

# UTG2000X系列可编程信号源

---

## 编程手册

V1.1

2024.07.03

**UNI-T®**

# 保证和声明

## 版权

2017 优利德科技（中国）股份有限公司

## 商标信息

**UNI-T**是优利德科技（中国）股份有限公司的注册商标。

## 文档编号

20240703

## 软件版本

V1.15.0019

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

## 声明

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

## 产品认证

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

## 联系我们

如您在使用此产品或本手册的过程中有任何问题或需求，可与 **UNI-T**联系：

电子邮箱：[infosh@uni-trend.com.cn](mailto:infosh@uni-trend.com.cn)

网址：<http://www.uni-trend.com.cn>

# SCPI 指令简介

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

## 指令格式

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

## 符号说明

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

- **大括号 { }**

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

- **竖线 |**

竖线用于分隔多个参数选项, 发送命令时必须选择其中一个参数。  
如:DISPlay:GRID:MODE { FULL | GRID | CROSS | NONE}命令。

- **方括号 [ ]**

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

- **三角括号 < >**

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

## 参数说明

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

- **布尔型**

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

- **整型**

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

- **实型**

除非另有说明，参数在有效值范围内可以取任意值。

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

- **离散型**

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

- **ASCII 字符串**

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

## 简写规则

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

## 数据返回

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

# SCPI 指令详解

## IEEE488.2 通用命令

### \*IDN?

➤ **命令格式:**

\*IDN?

➤ **功能描述:**

用于查询制造商名称、产品型号、产品序列号和软件版本号。

➤ **返回格式:**

制造商名称, 产品型号, 产品序列号, 由点号分隔的软件版本号。

**注意:** 返回的型号要与铭牌信息一致。

➤ **举例:**

UNI-T Technologies, UTG2000X, 000000001, 00.00.01

### \*RST

➤ **命令格式:**

\*RST

➤ **功能描述:**

用于恢复出厂设置并清空所有的错误信息及发送接收队列缓冲。

## SYSTEM 命令

用于对信号源进行最基本的操作, 主要包括全键盘锁定、系统设置数据的操作。

### :SYSTem:LOCK

➤ **命令格式:**

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

:SYSTem:LOCK?

➤ **功能描述:**

用于锁定或者解锁全键盘按键和触摸输入。

➤ **返回格式:**

查询返回全键盘锁定状态, 0 表示未锁定, 1 表示锁定。

➤ **举例:**

:SYSTem:LOCK ON            全键盘锁定

:SYSTem:LOCK OFF           全键盘解锁  
:SYSTem:LOCK?            查询返回 1，表示锁定

## **:SYSTem:CONFigure**

### ➤ **命令格式:**

:SYSTem:CONFigure <file>  
:SYSTem:CONFigure?

### ➤ **功能描述:**

用于读写配置文件，先发送该指令，然后发送配置文件数据到信号源。  
<file>表示配置文件。

### ➤ **返回格式:**

查询返回信号源当前配置文件数据。

### ➤ **举例:**

:SYSTem:CONFigure                    写入配置文件数据到信号源中并使其加载  
:SYSTem:CONFigure?                查询返回信号源当前配置文件数据二进制流

## **:SYSTem:PHASe:MODE**

### ➤ **命令格式:**

:SYSTem:PHASe:MODE {INDePendent | SYNChronization}  
:SYSTem:PHASe:MODE?

### ➤ **功能描述:**

控制通道间的相位模式，若为同步，则表示两个通道起始相位保持同步，否则相位独立。

### ➤ **返回格式:**

查询返回通道间的相位模式。

### ➤ **举例:**

:SYSTem:PHASe:MODE INDePendent    设置通道间为独立相位模式  
:SYSTem:PHASe:MODE?                查询返回 INDePendent

## **:SYSTem:LANGuage**

### ➤ **命令格式:**

:SYSTem:LANGuage {ENGLish|CHINese}  
:SYSTem:LANGuage?

### ➤ **功能描述:**

控制系统显示语言。

### ➤ **返回格式:**

查询返回系统显示语言。

➤ **举例：**

:SYSTem:LANGUage ENGLISH                    设置英文为系统显示语言  
:SYSTem:LANGUage?                            查询返回 ENGLISH

**:SYSTem:BEEP**

➤ **命令格式：**

:SYSTem:BEEP {{1 | ON} | {0 | OFF}}  
:SYSTem:BEEP?

➤ **功能描述：**

控制系统蜂鸣器开关

➤ **返回格式：**

查询返回蜂鸣器开关状态。

➤ **举例：**

:SYSTem:BEEP ON                            打开蜂鸣器  
:SYSTem:BEEP?                            查询返回 1

**:SYSTem:NUMBer:FORMat**

➤ **命令格式：**

:SYSTem:NUMBer:FORMat {COMMA|SPACE|NONE}  
:SYSTem:NUMBer:FORMat?

➤ **功能描述：**

控制系统数字格式的分隔符

➤ **返回格式：**

查询返回系统数字格式的分隔符。

➤ **举例：**

:SYSTem:NUMBer:FORMat NONE            设置无系统数字格式  
:SYSTem:NUMBer:FORMat?                查询返回 NONE

**:SYSTem:PICTure:FORMat**

➤ **命令格式：**

:SYSTem:PICTure:FORMat { BMP | JPEG | PNG}  
:SYSTem:PICTure:FORMat?

➤ **功能描述：**

设置获取图像和本地存储图像的数据格式。

➤ **返回格式：**

查询返回图像格式{ BMP | JPEG | PNG}。

➤ **举例：**

:SYSTem:PICTure:FORMat PNG                    设置 PNG 格式的图像数据  
:SYSTem:PICTure:FORMat?                    查询返回 PNG

**:SYSTem:BRIGhtness**

➤ **命令格式：**

:SYSTem:BRIGhtness { 10|30|50|70|90|100}  
:SYSTem:BRIGhtness?

➤ **功能描述：**

控制系统背光亮度等级

➤ **返回格式：**

查询返回系统背光亮度等级

➤ **举例：**

:SYSTem:BRIGhtness 30                    设置系统背光亮度 30%  
:SYSTem:BRIGhtness?                    查询返回 30

**:SYSTem:SLEEP:TIMe**

➤ **命令格式：**

:SYSTem:SLEEP:TIMe { CLOSe | 5MIN | 15MIN | 30MIN | 60MIN}  
:SYSTem:SLEEP:TIMe?

➤ **功能描述：**

控制系统休眠时间，单位是分钟

➤ **返回格式：**

查询返回休眠时间

➤ **举例：**

:SYSTem:SLEEP:TIMe 5 MIN                    设置系统 5 分钟之后自动休眠  
:SYSTem:SLEEP:TIMe?                    查询返回 5MIN

**:SYSTem:ECLK:STATus?**

➤ **命令格式：**

:SYSTem:ECLK:STATus?

➤ **功能描述：**

查询外部系统时钟源状态。

➤ **返回格式：**

查询返回外部时钟源状态，返回 0 表示无效，返回 1 表示有效。

➤ **举例：**



:SYSTEM:ECLK:STATus?                      查询返回 1,表示外部时钟源有效

### **:SYSTEM:CYMometer**

➤ **命令格式:**

:SYSTEM:CYMometer {{1 | ON} | {0 | OFF}}

:SYSTEM:CYMometer?

➤ **功能描述:**

控制系统频率计开关状态。

➤ **返回格式:**

查询返回系统频率计开关状态，0 表示关闭，1 表示打开。

➤ **举例:**

:SYSTEM:CYMometer ON                      打开系统频率计

:SYSTEM:CYMometer?                      查询返回 1

### **:SYSTEM:CYMometer:STATistics**

➤ **命令格式:**

:SYSTEM:CYMometer:STATistics {{1 | ON} | {0 | OFF}}

➤ **功能描述:**

打开或者关闭频率计的统计功能。

➤ **返回格式:**

查询返回系统频率计统计功能状态，0 表示关闭，1 表示打开。

➤ **举例:**

:SYSTEM:CYMometer:STATistics ON        打开系统频率计统计功能

:SYSTEM:CYMometer:STATistics?        查询返回 1

### **:SYSTEM:CYMometer:STATistics:TYPE**

➤ **命令格式:**

:SYSTEM:CYMometer:STATistics:TYPE {FREQUency|PERiod|DUTY|PWIDTh|NWIDTH}

➤ **功能描述:**

设置频率计统计参数的类型。

➤ **返回格式:**

查询返回频率计统计参数的类型。

➤ **举例:**

:SYSTEM:CYMometer:STATistics:TYPE FREQUency    统计频率计测量的频率值

:SYSTEM:CYMometer:STATistics:TYPE?            查询返回 FREQUency

## **:SYSTem:CYMometer:STATistics:COUNT?**

- **命令格式:**  
:SYSTem:CYMometer:STATistics:COUNT?
- **功能描述:**  
获取频率计的当前的统计次数。
- **返回格式:**  
查询返回频率计的当前的统计次数。
- **举例:**  
:SYSTem:CYMometer:STATistics:COUNT?      查询返回 40

## **:SYSTem:CYMometer:STATistics:CURRent?**

- **命令格式:**  
:SYSTem:CYMometer:STATistics:CURRent?
- **功能描述:**  
获取频率计的当前的统计模式下的当前测量值。
- **返回格式:**  
查询返回频率计的当前的统计模式下的当前测量值,以科学计数法返回。
- **举例:**  
:SYSTem:CYMometer:STATistics:CURRent? 查询返回 1e6

## **:SYSTem:CYMometer:STATistics:MAX?**

- **命令格式:**  
:SYSTem:CYMometer:STATistics:MAX?
- **功能描述:**  
获取频率计的当前的统计模式下统计的最大测量值。
- **返回格式:**  
查询返回频率计的当前的统计模式下统计的最大测量值,以科学计数法返回。
- **举例:**  
:SYSTem:CYMometer:STATistics:MAX? 查询返回 10e6

## **:SYSTem:CYMometer:STATistics:MIN?**

- **命令格式:**  
:SYSTem:CYMometer:STATistics:MIN?
- **功能描述:**  
获取频率计的当前的统计模式下统计的最小测量值。
- **返回格式:**

查询返回频率计的当前的统计模式下统计的最小测量值,以科学计数法返回。

➤ **举例:**

:SYSTem:CYMometer:STATistics:MIN? 查询返回 1e-2

**:SYSTem:CYMometer:STATistics:MEAN?**

➤ **命令格式:**

:SYSTem:CYMometer:STATistics:MEAN?

➤ **功能描述:**

频率计的当前的统计模式下统计的平均值。

➤ **返回格式:**

查询返回频率计的当前的统计模式下统计的平均值,以科学计数法返回。

➤ **举例:**

:SYSTem:CYMometer:STATistics:MEAN? 查询返回 1e5

**:SYSTem:CYMometer:FREQuency?**

➤ **命令格式:**

:SYSTem:CYMometer:FREQuency?

➤ **功能描述:**

获取频率计的当前测量的频率。

➤ **返回格式:**

查询返回获取频率计的当前测量的频率, 单位 Hz, 采用科学计数法返回数据。

➤ **举例:**

:SYSTem:CYMometer:FREQuency? 查询返回 2e+3

**:SYSTem:CYMometer:PERiod?**

➤ **命令格式:**

:SYSTem:CYMometer:PERiod?

➤ **功能描述:**

获取频率计的当前测量的周期。

➤ **返回格式:**

查询返回获取频率计的当前测量的周期, 单位 S, 采用科学计数法返回数据。

➤ **举例:**

:SYSTem:CYMometer:PERiod? 查询返回 2e-3

**:SYSTem:CYMometer:DUTY?**

➤ **命令格式:**

:SYSTem:CYMometer:DUTY?

➤ **功能描述:**

获取频率计的当前测量的占空比。

➤ **返回格式:**

查询返回获取频率计的当前测量的占空比，单位%。

➤ **举例:**

:SYSTem:CYMometer:DUTY?                      查询返回 20，表示占空比 20%

**:SYSTem:CYMometer:PWIDTh?**

➤ **命令格式:**

:SYSTem:CYMometer:PWIDTh?

➤ **功能描述:**

获取频率计的当前测量的正脉宽宽度。

➤ **返回格式:**

查询返回获取频率计的当前测量的正脉宽宽度，单位 s。

➤ **举例:**

:SYSTem:CYMometer:PWIDTh?                      查询返回 1e-3，表示占空比 1 毫秒。

**:SYSTem:CYMometer:NWIDTh?**

➤ **命令格式:**

:SYSTem:CYMometer:NWIDTh?

➤ **功能描述:**

获取频率计的当前测量的负脉宽宽度。

➤ **返回格式:**

查询返回获取频率计的当前测量的负脉宽宽度，单位 s。

➤ **举例:**

:SYSTem:CYMometer:NWIDTh?                      查询返回 1e-3，表示占空比 1 毫秒

**:SYSTem:COMMunicate:LAN:APPLy**

➤ **命令格式:**

:SYSTem:COMMunicate:LAN:APPLy

➤ **功能描述:**

用于立即生效当前设置的网络参数。

**:SYSTem:COMMunicate:LAN:GATEway**

➤ **命令格式:**

:SYSTem:COMMunicate:LAN:GATEway <gateway>

:SYSTem:COMMunicate:LAN:GATEway?

➤ **功能描述:**

用于设置默认网关。<gateway>属于 ASCII 字符串参数，格式为 xxx.xxx.xxx.xxx。

➤ **返回格式:**

查询返回默认网关。

➤ **举例:**

:SYST:COMM:LAN:GATE "192.168.1.1" 设置默认网关 192.168.1.1

:SYST:COMM:LAN:GATE? 查询返回 192.168.1.1

### **:SYSTem:COMMunicate:LAN:SMASK**

➤ **命令格式:**

:SYSTem:COMMunicate:LAN:SMASK <submask>

:SYSTem:COMMunicate:LAN:SMASK?

➤ **功能描述:**

用于设置子网掩码。<submask>属于 ASCII 字符串参数，格式为 xxx.xxx.xxx.xxx。

➤ **返回格式:**

查询返回子网掩码。

➤ **举例:**

:SYST:COMM:LAN:SMASK "255.255.255.0" 设置子网掩码 255.255.255.0

:SYST:COMM:LAN:SMASK? 查询返回 255.255.255.0

### **:SYSTem:COMMunicate:LAN:IPADdress**

➤ **命令格式:**

:SYSTem:COMMunicate:LAN:IPADdress <ip>

:SYSTem:COMMunicate:LAN:IPADdress?

➤ **功能描述:**

用于设置 IP 地址。<ip>属于 ASCII 字符串参数，格式为 xxx.xxx.xxx.xxx。

➤ **返回格式:**

查询返回 IP 地址。

➤ **举例:**

:SYST:COMM:LAN:IPAD "192.168.1.10" 设置 IP 地址 192.168.1.10

:SYST:COMM:LAN:IPAD? 查询返回 192.168.1.10

### **:SYSTem:COMMunicate:LAN:DHCP**

➤ **命令格式:**

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

:SYSTem:COMMunicate:LAN:DHCP?

➤ **功能描述:**

用于切换（自动 IP）和（手动 IP）配置模式。

➤ **返回格式:**

查询返回动态配置模式，0 表示（手动 IP），1 表示（自动 IP）。

➤ **举例:**

:SYST:COMM:LAN:DHCP ON                    打开 IP 动态配置

:SYST:COMM:LAN:DHCP?                    查询返回 1

### **:SYSTem:COMMunicate:LAN:MAC?**

➤ **命令格式:**

:SYSTem:COMMunicate:LAN:MAC?

➤ **返回格式:**

查询返回 MAC 物理地址。

➤ **举例:**

:SYST:COMM:LAN:MAC?                    查询返回 00-2A-A0-AA-E0-56

## **CHANnel 命令**

用于设置信号源通道相关功能。

### **:CHANnel<n>:MODE**

➤ **命令格式:**

:CHANnel<n>:MODE {CONTInue | MODulation| SWEep| BURSt }

:CHANnel<n>:MODE?

➤ **功能描述:**

设置指定通道信号模式，分别为 CONTInue、MODulation、SWEep、BURSt。

<n>: 通道号，n 取值 1、2。

➤ **返回格式:**

查询返回指定通道信号模式。

➤ **举例:**

:CHANnel1:MODE MODulation                    设置通道 1 信号调制输出

:CHANnel1:MODE?                    查询返回        MODulation

## **:CHANnel<n>:OUTPut**

### ➤ **命令格式:**

:CHANnel<n>:OUTPut {{1 | ON} | {0 | OFF}}

:CHANnel<n>:OUTPut?

### ➤ **功能描述:**

设置打开或关闭指定通道的输出。

<n>: 通道号, n 取值 1、2。

### ➤ **返回格式:**

查询返回指定通道的输出状态, 0 表示关闭, 1 表示打开。

### ➤ **举例:**

:CHANnel1:OUTPut ON                                 设置打开通道 1 输出

:CHANnel1:OUTPut?                                     查询返回 1

## **:CHANnel<n>:INVersion**

### ➤ **命令格式:**

:CHANnel<n>:INVersion {{1 | ON} | {0 | OFF}}

:CHANnel<n>:INVersion?

### ➤ **功能描述:**

设置打开或关闭指定通道反向。

<n>: 通道号, n 取值 1、2。

### ➤ **返回格式:**

查询返回指定通道的反向状态, 0 表示关闭, 1 表示打开。

### ➤ **举例:**

:CHANnel1:INVersion ON                                 设置打开通道 1 反向输出

:CHANnel1:INVersion?                                     查询返回 1

## **:CHANnel<n>:OUTPut:SYNC**

### ➤ **命令格式:**

:CHANnel<n>:OUTPut:SYNC {{1 | ON} | {0 | OFF}}

:CHANnel<n>:OUTPut:SYNC?

### ➤ **功能描述:**

设置通道同步输出状态。

注意: 设备只有一个同步输出接口, 同时只能打开一个通道的同步输出。

<n>: 通道号, n 取值 1、2。

### ➤ **返回格式:**

查询返回指定通道的同步输出状态，0 表示关闭，1 表示打开。

➤ **举例：**

:CHANnel1:OUTPut:SYNC ON                    设置打开通道 1 同步输出  
:CHANnel1:OUTPut:SYNC?                    查询返回 1

**:CHANnel<n>:LIMit:ENABle**

➤ **命令格式：**

:CHANnel<n>:LIMit:ENABle {{1 | ON} | {0 | OFF}}  
:CHANnel<n>:LIMit:ENABle?

➤ **功能描述：**

设置指定通道限幅开关。

<n>：通道号，n 取值 1、2。

➤ **返回格式：**

查询返回指定通道的限幅状态。

➤ **举例：**

:CHANnel1:LIMit:ENABle ON                    设置打开通道 1 限幅  
:CHANnel1:LIMit:ENABle?                    查询返回 1

**:CHANnel<n>:LIMit:LOWer**

➤ **命令格式：**

:CHANnel<n>:LIMit:LOWer {<voltage>}  
:CHANnel<n>:LIMit:LOWer?

➤ **功能描述：**

设置指定通道限幅下限值。

<voltage>表示电压，单位当前通道指定单位。

<n>：通道号，n 取值 1、2。

➤ **返回格式：**

查询返回指定通道的限幅下限值，采用科学计数法返回。

➤ **举例：**

:CHANnel1:LIMit:LOWer 2                    设置通道 1 限幅下限 2V  
:CHANnel1:LIMit:LOWer?                    查询返回 2e+0

**:CHANnel<n>:LIMit:UPPer**

➤ **命令格式：**

:CHANnel<n>:LIMit:UPPer {<voltage>}  
:CHANnel<n>:LIMit:UPPer?



➤ **功能描述:**

设置指定通道限幅上限值。

<voltage>表示电压，单位当前通道指定单位。

<n>: 通道号，n 取值 1、2。

➤ **返回格式:**

查询返回指定通道的限幅上限值，采用科学计数法返回。

➤ **举例:**

:CHANnel1:LIMit:UPPer 2                      设置通道 1 限幅上限 2V

:CHANnel1:LIMit:UPPer?                      查询返回 2e+0

### **:CHANnel<n>:AMPLitude:UNIT**

➤ **命令格式:**

:CHANnel<n>:AMPLitude:UNIT {VPP | VRMS | DBM}

:CHANnel<n>:AMPLitude:UNIT?

➤ **功能描述:**

设置指定通道输出幅度单位。

<n>: 通道号，n 取值 1、2。

➤ **返回格式:**

查询返回指定通道的输出幅度单位。

➤ **举例:**

:CHANnel1:AMPLitude:UNIT VPP              设置通道 1 输出幅度单位为 VPP

:CHANnel1:AMPLitude:UNIT?              查询返回 VPP

### **:CHANnel<n>:LOAD**

➤ **命令格式:**

:CHANnel<n>:LOAD <resistance>

:CHANnel<n>:LOAD?

➤ **功能描述:**

设置指定通道输出负载。

<resistance>表示负载电阻值，单位为 $\Omega$

<n>: 通道号，n 取值 1、2。

注意：阻值取值范围为 1~1000000,其中 1000000 对应于高阻。

➤ **返回格式:**

查询返回指定通道的负载阻值，采用科学计数法返回。

➤ **举例:**

:CHANnel1:LOAD 50                    设置通道 1 输出负载 50Ω  
:CHANnel1:LOAD?                    查询返回 50e+0

### **:CHANnel<n>:COPY**

➤ **命令格式:**

:CHANnel<n>:COPY

➤ **功能描述:**

执行通道复制操作，该条指令不支持查询操作。

<n>: 通道号，n 取值 1、2。

➤ **返回格式:**

无返回值。

➤ **举例:**

:CHANnel1:COPY                    通道 1 的数据拷贝到通道 2

### **:CHANnel<n>:PNCode**

➤ **命令格式:**

:CHANnel<n>:PNCode <code>

:CHANnel<n>:PNCode?

➤ **功能描述:**

设置指定通道 PN 码，该指令只对具备 PN 码功能的波形有效。

<code>: 表示 PN 码，如下所示:

{PN3|PN5|PN7|PN9|PN11|PN13|PN15|PN17|PN19|PN21|PN23|PN25|PN27|PN29|PN31|PN33}

<n>: 通道号，n 取值 1、2。

➤ **返回格式:**

查询返回指定通道 PN 码。

➤ **举例:**

:CHANnel1:PNCode PN9                    设置通道 1 PN 码为 PN9

:CHANnel1:PNCode?                    查询返回 PN9

### **:CHANnel<n>:TRIGger:SOURce**

➤ **命令格式:**

:CHANnel<n>:TRIGger:SOURce {INTernal|EXTernal|MANual}

:CHANnel<n>:TRIGger:SOURce?

➤ **功能描述:**

设置指定通道触发源，该指令只对扫频和猝发功能有效。

<n>: 通道号，n 取值 1、2。

➤ **返回格式:**

查询返回指定通道触发源。

➤ **举例:**

:CHANnel1:TRIGger:SOURce INTernal	设置通道 1 内部触发源
:CHANnel1:TRIGger:SOURce?	查询返回 INTernal

### **:CHANnel<n>:TRIGger:OUTPut**

➤ **命令格式:**

:CHANnel<n>:TRIGger:OUTPut {CLOSe|OPEn}

:CHANnel<n>:TRIGger:OUTPut?

➤ **功能描述:**

设置指定通道触发输出模式，该指令只对扫频和猝发功能有效。

<n>: 通道号, n 取值 1、2。

➤ **返回格式:**

查询返回指定通道触发输出模式。

➤ **举例:**

:CHANnel1:TRIGger:OUTPut OPEn	设置通道 1 触发输出使能
:CHANnel1:TRIGger:OUTPut?	查询返回 OPEn

### **:CHANnel<n>:MERge**

➤ **命令格式:**

:CHANnel<n>:MERge {{1 | ON} | {0 | OFF}}

:CHANnel<n>:MERge?

➤ **功能描述:**

设置指定通道合并输出开关。

1、2 通道合并只能从 1 和 2 通道输出。

<n>: 通道号, n 取值 1、2。

➤ **返回格式:**

查询返回指定通道合并开合状态。

➤ **举例:**

:CHANnel1:MERge ON	1 和 2 通道合并信号且信号从 1 通道输出
:CHANnel1:MERge?	返回 1

### **:CHANnel:COUPle<m>:ONOff**

➤ **命令格式:**

:CHANnel:COUPle<m>:ONOff {{1|ON} | {0|OFF}}

:CHANnel:COUPle<m>:ONOff?

➤ **功能描述:**

设置打开或关闭通道耦合。

<m>: 通道号, m 取值 1。

1 表示一和二通道耦合

➤ **返回格式:**

查询返回指定通道耦合的状态, 0 表示关闭, 1 表示打开。

➤ **举例:**

:CHANnel:COUPle1:ONOff ON                      设置打开通道耦合

:CHANnel:COUPle1:ONOff?                        查询返回 1

### **:CHANnel:COUPle<m>:TYPE**

➤ **命令格式:**

:CHANnel:COUPle<m>:TYPE {PARAM|TRACK}

:CHANnel:COUPle<m>:TYPE?

➤ **功能描述:**

设置指定通道耦合类型。该指令只有在通道耦合开关打开的状态下有效。

<m>: 通道号, m 取值 1。

1 表示一和二通道耦合

➤ **返回格式:**

查询返回通道耦合类型。

➤ **举例:**

:CHANnel:COUPle1:TYPE PARAM 设置通道耦合为参数耦合类型

:CHANnel:COUPle1:TYPE?                        查询返回 PARAM

### **:CHANnel:COUPle<m>:FREQuency**

➤ **命令格式:**

:CHANnel:COUPle<m>:FREQuency {{1 | ON} | {0 | OFF}}

:CHANnel:COUPle<m>:FREQuency?

➤ **功能描述:**

设置通道频率耦合开关, 通道只存在 1、2 通道耦合一种类型。该指令只有在参数跟随类型下有效。

<m>: 通道号, m 取值 1。

1 表示一和二通道耦合

➤ **返回格式:**

查询返回通道耦合开关状态, 0 表示关闭, 1 表示开启。

➤ **举例:**

:CHANnel:COUPle1:FREQuency ON            开启 1 和 2 通道耦合  
:CHANnel:COUPle1:FREQuency?            返回 1

### **:CHANnel:COUPle<m>:FREQuency:SCALe**

➤ **命令格式:**

:CHANnel:COUPle<m>:FREQuency:SCALe <scale>  
:CHANnel:COUPle<m>:FREQuency:SCALe?

➤ **功能描述:**

设置通道耦合频率比例，通道只存在 1、2 通道耦合一种类型。该指令只有在频率跟随打开下有效。

<scale >: 耦合频率比例。

<m>: 通道号，m 取值 1。

1 表示一和二通道耦合

➤ **返回格式:**

查询返回通道耦合频率比例，返回科学计数法。

➤ **举例:**

:CHANnel:COUPle1:FREQuency:SCALe 0.1        设置 2 通道比 1 通道耦合比例为 0.1  
:CHANnel:COUPle1:FREQuency:SCALe?        返回 1e-1

### **:CHANnel:COUPle<m>:FREQuency:DEV**

➤ **命令格式:**

:CHANnel:COUPle<m>:FREQuency:DEV <dev >  
:CHANnel:COUPle<m>:FREQuency:DEV?

➤ **功能描述:**

设置通道耦合频率偏差，通道只存在 1、2 通道耦合一种类型。该指令只有在频率跟随打开下有效。

<scale >: 耦合频率偏差，单位 Hz。

<m>: 通道号，m 取值 1。

1 表示一和二通道耦合

➤ **返回格式:**

查询返回通道耦合频率偏差，返回科学计数法。

➤ **举例:**

:CHANnel:COUPle1:FREQuency:DEV 100        设置 2 通道比 1 通道耦合偏差为 100Hz  
:CHANnel:COUPle1:FREQuency:DEV?        返回 1e+2

### **:CHANnel:COUPle<m>:PHASe**

➤ **命令格式:**

:CHANnel:COUPle<m>:PHASe {{1 | ON} | {0 | OFF}}

:CHANnel:COUPle<m>:PHASe?

➤ **功能描述:**

设置通道相位耦合开关，通道只存在 1、2 通道耦合一种类型。该指令只有在参数跟随类型下有效。

<m>: 通道号，m 取值 1。

1 表示一和二通道耦合

➤ **返回格式:**

查询返回通道耦合开关状态，0 表示关闭，1 表示开启。

➤ **举例:**

:CHANnel:COUPle1:PHASe ON            开启 1 和 2 通道耦合

:CHANnel:COUPle1:PHASe?            返回 1

### **:CHANnel:COUPle<m>:PHASe:SCALe**

➤ **命令格式:**

:CHANnel:COUPle<m>:PHASe:SCALe <scale>

:CHANnel:COUPle<m>:PHASe:SCALe?

➤ **功能描述:**

设置通道耦合相位比例，通道只存在 1、2 通道耦合一种类型。该指令只有在相位跟随打开下有效。

<scale >: 耦合相位比例。

<m>: 通道号，m 取值 12。

1 表示一和二通道耦合

➤ **返回格式:**

查询返回通道耦合相位比例，返回科学计数法。

➤ **举例:**

:CHANnel:COUPle1:PHASe:SCALe 0.1    设置 2 通道比 1 通道耦合比例为 0.1

:CHANnel:COUPle1:PHASe:SCALe?      返回 1e-1

### **:CHANnel:COUPle<m>:PHASe:DEV**

➤ **命令格式:**

:CHANnel:COUPle<m>:PHASe:DEV <dev >

:CHANnel:COUPle<m>:PHASe:DEV?

➤ **功能描述:**

设置通道耦合相位偏差，通道只存在 1、2 通道耦合一种类型。该指令只有在相位跟随打开下有效。

<scale >: 耦合相位偏差，单位°。

<m>: 通道号，m 取值 1。

1 表示一和二通道耦合

➤ **返回格式:**

查询返回通道耦合相位偏差，返回科学计数法。

➤ **举例:**

:CHANnel:COUPle1:PHASe:DEV 100            设置 2 通道比 1 通道耦合偏差为 100°  
:CHANnel:COUPle1:PHASe:DEV?            返回 1e+2

**:CHANnel:COUPle<m>:AMPLitude**

➤ **命令格式:**

:CHANnel:COUPle<m>:AMPLitude {{1 | ON} | {0 | OFF}}  
:CHANnel:COUPle<m>:AMPLitude?

➤ **功能描述:**

设置通道幅度耦合开关，通道只存在 1、2 通道耦合一种类型。该指令只有在参数跟随类型下有效。

<m>: 通道号，m 取值 1。

1 表示一和二通道耦合

➤ **返回格式:**

查询返回通道耦合幅度开关状态，0 表示关闭，1 表示开启。

➤ **举例:**

:CHANnel:COUPle1:AMPLitude ON            开启 1 和 2 通道耦合幅度  
:CHANnel:COUPle1:AMPLitude?            返回 1

**:CHANnel:COUPle<m>:AMPLitude:SCALE**

➤ **命令格式:**

:CHANnel:COUPle<m>:AMPLitude:SCALE <scale>  
:CHANnel:COUPle<m>:AMPLitude:SCALE?

➤ **功能描述:**

设置通道耦合幅度比例，通道只存在 1、2 通道耦合一种类型。该指令只有在幅度跟随打开下有效。

<scale >: 耦合幅度比例。

<m>: 通道号，m 取值 1。

1 表示一和二通道耦合

➤ **返回格式:**

查询返回通道耦合幅度比例，返回科学计数法。

➤ **举例:**

:CHANnel:COUPle1:AMPLitude:SCALE 0.1    设置 2 通道比 1 通道耦合比例为 0.1  
:CHANnel:COUPle1:AMPLitude:SCALE?    返回 1e-1

## **:CHANnel:COUPle<m>:AMPLitude:DEV**

### ➤ **命令格式:**

:CHANnel:COUPle<m>:AMPLitude:DEV <dev >

:CHANnel:COUPle<m>:AMPLitude:DEV?

### ➤ **功能描述:**

设置通道耦合幅度偏差，通道只存在 1、2 通道耦合一种类型。该指令只有在幅度跟随打开下有效。

<scale >: 耦合幅度偏差，单位 Vpp。

<m>: 通道号，m 取值 1。

1 表示一和二通道耦合

### ➤ **返回格式:**

查询返回通道耦合幅度偏差，返回科学计数法。

### ➤ **举例:**

:CHANnel:COUPle1:AMPLitude:DEV 1      设置 2 通道比 1 通道耦合偏差为 1Vpp

:CHANnel:COUPle1:AMPLitude:DEV?      返回 1e+2

## **:CHANnel:COUPle<m>:TRACk:PHASe:DEV**

### ➤ **命令格式:**

:CHANnel:COUPle<m>:TRACk:PHASe:DEV <dev >

:CHANnel:COUPle<m>:TRACk:PHASe:DEV?

### ➤ **功能描述:**

设置通道跟踪下的相位偏差，通道只存在 1、2 通道耦合一种类型。该指令只有在通道跟踪下有效。

<scale >: 相位偏差，单位°。

<m>: 通道号，m 取值 1。

1 表示一和二通道耦合

### ➤ **返回格式:**

查询返回通道跟踪下的相位偏差，返回科学计数法。

### ➤ **举例:**

:CHANnel:COUPle1:TRACk:PHASe:DEV 100      设置 2 通道比 1 通道耦合偏差为 100°

:CHANnel:COUPle1:TRACk:PHASe:DEV?      返回 1e+2

## **:CHANnel<n>:SELect**

### ➤ **命令格式:**

:CHANnel<n>:SELect

:CHANnel<n>:SELect?

### ➤ **功能描述:**



用于选择通道。

<n>: {1|2}, 分别表示{CH1|CH2}。

➤ **返回格式:**

查询返回 1 或 0, 分别代表 ON 或 OFF。

➤ **举例:**

:CHAN1:SElect	选择通道 1。
:CHAN1:SElect?	查询返回 1, 表示通道被选中。

## 连续

### :CHANnel<n>:BASE:WAVE

➤ **命令格式:**

:CHANnel<n>:BASE:WAVE { SINE | SQUARE | PULSE | RAMP | ARB | NOISE | DC | HARMONIC | PRBS|EXP|DPULSE }  
:CHANnel<n>:BASE:WAVE?

➤ **功能描述:**

设置指定通道基波类型。分别为正弦波、方波、脉冲波、三角波、任意波、噪声、直流、谐波、伪随机二进制序列、表达式、双脉冲。

<n>: 通道号, n 取值 1、2。

➤ **返回格式:**

查询返回指定通道的基波类型。

➤ **举例:**

:CHANnel1:BASE:WAVE SINE	设置通道 1 基本类型为正弦波
:CHANnel1:BASE:WAVE?	查询返回 SINE

### :CHANnel<n>:BASE:FREQUENCY

➤ **命令格式:**

:CHANnel<n>:BASE:FREQUENCY {<freq>}  
:CHANnel<n>:BASE:FREQUENCY?

➤ **功能描述:**

设置指定通道输出频率。

<freq>表示频率值, 单位 Hz。 (1e-6Hz ~ 当前波形允许最大频率)

<n>: 通道号, n 取值 1、2。

➤ **返回格式:**

查询返回指定通道的输出频率, 采用科学计数法返回。

➤ **举例：**

:CHANnel1:BASE:FREQuency 2000	设置通道 1 输出频率 2KHz
:CHANnel1:BASE:FREQuency?	查询返回 2e+3

**:CHANnel<n>:BASE:PERiod**

➤ **命令格式：**

:CHANnel<n>:BASE:PERiod { <period> }  
:CHANnel<n>:BASE:PERiod?

➤ **功能描述：**

设置指定通道输出周期。  
<period>表示周期，单位 S。  
若为正弦波：范围为（当前允许最大时间 ~ 1e3s）  
<n>：通道号，n 取值 1、2。

➤ **返回格式：**

查询返回指定通道的输出周期，采用科学计数法返回。

➤ **举例：**

:CHANnel1:BASE:PERiod 0.002	设置通道 1 输出周期 2ms
:CHANnel1:BASE:PERiod?	查询返回 2e-3

**:CHANnel<n>:BASE:PHASe**

➤ **命令格式：**

:CHANnel<n>:BASE:PHASe { <phase> }  
:CHANnel<n>:BASE:PHASe?

➤ **功能描述：**

设置指定通道输出相位。  
<phase>表示相位，单位°，范围-360~360。  
<n>：通道号，n 取值 1、2。

➤ **返回格式：**

查询返回指定通道的输出相位。

➤ **举例：**

:CHANnel1:BASE:PHASe 20	设置通道 1 输出相位为 20°
:CHANnel1:BASE:PHASe?	查询返回 20

**:CHANnel<n>:BASE:AMPLitude**

➤ **命令格式：**

:CHANnel<n>:BASE:AMPLitude { <amp> }

:CHANnel<n>:BASE:AMPLitude?

➤ **功能描述：**

设置指定通道输出幅度。

<amp;gt;表示电压，单位当前通道指定单位。1mVpp ~ 当前负载下输出的最大值。

若当前单位为 VPP，当前负载下最大值=当前负载\*20/(50+当前负载)

<n>：通道号，n 取值 1、2。

➤ **返回格式：**

查询返回指定通道的输出幅度，采用科学计数法返回。

➤ **举例：**

:CHANnel1:BASE:AMPLitude 2                      设置通道 1 输出幅度为 2V

:CHANnel1:BASE:AMPLitude?                      查询返回 2e+0

### **:CHANnel<n>:BASE:OFFSet**

➤ **命令格式：**

:CHANnel<n>:BASE:OFFSet { <voltage>}

:CHANnel<n>:BASE:OFFSet?

➤ **功能描述：**

设置指定通道输出直流偏移。

<voltage>表示电压，单位 V。范围为：0~±当前负载下最大直流。

当前负载下的最大直流= 当前负载\*10/(50+当前负载) - 当前交流最小值/2;

交流最小值为 2mVpp,直流模式取 0;

<n>：通道号，n 取值 1、2。

➤ **返回格式：**

查询返回指定通道的输出直流偏移，采用科学计数法返回。

➤ **举例：**

:CHANnel1:BASE:OFFSet 2                      设置通道 1 输出直流偏移为 2V

:CHANnel1:BASE:OFFSet?                      查询返回 2e+0

### **:CHANnel<n>:BASE:HIGH**

➤ **命令格式：**

:CHANnel<n>:BASE:HIGH { <voltage>}

:CHANnel<n>:BASE:HIGH?

➤ **功能描述：**

设置指定通道信号输出高值。

<voltage>表示电压，单位当前通道指定单位。

<n>: 通道号, n 取值 1、2。

➤ **返回格式:**

查询返回指定通道信号输出高值, 采用科学计数法返回。

➤ **举例:**

:CHANnel1:BASE:HIGH 2	设置通道 1 信号输出高值为 2V
:CHANnel1:BASE:HIGH?	查询返回 2e+0

**:CHANnel<n>:BASE:LOW**

➤ **命令格式:**

:CHANnel<n>:BASE:LOW { <voltage>}  
:CHANnel<n>:BASE:LOW?

➤ **功能描述:**

设置指定通道信号输出低值。

<voltage>表示电压, 单位当前通道指定单位。

<n>: 通道号, n 取值 1、2。

➤ **返回格式:**

查询返回指定通道信号输出低值, 采用科学计数法返回。

➤ **举例:**

:CHANnel1:BASE:LOW 2	设置通道 1 信号输出低值为 2V
:CHANnel1:BASE:LOW?	查询返回 2e+0

**:CHANnel<n>:BASE:DUTY**

➤ **命令格式:**

:CHANnel<n>:BASE:DUTY { <duty>}  
:CHANnel<n>:BASE:DUTY?

➤ **功能描述:**

设置指定通道信号输出占空比。

<duty>表示占空比, 单位%, 范围 0~100。

<n>: 通道号, n 取值 1、2。

➤ **返回格式:**

查询返回指定通道信号输出占空比。

➤ **举例:**

:CHANnel1:BASE:DUTY 20	设置通道 1 信号输出占空比为 20%
:CHANnel1:BASE:DUTY?	查询返回 20

## **:CHANnel<n>:BASE:ARB**

### ➤ **命令格式:**

:CHANnel<n>:BASE:ARB <source>,<filename>

:CHANnel<n>:BASE:ARB?

### ➤ **功能描述:**

设置指定通道加载基波任意波源下某文件任意波形数据。

<n>: 通道号, n 取值 1、2。

<source>: {INTernal|EXTernal|USER}, 分别内部、外部、自定义三种。

<filename>: 任意波形文件名称。

### ➤ **举例:**

:CHANnel1:BASE:ARB INTernal, "test.bsv"

## **:CHANnel<n>:BASE:ARB:SAMPLing**

### ➤ **命令格式:**

:CHANnel<n>:BASE:ARB:SAMPLing { <sampling>}

:CHANnel<n>:BASE:ARB:SAMPLing?

### ➤ **功能描述:**

设置指定通道任意波信号采样率。

注意: 此指令只对任意波逐点模式有作用。

<n>: 通道号, n 取值 1、2。

{ <sampling>}: 表示采样率, 单位 Sa/s, 范围 1e-6~625e+6Sa/s。

### ➤ **举例:**

:CHANnel1:BASE:ARB:SAMPLing 1000 设置通道 1 任意波采样率为 1000Sa/s

:CHANnel1:BASE:ARB:SAMPLing? 查询返回 1e3

## **:CHANnel<n>:RAMP:SYMMetry**

### ➤ **命令格式:**

:CHANnel<n>:RAMP:SYMMetry { < symmetry >}

:CHANnel<n>:RAMP:SYMMetry?

### ➤ **功能描述:**

设置指定通道斜坡信号输出对称度。

< symmetry >表示对称度, 单位%, 范围 0~100。

<n>: 通道号, n 取值 1、2。

### ➤ **返回格式:**

查询返回指定通道斜坡信号输出对称度。

➤ **举例：**

:CHANnel1:RAMP:SYMMetry 20	设置通道 1 斜波信号对称度为 20%
:CHANnel1:RAMP:SYMMetry?	查询返回 20

**:CHANnel<n>:PULSe:RISe**

➤ **命令格式：**

:CHANnel<n>:PULSe:RISe {<width>}  
:CHANnel<n>:PULSe:RISe?

➤ **功能描述：**

设置指定通道信号脉冲波上升沿脉宽。  
<width>表示脉宽，单位 S。  
<n>：通道号，n 取值 1、2。

➤ **返回格式：**

查询返回指定通道信号脉冲波上升沿脉宽，采用科学计数法返回。

➤ **举例：**

:CHANnel1:PULSe:RISe 0.002	设置通道 1 信号上升沿脉宽为 2ms
:CHANnel1:PULSe:RISe?	查询返回 2e-3

**:CHANnel<n>:PULSe:FALL**

➤ **命令格式：**

:CHANnel<n>:PULSe:FALL {<width>}  
:CHANnel<n>:PULSe:FALL?

➤ **功能描述：**

设置指定通道信号脉冲波下降沿脉宽。  
<width>表示脉宽，单位 S。  
<n>：通道号，n 取值 1、2。

➤ **返回格式：**

查询返回指定通道信号脉冲波下降沿脉宽，采用科学计数法返回。

➤ **举例：**

:CHANnel1:PULSe:FALL 0.002	设置通道 1 信号下降沿脉宽为 2ms
:CHANnel1:PULSe:FALL?	查询返回 2e-3

**:CHANnel<n>:PRBS:EDGEtime**

➤ **命令格式：**

:CHANnel<n>:PRBS:EDGEtime <time>  
:CHANnel<n>:PRBS:EDGEtime ?

➤ **功能描述：**

设置指定伪随机波边沿时间值，该指令只对伪随机波形有效。

< time >表示边沿时间，单位为 s

<n>：通道号，n 取值 1、2。

➤ **返回格式：**

查询返回指定伪随机波边沿时间值，采用科学计数法返回。

➤ **举例：**

:CHANnel1:PRBS:EDGEtime 1            设置通道 1 伪随机波边沿时间为 1s

:CHANnel1:PRBS:EDGEtime?            查询返回 1e+0

### **:CHANnel<n>:PRBS:BITRatio**

➤ **命令格式：**

:CHANnel<n>:PRBS:BITRatio <ratio>

:CHANnel<n>:PRBS:BITRatio?

➤ **功能描述：**

设置指定伪随机波码率值，该指令只对伪随机波有效。

< ratio >表示码率，单位为 bps

<n>：通道号，n 取值 1、2。

➤ **返回格式：**

查询返回指定伪随机波码率值，采用科学计数法返回。

➤ **举例：**

:CHANnel1:PRBS:BITRatio 1000000            设置通道 1 码率 1Mbps

:CHANnel1:PRBS:BITRatio?            查询返回 1e+6

### **:CHANnel<n>:NOISe:BANDwith**

➤ **命令格式：**

:CHANnel<n>:NOISe:BANDwith {<width>}

:CHANnel<n>:NOISe:BANDwith?

➤ **功能描述：**

设置指定通道噪声信号带宽。

<width>表示带宽，单位 Hz。

<n>：通道号，n 取值 1、2。

➤ **返回格式：**

查询返回指定通道噪声信号带宽，采用科学计数法返回。

➤ **举例：**

:CHANnel1:NOISe:BANDwith 2MHz            设置通道 1 噪声信号带宽 2MHz  
:CHANnel1:NOISe:BANDwith?                查询返回 2e+6

### **:CHANnel<n>:HARMonic:TYPe?**

➤ **命令格式:**

:CHANnel<n>:HARMonic:TYPe {ODD|EVEN|ALL|USER}  
:CHANnel<n>:HARMonic:TYPe?

➤ **功能描述:**

设置指定通道谐波类型。

<n>: 通道号, n 取值 1、2。

➤ **返回格式:**

查询返回指定通道谐波类型。

➤ **举例:**

:CHANnel1:HARMonic:TYPe ODD            设置通道 1 谐波类型为奇次谐波  
:CHANnel1:HARMonic:TYPe?                查询返回 ODD

### **:CHANnel<n>:HARMonic:TOTal:ORDer?**

➤ **命令格式:**

:CHANnel<n>:HARMonic:TOTal:ORDer <order>  
:CHANnel<n>:HARMonic:TOTal:ORDer?

➤ **功能描述:**

设置指定通道最大谐波次数。

< order >: 谐波次数, 范围 2~16。

<n>: 通道号, n 取值 1、2。

➤ **返回格式:**

查询返回指定通道最大谐波次数, 返回整型数据。

➤ **举例:**

:CHANnel1:HARMonic:TOTal:ORDer 2            设置通道 1 最大谐波 2 次  
:CHANnel1:HARMonic:TOTal:ORDer?            查询返回 2

### **:CHANnel<n>:HARMonic:USER:TYPe?**

➤ **命令格式:**

:CHANnel<n>:HARMonic:USER:TYPe #H<order>  
:CHANnel<n>:HARMonic:USER:TYPe?

➤ **功能描述:**

设置指定通道自定义谐波类型。



< order >: 自定义谐波类型, #H 代表 16 进制数。X0111 1111 1111 1111 位分别表示谐波开关。

<n>: 通道号, n 取值 1、2。

➤ **返回格式:**

查询返回指定通道自定义谐波类型, 返回整型数据。

➤ **举例:**

:CHANnel1:HARMonic:USER:TYPe #H7FFF 设置通道 1 自定义谐波类型

:CHANnel1:HARMonic:USER:TYPe? 查询返回 32767

### **:CHANnel<n>:HARMonic:ORDER<m>:AMPLitude?**

➤ **命令格式:**

:CHANnel<n>:HARMonic:ORDER<m>:AMPLitude <amp>

:CHANnel<n>:HARMonic:ORDER<m>:AMPLitude?

➤ **功能描述:**

设置指定通道下指定谐波次数的幅度值。

< amp >: 幅度值, 单位 Vpp。

<n>: 通道号, n 取值 1、2。

<m>: 谐波次数, m 取值 2~16。

➤ **返回格式:**

查询返回指定通道下指定谐波次数的幅度值, 采用科学计数法返回。

➤ **举例:**

:CHANnel1:HARM:ORDER2:AMPL 0.02 设置通道 1 下 2 次谐波的幅度值 20mVpp

:CHANnel1:HARM:ORDER2:AMPL? 查询返回 2e-2

### **:CHANnel<n>:HARMonic:ORDER<m>:PHASe?**

➤ **命令格式:**

:CHANnel<n>:HARMonic:ORDER<m>:PHASe <phase>

:CHANnel<n>:HARMonic:ORDER<m>:PHASe?

➤ **功能描述:**

设置指定通道下指定谐波次数的相位值。

<phase>: 相位值, 单位°。

<n>: 通道号, n 取值 1、2。

<m>: 谐波次数, m 取值 2~16。

➤ **返回格式:**

查询返回指定通道下指定谐波次数的相位值, 采用科学计数法返回。

➤ **举例:**

:CHANnel1:HARM:ORDer2:PHASe 20            设置通道 1 下 2 次谐波的相位值 20°  
:CHANnel1:HARM:ORDer2:PHASe?            查询返回 2e+1

### **:CHANnel<n>:ARB:MODE**

➤ **命令格式:**

:CHANnel<n>:ARB:MODE {DDS | POINTS }  
:CHANnel<n>:ARB:MODE?

➤ **功能描述:**

设置指定通道任意波输出模式，分别为 DDS 和逐点模式。

<n>: 通道号，n 取值 1、2。

➤ **返回格式:**

查询返回指定通道任意波模式。

➤ **举例:**

:CHANnel1:ARB:MODE DDS                    设置通道 1 任意波模式 DDS 输出模式  
:CHANnel1:ARB:MODE?                    查询返回 DDS

### **:CHANnel<n>:ARB:FILTer**

➤ **命令格式:**

:CHANnel<n>:ARB:FILTer {ZEROHOLD | LINE }  
:CHANnel<n>:ARB:FILTer?

➤ **功能描述:**

设置指定通道任意波输出插值方式，分别为零阶保持和线性插值。

<n>: 通道号，n 取值 1、2。

➤ **返回格式:**

查询返回指定通道任意波输出插值方式。

➤ **举例:**

:CHANnel1:ARB:FILTer LINE                设置通道 1 任意波以线性插值方式输出  
:CHANnel1:ARB:FILTer?                查询返回 LINE

### **:CHANnel<n>: EXP:EXPStart**

➤ **命令格式:**

:CHANnel<n>:EXP:EXPStart { < start value> }  
:CHANnel<n>:EXP:EXPStart?

➤ **功能描述:**

设置指定通道表达式信号输出的开始值（最小值）。

< start value>表示开始值（最小值），以弧度值输入。

<n>: 通道号, n 取值 1、2。

➤ **返回格式:**

查询返回指定通道表达式信号输出的开始值, 以科学计数法返回。

➤ **举例:**

:CHANnel1:EXP:EXPStart 0.1                    设置通道 1 表达式信号开始值为 0.1

:CHANnel1:EXP:EXPStart?                    查询返回 1e-01

**:CHANnel<n>: EXP:EXPEnd**

➤ **命令格式:**

:CHANnel<n>:EXP:EXPEnd { < end value>}

:CHANnel<n>:EXP:EXPEnd?

➤ **功能描述:**

设置指定通道表达式信号输出的结束值 (最大值)。

< end value>表示结束值, 以弧度值输入。

<n>: 通道号, n 取值 1、2。

➤ **返回格式:**

查询返回指定通道表达式信号输出的结束值 (最大值), 以科学计数法返回。

➤ **举例:**

:CHANnel1:EXP:EXPEnd 3.1415926            设置通道 1 表达式信号的结束值 (最大值) 为 3.1415926

:CHANnel1:EXP:EXPEnd?                    查询返回 3.1415926e+00

**:CHANnel<n>: EXP:EXPStr**

➤ **命令格式:**

:CHANnel<n>:EXP:EXPStr { < expstr>}

:CHANnel<n>:EXP:EXPStr?

➤ **功能描述:**

设置指定通道表达式信号的表达式式子。自变量只能写 x, 表达式中的字母全小写。

< expstr >表示表达式式子, 以字符串形式输入。

支持的函数有: sin(x),cos(x),tan(x),sinc(x), abs(x),lg(x), ln(x),sqrt(x),acos(x),

asin(x),atan(x),sinh(x), tanh(x),ceil(x),cosh(x),exp(x),fabs(x),floor(x)。

在数字变量之间不能省略操作符。比如 3\*x 不能写作 3x。表达式中不能存在空格。

<n>: 通道号, n 取值 1、2。

➤ **返回格式:**

查询返回指定通道表达式信号的表达式式子, 以字符串形式返回。

➤ **举例:**

:CHANnel1:EXP:EXPStr "sin(x)"设置通道 1 表达式信号的表达式式子为 sin(x)

:CHANnel1:EXP:EXPStr? 查询返回 sin(x)

### **:CHANnel<n>:DPULse:DELay**

➤ **命令格式:**

:CHANnel<n>:DPULse:DELay <delay>

:CHANnel<n>:DPULse:DELay?

➤ **功能描述:**

设置指定通道脉冲延时。

< delay >: 脉冲延时值, 单位 S。

<n>: 通道号, n 取值 1、2。

➤ **返回格式:**

查询返回指定通道脉冲延时, 采用科学计数法返回。

➤ **举例:**

:CHANnel1:DPULse:DELay 0.02 设置通道 1 下双脉冲脉冲延时值 20ms

:CHANnel1:DPULse:DELay? 查询返回 2e-2

### **:CHANnel<n>:DPULse:TOTal:ORDer**

➤ **命令格式:**

:CHANnel<n>:DPULse:TOTal:ORDer <order>

:CHANnel<n>:DPULse:TOTal:ORDer?

➤ **功能描述:**

设置指定通道双脉冲最大脉冲数。

< order >: 双脉冲最大脉冲数, 范围 2~30。

<n>: 通道号, n 取值 1、2。

➤ **返回格式:**

查询返回指定通道双脉冲最大脉冲数, 返回整型数据。

➤ **举例:**

:CHANnel1:DPULse:TOTal:ORDer 5 设置通道 1 双脉冲最大脉冲数为 5 个

:CHANnel1:DPULse:TOTal:ORDer? 查询返回 5

### **:CHANnel<n>:DPULse:ORDer**

➤ **命令格式:**

:CHANnel<n>:DPULse:ORDer <order>

:CHANnel<n>:DPULse:ORDer?

➤ **功能描述:**

设置指定通道当前双脉冲序数。

< order >: 双脉冲脉冲序数, 范围 0~29。

<n>: 通道号, n 取值 1、2。

➤ **返回格式:**

查询返回指定通道当前双脉冲序数, 返回整型数据。

➤ **举例:**

:CHANnel1:DPULse:ORDer 1      设置通道 1 当前双脉冲序数为 1

:CHANnel1:DPULse:ORDer?          查询返回 1

**:CHANnel<n>:DPULse:ORDer<m>:PULSewidth**

➤ **命令格式:**

:CHANnel<n>:DPULse:ORDer<m>:PULSewidth <width>

:CHANnel<n>:DPULse:ORDer<m>:PULSewidth?

➤ **功能描述:**

设置指定通道下指定双脉冲序数的脉冲宽度值。

< width >: 脉冲宽度值, 单位 S。

<n>: 通道号, n 取值 1、2。

<m>: 双脉冲序数, m 取值 0~29。

➤ **返回格式:**

查询返回指定通道下指定双脉冲序数的脉冲宽度值, 采用科学计数法返回。

➤ **举例:**

:CHANnel1:DPULse:ORDer2:PULSewidth 0.02      设置通道 1 下第 3 个脉冲的宽度值 20ms

:CHANnel1:DPULse:ORDer2:PULSewidth?          查询返回 2e-2

**:CHANnel<n>:DPULse:ORDer<m>:GAP**

➤ **命令格式:**

:CHANnel<n>:DPULse:ORDer<m>:GAP <gap>

:CHANnel<n>:DPULse:ORDer<m>:GAP?

➤ **功能描述:**

设置指定通道下指定双脉冲序数的脉冲间隙值。

< gap >: 脉冲间隙值, 单位 S。

<n>: 通道号, n 取值 1、2。

<m>: 双脉冲序数, m 取值 0~29。

➤ **返回格式:**

查询返回指定通道下指定双脉冲序数的脉冲间隙值, 采用科学计数法返回。

➤ **举例：**

:CHANnel1:DPULse:ORDer2:GAP 0.02      设置通道 1 下第 3 个脉冲的间隙值 20ms  
:CHANnel1:DPULse:ORDer2:GAP?      查询返回 2e-2

**:CHANnel<n>:DPULse:ORDer<m>:RISe**

➤ **命令格式：**

:CHANnel<n>:DPULse:ORDer<m>:RISe <rise>  
:CHANnel<n>:DPULse:ORDer<m>:RISe?

➤ **功能描述：**

设置指定通道下指定双脉冲序数的脉冲上升沿值。

< rise >: 脉冲上升沿值，单位 S。

<n>: 通道号，n 取值 1、2。

<m>: 双脉冲序数，m 取值 0~29。

➤ **返回格式：**

查询返回指定通道下指定双脉冲序数的脉冲上升沿值，采用科学计数法返回。

➤ **举例：**

:CHANnel1:DPULse:ORDer2:RISe 0.02      设置通道 1 下第 3 个脉冲的上升沿值 20ms  
:CHANnel1:DPULse:ORDer2:RISe?      查询返回 2e-2

**:CHANnel<n>:DPULse:ORDer<m>:FALI**

➤ **命令格式：**

:CHANnel<n>:DPULse:ORDer<m>:FALI <fall>  
:CHANnel<n>:DPULse:ORDer<m>:FALI?

➤ **功能描述：**

设置指定通道下指定双脉冲序数的脉冲下降沿值。

< fall >: 脉冲下降沿值，单位 S。

<n>: 通道号，n 取值 1、2。

<m>: 双脉冲序数，m 取值 0~29。

➤ **返回格式：**

查询返回指定通道下指定双脉冲序数的脉冲下降沿值，采用科学计数法返回。

➤ **举例：**

:CHANnel1:DPULse:ORDer2:FALI 0.02      设置通道 1 下第 3 个脉冲的下降沿值 20ms  
:CHANnel1:DPULse:ORDer2:FALI?      查询返回 2e-2

## **:CHANnel<n>:MODulate:TYPE**

### ➤ **命令格式:**

:CHANnel<n>:MODulate:TYPE <type>

:CHANnel<n>:MODulate:TYPE?

### ➤ **功能描述:**

设置指定通道信号调制类型。

<type>:

{AM|DSBAM|QAM|ASK|FM|FSK|ThreeFSK|FourFSK|PM|PSK|BPSK|QPSK|OSK|PWM|SUM}

分别为调幅、双边调幅、正交调制、幅移键控、调频、频移键控、三频键控、四频键控、调相、相移键控、双相移键控、四相移键控、震荡键控、脉宽调制、总和调制。

<n>: 通道号, n 取值 1、2。

### ➤ **返回格式:**

查询返回指定通道信号调制类型。

### ➤ **举例:**

:CHANnel1:MODulate:TYPE AM

设置通道 1 信号 AM 调制

:CHANnel1:MODulate:TYPE?

查询返回 AM

## **:CHANnel<n>:MODulate:WAVE**

### ➤ **命令格式:**

:CHANnel<n>:MODulate:WAVE { SINE|SQUare|UPRamp|DNRamp|ARBINOISE }

:CHANnel<n>:MODulate:WAVE?

### ➤ **功能描述:**

设置指定通道信号调制波类型, 分别为正弦波、方波、上三角、下三角、任意波、噪声。

<n>: 通道号, n 取值 1、2。

### ➤ **返回格式:**

查询返回指定通道信号调制波类型。

### ➤ **举例:**

:CHANnel1:MODulate:WAVE SINE

设置通道 1 信号调制波类型为正弦波

:CHANnel1:MODulate:WAVE?

查询返回 SINE

## **:CHANnel<n>:MODulate:SOURce**

### ➤ **命令格式:**

:CHANnel<n>:MODulate:SOURce { INTERNAL|EXTERNAL }

:CHANnel<n>:MODulate:SOURce?

➤ **功能描述:**

设置指定通道调制源，分别内部、外部两种。

<n>: 通道号，n 取值 1、2。

➤ **返回格式:**

查询返回指定通道调制源。

➤ **举例:**

:CHANnel1:MODulate:SOURce INTernal	设置通道一调制源为内部
:CHANnel1:MODulate:SOURce?	查询返回 INTernal

### **:CHANnel<n>:MODulate:FREQuency**

➤ **命令格式:**

:CHANnel<n>:MODulate:FREQuency {<freq>}

:CHANnel<n>:MODulate:FREQuency?

➤ **功能描述:**

设置指定通道信号调制频率。

<freq>表示频率，单位 Hz。

<n>: 通道号，n 取值 1、2。

➤ **返回格式:**

查询返回指定通道信号调制频率，返回采样科学计数法表示。

➤ **举例:**

:CHANnel1:MODulate:FREQuency 2000	设置通道 1 信号调制频率 2KHz
:CHANnel1:MODulate:FREQuency?	查询返回 2e+3

### **:CHANnel<n>:MODulate:IQMap**

➤ **命令格式:**

:CHANnel<n>:MODulate: IQMap {<IQ TYPE>}

:CHANnel<n>:MODulate: IQMap?

➤ **功能描述:**

设置指定 QAM 的 IQ 类型可以为：

QAM4, QAM8, QAM16, QAM32, QAM64, QAM128, QAM256。

< IQ TYPE >表示 IO 映射类型。

<n>: 通道号，n 取值 1、2。

➤ **返回格式:**

查询返回指定通道的 IQ 类型



➤ **举例：**

:CHANnel1:MODulate:IQMap QAM32      设置通道 1 调制 IQ 映射为 QAM32  
:CHANnel1:MODulate:IQMap?            查询返回 QAM32

### **:CHANnel<n>:MODulate:ARB**

➤ **命令格式：**

:CHANnel<n>:MODulate:ARB <source>,<filename>  
:CHANnel<n>:MODulate:ARB?

➤ **功能描述：**

设置指定通道加载调制任意波源下某文件任意波形数据。注意在选用外部时，需要在外部存储设备的根目录上建立一个 ArbWave 目录，将任意波放入这个目录。

<n>：通道号，n 取值 1、2。

<source>：{INTernal|EXTernal|USER}，分别内部、外部、自定义三种。

<filename>：任意波形文件名称。

➤ **举例：**

:CHANnel1:MODulate:ARB INTernal, "test.bsv"

### **:CHANnel<n>:MODulate:DEPTTh**

➤ **命令格式：**

:CHANnel<n>:MODulate:DEPTTh { <depth>  
:CHANnel<n>:MODulate:DEPTTh?

➤ **功能描述：**

设置指定通道调制深度。

<depth>表示调制深度，单位%。0% ~ 100%，其中 AM 调制深度为 0% ~ 120%

<n>：通道号，n 取值 1、2。

➤ **返回格式：**

查询返回指定通道调制深度。

➤ **举例：**

:CHANnel1:MODulate:DEPTTh 50            设置通道一调制深度为 50%  
:CHANnel1:MODulate:DEPTTh?            查询返回 50

### **:CHANnel<n>:MODulate:BITRatio**

➤ **命令格式：**

:CHANnel<n>:MODulate:BITRatio <ratio>  
:CHANnel<n>:MODulate:BITRatio?

➤ **功能描述：**

设置指定通道速率值，该指令只对具备速率功能的波形有效。

< ratio >表示位率，单位为 Hz

<n>：通道号，n 取值 1、2。

➤ **返回格式：**

查询返回指定通道速率值，采用科学计数法返回。

➤ **举例：**

:CHANnel1:MODulate:BITRatio 100000	设置通道 1 速率 100KHz
:CHANnel1:MODulate:BITRatio?	查询返回 1e+5

### **:CHANnel<n>:MODulate:RATio**

➤ **命令格式：**

:CHANnel<n>:MODulate:RATio <ratio>

:CHANnel<n>:MODulate:RATio?

➤ **功能描述：**

设置指定通道调制速率值，该指令只对具备速率功能调制类型有效。

< ratio >表示速率，单位为 Hz

<n>：通道号，n 取值 1、2。

➤ **返回格式：**

查询返回指定通道调制速率值，采用科学计数法返回。

➤ **举例：**

:CHANnel1:MODulate:RATio 100	设置通道 1 调制速率 100Hz
:CHANnel1:MODulate:RATio?	查询返回 1e+2

### **:CHANnel<n>:FM:FREQuency:DEV**

➤ **命令格式：**

:CHANnel<n>:FM:FREQuency:DEV { <freq>}

:CHANnel<n>:FM:FREQuency:DEV?

➤ **功能描述：**

设置指定通道频率偏差。

<freq>表示频率偏移，单位 Hz。0Hz ~ 当前基波频率

<n>：通道号，n 取值 1、2。

➤ **返回格式：**

查询返回指定通道频率偏移，采用科学计数法返回数据。

➤ **举例：**

:CHANnel1:FM:FREQuency:DEV 2000	设置通道一频率偏移为 2KHz
:CHANnel1:FM:FREQuency:DEV?	查询返回 2e+3

### **:CHANnel<n>:PM:PHASe:DEV**

➤ **命令格式:**

:CHANnel<n>:PM:PHASe:DEV { <phase>}

:CHANnel<n>:PM:PHASe:DEV?

➤ **功能描述:**

设置指定通道输出相位偏差。

< phase >表示相位偏移，单位°，范围 0~360。

<n>: 通道号，n 取值 1、2。

➤ **返回格式:**

查询返回指定通道输出相位偏移。

➤ **举例:**

:CHANnel1:PM:PHASe:DEV 30                    设置通道一相位偏移 30°

:CHANnel1:PM:PHASe:DEV?                    查询返回 30

### **:CHANnel<n>:PWM:DUTY:DEV**

➤ **命令格式:**

:CHANnel<n>:PWM:DUTY:DEV { <duty>}

:CHANnel<n>:PWM:DUTY:DEV?

➤ **功能描述:**

设置指定通道输出脉宽调制下占空比偏差。

< duty >表示占空比偏差，单位%，范围 0~100。

<n>: 通道号，n 取值 1、2。

➤ **返回格式:**

查询返回指定通道脉宽调制下占空比偏差，以科学计数法返回数据。

➤ **举例:**

:CHANnel1:PWM:DUTY:DEV 10                    设置通道一占空比偏差 10%

:CHANnel1:PWM:DUTY:DEV?                    查询返回 1e+1

### **:CHANnel<n>:FSK:FREQuency<m>**

➤ **命令格式:**

:CHANnel<n>:FSK:FREQuency<m> { <freq>}

:CHANnel<n>:FSK:FREQuency<m>?

➤ **功能描述:**

设置指定通道输出多频移键控跳频频率，必须要提前指定调制方式，此指令才可生效。

< freq >表示频率，单位 Hz。

<n>: 通道号, n 取值 1、2。

<m>: 频率序号, 2FSK 时取值 1; 3FSK 时取值 1、2; 4FSK 时取值 1、2、3;

➤ **返回格式:**

查询返回指定通道输出跳频频率, 以科学计数法返回数据。

➤ **举例:**

:CHANnel1:FSK:FREQ1 2000                      设置通道一输出跳频频率 2KHz

:CHANnel1:FSK:FREQ1?                      查询返回 2e+3

**:CHANnel<n>:PSK:PHASe<m>**

➤ **命令格式:**

:CHANnel<n>:PSK:PHASe<m> { < phase > }

:CHANnel<n>:PSK:PHASe<m>?

➤ **功能描述:**

设置指定通道输出多相移键控相位值, 必须要提前指定调制方式, 此指令才可生效。

< phase >表示相位, 单位°, 范围-360~360。

<n>: 通道号, n 取值 1、2。

<m>: 相位序号, PSK 取值 1; BPSK 时取值 1、2; QPSK 时取值 1、2、3、4;

➤ **返回格式:**

查询返回指定通道相移键控相位值, 以科学计数法返回数据。

➤ **举例:**

:CHANnel1:PSK:PHAS1 90                      设置通道一输出相位 90°

:CHANnel1:PSK:PHAS1?                      查询返回 9e+1

**:CHANnel<n>:OSK:TIME**

➤ **命令格式:**

:CHANnel<n>:OSK:TIME { <time> }

:CHANnel<n>:OSK:TIME?

➤ **功能描述:**

设置指定通道调制模式下震荡键控的震荡时间。

< time >表示震荡时间, 单位 S。

<n>: 通道号, n 取值 1、2。

➤ **返回格式:**

查询返回指定通道调制模式下震荡键控的震荡时间, 以科学计数法返回数据。

➤ **举例:**

:CHANnel1:OSK:TIME 0.002                      设置通道一震荡键控的震荡时间 2ms

:CHANnel1:OSK:TIME?

查询返回 2e-3

### **:CHANnel<n>:TRIGger:OUTEdge**

➤ **命令格式:**

:CHANnel<n>:TRIGger:OUTEdge { RISE|FALL}

:CHANnel<n>:TRIGger:OUTEdge?

➤ **功能描述:**

设置指定通道触发输出边沿，该指令只对扫频和猝发功能有效。并且只有在扫频和猝发的触发输出功能打开才有效。

<n>: 通道号, n 取值 1、2。

➤ **返回格式:**

查询返回指定通道触发输出边沿。

➤ **举例:**

:CHANnel1:TRIGger:OUTEdge RISE设置通道 1 上升沿触发输出模式

:CHANnel1:TRIGger:OUTEdge?            查询返回 RISE

## 扫频

### **:CHANnel<n>:SWEep:TYPE**

➤ **命令格式:**

:CHANnel<n>:SWEep:TYPE { LINE|LOG|STEP}

:CHANnel<n>:SWEep:TYPE?

➤ **功能描述:**

设置指定通道扫频模式，分别为线性扫频、对数扫频、步进扫频。

<n>: 通道号, n 取值 1、2。

➤ **返回格式:**

查询返回指定通道扫频模式。

➤ **举例:**

:CHANnel1:SWEep:TYPE LINE            设置通道一线性扫频模式

:CHANnel1:SWEep:TYPE?            查询返回 LINE

### **:CHANnel<n>:SWEep:FREQuency:START**

➤ **命令格式:**

:CHANnel<n>:SWEep:FREQuency:START <freq>

:CHANnel<n>:SWEep:FREQuency:START?

➤ **功能描述:**

设置指定通道扫频的起始频率。

< freq >表示频率，单位 Hz。

<n>: 通道号，n 取值 1、2。

➤ **返回格式:**

查询返回指定通道扫频的起始频率，以科学计数法返回数据。

➤ **举例:**

:CHANnel1:SWE:FREQ:STAR 2000	设置通道一输出扫频的起始频率 2KHz
:CHANnel1:SWE:FREQ:STAR?	查询返回 2e+3

### **:CHANnel<n>:SWEep:FREQuency:STOP**

➤ **命令格式:**

:CHANnel<n>:SWEep:FREQuency:STOP <freq>

:CHANnel<n>:SWEep:FREQuency:STOP?

➤ **功能描述:**

设置指定通道扫频的截止频率。

< freq >表示频率，单位 Hz。

<n>: 通道号，n 取值 1、2。

➤ **返回格式:**

查询返回指定通道输出扫频的截止频率，以科学计数法返回数据。

➤ **举例:**

:CHANnel1:SWE:FREQ:STOP 2000	设置通道一输出扫频的截止频率 2KHz
:CHANnel1:SWE:FREQ:STOP?	查询返回 2e+3

### **:CHANnel<n>:SWEep:TIME**

➤ **命令格式:**

:CHANnel<n>:SWEep:TIME <time>

:CHANnel<n>:SWEep:TIME?

➤ **功能描述:**

设置指定通道扫频时的扫描时间。

< time >表示时间，单位 S。范围为：1ms ~ 500s

<n>: 通道号，n 取值 1、2。

➤ **返回格式:**

查询返回指定通道扫频时的扫描时间，以科学计数法返回数据。

➤ **举例:**

:CHANnel1:SWEep:TIME 2	置通道一扫频时的扫描时间为 2S
------------------------	------------------

:CHANnel1:SWEEP:TIME?

查询返回 2e+0

### **:CHANnel<n>:SWEep:HOLD**

➤ **命令格式:**

:CHANnel<n>:SWEEP:HOLD <time>

:CHANnel<n>:SWEEP:HOLD?

➤ **功能描述:**

设置指定通道扫频驻留时间，该指令只在步进扫频时有效。

< time >表示时间，单位 S。

<n>: 通道号，n 取值 1、2。

➤ **返回格式:**

查询返回指定通道扫频驻留时间，以科学计数法返回数据。

➤ **举例:**

:CHANnel1:SWEEP:HOLD 2

置通道一扫频时的驻留时间为 2S

:CHANnel1:SWEEP:HOLD?

查询返回 2e+0

### **:CHANnel<n>:SWEep:STEPs**

➤ **命令格式:**

:CHANnel<n>:SWEEP:STEPs <steps>

:CHANnel<n>:SWEEP:STEPs?

➤ **功能描述:**

设置指定通道步进扫频时的总步数，该指令只在步进扫频时有效。

< steps >: 步数

<n>: 通道号，n 取值 1、2。

➤ **返回格式:**

查询返回指定通道扫频时的扫描步数，返回整型数据。

➤ **举例:**

:CHANnel1:SWEEP:STEPs 10

设置指定通道步进扫频 10 步

:CHANnel1:SWEEP:STEPs?

查询返回 10

### **:CHANnel<n>:SWEep:TRIGger**

➤ **命令格式:**

:CHANnel<n>:SWEep:TRIGger

➤ **功能描述:**

触发指定通道扫频输出，该参数仅在触发模式设置为手动触发时候有效。

➤ **举例:**

## 猝发

### :CHANnel<n>:BURSt:TYPe

➤ **命令格式:**

```
:CHANnel<n>:BURSt:TYPe {NCYC|GATe|INFinIt}
:CHANnel<n>:BURSt:TYPe?
```

➤ **功能描述:**

设置指定通道猝发类型，分别为 N 周期、门控、无限。

<n>: 通道号，n 取值 1、2。

➤ **返回格式:**

查询返回指定通道猝发类型。

➤ **举例:**

```
:CHANnel1:BURSt:TYPe NCYC          设置通道一通道 N 周期猝发
:CHANnel1:BURSt:TYPe?              查询返回 2e+0
```

### :CHANnel<n>:BURSt:PERiod

➤ **命令格式:**

```
:CHANnel<n>:BURSt:PERiod <period >
:CHANnel<n>:BURSt:PERiod?
```

➤ **功能描述:**

设置指定通道猝发周期。

< period >表示时间，单位 S。

<n>: 通道号，n 取值 1、2。

➤ **返回格式:**

查询返回指定通道猝发周期，以科学计数法返回数据。

➤ **举例:**

```
:CHANnel1:BURSt:PERiod 0.005       设置通道一猝发周期 5ms
:CHANnel1:BURSt:PERiod?            查询返回 5e-3
```

### :CHANnel<n>:BURSt:PHASe

➤ **命令格式:**

```
:CHANnel<n>:BURSt:PHASe <phase>
:CHANnel<n>:BURSt:PHASe?
```



➤ **功能描述：**

设置指定通道猝发起始相位。

< phase >表示相位，单位°。范围为：0 ~ 360

<n>：通道号，n 取值 1、2。

➤ **返回格式：**

查询返回指定通道猝发起始相位，以科学计数法返回数据。

➤ **举例：**

:CHANnel1:BURSt:PHASe 18	设置通道一猝发起始相位 18°
:CHANnel1:BURSt:PHASe?	查询返回 1.8e+1

### **:CHANnel<n>:BURSt:CYCLes**

➤ **命令格式：**

:CHANnel<n>:BURSt:CYCLes <cycles>

:CHANnel<n>:BURSt:CYCLes?

➤ **功能描述：**

设置指定通道猝发循环次数。

< cycles >表示循环次数，整型数据。

<n>：通道号，n 取值 1、2。

➤ **返回格式：**

查询返回指定通道猝发循环次数。

➤ **举例：**

:CHANnel1:BURSt:CYCLes 2	设置指定通道猝发循环次数为 2
:CHANnel1:BURSt:CYCLes?	查询返回 2

### **:CHANnel<n>:BURSt:GATe:POLarity**

➤ **命令格式：**

:CHANnel<n>:BURSt:GATe:POLarity {POSitive|NEGative}

:CHANnel<n>:BURSt:GATe:POLarity?

➤ **功能描述：**

设置指定通道门控猝发极性，分别为正极性、负极性。

<n>：通道号，n 取值 1、2。

➤ **返回格式：**

查询返回指定通道门控猝发极性。

➤ **举例：**

:CHANnel1:BURSt:GATe:POLarity POSitive	设置通道一门控猝发极性为正极性
--	-----------------

:CHANnel1:BURSt:GATe:POLarity?

查询返回 POSitive

### **:CHANnel<n>:BURSt:TRIGger**

➤ **命令格式:**

:CHANnel<n>:BURSt:TRIGger

➤ **功能描述:**

触发指定通猝发输出，该参数仅在触发模式设置为手动触发时候有效。

➤ **举例:**

:CHANnel1:BURSt:TRIGger

触发一次猝发信号输出

## **WARB 命令**

用于写任意波形文件指令，包括基本任意波形和调制任意波形写配置。

### **:WARB<n>:MODulate**

➤ **命令格式:**

:WARB<n>:MODulate <arb file>

➤ **功能描述:**

用于写调制任意波形，先发送该指令，然后发送任意波形文件数据到信号源。

<arb file>表示任意波形文件名称，只支持 bsv 文件格式。

➤ **举例:**

:WARB1:MODulate "test.bsv"

写通道一调制任意波形文件

### **:WARB<n>:CARRier**

➤ **命令格式:**

:WARB<n>:CARRier <arb file>

➤ **功能描述:**

用于写基波任意波形，先发送该指令，然后发送任意波形文件数据到信号源。

<arb file>表示任意波形文件名称，只支持 bsv 文件格式。

➤ **举例:**

:WARB1:CARRier "test.bsv"

写通道一基波任意波形文件

## **DISPlay 命令**

用于信号源显示相关信息。

## **:DISPlay:DATA?**

➤ **命令格式:**

:DISPlay:DATA?

➤ **功能描述:**

用于查询当前设备屏幕的图像数据，默认返回 BMP 格式图像数据，返回的图像数据格式由:[:SYSTem:PICTure:FORMat](#) 指令决定。

➤ **返回格式:**

查询返回图像数据，返回的数据符合 IEEE 488.2 #格式的二进制数据。

➤ **举例:**

:DISPlay:DATA?

查询返回图像数据

数据格式: #800012345+图像数据

## 编程说明

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

### 编程准备

编程准备工作仅适用于在 Windows 操作系统下使用 Visual Studio 和 LabVIEW 开发工具进行编程。

首先确认您的电脑上是否已经安装 NI 的 VISA 库（可到 <https://www.ni.com/en-ca/support/downloads/drivers/download.ni-visa.html> 下载），本文中默认安装路径为 C:\Program Files\IVI Foundation\VISA。

通过仪器设备的 USB 或 LAN 接口与 PC 建立通信，请使用 USB 数据线将仪器设备后面板的 USB DEVICE 接口与 PC 的 USB 接口相连，或者使用 LAN 数据线将仪器设备后面板的 LAN 口与 PC 的 LAN 接口相连。

# VISA 编程示例

本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，并结合编程手册的命令实现对仪器设备的控制。通过下面的例子，你可以开发更多应用。

## VC++ 示例

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

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

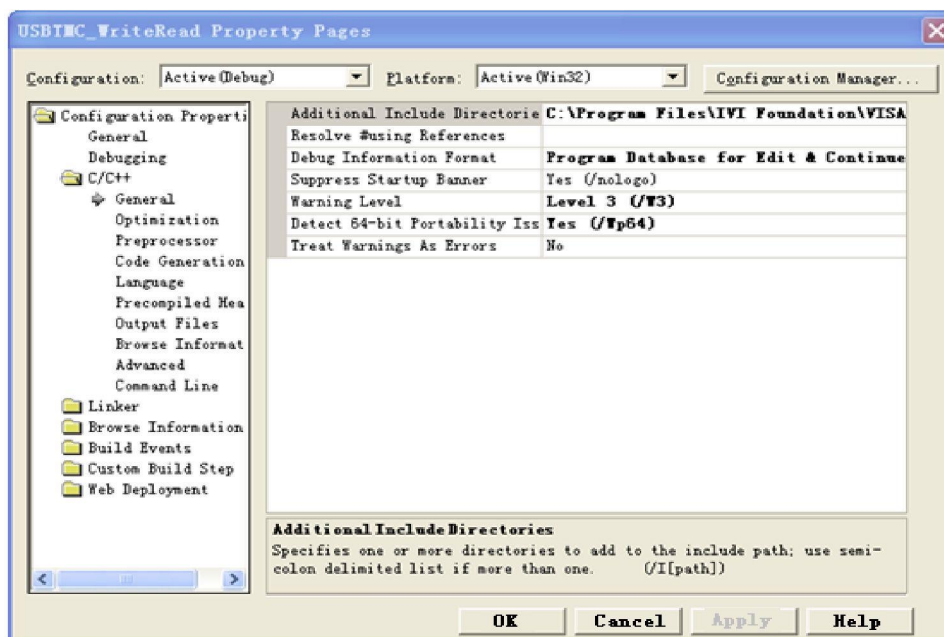
### a) 静态库：

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

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

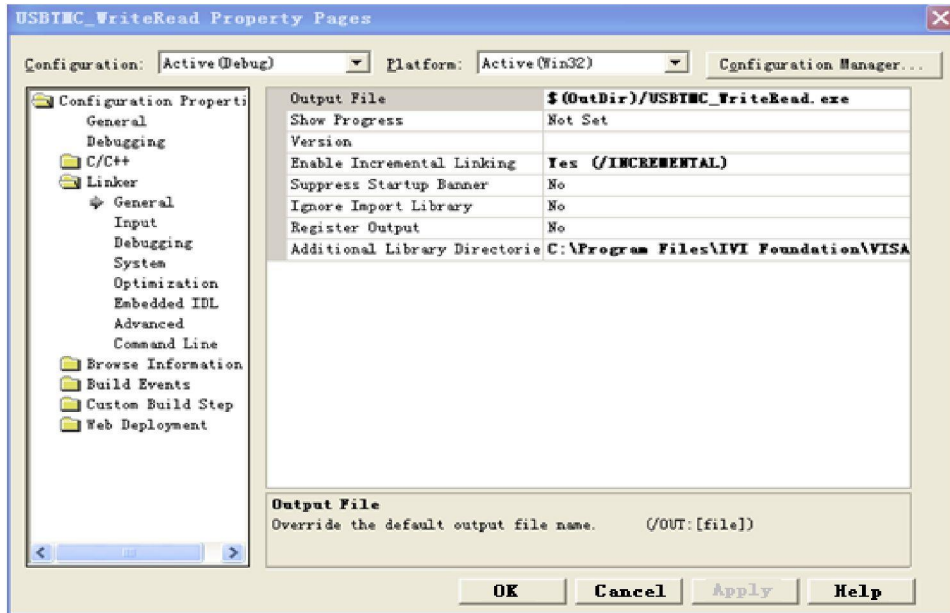
### b) 动态库：

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

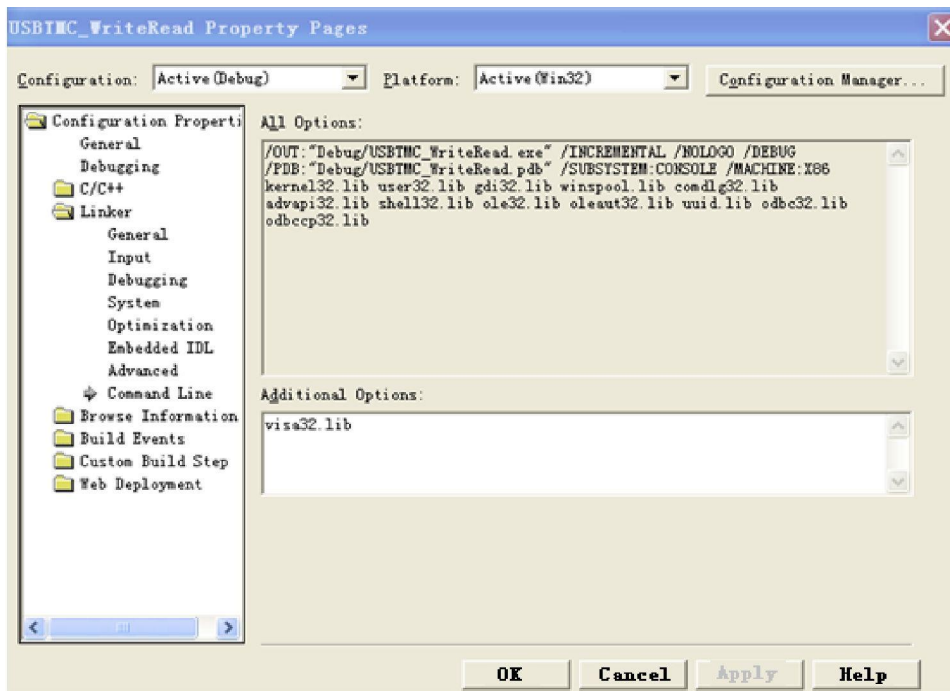


在属性对话框左侧选择"Linker-General",并将"Additional Library Directories"项的值设置为 NI-VISA

的安装路径, (例如: C:\Program Files\IVI Foundation\VISA\WinNT\include), 如下图所示:



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



在 projectname.cpp 文件上添加 visa.h 文件:

```
#include <visa.h>
```

1. 源码:

a) USBTMC 示例

```
int usbtmc_test()
{
    /** This code demonstrates sending synchronous read & write commands
     * to an USB Test & Measurement Class (USBTMC) instrument using NI-VISA
     * The example writes the "*IDN?\n" string to all the USBTMC
```

```

* devices connected to the system and attempts to read back
* results using the write and read functions.
* Open Resource Manager
* Open VISA Session to an Instrument
* Write the Identification Query Using viPrintf
* Try to Read a Response With viScanf
* Close the VISA Session*/
ViSession defaultRM;
ViSession instr;
ViUInt32 numInstrs;
ViFindList findList;
ViStatus status;
char instrResourceString[VI_FIND_BUFLLEN];
unsigned char buffer[100];
int i;
status = viOpenDefaultRM(&defaultRM);
if (status < VI_SUCCESS)
{
    printf("Could not open a session to the VISA Resource Manager!\n");
    return status;
}
/*Find all the USB TMC VISA resources in our system and store the number of resources in the system in
numInstrs.*/
status = viFindRsrc(defaultRM, "USB?*INSTR", &findList, &numInstrs, instrResourceString);
if (status<VI_SUCCESS)
{
    printf("An error occurred while finding resources. \nPress Enter to continue.");
    fflush(stdin);
    getchar();
    viClose(defaultRM);
    return status;
}
/** Now we will open VISA sessions to all USB TMC instruments.
* We must use the handle from viOpenDefaultRM and we must
* also use a string that indicates which instrument to open. This
* is called the instrument descriptor. The format for this string
* can be found in the function panel by right clicking on the
* descriptor parameter. After opening a session to the
* device, we will get a handle to the instrument which we
* will use in later VISA functions. The AccessMode and Timeout
* parameters in this function are reserved for future
* functionality. These two parameters are given the value VI_NULL. */
for (i = 0; i < int(numInstrs); i++)
{
    if (i > 0)
    {

```

```

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

/**Now we will close the session to the instrument using viClose. This operation frees all
system resources.*/
status = viClose(defaultRM);
printf("Press Enter to exit.");
fflush(stdin);
getchar();
return 0;
}

int _tmain(int argc, _TCHAR* argv[])
{
    usbtmc_test();
}

```



```

    return 0;
}

```

## b) TCP/IP 示例

```

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

```

## C#示例

- 环境：Window 系统, Visual Studio。
- 描述：通过 USBTMC 和 TCP/IP 访问仪器设备，并在 NI-VISA 上发送"\*IDN?"命令来查询设备信息。
- 步骤：
  1. 打开 Visual Studio 软件，新建一个 C# console project。
  2. 添加 VISA 的 C#引用 Ivi.Visa.dll 和 NationalInstruments.Visa.dll。
  3. 源码：
    - a) USBTMC 示例

```

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

    void Main(string[] args)
    {

```

```

        usbtmc_test();
    }
}

```

## b) TCP/IP 示例

```

class Program
{
    void tcp_ip_test(string ip)
    {
        using (var rmSession = new ResourceManager())
        {
            try
            {
                var resource = string.Format("TCPIP0::{0}::inst0::INSTR", ip);
                var mbSession = (MessageBasedSession)rmSession.Open(resource);
                mbSession.RawIO.Write("*IDN?\n");
                System.Console.WriteLine(mbSession.RawIO.ReadString());
            }
            catch (Exception ex)
            {
                System.Console.WriteLine(ex.Message);
            }
        }
    }

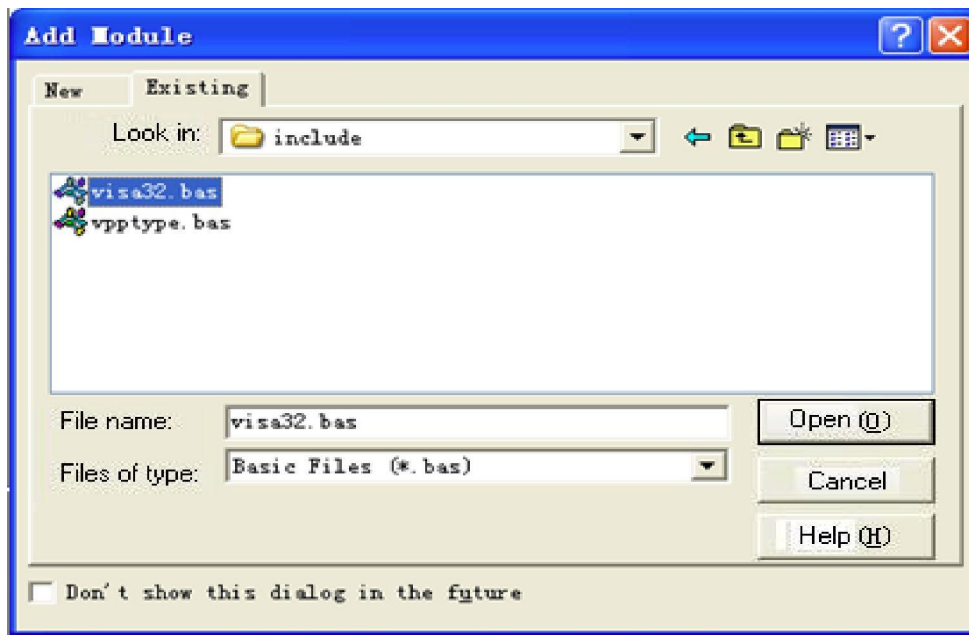
    void Main(string[] args)
    {
        tcp_ip_test("192.168.20.11");
    }
}

```

## VB示例

- 环境：Window 系统, Microsoft Visual Basic 6.0。
- 描述：通过 USBTMC 和 TCP/IP 访问仪器设备，并在 NI-VISA 上发送"\*IDN?"命令来查询设备信息。
- 步骤：
  1. 打开 Visual Basic 软件，并新建一个标准的应用程序项目。
  2. 设置调用 NI-VISA 库项目环境：点击 Existing tab of Project>>Add

Existing Item, 在 NI-VISA 安装路径下的"include"文件夹中查找 visa32.bas 文件并添加该文件。如



下图所示：

### 3. 源码：

#### a) USBTMC 示例

```
PrivateFunction usbtmc_test() AsLong
```

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

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

```

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

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

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

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

```

```
GoTo NextFind
```

```
EndIf
```

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

```
' After the data has been read the response is displayed.
```

```
status = viRead(instrsesn, Buffer, MAX_CNT, retCount)
```

```
If (status < VI_SUCCESS) Then
```

```
    resultTxt.Text = "Error reading a response from the device." + CStr(i + 1)
```

```
Else
```

```
    resultTxt.Text = "Read from device: " + CStr(i + 1) + " " + Buffer
```

```
EndIf
```

```
    status = viClose(instrsesn)
```

```
Next i
```

```
' Now we will close the session to the instrument using
```

```
' viClose. This operation frees all system resources.
```

```
status = viClose(defaultRM)
```

```
usbtmc_test = 0
```

```
EndFunction
```

## b) TCP/IP 示例

```
PrivateFunction tcp_ip_test(ByVal ip AsString) AsLong
```

```
Dim outputBuffer AsString * VI_FIND_BUFLen
```

```
Dim defaultRM AsLong
```

```
Dim instrsesn AsLong
```

```
Dim status AsLong
```

```
Dim count AsLong
```

```
' First we will need to open the default resource manager.
```

```
status = viOpenDefaultRM(defaultRM)
```

```
If (status < VI_SUCCESS) Then
```

```
    resultTxt.Text = "Could not open a session to the VISA Resource Manager!"
```

```
    tcp_ip_test = status
```

```
ExitFunction
```

```
EndIf
```

```
' Now we will open a session via TCP/IP device
```

```
status = viOpen(defaultRM, "TCPIP0::" + ip + "::inst0::INSTR", VI_LOAD_CONFIG, VI_NULL, instrsesn)
```

```
If (status < VI_SUCCESS) Then
```

```
    resultTxt.Text = "An error occurred opening the session"
```

```
    viClose(defaultRM)
```

```
    tcp_ip_test = status
```

```
ExitFunction
```

```
EndIf
```

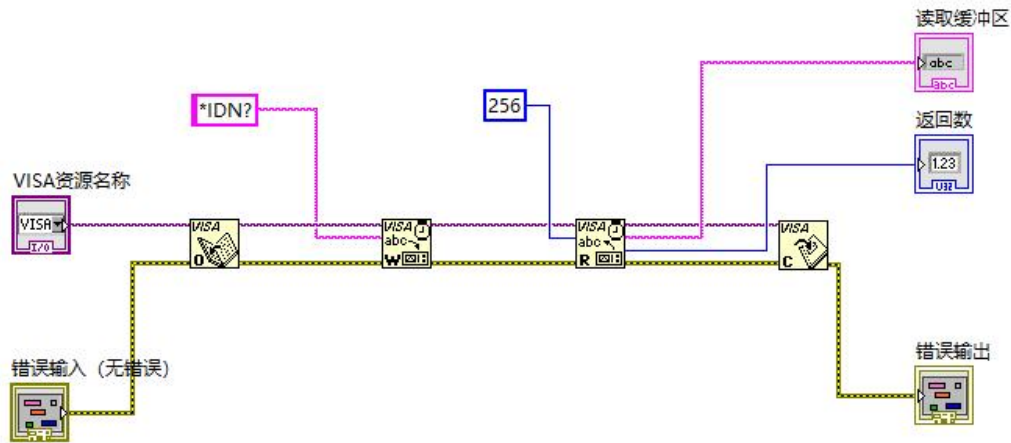
```

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

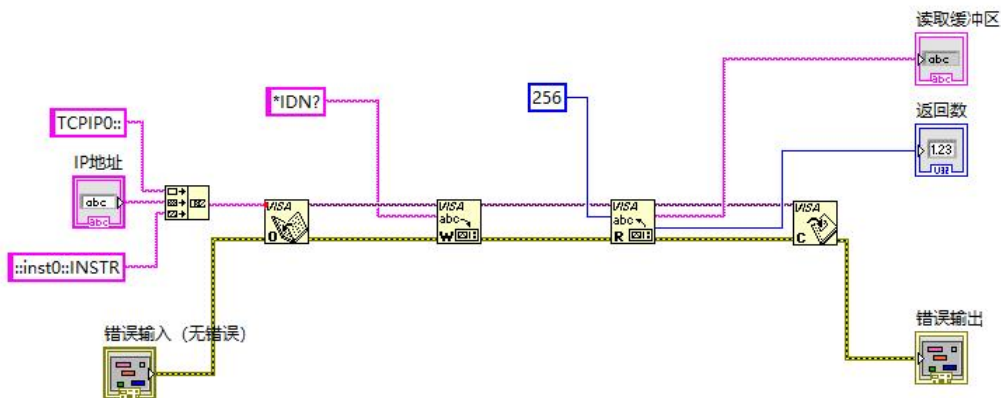
```

## LabVIEW示例

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



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



## MATLAB示例

- 环境：Window 系统, MATLAB。
- 描述：通过 USBTMC 和 TCP/IP 访问仪器设备, 并在 NI-VISA 上发送"\*IDN?"命令来查询设备信息。
- 步骤：
  1. 打开 MATLAB 软件, 点击在 Matlab 界面的 File>>New>>Script 创建一个空的 M 文件。
  2. 源码：
    - a) USBTMC 示例



```

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

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

%Open the VISA object created
fopen(vu);

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

%Request the data

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

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

```

#### b) TCP/IP 示例

```

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

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

%Open the VISA object created

fopen(vt);

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

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

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

```

```
clear vt;
```

```
end
```

## Python示例

- 环境：Window 系统, Python3.8, PyVISA 1.11.0。
- 描述：通过 USBTMC 和 TCP/IP 访问仪器设备，并在 NI-VISA 上发送"\*IDN?"命令来查询设备信息。
- 步骤：
  1. 首先安装 python，然后打开 Python 脚本编译软件，创建一个空的 test.py 文件。
  2. 使用 `pip install PyVISA` 指令安装 PyVISA，如无法安装，请参考此链接使用说明 (<https://pyvisa.readthedocs.io/en/latest/>)

3. 源码：

a) USBTMC 示例

```
import pyvisa
```

```
rm = pyvisa.ResourceManager()
```

```
rm.list_resources()
```

```
my_instrument = rm.open_resource('USB0::0x5345::0x1234::SN20220718::INSTR')
```

```
print(my_instrument.query('*IDN?'))
```

b) TCP/IP 示例

```
import pyvisa
```

```
rm = pyvisa.ResourceManager()
```

```
rm.list_resources()
```

```
my_instrument = rm.open_resource('TCPIP0::192.168.20.11::inst0::INSTR')
```

```
print(my_instrument.query('*IDN?'))
```

## 编程应用实例

### 配置正弦波

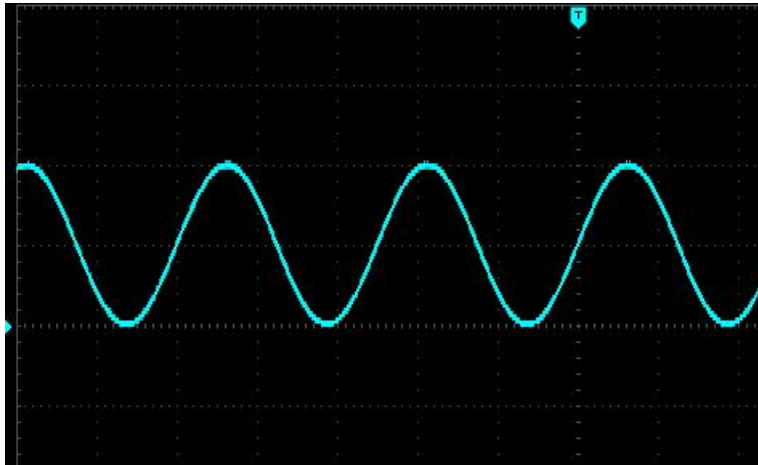
本部分将介绍如何配置正弦波函数。

#### 说明

正弦波具有幅度、偏移以及相对于同步脉冲的相位。可使用高电压值和低电压值设置其幅度和偏移。

#### 示例

下列波形可由 SCPI 命令系列设置，其中高电平和低电平可用于代替 `:CHANnel1:BASE:AMPLitude` 和 `:CHANnel1:BASE:OFFSet`。



以下命令可生成如上所示的正弦波。

```
:CHANnel1:MODE CONTInue  
:CHANnel1:BASE:WAVe SINE  
:CHANnel1:BASE:FREQuency 2000  
:CHANnel1:BASE:HIGh 2  
:CHANnel1:BASE:LOW 0  
:CHANnel1:BASE:PHAsE 20  
:CHANnel1:OUTPut ON
```

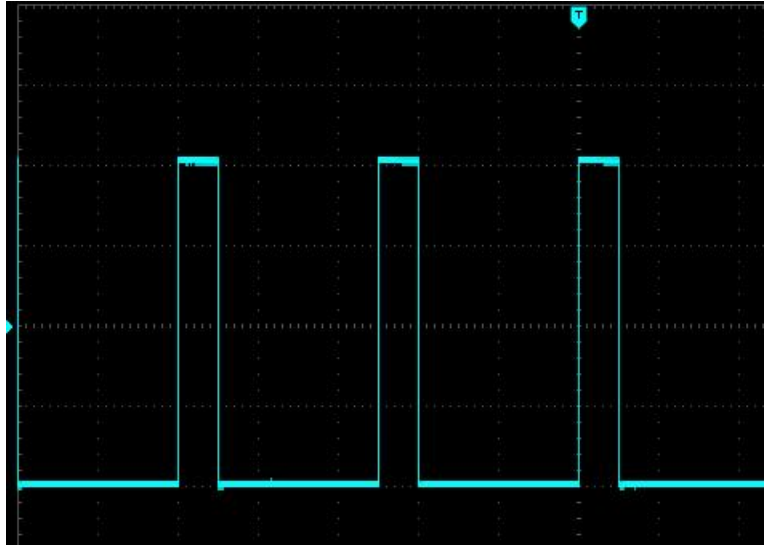
### 配置方波

#### 说明

方波具有幅度、偏移以及相对于同步脉冲的相位。它还具有占空比和周期。可使用高电压值和低电压值设置其幅度和偏移。

## 示例

下列波形可由 SCPI 命令系列设置。



以下命令可生成如上所示的方波。

```
:CHANnel1:MODE CONTinue  
:CHANnel1:BASE:WAVE SQUARE  
:CHANnel1:BASE:FREQUENCY 40000  
:CHANnel1:BASE:AMPLitude 2  
:CHANnel1:BASE:OFFSet 0  
:CHANnel1:BASE:PHase 90  
:CHANnel1:BASE:DUTY 20  
:CHANnel1:OUTPut ON
```

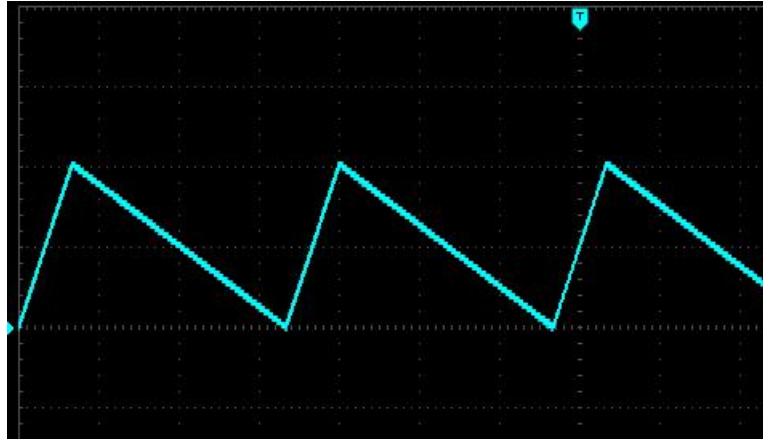
## 配置锯齿波

### 说明

锯齿波具有幅度、偏移以及相对于同步脉冲的相位。它还具有用于创建三角波形和其他类似波形的对称性。可使用高电压值和低电压值设置其幅度和偏移。

### 示例

下列波形可由 SCPI 命令系列设置，其中高电平和低电平可用于代替 `:CHANnel1:BASE:AMPLitude` 和 `:CHANnel1:BASE:OFFSet`。



以下命令可生成如上所示的锯齿波。

```
:CHANnel1:MODE CONTinue  
:CHANnel1:BASE:WAVE RAMP  
:CHANnel1:BASE:FREQuency 30000  
:CHANnel1:BASE:HIGH 2  
:CHANnel1:BASE:LOW 0  
:CHANnel1:BASE:PHAsE 90  
:CHANnel1:RAMP:SYMMetry 20  
:CHANnel1:OUTPut ON
```

## 配置脉冲波

### 说明

脉冲波具有幅度、偏移以及相对于同步脉冲的相位。它还添加边沿斜率和占空比(或脉冲宽度)。可使用高电压值和低电压值设置其幅度和偏移。

### 示例

下列波形可由 SCPI 命令系列设置，其中高电平和低电平可用于代替 :CHANnel1:BASE:AMPLitude 和 :CHANnel1:BASE:OFFSet。



以下命令可生成如上所示的脉冲波。

```
:CHANnel1:MODE CONTInue  
:CHANnel1:BASE:WAVe PULSe  
:CHANnel1:BASE:FREQUency 100000  
:CHANnel1:BASE:HIGH 2  
:CHANnel1:BASE:LOW 0  
:CHANnel1:BASE:PHAsE 270  
:CHANnel1:BASE:DUTY 20  
:CHANnel1:PULSe:RISe 0.0000002  
:CHANnel1:PULSe:FALL 0.0000002  
:CHANnel1:OUTPut ON
```

## 配置任意波

本部分将介绍如何配置任意波形。

### 说明

谐波具有频率、幅度、偏移以及相位。它还添加模式、波形文件。

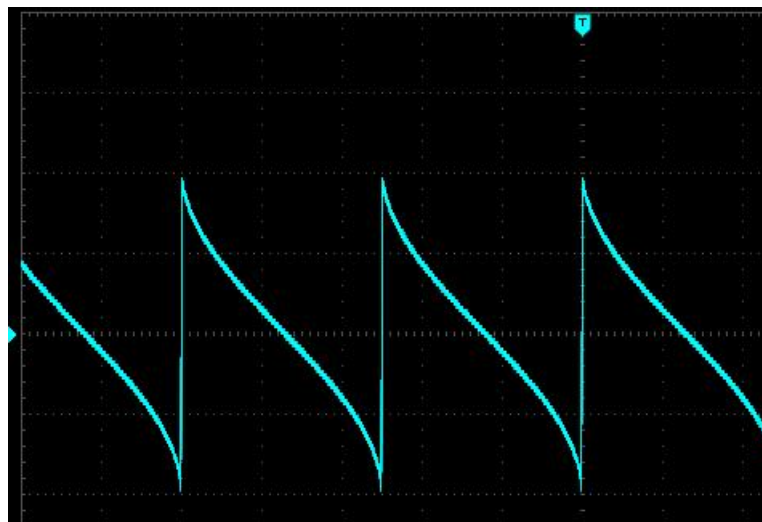
### 示例

下面的代码可加载和修改内置任意波形。

```
:CHANnel1:MODE CONTInue  
:CHANnel1:BASE:WAVe ARB  
:CHANnel1:ARB:MODE DDS  
:CHANnel1:BASE:ARB INTernal,"ACos.bsv"  
:CHANnel1:BASE:FREQUency 200000
```

```
:CHANnel1:BASE:AMPLitude 2  
:CHANnel1:BASE:OFFSet 0  
:CHANnel1:BASE:PHAsE 90  
:CHANnel1:OUTPut ON
```

从这些命令生成的波形如下所示。



## 配置谐波

### 说明

谐波具有幅度、偏移以及相位。它还添加谐波总次数、谐波幅度、谐波相位。可使用高电压值和低电压值设置其幅度和偏移。

### 示例

下列波形可由 SCPI 命令系列设置，其中高电平和低电平可用于代替 `:CHANnel1:BASE:AMPLitude` 和 `:CHANnel1:BASE:OFFSet`。



以下命令可生成如上所示的谐波。

```
:CHANnel1:MODE CONTinue  
:CHANnel1:BASE:WAVE HARMonic  
:CHANnel1:BASE:FREQuency 1000  
:CHANnel1:BASE:HIGH 1  
:CHANnel1:BASE:LOW 0  
:CHANnel1:BASE:PHAsE 90  
:CHANnel1:HARMonic:TOTal:ORDer 10  
:CHANnel1:HARMonic:TYPe ALL  
:CHANnel1:HARM:ORDER2:AMPL 0.02  
:CHANnel1:HARM:ORDER2:PHAsE 20  
:CHANnel1:HARM:ORDER3:AMPL 0.01  
:CHANnel1:HARM:ORDER3:PHAsE 30  
:CHANnel1:OUTPut ON
```

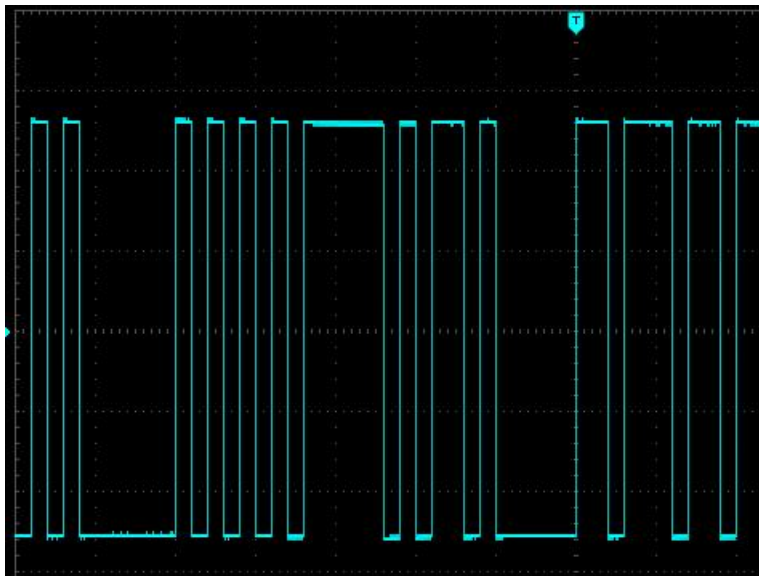
## 配置伪随机波

### 说明

伪随机波具有位率、偏移、边沿时间、码元等属性。可使用高电压值和低电压值设置其幅度和偏移。

### 示例

下列波形可由 SCPI 命令系列设置，其中高电平和低电平可用于代替 :CHANnel1:BASE:AMPLitude 和 :CHANnel1:BASE:OFFSet。





以下命令可生成如上所示的谐波。

```
:CHANnel1:MODE CONTinue
```

```
:CHANnel1:BASE:WAVE PRBS
```

```
:CHANnel1:PRBS:BITRatio 1000000
```

```
:CHANnel1:BASE:HIGH 1
```

```
:CHANnel1:BASE:LOW 0
```

```
:CHANnel1:PNCode PN9
```

```
:CHANnel1:OUTPut ON
```

## 附录 1: <key>列表

按键命令关键字	功能描述	LED 灯
F1	选择当前菜单的第一个菜单项	
F2	选择当前菜单的第二个菜单项	
F3	选择当前菜单的第三个菜单项	
F4	选择当前菜单的第四个菜单项	
F5	选择当前菜单的第五个菜单项	
F6	选择当前菜单的第六个菜单项	
MODE	输出模式	√
WAVE	波形类型	
UTILity	系统	
TRIGger	触发	√
CH1	通道一按键	√
CH2	通道二按键	√
LEFT	方向键左	
RIGHT	方向键右	
UP	方向键上	
DOWN	方向键下	
OK	确认键	
NUM0	数字键 0	
NUM1	数字键 1	
NUM2	数字键 2	
NUM3	数字键 3	
NUM4	数字键 4	
NUM5	数字键 5	
NUM6	数字键 6	
NUM7	数字键 7	
NUM8	数字键 8	
NUM9	数字键 9	
DOT	数字键小数点	
SYMBOL	数字键符号	