UTG4000A系列可编程信号源

编程手册

REV 00

2023.06.19

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

20230619

软件版本

V2.01.0012

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 ISO9001：2008 标准和 ISO14001：2004 标准， 并进一步认证本产品符合其它国际标准组织成员的相关标准。

联系我们

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

电子邮箱：infosh@uni-trend.com.cn

网址：http://www.uni-trend.com.cn

SCPI 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#3123xxxxxxxxxxxxxxxxxxx\n 表示的具有 123 个字节有效批量数据返回格式，其中 ‘3’ 表示 “123” 占 3 个字符位。

SCPI 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, UTG4000, 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。
- **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?
- **功能描述：**
  获取控制通道间的相位模式。
- **返回格式：**
  查询返回通道间的相位模式。
- **举例：**
  :SYSTem:PHASe:MODe?　　查询返回 SYNChronization

:SYSTem:LANGuage

- **命令格式：**
  :SYSTem:LANGuage {ENGLish|SIMChinese|TRAChinese|KOREan}
  :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 SIMChinese　　设置简体中文为系统显示语言
  :SYSTem:LANGuage?　　查询返回 SIMChinese

: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}
  :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 COMMa	设置数字格式为逗号
:SYSTem:NUMBer:FORMat?	查询返回 COMMa

:SYSTem:BRIGhtness

- **命令格式：**
  :SYSTem:BRIGhtness { 30|40|50|60|70|80|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: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光状态，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: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

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:INVersion

- **命令格式：**

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

  :CHANnel<n>:OUTPut:SYNC:INVersion?

- **功能描述：**

  设置通道同步输出反相。

  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反向状态，0 表示关闭，1 表示打开。

- **举例：**

  :CHANnel1:OUTPut:SYNC:INVersion ON　　设置通道 1 同步输出反相

  :CHANnel1:OUTPut:SYNC:INVersion?　　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 | DBM | VRMS}

: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Ω

  <n>：通道号，n 取值 1、2。

  注意：阻值取值范围为 1~10000,其中 10000 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>:PSK:PNCode

- **命令格式：**

  :CHANnel<n>:PSK:PNCode　<code>

  :CHANnel<n>:PSK:PNCode?

- **功能描述：**

  设置指定通道 PN 码，该指令对双相移键控调制和四相移键控调制下的 PN 码功能的波形有效。

  <code>：表示 PN 码，如下所示：

  {PN7|PN9|PN15|PN21}

  <n>：通道号，n 取值 1、2。

- **返回格式：**

  查询返回指定通道双相移键控调制和四相移键控调制下的 PN 码。

- **举例：**

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

  :CHANnel1:PSK:PNCode?　　查询返回 PN9

:CHANnel<n>:QAM:PNCode

- **命令格式：**

  :CHANnel<n>:QAM:PNCode　<code>

  :CHANnel<n>:QAM:PNCode?

- **功能描述：**

  设置指定通道 PN 码，该指令只对正交调制的 PN 码功能的波形有效。

  <code>：表示 PN 码，如下所示：

  {PN7|PN9|PN11|PN15|PN17|PN21|PN23|PN25}

  <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:QAM:PNCode PN9　　设置通道 1 PN 码为 PN9

  :CHANnel1:QAM:PNCode?　　查询返回 PN9

:CHANnel<n>:TRIGger:SOURce

- **命令格式：**

  :CHANnel<n>:TRIGger:SOURce {INTernal|EXTRise|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|RISe|FALL}

  :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 RISe　　设置通道 1 上升沿触发输出模式

  :CHANnel1:TRIGger:OUTPut?　　查询返回 RISe

连续

:CHANnel<n>:BASE:WAVe

- **命令格式：**

  :CHANnel<n>:BASE:WAVe　{ SINe | SQUare | PULSe | RAMP | ARB | NOISe | DC | HARMonic }

  :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、谐波二进制序列。

<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:BWAVe?　　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-6s ~ 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限幅上限值，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，采用科学计数法返回。
- **举例：**
  :CHANnel1:BASE:PHAse 20　　设置通道 1 输出限位为 20°
  :CHANnel1:BASE:PHAse?　　查询返回 2e1

: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>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，采用科学计数法返回。

- **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 2e1

: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>:RAMP:SYMMetry

- **命令格式：**

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

  :CHANnel<n>:RAMP:SYMMetry?

- **功能描述：**

  设置指定通道斜波信号输出对称度。

  < 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波信号输出对称度。

- **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>: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?

- **命令格式：**
  :CHANnel<n>:HARMonic:ORDer <order>
  :CHANnel<n>:HARMonic:ORDer?
- **功能描述：**
  设置指定通道谐波序号。

< order >：谐波序号，范围 2~16。

<n>：通道号，n 取值 1、2。

- **返回格式：**

  查询返回指定通道谐数序号，返回整型数据。

- **举例：**

  :CHANnel1:HARMonic:ORDer 2　　设置通道 1 谐波序号为 2

  :CHANnel1:HARMonic:ORDer?　　查询返回 2

: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

调制

: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|FM|PM|ASK|FSK|PSK|BPSK|QPSK|OSK|QAM|PWM|SUM}
  分别为调幅、调频、调相、幅移键控、频移键控、相移键控、双相移键控、四相移键控、震荡键控、正交调制、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|ARB|NOISe }
  :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。

  <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:DEPTh

- **命令格式：**

  :CHANnel<n>:MODulate:DEPTh { <depth> }

  :CHANnel<n>:MODulate:DEPTh?

- **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，采用科学计数法返回。

- **举例：**

  :CHANnel1:MODulate:DEPTh 50　　设置通道一调制深度为 50%

  :CHANnel1:MODulate:DEPTh?　　查询返回 5e1

: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 位率 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 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，采用科学计数法返回数据。

- **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 3e1

:CHANnel<n>:PWM:DUTY

- **命令格式：**

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

:CHANnel<n>:PWM:DUTY?

- **功能描述：**

设置指定通道输出脉宽调制下占空比。

< 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：**

查询返回指定通道脉宽调制下脉宽偏差，以科学计数法返回数据。

- **举例：**

:CHANnel1:PWM:DUTY 10　　设置通道一占空比 10%

:CHANnel1:PWM:DUTY?　　查询返回 1e+1

:CHANnel<n>:FSK:FREQuency

- **命令格式：**

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

:CHANnel<n>:FSK:FREQuency?

- **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。

- **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:FREQ 2000　　设置通道一输出跳频频率 2KHz

:CHANnel1:FSK:FREQ?　　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 0~360。

<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 2ms　　设置通道一震荡键控的震荡时间 2ms

:CHANnel1:OSK:TIMe?　　查询返回 2e-3

扫频

:CHANnel<n>:SWEep:TYPe

- **命令格式：**

:CHANnel<n>:SWEep:TYPe { LINe|LOG }

: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。

<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: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：**

  :CHANnel1:SWEep:TRIGger　　触发一次扫频信号输出

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 5ms　　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。

  <n>：通道号，n 取值 1、2。

- **举例：**

  :CHANnel1:BURSt:TRIGger　　触发一次猝发信号输出

WARB 指令

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，波形数据最大 4k 个点，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，此指令只支持.bsv 格式数据。

- **举例：**

  WARB1:MODulate　　写通道一调制任意波形文件

WARB<n>:CARRier

- **命令格式：**

  AWG:WARB<n>:CARRier <arb file>

- **功能描述：**

用于写基波任意波形，波形数据最大 4k 个点，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，此指令只支持.bsv 格式数据。

- **举例：**

  WARB1: CARRier　　写通道一基波任意波形文件

DIGital 命令

用于输出数字通讯信号，例如 UART、SPI、I2C 等通讯信号。

:DIGital

- **命令格式：**

  :DIGital {{1 | ON} | {0 | OFF}}

  :DIGital?
- **功能描述：**

  设置指定通道打开或关闭数字通讯信号功能。
- **返回格式：**

  查询返回数字通讯信号功能打开或关闭，0 表示关闭，1 表示打开。
- **举例：**

  :DIGital ON　　打开数字通讯信号功能

  :DIGital?　　查询返回 1

:DIGital:TYPe

- **命令格式：**

  :DIGital:TYPe {UART|IIC|SPI|DArb}

  :DIGital:TYPe?
- **功能描述：**

  设置指定通道输出数字通讯信号类型，分别为 UART、IIC、SPI、DArb。
- **返回格式：**

  查询返回指定通道输出数字通讯信号类型。
- **举例：**

  :DIGital:TYPe UART　　设置输出 UART 通讯信号

  :DIGital:TYPe?　　查询返回 UART

:DIGital:FORMat

- **命令格式：**

:DIGital:FORMat { DEC|HEX|CHAR }

:DIGital:FORMat?

- **功能描述：**

  设置指定通道输出数字通讯信号数据格式，分别为十进制数据、十六进制数据和 ASCII 数据。

- **返回格式：**

  查询返回指定通道输出数字通讯信号数据格式。

- **举例：**

:DIGital:FORMat HEX	设置数字通讯信号数据格式为十六进制数据
:DIGital:FORMat?	查询返回 HEX

:DIGital:AS

- **命令格式：**

  :DIGital:AS {{1 | ON} | {0 | OFF}}

  :DIGital:AS?

- **功能描述：**

  设置指定通道输出数字通讯信号发送模式。OFF 表示手动发送；ON 表示自动发送。

- **返回格式：**

  查询返回指定通道输出数字通讯信号是否开启自动发送模式，1 表示开启，0 表示未开启。

- **举例：**

:DIGital:AS ON	设置数字通讯信号自动发送
:DIGital:AS?	查询返回 1

:DIGital:AS:INTerval

- **命令格式：**

  :DIGital:AS:INTerval <time>

  :DIGital:AS:INTerval?

- **功能描述：**

  设置指定通道自动输出数字通讯信号间隔时间。

  <time>：间隔时间，单位 S。

- **返回格式：**

  查询返回指定通道输出数字通讯信号间隔时间，采用科学计数法返回。

- **举例：**

:DIGital:AS:INTerval 10ms	设置自动输出数字通讯信号间隔时间 10ms
:DIGital:AS:INTerval?	查询返回 1e-2

:DIGital:TRIGger

- **命令格式：**
  :DIGital:TRIGger
- **功能描述：**
  触发指定通道输出数字通讯信号发送，在信号手动发送模式下有效。
- **举例：**
  :DIGital:TRIGger　　触发一次数字信号发送

:DIGital:DATA

- **命令格式：**
  :DIGital:DATA <data>
  :DIGital:DATA?
- **功能描述：**
  设置指定通道自动输出数字通讯信号数据。
  <data>：二进制字节流数据。
- **返回格式：**
  查询返回指定通道输出数字通讯信号数据，返回二进制字节流数据。
- **举例：**
  :DIGital:DATA　　写入数字通讯信号数据到信号源中
  :DIGital:DATA?　　查询返回二进制字节流

UART

:DIGital:UART:BAUDrate

- **命令格式：**
  :DIGital:UART:BAUDrate <baudrate>
  :DIGital:UART:BAUDrate?
- **功能描述：**
  设置指定通道输出数字 UART 通讯信号的波特率。
  <baudrate>：波特率，单位 bps，整型类型数据。
- **返回格式：**
  查询返回指定通道输出数字 UART 通讯信号的波特率，返回整型数据。
- **举例：**
  :DIGital:UART:BAUDrate 115200　　设置 UART 通讯信号的波特率 115200
  :DIGital:UART:BAUDrate?　　查询返回 115200

:DIGital:UART:DATa

- **命令格式：**
  :DIGital:UART:DATa <bit >
  :DIGital:UART:DATa?
- **功能描述：**
  设置指定通道输出数字 UART 通讯信号的数据位。
  <bit >：数据位，整型类型数据，范围 4~8。
- **返回格式：**
  查询返回指定通道输出数字 UART 通讯信号的数据位，返回整型数据。
- **举例：**
  :DIGital:UART:DATa 4　　设置 UART 通讯信号的数据位 4
  :DIGital:UART:DATa?　　查询返回 4

:DIGital:UART:STOP

- **命令格式：**
  :DIGital:UART:STOP <bit >
  :DIGital:UART:STOP?
- **功能描述：**
  设置指定通道输出数字 UART 通讯信号的停止位。
  <bit >：停止位，整型类型数据，范围 1~2。
- **返回格式：**
  查询返回指定通道输出数字 UART 通讯信号的停止位，返回整型数据。
- **举例：**
  :DIGital:UART:STOP 1　　设置 UART 通讯信号的停止位 1
  :DIGital:UART:STOP?　　查询返回 1

:DIGital:UART:PARity

- **命令格式：**
  :DIGital:UART:PARity {NONE|EVEN|ODD}
  :DIGital:UART:PARity?
- **功能描述：**
  设置指定通道输出数字 UART 通讯信号的校验位，分别为无校验、奇校验、偶校验。
- **返回格式：**
  查询返回指定通道输出数字 UART 通讯信号的校验位，返回整型数据。
- **举例：**
  :DIGital:UART:PARity NONE　　设置 UART 通讯信号的无校验位
  :DIGital:UART:PARity?　　查询返回 NONE

IIC

:DIGital:IIC:CLOCk

- **命令格式：**
  :DIGital:IIC:CLOCk　<freq>
  :DIGital:IIC:CLOCk?
- **功能描述：**
  设置指定通道输出数字 IIC 通讯信号的时钟。
  <freq>：时钟，单位 Hz。
- **返回格式：**
  查询返回指定通道输出数字 IIC 通讯信号的时钟，采用科学计数法返回。
- **举例：**
  :DIGital:IIC:CLOCk 1000　　设置 IIC 通讯信号的时钟 1KHz
  :DIGital:IIC:CLOCk?　　查询返回 1e+3

:DIGital:IIC:ADDRess

- **命令格式：**
  :DIGital:IIC:ADDRess　<address>
  :DIGital:IIC:ADDRess?
- **功能描述：**
  设置指定通道输出数字 IIC 通讯信号的地址。
  <address>：地址，整型类型数据。
- **返回格式：**
  查询返回指定通道输出数字 IIC 通讯信号的地址。
- **举例：**
  :DIGital:IIC:ADDRess 3　　设置 IIC 通讯信号的地址 3
  :DIGital:IIC:ADDRess?　　查询返回 3

SPI

:DIGital:SPI:CLOCk

- **命令格式：**
  :DIGital:SPI:CLOCk　<freq>
  :DIGital:SPI:CLOCk?
- **功能描述：**
  设置指定通道输出数字 SPI 通讯信号的时钟。
  <freq>：时钟，单位 Hz。

- **返回格式：**

  查询返回指定通道输出数字 SPI 通讯信号的时钟，采用科学计数法返回。

- **举例：**

  :DIGital:SPI:CLOCk 1000　　设置 SPI 通讯信号的时钟 1KHz

  :DIGital:SPI:CLOCk?　　查询返回 1e+3

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。

- **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+位图数据

工厂内部指令

用于控制设备操作面板上的按键和旋钮等相关特殊指令。

:SYSTem:INFO?

- **命令格式：**
  :SYSTem:INFO?
- **功能描述：**
  用于查询内置的通讯协议版本号等相关系统信息。
- **返回格式：**
  {协议版本号},{软件版本号},{硬件版本号},{仪器类型},{机型内部代号},{带宽},{采样率},{通道数量},{显示名称},{渠道信息},{序列号},{生产日期},{自定义信息}
  说明：用逗号隔开，按位置填充指定的信息。
- **举例：**
  UTD7104BG:
  20190929,1.02.3,1.00.1,DSO,BG,100MHz,1GSa/s,2CH,UTD7104BG,EN,c4f31266798,2019-09-32,*
  返回格式详细说明：
  - 协议版本号
    内置的版本号，每次修改都要递增，这样管理系统就可以通过判断这个版本号，来确定是否要提醒用户升级固件或者提示某些功能不可用。
    版本号的级数可有下位机自己控制。
  - 软件版本号：1.02.3
  - 硬件版本号：1.00.1
  - 仪器类别：DSO
    示例中： DSO 代表示波器
  - 内置机型代号：BG，每一种系列机型都有一个固定的机型代码，以便上位机区分。
    示例中： BG 就代表 UTD7000BG
    upo-8z 就代表 UPO3000CS 系列，upo-7z 就代表新的 UPO2000CS 系列
  - UTD7104BG：机型显示名称
  - 100MHz：带宽
    示例中： 100MHz
  - 1GSa/s ：采样率
    示例中： 1GS/s
  - 2CH ：通道数量
    示例中： 2 通道
  - 渠道信息： EN:教育 ； **:经/外销
  - 序列号： 产品唯一序列号
  - 生产日期 ： 取自系统信息
  - 自定义信息 ：如果不存在，则以"*"占位。自定义消息中不能出现",”号。

:KEY:<key>

- **命令格式：**
  :KEY:<key>
  :KEY:<key>:LOCK { {1 | ON} | {0 | OFF} }
  :KEY:<key>:LOCK?
  :KEY:<key>:LED?
- **功能描述：**
  用于设置按键功能及该按键的锁定/解锁。按键<key>的定义和描述，详见附录 1：<key>列表。
- **返回格式：**
  查询返回按键锁定状态或者具有 LED 按键灯状态。
  锁定状态：0 表示未锁定，1 表示锁定；
  LED 灯状态：0 表示不亮，1 表示亮。
- **举例：**

:KEY:MOD	设置信号源的调制功能
:KEY:MOD:LOCK ON/OFF	锁定/解锁按键
:KEY:MOD:LOCK?	查询返回该按键锁定状态，1 表示锁定
:KEY:MOD:LED?	查询返回 LED 灯状态，0 表示不亮

:KEY:LOCK?

- **命令格式：**
  :KEY:LOCK?
- **功能描述：**
  用于查询所有按键的锁定状态。
- **返回格式：**
  查询返回所有按键的锁定状态，返回字符序列，每个字符代表一个按键的锁定状态，锁定为 ASCII’1’,未锁定为 ASCII’0’，按照附录 1：<key>列表顺序返回锁定状态。
- **举例：**
  一共 10 个按键，只有第 4、5 两个按键锁定，返回 ASCII 字符串 0001100000

:KEY:LED?

- **命令格式：**
  :KEY:LED?
- **功能描述：**
  用于查询所有带灯按键灯状态。
- **返回格式：**
  查询返回所有按键的灯状态，返回字符序列，每个字符代表一个按键灯状态，亮为 ASCII’1’,不亮为 ASCII’0’，按照附录 1：<key>列表带灯的按键顺序返回灯状态。
- **举例：**
  一共 2 个按键带灯，CH1 按键亮，CH2 按键不亮，返回 ASCII 字符串 10

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。

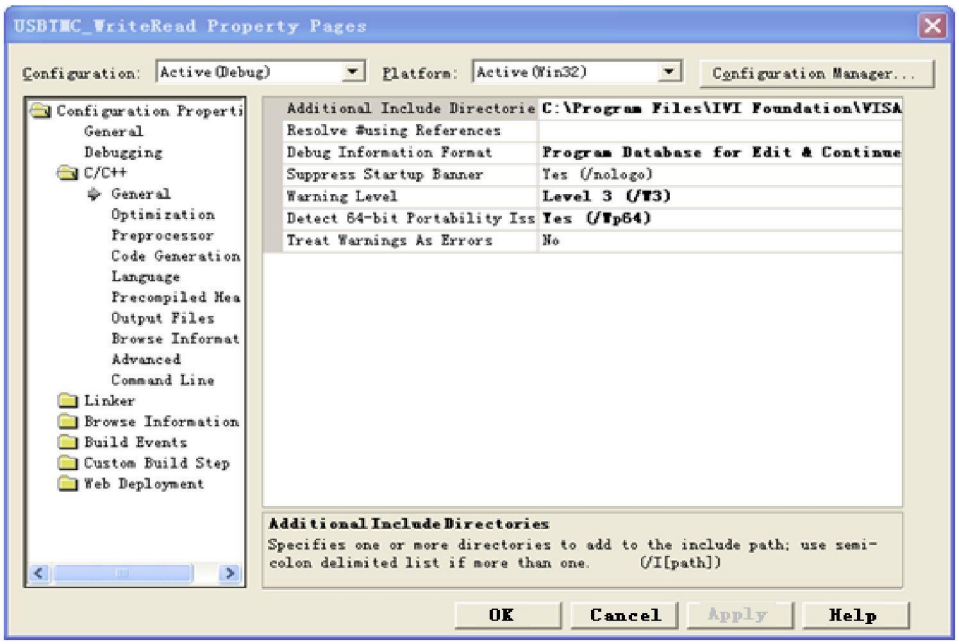
a) 静态库：

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：

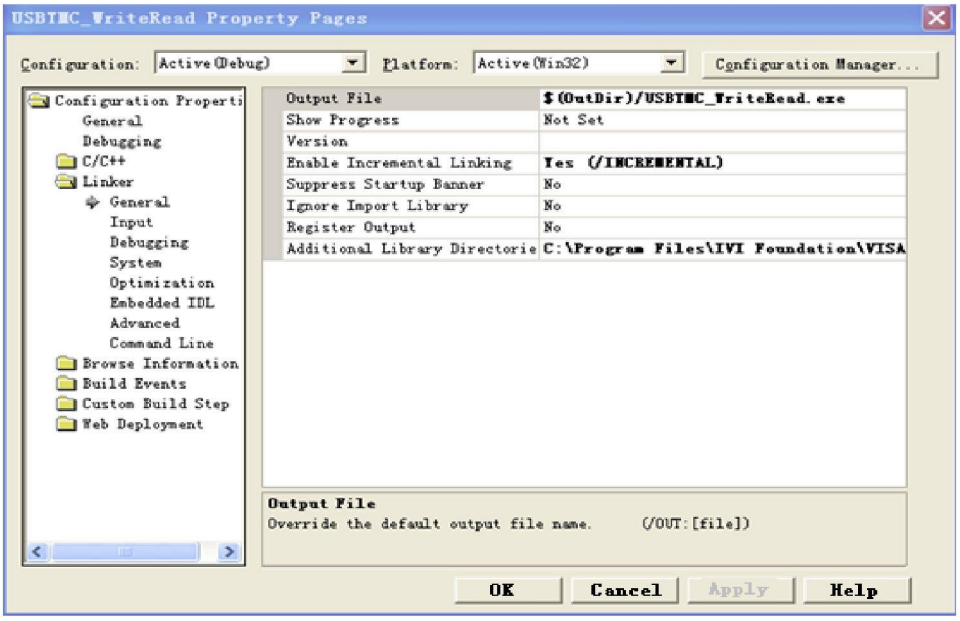
```
#include "visa.h"
#pragma comment(lib,"visa32.lib")
```

b) 动态库：

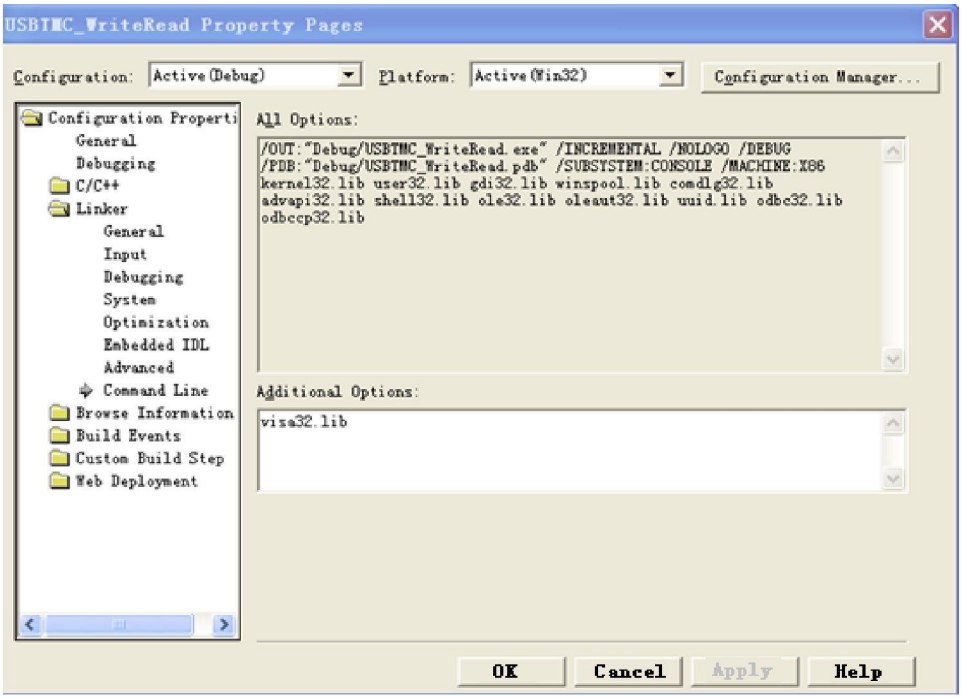
点击"project>>properties"，在属性对话框左侧选择"c/c++---General"中，将 "Additional Include Directories"项的值设置为 NI-VISA 的安装路径，(例如：C:\ProgramFiles\IVI Foundation\VISA\WinNT\include),如下图所示:



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_BUFLEN];
    unsigned char buffer[100];
    int i;
```

```
    status = viOpenDefaultRM(&defaultRM);
    if (status < VI_SUCCESS)
    {
        printf("Could not open a session to the VISA Resource Manager!\n");
        return status;
    }
    /*Find all the USB TMC VISA resources in our system and store the number of resources
    in the system in numInstrs.*/
    status = viFindRsrc(defaultRM, "USB?*INSTR", &findList, &numInstrs,
instrResourceString);
    if (status<VI_SUCCESS)
    {
        printf("An error occurred while finding resources. \nPress Enter to continue.");
        fflush(stdin);
        getchar();
        viClose(defaultRM);
        return status;
    }
    /** Now we will open VISA sessions to all USB TMC instruments.
    *     We must use the handle from viOpenDefaultRM and we must
    *     also use a string that indicates which instrument to open. This
    *     is called the instrument descriptor. The format for this string
    *     can be found in the function panel by right clicking on the
    *     descriptor parameter. After opening a session to the
    *     device, we will get a handle to the instrument which we
    *     will use in later VISA functions. The AccessMode and Timeout
    *     parameters in this function are reserved for future
    *     functionality. These two parameters are given the value VI_NULL. */
    for (i = 0; i < int(numInstrs); i++)
    {
        if (i > 0)
        {
            viFindNext(findList, instrResourceString);
        }
        status = viOpen(defaultRM, instrResourceString, VI_NULL, VI_NULL, &instr);
        if (status < VI_SUCCESS)
        {
            printf("Cannot open a session to the device %d. \n", i + 1);
            continue;
        }
        /** At this point we now have a session open to the USB TMC instrument.
        *We will now use the viPrintf function to send the device the string "*IDN?\n",
        *asking for the device's identification. */
        char * cmmand = "*IDN?\n";
        status = viPrintf(instr, cmmand);
```

```
            if (status < VI_SUCCESS)
            {
                printf("Error writing to the device %d. \n", i + 1);
                status = viClose(instr);
                continue;
            }
            /** Now we will attempt to read back a response from the device to
            *the identification query that was sent. We will use the viScanf
            *function to acquire the data.
            *After the data has been read the response is displayed. */
            status = viScanf(instr, "%t", buffer);
            if (status < VI_SUCCESS)
            {
                printf("Error reading a response from the device %d. \n", i + 1);
            }
            else
            {
                printf("\nDevice %d: %s\n", i + 1, buffer);
            }
            status = viClose(instr);
        }
        /*Now we will close the session to the instrument using viClose. This operation frees all
        system resources.*/
        status = viClose(defaultRM);
        printf("Press Enter to exit.");
        fflush(stdin);
        getchar();
        return 0;
}

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

b) TCP/IP 示例

```
int tcp_ip_test(char *pIP)
{
        char outputBuffer[VI_FIND_BUFLEN];
        ViSession defaultRM, instr;
        ViStatus status;
        /* First we will need to open the default resource manager. */
        status = viOpenDefaultRM(&defaultRM);
```

```
    if (status < VI_SUCCESS)
    {
        printf("Could not open a session to the VISA Resource Manager!\n");
    }
    /* Now we will open a session via TCP/IP device */
    char head[256] = "TCPIP0::";
    char tail[] = "::inst0::INSTR";
    strcat(head, pIP);
    strcat(head, tail);
    status = viOpen(defaultRM, head, VI_LOAD_CONFIG, VI_NULL, &instr);
    if (status < VI_SUCCESS)
    {
        printf("An error occurred opening the session\n");
        viClose(defaultRM);
    }
    status = viPrintf(instr, "*idn?\n");
    status = viScanf(instr, "%t", outputBuffer);
    if (status < VI_SUCCESS)
    {
        printf("viRead failed with error code: %x \n", status);
        viClose(defaultRM);
    }
    else
    {
        printf("\nMesseage read from device: %*s\n", 0, outputBuffer);
    }
    status = viClose(instr);
    status = viClose(defaultRM);
    printf("Press Enter to exit.");
    fflush(stdin);
    getchar();
    return 0;
}

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

C#示例

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

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

a) USBTMC 示例

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

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

b) TCP/IP 示例

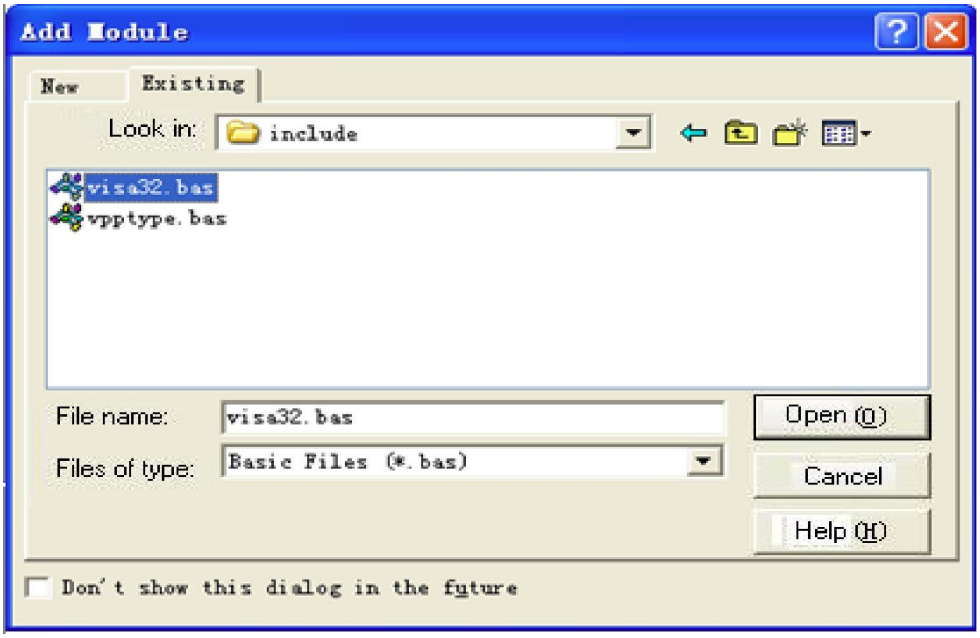
```
class Program
 {
      void tcp_ip_test(string ip)
      {
          using (var rmSession = new ResourceManager())
          {
              try
              {
                  var resource = string.Format("TCPIP0::{0}::inst0::INSTR", ip);
                  var mbSession = (MessageBasedSession)rmSession.Open(resource);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

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

VB 示例

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

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



3. 源码：

a) USBTMC 示例

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

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

' First we must call viOpenDefaultRM to get the manager
' handle. We will store this handle in defaultRM.
status = viOpenDefaultRM(defaultRM)
```

```
If(status < VI_SUCCESS) Then
    resultTxt.Text = "Could not open a session to the VISA Resource Manager!"
    usbtmc_test = status
ExitFunction
EndIf

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

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

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

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

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

b) TCP/IP 示例

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

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

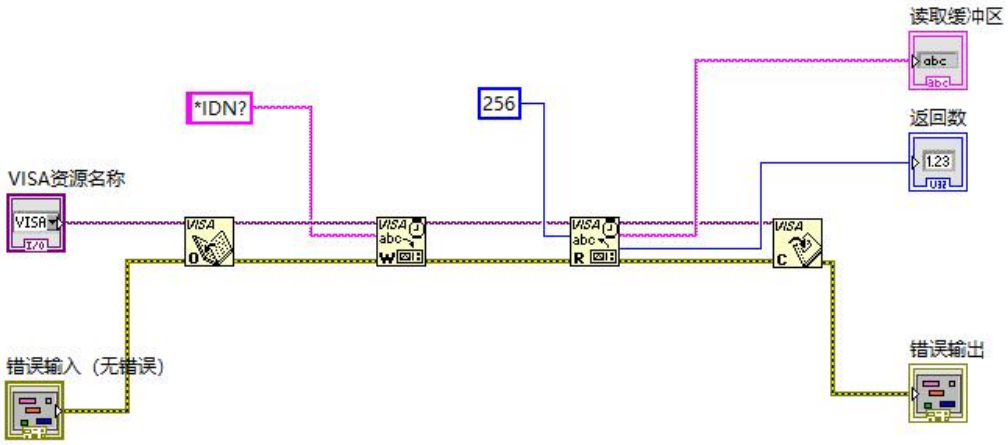
' Now we will open a session via TCP/IP device
status = viOpen(defaultRM, "TCPIP0::" + ip + "::inst0::INSTR", VI_LOAD_CONFIG, VI_NULL,
instrsesn)
If (status < VI_SUCCESS) Then
    resultTxt.Text = "An error occurred opening the session"
    viClose(defaultRM)
    tcp_ip_test = status
ExitFunction
```

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

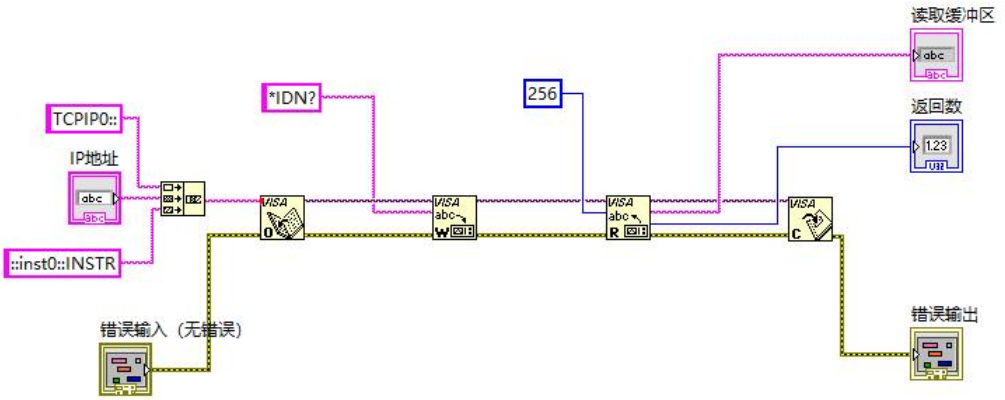
LabVIEW 示例

- 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;

end
```

b) TCP/IP 示例

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

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

%Open the VISA object created

fopen(vt);

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

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

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

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

End
```

Python 示例

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

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

a) USBTMC 示例

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

b) TCP/TP 示例

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

编程应用实例

配置正弦波

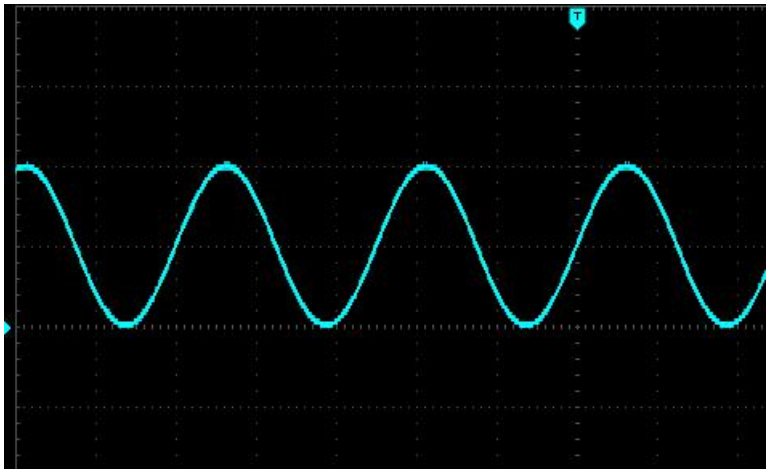
本部分将介绍如何配置正弦波函数。

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
```

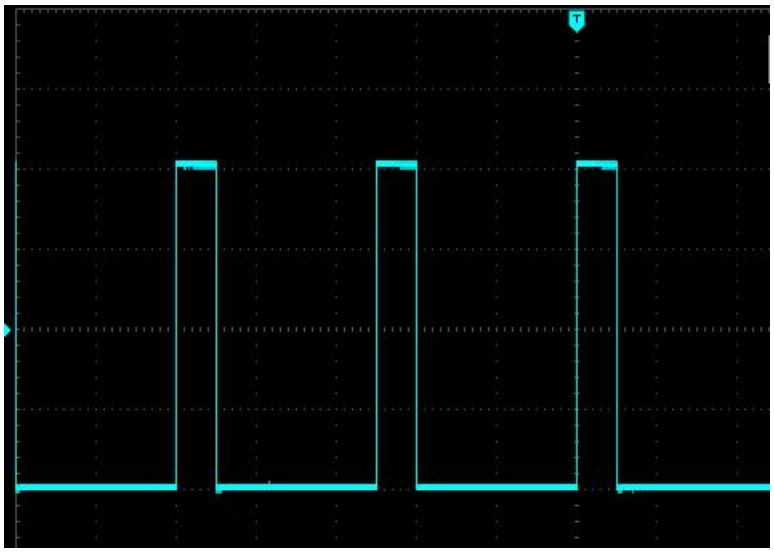
配置方波

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

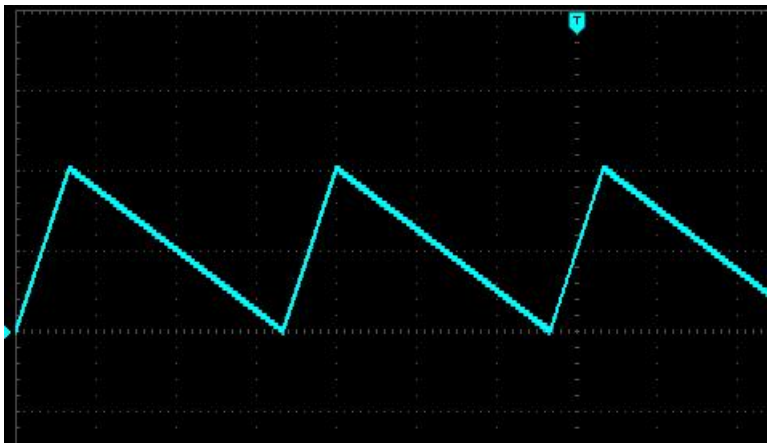
配置锯齿波

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

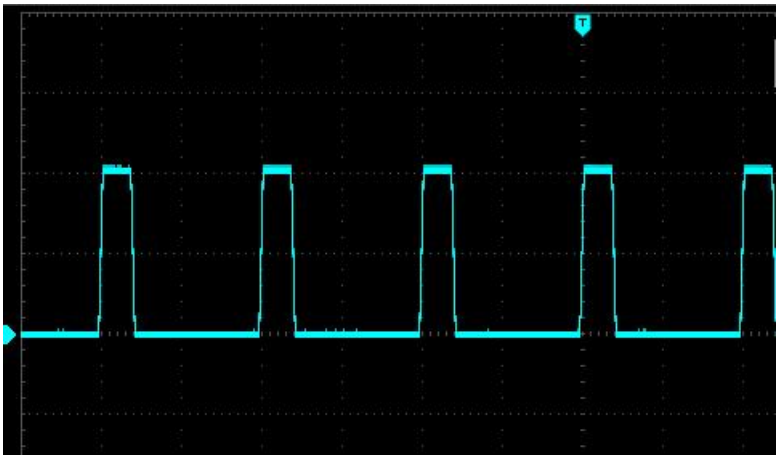
说明

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

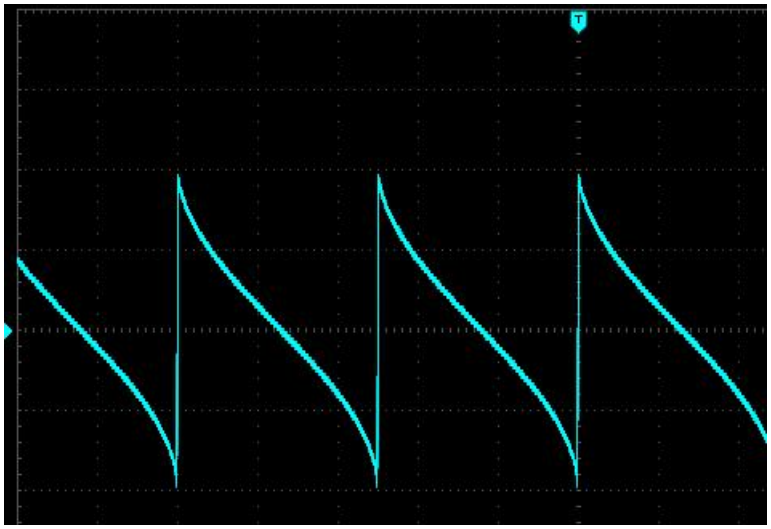
谐波具有频率、幅度、偏移以及相位。它还添加模式、波形文件。

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。



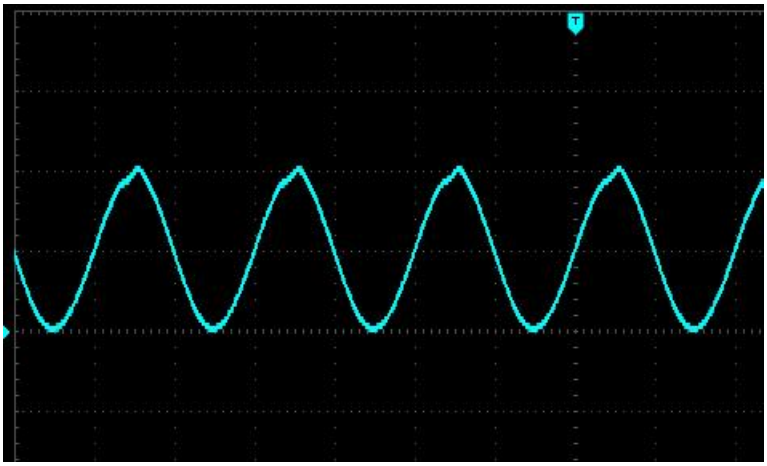
配置谐波

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

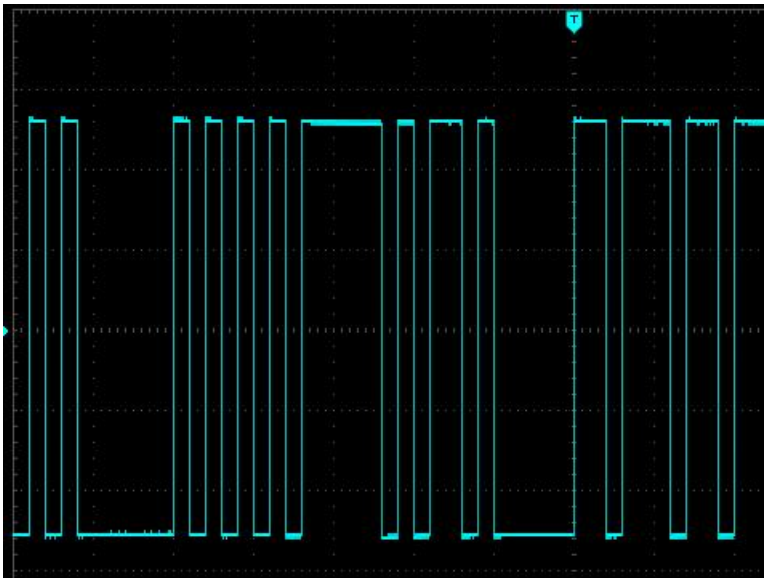
配置伪随机波

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
USER	用户	
DIGital	数字	√
COUNter	计数	√
MOD	模式	
SWEep	扫描	
BURSt	猝发	
SINe	正弦波	√
SQUare	方波	√
RAMP	三角波	√
PULSe	脉冲波	√
ARB	任意波	√
HARMonic	谐波	√
NOISe	噪声	√
DC	直流	√
CH1	通道一按键	√
CH2	通道二按键	√
SYNC	同步	
TRIGger	触发	√
PRESet	预置复位	
STORage	存储	
HELP	帮助	
UTILity	系统	
F1	F1 功能键	
F2	F2 功能键	
F3	F3 功能键	
F4	F4 功能键	
F5	F5 功能键	
F6	F6 功能键	
PGDN	切换下一页	

RIGHt	方向键右	
LEFT	方向键左/退格	
OK	旋钮按下	
UP	旋钮左旋	
DOWN	旋钮右旋	
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	