

# SDG 系列 任意波形发生器

 SIGLENT® 鼎阳

编程手册  
PG02\_C01A



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

## 目录

<b>1</b>	<b>编程概述.....</b>	<b>4</b>
1.1	建立通信 .....	4
1.1.1	NI-VISA 的安装.....	4
1.1.2	连接仪器.....	8
1.2	远程控制的实现 .....	9
1.2.1	用户自定义程序.....	9
1.2.2	通过 NI-MAX 发送 SCPI 命令.....	9
1.2.3	通过 Telnet 发送 SCPI 命令.....	9
1.2.4	通过 Socket 发送 SCPI.....	11
<b>2</b>	<b>SCPI 语言简介 .....</b>	<b>12</b>
2.1	有关命令和查询 .....	12
2.2	描述 .....	12
2.3	用法 .....	12
2.4	命令符号 .....	12
2.5	命令&查询表 .....	12
<b>3</b>	<b>命令与查询.....</b>	<b>15</b>
3.1	IEEE488.2 通用命令介绍 .....	15
3.1.1	*IDN.....	15
3.1.2	*OPC.....	16
3.1.3	*RST.....	17
3.2	通用头部命令 .....	17
3.3	输出命令 .....	18
3.4	基本波形命令 .....	19
3.5	调制波形命令 .....	22
3.6	扫描波形命令 .....	26
3.7	脉冲串命令 .....	28
3.8	参数复制命令 .....	30
3.9	任意波形命令 .....	31
3.10	同步命令 .....	32
3.11	数字格式命令 .....	33
3.12	语言命令 .....	33
3.13	配置命令 .....	34
3.14	蜂鸣器命令 .....	34
3.15	屏幕保存命令 .....	35
3.16	时钟源命令 .....	35
3.17	频率计命令 .....	36
3.18	反相命令 .....	37
3.19	通道耦合命令 .....	37
3.20	过压保护命令 .....	39

3.21	保存列表命令 .....	39
3.22	任意波数据命令 .....	42
3.23	虚拟键命令 .....	43
3.24	IP 命令 .....	46
3.25	子网掩码命令 .....	47
3.26	网关命令 .....	47
3.27	采样率命令 .....	48
3.28	谐波命令 .....	49
3.29	波形合并命令 .....	50
3.30	模式选择命令 .....	50
3.31	MULTI-DEVICE SYNC .....	51
3.32	IQ 命令 .....	52
3.31.1	:IQ:CENterfreq .....	52
3.31.2	:IQ:SAMPlerate .....	52
3.31.3	:IQ:SYMBolrate .....	53
3.31.4	:IQ:AMPLitude .....	53
3.31.5	:IQ:IQADjustment:GAIN .....	53
3.31.6	:IQ:IQADjustment:IOFFset .....	54
3.31.7	:IQ:IQADjustment:QOFFset .....	54
3.31.8	:IQ:IQADjustment:QSKew .....	54
3.31.9	:IQ:TRIGger:SOURce .....	55
3.31.10	:IQ:WAVEload:BUILtin .....	55
3.31.11	:IQ:WAVEload:USERstored .....	56
3.31.12	:IQ:FrequencySampling .....	56
<b>4</b>	<b>编程示例 .....</b>	<b>57</b>
4.1	VISA 应用示例 .....	57
4.1.1	VC++ 示例 .....	57
4.1.2	VB 示例 .....	64
4.1.3	MATLAB 示例 .....	69
4.1.4	LabVIEW 示例 .....	71
4.1.5	Python 示例 .....	73
4.2	SOCKETS 应用示例 .....	76
4.2.1	Python 示例 .....	76
<b>5</b>	<b>索引 .....</b>	<b>79</b>

# 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

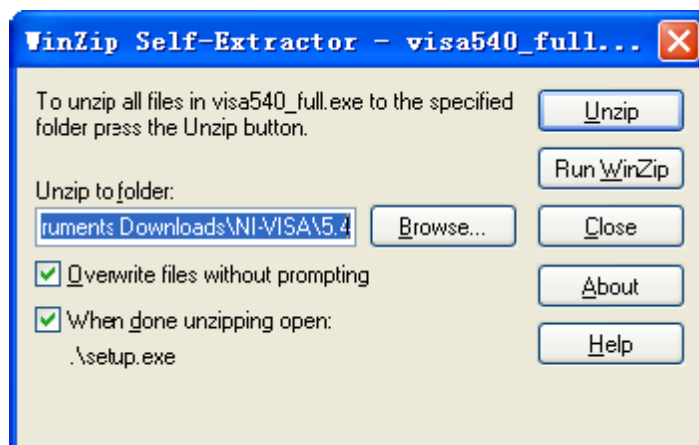
在编程之前，请确保正确安装 NI-VISA 软件的最新版本。

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-VISA 5.4 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：

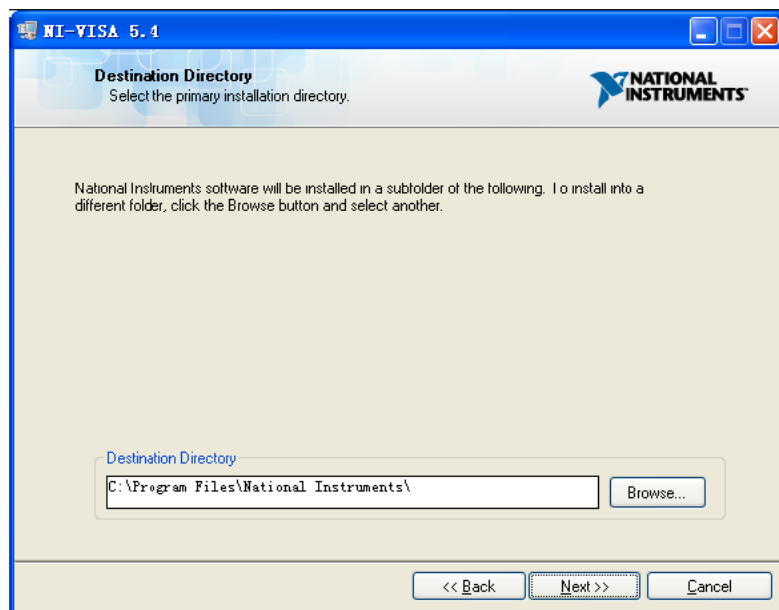


- c. 点击 Unzip 解压文件，当解压完成后，安装程序将自动执行。若你的计算机需要安

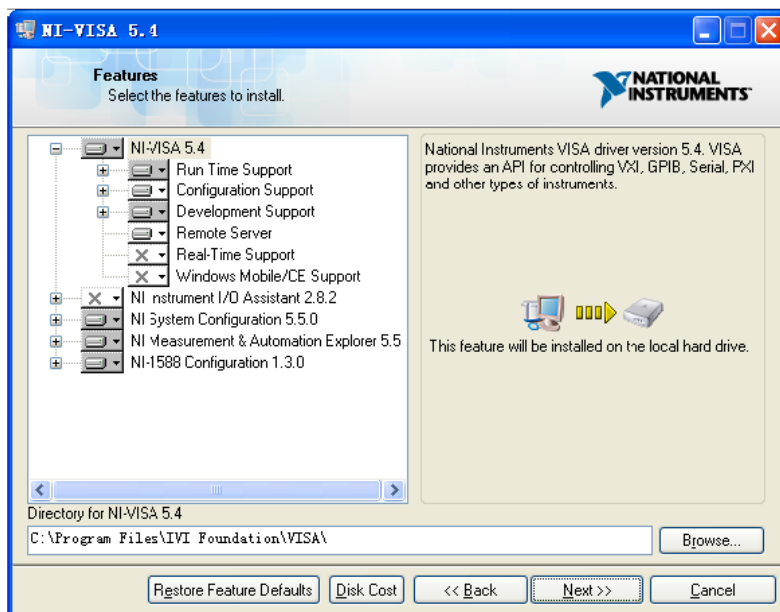
装.NET Framework4，则在安装过程会自动安装.NET Framework4。



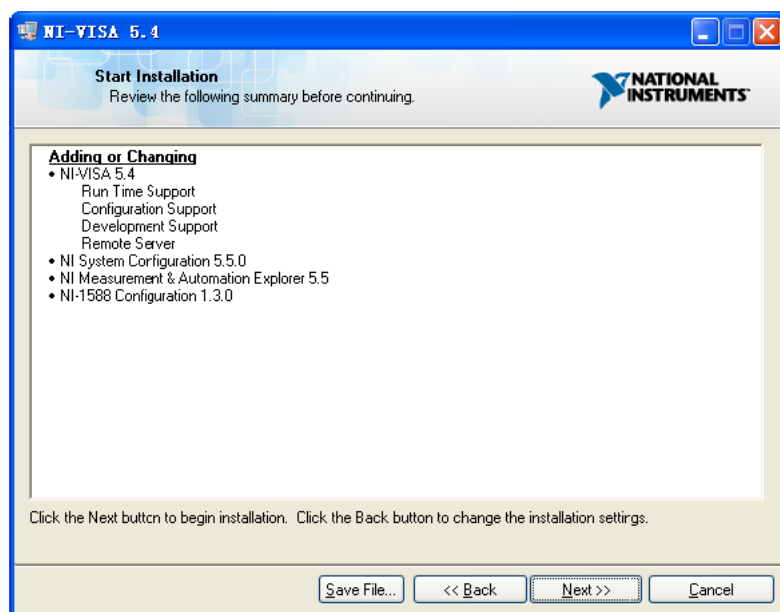
d. NI-VISA安装对话框如上图所示，点击Next开始安装过程。



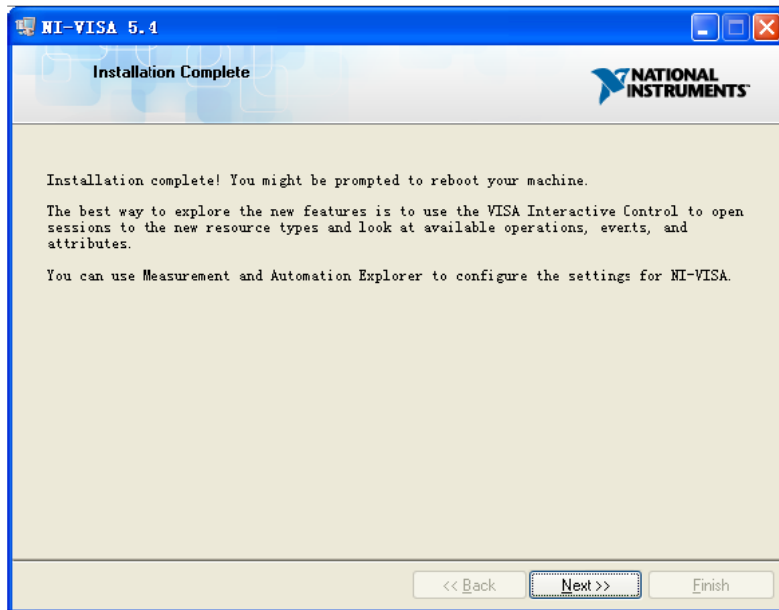
e. 设置安装路径，默认路径为“C:\Program Files\National Instruments\”。你也可以修改安装路径。点击Next，对话框如下图所示。



- 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：

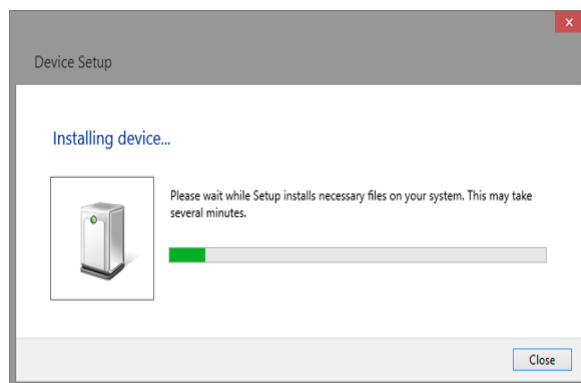


h. 安装完成后，重启电脑。

## 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。

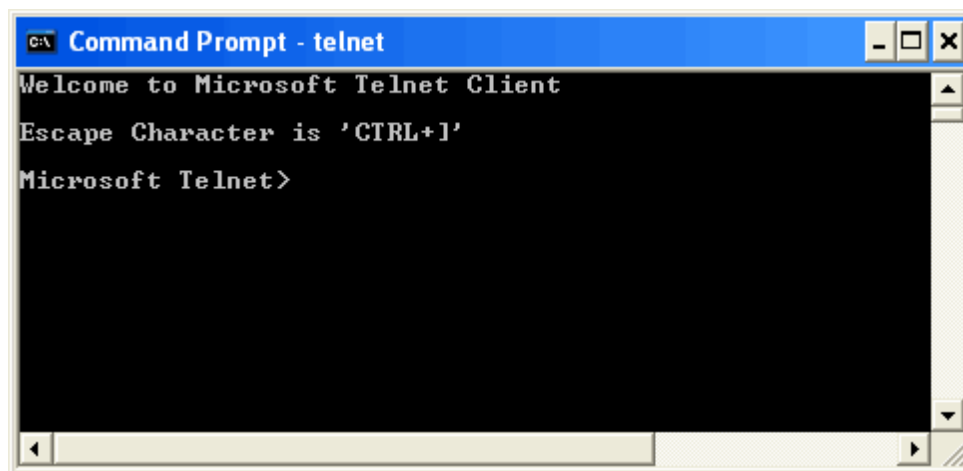
### 1.2.2 通过 NI-MAX 发送 SCPI 命令

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：

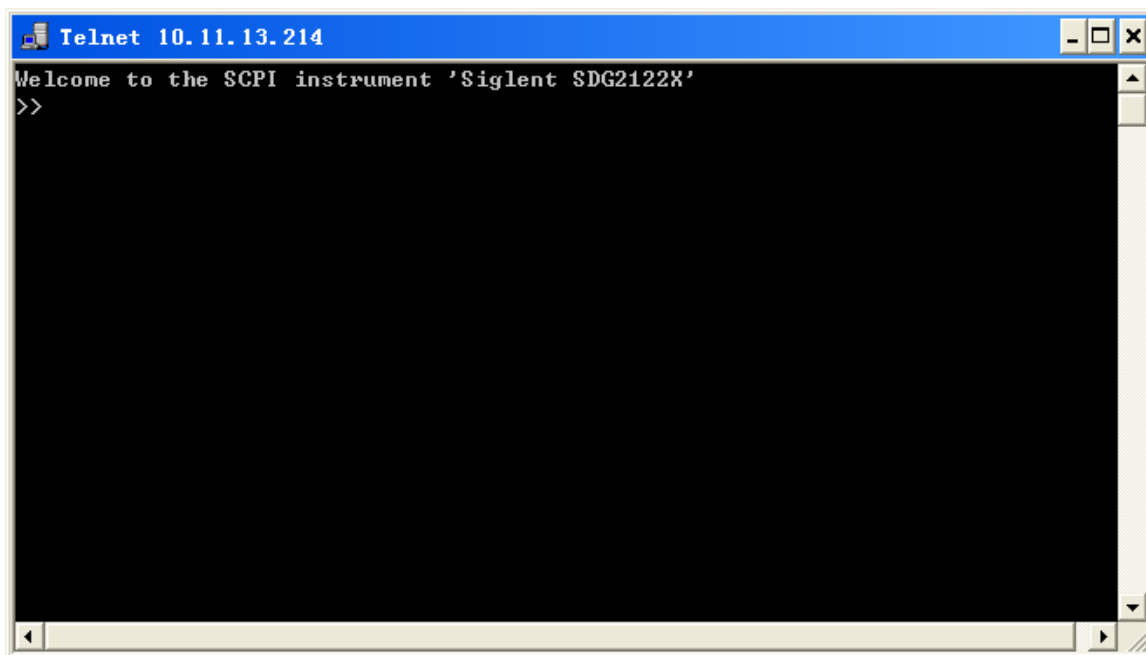
1. 在计算机桌面，点击开始>所有程序>附件>命令提示符
2. 在命令提示符窗口，输入 Telnet
3. 按下 Enter 键。将弹出 Telnet 显示窗



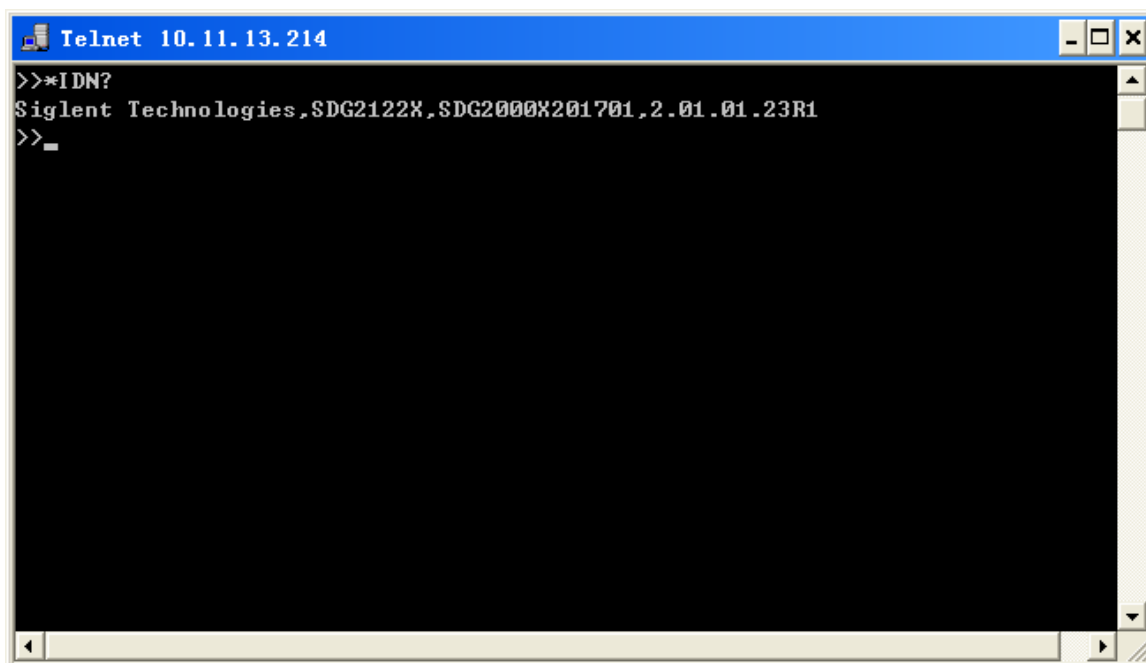
4. 在 Telnet 命令行，键入：

`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. 通过在提示符后键入 `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。

<b>SOCKET 地址</b>	IP 地址 + 端口号
<b>IP 地址</b>	SDG IP 地址
<b>端口号</b>	5025

相关内容，请查阅第 4.2 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 SDGxxxx 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
<a href="#">*IDN</a>	*IDN	系统	获取设备 id
<a href="#">*OPC</a>	*OPC	系统	设置或获取事件状态寄存器 (ESR) 的 OPC 位。

短格式	长格式	子系统	命令/查询的作用
<a href="#">*RST</a>	*RST	系统	设为出厂设置。
<a href="#">CHDR</a>	COMM_HEADER	信号	设置或获取命令格式
<a href="#">OUTP</a>	OUTPUT	信号	设置或获取输出状态。
<a href="#">BSWV</a>	BASIC_WAVE	信号	设置或获取基本波形参数。
<a href="#">MDWV</a>	MODULATEWAVE	信号	设置或获取调制参数。
<a href="#">SWWV</a>	SWEEPWAVE	信号	设置或获取扫频参数。
<a href="#">BTWV</a>	BURSTWAVE	信号	设置或获取脉冲串功能参数
<a href="#">PACP</a>	PARACOPY	信号	将参数从通道复制到另一通道
<a href="#">ARWV</a>	ARBWAVE	数据	设置任意波类型
<a href="#">SYNC</a>	SYNC	信号	设置或获取同步信号
<a href="#">NBFM</a>	NUMBER_FORM AT	系统	设置或获取数据格式
<a href="#">LAGG</a>	LANGUAGE	系统	设置或获取语言
<a href="#">SCFG</a>	SYS_CFG	系统	设置或获取上电开机模式
<a href="#">BUZZ</a>	BUZZER	系统	设置或获取蜂鸣器状态
<a href="#">SCSV</a>	SCREEN_SAVE	系统	设置或获取屏幕保护状态
<a href="#">ROSC</a>	ROSCILLATOR	信号	设置或获取时钟源的状态
<a href="#">FCNT</a>	FREQCOUNTER	信号	设置或获取频率计参数
<a href="#">INVT</a>	INVERT	信号	设置或获取当前通道的极性
<a href="#">COUP</a>	COUPLING	信号	设置或获取通道耦合参数
<a href="#">VOLTPRT</a>	VOLTPRT	系统	设置或获取过压保护的状态
<a href="#">STL</a>	STORELIST	信号	列出所有存储波形
<a href="#">WVDT</a>	WVDT	信号	设置或获取任意波波形数据
<a href="#">VKEY</a>	VIRTUALKEY	系统	设置虚拟键
<a href="#">SYST:CO MM:LAN:I PAD</a>	SYSTEM:COMMUNICATE:LAN:IPADDRESS	系统	设置或获取设备 IP 地址
<a href="#">SYST:CO MM:LAN:SMAS</a>	SYSTEM:COMMUNICATE:LAN:SUBNETMASK	系统	设置或获取设备子网掩码
<a href="#">SYST:CO MM:LAN:GAT AT</a>	SYSTEM:COMMUNICATE:LAN:GATEWAY	系统	设置或获取设备网关
<a href="#">SRATE</a>	SAMPLERATE	信号	设置或获取采样频率，仅在逐点输出（TrueArb）模式下使用。
<a href="#">HARM</a>	HARMonic	信号	设置或获取谐波信息
<a href="#">CMBN</a>	CoMBiNe	信号	设置或获取波形合并信息
<a href="#">MODE</a>	MODE	信号	设置或获取波形的相位模式
<a href="#">CASCADE</a>	CASCADE	系统	设置多设备同步
<a href="#">IQ:CENT</a>	IQ:CENTerfreq	信号	设置或获取 I/Q 调制的中心频率
<a href="#">IQ:SAMP</a>	IQ:SAMPlerate	信号	设置 I/Q 采样率

短格式	长格式	子系统	命令/查询的作用
<a href="#">IQ:SYMB</a>	IQ:SYMBOLrate	信号	设置 I/Q 符号率
<a href="#">IQ:AMPL</a>	IQ:AMPLitude	信号	设置 I/Q 幅度
<a href="#">IQ:IQAD:GAIN</a>	IQ:IQADjustment:GAIN	信号	在保持复合状态下调节 I 与 Q 比例
<a href="#">IQ:IQAD:IOFF</a>	IQ:IQADjustment:IOFFset	信号	调节 I 通道的偏置
<a href="#">IQ:IQAD:QOFF</a>	IQ:IQADjustment:QOFFset	信号	调节 Q 通道的偏置
<a href="#">IQ:IQAD:QSK</a>	IQ:IQADjustment:QSKew	信号	通过增大或减小 Q 的相位角来调节矢量 I 和矢量 Q 之间的相位角
<a href="#">IQ:TRIG:SOURCE</a>	IQ:TRIGGER:SOURCE	信号	设置 I/Q 的触发源。
<a href="#">IQ:WAVE:BUILD</a>	IQ:WAVEload:BUILDin	信号	从内建波形列表选择 I/Q 波形
<a href="#">IQ:WAVE:USER</a>	IQ:WAVEload:USERstored	信号	从用户保存波形列表选择 I/Q 波形。
<a href="#">:IQ:FrequencySampling</a>	:IQ:FrequencySampling	信号	设置 I/Q 采样率

## 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。显示内容包含厂商、机型、序列号、软件版本和硬件版本。
查询语法	*IDN?
响应格式	<p>格式 1: *IDN, &lt;设备 id&gt;,&lt;机型&gt;,&lt;序列号&gt;,&lt;软件版本&gt;,&lt;硬件版本&gt;.</p> <p>格式 2: &lt;设备商&gt;,&lt;机型&gt;,&lt;序列号&gt;,&lt;软件版本&gt;,&lt;硬件版本&gt;.</p> <p>&lt;设备 id&gt;:=“SDG”被用于识别仪器。.</p> <p>&lt;设备商&gt;:=“Siglent Technologies”.</p> <p>&lt;机型&gt;:=机器型号</p> <p>&lt;序列号&gt;:=每个产品有自己的编号，此序列号标注产品的唯一性。</p> <p>&lt;软件版本&gt;:=与软件版本信息相关的编号</p> <p>&lt;硬件版本&gt;:=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。</p>
示例	<p>读取版本信息。</p> <p>*IDN?</p> <p>返回值:</p> <p><i>Siglent Technologies,SDG6052X, SDG6XBAX1R0034, 6.01.01.28</i>(每个版本可能不同)</p>

备注:

- 下表显示不同系列的信号源上述命令的应答格式。

参数/命令	SDG800	SDG1000	SDG2000X	SDG5000	SDG1000X	SDG6000X /X-E
响应格式	格式 1	格式 1	格式 2	格式 1	格式 2	格式 2

- <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	格式 1: *OPC 1 格式 2: 1

备注：1. 下表显示不同系列的信号源上述命令的应答格式。

参数/命令	SDG800	SDG1000	SDG2000X	SDG5000	SDG1000X	SDG6000X /X-E
响应格式	格式 1	格式 1	格式 2	格式 1	格式 2	格式 2



### 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	将信号发生器重置成默认设置： <i>*RST</i>

## 3.2 通用头部命令

描述	此命令用于改变查询命令的返回格式。“SHORT”参数返回短格式。 “LONG”参数返回长格式。“OFF”不返回任何内容。
命令语法	CHDR(Comm_HeaDeR)<参数> <参数>:= {SHORT, LONG, OFF}
查询语法	Comm_HeaDeR?
响应格式	CHDR <参数>
示例	设置查询命令格式为长格式： <i>CHDR LONG</i>  读取查询命令格式： <i>CHDR?</i> 返回值： <i>COMM_HEADER LONG</i>

备注：下表显示不同系列的信号源上述命令的的可用性

参数/命令	SDG800	SDG1000	SDG2000X	SDG5000	SDG1000X	SDG6000X /X-E
CHDR	有	有	无	有	无	无

### 3.3 输出命令

描述	<p>启用和禁用通道对应前面板[OUTPUT]接口的输出。 查询结果返回“ON”、“OFF”、LOAD 和 PLRT 参数。</p>
命令语法	<p>&lt;通道&gt;:OUTPutON OFF,LOAD,&lt;负载&gt;,PLRT,&lt;极性&gt; &lt;通道&gt;:={C1,C2}. &lt;负载&gt; := {见备注}. 默认单位为 ohm。 &lt;极性&gt; := {NOR, INVT},其中 NOR 代表正常, INVT 代表反相。</p> <p>&lt;噪声叠加&gt;: C1:NOISE_ADD STATE, ON OFF ,RATIO,&lt;信噪比的值&gt;或者C1:NOISE_ADD STATE, ON OFF,RATIO_DB,&lt;噪声dB值&gt; &lt;信噪比的值&gt;:= {2.1-100000000} &lt;噪声dB值&gt;:= {3.24886-80}, 默认单位为dB</p> <p>&lt;幅度限制&gt;:C1:BSWVsMAX_OUTPUT_AMP, &lt;幅度限制&gt; &lt;幅度限制&gt;:={1-20}, 最大输出幅度 VPP 值</p>
查询语法	<p>&lt;通道&gt;:OUTPut? &lt;噪声叠加&gt;:C1:NOISE_ADD? &lt;幅度限制&gt;:C1:BSWV?</p>
响应格式	<p>&lt;channel&gt;:OUTP ON OFF,LOAD,&lt;load&gt;,PLRT, &lt;polarity&gt; &lt;channel&gt;:NOISE_ADDSTATE, ON OFF,RATIO,&lt; S/N &gt; , RATIO_DB,&lt; S/N (dB)&gt; &lt;channel&gt;:BSWV WVTP,&lt;type&gt;,FRQ, &lt;frequency&gt;,PERI, &lt;period&gt;,AMP,&lt;amplitude&gt;, AMPVRMS, &lt;Amplitude&gt;,MAX_OUTPUT_AMP, &lt;Amplitude&gt;,OFST, &lt;offset&gt;, HLEV, &lt;high level&gt;, LLEV, &lt;low level&gt;, PHSE, &lt;phase&gt;</p>
示例	<p>打开 CH1: <i>C1:OUTP ON</i></p> <p>读取 CH1 输出状态: <i>C1:OUTP?</i> 返回值: <i>C1:OUTP ON,LOAD,HZ,PLRT,NOR</i></p> <p>设置 CH1 的负载为 50ohm: <i>C1:OUTP LOAD,50</i></p>

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**

设置 CH1 的噪声叠加打开并设置信噪比为 120:

**C1:NOISE\_ADD STATE,ON,RATIO,120**

设置 CH1 的最大输出幅度为 5V:

**C1:BSWVsMAX\_OUTPUT\_AMP,5**

备注: 下表显示不同系列的信号源上述命令的可用性

参数/命令	SDG800	SDG1000	SDG2000X	SDG5000	SDG1000X	SDG6000X /X-E
<channel>	无	有	有	有	有	有
LOAD	50, HZ	50~10000, HZ	50~100000 , HZ	50, HZ	50~100000 , HZ	50~100000 , HZ

\* "HZ"代表高阻态

## 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>:= {下表参数}.  
 <值>:= {相关参数的值}.

参数	值	描述
WVTP	<type>	:= {SINE, SQUARE, RAMP, PULSE, NOISE, ARB,DC, PRBS,IQ}。如果命令未设置基本波形类型, WVTP 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为噪声和直流, 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 为噪声和直流, 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>	:= 脉冲延时。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。
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 = $2^{\text{长度}} - 1$ 。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	<prbs Logic level>	:= {TTL_CMOS, LVTTTL_LVCMOS, ECL, LVPECL, LVDS}。PRBS: 逻辑电平, 设置幅度和偏移。仅当 WVTP 是 PRBS 时，才能设置该参数。
AMPVRMS	<amplitude>	:= amplitude. 单位是 V <sub>rms</sub> 。

AMPDBM	<amplitude>	:= amplitude. 单位是 dBm
--------	-------------	-----------------------

## 查询语法

&lt;通道&gt;: BaSic\_WaVe?

&lt;通道&gt;:={C1, C2}.

## 响应格式

&lt;通道&gt;:BSWV&lt;参数&gt;

&lt;参数&gt; :={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,100HZ,PERI,0.01S,AMP,2V,  
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*

设置 C1 的输出电压幅值为 3dBm:

*C1:BSWV AMPDBM,3*

设置逻辑电平为 TTL\_CMOS:

*C1:BSWV LOGICLEVEL,TTL\_CMOS*

备注:

1. 下表显示不同系列的信号源上述命令的可用性

Parameter /command	SDG800	SDG1000	SDG2000 X	SDG5000	SDG1000 X	SDG6000 X	SDG6000 X-E
<channel>	无	有	有	有	有	有	有
RISE	有	无	有	有	有	有	有
FALL	有	无	有	有	有	有	有
DLY	无	有	有	有	有	有	有
BANDSTATE	无	无	有	无	无	有	有
BANDWIDTH	无	无	有	无	无	有	有
LENGTH	无	无	无	无	无	有	无

EDGE	无	无	无	无	无	有	无
DIFFSTATE	无	无	无	无	无	有	无
BITRATE	无	无	无	无	无	有	无
LOGICLEVEL	无	无	无	无	无	有	无
AMPDBM	无	无	有	无	有	有	有

2. 在 SDG1000X, 若波形合并打开, WVTP 不能设置为方波。

### 3.5 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
STATE	<state>	:= {ON (打开), OFF (关闭)}. 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 (内触发), EXT (外触发)}. AM 触发源。
AM, MDSP	<mod wave shape>	:= {SINE, SQUARE, TRIANGLE, UPRAMP, DNRAMP, NOISE, ARB}. AM 调制波形。仅当触发源设置为内触发时, 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。触发源设置为内触发时, 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 (内触发), EXT (外触发)}. DSBAM 触发源。
DSBAM, MDSP	<mod wave shape>	:= {SINE, SQUARE, TRIANGLE, UPRAMP, DNRAMP, NOISE, ARB}. DSB AM 调制波形。触发源设置为内触发时, 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。设置为内触发

		时, 该参数才能设置。
FM, SRC	<src>	:= {INT (内触发), EXT (外触发)}。FM 触发源
FM, MDSP	<mod wave shape>	:= {SINE, SQUARE, TRIANGLE, UPRAMP, DNRAMP, NOISE, ARB}。 FM 调制波形。触发源设置为内触发时, 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。触发源设置为内触发时, 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。触发源设置为内触发时, 该参数才能设置。
PM, SRC,	<src>	:= {INT (内触发), EXT (外触发)}。PM 触发源。
PM, MDSP	<mod wave shape>	:= {SINE, SQUARE, TRIANGLE, UPRAMP, DNRAMP, NOISE, ARB}。 PM 调制波形。触发源设置为内触发时, 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。触发源设置为内触发时, 该参数才能设置。
PM, DEVI	<PM phase offset>	:= {0 至 360}。PM 相位偏差。单位是 ‘°’。触发源设置为内触发时, 该参数才能设置。
PWM, SRC	<src>	:= {INT (内触发), EXT (外触发)}。PWM 触发源。
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。触发源设置为内触发时, 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。触发源设置为内触发时, 该参数才能设置。
ASK, SRC	<src>	:= {INT (内触发), EXT (外触发)}。ASK 触发源。
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。触发源设置为内触发时, 该参数才能设置。
FSK, SRC	<src>	:= {INT (内触发), EXT (外触发)}。FSK 触发源。

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。触发源设置为内触发时，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 触发源。
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。触发源设置为内触发时，该参数才能设置。
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：

- 1.如果载波设为噪声，将不能打开调制模式。
- 2.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>:MoDulateWaVe? <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 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,AMPV  
RMS,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.备注：下表显示不同系列的信号源上述命令的的可用性

参数/命令	SDG800	SDG1000	SDG2000X	SDG5000	SDG1000X	SDG6000X /X-E
<channel>	无	有	有	有	有	有
<type>, SRC	无	有	有	有	有	有
CARR, DLY	无	有	有	有	有	有
CARR, RISE	有	无	有	有	有	有
CARR, FALL	有	无	有	有	有	有

<类型>:={AM,FM,PM,FSK,ASK,PSK, DSBAM,PWM}.

## 3.6 扫描波形命令

**描述** 该命令用于设置或者获取扫描波形的参数。

**命令语法**

```
<通道>:SweepWaVe <参数>,<值>
<通道>:={C1, C2}
<参数>:= {下表的参数}
<值>:={相关的值}
```

参数	值	描述
STATE	<state>	:= {ON (打开), OFF (关闭)}。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’。
STOP	<stop_freq>	:= 终止频率。与基本波形频率相同。单位是赫兹 ‘Hz’。
SWMD	<sweep_mode>	:= {LINE (线性), LOG (对数)}, 其中 LINE 代表线性扫频和 LOG 代表对数扫频。
DIR	<direction>	:= {UP (向上), DOWN (向下)}。扫描方向
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 和 MAN 时有效。
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 ‘%’。

MARK_STATE	<state>	:= { ON, OFF}.
MARK_FREQ	<frequency>	:= 频率标记。单位是赫兹“Hz”。范围在起始频率到终止频率之间

查询语法 <通道>:SWeepWaVe?

<通道>:={C1, C2}.

响应格式 <通道>:SWWV <参数>

<参数> :={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 的扫描时间为 1 秒:

*C1:SWWV TIME,1*

设置 CH1 的终止时间为 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,100HZ,START,100HZ,TRSR,MAN,TRMD,OFF,SWMD,LINE,DIR,UP,CARR,WVTP,SQUARE,FRQ,1000HZ,AMP,4V,OFST,0V,DUTY,50,PHSE,0*

读取状态为 OFF 的 CH2 的扫描参数:

*C2:SWWV?*

返回值:

*C2:SWWV STATE,OFF*

设置 C1 通道扫描状态的频率标记为 1kHz

*C1:SWWV MARK\_STATE,ON,MARK\_FREQ,1000*

备注：下表显示不同系列的信号源上述命令的的可用性

参数/命令	SDG800	SDG1000	SDG2000X	SDG5000	SDG1000X	SDG6000X /X-E
<channel>	无	有	有	有	有	有
TRMD	无	有	有	有	有	有
EDGE	无	有	有	有	有	有

## 3.7 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>:BurstTWaVe <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
STATE	<state>	:= {ON (打开), OFF (关闭)}。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。在下列情况下, 该值无效: <ul style="list-style-type: none"> <li>在载波为噪声</li> <li>GATE_NCYC 是门控 (“X”系列除外)</li> <li>TRSR 是外触发</li> </ul>
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。
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

&lt;通道&gt;:BTWV(BurstWaVe)?

&lt;通道&gt;:={C1, C2}

## 响应格式

&lt;通道&gt;:BTWV &lt;参数&gt;

&lt;参数&gt;:={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 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**

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*

备注：下表显示不同系列的信号源上述命令的的可用性

参数/命令	SDG800	SDG1000	SDG2000X	SDG5000	SDG1000X	SDG6000X /X-E
<channel>	无	有	有	有	有	有
TRMD	无	有	有	有	有	有
EDGE	无	有	有	有	有	有
CARR, DLY	有	有	有	有	有	有
CARR, RISE	有	无	有	有	有	有
CARR, FALL	有	无	有	有	有	有

## 3.8 参数复制命令

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*

备注：下表显示不同系列的信号源上述命令的的可用性

参数/命令	SDG800	SDG1000	SDG2000X	SDG5000	SDG1000X	SDG6000X /X-E
PACP	无	有	有	有	有	有

## 3.9 任意波形命令

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	<通道>:ArbWaVe INDEX,<索引号> <通道>:ArbWaVe NAME,<名称> <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
.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： <i>C1:ARWV INDEX,2</i>  读取 CH1 的当前波形 <i>C1:ARWV?</i> 返回值： <i>C1:ARWV INDEX,2,NAME,StairUp</i>  通过波形名称设置 CH1 的波形为 wave_1 <i>C1:ARWV NAME, wave_1</i>
相关命令	<a href="#">STL</a>

索引号	名称	索引号	名称	索引号	名称	索引号	名称
0	Sine	12	Logfall	24	Gmonopuls	36	Triang
1	Noise	13	Logrise	25	Tripuls	37	Harris
2	StairUp	14	Sqrt	26	Cardiac	38	Bartlett
3	StairDn	15	Root3	27	Quake	39	Tan
4	Stairud	16	X^2	28	Chirp	40	Cot
5	Ppulse	17	X^3	29	Twotone	41	Sec
6	Npulse	18	Sinc	30	Snr	42	Csc
7	Trapezia	19	Gaussian	31	Hamming	43	Asin
8	Upramp	20	Dlorentz	32	Hanning	44	Acos
9	Dnramp	21	Haversine	33	Kaiser	45	Atan
10	Exp_fall	22	Lorentz	34	Blackman	46	Acot

11	Exp_rise	23	Gauspuls	35	Gausswin	47	Square
----	----------	----	----------	----	----------	----	--------

备注:

1、此表仅仅是一个示例，波形的索引号取决于使用的机型，可用“STL?”命令获取索引和名称之间的准确映射关系。

备注：下表显示不同系列的信号源上述命令的的可用性

参数/命令	SDG800	SDG1000	SDG2000X	SDG5000	SDG1000X	SDG6000X /X-E
<channel>	无	有	有	有	有	有
INDEX	有	有	有（仅有内建波形）	有	有（仅有内建波形）	有（仅有内建波形）
NAME	有	有	有（仅有用户定义波形）	有	有（仅有用户定义波形）	有（仅有用户定义波形）

## 3.10 同步命令

### 描述

此命令用于设置同步信号。

### 命令语法

<通道>:SYNC <状态>  
 <通道>:={C1,C2}.  
 <状态>:={ON,OFF}.  
 SYNC TYPE, < 类型 >  
 < 类型 >:={CH1,CH2,MOD\_CH1,MOD\_CH2}.

### 查询语法

<通道>:SYNC?  
 <通道>:={C1, C2}.

### 响应格式

<通道>:SYNC <状态>

### 示例

打开 CH1 的同步功能并设置：  
**C1:SYNC ON,TYPE,MOD\_CH1**

读取 CH1 的同步功能状态

**C1:SYNC?**

返回值：

**C1:SYNC ON,TYPE,MOD\_CH1**

备注：下表显示不同系列的信号源上述命令的的可用性

Parameter /command	SDG800	SDG1000	SDG2000X	SDG5000	SDG1000X	SDG6000X /X-E
--------------------	--------	---------	----------	---------	----------	---------------



SYNC	无	有	有	有	有	有
------	---	---	---	---	---	---

### 3.11 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	<pre>NumBer_ForMatPNT,&lt;pnt&gt; NumBer_ForMa SEPT,&lt;sept&gt; &lt;pnt&gt;:={Dot (点), Comma (逗号)}. 小数点格式 &lt;sept&gt;:={Space (空格), Off (关闭), On (打开)}. 分隔符格式</pre>
查询语法	NBFM?
响应格式	NBFM PNT,<pnt>, SEPT,<sept>
示例	<p>设置小数点格式为点: <i>NBFM PNT, DOT</i></p> <p>设置分割符号为打开: <i>NBFM SEPT, ON</i></p> <p>读取数字格式: <i>NBFM?</i> 返回值 <i>NBFM PNT, DOT, SEPT, ON</i></p>

### 3.12 语言命令

描述	此命令设置或者获取系统语言。
命令语法	<pre>LAnGuaGe&lt;语言&gt; &lt;语言&gt;:={EN, CH, RU}, 其中 EN 是英语, CH 是中文, 以及 RU 是俄语。</pre>
查询语法	LAnGuaGe?
响应格式	LAGG <语言>
示例	<p>设置语言为英语: <i>LAGG EN</i></p>

读取当前系统语言：

*LAGG?*

返回值：

*LAGG EN*

备注：下表显示不同系列的信号源上述命令的的可用性

参数/命令	SDG800	SDG1000	SDG2000X	SDG5000	SDG1000X	SDG6000X /X-E
RU	无	有	无	无	无	无

### 3.13 配置命令

描述	设置或者获取上电开机模式
命令语法	<p>Sys_CFG&lt;模式&gt;</p> <p>&lt;模式&gt;:= {DEFAULT（默认）, LAST（上次）}</p>
查询语法	Sys_CFG?
响应格式	SCFG<模式>
示例	<p>设置上电开机系统为上次：</p> <p><i>SCFG LAST</i></p>

### 3.14 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	<p>BUZZer&lt;状态&gt;</p> <p>&lt;状态&gt;:= {ON, OFF}.</p>
查询语法	BUZZer?
响应格式	BUZZ<状态>
示例	<p>打开蜂鸣器：</p> <p><i>BUZZ ON</i></p>

## 3.15 屏幕保存命令

描述	此命令用于关闭或者设置屏幕保存时间（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	设置屏幕保存时间为 5 分钟： <i>SCSV 5</i>
	读取当前的屏幕保存时间： <i>SCreen_SaVe?</i> 返回值： <i>SCSV 5MIN</i>

## 3.16 时钟源命令

描述	此命令设置或者获取时钟源。
命令语法	ROSCillator <src> <src>:= {INT（内部时钟）， EXT（外部时钟）} ROSC 10MOUT,ON OFF
查询语法	ROSCillator?
响应格式	ROSC <src>,10MOUT, ON OFF
示例	设置内部时基作为时钟源： <i>ROSC INT</i>
	Set 10M output: <i>ROSC 10MOUT,ON</i>

备注：下表显示不同系列的信号源上述命令的的可用性

参数/命令	SDG800	SDG1000	SDG2000X	SDG5000	SDG1000X	SDG6000X /X-E
ROSC	无	有	有	有	有	有

## 3.17 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’。不能设置。
PW	<pos_width>	测量正脉宽。单位是秒 ‘s’，不能设置。
NW	<neg_width>	测量负脉宽。单位是秒 ‘s’，不能设置。
DUTY	<duty>	测量占空比。单位是 ‘%’，不能设置。
FRQDEV	<freq_dev>	测量频率偏差。单位是 ‘ppm’，不能设置。
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

**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*

备注：下表显示不同系列的信号源上述命令的的可用性

参数/命令	SDG800	SDG1000	SDG2000X	SDG5000	SDG1000X	SDG6000X /X-E
FCNT	无	有	有	有	有	有

### 3.18 反相命令

**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

备注：下表显示不同系列的信号源上述命令的的可用性

参数/命令	SDG800	SDG1000	SDG2000X	SDG5000	SDG1000X	SDG6000X /X-E
<channel>	无	有	有	有	有	有

### 3.19 通道耦合命令

**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_enable>	:={ON, OFF}

		通道跟踪状态
STATE	<state>	:= {ON, OFF} 通道耦合状态
BSCH	<bsch>	:= {CH1, CH2} 基本通道
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’
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’。

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*

备注：下表显示不同系列的信号源上述命令的的可用性

参数/命令	SDG800	SDG1000	SDG2000X	SDG5000	SDG1000X	SDG6000X /X-E
COUP	无	有	有	有	有	有
TRACE	无	无	有	无	有	有

STATE	无	有	无	有	无	无
BSCH	无	有	无	有	无	无
FCOUP	无	无	有	无	有	有
FRAT	无	无	有	无	有	有
PCOUP	无	无	有	无	有	有
PRAT	无	无	有	无	有	有
ACOUP	无	无	有	无	有	有
ARAT	无	无	有	无	有	有
ADEV	无	无	有	无	有	有

## 3.20 过压保护命令

**描述** 此命令用于设置或者获取过压保护状态。

**命令语法** VOLT`PRT`<状态>  
<状态>:= {ON,OFF}

**查询语法** VOLT`PRT`?

**响应格式** VOLT`PRT`<状态>

备注：下表显示不同系列的信号源上述命令的的可用性

参数/命令	SDG800	SDG1000	SDG2000X	SDG5000	SDG1000X	SDG6000X /X-E
VOLT <code>PRT</code>	无	有	有	无	有	有

## 3.21 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? BUILDIN|USER

**示例** 读取保存在 SDG1000 所有任意波数据：

*STL?*

返回值：

*STL M0, sine, M1, noise, M2, stairup, M3, staidn, M4, stairud, M5, ppulse, M6, npulse, M7, trapezia, M8, upramp, M9, dn ramp,*

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

读取保存在 SDG2000X 中的内建波形数据:

STL? BUILDIN

返回值:

STL 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, Diorentz, 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*

读取来自 SDG1000X 的用户定义的波形数据：

**STL? USER**

返回值：

*STLWVNM,sinc\_8M,sinc\_3000000,sinc\_1664000,  
ramp\_8M,sinc\_2000000,sinc\_50000,square\_8M,sinc\_5000,  
wave1,square\_1M*

备注：下表显示不同系列的信号源上述命令的的可用性

参数/命令	SDG800	SDG1000	SDG2000X	SDG5000	SDG1000X	SDG6000X /X-E
STL? return	内建波形 用户定义 波形	内建波形 用户定义 波形	内建波形	内建波形 用户定义 波形	内建波形	内建波形
BUILDIN	无	无	有	无	有	有
USER	无	无	有	无	有	有

## 3.22 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 <索引号>,<参数>,<值>  
 <通道>:={C1, C2}.  
 <索引号>:={Mn}。波形索引号。详细内容请查阅备注 2。  
 <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
TYPE	<类型>	:= {0 to 5} 0 –通用 1 –数学 2 –工程 3 –窗口 4 –三角函数 5 –用户定义 "X"系列该参数无效。
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 需要从一个波形文件中读取波形数据。

**查询语法** 格式 1: WVDT? Mn

格式 2: WVDT? USER,<wave\_name>  
 <波形名称>:={用户定义的波形名称}.

**示例** 示例在 4.1.5 节处。

**备注:**

1. 下表显示不同系列的信号源上述命令的的可用性。

参数/命令	SDG800	SDG1000	SDG2000X	SDG5000	SDG1000X	SDG6000X-E	SDG6000X
TYPE	有	有	无	有	无	无	无
<length>	32KB	32KB	4B~16MB	32KB,	4B~16MB	4B~16MB	4B~40MB

				1024KB			
USER	无	无	有	无	有	有	有
Format of WVDT?	格式 1	格式 1	内建: 格式 1 用户定义: 格式 2	格式 1	内建: 格式 1 用户定义: 格式 2	内建: 格式 1 用户定义: 格式 2	内建: 格式 1 用户定义: 格式 2

2. 下表展示了每个 SDG 系列的 Mn 参数的详细信息。

机型	Mn 参数的描述
SDG800	0<=n<=59. M0~M49: 内建(32KB). M50~M59: 用户定义(32KB).
SDG1000	0<=n<=59. M0~M49: 内建(32KB). M50~M59:用户定义(32KB).
SDG2000X	0<=n<=196. M0~M196: 内建(32KB). 发送波形数据时不需要设置该参数。
SDG5000	0<=n<=68. M0~M35: 内建(32KB). M36~M59:用户定义(32KB). M60~M67:用户定义(1024KB).
SDG1000X	0<=n<=196. M0~M196: 内建(32KB). 发送波形数据时不需要设置该参数。
SDG6000X/X-E	0<=n<=196. M0~M196: 内建(32KB). 发送波形数据时不需要设置该参数。

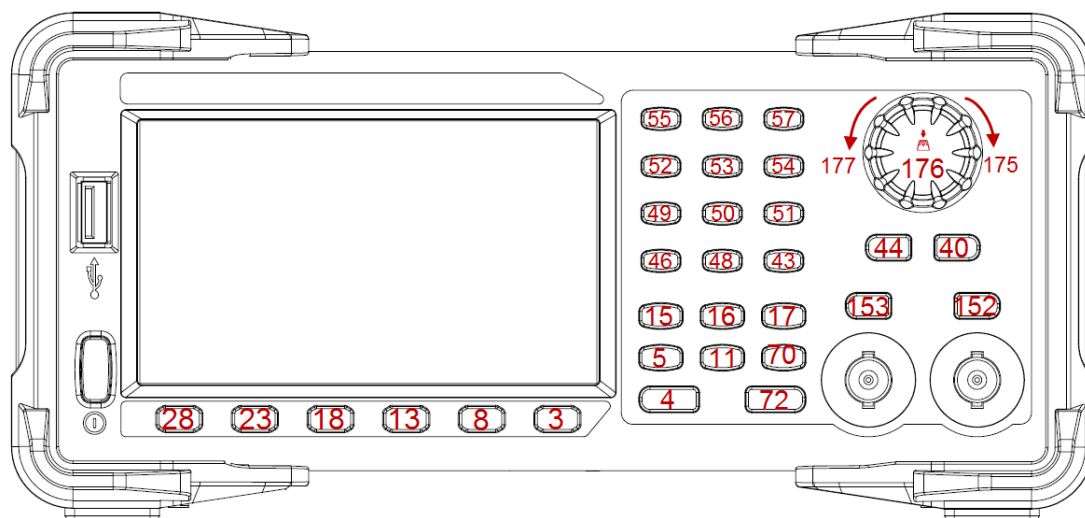
## 3.23 虚拟键命令

**描述** 此命令用于模拟前面板的按键的状态

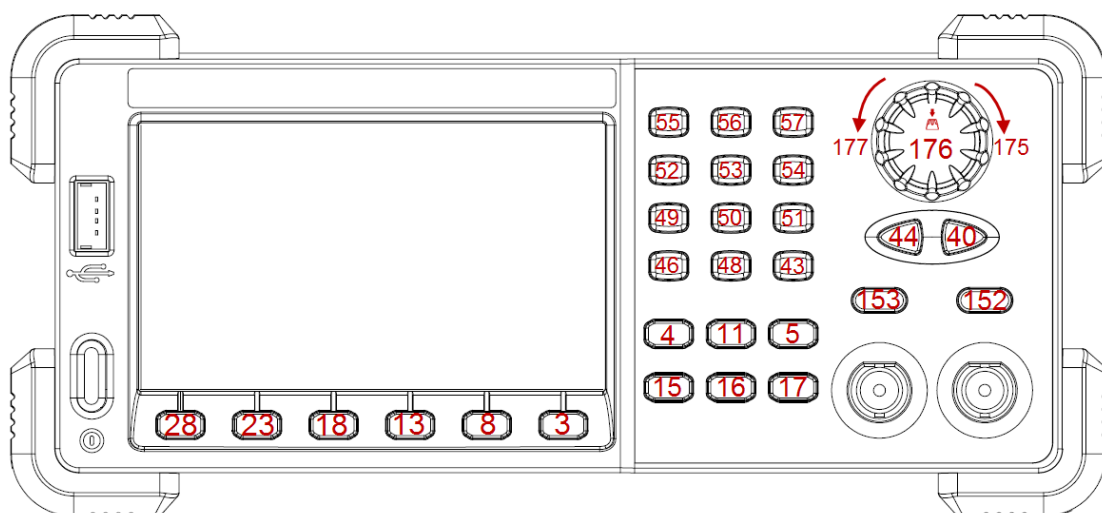
**命令语法** VirtualKEY VALUE,<值>,STATE,<状态>  
 <值>:= {下表显示虚拟按键的索引号或者名称}.  
 <状态>:={0,1},“1”是有效的虚拟值,即按下前面板的对应按键;而“0”是无效值。

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

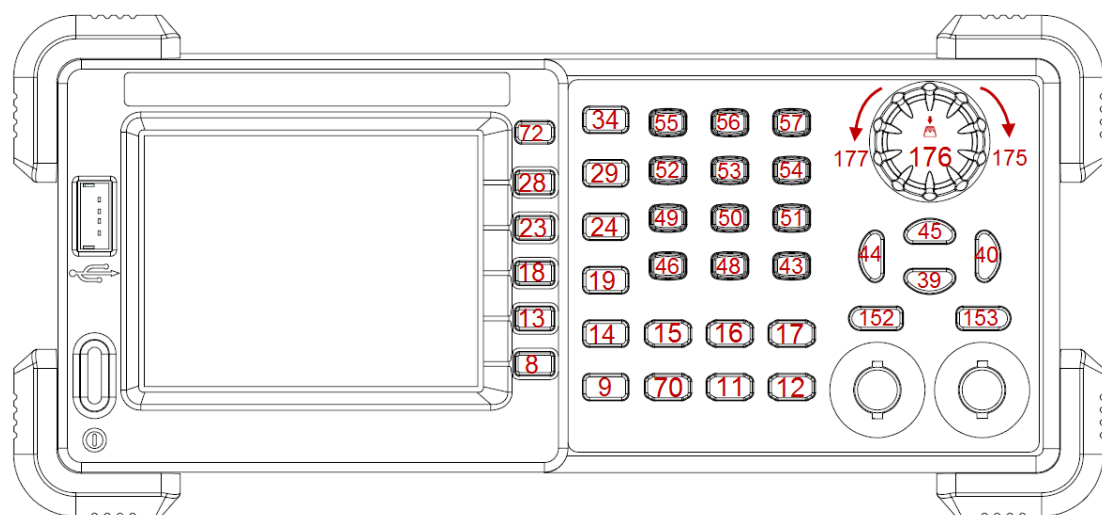
KB_FUNC5	8	KB_NUMBER_8	56
KB_FUNC6	3	KB_NUMBER_9	57
KB_SINE	34	KB_POINT	46
KB_SQUARE	29	KB_NEGATIVE	43
KB_RAMP	24	KB_LEFT	44
KB_PULSE	19	KB_RIGHT	40
KB_NOISE	14	KB_UP	45
KB_ARB	9	KB_DOWN	39
KB_MOD	15	KB_OUTPUT1	153
KB_SWEEP	16	KB_OUTPUT2	152
KB_BURST	17	KB_KNOB_RIGHT	175
KB_WAVES	4	KB_KNOB_LEFT	177
KB_UTILITY	11	KB_KNOB_DOWN	176
KB_PARAMETER	5	KB_HELP	12
KB_STORE_RECALL	70	KB_CHANNEL	72
KB_NUMBER_0	48		
KB_NUMBER_1	49		
KB_NUMBER_2	50		
KB_NUMBER_3	51		



SDG1000X/SDG2000X/SDG6000X/SDG6000X-E 上的按键和指示



SDG5000 上的按键和指示



SDG800/SDG1000 上的按键和指示

示例 `VKEY VALUE,15,STATE,1`

`VKEY VALUE,KB_SWEEP,STATE,1`

备注：下表显示不同系列的信号源上述命令的的可用性。

参数/命令	SDG800	SDG1000	SDG2000X	SDG5000	SDG1000X	SDG6000X /X-E
KB_FUNC6	无	无	有	有	有	有
KB_STORE_RECAL L	有	有	有	无	有	有
KB_HELP	有	有	无	无	无	无
KB_CHANNEL	无	有	有	无	有	有
KB_SINE	有	有	无	无	无	无

KB_SQUARE	有	有	无	无	无	无
KB_RAMP	有	有	无	无	无	无
KB_PULSE	有	有	无	无	无	无
KB_NOISE	有	有	无	无	无	无
KB_ARB	有	有	无	无	无	无
KB_UP	有	有	无	无	无	无
KB_DOWN	有	有	无	无	无	无

## 3.24IP 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”

备注：下表显示不同系列的信号源上述命令的的可用性。

参数/命令	SDG800	SDG1000	SDG2000X	SDG5000	SDG1000X	SDG6000X /X-E
SYST:COMM:LAN:IPAD	无	无	有	无	有	有

## 3.25 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”

备注：下表显示不同系列的信号源上述命令的的可用性。

参数/命令	SDG800	SDG1000	SDG2000X	SDG5000	SDG1000X	SDG6000X /X-E
SYST:COMM:LAN:SMAS	无	无	有	无	有	有

## 3.26 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	<p>设置网关为 10.11.13.5: SYSTem:COMMunicate:LAN:GATeway "10.11.13.5"</p> <p>获取网关: SYSTem:COMMunicate:LAN:GATeway?</p> <p>返回值: "10.11.13.5"</p>

备注：下表显示不同系列的信号源上述命令的的可用性。

参数/命令	SDG800	SDG1000	SDG2000X	SDG5000	SDG1000X	SDG6000X /X-E
SYST:COMM:LAN:GAT	无	无	有	无	有	有

## 3.27 采样率命令

**描述** 设置或者获取任意波模式、采样率和插值方法。采样率和插值方法仅能在逐点输出模式下设置。

**命令语法** <通道>:SampleRATE MODE,<模式>,VALUE, <采样率>,INTER,<插值>

<通道> :=<C1, C2>.

<模式> :={DDS, TARB}, 其中 TARB 是逐点输出模式。

<采样率> :=采样率。单位是 Sa/s。

<插值>:={LINE, HOLD ,SINC, SINC27, SINC13}, 其中 LINE 是线性插值, 而 HOLD 是零阶保持。

**查询语法** <通道>:SRATE?

**示例** 获取 CH1 的采样率:

C1:SRATE?

返回值:

C1:SRATE MODE,DDS

设置 CH1 为逐点输出模式:

C1:SRATE MODE,TARB

设置 CH1 的采样率为 1000000Sa/s:



*C1:SRATE VALUE,1000000*

设置 CH1 为逐点输出模式且插值模式为 SINC13

*C1:SRATE MODE,TARB,INTER,SINC13*

备注：下表显示不同系列的信号源上述命令的的可用性。

参数/命令	SDG800	SDG1000	SDG2000X	SDG5000	SDG1000X	SDG6000X /X-E
SRATE	无	无	有	无	无	有
INTER	无	无	无	无	无	有

## 3.28 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>:HARMonicHARMSTATE,<状态>,HARMTYPE,  
<类型>,<级数>,<单位>,<值>, HARMPHASE,<相位>

<状态>:= <ON, OFF>.  
<类型>:= <EVEN (偶级数), ODD (奇级数), ALL (所有级数)>.  
<级数>:= {1,2,...,M}, 其中 M 是支持的最大级数。  
<单位> :=< HARMAMP, HARMDBC>.  
<值>:=指定谐波的幅值。值的范围取决于机型。当<单位>= HARMAMP, 单位是伏特, 峰峰值“Vpp”, 而当<单位>= HARMDBC, 单位是“dBc”。  
<相位>:= {0~360},单位是“度”。

**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,H  
ARMAMP,2.004748935V,HARMDBC,-6dBc,HARMPHASE,0*

备注：下表显示不同系列的信号源上述命令的的可用性。

参数/命令	SDG800	SDG1000	SDG2000X	SDG5000	SDG1000X	SDG6000X /X-E
HARM	无	无	有	无	有	有

### 3.29 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*

备注：下表显示不同系列的信号源上述命令的的可用性。

参数 /命令	SDG800	SDG1000	SDG2000X	SDG5000	SDG1000X	SDG6000X /X-E
CMBN	无	无	有	无	有	有

### 3.30 模式选择命令

**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），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*

备注：下表显示不同系列的信号源上述命令的的可用性。

参数 /命令	SDG800	SDG1000	SDG2000X	SDG5000	SDG1000X	SDG6000X /X-E
MODE	无	无	有	无	有	有

### 3.31 Multi-Device Sync

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 STATE,ON|OFF,MODE, <模式>,DELAY, <延时>  
                                 <模式>:={MASTER,SLAVE}, MASTER 为主机模式, SLAVE 为从  
                                 机模式  
                                 <延时>:={0-0.000025},单位为秒, 只能在从机模式下设置该参数

查询语法                      CASCADE?

响应格式                      从机模式下的返回值：  
                                 CASCADE STATE,ON,MODE,SLAVE,DELAY, <DELAY>  
                                 主机模式下的返回值：  
                                 CASCADE STATE,ON,MODE,MASTER

示例                              设置为从机模式并设置延时为 *0.0000001s*:  
*CASCADE STATE,ON,MODE,SLAVE,DELAY,0.0000001*

## 3.32IQ 命令

下表显示每个 SDG 系列 IQ 命令可用性。

参数 /命令	SDG800	SDG1000	SDG2000 X	SDG5000	SDG1000 X	SDG6000 X	SDG6000 X-E
IQ	无	无	无	无	无	有	无

### 3.31.1 :IQ:CENTerfreq

描述	该命令设置 I/Q 调制的中心频率。
命令语法	<code>[[:SOURce]:IQ:CENTerfreq&lt;中心频率&gt;&lt;单位&gt;</code> <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	<code>[[:SOURce]:IQ:CENTerfreq?</code>
响应格式	<中心频率> (用 Hz 表示)
示例	设置中心频率为 1kHz: <code>:SOURce:IQ:CENTerfreq 1000Hz</code>

### 3.31.2 :IQ:SAMPlerate

描述	该命令设置 I/Q 的采样率
命令语法	<code>[[:SOURce]:IQ:SAMPlerate&lt;采样率&gt;&lt;单位&gt;</code> <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	<code>[[:SOURce]:IQ:SAMPlerate?</code>
响应格式	<采样率(用 Hz 表示)
示例	设置采样率为 100kHz: <code>:IQ:SAMPlerate 1000000</code> 或: <code>:IQ:SAMP 100kHz</code>

### 3.31.3 :IQ:SYMBOLrate

描述	该命令用于设置 I/Q 的符号率
命令语法	[[:SOURce]: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	[[:SOURce]: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: <i>:IQ:SYMB 1MS/s</i>

### 3.31.4 :IQ:AMPLitude

描述	该命令设置 I/Q 幅度。
命令语法	[[:SOURce]: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	[[:SOURce]:IQ:AMPLitude?
响应格式	<幅值>(表示为 Vrms.)
示例	设置 IQ 的幅值( $\sqrt{I^2+Q^2}$ )为 0.2Vrms: <i>:IQ:AMPL 0.2</i>

### 3.31.5 :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	[[:SOURce]:IQ:IQADjustment:GAIN <增益比><单位> <增益比> := I 与 Q 的增益比。默认单位为 dB.
查询语法	[[:SOURce]: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`

### 3.31.6 :IQ:IQADjustment:IOffset

**描述** 此命令调节 I 通道的偏移量。

**命令语法** `[[:SOURce]: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** `[[:SOURce]: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`

### 3.31.7 :IQ:IQADjustment:QOffset

**描述** 此命令调节 Q 通道的偏移量。

**命令语法** `[[:SOURce]: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** `[[:SOURce]: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 -0.001V`

### 3.31.8 :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** `[[:SOURce]:IQ:IQADjustment:QSkew <角度>`  
 <角度> := 角度。单位是度。

**查询语法** `[[:SOURce]:IQ:IQADjustment:QSkew?`

响应格式	<角度>(使用度表示)
示例	设置 Q 角度为 1° : <i>:IQ:IQADjustment:QSKew1.0</i>

### 3.31.9 :IQ:TRIGger:SOURce

描述	该命令设置 I/Q 的触发源。
命令语法	[[:SOURce]: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	[[:SOURce]:IQ:TRIGger:SOURce?
响应格式	<触发源>
示例	设置触发源为内触发: <i>:IQ:TRIGger:SOURce INTernal</i>

### 3.31.10 :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	[[:SOURce]:IQ:WAVEload:BUILtin<波形名称> <波形名称> := {下表的波形名}.
查询语法	[[:SOURce]: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: <i>:IQ:WAVE:BUIL 2ASK</i>

2ASK	4ASK	8ASK	BPSK	4PSK
8PSK	DBPSK	4DPSK	8DPSK	8QAM
16QAM	32QAM	64QAM	128QAM	256QAM

### 3.31.11 :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	[[:SOURce]:IQ:WAVEload:USERstored<波形名称> <波形名称> := {来自用户存储波形的名称}.
查询语法	[[:SOURce]:IQ:WAVEload?
响应格式	BUILtin USERstored <波形名称>
示例	设置 I/Q 波形为用户存储波形 UserIQ_1.arb: <i>:IQ:WAVEload:USERstored "UserIQ_1.arb"</i>

### 3.31.12 :IQ:FrequencySampling

DESCRIPTION	此命令用于设置 I/Q 波形的采样率
COMMAND SYNTAX	[[:SOURce]:IQ: FrequencySampling <采样率> <采样率> := {1000-300000000}. The unit is S/s
QUERY SYNTAX	[[:SOURce] :IQ:FrequencySampling? [:SOURce] :IQ:FrequencySamplingLimit?
RESPONSE FORMAT	<采样率> MAX,<最大采样率>,MIN,<最小采样率>
EXAMPLE	设置 I/Q 波形的采样率为 2 MS/s <i>:IQ:FrequencySampling 2000000</i>



## 4 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。

### 4.1 VISA 应用示例

#### 4.1.1 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。

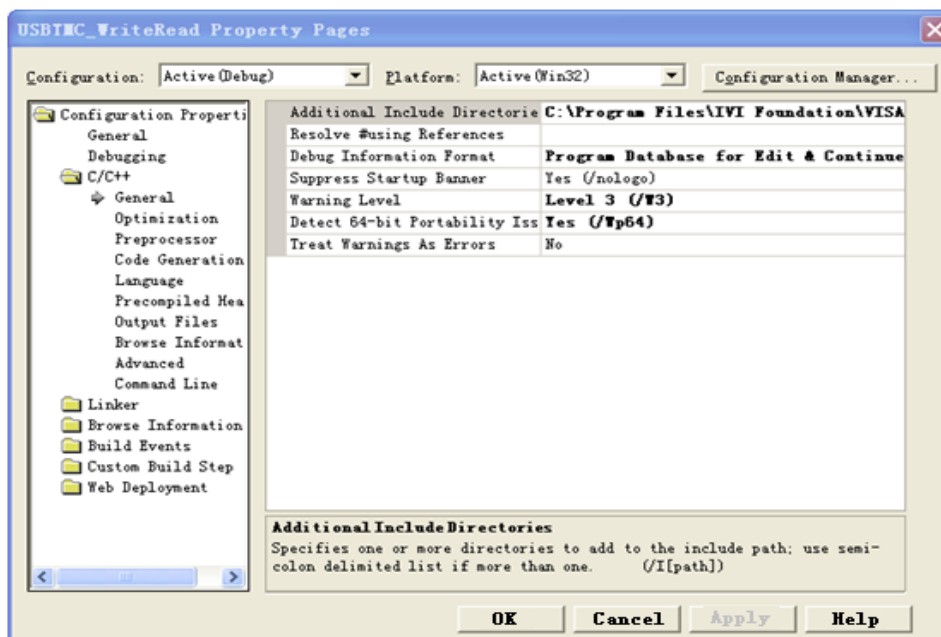
**步骤:**

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：

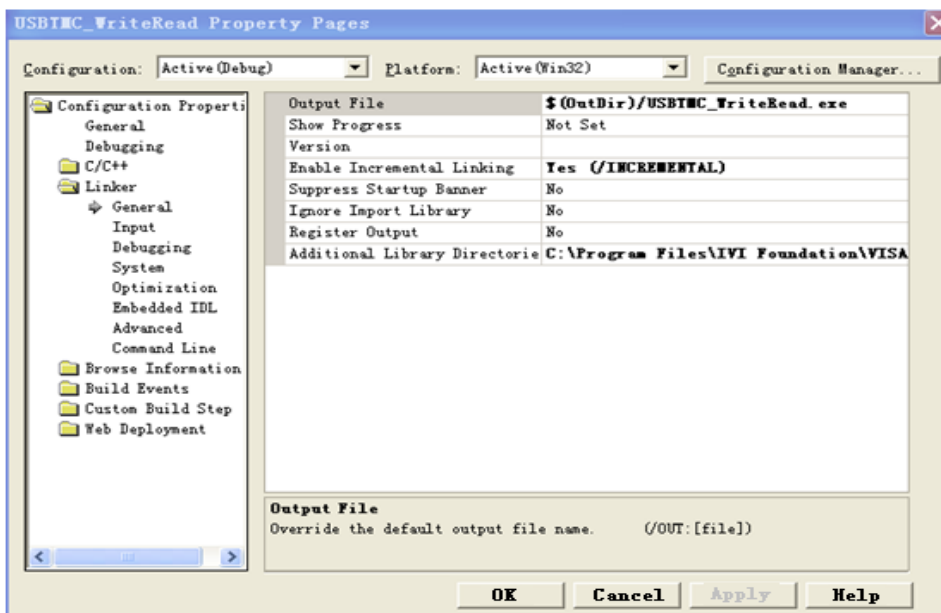
在 NI-VISA 安装路径找: visa.h、visatype.h、visa32.lib 文件，将它们复制到 VC++ 项目的根路径下并添加到项目中。在 projectname.cpp 文件上添加下列两行代码：

```
#include "visa.h"
#pragma comment(lib, "visa32.lib")
```
  - b) 动态：

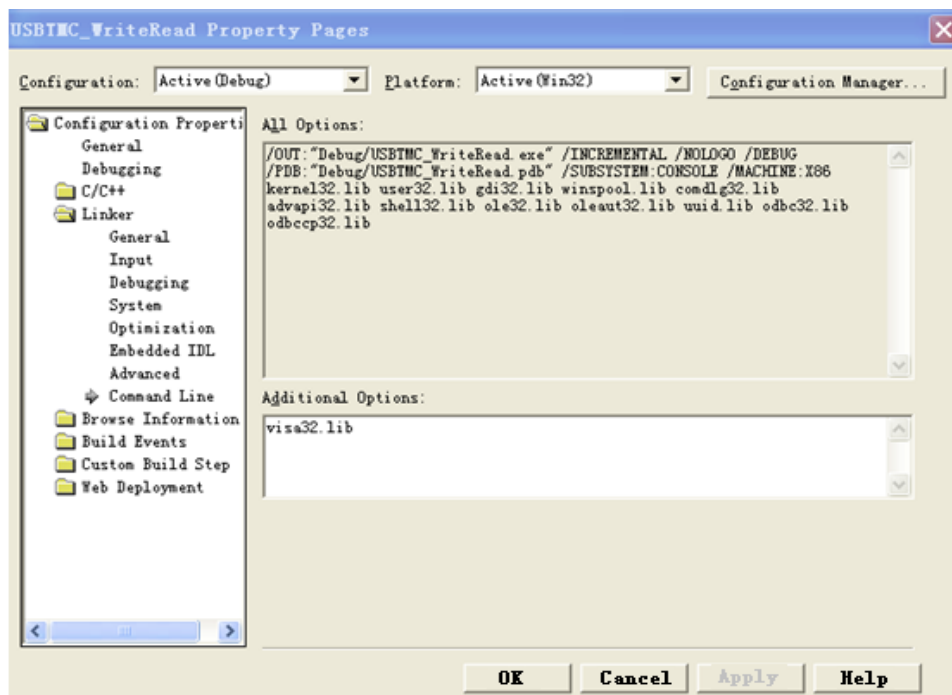
点击 "project >> properties"，在属性对话框左侧选择 "c/c++ --- General" 中，将 "Additional Include Directories" 项的值设置为 NI-VISA 的安装路径 (例如: C:\Program Files\IVI Foundation\VISA\WinNT\include)，如下图所示：



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_BUFLLEN];
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 * ccommand = "*IDN?\n";
    status = viPrintf (instr, ccommand);
    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[])

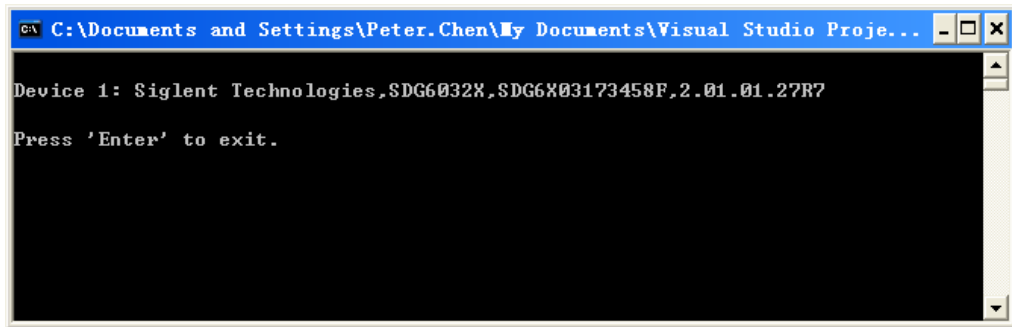
```

```

{
    Usbtmc_test();
    return 0;
}

```

### 运行结果:



### b) TCP/IP:

```

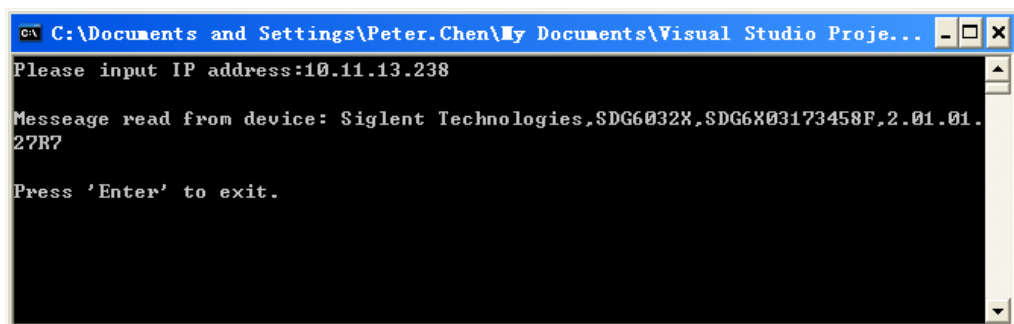
int TCP_IP_Test(char *pIP)
{
    char outputBuffer[VI_FIND_BUFLLEN];
    ViSession defaultRM, instr;
    ViStatus status;
    /* First we will need to open the default resource manager. */
    status = viOpenDefaultRM (&defaultRM);
    if (status<VI_SUCCESS)
    {
        printf("Could not open a session 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 ("\nMessage 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:



```
C:\Documents and Settings\Peter.Chen\My Documents\Visual Studio Proje...
Please input IP address:10.11.13.238
Message read from device: Siglent Technologies,SDG6032X,SDG6X03173458F,2.01.01.27R7
Press 'Enter' to exit.
```

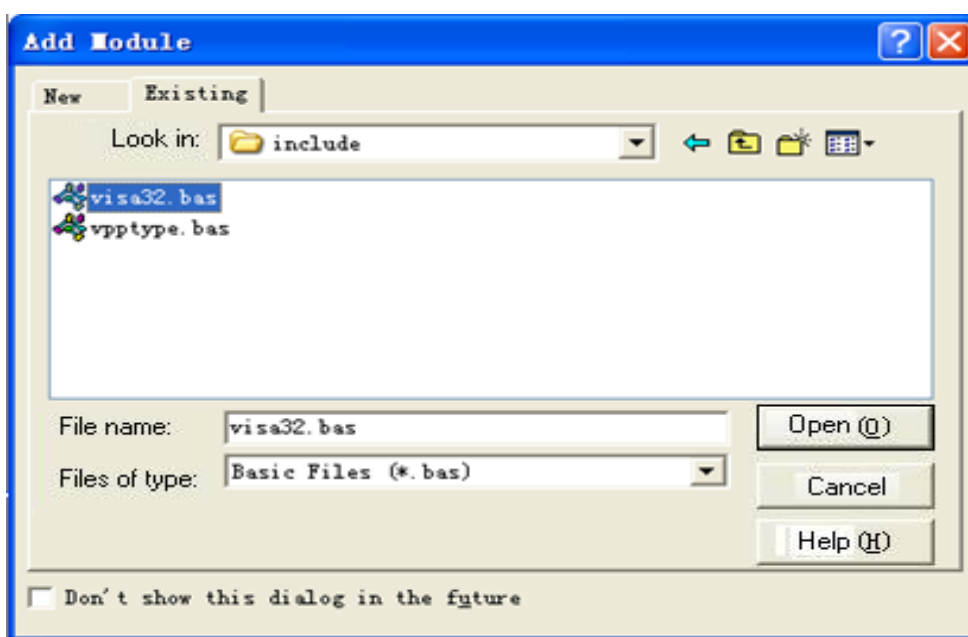
## 4.1.2 VB 示例

**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
```

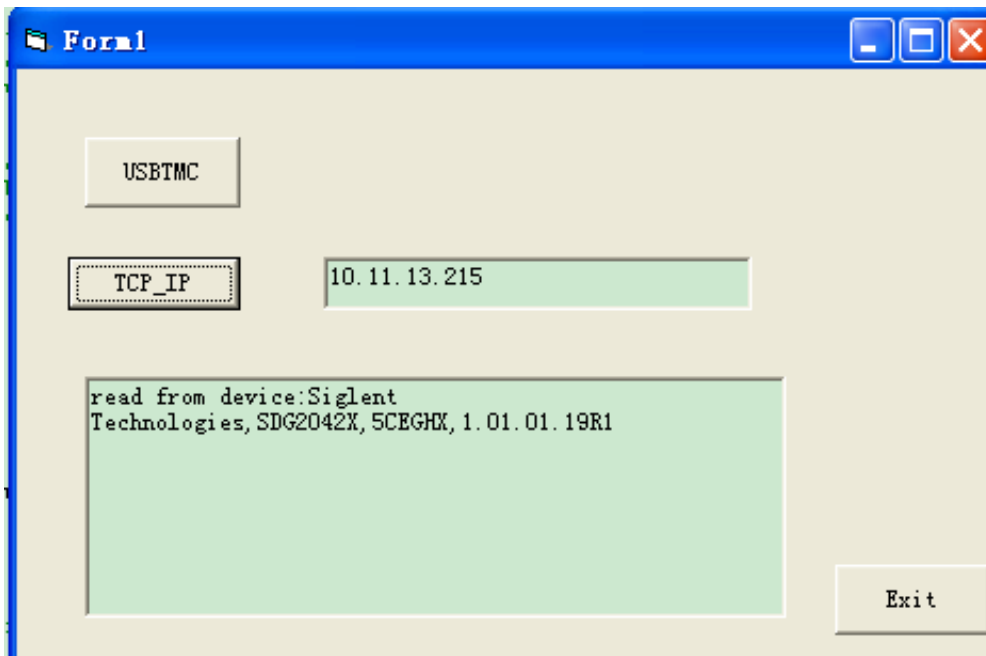
```
PrivateSub tcpipBtn_Click()
```

```
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:



### 4.1.3 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::sdg2000x::INSTR');

%Open the VISA object created
fopen(vu);

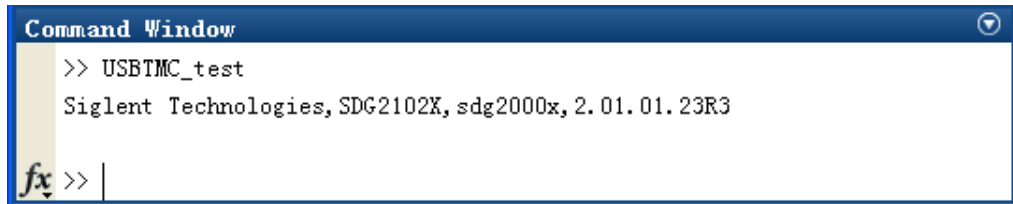
%Send the string "*IDN?",asking for the device's identification.
fprintf(vu, '*IDN?');

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

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

end
```

**运行结果:**



```
Command Window
>> USBTMC_test
Siglent Technologies, SDG2102X, sdg2000x, 2.01.01.23R3
fx >> |
```

## 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);

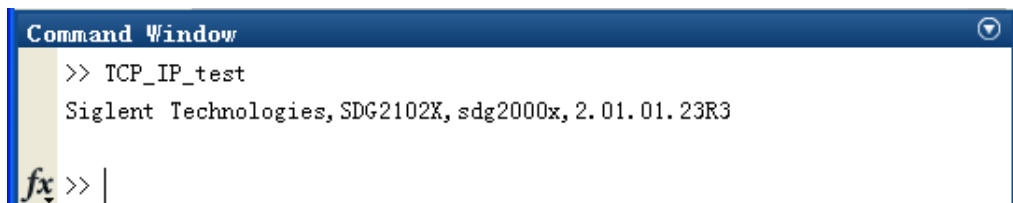
%Send the string "*IDN?", asking for the device's identification.
fprintf(vt, '*IDN?');

%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
fx >> |
```

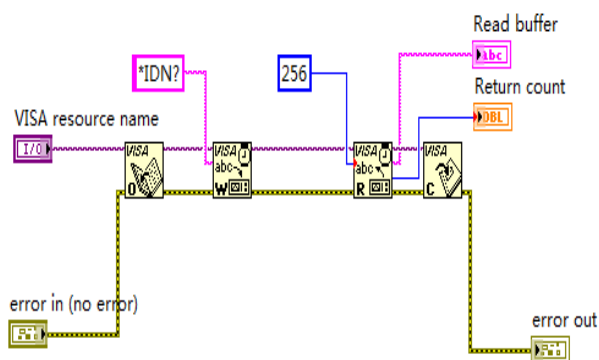
## 4.1.4 LabVIEW 示例

环境：Windows 7 32 位系统, LabVIEW 2011

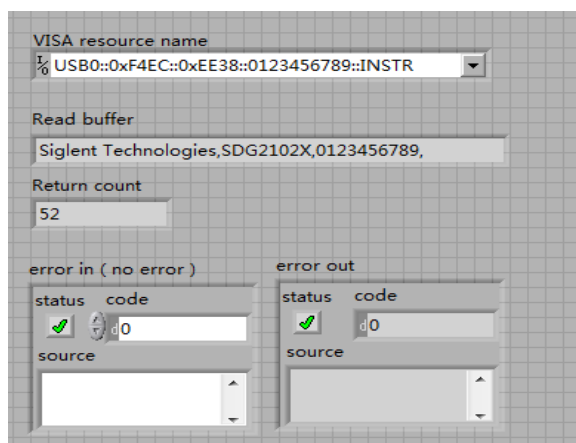
描述：分别通过 USBTMC 和 TCP/IP 访问信号源，并在 NI-VISA 上发送 “\*IDN?” 命令来查询设备信息。

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：

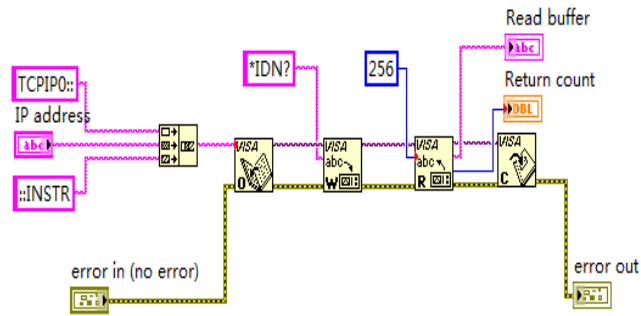


5. 从 VISA 资源名列表中选择设备资源并运行程序。

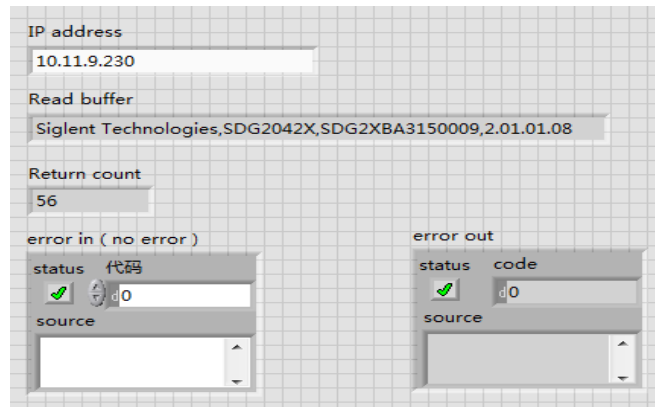


在此例中，VI 打开了一个 USBTMC 设备的 VISA 会话，并向设备写 \*IDN? 命令，以及回读的响应值。当所有通信完成时，VI 将关闭 VISA 会话。

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。





## 4.1.5 Python 示例

**Environment:** Python2.7, PyVISA 1.4

(请在安装 Python2.7 后安装 PyVISA。请在此网址上获取 PyVISA 安装指导：

<https://pyvisa.readthedocs.io/en/stable/getting.html>)

**描述:** 使用 Python 脚本新建一个 8 个点的任意波形(0x1000, 0x2000, 0x3000, 0x4000, 0x5000, 0x6000, 0x7000, 0x7fff)并将波形数据保存为“wave1.bin”,然后将其发送到设备,最后从设备读取该波形并保存为“wave2.bin”。

下面是脚本的代码:

```
#!/usr/bin/env python2.7
# -*- coding: utf-8 -*-

import visa
import time
import binascii

#USB resource of Device
device_resource = "USB0::0xF4EC::0x1101::#15::INSTR"

#Little endian, 16-bit2's complement
wave_points = [0x0010, 0x0020, 0x0030, 0x0040, 0x0050, 0x0060, 0x0070, 0xff7f]

def create_wave_file():
    """create a file"""
    f = open("wave1.bin", "wb")
    for a in wave_points:
        b = hex(a)
        b = b[2:]
        len_b = len(b)
        if (0 == len_b):
            b = '0000'
        elif (1 == len_b):
            b = '000' + b
        elif (2 == len_b):
            b = '00' + b
        elif (3 == len_b):
            b = '0' + b
        c = binascii.a2b_hex(b) #Hexadecimal integer to ASCII encoded string
        f.write(c)
```

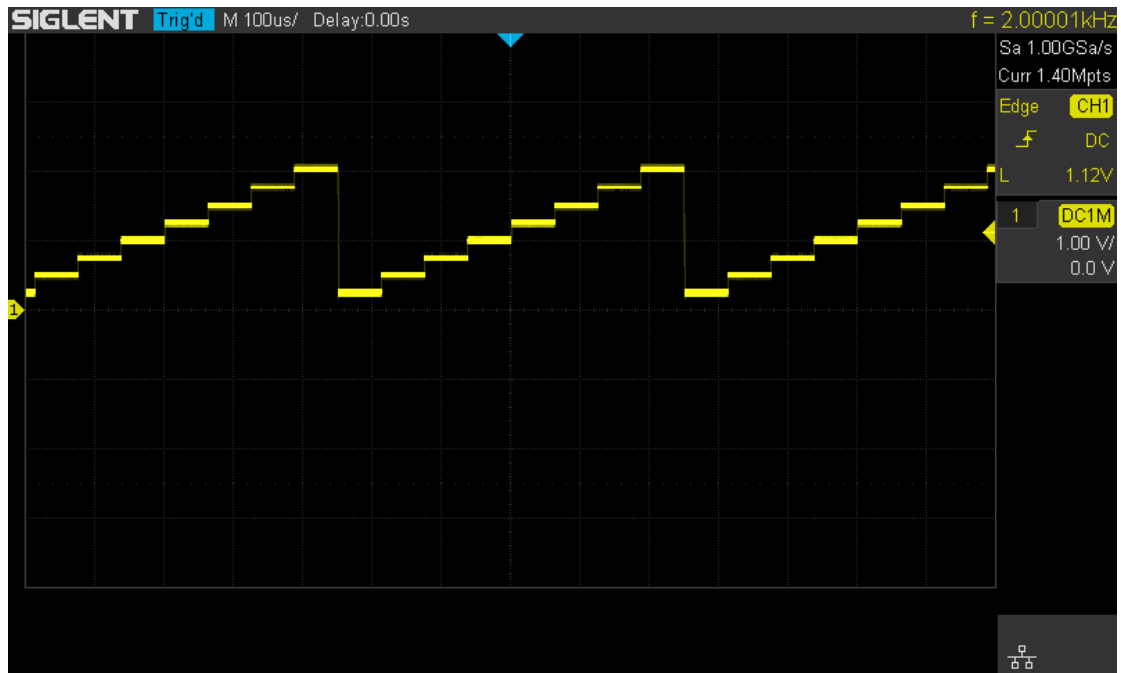
```
f.close()

def send_wawe_data(dev):
    """send wave1.bin to the device"""
    f = open("wave1.bin", "rb")    #wave1.bin is the waveform to be sent
    data = f.read()
    print 'write bytes:',len(data)
    dev.write("C1:WVDT
WVNM,wave1,FREQ,2000.0,AMPL,4.0,OFST,0.0,PHASE,0.0,WAVEDATA,%s" % (data))
#"X" series (SDG1000X/SDG2000X/SDG6000X/X-E)
    dev.write("C1:ARWV NAME,wave1")
    f.close()

def get_wawe_data(dev):
    """get wave from the device"""
    f = open("wave2.bin", "wb")    #save the waveform as wave2.bin
    dev.write("WVDT? user,wave1")    #"X" series (SDG1000X/SDG2000X/SDG6000X/X-E)
    time.sleep(1)
    data = dev.read()
    data_pos = data.find("WAVEDATA,") + len("WAVEDATA,")
    print data[0:data_pos]
    wave_data = data[data_pos:]
    print 'read bytes:',len(wave_data)
    f.write(wave_data)
    f.close()

if __name__ == '__main__':
    """
    device = visa.instrument(device_resource, timeout=5000, chunk_size = 40*1024)
    create_wawe_file()
    send_wawe_data(device)
    get_wawe_data(device)
```

输出波形:



## 4.2 Sockets 应用示例

### 4.2.1 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

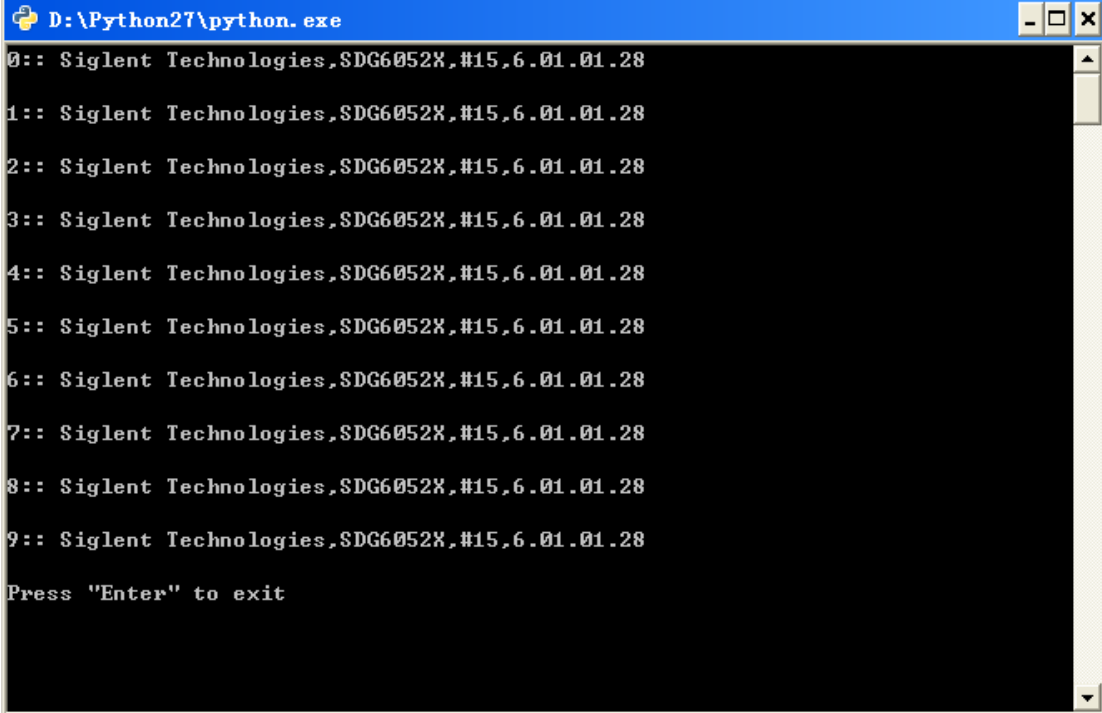
def SocketClose(Sock):
    #close the socket
    Sock.close()
    time.sleep(.300)

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



```
D:\Python27\python.exe
0:: Siglent Technologies,SDG6052X,#15,6.01.01.28
1:: Siglent Technologies,SDG6052X,#15,6.01.01.28
2:: Siglent Technologies,SDG6052X,#15,6.01.01.28
3:: Siglent Technologies,SDG6052X,#15,6.01.01.28
4:: Siglent Technologies,SDG6052X,#15,6.01.01.28
5:: Siglent Technologies,SDG6052X,#15,6.01.01.28
6:: Siglent Technologies,SDG6052X,#15,6.01.01.28
7:: Siglent Technologies,SDG6052X,#15,6.01.01.28
8:: Siglent Technologies,SDG6052X,#15,6.01.01.28
9:: Siglent Technologies,SDG6052X,#15,6.01.01.28
Press "Enter" to exit
```

## 5 索引

[\\*IDN](#)

[\\*OPC](#)

[\\*RST](#)

### A

[ARWV ArbWaVe](#)

### B

[BSWV BaSic WaVe](#)

[BTWV BursTWaVe](#)

[BUZZ BUZZer](#)

### C

[CASCADE](#)

[CHDR Comm HeaDeR](#)

[COUP COUPling](#)

[CMBN CoMBiNe](#)

### F

[FCNT FreqCouNter](#)

### H

[HARM HARMonic](#)

### I

[IQ:CENtIQ:CENTerfreq](#)

[IQ:SAMPIQ:SAMPlerate](#)

[IQ:SYMBIQ:SYMBolrate](#)

[IQ:AMPLIQ:AMPLitude](#)

[IQ:IQAD:GAINIQ:IQADjustment:GAIN](#)

[IQ:IQAD:IOFFsetIQ:IQADjustment:IOFFset](#)

[IQ:IQAD:QOFFsetIQ:IQADjustment:QOFFset](#)

[IQ:IQAD:QSKIQ:IQADjustment:QSKew](#)

[IQ:TRIG:SOURIQ:TRIGger:SOURce](#)

[IQ:WAVE:BUILIQ:WAVEload:BUILtin](#)

[IQ:WAVE:USERIQ:WAVEload:USERstored](#)

[:IQ:FrequencySampling](#)

[IVNT INVERT](#)

### L

[LAGG LAnGuaGe](#)

**M**

[MDWV MoDulateWaVe](#)

[MODE MODE](#)

**N**

[NBFM NumBer ForMat](#)

**O**

[OUTP OUTPut](#)

**P**

[PACP ParaCoPy](#)

**R**

[ROSC ROSCillator](#)

**S**

[SCFG Sys\\_CFG](#)

[SCSV SScreen SaVe](#)

[SWWV SweepWaVe](#)

[SYNC SYNC](#)

[STL StoreList](#)

[SYST:COMM:LAN:IPADSYSTem:COMMunicate:LAN:IPADdress](#)

[SYST:COMM:LAN:SMASSYSTem:COMMunicate:LAN:SMASK](#)

[SYST:COMM :LAN:GAT SYSTem:COMMunicate:LAN:GATeway](#)

[SRATE SampleRATE](#)

**W**

[WVDT WVDT](#)

**V**

[VOLTPRTVOLTPRT](#)

[VKEY VirtualKEY](#)



## 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。

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、台式万用表、射频信号源、直流电源、电子负载等基础测试测量仪器产品。2007年,鼎阳与高端示波器领导者美国力科建立了全球战略合作伙伴关系。2011年,鼎阳发展成为中国销量领先的数字示波器制造商。2014年,鼎阳发布了带宽高达1GHz的中国首款智能示波器SDS3000系列,引领实验室功能示波器向智能示波器过渡的趋势。2017年,鼎阳发布了多项参数突破国内技术瓶颈的SDG6000X系列脉冲/任意波形发生器。2018年,鼎阳推出了旗舰版高端示波器SDS5000X系列;同年发布国内第一款集频谱分析仪和矢量网络分析仪于一体的产品SVA1000X。目前,鼎阳已经在美国克利夫兰和德国汉堡成立子公司,产品远销全球70多个国家,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](http://www.siglent.com)

## 声明

 是深圳市鼎阳科技股份有限公司的注册商标,事先未经允许,不得以任何形式或通过任何方式复制本手册中的任何内容。  
本资料中的信息代替原先的此前所有版本。技术数据如有变更,恕不另行通告。

## 技术许可

对于本文档中描述的硬件和软件,仅在得到许可的情况下才会提供,并且只能根据许可进行使用或复制。

