UTG900系列可编程信号源

编程手册

REV 00

2022.07.15

UNI-T®

保证和声明

版权

商标信息

UNI-T是优利德中国科技有限公司的注册商标。

文档编号

20220715

软件版本

V3.06.004

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

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.

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，其中实型数据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、信号源型号、产品序列号和软件版本号。
- **返回格式：**
  制造商名称，信号源型号、产品序列号，由点号分隔的软件版本号。
- **举例：**
  UNI-T Technologies, UTG900, 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 {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: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 | 1 |5 | 15 |30 |60 }

: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 1　　设置系统 1 分钟之后自动休眠

  :SYSTem:SLEEP:TIMe?　　查询返回 1

: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。

  注意：打开该功能会关闭通道的同步输出。

- **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，采样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，采样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%

CHANnel 命令

用于设置信号源通道相关功能。

: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+

: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>:BASE:WAVe

- **命令格式：**
  :CHANnel<n>:BASE:WAVe { SINe | SQUare | PULSe | RAMP | ARB | NOISe | DC }
  :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。
  <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。

- **举例：**

:CHANnel1:BASE:PHAse 20　　设置通道 1 输出限位为 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>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>: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>:MODe

- **命令格式：**

:CHANnel<n>:MODe {CONTINUE | AM | PM | FM | FSK | Line | Log }

:CHANnel<n>:MODe?

- **功能描述：**

  设置指定通道信号模式，分别为 CONTINUE 、AM、PM、FM、FSK、Line、Log。

  <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 AM	设置通道 1 信号幅度调制模式输出
:CHANnel1:MODe?	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 信号信号调制波类型为正弦波
:CHANnel1:MODulate:WAVe?	查询返回 SINe

: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:ARB:INDex

- **命令格式：**
  :CHANnel<n>:MODulate:ARB:INDex {<index >}
  :CHANnel<n>:MODulate:ARB:INDex?
- **功能描述：**
  设置指定通道加载信号源存储的调制任意波序号。
  <index >表示任意波序号。
  <n>：通道号，n 取值 1、2。
- **返回格式：**
  查询返回指定通道调制任意波序号。
- **举例：**
  :CHANnel1:MODulate:ARB:IND 2　　通道 1 加载信号源存储的第二种调制任意波
  :CHANnel1:MODulate:ARB:IND?　　查询返回 2

:CHANnel<n>:MODulate:ARB:SOURce

- **命令格式：**
  :CHANnel<n>:MODulate:ARB:SOURce { INTernal|EXTernal }
  :CHANnel<n>:MODulate:ARB:SOURce?
- **功能描述：**
  设置指定通道调制任意波源，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任意波源。
- **举例：**
  :CHANnel1:MODulate:ARB:SOURce INTernal　　设置通道一调制任意波源为内部
  :CHANnel1:MODulate:ARB:SOURce?　　查询返回 INTernal

: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: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。
- **举例：**
  :CHANnel1:MODulate:DEPTh 50 设置通道一调制深度为 50%
  :CHANnel1:MODulate:DEPTh? 查询返回 50

:CHANnel<n>:ARB:INDex

- **命令格式：**
  :CHANnel<n>:ARB:INDex {<index >}
  :CHANnel<n>:ARB:INDex?
- **功能描述：**
  设置指定通道加载信号源存储的任意波序号。
  <index >表示任意波序号。
  <n>：通道号，n 取值 1、2。
- **返回格式：**
  查询返回指定通道任意波序号。
- **举例：**
  :CHANnel1:ARB:IND 2 设置通道 1 加载信号源存储的第二种任意波
  :CHANnel1:ARB:IND? 查询返回 2

:CHANnel<n>:ARB:SOURce

- **命令格式：**
  :CHANnel<n>:ARB:SOURce { INTernal|EXTernal }
  :CHANnel<n>:ARB:SOURce?
- **功能描述：**
  设置指定通道任意波源，分别内部、外部两种。
  <n>：通道号，n 取值 1、2。

- **返回格式：**

  查询返回指定通道任意波源。

- **举例：**

  :CHANnel1:ARB:SOURce INTernal　　设置通道一任意波源为内部

  :CHANnel1:ARB:SOURce?　　查询返回 INTernal

: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，采样科学计数法返回数据。

- **举例：**

  :CHANnel<n>:FM:FREQuency:DEV 2000　　设置通道一频率偏移 2KHz

  :CHANnel<n>: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：**

  :CHANnel<n>:PM:PHASe:DEV 30　　设置通道一相位偏移 30°

  :CHANnel<n>:PM:PHASe:DEV?　　查询返回 30

:CHANnel<n>:FSK:HOPPing:FREQuency

- **命令格式：**

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

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

- **功能描述：**

  设置指定通道输出跳频频率。

  < 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跳频频率，以科学计数法返回数据。

- **举例：**

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

  :CHANnel1:FSK:HOPP:FREQ?　　查询返回 2e+3

: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　　设置通道一扫频时的扫描时间为 2S
  :CHANnel1:SWEEP:TIMe?　　查询返回 2e+0

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，波形数据固定 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。
- **举例：**
  :WARB1:MODulate　　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，波形数据固定 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。
- **举例：**
  :WARB1: CARRier　　写通道一基波任意波形文件

DISPlay 命令

用于获取信号源显示相关信息。

:DISPlay?

- **命令格式：**
  :DISPlay?
- **功能描述：**
  用于查询示波器当前屏幕的图像数据。
- **返回格式：**
  查询返回图像数据，返回的数据符合附录 2：IEEE 488.2 二进制数据格式。
- **举例：**
  :DISPlay? 查询返回图像数据

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。

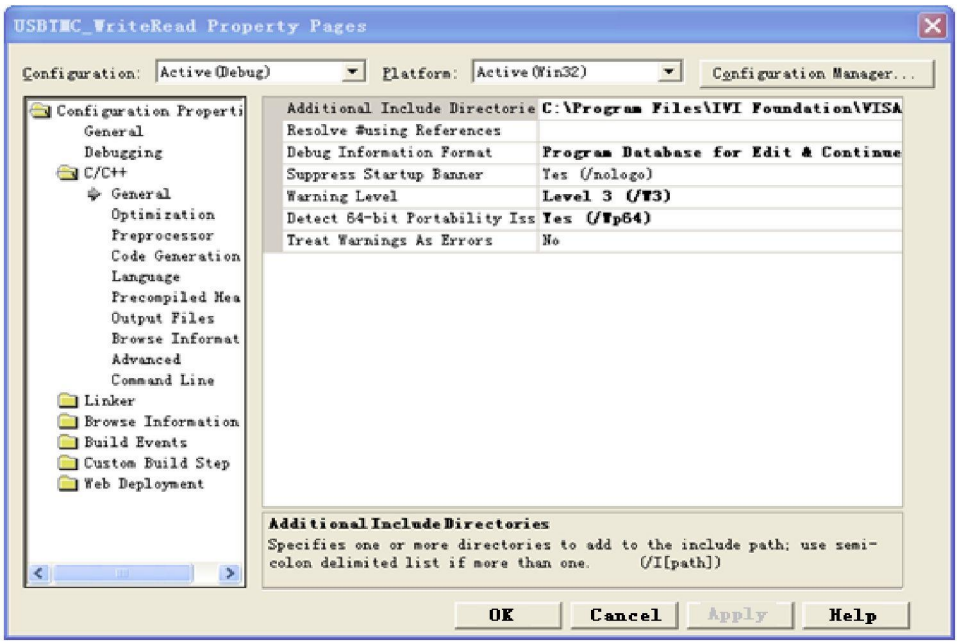
a) 静态库：

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：

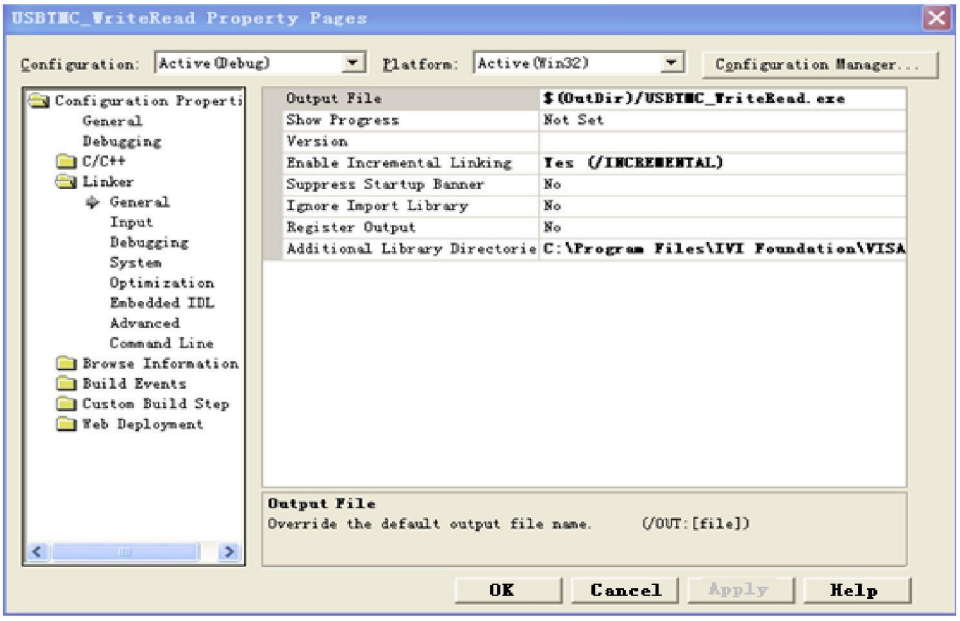
```
#include "visa.h"
#pragma comment(lib,"visa32.lib")
```

b) 动态库：

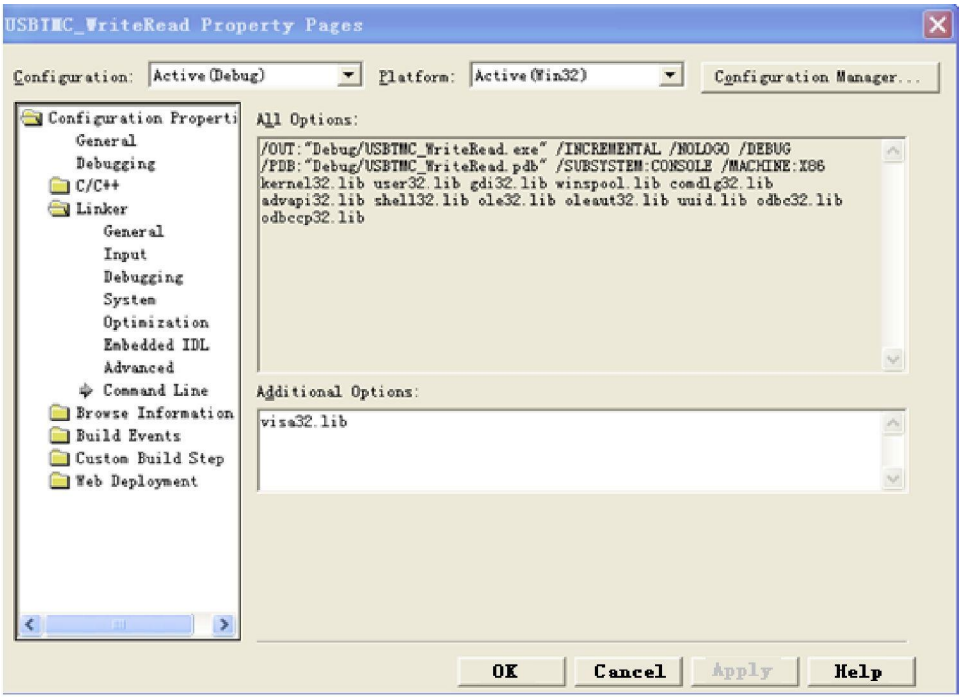
点击"project>>properties"，在属性对话框左侧选择"c/c++---General"中，将 "Additional Include Directories"项的值设置为 NI-VISA 的安装路径，(例如：C:\ProgramFiles\IVI Foundation\VISA\WinNT\include),如下图所示:



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_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();
    }
}
```

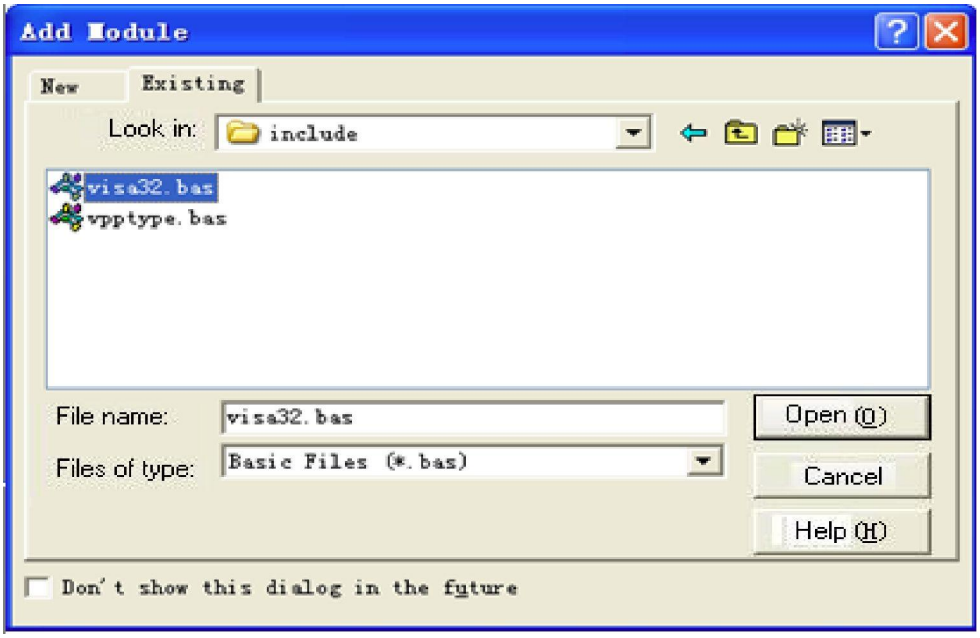
b) TCP/IP 示例

```
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

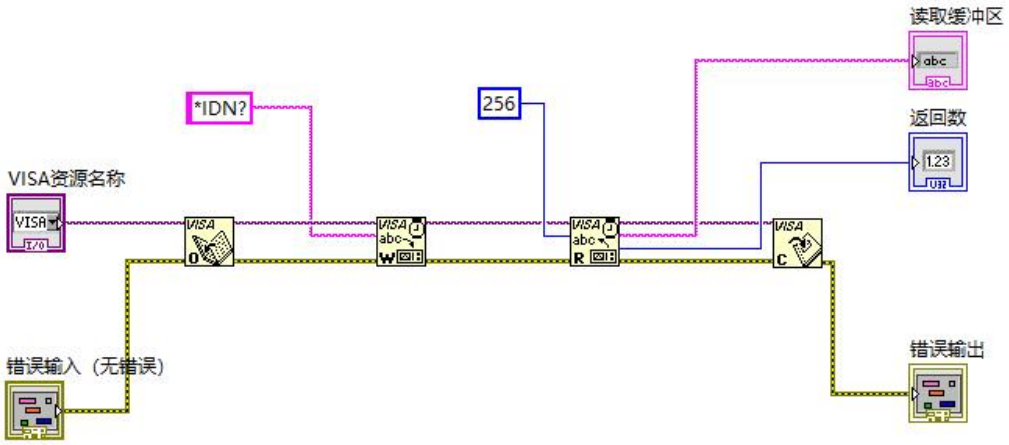
```
PrivateFunction tcp_ip_test(ByVal ip AsString) AsLong
Dim outputBuffer AsString * VI_FIND_BUFLEN
Dim defaultRM AsLong
Dim instrsesn AsLong
Dim status AsLong
Dim count AsLong

' 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
```

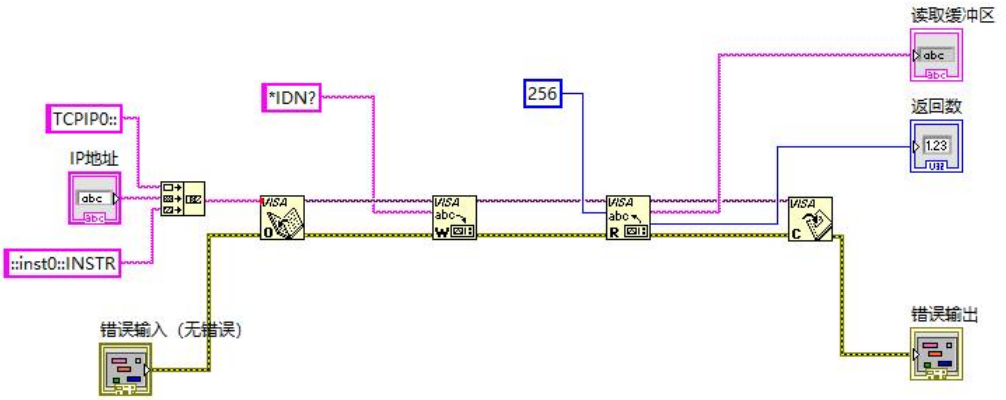
LabVIEW 示例

- 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

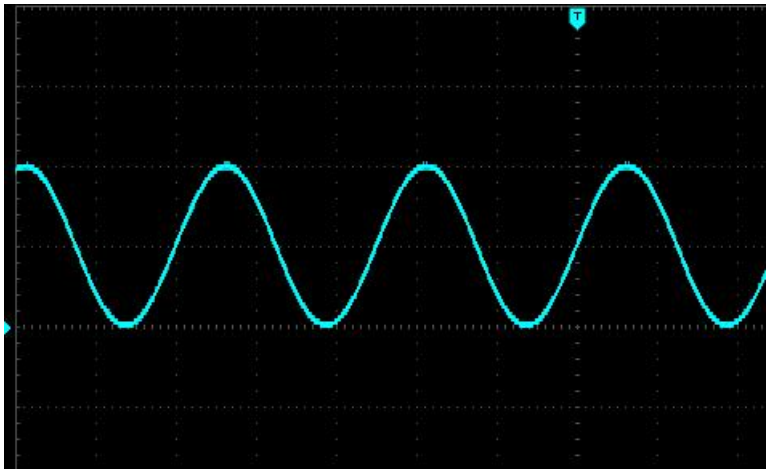
本部分将介绍如何配置正弦波函数。

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
```

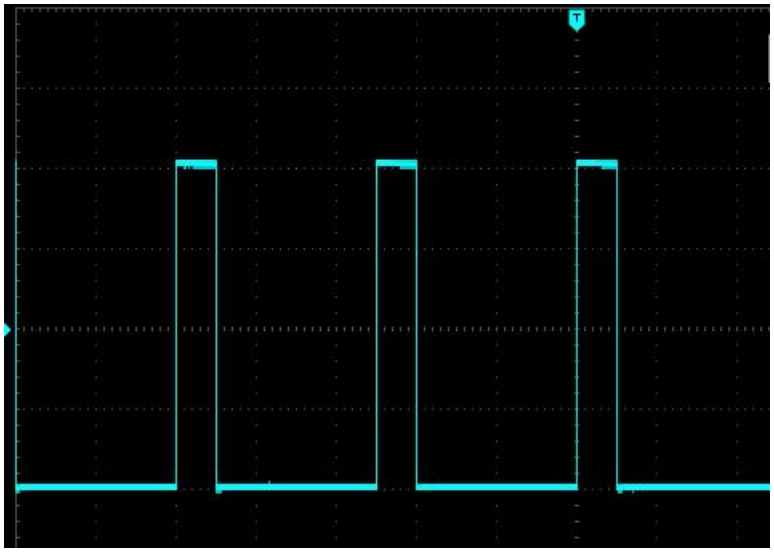
配置方波

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

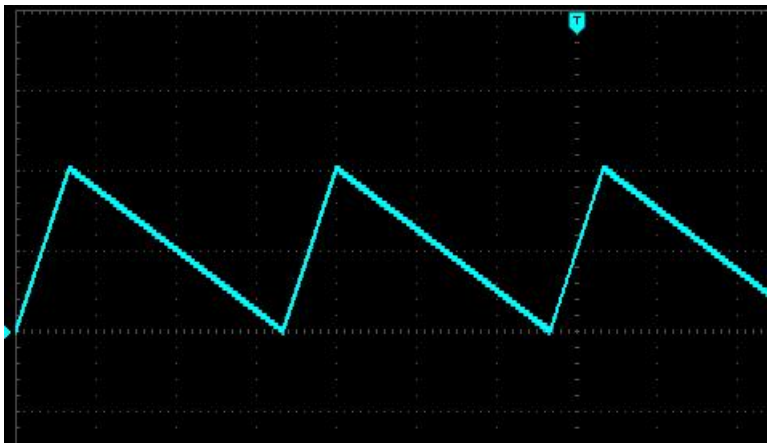
说明

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

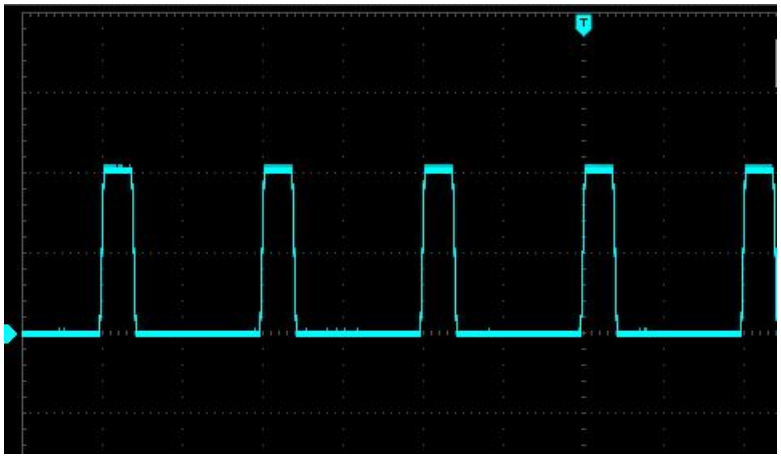
配置脉冲波

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
```

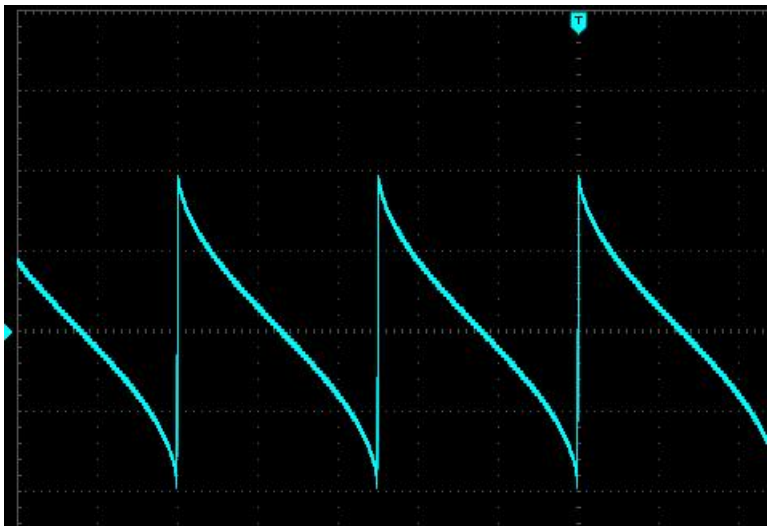
:CHANnel1:BASE:AMPLitude 2

: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。



附录 1：按键列表

按键	功能描述	LED 灯
Wave	波形类型	
Mode	输出模式	√
Utility	系统	
Symbol	数字键符号	
Dot	数字键小数点	
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	
Up	方向键上	
Down	方向键下	
Left	方向键左	
Right	方向键右	
OK	确认键	
CH1	通道一按键	√
CH2	通道二按键	√
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	

附录 2：IEEE 488.2 二进制数据格式

DATA 是数据流,其他为 ASCII 字符,如下图所示: <#812345678 + DATA + \n>

开始符 (1Byte)	长度位宽 (1Byte)	数据总长度 (位宽 Byte)	DATA（n Byte)	结束符 (1Byte)
#	x	x x x x x x x x	.....	\n