

UTD2000系列可编程数字存储示波器

编程手册

REV 0

2023. 02

UNI-T®

保证和声明

版权

2017 优利德中国科技有限公司

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

2023.02.16

软件版本

V1.08.0046

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

- 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：

网址：<https://www.uni-trend.com.cn>

本编程手册仅适用于以下机型：UTD2000 系列

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。

注意：当返回数据为无效数据均用*表示

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、示波器型号、产品序列号和软件版本号。
- **返回格式:**
制造商名称, 示波器型号, 产品序列号, 由点号分隔的软件版本号。
- **举例:**
UNI-T Technologies, UP02000CS, UP01000, 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

用于对示波器进行最基本的操作, 主要包括运行控制、全键盘锁定、错误队列和系统设置数据的操作。

:RUN

- **命令格式:**
:RUN
- **功能描述:**
用于开始示波器波形采样工作, 如需停止工作, 需要执行:STOP 命令。

:STOP

- **命令格式:**
:STOP
- **功能描述:**
用于停止示波器波形采样工作, 如需恢复工作, 需要执行:RUN 命令。

:AUTO

➤ 命令格式:

:AUTO

➤ 功能描述:

用于自动设定仪器的控制值，通过自动设置使输入的波形达到最佳显示效果。

: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/:SYST:LOCK 1

全键盘锁定

:SYSTem:LOCK OFF/:SYST:LOCK 0

全键盘解锁

:SYSTem:LOCK?

查询返回 1，表示锁定

:SYSTem:ERRor

➤ 命令格式:

:SYSTem:ERRor

:SYSTem:ERRor?

➤ 功能描述:

用于清空错误消息队列。

➤ 返回格式:

查询返回最后一次消息错误，查询以“<消息编号>,<消息内容>”格式返回错误消息，其中<消息编号>是一个整数，<消息内容>是一个带双引号的 ASCII 字符串。

如-113,"Undefined header; command cannot be found"。

➤ 举例:

:SYSTem:ERR

清空错误队列

:SYSTem:ERR?

查询返回:

-113,"Undefined header; command cannot be found"

表示未定义指令头

:SYSTem:SETup

➤ 命令格式:

:SYSTem:SETup <setup_data>

:SYSTem:SETup?

➤ 功能描述:

用于配置系统设置数据。其中<setup_data>为符合 IEEE 488.2 #格式的二进制数据。

发送时，直接在命令字符串后跟数据流，一次性完成发送。

读取时，请确保有足够的缓冲接收数据流，否则在读取时程序可能异常。

➤ **返回格式：**

查询返回系统设置数据，返回的数据符合 IEEE 488.2 #格式的二进制数据。

:SYSTem:LANGUage

➤ **命令格式：**

:SYSTem:LANGUage

{ ENGLish | SIMPlifiedchinese | TRADitionalchinese | JAPANese | KORean | GERMan |
FRENch | RUSSian}

:SYSTem:LANGUage?

➤ **功能描述：**

用于设置系统语言，分别英语、简体中文、繁体中文、日语、汉语、德语、法语、俄语。

➤ **返回格式：**

查询返回{ ENGLish | SIMPlifiedchinese | TRADitionalchinese | JAPANese | KORean |
GERMan | FRENch | RUSSian}。

➤ **举例：**

:SYSTem:LANGUage SIMP

设置系统语言为简体中文

:SYSTem:LANGUage?

查询返回 SIMPlifiedchinese

:SYSTem:CAL

➤ **命令格式：**

:SYSTem:CAL

➤ **功能描述：**

用于设置系统自校准，自校正期间，不能正常通信。

:SYSTem:CLEAR

➤ **命令格式：**

:SYSTem:CLEAR

➤ **功能描述：**

用于清除系统所有的存储波形和设置数据。

:SYSTem:CYMometer:FREQuency?

➤ **命令格式：**

:SYSTem:CYMometer:FREQuency?

➤ **功能描述：**

用于获取频率计测量的频率值。

➤ **返回格式：**

查询返回频率计测量的频率值，无效值时返回*。

➤ **举例：**

:SYSTem:CYMometer:FREQuency? 查询返回 1.20000E+3

:SYSTem:SQUare:SElect

➤ 命令格式:

:SYSTem:SQUare:SElect { 10Hz | 100Hz | 1KHz | 10KHz }
:SYSTem:SQUare:SElect?

➤ 功能描述:

用于设置选择方波输出。

➤ 返回格式:

查询返回 { 10Hz | 100Hz | 1KHz | 10KHz }。

➤ 举例:

:SYSTem:SQUare:SElect 10Hz 选择 10Hz 的方波输出
:SYSTem:SQUare:SElect? 查询返回 10Hz

:SYSTem:MNUDisplay

➤ 命令格式:

:SYSTem:MNUDisplay { 5S | 10S | 20S | INFinite}
:SYSTem:MNUDisplay?

➤ 功能描述:

用于设置菜单的显时间，INFinite 表示菜单一直显示。

➤ 返回格式:

查询返回 { 5S | 10S | 20S | INFinite }。

➤ 举例:

:SYSTem:MNUDisplay 5S 设置菜单显示时基为 5S 之后，自动收回。
:SYSTem:MNUDisplay? 查询返回 5S

:SYSTem:BRIGhtness

➤ 命令格式:

:SYSTem:BRIGhtness <count>
:SYSTem:BRIGhtness?

➤ 功能描述:

用于设置屏幕亮度，<count>取值为 1~100，数字越大屏幕越亮。

➤ 返回格式:

查询返回当前屏幕亮度。

➤ 举例:

:SYSTem:BRIGhtness 50 设置屏幕亮度 50
:SYSTem:BRIGhtness? 查询返回 50

:SYSTem:VERSion?

➤ 命令格式:

:SYSTem:VERSion?

➤ 返回格式:

查询返回版本信息，128 字节的字符串信息。

HW 为硬件版本号，SW 为软件版本号，PD 为生成日期，ICV 为协议版本号。

➤ **举例：**

:SYST:VERS?

查询返回 HW:1.0;SW:1.0;PD:2014-11-20;ICV:1.4.0

: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

:SYSTem:AUTO:CHANnel

➤ **命令格式:**

:SYSTem:AUTO:CHANnel {AUTO|KEEP}
:SYSTem:AUTO:CHANnel?

➤ **功能描述:**

用于自动设置时通道属性是否保持当前状态。

AUTO 表示通道按预置参数自动设置；KEEP 表示通道在保持当前状态前提进行自动设置。

➤ **返回格式:**

查询{AUTO|KEEP}。

➤ **举例:**

:SYSTem:AUTO:CHANnel KEEP 通道保持当前状态自动设置。
:SYSTem:AUTO:CHANnel? 查询返回 KEEP。

:SYSTem:AUTO:ACQuire

➤ **命令格式:**

:SYSTem:AUTO:ACQuire {AUTO|KEEP}
:SYSTem:AUTO:ACQuire?

➤ **功能描述:**

用于自动设置时采样属性是否保持当前状态。

AUTO 表示采样按预置参数自动设置；KEEP 表示采样在保持当前状态前提进行自动设置。

➤ **返回格式：**

查询 {AUTO|KEEP}。

➤ **举例：**

:SYSTem:AUTO:ACQuire KEEP 采样保持当前状态自动设置。
:SYSTem:AUTO:ACQuire? 查询返回 KEEP。

:SYSTem:AUTO:TRIGger

➤ **命令格式：**

:SYSTem:AUTO:TRIGger {AUTO|KEEP}
:SYSTem:AUTO:TRIGger?

➤ **功能描述：**

用于自动设置时触发属性是否保持当前状态。

AUTO 表示触发按预置参数自动设置；KEEP 表示触发在保持当前状态前提进行自动设置。

➤ **返回格式：**

查询 {AUTO|KEEP}。

➤ **举例：**

:SYSTem:AUTO:TRIGger KEEP 触发保持当前状态自动设置。
:SYSTem:AUTO:TRIGger? 查询返回 KEEP。

:SYSTem:AUTO:SIGNal

➤ **命令格式：**

:SYSTem:AUTO:SIGNal {AUTO|KEEP}
:SYSTem:AUTO:SIGNal?

➤ **功能描述：**

用于自动设置时输入信号的通道 (活动通道) 是否保持当前状态。

AUTO 表示活动通道按预置参数自动设置；KEEP 表示活动通道在保持当前状态前提进行自动设置。

➤ **返回格式：**

查询 {AUTO|KEEP}。

➤ **举例：**

:SYSTem:AUTO:SIGNal KEEP 活动通道保持当前状态自动设置。
:SYSTem:AUTO:SIGNal? 查询返回 KEEP。

KEY 命令

用于控制示波器操作面板上的按键和旋钮。

: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: [按键列表](#)。
- **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:AUTO	自动设置示波器的各项控制值
:KEY:AUTO:LOCK ON/OFF	锁定/解锁按键
:KEY:AUTO:LOCK?	查询返回该按键锁定状态, 1 表示锁定
:KEY:AUTO:LED?	查询返回 LED 灯状态, 0 表示不亮

CHANnel 命令

用于对每个通道单独进行设置。

:CHANnel<n>:BWLimit

- **命令格式:**
:CHANnel<n>:BWLimit {{1|ON}|{0|OFF}}
:CHANnel<n>:BWLimit?
- **功能描述:**
用于设置带宽限制功能为 ON (打开限制带宽至 20MHz, 以减少显示的噪音) 或 OFF (关闭带宽限制实现满带宽显示)。
<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:BWL ON	打开通道 1 的带宽限制。
:CHAN1:BWL?	查询返回 1, 表示已经打开通道 1 的带宽限制。

:CHANnel<n>:COUPling

- **命令格式:**
:CHANnel<n>:COUPling {DC|AC|GND}
:CHANnel<n>:COUPling?
- **功能描述:**
用于设置通道的耦合方式。DC (直流) 表示可通过输入信号的交流和直流分量; AC (交流) 表示阻挡输入信号的直流分量; GND (接地) 表示断开输入信号。
<n>: {1|2}, 分别表示 {CH1|CH2}。
- **返回格式:**
查询 AC、DC 或 GND。
- **举例:**

:CHAN1:COUP DC	设置通道 1 的耦合方式为直流。
:CHAN1:COUP?	查询返回 DC。

:CHANnel<n>:DISPlay

➤ 命令格式:

```
:CHANnel<n>:DISPlay { {1|ON} | {0|OFF} }  
:CHANnel<n>:DISPlay?
```

➤ 功能描述:

用于设置指定通道 ON (打开) 或 OFF (关闭)。
<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:DISP ON           打开通道 1。  
:CHAN1:DISP?           查询返回 1, 表示已经打开通道 1。
```

:CHANnel<n>:INVert

➤ 命令格式:

```
:CHANnel<n>:INVert { {1|ON} | {0|OFF} }  
:CHANnel<n>:INVert?
```

➤ 功能描述:

用于设置波形反相功能为 ON (打开波形反相功能) 或 OFF (恢复波形正常显示)。
<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:INV OFF           关闭通道 1 的反相显示功能。  
:CHAN1:INV?           查询返回 0, 表示已关闭通道 1 反相功能。
```

:CHANnel<n>:PROBe

➤ 命令格式:

```
:CHANnel<n>:PROBe {0.01X | 0.02X | 0.05X | 0.1X | 0.2X | 0.5X | 1X | 2X | 5X | 10X | 20X | 50X  
| 100X | 200X | 500X | 1000X }  
:CHANnel<n>:PROBe?
```

➤ 功能描述:

用于设置与探头相对应的探头衰减因数。
<n>: {1|2}, 分别表示 {CH1|CH2}。

➤ 返回格式:

查询返回 {0.01X | 0.02X | 0.05X | 0.1X | 0.2X | 0.5X | 1X | 2X | 5X | 10X | 20X | 50X | 100X | 200X
| 500X | 1000X }。

➤ 举例:

```
:CHAN1:PROB 10X         设置通道 1 探头衰减因数为 10。  
:CHAN1:PROB?         查询返回 10X。
```

:CHANnel<n>:OFFSet

➤ 命令格式:

:CHANnel<n>:OFFSet <offset>

:CHANnel<n>:OFFSet?

➤ 功能描述:

用于设置波形在垂直方向上的位移。

<n>: {1|2|5|6|7}, 分别表示 {CH1|CH2|MATH|REFA|REFB}。

➤ 返回格式:

查询返回 offset 的设置值, 采用科学计数法, 单位和当前通道单位相关。

➤ 举例:

:CHAN1:OFFS 20V

设置通道 1 垂直位移为 20V。

:CHAN1:OFFS?

查询返回 2.000e001。

:CHANnel<n>:SCALe

➤ 命令格式:

:CHANnel<n>:SCALe {<scale> | UP | DOWN}

:CHANnel<n>:SCALe?

➤ 功能描述:

用于设置示波器在垂直方向上伏格的档位。

<scale>: 伏格档位值;

UP: 示波器当前档位基础上加一档;

DOWN : 示波器当前档位基础上减一档。

<n>: {1|2|5|6|7}, 分别表示 {CH1|CH2|MATH|REFA|REFB}。

➤ 返回格式:

查询返回伏格的档位的当前值, 采用科学计数法, 单位 V。

➤ 举例:

:CHAN1:SCAL 20V

设置通道 1 伏格档位为 20V。

:CHAN1:SCAL?

查询返回 2.000e001。

:CHAN1:SCAL UP

在 20V 伏格档位上加一档。

:CHANnel<n>:UNITs

➤ 命令格式:

:CHANnel<n>:UNITs {VOLTs|AMPeRes }

:CHANnel<n>:UNITs?

➤ 功能描述:

用于设置通道单位为 VOLTs (电压)、AMPeRes (电流)。

<n>: {1|2}, 分别表示 {CH1|CH2}。

➤ 返回格式:

查询返回 VOLTs、AMPeRes。

➤ 举例:

:CHAN1:UNIT VOLT

设置通道 1 单位为电压。

:CHAN1:UNIT?

查询返回 VOLTs。

:CHANnel<n>:VERNier

➤ 命令格式:

```
:CHANnel<n>:VERNier { {1|ON} | {0|OFF} }  
:CHANnel<n>:VERNier?
```

➤ 功能描述:

用于设置档位调节方式。当设置为 ON（打开）时为微调（Fine），微调在粗调设置范围之间进一步细分，以改善垂直分辨率；当设置为 OFF（关闭）时为粗调（Coarse），粗按 1-2-5 进制设定垂直灵敏度。
<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:VERN ON           打开通道 1 微调功能。  
:CHAN1:VERN?            查询返回 1。
```

: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|5|6|7}，分别表示 {CH1|CH2|MATH|REFA|REFB}。

➤ 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 1 被选中。
```

TIMEbase 命令

用于改变当前通道的水平刻度（时基）和触发在内存中的水平位置（触发位移）。改变水平刻度会使波形相对屏幕中心扩张或收缩，改变水平位置则使波形相对于屏幕中心的位置有偏移。

:TIMEbase:MODE

➤ 命令格式:

```
:TIMEbase:MODE {MAIN | WINDow}  
:TIMEbase:MODE?
```

➤ 功能描述:

用于设置时基模式，MAIN（主时基）或 WINDow（缩放时基<Zoomed>）。

➤ 返回格式:

查询返回 MAIN 或 WINDow。

➤ 举例:

:TIM:MODE MAIN 设置时基模式为主时基。
:TIM:MODE? 查询返回 MAIN。

:TIMbase:OFFSet

➤ **命令格式:**

:TIMbase:OFFSet <offset>
:TIMbase:OFFSet?

➤ **功能描述:**

用于调整 MAIN（主时基）时基偏移量，即波形位置相对屏幕中心的偏移。

➤ **返回格式:**

查询返回<offset>值。采用科学计数法，单位为 s。

➤ **举例:**

:TIM:OFFS 1s 设置主时基偏移量为 1s。
:TIM:OFFS? 查询返回 1.000e000。

:TIMbase:WINDow:OFFSet

➤ **命令格式:**

:TIMbase:WINDow:OFFSet <offset>
:TIMbase:WINDow:OFFSet?

➤ **功能描述:**

用于调整 WINDow（缩放时基<Zoomed>）时基偏移量，即波形位置相对屏幕中心的偏移。

➤ **返回格式:**

查询返回<offset>值。采用科学计数法，单位为 s。

➤ **举例:**

:TIM:WIND:OFFS 1 设置 WINDow 时基偏移量为 1s。
:TIM:WIND:OFFS? 查询返回 1.000e000。

:TIMbase:SCALe

➤ **命令格式:**

:TIMbase:SCALe {<scale> | UP | DOWN}
:TIMbase:SCALe?

➤ **功能描述:**

用于设置 MAIN（主时基）的时基档位，即 s/div（秒/格）。

<scale>: 时基档位值;

UP: 示波器当前档位基础上加一档;

DOWN : 示波器当前档位基础上减一档。

➤ **返回格式:**

查询返回< scale>值。采用科学计数法，单位为 s/div。

➤ **举例:**

:TIM:SCAL 2 设置主时基偏移量为 2s/div。
:TIM:SCAL? 查询返回 2.000e000。

:TIMebase:WINDow:SCALe

➤ 命令格式:

:TIMebase:WINDow:SCALe {<scale> | UP | DOWN}

:TIMebase:WINDow:SCALe?

➤ 功能描述:

用于设置 WINDow (缩放时基<Zoomed>) 时基档位, 即 s/div (秒/格)。

➤ 返回格式:

查询返回< scale>值。采用科学计数法, 单位为 s/div。

➤ 举例:

:TIM:WIND:SCAL 2

设置 WINDow 时基偏移量为 2s/div。

:TIM:WIND:SCAL?

查询返回 2.000e000。

FUNCTION 命令

用于显示 CH1、CH2 通道波形加、减、乘、除、与、或、非、异或以及 FFT 运算的结果, 设置滤波器, 利用表达式运算。

:FUNCTION:MATH:MODE

➤ 命令格式:

FUNCTION:MATH:MODE {MATH | FFT | FILTER}

FUNCTION:MATH:MODE?

➤ 功能描述:

用于选择 MATH 功能模式。

➤ 返回格式:

查询返回 {MATH | FFT | FILTER }。

➤ 举例:

FUNC:MATH:MODE FFT

选择 MATH 模式为 FFT 模式

FUNC:MATH:MODE?

查询返回 FFT

:FUNCTION:OPERation

➤ 命令格式:

:FUNCTION:OPERation {ADD | SUBTRACT | MULTIPLY | DIVIDE }

:FUNCTION:OPERation?

➤ 功能描述:

用于设置函数运算符, 包含基本和逻辑运算。分别为加、减、乘、除、与、或、非、异或。

➤ 返回格式:

查询返回 {ADD | SUBTRACT | MULTIPLY | DIVIDE }。

➤ 举例:

:FUNCTION:OPERation ADD

使用相加操作符实现: src1+src2

:FUNCTION:OPERation?

查询返回 ADD

:FUNCTION:SOURCE<m>

➤ 命令格式:

:FUNCTION:SOURCE<m> {CHANNEL1 | CHANNEL2}

:FUNCTION:SOURCE<m>?

➤ 功能描述:

SOURCE <m>表示源 1 或源 2，其中<m>取值为 1、2。

SOURCE1 用于选择操作符数学函数的第一个源，也可作为 Filter、FFT 的单一源。

SOURCE2 用于选择操作符数学函数的第二个源，Filter、FFT 等单一源不适用。

<value>表示 CHANNEL<n>，其中<n>取值为 1/2 {CH1/ CH2}。

➤ 返回格式:

查询返回 CHANNEL1、CHANNEL2。

➤ 举例:

:FUNCTION:SOUR1 CHAN1	将一通道作为第一个源
:FUNCTION:SOUR1?	查询返回 CHANNEL1
:FUNCTION:SOUR2 CHAN2	将二通道作为第二个源
:FUNCTION:SOUR2?	查询返回 CHANNEL2
:FUNCTION:OPERATION ADD	将源一和源二通道相加

:FUNCTION:FFT:WINDOW

➤ 命令格式:

:FUNCTION:FFT:WINDOW {RECTANGULAR | HANNING | HANNING | HAMMING | BMAN}

:FUNCTION:FFT:WINDOW?

➤ 功能描述:

FFT 加窗截取信号。RECT、HANN、HAMM、BMAN 分别为矩形窗、汉宁窗、汉明窗、布莱克曼窗。

➤ 返回格式:

查询返回 {RECTANGULAR | HANNING | HAMMING | BMAN}。

➤ 举例:

:FUNCTION:SOUR1 CHAN1	将一通道作为源
:FUNC:FFT:WIND HAMM	加汉明窗
:FUNC:FFT:WIND?	查询返回 HAMMING

:FUNCTION:FFT:VTYPE

➤ 命令格式:

:FUNCTION:FFT:VTYPE {VRMS | DBRMS}

:FUNCTION:FFT:VTYPE?

➤ 功能描述:

选择 FFT 垂直方向的单位为 dBRMS 或者 VRMS。dBRMS 表示功率均方根，VRMS 表示电压均方根。

➤ 返回格式:

查询返回 VRMS、DBRMS。

➤ 举例:

:FUNCTION:SOUR1 CHAN1	将一通道作为源
:FUNC:FFT:VTYP VRMS	设置 FFT 垂直方向的单位电压均方根

:FUNC:FFT:VTYP?

查询返回 VRMS

:FUNCtion:FFT:FREQuency

➤ **命令格式:**

:FUNCtion:FFT:FREQuency?

➤ **功能描述:**

获得 FFT 之后频谱波形的中心频率。

➤ **返回格式:**

查询返回 1.000e003, 单位为 Hz。

➤ **举例:**

:FUNCtion:SOUR1 CHAN1

将一通道作为源

:FUNC:FFT:FREQ?

查询返回 1.000e003

:FUNCtion:FILTer:TYPE

➤ **命令格式:**

:FUNCtion:FILTer:TYPE {LP|HP|BP|BS}

:FUNCtion:FILTer:TYPE?

➤ **功能描述:**

设置滤波器类型。LP、HP、BP、BS 分别表示低通滤波器, 高通滤波器, 带通滤波器, 带阻滤波器。

➤ **返回格式:**

查询返回 LP、HP、BP、BS。

➤ **举例:**

:FUNCtion:SOUR1 CHAN1

将一通道作为源

:FUNC:FILT:TYPE BP

设置为带通滤波器

:FUNC:FILT:TYPE?

查询返回 BP

:FUNCtion:FILTer:FREQuency:HIGH

➤ **命令格式:**

:FUNCtion:FILTer:FREQuency:HIGH <freq>

:FUNCtion:FILTer:FREQuency:HIGH?

➤ **功能描述:**

设置滤波器上限截止频率值。适用于高通滤波器、带通滤波器、带阻滤波器。

➤ **返回格式:**

查询返回 1.000e003, 单位为 Hz。

➤ **举例:**

:FUNCtion:SOUR1 CHAN1

将一通道作为源

:FUNC:FILT:FREQ:HIGH 1KHz

设置滤波器上限为 1KHz 截止频率

:FUNC:FILT:FREQ:HIGH?

查询返回 1.000e003

:FUNCtion:FILTer:FREQuency:LOW

➤ **命令格式:**

:FUNCtion:FILTer:FREQuency:LOW <freq>

➤ **举例：**

:MEAS:SOUR CHAN1

选择通道一为测量源

:MEAS:SOUR?

返回 CHANnel1

:MEASure:SLAVe:SOURce

➤ **命令格式：**

:MEASure:SLAVe:SOURce <source>

:MEASure:SLAVe:SOURce?

➤ **功能描述：**

用于选择测量从信源。<source>为 CHANnel<n>，其中 n 取值 1、2。

➤ **返回格式：**

查询返回 {CHANnel1 | CHANnel2 | MATH}。

➤ **举例：**

:MEAS:SLAV:SOUR CHAN1

选择通道一为测量源

:MEAS:SLAV:SOUR?

返回 CHANnel1

:MEASure:PDUTy?

➤ **命令格式：**

:MEASure:PDUTy? [<source>]

➤ **功能描述：**

用于测量指定通道波形的正占空比。其中<source>取值为 CHANnel1、CHANnel2、MATH，省略表示当前通道。

➤ **返回格式：**

查询返回 5.000e001，单位%。

:MEASure:NDUTy?

➤ **命令格式：**

:MEASure:NDUTy? [<source>]

➤ **功能描述：**

用于测量指定通道波形的负占空比。其中<source>取值为 CHANnel1、CHANnel2、MATH，省略表示当前通道。

➤ **返回格式：**

查询返回 5.000e001，单位%。

:MEASure:PDELay?

➤ **命令格式：**

:MEASure:PDELay? [<source1>,<source2>]

➤ **功能描述：**

用于测量<source1>、<source2>相对于上升沿的时间延迟。其中<source>取值为 CHANnel1、CHANnel2、MATH，省略表示当前通道。

➤ **返回格式：**

查询返回-1.000e-004，单位为 s。

➤ **举例：**

测量相对于上升沿的时间延迟
:MEASure:PDEL? CHAN1, CHAN2

:MEASure:NDElay?

➤ **命令格式：**

:MEASure:NDElay? [<source1>, <source2>]

➤ **功能描述：**

用于测量<source1>、<source2>相对于下降沿的时间延迟。其中<source>取值为 CHANne11、CHANne12、MATH，省略表示当前通道。

➤ **返回格式：**

查询返回-1.000e-004，单位为 s。

➤ **举例：**

测量相对于下降沿的时间延迟
:MEASure:NDEL? CHAN1, CHAN2

:MEASure:PHASe?

➤ **命令格式：**

:MEASure:PHASe? [<source1>, <source2>]

➤ **功能描述：**

用于定时测量<source1>相对于<source2>超前或者滞后的时间量，以度表示，360° 为一周期。其中<source>取值为 CHANne11、CHANne12、MATH，省略表示当前通道。

➤ **返回格式：**

查询返回 1.000e001，单位为度。

➤ **举例：**

测量<source1>相对于<source2>超前或者滞后的时间量
:MEASure:PHAS? CHAN1, CHAN2

:MEASure:VPP?

➤ **命令格式：**

:MEASure:VPP? [<source>]

➤ **功能描述：**

用于测量指定通道波形的峰峰值。其中<source>取值为 CHANne11、CHANne12、MATH，省略表示当前通道。

➤ **返回格式：**

查询返回 3.120e000，单位为 V。

:MEASure:VMAX?

➤ **命令格式：**

:MEASure:VMAX? [<source>]

➤ **功能描述：**

用于测量指定通道波形的最大值。其中<source>取值为 CHANne11、CHANne12、MATH，省略表示当前通道。

➤ **返回格式：**

查询返回 2.120e000, 单位为 V。

:MEASure:VMIN?

➤ **命令格式:**

:MEASure:VMIN? [<source>]

➤ **功能描述:**

用于测量指定通道波形的最小值。其中<source>取值为 CHANnel1、CHANnel2、MATH, 省略表示当前通道。

➤ **返回格式:**

查询返回-2.120e000, 单位为 V。

:MEASure:VAMPlitude?

➤ **命令格式:**

:MEASure:VAMPlitude? [<source>]

➤ **功能描述:**

用于测量指定通道波形的幅值。其中<source>取值为 CHANnel1、CHANnel2、MATH, 省略表示当前通道。

➤ **返回格式:**

查询返回 3.120e000, 单位为 V。

:MEASure:VTOP?

➤ **命令格式:**

:MEASure:VTOP? [<source>]

➤ **功能描述:**

用于测量指定通道波形的顶端值。其中<source>取值为 CHANnel1、CHANnel2、MATH, 省略表示当前通道。

➤ **返回格式:**

查询返回 3.120e000, 单位为 V。

:MEASure:VBASe?

➤ **命令格式:**

:MEASure:VBASe? [<source>]

➤ **功能描述:**

用于测量指定通道波形的底端值。其中<source>取值为 CHANnel1、CHANnel2、MATH, 省略表示当前通道。

➤ **返回格式:**

查询返回-3.120e000, 单位为 V。

:MEASure:VMIDdle?

➤ **命令格式:**

:MEASure:VMIDdle? [<source>]

➤ **功能描述:**

用于测量指定通道波形的中间值。其中<source>取值为 CHANnel1、CHANnel2、MATH, 省略表示当前通道。

➤ **返回格式:**

查询返回 0.120e000, 单位为 V。

:MEASure:VAverage?

➤ **命令格式:**

:MEASure:VAverage? [<interval>][,<source>]

➤ **功能描述:**

用于测量指定通道波形的平均值。其中<source>指定通道，取值为 CHANnel1、CHANnel2、MATH，如果没有指定<source>，则默认当前通道；<interval>指定测量间距，取值为 CYCLE、DISPlay，其中 CYCLE 表示整数循环周期，DISPlay 表示全屏，如果没有指定<interval>，则默认 DISPlay。

➤ **返回格式:**

查询返回 1.120e000，单位为 V。

:MEASure:VRMS?

➤ **命令格式:**

:MEASure:VRMS? [<interval>][,<source>]

➤ **功能描述:**

用于测量指定通道波形的均方根值。其中<source>指定通道，取值为 CHANnel1、CHANnel2、MATH，如果没有指定<source>，则默认当前通道；<interval>指定测量间距，取值为 CYCLE、DISPlay，其中 CYCLE 表示整数循环周期，DISPlay 表示全屏，如果没有指定<interval>，则默认 DISPlay。

➤ **返回格式:**

查询返回 1.230e000，单位为 V。

:MEASure:AREa?

➤ **命令格式:**

:MEASure:AREa? [<interval>][,<source>]

➤ **功能描述:**

用于测量指定通道波形的面积。其中<source>指定通道，取值为 CHANnel1、CHANnel2、MATH，如果没有指定<source>，则默认当前通道；<interval>指定测量间距，取值为 CYCLE、DISPlay，其中 CYCLE 表示整数循环周期，DISPlay 表示全屏，如果没有指定<interval>，则默认 DISPlay。

➤ **返回格式:**

查询返回 3.456e002，单位为 Vs。

:MEASure:OVERshoot?

➤ **命令格式:**

:MEASure:OVERshoot? [<source>]

➤ **功能描述:**

用于测量指定通道波形的过冲。其中<source>取值为 CHANnel1、CHANnel2、MATH，省略表示当前通道。

➤ **返回格式:**

查询返回 1.230e002，单位为 V。

:MEASure:PREShoot?

➤ **命令格式:**

:MEASure:PREShoot? [<source>]

- **功能描述:**
用于测量指定通道波形的预冲。其中<source>取值为 CHANne11、CHANne12、MATH，省略表示当前通道。
- **返回格式:**
查询返回 1.230e-002，单位为 V。

:MEASure:FREQuency?

- **命令格式:**
:MEASure:FREQuency? [<source>]
- **功能描述:**
用于测量指定通道波形的频率。其中<source>取值为 CHANne11、CHANne12、MATH，省略表示当前通道。
- **返回格式:**
查询返回 2.000e003，单位 Hz。

:MEASure:RISetime?

- **命令格式:**
:MEASure:RISetime? [<source>]
- **功能描述:**
用于测量指定通道波形的上升时间。其中<source>取值为 CHANne11、CHANne12、MATH，省略表示当前通道。
- **返回格式:**
查询返回 5.000e-005，单位 s。

:MEASure:FALLtime?

- **命令格式:**
:MEASure:FALLtime? [<source>]
- **功能描述:**
用于测量指定通道波形的下降时间。其中<source>取值为 CHANne11、CHANne12、MATH，省略表示当前通道。
- **返回格式:**
查询返回 5.000e-005，单位 s。

:MEASure:PERiod?

- **命令格式:**
:MEASure:PERiod? [<source>]
- **功能描述:**
用于测量指定通道波形的周期。其中<source>取值为 CHANne11、CHANne12、MATH，省略表示当前通道。
- **返回格式:**
查询返回 5.000e-003，单位 s。

:MEASure:PWIDth?

- **命令格式:**
:MEASure:PWIDth? [<source>]

- **功能描述:**
用于测量指定通道波形的正脉宽。其中<source>取值为 CHANne11、CHANne12、 MATH, 省略表示当前通道。
- **返回格式:**
查询返回 5.000e-003, 单位 s。

:MEASure:NWIDth?

- **命令格式:**
:MEASure:NWIDth? [<source>]
- **功能描述:**
用于测量指定通道波形的负脉宽。其中<source>取值为 CHANne11、CHANne12、 MATH, 省略表示当前通道。
- **返回格式:**
查询返回 5.000e-003, 单位 s。

:MEASure:PULSes?

- **命令格式:**
:MEASure:PULSes? [<source>]
- **功能描述:**
用于测量指定通道波形的正脉冲个数。其中<source>取值为 CHANne11、CHANne12, 省略表示当前通道。
- **返回格式:**
查询返回 2.000e+000, 表示 2 个脉冲。

:MEASure:FRR?

- **命令格式:**
:MEASure:FRR? [<source1>, <source2>]
- **功能描述:**
用于测量<source1>与<source2>第一个上升沿之间的时间。其中<source>取值为 CHANne11、CHANne12、 MATH, 省略表示当前通道。
- **返回格式:**
查询返回 5.000e-003, 单位 s。

:MEASure:FRF?

- **命令格式:**
:MEASure:FRF? [<source1>, <source2>]
- **功能描述:**
用于测量<source1>第一个上升沿与<source2>第一个下降沿之间的时间。其中<source>取值为 CHANne11、CHANne12、 MATH, 省略表示当前通道。
- **返回格式:**
查询返回 5.000e-003, 单位 s。

:MEASure:FFR?

- **命令格式:**

:MEASure:FFR? [<source1>, <source2>]

➤ **功能描述:**

用于测量<source1>第一个下降沿与<source2>第一个上升沿之间的时间。其中<source>取值为 CHANne11、CHANne12、 MATH, 省略表示当前通道。

➤ **返回格式:**

查询返回 5.000e-003, 单位 s。

:MEASure:FFF?

➤ **命令格式:**

:MEASure:FFF? [<source1>, <source2>]

➤ **功能描述:**

用于测量<source1>与<source2>第一个下降沿之间的时间。其中<source>取值为 CHANne11、CHANne12、 MATH, 省略表示当前通道。

➤ **返回格式:**

查询返回 5.000e-003, 单位 s。

:MEASure:LRR?

➤ **命令格式:**

:MEASure:LRR? [<source1>, <source2>]

➤ **功能描述:**

用于测量<source1>第一个上升沿与<source2>最后一个上升沿之间的时间。其中<source>取值为 CHANne11、CHANne12、 MATH, 省略表示当前通道。

➤ **返回格式:**

查询返回 5.000e-003, 单位 s。

:MEASure:LRF?

➤ **命令格式:**

:MEASure:LRF? [<source1>, <source2>]

➤ **功能描述:**

用于测量<source1>第一个上升沿与<source2>最后一个下降沿之间的时间。其中<source>取值为 CHANne11、CHANne12、 MATH, 省略表示当前通道。

➤ **返回格式:**

查询返回 5.000e-003, 单位 s。

:MEASure:LFR?

➤ **命令格式:**

:MEASure:LFR? [<source1>, <source2>]

➤ **功能描述:**

用于测量<source1>第一个下降沿与<source2>最后一个上升沿之间的时间。其中<source>取值为 CHANne11、CHANne12、 MATH, 省略表示当前通道。

➤ **返回格式:**

查询返回 5.000e-003, 单位 s。

:MEASure:LFF?

➤ **命令格式:**

:MEASure:LFF? [<source1>, <source2>]

➤ **功能描述:**

用于测量<source1>第一个下降沿与<source2>最后一个下降沿之间的时间。其中<source>取值为 CHANne11、CHANne12、MATH，省略表示当前通道。

➤ **返回格式:**

查询返回 5.000e-003，单位 s。

TRIGger 命令

用于控制触发器扫描模式和触发器规范，触发决定了示波器何时开始采集数据和显示波形。

触发控制

:TRIGger:MODE

➤ **命令格式:**

:TRIGger:MODE <mode>

:TRIGger:MODE?

➤ **功能描述:**

用于设置触发方式。

<mode>分别为 EDGE（边沿触发）、PULSe（脉宽触发）、VIDeo（视频触发）、SLOPe（斜率触发）、ALTErnat（交替触发）。

➤ **返回格式:**

查询返回触发方式。

➤ **举例:**

:TRIGger:MODE EDGE 设置边沿触发

:TRIGger:MODE? 查询返回 EDGE

:TRIGger:SWEep

➤ **命令格式:**

:TRIGger:SWEep {AUTO|NORMal|SINGle}

:TRIGger:SWEep?

➤ **功能描述:**

用于选择触发扫描模式。

AUTO（自动）：在没有触发条件情况下，内部将产生触发信号，强制触发。

NORMal（普通）：只有满足触发条件时才能触发。

SINGle（单次）：在符合触发条件情况下进行一次触发，然后停止。

- **返回格式:**
查询返回触发扫描模式 {AUTO|NORMAl|SINGle}。
- **举例:**
:TRIGger:SWEep AUTO 设置通道一为自动触发模式
:TRIGger:SWEep? 查询返回 AUTO

:TRIGger:LEVel:ASETup

- **命令格式:**
:TRIGger:LEVel:ASETup
- **功能描述:**
用于将触发电平设置于信号幅值的垂直中点处。
- **举例:**
:TRIG:LEVel:ASETup 将触发电平位置置于中心处

:TRIGger:STATus?

- **命令格式:**
:TRIGger:STATus?
- **功能描述:**
查询当前的示波器触发运行状态。
- **返回格式:**
查询返回 STOP/ARMED/READY/TRIGED/AUTO/SCAN /RESET / REPLAY/ WAIT。
- **举例:**
:TRIGger:STATus? 查询返回 AUTO

:TRIGger:LEVel

- **命令格式:**
:TRIGger:LEVel <level>
:TRIGger:LEVel?
- **功能描述:**
用于设置正常触发的触发电平值。<level>数值必须根据幅格挡、屏幕信息换算后设置。
- **返回格式:**
查询返回<level>的设置值，单位 V。
- **举例:**
:TRIG:LEV 2 设置触发的触发电平为 2V
:TRIG:LEV? 查询返回 2.000e000

:TRIGger:LEVel:LOW

- **命令格式:**
:TRIGger:LEVel:LOW <level>

- **功能描述:**
用于设置耦合方式, DC (直流)、AC (交流)、LF (低频抑制)、HF (高频抑制)、NOISE (噪声抑制)。
只有 VIDEo 不支持。
- **返回格式:**
查询返回耦合方式 {DC|AC|LF|HF|NOISE}。
- **举例:**
:TRIGger:COUPling AC 设置边沿触发为交流
:TRIGger:COUPling? 查询返回 AC

边沿触发

:TRIGger:EDGE:SLOPe

- **命令格式:**
:TRIGger:EDGE:SLOPe {POSitive|NEGative|ALTerNation}
:TRIGger:EDGE:SLOPe?
- **功能描述:**
用于设置触发的边沿类型, POSitive (上升沿)、NEGative (下降沿)、ALTerNation (上升下降沿)。
- **返回格式:**
查询返回触发的边沿类型 { POSitive | NEGative | ALTerNation }。
- **举例:**
:TRIGger:EDGE:SLOP POS 设置边沿触发为上升沿
:TRIGger:EDGE:SLOP? 查询返回 POSitive

脉宽触发

:TRIGger:PULSe:QUALifier

- **命令格式:**
:TRIGger:PULSe:QUALifier {GREaterthan | LESSthan | RANG}
:TRIGger:PULSe:QUALifier?
- **功能描述:**
用于设置脉冲时间设置条件, GREaterthan (大于)、LESSthan (小于)、RANG (范围之内)。
- **返回格式:**
查询返回 { GREaterthan | LESSthan | RANG }。
- **举例:**
:TRIGger:PULSe:QUALifier GRE 设置脉冲条件为大于
:TRIGger:PULSe:QUALifier? 查询返回 GREaterthan

:TRIGger:PULSe:POLarity

➤ 命令格式:

:TRIGger:PULSe:POLarity {POSitive | NEGative}

:TRIGger:PULSe:POLarity?

➤ 功能描述:

用于设置脉冲极性，POSitive（正脉宽）、NEGative（负脉宽）。

➤ 返回格式:

查询返回{ POSitive | NEGative }。

➤ 举例:

:TRIGger:PULSe:POL POS

设置脉冲极性为正脉宽

:TRIGger:PULSe:POL?

查询返回 POSitive

:TRIGger:PULSe:TIME

➤ 命令格式:

:TRIGger:PULSe:TIME <time>

:TRIGger:PULSe:TIME?

➤ 功能描述:

用于设置脉宽触发时间间隔。

➤ 返回格式:

查询返回当前时间间隔，单位 s。

➤ 举例:

:TRIGger:PULSe:TIME 1

设置脉冲宽度为 1s

:TRIGger:PULSe:TIME?

查询返回 1.000e000

:TRIGger:PULSe:TIME:UPPer

➤ 命令格式:

:TRIGger:PULSe:TIME:UPPer <time>

:TRIGger:PULSe:TIME:UPPer?

➤ 功能描述:

用于设置脉宽触发时间上限。

➤ 返回格式:

查询返回当前时间上限，单位 s。

➤ 举例:

:TRIGger:PULSe:TIME:UPPer 1

设置脉冲宽度为 1s

:TRIGger:PULSe:TIME:UPPer?

查询返回 1.000e000

:TRIGger:PULSe:TIME:LOWer

➤ 命令格式:

:TRIGger:PULSe:TIME:LOWer <time>

:TRIGger:PULSe:TIME:LOWer?

➤ **功能描述:**

用于设置脉宽触发时间下限。

➤ **返回格式:**

查询返回当前时间下限，单位 s。

➤ **举例:**

:TRIGger:PULSe:TIME:LOWer 1

设置脉冲宽度为 1s

:TRIGger:PULSe:TIME:LOWer?

查询返回 1.000e000

视频触发

:TRIGger:VIDeo:MODE

➤ **命令格式:**

:TRIGger:VIDeo:MODE { ODD | EVEN | LINE | ALINes }

:TRIGger:VIDeo:MODE?

➤ **功能描述:**

用于设置视频触发同步方式，ODD（奇数）、EVEN（偶数）、LINE（指定行）、ALINes（所有行）。

➤ **返回格式:**

查询返回 { ODD | EVEN | LINE | ALIN }。

➤ **举例:**

:TRIGger:VIDeo:MODE ODD

设置视频触发同步模式为奇数场

:TRIGger:VIDeo:MODE?

查询返回 ODD

:TRIGger:VIDeo:STANdard

➤ **命令格式:**

:TRIGger:VIDeo:STANdard { NTSC | PAL }

:TRIGger:VIDeo:STANdard?

➤ **功能描述:**

用于设置视频标准。

➤ **返回格式:**

查询返回 { NTSC | PAL }。

➤ **举例:**

:TRIGger:VIDeo:STANdard NTSC

设置视频标准为 NTSC

:TRIGger:VIDeo:STANdard?

查询返回 NTSC

:TRIGger:VIDeo:LINE

➤ **命令格式:**

:TRIGger:VIDeo:LINE <value>

:TRIGger:VIDEO:LINE?

➤ **功能描述:**

用于设置视频同步的指定行数。<value>表示指定的行数，范围和视频标准相关。

➤ **返回格式:**

查询返回当前所指定的行数。

➤ **举例:**

:TRIG:VIDEO:LINE 50

设置视频同步指定的行数为 50

:TRIG:VIDEO:LINE?

查询返回 50

斜率触发

:TRIGger:SLOPe:QUALifier

➤ **命令格式:**

:TRIGger:SLOPe:QUALifier {GREaterthan | LESSthan | RANG }

:TRIGger:SLOPe:QUALifier?

➤ **功能描述:**

用于设置斜率时间设置条件，GREaterthan（大于）、LESSthan（小于）、RANG（范围之内）。

➤ **返回格式:**

查询返回 {GREaterthan | LESSthan | RANG}。

➤ **举例:**

:TRIGger:SLOPe:QUALifier GRE

设置斜率条件为大于

:TRIGger:SLOPe:QUALifier?

查询返回 GREaterthan

:TRIGger:SLOPe:SLOPe

➤ **命令格式:**

:TRIGger:SLOPe:SLOPe {POSitive|NEGative}

:TRIGger:SLOPe:SLOPe?

➤ **功能描述:**

用于设置触发斜率类型，POSitive（上升）、NEGative（下降）。

➤ **返回格式:**

查询返回 {POSitive|NEGative}。

➤ **举例:**

:TRIGger:SLOPe:SLOPe POS

斜率触发为上升模式

:TRIGger:SLOPe:SLOPe?

查询返回 POSitive

:TRIGger:SLOPe:TIME

➤ **命令格式:**

:TRIGger:SLOPe:TIME <time>

:TRIGger:SLOPe:TIME?

➤ **功能描述:**
用于设置斜率触发模式的时间间隔。

➤ **返回格式:**
查询返回当前时间间隔, 单位 s。

➤ **举例:**
:TRIGger:SLOPe:TIME 1 斜率触发模式时间间隔设置 1s
:TRIGger:SLOPe:TIME? 查询返回 1.000e000

:TRIGger:SLOPe:TIME:UPPer

➤ **命令格式:**
:TRIGger:SLOPe:TIME:UPPer <time>
:TRIGger:SLOPe:TIME:UPPer?

➤ **功能描述:**
用于设置斜率触发时间上限。

➤ **返回格式:**
查询返回当前时间上限, 单位 s。

➤ **举例:**
:TRIGger:SLOPe:TIME:UPPer 1 设置脉冲宽度为 1s
:TRIGger:SLOPe:TIME:UPPer? 查询返回 1.000e000

:TRIGger:SLOPe:TIME:LOWer

➤ **命令格式:**
:TRIGger:SLOPe:TIME:LOWer <time>
:TRIGger:SLOPe:TIME:LOWer?

➤ **功能描述:**
用于设置斜率触发时间下限。

➤ **返回格式:**
查询返回当前时间下限, 单位 s。

➤ **举例:**
:TRIGger:SLOPe:TIME:LOWer 1 设置脉冲宽度为 1s
:TRIGger:SLOPe:TIME:LOWer? 查询返回 1.000e000

:TRIGger:SLOPe:THReshold

➤ **命令格式:**
:TRIGger:SLOPe:THReshold {LOW|HIGH|LH}
:TRIGger:SLOPe:THReshold?

➤ **功能描述:**
用于设置斜率触发模式的阈值模式。

➤ **返回格式:**

查询返回 {LOW|HIGH|LH}。

- **举例：**
:TRIGger:SLOPe:THR HIGH 斜率触发阈值为高模式
:TRIGger:SLOPe:THR? 查询返回 HIGH

交替触发

:TRIGger:ALternat:TYPE

- **命令格式：**
:TRIGger:ALternat:TYPE { EDGE | PULSe | SLOPe}
:TRIGger:ALternat:TYPE?
- **功能描述：**
用于设置交替触发模式，分别为边沿、脉宽、斜率。
- **返回格式：**
查询返回 { EDGE | PULSe | SLOPe }。
- **举例：**
:TRIGger:ALternat:TYPE EDGE 设置交替触发模式为边沿触发
:TRIGger:ALternat:TYPE? 查询返回 EDGE

CURSor 命令

用于设置光标参数，对屏幕波形数据进行测量。

:CURSor:MODE

- **命令格式：**
:CURSor:MODE { TRACK | INdependent }
:CURSor:MODE?
- **功能描述：**
用于设置光标模式的光标模式。
TRACK（跟踪）、INdependent（独立）。
- **返回格式：**
查询返回 { TRACK | INdependent }。
- **举例：**
:CURSor:MODE TRACK 设置光标为跟踪模式
:CURSor:MODE? 查询返回 TRACK

:CURSor:TYPE

- **命令格式：**
:CURSor:TYPE { AMPLitude | TIME | CLOSe }

:CURSor:TYPE?

➤ **功能描述:**

用于设置光标模式的光标类型。

AMPlitude (幅度)、TIME (时间)、CLOSe (关闭)。

➤ **返回格式:**

查询返回 {AMPlitude | TIME | CLOSe }。

➤ **举例:**

:CURSor:TYPE AMP 设置光标类型为幅度
:CURSor:TYPE? 查询返回 AMPlitude

:CURSor:SOURce

➤ **命令格式:**

:CURSor:SOURce <source>

:CURSor:SOURce?

➤ **功能描述:**

用于设置光标测量时的源。

<source>取值 { CHANne11 | CHANne12 | REF | MATH}。

➤ **返回格式:**

查询返回 { CHANne11 | CHANne12 | REF | MATH}。

➤ **举例:**

:CURSor:SOURce CHAN1 设置通道一为光标源
:CURSor:SOURce? 查询返回 CHANne11

:CURSor:CURA

➤ **命令格式:**

:CURSor:CURA <value>

:CURSor:CURA?

➤ **功能描述:**

用于设置光标线 A 的横向位置或者纵向位置。与 CURSor:TYPE 指令相关，幅度表示设置纵向位置，时间表示设置横向位置。竖线范围[0, 699]，横线范围[28, 227]。

➤ **返回格式:**

查询返回光标线 A 位置。

➤ **举例:**

:CURSor:CURA 50 设置手动光标线 A 位置为 50
:CURSor:CURA? 查询返回 50

:CURSor:CURB

➤ **命令格式:**

:CURSor:CURB <value>

:CURSor:CURB?

➤ **举例：**

:CURSor:XY? 查询返回 XY 模式全部光标测量参数，例如"1.0E+02, 3.0E+01, "

ACQuire 命令

用于设置示波器的采样方式。

:ACQuire:TYPE

➤ **命令格式：**

:ACQuire:TYPE {NORMal | AVERAge | PEAKdetect | HRESolution }
:ACQuire:TYPE?

➤ **功能描述：**

用于设置示波器的采样获取方式。

NORMal（正常）、AVERAge（平均）、PEAKdetect（峰值）、HRESolution（高分辨率）。

➤ **返回格式：**

查询返回 {NORMal | AVERAge | PEAKdetect | ENVELOpe | HRESolution }。

➤ **举例：**

:ACQ:TYPE AVER 设置获取方式为平均。
:ACQ:TYPE? 查询返回 AVERAge。

:ACQuire:AVERAges:COUNT

➤ **命令格式：**

:ACQuire:AVERAges:COUNT <count>
:ACQuire:AVERAges:COUNT?

➤ **功能描述：**

用于设置示波器在平均采样时的平均采样次数。其中<count>以 2 的 N 次幂步进，在 2 到 8192 间取值， $1 \leq N \leq 30$ 。

➤ **返回格式：**

查询返回当前平均采样次数。

➤ **举例：**

:ACQ:AVER:COUN 32 设置平均采样平均次数为 32。
:ACQ:AVER:COUN? 查询返回 32。

:ACQuire:MODE

➤ **命令格式：**

:ACQuire:MODE {EQUIvalent | REALtime}
:ACQuire:MODE?

➤ **功能描述：**

用于设置采样方式，分别为等效采样和实时采样。

➤ **返回格式：**

查询返回 {EQUIvalent | REALtime}。

➤ **举例：**

:ACQ:MODE EQU 设置采样方式为等效采样。

:ACQ:MODE?

查询返回 EQUIVALENT

:ACQUIRE:FAST

➤ **命令格式:**

:ACQUIRE:FAST {{1 | ON} | {0 | OFF}}

:ACQUIRE:FAST?

➤ **功能描述:**

用于打开或关闭快速采集。

➤ **返回格式:**

查询返回快速采集状态，0 表示打开快速采集，1 表示关闭快速采集。

➤ **举例:**

:ACQUIRE:FAST ON

打开快速采集

:ACQUIRE:FAST?

查询返回 1，表示打开快速采集

DISPLAY 命令

用于设置或查询示波器的显示功能或数据。

:DISPLAY:DATA?

➤ **命令格式:**

:DISPLAY:DATA?

➤ **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 BMP 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

:DISPLAY:FORMAT

➤ **命令格式:**

:DISPLAY:FORMAT { VECTORS | DOTS }

:DISPLAY:FORMAT?

➤ **功能描述:**

用于设置采样点的显示格式。VECTORS（矢量显示）、DOTS（直接显示采样点）

➤ **返回格式:**

查询返回 { VECTORS | DOTS }。

➤ **举例:**

:DISPLAY:FORMAT VECT

设置采样点矢量显示

:DISPLAY:FORMAT?

查询返回 VECTORS

:DISPlay:GRADing:TIME

- **命令格式:**
:DISPlay:GRADing:TIME {CLOSe|SHORt|LONG|INFinite}
:DISPlay:GRADing:TIME?
- **功能描述:**
用于设置或查询余辉时间。
- **返回格式:**
查询返回 {CLOSe|SHORt|LONG|INFinite}。
- **举例:**
:DISPlay:GRADing:TIME SHORt 设置余辉时间为短余辉
:DISPlay:GRADing:TIME? 查询返回 SHORt

:DISPlay:GRID

- **命令格式:**
:DISPlay:GRID {FULL|HALF|CORSS|NONE }
:DISPlay:GRID?
- **功能描述:**
用于设置或查询屏幕显示的网格类型。
- **返回格式:**
查询返回 {FULL|HALF|CORSS|NONE }。
- **举例:**
:DISPlay:GRID HALF 设置网格类型半格模式，只显示网格，不显示坐标。
:DISPlay:GRID? 查询返回 HALF

:DISPlay:GRID:BRIGhtness

- **命令格式:**
:DISPlay:GRID:BRIGhtness <count>
:DISPlay:GRID:BRIGhtness?
- **功能描述:**
用于设置网格亮度，<count>取值为 10~100，数字越大网格越亮。
- **返回格式:**
查询返回当前网格亮度。
- **举例:**
:DISPlay:GRID:BRIGhtness 50 设置网格亮度 50
:DISPlay:GRID:BRIGhtness? 查询返回 50

:DISPlay:WAVE:BRIGhtness

- **命令格式:**
:DISPlay:WAVE:BRIGhtness <count>
:DISPlay:WAVE:BRIGhtness?
- **功能描述:**
用于设置波形亮度，<count>取值为 10~100，数字越大波形越亮。

- **返回格式:**
查询返回当前波形亮度。
- **举例:**
:DISPlay:WAVE:BRIGhtness 50 设置波形亮度 50
:DISPlay:WAVE:BRIGhtness? 查询返回 50

:DISPlay:CLEAr

- **命令格式:**
:DISPlay:CLEAr
- **功能描述:**
用于清除示波器屏幕上的波形，如果示波器处于 RUN 状态，则清除后继续显示新波形。

:DISPlay:TYPE

- **命令格式:**
:DISPlay:TYPE {XY12 |YT}
:DISPlay:TYPE?
- **功能描述:**
用于设置时基显示类型为 XY12 (X-Y 方式：在水平轴上显示通道 1 幅值，垂直轴上显示通道 2 幅值)、YT (Y-T 方式：显示垂直电压与水平时间的相对关系)。
- **返回格式:**
查询返回 XY12、YT。
- **举例:**
:DISP:TYPE YT 设置时基格式为 YT 方式。
:DISP:TYPE? 查询返回 YT。

WAVeform 命令

用于读取示波器屏幕中的波形数据及相关参数。

:WAVeform:MODE

- **命令格式:**
:WAVeform:MODE {NORMAl | RAW}
:WAVeform:MODE?
- **功能描述:**
NORMAl：读取当前屏幕显示的波形数据，此波形数据点数为固定点数。
RAW：读取内存中的波形数据，此波形数据点数和存储深度相关。注意：内存中的数据必须在示波器停止状态下才能进行读取，MATH 通道下该指令无效。
- **返回格式:**
查询返回 {NORMAl | RAW}。
- **举例:**
:WAVeform:MODE RAW 设置波形数据的读取模式为内存读取
:WAVeform:MODE? 查询返回 RAW

:WAVeform:FORMat

➤ 命令格式:

:WAVeform:FORMat { WORD | BYTE | ASCII }

:WAVeform:FORMat?

➤ 功能描述:

示波器默认波形数据格式 AD 波形点数据

BYTE: 返回 AD 数据, 一个波形点占一个字节 (即 8 位)。

WORD: 返回 AD 数据, 一个波形点占两个字节 (即 16 位), 低 8 位有效, 高 8 位为 0。

ASCII: 返回波形以科学计数形式返回各波形点的实际电压值, 各电压值之间以逗号分隔且符合 IEEE488.2 二进制数据格式。

例如: #412342.00000E+01, 2.20000E+01, 2.30000E+01..... \n。

➤ 返回格式:

查询返回 { WORD | BYTE | ASCII }。

➤ 举例:

:WAVeform:FORMat BYTE

波形 AD 数据的返回格式为单字节模式

:WAVeform:FORMat?

查询返回 BYT

:WAVeform:START

➤ 命令格式:

:WAVeform:START <start>

:WAVeform:START?

➤ 功能描述:

设置或查询波形数据读取的起始位置, < start>整型数据类型。

NORMAL: 1 到 1400

RAW: 1 至当前最大的存储深度点数

➤ 返回格式:

查询返回起始位置。

➤ 举例:

:WAVeform:START 200

设置波形数据读取的起始点为 200

:WAVeform:START?

查询返回 200

:WAVeform:STOP

➤ 命令格式:

:WAVeform:STOP <stop>

:WAVeform:STOP?

➤ 功能描述:

设置或查询波形数据读取的截止位置, < stop>整型数据类型。

NORMAL: < stop>范围 1 到 1400

RAW: < stop>范围 1 至当前最大的存储深度点数

➤ 返回格式:

查询返回截止位置。

➤ 举例:

:WAVeform:STOP 400 设置波形数据读取的结束点为 400
:WAVeform:STOP? 查询返回 400

:WAVeform:SOURce

➤ **命令格式:**

:WAVeform:SOURce {CHANnel<n>| MATH}
:WAVeform:SOURce?

➤ **功能描述:**

用于设置当前要查询波形数据的信号源，如果不发送该指令，表示要查询当前通道的波形数据。

➤ **返回格式:**

查询返回 { CHANnel1 | CHANnel2 | MATH }。

➤ **举例:**

:WAVeform:SOURce CHAN1 设置当前要查询波形数据的信号源为通道一
:WAVeform:SOURce? 查询返回 CHANnel1

:WAVeform:POINts

➤ **命令格式:**

:WAVeform:POINts <points>
:WAVeform:POINts?

➤ **功能描述:**

用于设置需要返回的波形点数，默认值为 0。

➤ **返回格式:**

查询返回需要返回的波形点数。

➤ **举例:**

:WAVeform:POINts 120 设置需要返回的 120 个波形点数
:WAVeform:POINts? 查询返回 120

:WAVeform:DATA?

➤ **命令格式:**

:WAVeform:DATA?

➤ **功能描述:**

用于读取指定数据源中的波形数据。

➤ **返回格式:**

WAVeform:POINts 指定数量的波形数据，波形数据源与 :WAVeform:SOURce 相关，数据格式与 WAVeform:FORMat 相关，返回的数据符合 IEEE 488.2 #格式的二进制数据。

➤ **举例:**

获得指定数据源的波形数据指令顺序执行如下:

◆ 获得屏幕波形数据流程

:WAVeform:SOURce CHAN1 设置当前要查询波形数据的信号源为通道一
:WAVeform:MODE NORMal 设置读取屏幕显示波形数据
:WAVeform:FORMat BYTE 波形数据的返回格式为 AD 单字节模式
:WAVeform:DATA? 获得波形数据

◆ 获得内存波形数据流程，此流程只有在停止状态下有效

:WAVeform:SOURce CHAN1 设置当前要查询波形数据的信号源为通道一
 :WAVeform:MODE RAW 设置读取内存波形数据
 :WAVeform:FORMat BYTE 波形数据的返回格式为 AD 单字节模式
 :WAVeform:POINts 5000 读取内存波形点数为 5000
 :WAVeform:DATA? 获得内存中一块的波形数据

说明: 分批次读取内存数据时, 每次读回的数据只是内存中一块区域的数据, 相邻两块间的波形数据连续, 每块数据符合 IEEE 488.2 #格式的数据格式

:WAVeform:PREamble?

➤ 命令格式:

:WAVeform:PREamble?

➤ 功能描述:

查询返回当前系统波形设置参数。

➤ 返回格式:

查询返回以逗号 “,” 隔开。

返回的数据格式: Format, Type, Points, Count, Xinc, Xor, Xref, Yinc, Yor, Yref。

Format: BYTE (0)、WORD (1)、ASCII (2)。

Type: NORMAL (0)、PEAK (1)、AVER (2)、ENVELOPE (3)、HRESOLUTION (4)。

Points: 需要返回的波形数据点数。

Count: 在平均采样下为平均次数, 其它模式下为 1。

Xinc: 波形数据源 X 方向两点之间的时间差。

Xor: 触发点相对时间。

Xref: X 基准。

Yinc: Y 方向单位电压。

Yor: Y 方向相对 YREF 的零点位置。

Yref: Y 方向参考值, 屏幕中点。

➤ 举例:

:WAVeform:PREamble? 返回 1, 0, 0, 1, 8.000e-009, -6.000e-006, 0, 4.000e-002, 0.000e000, 100

:WAVeform:XINcrement?

➤ 命令格式:

:WAVeform:XINcrement?

➤ 功能描述:

用于查询当前选中通道源 X 方向上相邻两点之间的时间间隔。

返回值与当前的数据读取模式相关:

NORMAL 模式下, XINcrement=TimeScale/时基挡横向的波形点数。

RAW 模式下, XINcrement=1/SampleRate。

➤ 返回格式:

查询返回时基数, 单位 s。

➤ 举例:

:WAV:XINC? 查询返回 3.000e-003

- 存储内部设置文件名必须是 "system-set-up01.set"~ "system-set-up255.set", 最大 255 个文件。
- 存储内部文 bsv 文件文件名必须是 "wave01.bsv"~ "wave255.bsv", 最大 255 个文件。

:FILE:SAVE

➤ 命令格式:

:FILE:SAVE <filename>[, <source>][, <disk>]

➤ 功能描述:

用于保存通道波形或者设置数据到文件中。

<filename>表示文件名称, 文件名称必须是字符串类型数据, 需带双引号, 例如 "test.bsv"

- 文件名为*.bsv 表示以后缀名格式保存某个通道的波形到该文件中
- 文件名为*.set 表示某个设置数据到该文件中

<source >表示物理通道 {CHANnel1 | CHANnel2 }, 可选参数, 保存波形数据时才有效。

- CHANnel1 表示通道 1
- CHANnel2 表示通道 2

<disk>表示存储介质 { FLASH | UDISK }, 可选参数, 忽略表示 FLASH 内部数据。

- FLASH 表示内部数据
- UDISK 表示 U 盘数据

➤ 举例:

FILE:SAVE "test.bsv", CHANnel1, UDISK	通道 1 波形数据保存成 U 盘的 test.bsv 文件
FILE:SAVE "system-set-up01.set"	示波器配置数据保存成内部介质 1 号位置
FILE:SAVE "wave01.bsv", CHANnel1, FLASH	通道 1 波形数据保存到内部介质
FILE:SAVE "wave01.bsv", CHANnel1	通道 1 波形数据保存到内部介质
FILE:SAVE "system-set-up01.set", FLASH	示波器配置数据保存成内部介质
FILE:SAVE "system-set-up01.set"	示波器配置数据保存成内部介质 1 号位置

注意:

- 存储内部设置文件名必须是 "system-set-up01.set"~ "system-set-up255.set", 最大 255 个文件。
- 存储内部文 bsv 文件文件名必须是 "wave01.bsv"~ "wave255.bsv", 最大 255 个文件。

RECORD 命令

用于示波器波形录制功能相关设置。

:RECORD:START

➤ 命令格式:

:RECORD:START { {1|ON} | {0|OFF} }
:RECORD:START?

➤ 功能描述:

用于设置波形录制 ON (开始) 或 OFF (停止)。

➤ 返回格式:

查询返回 1 或 0, 分别代表 ON 或 OFF。

➤ 举例:

:RECORD:START ON 开始波形录制

:RECOrd:STARt?

查询返回 1，表示已启动波形录制

:RECOrd:PLAY

➤ **命令格式:**

:RECOrd:PLAY { {1|ON} | {0|OFF} }

:RECOrd:PLAY?

➤ **功能描述:**

用于设置录制波形播放 ON（开始）或 OFF（停止）。

➤ **返回格式:**

查询返回 1 或 0，分别代表 ON 或 OFF。

➤ **举例:**

:RECOrd:PLAY ON

开始录制波形播放

:RECOrd:PLAY?

查询返回 1，表示已启动录制波形播放

:RECOrd:CURREnt

➤ **命令格式:**

:RECOrd:CURREnt <value>

:RECOrd:CURREnt?

➤ **功能描述:**

用于设置或查询录制波形播放的当前帧。

➤ **返回格式:**

查询返回录制波形播放的当前帧，整型数据。

➤ **举例:**

:RECOrd:CURREnt 100

设置波形播放的当前帧为 100

:RECOrd:CURREnt?

查询返回 100

PF 命令

用于示波器通过/失败测试功能相关设置。

:PF:ENABle

➤ **命令格式:**

:PF:ENABle { {1|ON} | {0|OFF} }

:PF:ENABle?

➤ **功能描述:**

用于设置或查询通过/失败测试功能 ON（打开）或 OFF（关闭）。

➤ **返回格式:**

查询返回 1 或 0，分别代表 ON 或 OFF。

➤ **举例:**

:PF:ENABle ON

打开通过/失败测试功能

:PF:ENABle?

查询返回 1

:PF:SOURCe

➤ 命令格式:

:PF:SOURCe <source>

:PF:SOURCe?

➤ 功能描述:

用于设置或查询通过/失败测试的测量源。

<source>: CHANnel<n>, 其中 n 取值 1、2。

➤ 返回格式:

查询返回 {CHANnel1 | CHANnel2}。

➤ 举例:

:PF:SOURCe CHANnel1

设置测量源为通道一

:PF:SOURCe?

查询返回 CHANnel1

:PF:OPERate

➤ 命令格式:

:PF:OPERate {RUN|STOP}

:PF:OPERate?

➤ 功能描述:

用于设置运行或停止通过/失败测试。

➤ 返回格式:

查询返回 {RUN|STOP}。

➤ 举例:

:PF:OPERate RUN

运行通过/失败测试

:PF:OPERate?

查询返回 RUN

:PF:OUTPut

➤ 命令格式:

:PF:OUTPut {PASS|FAILED}

:PF:OUTPut?

➤ 功能描述:

用于设置或查询通过/失败测试的输出。

➤ 返回格式:

查询返回 {PASS|FAILED}。

➤ 举例:

:PF:OUTPut PASS

设置通过/失败测试输出为通过

:PF:OUTPut?

查询返回 PASS

:PF:STOP:TYPe

➤ 命令格式:

:PF:STOP:TYPe {PCOUNT|FCOUNT}

:PF:STOP:TYPe?

➤ 功能描述:

用于设置或查询通过/失败测试停止类型。
PCOUNT 表示通过次数；FCOUNT 表示失败次数。

➤ **返回格式：**

查询返回 {PCOUNT|FCOUNT}。

➤ **举例：**

:PF:STOP:TYPE PCOUNT 设置通过/失败测试停止类型为通过次数
:PF:STOP:TYPE? 查询返回 PCOUNT

:PF:STOP:QUALifier

➤ **命令格式：**

:PF:STOP:QUALifier {LEQual | GEQual}
:PF:STOP:QUALifier?

➤ **功能描述：**

用于设置或查询通过/失败测试停止条件。
GEQual 表示大于等于；LEQual 表示小于等于。

➤ **返回格式：**

查询返回 {LEQual | GEQual}。

➤ **举例：**

:PF:STOP:QUALifier GEQual 设置通过/失败测试停止条件为 \geq
:PF:STOP:QUALifier? 查询返回 GEQual

:PF:STOP:THReshold

➤ **命令格式：**

:PF:STOP:THReshold <value>
:PF:STOP:THReshold?

➤ **功能描述：**

用于设置或查询通过/失败测试停止阈值。
<value>: 停止阈值，其范围 1~10000，具体范围根据示波器设备自适应。

➤ **返回格式：**

查询返回停止阈值，整型数据。

➤ **举例：**

:PF:STOP:THReshold 100 设置通过/失败测试停止阈值为 100
:PF:STOP:THReshold? 查询返回 100

:PF:TEMPlate:SOURce

➤ **命令格式：**

:PF:TEMPlate:SOURce <source>
:PF:TEMPlate:SOURce?

➤ **功能描述：**

用于设置或查询通过/失败测试模板设置的参考通道，。
<source>: {CHANnel<n> | REF }，其中 n 取值 1、2；如果参考通道设置成参考波形 (REF)，使用 :PF:TEMPlate:REF:LOAD 指令加载参考波形。

➤ **返回格式：**

查询返回 {CHANnel1 | CHANnel2 | REF }。

➤ **举例：**

:PF:TEMPlate:SOURce CHANnel1 设置模板设置的参考通道为通道一
:PF:TEMPlate:SOURce? 查询返回 CHANnel1

:PF:TEMPlate:REF:LOAD

➤ **命令格式：**

:PF:TEMPlate:REF:LOAD <filename>

➤ **功能描述：**

用于设置或查询通过/失败测试模板设置的参考通道加载指定文件波形。

➤ **举例：**

PF:TEMPlate:REF:LOAD "test.dat" 模板设置的参考通道加载 test.dat 波形文件

:PF:TEMPlate:X

➤ **命令格式：**

:PF:TEMPlate:X <value>

:PF:TEMPlate:X?

➤ **功能描述：**

用于设置或查询通过/失败测试模板设置的水平容限。

<value>: 水平容限, 其范围 1~100, 具体范围根据示波器设备自适应。

➤ **返回格式：**

查询返回模板设置的水平容限, 整型数据。

➤ **举例：**

:PF:TEMPlate:X 50 设置模板设置的水平容限为 50
:PF:TEMPlate:X? 查询返回 50

:PF:TEMPlate:Y

➤ **命令格式：**

:PF:TEMPlate:Y <value>

:PF:TEMPlate:Y?

➤ **功能描述：**

用于设置或查询通过/失败测试模板设置的垂直容限。

<value>: 垂直容限, 其范围 1~100, 具体范围根据示波器设备自适应。

➤ **返回格式：**

查询返回模板设置的垂直容限, 整型数据。

➤ **举例：**

:PF:TEMPlate:Y 50 设置模板设置的垂直容限为 50
:PF:TEMPlate:Y? 查询返回 50

:PF:RESult?

➤ **命令格式：**

:PF:RESult?

➤ **功能描述:**

用于查询通过/失败测试统计结果。

返回数据格式: <pass>, <failed>, <total>, 其中<pass>表示通过次数, <failed>表示失败的次数, <total>表示总的次数。

➤ **返回格式:**

查询返回通过/失败测试统计结果。

➤ **举例:**

:PF:RESult?

查询返回 35, 42, 77

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。

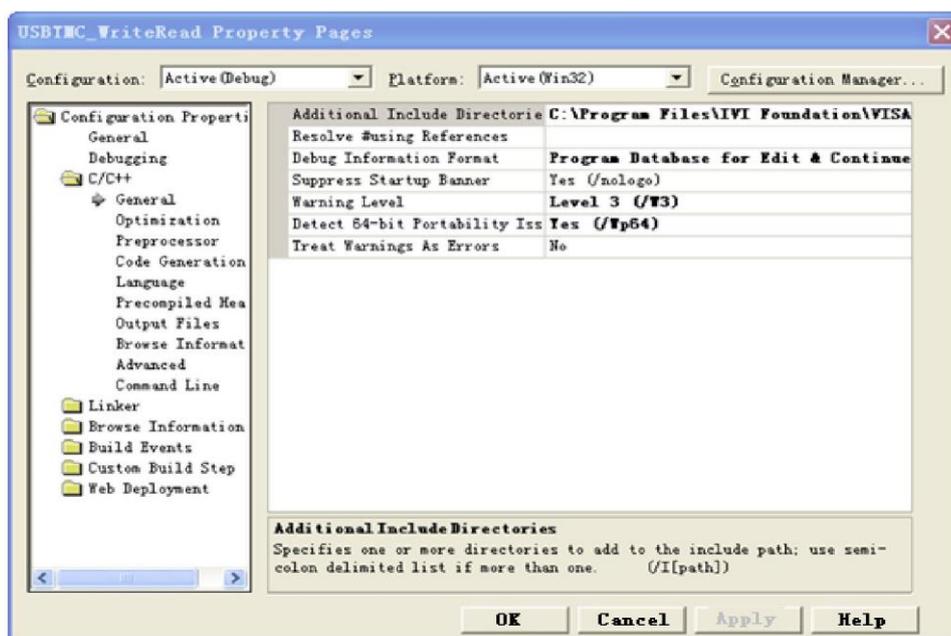
a) 静态库:

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:

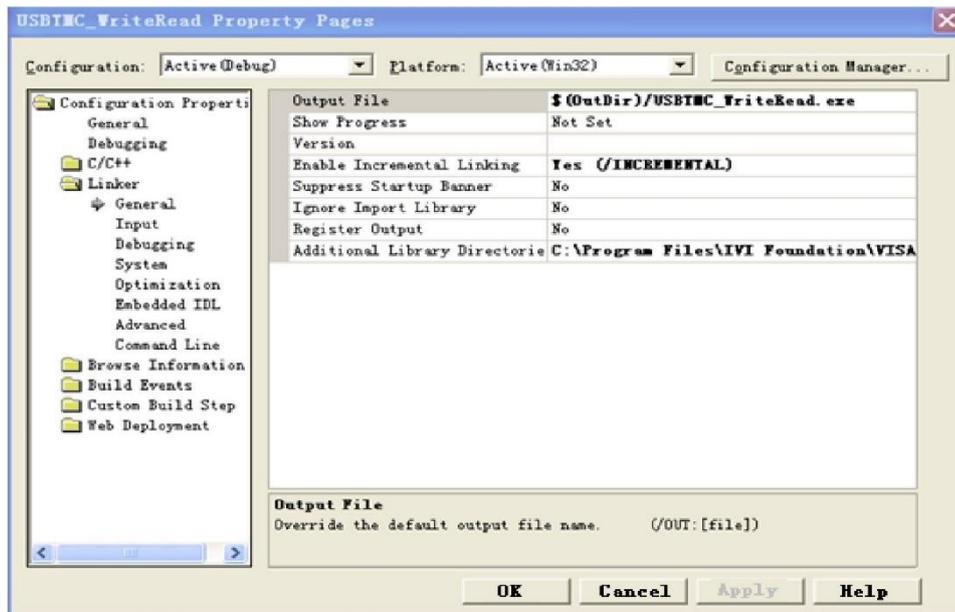
```
#include "visa.h"  
#pragma comment(lib, "visa32.lib")
```

b) 动态库:

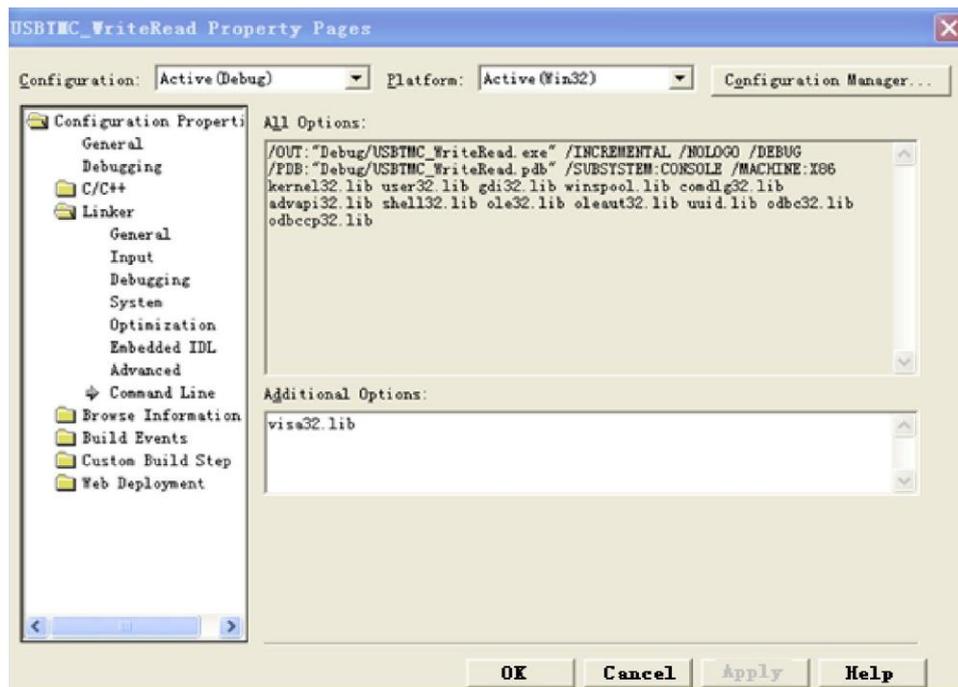
点击 "project>>properties", 在属性对话框左侧选择 "c/c++---General" 中, 将 "Additional Include Directories" 项的值设置为 NI-VISA 的安装路径, (例如: C:\Program Files\IVI Foundation\VISA\WinNT\include), 如下图所示:



在属性对话框左侧选择 "Linker-General", 并将 "Additional Library Directories" 项的值设置为 NI-VISA 的安装路径, (例如: C:\Program Files\IVI Foundation\VISA\WinNT\include), 如下图所示:



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



在 projectname.cpp 文件上添加 visa.h 文件:

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

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 * cmmmand = "*IDN?\n";
    status = viPrintf(instr, cmmmand);
    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

```

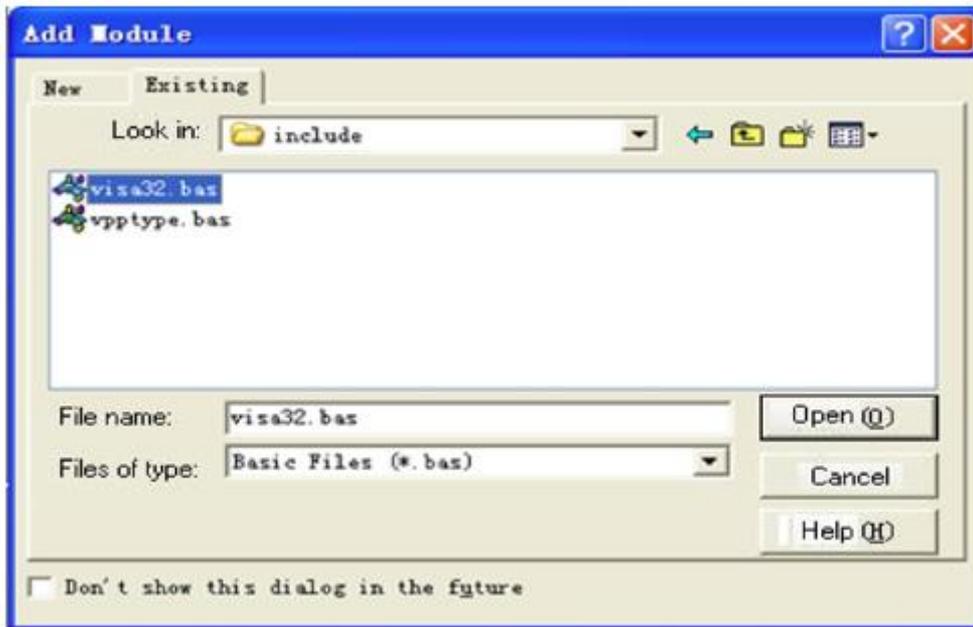
class Program
{
    void tcp_ip_test(string ip)
    {
        using (var rmSession = new ResourceManager())
        {
            try
            {
                var resource = string.Format("TCP/IP0::{0}::inst0::INSTR", ip);
                var mbSession = (MessageBasedSession)rmSession.Open(resource);
                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

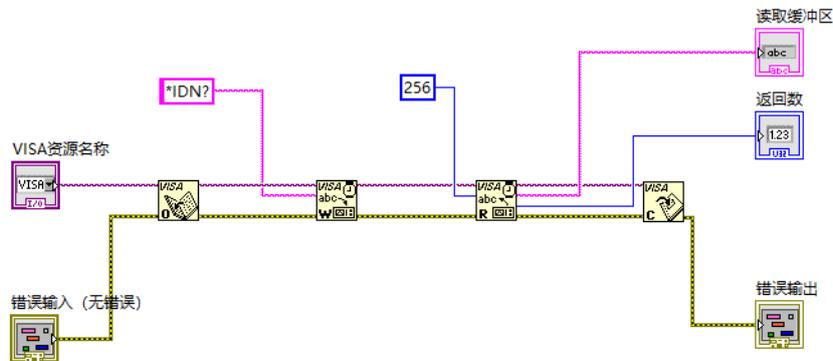
' First we will need to open the default resource manager.
status = viOpenDefaultRM(defaultRM)
If (status < VI_SUCCESS) Then
    resultTxt.Text = "Could not open a session to the VISA Resource Manager!"
    tcp_ip_test = status
ExitFunction
EndIf

' 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

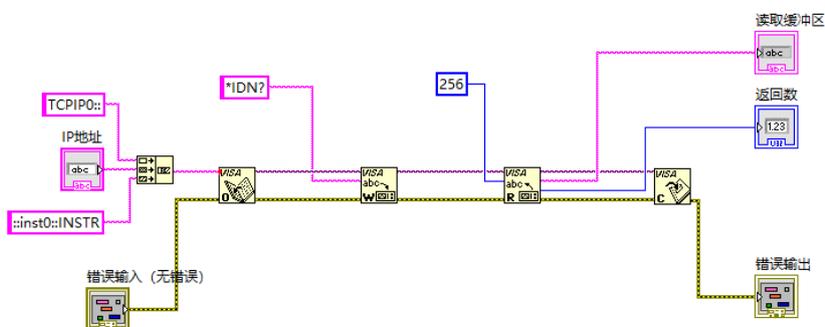
```

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 I/O。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-TCP/IP 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

设置带宽限制

当在观察低频信号时需要减少信号中的高频噪声，则衰减信号中 20MHz 以上的高频信号。可使用如下命令设置带宽限制，如设置通道 1：

```
CHANnel1:BWLimit ON      #打开通道 1 的带宽限制。
CHANnel1:BWLimit?       #查询返回 1，表示已经打开通道 1 的带宽限制。
```

设置伏格档位

设置通道伏格档位，可直接通过如下命令进行设置，如设置通道 1 的伏格档位。

```
CHANnel1:SCALE 500mV    #设置通道 1 的伏格档位为：500mV。
CHANnel1:SCALE?        #查询通道 1 的伏格档位值。
```

设置时基档位

设置示波器的时基档位，可直接通过如下命令进行设置。

```
TIMebase:SCALE 0.005    #设置示波器的时基档位为：5ms。
TIMebase:SCALE?        #查询示波器的时基档位。
```

查询幅度值

当需要查询幅度测量结果时，可直接通过如下命令进行查询，而不需要打开测量窗口。如查询通道 1 波形的幅度值。

```
MEASure:VPP? CHANnel1   #查询通道 1 波形的幅度值。
```

查询上升延迟时间值

当需要查询上升延迟测量时间时，可直接通过如下命令进行查询，而不需要打开测量窗口。如查询通道 1 与通道 2 的上升延迟值。

```
MEASure:PDElay? CHANnel1, CHANnel2 #查询通道 1 与通道 2 上升延迟时间值。
```

附录 1：按键列表

按键	功能描述	LED 灯
CH1	通道一开关	√
CH2	通道二开关	√
MATH	数学运算功能及其菜单	√
AUTO	自动设置示波器的各项控制值，以显示适宜观察的波形	
RS	控制示波器的运行状态，连续发送该命令，示波器将在停止和运行状态切换	√
TMENu	触发菜单	
DEFault	恢复默认设置	
HELP	帮助系统	
HMENu	水平系统菜单	
DISPlay	显示菜单	
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	
MENu	菜单显示功能开关	
PSCReen	一键打印或者一键保存屏幕图像	
MEASure	测量功能	
CURSor	光标测量功能及其菜单	
ACQuire	采样菜单	
STORage	存储菜单	
UTILity	系统辅助菜单	
FKNob	多功能旋钮	
FKNLeft	多功能旋钮左旋	

FKNR ight	多功能旋钮右旋	
VPKNob	垂直位置旋钮	
VPKNLeft	垂直位置旋钮左旋	
VPKNR ight	垂直位置旋钮右旋	
HPKNob	水平位置旋钮	
HPKNLeft	水平位置旋钮左旋	
HPKNR ight	水平位置旋钮右旋	
TPKNob	触发位置旋钮	
TPKNLeft	触发位置旋钮左旋	
TPKNR ight	触发位置旋钮右旋	
VBKNob	电压基准旋钮	
VBKNLeft	电压基准旋钮左旋	
VBKNR ight	电压基准旋钮右旋	
TBKNob	时间基准旋钮	
TBKNLeft	时间基准旋钮左旋	
TBKNR ight	时间基准旋钮右旋	
SETZERO	位置置零	

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：<#12345678 + 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

附录 3：光标测量 XY 模式列表

位序	名称
1	光标 Ax 值
2	光标 Bx 值
3	Bx-Ax 差值
4	通道 1 的 Ay 值
5	通道 1 的 By 值
6	通道 1 的 By-Ay 差值
7	通道 2 的 Ay 值
8	通道 2 的 By 值
9	通道 2 的 By-Ay 差值
10	两通道的 Ay 极坐标半径
11	两通道的 By 极坐标半径
12	两通道的 By-Ay 极坐标半径
13	两通道的 Ay 极坐标夹角
14	两通道的 By 极坐标夹角
15	两通道的 By-Ay 极坐标夹角
16	通道 2 的 Ay 与通道 1 的 Ay 之积
17	通道 2 的 By 与通道 1 的 By 之积
18	通道 2 的 By-Ay 与通道 1 的 By-Ay 之积
19	通道 2 的 Ay 与通道 1 的 Ay 之比
20	通道 2 的 By 与通道 1 的 By 之比
21	通道 2 的 By-Ay 与通道 1 的 By-Ay 之比