作文件名称 复制文件夹：源文件路径包含操作文件夹和目标路径不包含操作文件名称
查询命令	
示例	复制路径为“Local/test/1000pts.bin”的文件并粘贴至 “Local/test.bin” *MMEMory:COPY “test/1000pts.bin”,”test.bin”* 复制路径为”Local/src”的文件/文件夹并粘贴至“Local/copy/”文件夹中 *MMEMory:COPY “src”, ”copy”*

3.41.6 MMEMory:MOVE

描述	此命令移动一个文件或文件夹到新的路径下
命令语法	MMEMory:MOVE <参数> <参数>:= “源文件/文件夹的路径”，“目标路径” 移动文件：源文件路径和目标路径均包含操作文件名称 移动文件夹：源文件路径包含操作文件夹和目标路径不包含操作文件名称
查询语法	
响应格式	

示例	移动文件路径为”Local/test/1000pts.bin” 的文件到路径“Local/1000pts.bin” *MMEMory:MOVE “test/1000pts.bin”,”1000pts.bin”* 移动文件夹路径为”Local/src” 到文件夹路径为 “Local/copy/”下 *MMEMory:MOVE “src”, ”copy/”*

3.41.7 MMEMory:SAVE:XML

描述	此命令保存一个 xml 配置文件到默认路径或指定路径
命令语法	MMEMory:SAVE:XML <参数> <参数>:= {保存路径，包括文件名和后缀}。
查询语法	
响应格式	
示例	保存 test.xml 文件到本地 *MMEMory:SAVE:XML "Local/test.xml" 或* *MMEMory:SAVE:XML "test.xml"* 保存 test.xml 文件到 U 盘 *MMEMory:SAVE:XML "U-disk0/test.xml"*

3.41.8 MMEMory:LOAD:XML

描述	此命令从默认路径或指定路径加载一个 xml 配置文件
命令语法	MMEMory:LOAD:XML <参数> <参数>:= {路径，包括文件名和后缀}。
查询语法	
响应格式	
示例	从本地加载 test.xml 文件 *MMEMory:LOAD:XML "Local/test.xml" 或* *MMEMory:LOAD:XML "test.xml"* 从 U 盘加载 test.xml 文件 *MMEMory:LOAD:XML "U-disk0/test.xml"*

3.41.9 MMEMory:TRANSfer

描述	此命令可发送自定义的数据到指定路径下指定的 bin 文件
命令语法	MMEMory:TRANSfer <参数>,#{data} <参数>:= {保存路径，包括路径、文件名及后缀} {data}:=数据的长度的长度+数据长度+二进制数据
示例	发送数据长度的长度为 1，数据长度为 4 的数据到本地下的 wave1.bin *MMEMory:TRANSfer "Local/wave1.bin",#14ABCD*

4 波形格式

本章节给出信号源使用到的波形文件格式的说明。通过这些说明，你可以了解如何自定义编辑波形文件，并结合上个章节列出的命令实现对信号源控制。

4.1 bin

bin 文件内容为二进制，文件内容就是各个点的码字值（码字范围-32768~32767），不支持手动编辑，导入文件时，保持当前的幅度，频率、偏移信息，直接将各码字值转换为电压输出。

4.2 csv/dat

csv 与 dat 文件内容为 text。SDG6000X 支持的 CSV 文件分为头部信息段与波形数据段两部分。

1. csv 头部信息段内容包含以下四项，可以有多的信息项，但必须包含以下四项，多的项会被忽略。每项一行，以换行符结束

信息项	描述
amp,value	波形的幅度
offset,value	波形的偏移
frequency,value	波形的频率
data length,value	波形的长度

2. 数据段数据每一组占一行。可用以下四种字符串之一作为起始标识（标识符占一行），放在正式数据之前。

   Second,Volt

   xpos,value

   Time,Ampl

   Second,Value

   csv 格式数据示例图如下：

1	data length	16384	
2	frequency	1000	
3	amp	2	
4	offset	0	
5	phase	0	
6			
7			
8			
9			
10			
11			
12			
13	xpos	value	
14	1	0	
15	2	0.000383	

csv 文件去掉头部信息段，只保留数据段，就是.dat 格式

dat 格式数据示例图如下：

```
Source,CH1
Second,Value
-1.0000000000E-04,-3.764706E-02
-9.9999800000E-05,-3.764706E-02
-9.9999600000E-05,-4.705882E-02
-9.9999400000E-05,-6.117647E-02
-9.9999200000E-05,-4.705882E-02
-9.9999000000E-05,-6.117647E-02
-9.9998800000E-05,-2.823529E-02
-9.9998600000E-05,-4.235294E-02
-9.9998400000E-05,-3.764706E-02
```

5 编程示例

本章节给出了一些编程示例。通过这些例子，你可以了解如何使用 VISA 或者 sockets，并结合上节列出的命令实现对信号源控制。通过下面的例子，你可以开发更多应用。

5.1 VISA 应用示例 VC++示例

环境：Win7 32 位系统，Visual Studio

描述：分别通过 USBTMC 和 TCP/IP 访问信号源，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。此处给出两种设置方法，分别为静态和动态：

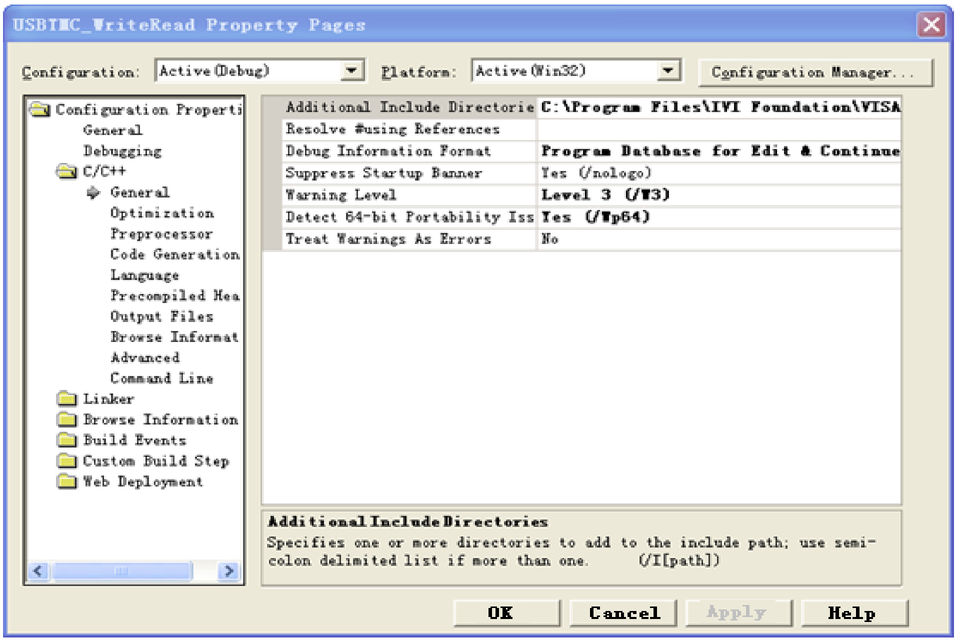
a) 静态：

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：

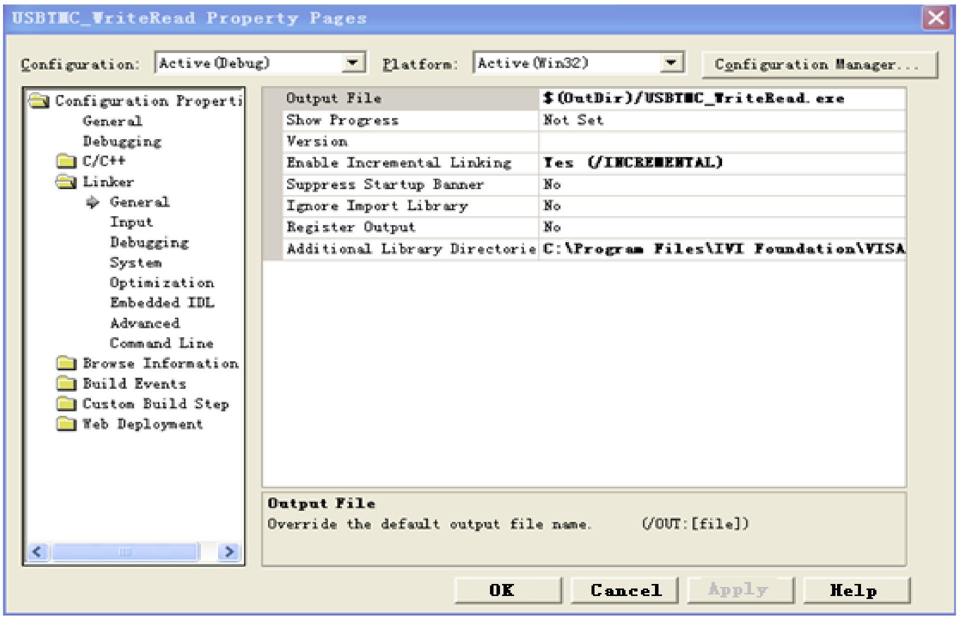
```
#include "visa.h"
#pragma comment(lib,"visa32.lib")
```

b) 动态：

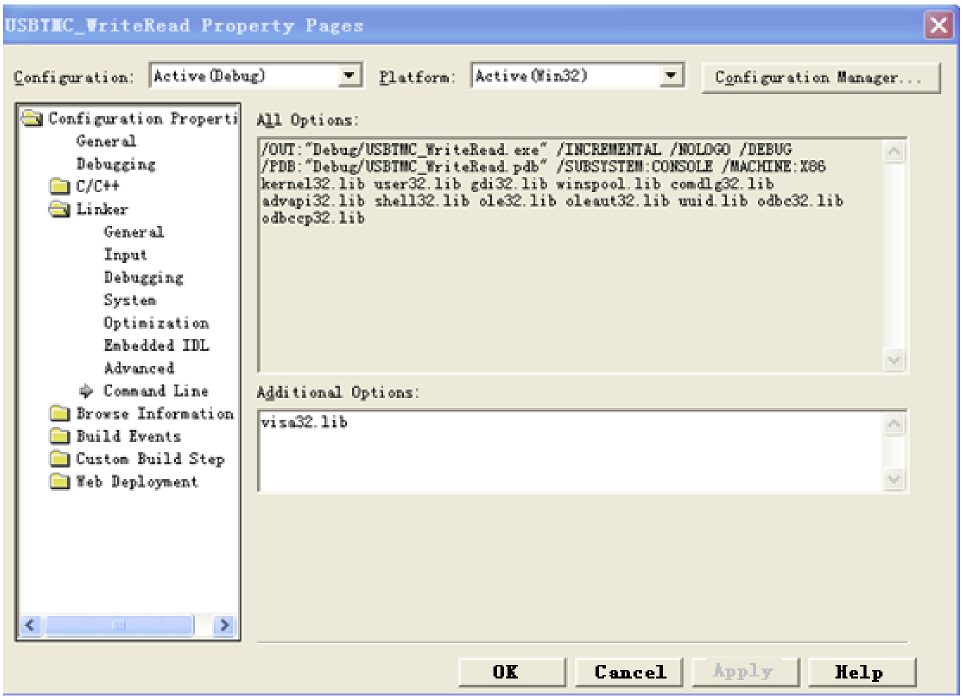
点击"project>>properties"，在属性对话框左侧选择“c/c++---General”中，将“Additional Include Directories”项的值设置为 NI-VISA 的安装路径（例如：C:\Program Files\IVI Foundation\VISA\WinNT\include)，如下图所示：



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>
```

3. 编码：

a) USBTMC:

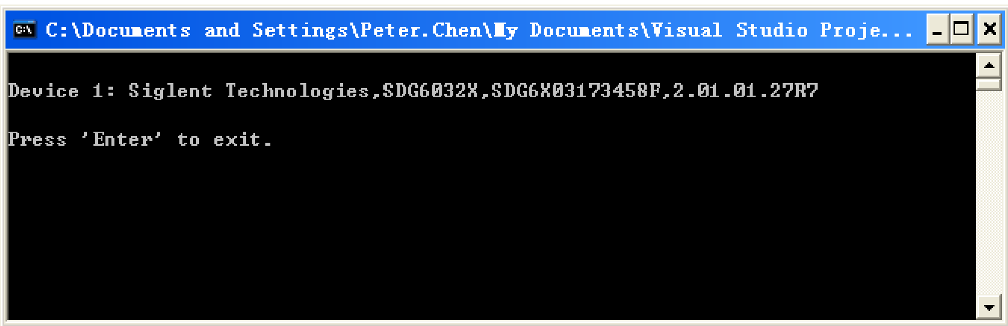
```
int Usbtmc_test()
{
/* This code demonstrates sending synchronous read & write commands */
/* to an USB Test & Measurement Class (USBTMC) instrument using       */
/* NI-VISA                                                             */
/* The example writes the "*IDN?\n" string to all the USBTMC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

```
ViSession    defaultRM;
ViSession    instr;
ViUInt32    numInstrs;
ViFindList    findList;
ViStatus    status;
char  instrResourceString[VI_FIND_BUFLEN];
unsigned   char buffer[100];
int     i;
/** First we must call viOpenDefaultRM to get the manager
* handle.    We will store this handle in defaultRM.*/
status=viOpenDefaultRM (&defaultRM);
if (status<VI_SUCCESS)
{
printf ("Could not open a session to the VISA Resource Manager!\n");
return        status;
}
/* Find all the USB TMC VISA resources in our system and store the    number of resources in the system in
numInstrs.                              */
status = viFindRsrc (defaultRM, "USB?*INSTR", &findList, &numInstrs, instrResourceString);
if (status<VI_SUCCESS)
{
      printf ("An error occurred while finding resources.\nPress 'Enter' to continue.");
      fflush(stdin);
      getchar();
      viClose (defaultRM);
      return        status;
}
/** Now we will open VISA sessions to all USB TMC instruments.
* We must use the handle from viOpenDefaultRM and we must
* also use a string that indicates which instrument to open.    This
* is called the instrument descriptor.    The format for this string
* can be found in the function panel by right clicking on the
* descriptor parameter. After opening a session to the
* device, we will get a handle to the instrument which we
* will use in later VISA functions.    The AccessMode and Timeout
```

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

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

    int _tmain(int argc, _TCHAR* argv[])
    {
    Usbtmc_test();
    return 0;
    }
```

运行结果：



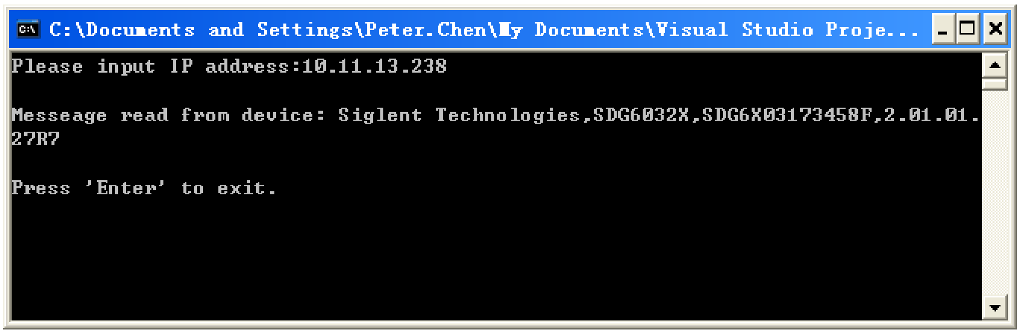
b) TCP/IP:

```
    int TCP_IP_Test(char *pIP)
    {
    char outputBuffer[VI_FIND_BUFLEN];
```

```
ViSession defaultRM, instr;
ViStatus status;
/* First we will need to open the default resource manager. */
status = viOpenDefaultRM (&defaultRM);
if (status<VI_SUCCESS)
{
	printf("Could not open a session to the VISA Resource Manager!\n");
}
/* Now we will open a session via TCP/IP device */
char head[256] ="TCPIP0::";
char tail[] ="::INSTR";
strcat(head,pIP);
strcat(head,tail);
status = viOpen (defaultRM, head, VI_LOAD_CONFIG, VI_NULL, &instr);
if (status<VI_SUCCESS)
{
	printf ("An error occurred opening the session\n");
	viClose(defaultRM);
}
status = viPrintf(instr, "*idn?\n");
status = viScanf(instr, "%t", outputBuffer);
if (status<VI_SUCCESS)
{
	printf("viRead failed with error code: %x \n",status);
	viClose(defaultRM);
}
else
{
	printf ("\nMesseage read from device: %*s\n", 0,outputBuffer);
}
status = viClose (instr);
status = viClose (defaultRM);
printf("Press 'Enter' to exit.");
fflush(stdin);
```

```
getchar();
return 0;
}

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

运行结果：



5.1.2 VB 示例

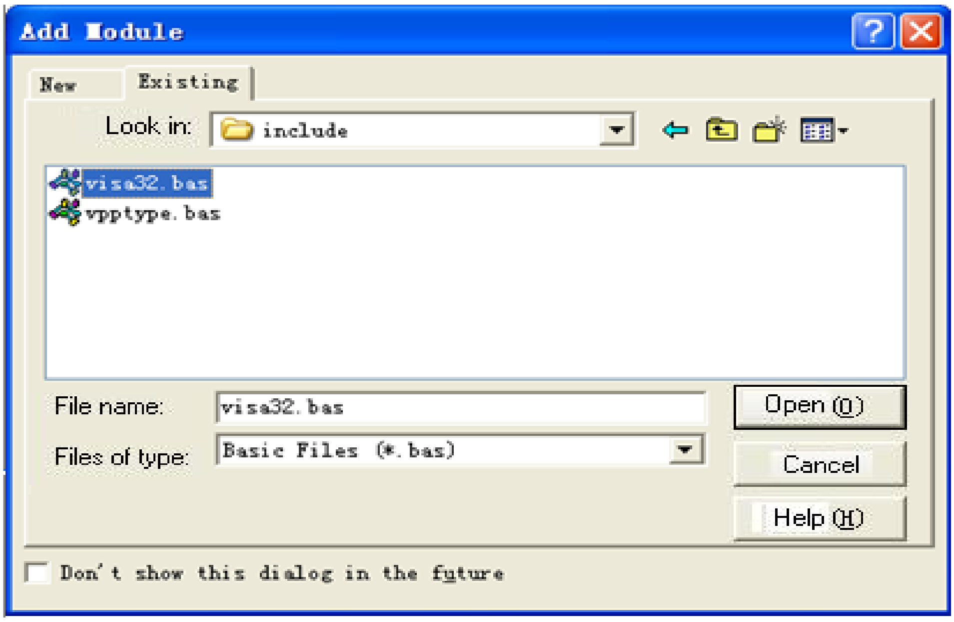
环境： Windows7 32 位系统，Microsoft Visual Basic 6.0

描述： 分别通过 USBTMC 和 TCP/IP 访问信号源，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。

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：



2. 编码：

a) USBTMC:

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

```
'       Write the Identification Query Using viWrite
'       Try to Read a Response With viRead
'       Close the VISA Session
Const MAX_CNT = 200

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

' First we must call viOpenDefaultRM to get the manager
' handle. We will store this handle in defaultRM.
    status = viOpenDefaultRM(defaultRM)
If (status < VI_SUCCESS) Then
        resultTxt.Text = "Could not open a session to the VISA Resource Manager!"
        Usbtmc_test = status
ExitFunction
EndIf

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

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

' At this point we now have a session open to the USB TMC instrument.
' We will now use the viWrite function to send the device the string "*IDN?",
' asking for the device's identification.
        status = viWrite(instrsesn, "*IDN?", 5, retCount)
If (status < VI_SUCCESS) Then
            resultTxt.Text = "Error writing to the device."
            status = viClose(instrsesn)
GoTo NextFind
EndIf

' Now we will attempt to read back a response from the device to
' the identification query that was sent.    We will use the viRead
' function to acquire the data.
' After the data has been read the response is displayed.
```

```
            status = viRead(instrsesn, Buffer, MAX_CNT, retCount)
    If (status < VI_SUCCESS) Then
                resultTxt.Text = "Error reading a response from the device." + CStr(i + 1)
    Else
                resultTxt.Text = "Read from device: " + CStr(i + 1) + " " + Buffer
    EndIf
            status = viClose(instrsesn)
    Next i

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

b) TCP/IP:

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

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

    ' Now we will open a session via TCP/IP device
        status = viOpen(defaultRM, "TCPIP0::" + ip + "::INSTR", VI_LOAD_CONFIG, VI_NULL, instrsesn)
```

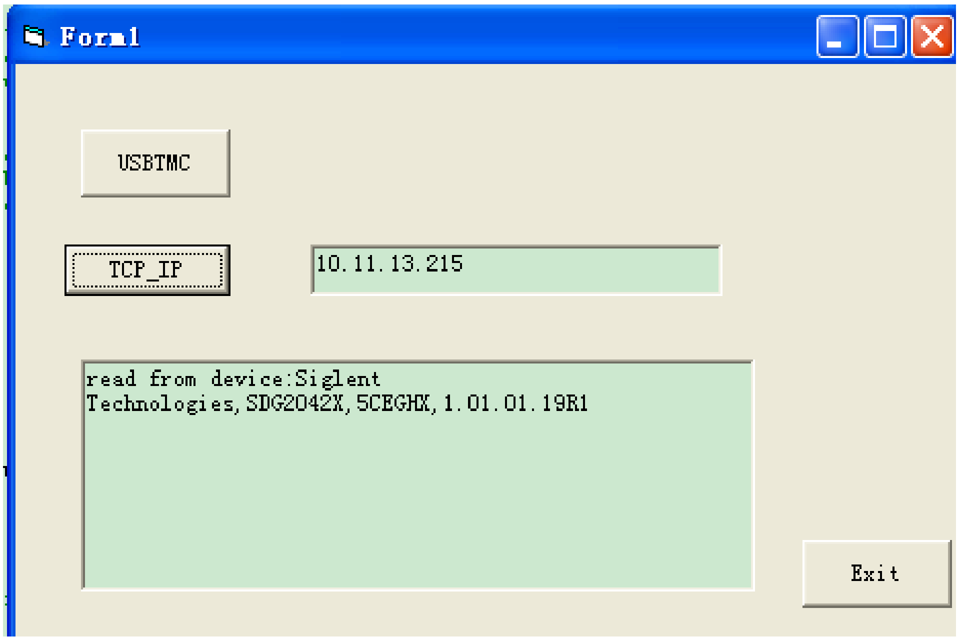
```
If (status < VI_SUCCESS) Then
        resultTxt.Text = "An error occurred opening the session"
        viClose(defaultRM)
        TCP_IP_Test = status
ExitFunction
EndIf

    status = viWrite(instrsesn, "*IDN?", 5, count)
If (status < VI_SUCCESS) Then
        resultTxt.Text = "Error writing to the device."
EndIf
    status = viRead(instrsesn, outputBuffer, VI_FIND_BUFLEN, count)
If (status < VI_SUCCESS) Then
        resultTxt.Text = "Error reading a response from the device." + CStr(i + 1)
Else
        resultTxt.Text = "read from device:" + outputBuffer
EndIf
    status = viClose(instrsesn)
    status = viClose(defaultRM)
    TCP_IP_Test = 0
EndFunction
```

c) Button control code:

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

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

运行结果：



5.1.3 MATLAB 示例

环境：Windows 7 32 位系统，MATLAB R2013a

描述：分别通过 USBTMC 和 TCP/IP 访问信号源，并在 NI-VISA 上发送“*IDN?”命令来查询设备信息。

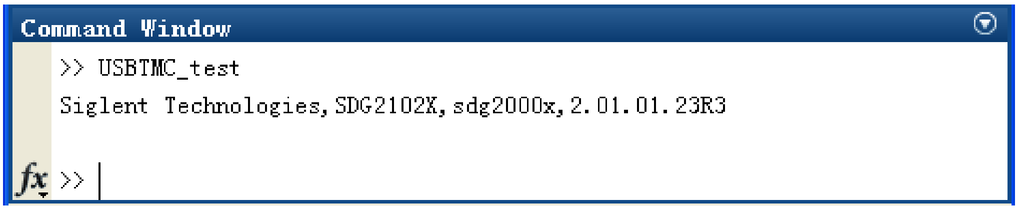
步骤：

1. 打开 MATLAB 软件，并修改当前目录。在本示例中，当前目录修改为："D:\USBTMC_TCPIP_Demo"。
2. 点击在 Matlab 界面的 File>>New>>Script 创建一个空的 M 文件。
3. 编码：

a) USBTMC:

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

%Create a VISA-USB object connected to a USB instrument
vu = visa('ni','USB0::0xF4ED::0xEE3A::sdg6000x::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
```

运行结果：

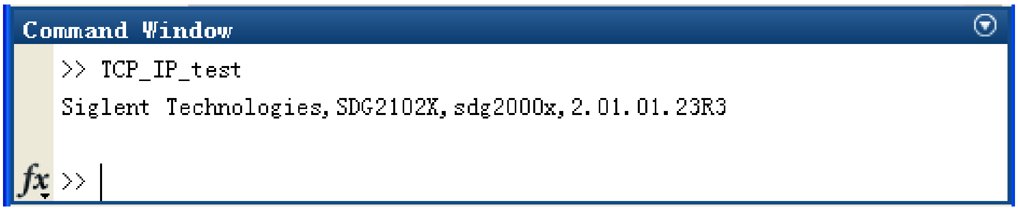


b) TCP/IP：

写一个名为 TCP_IP_Test 的函数：

```
function TCP_IP_test()
% This code demonstrates sending synchronous read & write commands
% to an TCP/IP instrument using NI-VISA

%Create a VISA-TCPIP object connected to an instrument
%configured with IP address.
vt = visa('ni',['TCPIP0::','10.11.13.32','::INSTR']);

%Open the VISA object created
fopen(vt);

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


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


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


 end
```

运行结果：


Command Window
>> TCP_IP_test
Siglent Technologies,SDG2102X,sdg2000x,2.01.01.23R3
>>

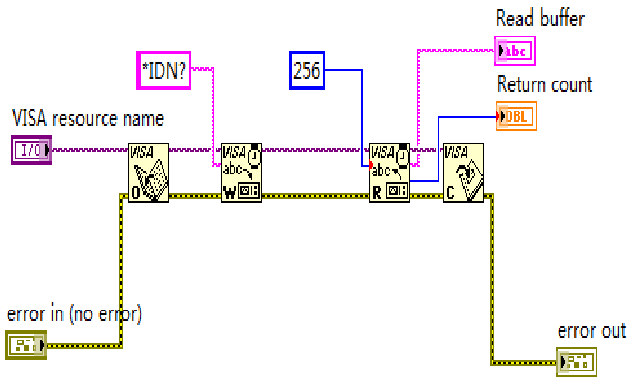
5.1.4 LabVIEW 示例

环境：Windows 7 32 位系统，LabVIEW 2011

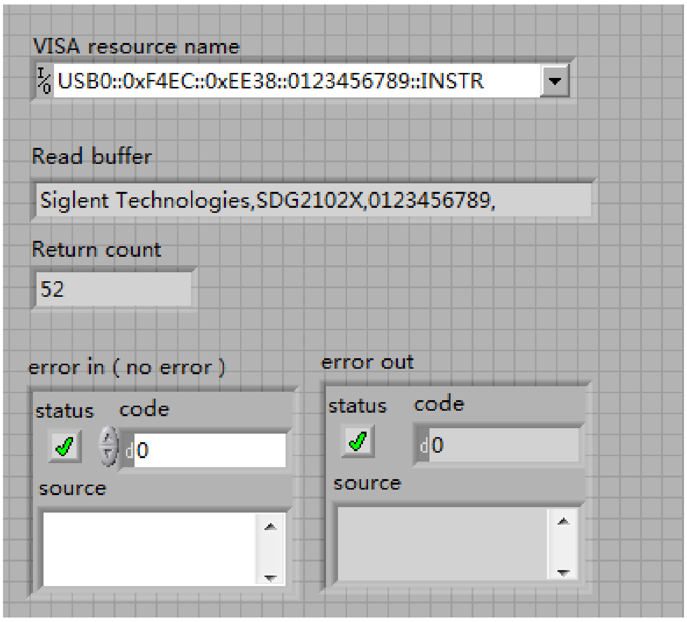
描述：分别通过 USBTMC 和 TCP/IP 访问信号源，并在 NI-VISA 上发送“*IDN?”命令来查询设备信息。

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 Palette 中选择和添加下列功能：**VISA Write、VISA Read、VISA Open 和 VISA Close**。
4. 如下图连接它们：



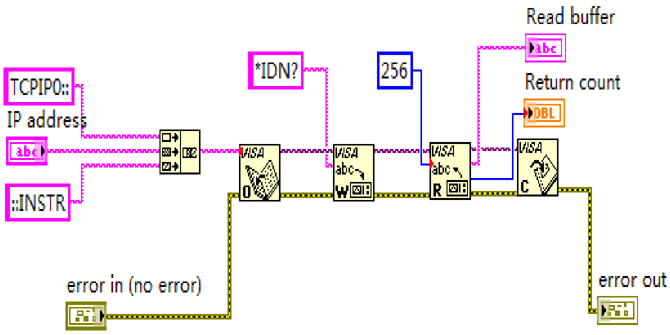
5. 从 VISA 资源名列表中选择设备资源并运行程序。



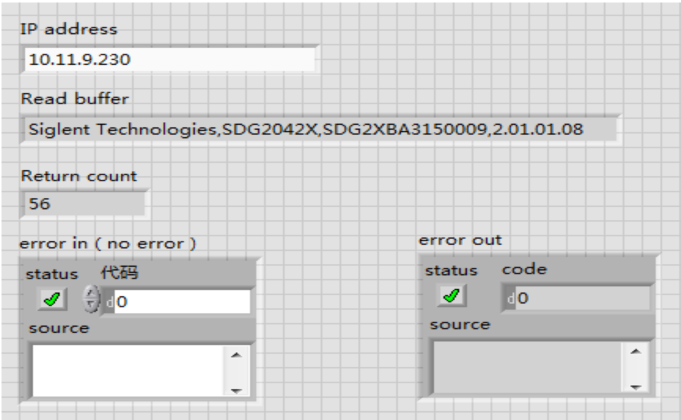
在此例中，VI 打开了一个 USBTMC 设备的 VISA 会话，并向设备写*IDN? 命令，以及回读的响应值。当所有通信完成时，VI 将关闭 VISA 会话。

6. 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 Mod >>Synchronous”以实现同步写入或读取数据。

7. 按照下图连接它们：



8. 输入 IP 地址并运行程序。



5.1.5 Python3 示例

环境：Python3.6.5，PyVISA 1.9

描述：示例中创建的波形点数，值代表的是每个波形点数的码字，用于确定波形点数的电平（每个波形点数的码字范围是-32767~32767）。

```
#!/usr/bin/env python3.6.5
# -*- coding: utf-8 -*-

import pyvisa as visa
import struct

#USB resource of Device
sdg = visa.ResourceManager().open_resource("USB0::0xF4EC::0x1105::SDG6000XABC002::INSTR")
data = [32767, 32767, 32767, 32767, 32767, 32767, 16384,16384,32767, 32767, 32767, 32767, 32767, 32767, -32767, -32767, -32767, -32767, -32767, -32767, -32767, -16384,-16384,-32767, -32767, -32767, -32767, -32767]

def int_to_two_byte_small_endian(n):
    return struct.pack('<h', n)  # '<h' 表示小端序的短整型（2 个字节）

final_byte_data = b''.join(int_to_two_byte_small_endian(num) for num in data)
print('write bytes:', len(final_byte_data))
print(final_byte_data)
cmd = bytes('C1:WVDT WVNM,wave1,FRQ,2500,AMP,2.5,OFST,0.2,WAVEDATA,', 'utf-8') + final_byte_data
sdg.write_raw(cmd)

# 通过读取 bin 文件的数据，可以将已有波形写入设备中
f = open(r"C:\Users\Administrator\Desktop\wave1.bin", "rb")
data1 = f.read()
print ('write bytes:',len(data1))
cmd = bytes('C1:WVDT WVNM,wave1,FRQ,2500,AMP,2.5,OFST,0.2,WAVEDATA,', 'utf-8') + data1
sdg.write_raw(cmd)
f.close()
```

5.2 Sockets 应用示例

5.2.1 Python 示例

Python 有一个用于访问 socket 接口的低级的网络模块。基于 sockets 编写的 Python 脚本可用于用于执行各种测试和测量任务。

环境：Windows 7 32 位系统，Python v2.7.5

Description：打开一个 socket，发送一个查询操作并循环执行 10 次后关闭 socket。注意：程序中的 SCPI 命令的字符串必须以"\n"字符（换行）作为结尾。

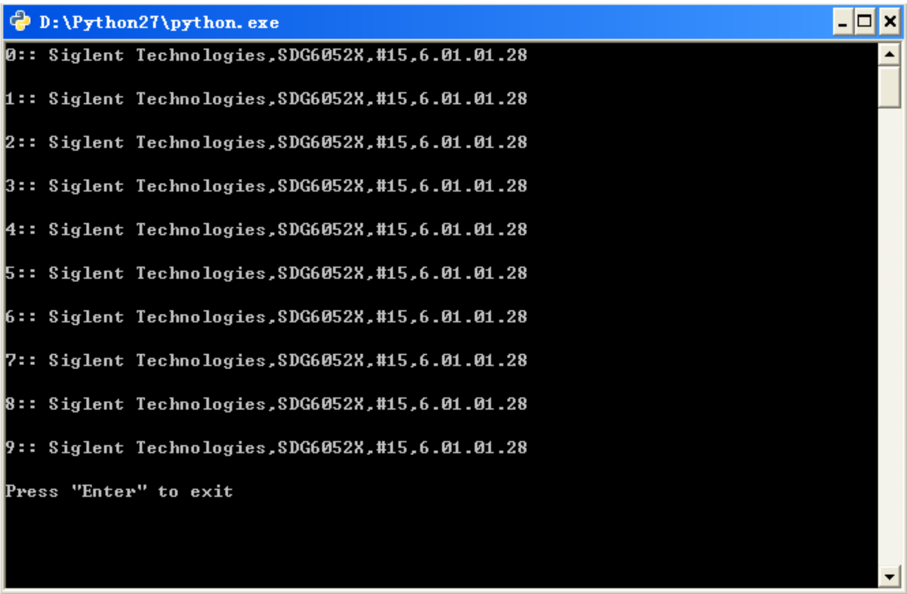
下面是脚本的代码：

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

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

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

remote_ip = "10.11.13.40"    # should match the instrument's IP address
port = 5025 # the port number of the instrument service
count = 0
def SocketConnect():
    try:
        #create an AF_INET, STREAM socket (TCP)
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    except socket.error:
        print ('Failed to create socket.')
        sys.exit();
    try:
```

```
        #Connect to remote server
        s.connect((remote_ip , port))
    except socket.error:
        print ('failed to connect to ip ' + remote_ip)
    return s

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

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

def main():
    global remote_ip
    global port
    global count

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

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

运行结果：



6 索引

SIGLENT® 鼎阳

关于鼎阳

鼎阳科技（SIGLENT）是通用电子测试测量仪器领域的行业领军企业，A 股上市公司。

2002 年，鼎阳科技创始人开始专注于示波器研发，2005 年成功研制出鼎阳第一款数字示波器。历经多年发展，鼎阳产品已扩展到数字示波器、手持示波表、函数/任意波形发生器、频谱分析仪、矢量网络分析仪、射频/微波信号源、台式万用表、直流电源、电子负载、精密源表等基础测试测量仪器产品，是全球极少数能够同时研发、生产、销售数字示波器、信号发生器、频谱分析仪和矢量网络分析仪四大通用电子测试测量仪器主力产品的厂家之一，国家重点“小巨人”企业。同时也是国内主要竞争对手中极少数同时拥有这四大主力产品并且四大主力产品全线进入高端领域的厂家。公司总部位于深圳，在马来西亚槟城州设有生产基地，在美国克利夫兰、德国奥格斯堡、日本东京成立了子公司，在成都成立了分公司，产品远销全球 80 多个国家和地区，SIGLENT 已经成为全球知名的测试测量仪器品牌。

联系我们

深圳市鼎阳科技股份有限公司
全国免费服务热线：400-878-0807
网址：www.siglent.com

声明

技术许可