Remote Control Manual

Digital Oscilloscopes Series

RC0102X-E01A

SIGLENT TECHNOLOGIES CO., LTD

Catalogue

Programming Overview 3
  Build communication.. 3
    Install NI-VISA.. 3
    Connect the instrument 7
  How To Remote Control.. 8
    a. User-defined Programming.. 8
    b .Send SCPI Commands via NI-VISA 8
About these Commands & Queries 9
  How do they be listed?.. 9
  How do they be described? 9
  Where can they be used? 9
Command Notation.. 10
Table of Commands & Queries 11
Commands & Queries 17
Programming Examples 132
  Example of VC++ 133
  Example of VB.. 142
  Example of MATLAB.. 153
  Example of LabVIEW 156
Index 159

Programming Overview

This chapter introduces how to build communication between digital oscilloscope and the PC. It also introduces how to remote control.

Build communication

Install NI-VISA

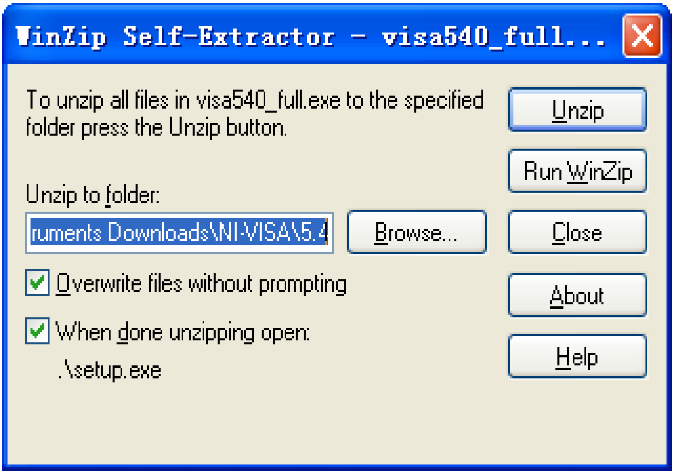
Before programming, you need to install NI-VISA, which you can download from the NI-VISA web site. About NI-VISA, there are full version and Run-Time Engine version. The full version include NI device driver and a tool named NI MAX that is a user interface to control the device. The Run-Time Engine version which is much smaller than the full version only include NI device driver.

For example, you can get NI-VISA 5.4 full version from: http://www.ni.com/download/ni-visa-5.4/4230/en/.

You also can download NI-VISA Run-Time Engine 5.4 to your PC and install it as default selection. Its installation process is similar with the full version.

After you downloaded the file you can follow the steps below to install it:

a.Double click the visa540_full.exe, dialog shown as below:



WinZip Self-Extractor - visa540_full...

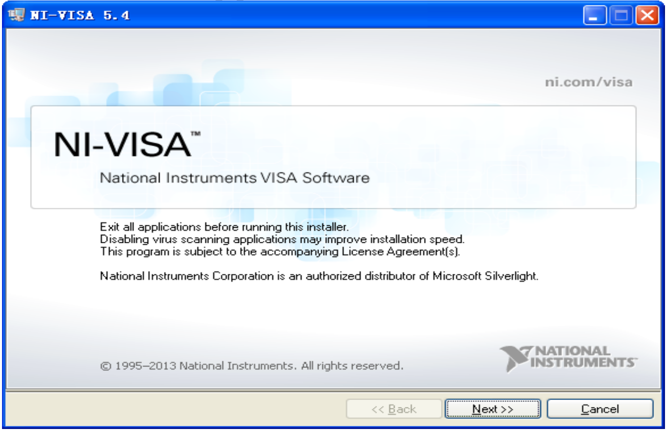
To unzip all files in visa540_full.exe to the specified folder press the Unzip button.

Unzip to folder: ruments Downloads\NI-VISA\5.4

☑ Overwrite files without prompting

☑ When done unzipping open:
.\setup.exe

b.Click Unzip, the installation process will automatically launch after unzipping files. If your computer needs to install .NET Framework 4, its Setup process will auto start.



NI-VISA 5.4

ni.com/visa

NI-VISA™

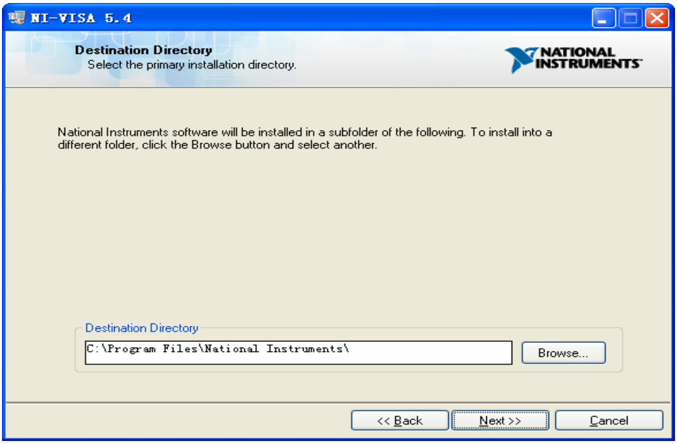
National Instruments VISA Software

Exit all applications before running this installer.
Disabling virus scanning applications may improve installation speed.
This program is subject to the accompanying License Agreement(s).

National Instruments Corporation is an authorized distributor of Microsoft Silverlight.

© 1995–2013 National Instruments. All rights reserved.

c.The NI-VISA installing dialog is shown above. Click Next to start the installation process.



Destination Directory

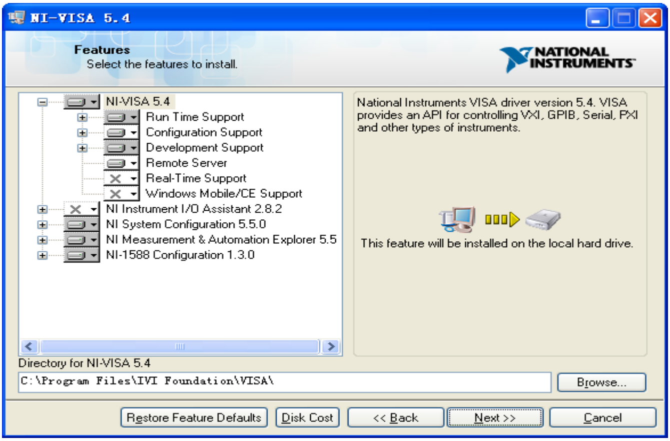
Select the primary installation directory.

National Instruments software will be installed in a subfolder of the following. To install into a different folder, click the Browse button and select another.

Destination Directory

C:\Program Files\National Instruments\

Set the install path, default path is “C:\Program Files\National Instruments\”, you can change it. Click Next, dialog shown as above.



Features

Select the features to install.

- NI-VISA 5.4
  - Run Time Support
  - Configuration Support
  - Development Support
  - Remote Server
  - ☒ Real-Time Support
  - ☒ Windows Mobile/CE Support
- ☒ NI Instrument I/O Assistant 2.8.2
- NI System Configuration 5.5.0
- NI Measurement & Automation Explorer 5.5
- NI-1588 Configuration 1.3.0

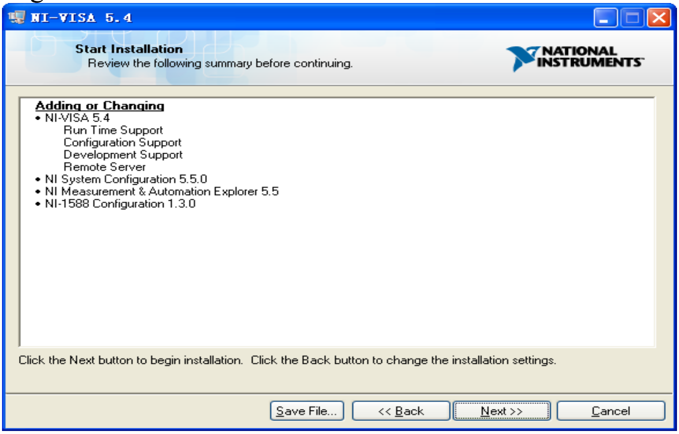
National Instruments VISA driver version 5.4. VISA provides an API for controlling VXI, GPIB, Serial, PXI and other types of instruments.

This feature will be installed on the local hard drive.

Directory for NI-VISA 5.4

C:\Program Files\IVI Foundation\VISA\

d.Click Next twice, in the License Agreement dialog, select the “I accept the above 2 License Agreement(s).”,and click Next, dialog shown as below:



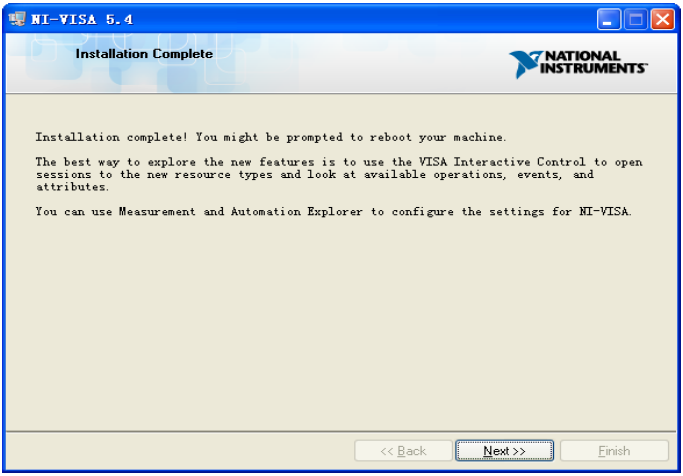
Start Installation

Review the following summary before continuing.

Adding or Changing
- NI-VISA 5.4
  - Run Time Support
  - Configuration Support
  - Development Support
  - Remote Server
- NI System Configuration 5.5.0
- NI Measurement & Automation Explorer 5.5
- NI-1588 Configuration 1.3.0

Click the Next button to begin installation. Click the Back button to change the installation settings.

e.Click Next to run installation.



Installation Complete

Installation complete! You might be prompted to reboot your machine.

The best way to explore the new features is to use the VISA Interactive Control to open sessions to the new resource types and look at available operations, events, and attributes.

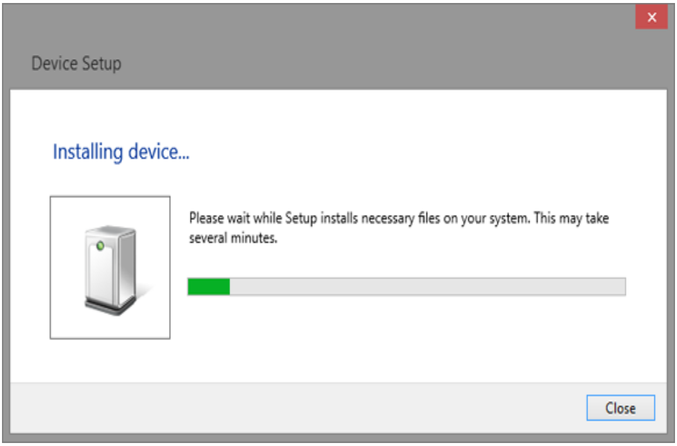
You can use Measurement and Automation Explorer to configure the settings for NI-VISA.

Now the installation is complete, reboot your PC.

Connect the instrument

Depending on your specific model your oscilloscope may be able to communicate with a PC through the USB or LAN interface. This manual takes the USB as an example. (For instructions to communicate with a PC through the LAN interface see the User Manual.)

a.Connect the USB Device interface at the rear panel of the oscilloscope and the USB Host interface of the PC using a USB cable. Assuming your PC is already turned on, turn on your oscilloscope and your PC will display the "Device Setup" screen as it automatically installs the device driver as shown below.



Installing device...

Please wait while Setup installs necessary files on your system. This may take several minutes.

b.Wait for the installation to complete and then proceed to the next step.

How To Remote Control

a. User-defined Programming

Users can use SCPI commands to program and control the digital oscilloscope. For details, refer to the introductions in "**Programming Examples**".

b .Send SCPI Commands via NI-VISA

You can control the oscilloscope remotely by sending SCPI commands via NI-VISA software.

About these Commands & Queries

This section lists describes the remote control commands and queries recognized by the instrument. All commands and queries can be executed in either local or remote state.

The description for each command or query, with syntax and other information, begins on a new page. The name (header) is given in both long and short form at the top of the page, and the subject is indicated as a command or query or both. Queries perform actions such as obtaining information, and are recognized by the question mark (?) following the header.

How do they be listed?

The descriptions are listed in alphabetical order according to their long form. Thus the description of ATTENUATION, whose short form is ATTN, is listed before that of AUTO SETUP, whose short form is ASET.

How do they be described?

In the descriptions themselves, a brief explanation of the function performed is given. This is followed by a presentation of the formal syntax, with the header given in Upper-and-Lower-Case characters and the short form derived from it in ALL UPPER-CASE characters. Where applicable, the syntax of the query is given with the format of its response.

Where can they be used?

The commands and queries listed here can be used for Digital Oscilloscopes Series digital instruments.

Command Notation

The following notation is used in the commands:

< > Angular brackets enclose words that are used as placeholders, of
which there are two types: the header path and the data parameter
of a command.

: = A colon followed by an equals sign separates a placeholder from
the description of the type and range of values that may be used in
a command instead of the placeholder.

{} Braces enclose a list of choices, one of which one must be made.

[ ] Square brackets enclose optional items.

… An ellipsis indicates that the items both to its left and right may be
repeated a number of times.

As an example, consider the syntax notation for the command to set the vertical input sensitivity:

```
<channel>:VOLT_DIV <v_gain>
<channel> : = {C1, C2, C3, C4}
<v_gain>: = 2 mV to 10 V
```

The first line shows the formal appearance of the command, with <channel> denoting the placeholder for the header path and <v_gain> the placeholder for the data parameter specifying the desired vertical gain value. The second line indicates that one of four channels must be chosen for the header path. And the third explains that the actual vertical gain can be set to any value between 2 mV and 10 V.

Table of Commands & Queries

Short Form	***Long Form***	***Subsystem***	***What the Command or Query Does***
ACQW	ACQUIRE_WAY	ACQUISITION	Specifies the acquisition mode.
ALST?	ALL_STATUS?	STATUS	Reads and clears the contents of all status registers.
ARM	ARM_ACQUISITION	ACQUISITION	Changes acquisition state from “stopped” to “single”.
ATTN	ATTENUATION	ACQUISITION	Selects the vertical attenuation factor of the probe
ACAL	AUTO_CALIBRATE	MISCELLANEOUS	Enables or disables automatic calibration.
ASET	AUTO_SETUP	ACQUISITION	Adjusts vertical, time base and trigger parameters.
AUTTS	AUTO_TYPESET	ACQUISITION	Selects the display type of automatic setup.
AVGA	AVERAGE_ACQUIRE	ACQUISITION	Selects the average times of average acquisition.
BWL	BANDWIDTH_LIMIT	ACQUISITION	Enables/disables the bandwidth-limiting low-pass filter.
BUZZ	BUZZER	MISCELLANEOUS	Controls the built-in piezo-electric buzzer.
*CAL?	*CAL?	MISCELLANEOUS	Performs complete internal calibration of the instrument.
CHDR	COMM_HEADER	COMMUNICATION	Controls formatting of query responses.
*CLS	*CLS	STATUS	Clears all status data registers.
CMR?	CMR?	STATUS	Reads and clears the Command error Register (CMR).
CONET	COMM_NET	COMMUNICATION	Specifies network addresses of scope and printers.
CPL	COUPLING	ACQUISITION	Selects the specified input channel’s coupling mode.

CRMS	CURSOR_MEASURE	CURSOR	Specifies the type of cursor/parameter measurement.
CRST?	CURSOR_SET?	CURSOR	Allows positioning of any one of eight cursors.
CRVA?	CURSOR_VALUE?	CURSOR	Returns trace values measured by specified cursors.
CSVS	CSV_SAVE	SAVE/RECALL	Saves specified waveform data of CSV format to USB device.
CYMT	CYMOMETER	FUNCTION	Returns the current cymometer value which displaying on the screen.
DATE	DATE	MISCELLANEOUS	Changes the date/time of the internal real-time clock.
DDR?	DDR?	STATUS	Clears the Device Dependent Register (DDR).
DEF	DEFINE?	FUNCTION	Specifies math expression for function evaluation.
DELF	DELETE_FILE	MASS STORAGE	Deletes files from mass storage.
DIR	DIRECTORY	MASS STORAGE	Creates and deletes file directories.
DTJN	DOT_JOIN	DISPLAY	Controls the interpolation lines between data points.
*ESE	*ESE	STATUS	Sets the Standard Event Status Enable register (ESE).
*ESR?	*ESR?	STATUS	Reads, clears the Event Status Register (ESR).
EXR?	EXR?	STATUS	Reads, clears the Execution error Register (EXR).
FLNM	FILENAME	MASS STORAGE	Changes default filenames.
FRTR	FORCE_TRIGGER	ACQUISITION	Forces the instrument to make one acquisition.
FVDISK	FORMAT_VDISK	MASS STORAGE	Reads the capability of the USB device.
FILT	FILTER	FUNCTION	Enables or disables the filter of specified source.
FILTS	FILT_SET	FUNCTION	Selects the type of filter, and sets the limit value of

			filter.
FFTW	FFT_WINDOW	FUNCTION	Selects the window of FFT.
FFTZ	FFT_ZOOM	FUNCTION	Selects the zoom in/out times of FFT trace.
FFTS	FFT_SCALE	FUNCTION	Selects the vertical scale of FFT trace.
FFTF	FFT_FULLSCREEN	FUNCTION	Enables or disables to display the FFT trace full screen.
GRDS	GRID_DISPLAY	DISPLAY	Selects the type of grid
GCSV	GET_CSV	WAVEFORMTRANS	Specifies waveform data of format to controller.
HMAG	HOR_MAGNIFY	DISPLAY	Horizontally expands the selected expansion trace.
HPOS	HOR_POSITION	DISPLAY	Horizontally positions intensified zone's center.
HCSU	HARDCOPY_SETUP	HARD COPY	Configures the hard-copy driver.
*IDN?	*IDN?	MISCELLANEOUS	For identification purposes.
INTS	INTENSITY	DISPLAY	Sets the grid or trace/text intensity level.
INR?	INR?	STATUS	Reads, clears INternal state change Register (INR).
INVS	INVERT_SET	DISPLAY	Invert the trace or the math waveform of specified source.
LOCK	LOCK	MISCELLANEOUS	Lock keyboard
MTVP	MATH_VERT_POS	ACQUISITION	Controls the vertical position of math waveform of specified source.
MTVD	MATH_VERT_DIV	ACQUISITION	Controls the vertical sensitivity of math waveform of specified source.
MSIZ	MEMORY_SIZE	FUNCTION	Returns the maximal memory size
OFST	OFFSET	ACQUISITION	Allows output channel vertical offset adjustment.
*OPC	*OPC	STATUS	Sets the OPC bit in the Event Status Register (ESR).
*OPT?	*OPT?	MISCELLANEOUS	Identifies oscilloscope

			options.
PACL	PARAMETER_CLR	CURSOR	Clears all current parameters in Custom, Pass/Fail.
PACU	PARAMETER_CUSTOM	CURSOR	Controls parameters with customizable qualifiers.
PAVA?	PARAMETER_VALUE?	CURSOR	Returns current parameter, mask test values.
PDET	PEAK_DETECT	ACQUISITION	Switches the peak detector ON and OFF.
PERS	PERSIST	DISPLAY	Enables or disables the persistence display mode.
PESU	PERSIST_SETUP	DISPLAY	Selects display persistence duration.
PNSU	PANEL_SETUP	SAVE/RECALL	Complements the *SAV/*RST commands.
PFDS	PF_DISPLAY	FUNCTION	Enables or disables to display the test and the message options of pass/fail.
PFST	PF_SET	FUNCTION	Sets the X mask and the Y mask.
PFSL	PF_SAVELOAD	SAVE/RECALL	Saves or recalls the created mask setting.
PFCT	PF_CONTROL	FUNCTION	Selects the “operate”, “output” and the “stop on output” which are the options of pass/fail.
PFCM	PF_CREATEM	FUNCTION	Creates the mask of the pass/fail.
PFDD	PF_DATEDIS	FUNCTION	Return the number of the pass/fail monitor which can be displayed on the screen.
*RCL	*RCL	SAVE/RECALL	Recalls one of five non-volatile panel setups.
RCPN	RECALL_PANEL	SAVE/RECALL	Recalls a front-panel setup from mass storage.
*RST	*RST	SAVE/RECALL	The *RST command initiates a device reset.
REFS	REF_SET	FUNCTION	Sets the reference waveform and its options.
*SAV	*SAV	SAVE/RECALL	Stores current state in non-volatile internal memory.
SCDP	SCREEN_DUMP	HARD COPY	Causes a screen dump to controller.

SCSV	SCREEN_SAVE	DISPLAY	Controls the automatic screen saver.
*SRE	*SRE	STATUS	Sets the Service Request Enable register (SRE).
*STB?	*STB?	STATUS	Reads the contents of IEEE 488.
STOP	STOP	ACQUISITION	Immediately stops signal acquisition.
STO	STORE	WAVEFORMTRANS	Stores a trace in internal memory or mass storage.
STPN	STORE_PANEL	SAVE/RECALL	Stores front-panel setup to mass storage.
STST	STORE_SETUP	WAVEFORMTRANS	Controls the way in which traces are stored.
SAST	SAMPLE_STATUS	ACQUISITION	Return the acquisition status of the scope
SARA	SAMPLE_RATE	ACQUISITION	Return the sample rate of the scope
SANU	SAMPLE_NUM	ACQUISITION	Return the number of sampled points available from last acquisition and the trigger position
SKEW	SKEW	ACQUISITION	Sets the skew of specified trace.
SXSA	SINXX_SAMPLE	ACQUISITION	Sets the type of the interpolation.
TDIV	TIME_DIV	ACQUISITION	Modifies the time base setting.
TMPL	TEMPLATE	WAVEFORM TRANSFER	Produces a complete waveform template copy.
TRA	TRACE	DISPLAY	Enables or disables the display of a trace.
*TRG	*TRG	ACQUISITION	Executes an ARM command.
TRCP	TRIG_COUPLING	ACQUISITION	Sets the coupling mode of the specified trigger source.
TRDL	TRIG_DELAY	ACQUISITION	Sets the time at which the trigger is to occur.
TRLV	TRIG_LEVEL	ACQUISITION	Adjusts the trigger level of the specified trigger source.
TRLV2	TRIG_LEVEL2	ACQUISITION	Adjusts the second trigger level of the specified trigger source.
TRMD	TRIG_MODE	ACQUISITION	the trigger mode.

TRSE	TRIG_SELECT	ACQUISITION	Selects the condition that will trigger acquisition.
TRSL	TRIG_SLOPE	ACQUISITION	Sets the trigger slope of the specified trigger source.
TRWI	TRIG_WINDOW	ACQUISITION	Return relative height of the trigger window
TRPA	TRIG_PATTERN	ACQUISITION	Sets the condition of the pattern trigger
UNIT	UNIT	ACQUISITION	Sets the unit of specified trace.
VPOS	VERT_POSITION	DISPLAY	Adjusts the vertical position of the FFT trace.
VDIV	VOLT_DIV	ACQUISITION	Sets the vertical sensitivity.
WF	WAVEFORM	WAVEFORMTRANS	Gets the waveform from the instrument.
WFSU	WAVEFORM_SETUP	WAVEFORMTRANS	Specifies amount of waveform data to go to controller.
WAIT	WAIT	ACQUISITION	Prevents new analysis until current has been completed.
XYDS	XY_DISPLAY	DISPLAY	Enables or disables to display the XY format

Commands & Queries

ACQUISITION

ACQUIRE_WAY,ACQW

Command /Query

DESCRIPTION

The ACQUIRE_WAY command specifies the acquisition mode.

The ACQUIRE_ WAY? Query returns the current acquisition mode.

COMMAND SYNTAX

```
ACQUIRE_WAY <mode>[,<time>]
 <mode> :=
 { SAMPLING,PEAK_DETECT,AVERAGE,
   HIGH_RES }
  <time> := {4, 16, 32, 64,128,256,512,1024}
```

Note : The <time> parameter only can be set with the average acuisition mode.

QUERY SYNTAX

```
ACQUIRE_WAY?
```

RESPONSE FORMAT

```
ACQUIRE_WAY  <mode>[,<time>]
```

EXAMPLE

The following command sets the acquisition mode to average mode,and also sets the average time to 16.

Command message:

```
ACQW AVERAGE,16
```

RELATED COMMANDS

AVGA,PDET

STATUS **ALL_STATUS?, ALST?**

Query

DESCRIPTION

The ALL_STATUS? Query reads and clears the contents of all status registers: STB, ESR, INR, DDR, CMR, EXR and URR except for the MAV bit (bit 6) of the STB register. For an interpretation of the contents of each register, refer to the appropriate status register.

The ALL_STATUS? Query is useful in a complete overview of the state of the instrument.

QUERY SYNTAX

ALl_STatus?

RESPONSE FORMAT

ALl_STatus
STB,<value>,ESR,<value>,INR,<value>,DDR,<value>,CMR,<value>,EXR,<value>,URR,<value>

<value> : = 0 to 65535

EXAMPLE

The following instruction reads the contents of all the status registers:

Command message:
ALST?

Response message:
ALST STB, 0, ESR, 52, INR, 5, DDR, 0, CMR, 4, EXR, 24, URR, 0

RELATED COMMANDS

*CLS, CMR? , DDR? ,*ESR? , EXR? , *STB? , URR?

ACQUISITION

ARM_ACQUISITION, ARM

Command

DESCRIPTION

The ARM_ACQUISITION command enables the signal acquisition process by changing the acquisition state (trigger mode) from “stopped” to “single”.

COMMAND SYNTAX

ARM acquisition

EXAMPLE

The following command enables signal acquisition:

Command message:

```
ARM
```

RELATED COMMANDS

STOP, *TRG, TRIG_MODE, WAIT

ACQUISITION

ATTENUATION, ATTN

Command /Query

DESCRIPTION

The ATTENUATION command selects the vertical attenuation factor of the probe. Values of 1, 5, 10, 50, 100, 500, and 1000 may be specified.

The ATTENUATION? Query returns the attenuation factor of the specified channel.

COMMAND SYNTAX

<channel>: ATTeNuation <attenuation>
<channel> : = {C1, C2, C3, C4}
<attenuation>：= {1, 5, 10, 50, 100, 500, 1000}

QUERY SYNTAX

<channel>: ATTeNuation?

RESPONSE FORMAT

<channel>: ATTeNuation <attenuation>

EXAMPLE

The following command sets to 100 the attenuation factor of Channel 1:

Command message:
C1:ATTN 100

MISCELLANEOUS

AUTO_CALIBRATE, ACAL

Command /Query

DESCRIPTION

The AUTO_CALIBRATE command is used to enable or disable the quick calibration of the instrument.

The quick calibration may be disabled by issuing the command ACAL OFF. Whenever it is convenient, a *CAL? Query may be issued to fully calibrate the oscilloscope.

The response to the AUTO_CALIBRATE?
Query indicates whether quick -calibration is enabled.

The command is only used in the CFL series instrument.

COMMAND SYNTAX

Auto_CALibrate <state>
<state> : = {ON, OFF}

QUERY SYNTAX

Auto_CALibrate?

RESPONSE FORMAT

Auto_CALibrate <state>

EXAMPLE

The following instruction disables quick-calibration:

Command message:
ACAL OFF

RELATED COMMANDS

*CAL?

ACQUISITION

AUTO_SETUP, ASET

Command

DESCRIPTION

The AUTO_SETUP command attempts to identify the waveform type and automatically adjusts controls to produce a usable display of the input signal.

COMMAND SYNTAX

```
AUTO_SETUP
```

EXAMPLE

The following command instructs the oscilloscope to perform an auto-setup:

Command message:
ASET

RELATED COMMANDS

AUTTS

ACQUISITION

AUTO_TYPESET, AUTTS

Command /Query

DESCRIPTION

The AUTO_TYPESET command selects the specified type of automatically adjusting which is used to display.

COMMAND SYNTAX

AUTO_TYPESET <type>

<type> : = {SP,MP,RS,DRP,RC}
SP means only one period to be displayed, MP means multiple periods to be displayed, RS means the waveform is triggered on the rise side, DRP means the waveform is triggered on the drop side, and RC means to go back to the state before auto set.

QUERY SYNTAX

AUTO_TYPESET?

RESPONSE FORMAT

AUTO_TYPESET <type>

EXAMPLE

The following command sets the type of automatic adjustment to multiple periods:

Command message:
AUTTS MP

RELATED COMMANDS

ASET

ACQUISITION AVERAGE_ACQUIRE, AVGA

Command /Query

DESCRIPTION

The AVERAGE_ACQUIRE command selects the average times of average acquisition.

The response to the AVERAGE_ACQUIRE query indicates the times of average acquisition.

COMMAND SYNTAX

AVERAGE_ACQUIRE <time>

<time> : = {4, 16, 32, 64,128,256,512,1024}

QUERY SYNTAX

AVERAGE_ACQUIRE?

RESPONSE FORMAT

AVERAGE_ACQUIRE <time>

EXAMPLE

The following turns the average times of average acquisition 16:

Command message:
AVGA 16

ACQUISITION

BANDWIDTH_LIMIT, BWL

Command /Query

DESCRIPTION

BANDWIDTH_LIMIT enables or disables the bandwidth-limiting low-pass filter. If the bandwidth filters are on, it will limit the bandwidth to reduce display noise. When you turn Bandwidth Limit ON, the Bandwidth Limit value is set to 20 MHz. It also filters the signal to reduce noise and other unwanted high frequency components.

The response to the BANDWIDTH_LIMIT? Query indicates whether the bandwidth filters are on or off.

COMMAND SYNTAX

BandWidth_Limit <channel>, <mode>
[, <channel>, <mode> [, <channel>, <mode>
[, <channel>, <mode>]]]

<channel> : = {C1, C2, C3, C4}
<mode>： = {ON, OFF}

QUERY SYNTAX

BandWidth_Limit?

RESPONSE FORMAT

BandWidth_Limit <channel>, <mode> [, <channel>, <mode> [, <channel>, <mode> [, <channel>, <mode>]]]

EXAMPLE

The following turns on the bandwidth filter for all channels, when Global_BWL is on (as it is by default

The following turns the bandwidth filter on for Channel 1only:

Command message:
BWL C1, ON

MISCELLANEOUS

BUZZER, BUZZ

Command /Query

DESCRIPTION

The BUZZER command enables or disables sound switch.

The response to the BUZZER? query indicates whether the sound switch is enabled.

COMMAND SYNTAX

BUZZer <state>
<state>：= {ON, OFF}

QUERY SYNTAX

BUZZER?

RESPONSE FORMAT

BUZZER <state>

EXAMPLE

Sending the following code will let the oscilloscope turn on the sound switch.

Command message:
BUZZ ON

MISCELLANEOUS

*CAL?

Query

DESCRIPTION

The *CAL? query cause the oscilloscope to perform an internal self-calibration and generates a response.

QUERY SYNTAX

*CAL?

RESPONSE FORMAT

*CAL <diagnostics>
<diagnostics> : = 0
0 = Calibration successful

EXAMPLE

The following instruction forces a self-calibration:

Command message:
*CAL?

Response message:
*CAL 0

RELATED COMMANDS

AUTO_CALIBRATE

COMMUNICATION COMM_HEADER, CHDR

Command/ Query

DESCRIPTION

The COMM_HEADER command controls the way the oscilloscope formats responses to queries. There are three response formats: LONG, in which responses start with the long form of the header word; SHORT, where responses start with the short form of the header word; and OFF, for which headers are omitted from the response and units in numbers are suppressed.

Unless you request otherwise, the SHORT response format is used.

This command does not affect the interpretation of messages sent to the oscilloscope. Headers can be sent in their long or short form regardless of the COMM_HEADER setting.

Querying the vertical sensitivity of Channel 1 may result in one of the following responses:

COMM_HEADER	RESPONSE
LONG	C1:VOLT_DIV 200E-3 V
SHORT	C1:VDIV 200E-3 V
OFF	200E-3

COMMAND SYNTAX

Comm_HeaDeR <mode>
<mode> : = {SHORT, LONG, OFF}

QUERY SYNTAX

Comm_HeaDeR?

RESPONSE FORMAT

Comm_HeaDeR <mode>

EXAMPLE

The following code sets the response header format to SHORT:

Command message:
CHDR SHORT

STATUS

*CLS
Command

DESCRIPTION

The *CLS command clears all the status data registers.

COMMAND SYNTAX

*CLS

EXAMPLE

The following command causes all the status data registers to be cleared:

Command message:
*CLS

RELATED COMMANDS

ALL_STATUS, CMR, DDR, *ESR, EXR, *STB, URR

STATUS **CMR?**

Query

DESCRIPTION

The CMR? Query reads and clears the contents of the Command error Register (CMR) --- see table next page---which specifies the last syntax error type detected by the instrument.

QUERY SYNTAX

CMR?

RESPONSE FORMAT

CMR <value>
<value> : = 0 to 14

EXAMPLE

The following instruction reads the contents of the CMR register:

Command message:
CMR?

Response message:
CMR 0

RELATED COMMANDS

ALL_STATUS? ,*CLS

ADDITIONAL INFORMATION

Command Error Status Register Structure (CMR)

Command Error Status Register Structure (CMR)	
Value	Description
1	Unrecognized command/query header
2	Invalid character
3	Invalid separator
4	Missing parameter
5	Unrecognized keyword
6	String error
7	Parameter cannot allowed
8	Command String Too Long
9	Query cannot allowed
10	Missing Query mask
11	Invalid parameter
12	Parameter syntax error
13	Filename too long

MISCELLANEOUS

COMM_NET, CONET

Command /Query

DESCRIPTION

The COMM_NET command changes the IP address of the oscilloscope's internal network interface.

The COMM_NET? query returns the IP address of the oscilloscope's internal network interface.

COMMAND SYNTAX

COMM_NET <ip_add0>, <ip_add1>, <ip_add2>, <ip_add3>

< ip_add >:= 0 to 255

QUERY SYNTAX

COMM_NET?

RESPONSE FORMAT

COMM_NET <ip_add0>, <ip_add1>, <ip_add2>, <ip_add3>

EXAMPLE

This instruction will change the IP address to 10.11.0.230:

Command message:
CONET 10,11,0,230

ACQUISITION

COUPLING, CPL

Command /Query

DESCRIPTION

The COUPLING command selects the coupling mode of the specified input channel.

The COUPLING? query returns the coupling mode of the specified channel.

COMMAND SYNTAX

<channel>: CouPLing <coupling>
<channel> : = {C1, C2, C3, C4}
<coupling> : = {A1M, A50, D1M, D50, GND}
The A of the <coupling> is alternating current. The D of the <coupling> is direct current.1M and 50 is the impedance of input. Some series (CML) couldn't have the set of input impedance.

QUERY SYNTAX

<channel>: CouPLing?

RESPONSE FORMAT

<channel>: CouPLing <coupling>

EXAMPLE

The following command sets the coupling of Channel 2 to 50 ΩDC:

Command message:
C2: CPL D50

CURSOR

CURSOR_MEASURE, CRMS

Command /Query

DESCRIPTION

The CURSOR_MEASURE command specifies the type of cursor or parameter measurement to be displayed

The CURSOR_MEASURE? query indicates which cursors or parameter measurements are currently displayed.

COMMAND SYNTAX

CuRsor_MeaSure <mode>
<mode>={ OFF,ON}

QUERY SYNTAX

CuRsor_MeaSure?

RESPONSE FORMAT

CuRsor_MeaSure <mode>

EXAMPLE

The following command determines cursor function is turned off:

Command message:
CRMS OFF

RELATED COMMANDS

CURSOR_VALUE, PARAMETER_VALUE

CURSOR CURSOR_SET, CRST

Command /Query

DESCRIPTION

The CURSOR_SET command allows the user to position any one of the eight independent cursors at a given screen location. The positions of the cursors can be modified or queried even if the required cursor is not currently displayed on the screen. When setting a cursor position, a trace must be specified, relative to which the cursor will be positioned.

The CURSOR_SET? Query indicates the current position of the cursor(s). The values returned depend on the grid type selected.

Notation	
VREF	The volt-value of curA under manual cursor mode
VDIF	The volt -value of curB under manual cursor mode
TREF	The time value of curA under manual cursor mode
TDIF	The time value of curB under manual cursor mode

COMMANDSYNTAX

<trace>:CuRsor_SeT<cursor>,<position>[,<cursor>,<position>,<cursor> ,<position>]

< trace > : = {C1, C2, C3, C4}
<cursor> : ={ VREF,VDIF,TREF,TDIF}
<position>： = 0.1 to 13.9 DIV (horizontal of track, the range of the value is related to the size of the screen)
<position>： = -4 to 4 DIV (vertical)
<position>： = -6(or -9) to 6 DIV (horizontal of manual, the range of the value is related to the size of the screen)

QUERY SYNTAX

<trace>: CuRsor_SeT? [<cursor>, …<cursor>]
<cursor> :={ VREF, VDIF, TREF, TDIF}

RESPONSE FORMAT

<trace>:CuRsor_SeT <cursor>, <position> [, <cursor>, <position>, <cursor>, <position>]

EXAMPLE

The following command positions the VREF and VDIF cursors at +3 DIV and −1 DIV respectively, using C1 as a reference:

Command message:
C1: CRST VREF, 3DIV, VDIF, −1DIV

RELATED COMMANDS

CURSOR_MEASURE, CURSOR_VALUE, PARAMETER_VALUE

CURSOR

CURSOR_VALUE?, CRVA?

Query

DESCRIPTION

The CURSOR_VALUE? Query returns the values measured by the specified cursors for a given trace. (The PARAMETER_VALUE? query is used to obtain measured waveform parameter values.)

Notation	
HREL	the cursor value under track cursor mode
VREL	the dalta volt-value under manual cursor mode

QUERY SYNTAX

<trace>: CuRsor_Value? [<mode>,…<mode>]
<trace> : = { C1, C2, C3, C4}
<mode> : = { HREL, VREL }

RESPONSE FORMAT

<trace> : CuRsor_Value HREL, <delta_hori>,<delta_vert>,<A->T>, <A->V>,<(delta_vert)/(delta_hori)>
<trace> : CuRsor_Value VREL,<delta_vert>

EXAMPLE

The following query reads the dalta volt value under manual cursor mode (VREL) on Channel 2:

Command message:
C2:CRVA? VREL

Response message:
C2:CuRsor_Value VREL 1.00V

RELATED COMMANDS

CURSOR_SET, PARAMETER_VALUE

SAVE/RECALL CSV_SAVE, CSVS

Command /Query

DESCRIPTION

The CSV_SAVE command selects the specified option of storing CSV format waveform.

The CSV_SAVE? query returns the option of storing waveform data of CSV format.

COMMAND SYNTAX

CSV_SAVE SAVE,<state>

The option SAVE is that if the waveform data is stored with parameter.
<save>： = {OFF, ON}

QUERY SYNTAX

CSV_SAVE?

RESPONSE FORMAT

CSV_SAVE SAVE, <state>

EXAMPLE

The following command sets "para" save to off

Command message:
CSV_SAVE SAVE,OFF

FUNCTION CYMOMETER, CYMT

Query

DESCRIPTION

The response to the CYMOMETER? query is the value of cymometer which displaying on the screen of the instrument. When the signal frequency is less than 10Hz, it returns 10Hz.

QUERY SYNTAX

CYMOMETER?

RESPONSE FORMAT

CYMOMETER <option>

EXAMPLE

The following instruction returns the value of cymometer which displaying on the screen of the instrument.

Response message:
CYMT 10Hz

MISCELLANEOUS

DATE

Command /Query

DESCRIPTION

The DATE command changes the date/time of the oscilloscope's internal real-time clock.

The command is only used in the CFL series instrument.

COMMAND SYNTAX

DATE <day>, <month>, <year>, <hour>, <minute>, <second>

<day> : = 1 to 31
<month> : = {JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP,OCT, NOV, DEC}
<year> : = 1990 to 2089
<hour> : = 0 to 23
<minute> : = 0 to 59
<second> : = 0 to 59

QUERY SYNTAX

DATE?

RESPONSE FORMAT

DATE <day>, <month>, <year>, <hour>, <minute>, <second>

EXAMPLE

This instruction will change the date to NOV. 1, 2009 and the time to 14:38:16:

Command message:
DATE 1, NOV, 2009,14,38,16

STATUS

DDR?

Query

DESCRIPTION

The DDR? Query reads and clears the contents of the Device Dependent or device specific error Register (DDR). In the case of a hardware failure, the DDR register specifies the origin of the failure.

QUERY SYNTAX

DDR?

RESPONSE FORMAT

DDR <value>
<value> : = 0 to 65535

EXAMPLE

The following instruction reads the contents of the DDR register:

Command message:
DDR?

Response message:
DDR 0

RELATED COMMANDS

ALL_STATUS? ,*CLS

FUNCTION

DEFINE, DEF

Command /Query

DESCRIPTION

The DEFINE command specifies the mathematical expression to be evaluated by a function.

COMMAND SYNTAX

DEFine EQN,'<equation>'
<equation> the mathematical expression

Function Equations	
<source1> + <source2>	Addition
<source1> - <source2>	Subtraction
<source1>*<source2>	Multiplication
<source1>/<source2>	Ratio
FFT(source x)	FFT
INTG(source x)	Integral
DIFF(source x)	Differentiator
SQRT(source x)	Square Root

QUERY SYNTAX

DEFine?

RESPONSE FORMAT

DEFine EQN,'<equation>'

EXAMPLE

Command message:
DEFine EQN,'C1*C2'

MASS STORAGE DELETE_FILE, DELF

Command

DESCRIPTION

The DELETE_FILE command deletes files from the currently selected directory on mass storage.

COMMAND SYNTAX

DELete_File DISK, <device>, FILE, '<filename>'

<device>：={UDSK}

<filename>：= a file of specified directory and the specified file should up to eight characters.

EXAMPLE

The following command deletes a front-panel setup from the directory named SETUP in a USB memory device:

Command message:

DELF DISK, UDSK, FILE, '/ SETUP /001.SET'

RELATED COMMANDS DIRECTORY

MASS STORAGE

DIRECTORY, DIR

Command /Query

DESCRIPTION

The DIRECTORY command is used to manage the creation and deletion of file directories on mass storage devices. It also allows selection of the current working directory and listing of files in the directory.

The query response consists of a double-quoted string containing a DOS-like listing of the directory.

COMMAND SYNTAX

Directory DISK, <device>, ACTION, <action>, ‘<directory>’

QUERY SYNTAX

Directory? DISK, <device> [, ‘<directory>’]

<device>：={UDSK}

<action>：={CREATE, DELETE}

< directory >：= A legal DOS path or filename. (This can include the ‘/’ character to define the root directory.)

RESPONSE FORMAT

DIRectory DISK, <device> “<directory>”

EXAMPLE

The following asks for a listing of the directory of a USB memory device:

Command message:

```
DIR? DISK, UDSK
```

Response message:

```
DIRectory DISK, UDSK,"A:
SDS1000X
SDS1000AA
BB.SET                  2.00 KB
SDS00001.SET            2.00 KB
SDS00002.SET            2.00 KB

3 File(s), 2 DIR(s)
"
```

RELATED COMMANDS

DELF

DISPLAY

DOT_JOIN, DTJN

Command /Query

DESCRIPTION

The DOT_JOIN command controls the interpolation lines between data points.

COMMAND SYNTAX

DoT_JoiN <state>
<state> : = {ON, OFF}

QUERY SYNTAX

DoT_JoiN?

RESPONSE FORMAT

DoT_JoiN <state>

EXAMPLE

The following instruction turns off the interpolation lines:

Command message:
DTJN OFF

STATUS

*ESE

Command /Query

DESCRIPTION

The *ESE command sets the Standard Event Status Enable register (ESE). This command allows one or more events in the ESR register to be reflected in the ESB summary message bit (bit 5) of the STB register.

COMMAND SYNTAX

*ESE <value>
<value> : = 0 to 255

QUERY SYNTAX

*ESE?

RESPONSE FORMAT

*ESE <value>

EXAMPLE

The following instruction allows the ESB bit to be set if a user request (URQ bit 6, i.e. decimal 64) and/or a device dependent error (DDE bit 3, i.e. decimal 8) occurs. Summing these values yields the ESE register mask 64+8=72.

Command message:
*ESE 72

RELATED COMMANDS

*ESR

STATUS

*ESR?

Query

DESCRIPTION

The *ESR? query reads and clears the contents of the Event Status Register (ESR). The response represents the sum of the binary values of the register bits 0 to 7.

QUERY SYNTAX

```
*ESR?
```

RESPONSE FORMAT

```
*ESR <value>
<value> : = 0 to 255
```

EXAMPLE

The following instruction reads and clears the contents of the ESR register:

Command message:

```
*ESR?
```

Response message:

```
*ESR 0
```

RELATED COMMANDS

ALL_STATUS, *CLS, *ESE

ADDITIONAL INFORMATION

Standard Event Status Register (ESR)

Bit	Bit Value	Bit Name	Description		Note
15~8			0	reserved by IEEE 488.2	
7	128	PON	1	Power off-to-ON transition as occurred	(1)
6	64	URQ	1	User Request has been issued	(2)
5	32	CME	1	Command parser Error has been detected	(3)
4	16	EXE	1	Execution Error detected	(4)
3	8	DDE	1	Device specific Error occurred	(5)
2	4	QYE	1	Query Error occurred	(6)
1	2	RQC	1	Instrument never requests bus control	(7)
0	1	OPC	1	Instrument never requests bus control	(8)

Notes

(1) The Power On (PON) bit is always turned on (1) when the unit is powered up.

(2) The User Request (URQ) bit is set true (1) when a soft key is pressed. An associated register URR identifies which key was selected. For further details refer to the URR? query.

(3) The CoMmand parser Error bit (CME) is set true (1) whenever a command syntax error is detected. The CME bit has an associated CoMmand parser Register (CMR) which specifies the error code. Refer to the query CMR? for further details.

(4) The EXecution Error bit (EXE) is set true (1) when a command cannot be executed due to some device condition (e.g. oscilloscope in local state) or a semantic error. The EXE bit has an associated Execution Error Register (EXR) which specifies the error code. Refer to query EXR? for further details.

(5) The Device specific Error (DDE) is set true (1) whenever a hardware failure has occurred at power-up, or execution time, such as a channel overload condition, a trigger or a timebase circuit defect. The origin of the failure may be localized via the DDR? or the self test *TST? query.

(6) The Query Error bit (QYE) is set true (1) whenever (a) an attempt is made to read data from the Output Queue when no output is either present or pending, (b) data in the Output Queue has been lost, (c) both output and input buffers are full (deadlock state), (d) an attempt is made by the controller to read before having sent an <END>, (e) a command is received before the response to the previous query was read (output buffer flushed).

(7) The ReQuest Control bit (RQC) is always false (0), as the oscilloscope has no GPIB controlling capability.

(8) The OPeration Complete bit (OPC) is set true (1) whenever *OPC has been received, since commands and queries are strictly executed in sequential order. The oscilloscope starts processing a command only when the previous command has been entirely executed.

STATUS

*EXR?

Query

DESCRIPTION

The EXR? query reads and clears the contents of the Execution error Register (EXR). The EXR register specifies the type of the last error detected during execution.

QUERY SYNTAX

EXR?

RESPONSE FORMAT

EXR <value>
<value> : = to

EXAMPLE

The following instruction reads the contents of the EXR register:

Command message:
EXR?

Response message (if no fault):
EXR 0

RELATED COMMANDS

ALL_STATUS, *CLS

ADDITIONAL INFORMATION

Execution Error Status Register Structure (EXR)	
Value	Description
21	Permission error. The command cannot be executed in local mode.
22	Environment error. The instrument is not configured to correctly process a command. For instance, the oscilloscope cannot be set to RIS at a slow timebase.
23	Option error. The command applies to an option which has not been installed.
25	Parameter error. Too many parameters specified.
26	Non-implemented command.
32	Waveform descriptor error. An invalid waveform descriptor has been detected.
36	Panel setup error. An invalid panel setup data block has been detected.
50	No mass storage present when user attempted to access it.
53	Mass storage was write protected when user attempted to create, or a file, to delete a file, or to format the device.
58	Mass storage file not found.
59	Requested directory not found.
61	Mass storage filename not DOS compatible, or illegal filename.
62	Cannot write on mass storage because filename already exists.

MASS STORAGE FILENAME, FLNM

Command /Query

DESCRIPTION

The FILENAME command is used to change the default filename given to any traces, setups and hard copies when they are being stored to a mass storage device.

COMMAND SYNTAX

FiLeNaMe TYPE, <type>, FILE, '<filename>'
<type>:={ C1,C2,C3, C4, SETUP,TA, TB, TC, TD, HCOPY}
<filename> : = an alphanumeric string of up to 8 characters forming a legal DOS filename.

Note: the file's extension can be specified automatically by the oscilloscope.

QUERY SYNTAX

FiLeNaMe? TYPE, <type>
<type> :={ ALL, C1, C2, C3, C4, SETUP, TA, TB, TC, TD, HCOPY}

RESPONSE FORMAT

FiLeNaMe TYPE, <type>, FILE, "<filename>"
[,TYPE, <type>, FILE, "<filename>"...]

EXAMPLE

The following command designates channel 1 waveform files to be "TESTWF.DAV":

Command message:
FLNM TYPE, C1, FILE, 'TESTWF'

RELATED COMMANDS

DIRECTORY, DELETE_FILE

MASS STORAGE

FORMAT_VDISK, FVDISK

Query

DESCRIPTION

The FORMAT_VDISK? query reads the capability of the USB memory device.

QUERY SYNTAX

Format_VDISK？

RESPONSE FORMAT

Format_VDISK <capability>
<capability>:= the capability of the USB memory device.

EXAMPLE

The following query reads the capability of the USB device.

Command message:
Format_VDISK?

Response message:
Format_VDISK 963 MB

FUNCTION FFT_WINDOW, FFTW

Command /Query

DESCRIPTION

The FFT_WINDOW command selects the window of FFT(Fast Fourier Transform algorithm).

The response to the FFT_WINDOW? query indicates current window of FFT

COMMAND SYNTAX

FFT_WINDOW <window>
< window > : = {RECT,BLAC,HANN,HAMM}
RECT is short for rectangle.
BLAC is short for Blackman.
HANN is short for hanning.
HAMM is short for hamming,

QUERY SYNTAX

FFT_WINDOW?

RESPONSE FORMAT

FFT_WINDOW,<window>

EXAMPLE

The following command sets the FFT window to hamming:

Command message:
FFTW HAMM

FUNCTION FFT_ZOOM, FFTZ

Command /Query

DESCRIPTION

The FFT_ZOOM command selects the specified zoom of FFT.

The response to the FFT_ZOOM? query indicates current zoom in/out times of FFT

COMMAND SYNTAX

FFT_ZOOM <zoom>
< zoom > : = {1,2,5,10}

QUERY SYNTAX

FFT_ZOOM？

RESPONSE FORMAT

FFT_ZOOM,<zoom>

EXAMPLE

The following command sets the zoom factor of FFT to 1X:

Command message:
FFTZ 1

FUNCTION

FFT_SCALE, FFTS

Command /Query

DESCRIPTION

The FFT_SCALE command selects the specified scale of FFT(Fast Fourier Transform algorithm).

The response to the FFT_SCALE? query indicates current vertical scale of FFT waveform.

COMMAND SYNTAX

FFT_SCALE <scale>
< scale > : = {VRMS,DBVRMS}

QUERY SYNTAX

FFT_SCALE?

RESPONSE FORMAT

FFT_SCALE,< scale >

EXAMPLE

The following command turns the vertical scale of FFT to dBVrms:

Command message:
FFTS DBVRMS

FUNCTION FFT_FULLSCREEN, FFTF

Command /Query

DESCRIPTION

The FFT_FULLSCREEN command enables or disables to display the FFT waveform full screen.

The response to the FFT_FULLSCREEN? query indicates whither the FFT waveform is full screen displayed.

COMMAND SYNTAX

FFT_FULLSCREEN <state>
< state > : = {ON,OFF}

QUERY SYNTAX

FFT_FULLSCREEN?

RESPONSE FORMAT

FFT_FULLSCREEN < state >

EXAMPLE

The following command enables to display the FFT waveform full screen:

Command message:
FFTF ON

DISPLAY

GRID_DISPLAY, GRDS

Command /Query

DESCRIPTION

The GRID_DISPLAY command selects the type of the grid which is used to display.

The response to the GRID_DISPLAY? query indicates current type of the grid

COMMAND SYNTAX

GRID_DISPLAY <type>
< type > : = {FULL,HALF,OFF}

QUERY SYNTAX

GRID_DISPLAY?

RESPONSE FORMAT

GRID_DISPLAY < type >

EXAMPLE

The following command changes the type of grid to full grid:

Command message:
GRID_DISPLAY FULL

WAVEFORMTRANS GET_CSV, GCSV

Query

DESCRIPTION

The response to the GET_CSV? query indicates current waveform of CSV format.

The GET_CSV? query have option to set. They are the same as the options of CSVS.

QUERY SYNTAX

GET_CSV？ SAVE,<state>

The option SAVE is that if the waveform data have parameters.
<save>：= {OFF,ON}

RESPONSE FORMAT

the waveform date of CSV format

EXAMPLE

The following command transfers the waveform data of CSV format to the controller. It has parameters information.

Command message:
GET_CSV？ SAVE,ON

HOR_MAGNIFY, HMAG

Command /Query

DESCRIPTION

The HOR_MAGNIFY command horizontally expands the selected expansion trace by a specified factor. Magnification factors not within the range of permissible values will be rounded off to the closest legal value.

If the specified factor is too large for any of the expanded traces (depending on their current source), it is reduced to an acceptable value and only then applied to the traces. The VAB bit (bit 2) in the STB register is set when a factor outside the legal range is specified.

The HOR_MAGNIFY query returns the current magnification factor for the specified expansion function.

COMMAND SYNTAX

<exp_trace>: Hor_MAGnify <factor>
<exp_trace>: = {TA, TB, TC, TD}
<factor> : = 1 to 2,000,000 The range of <factor> it is related to the current timebase and the range of the timebase

QUERY SYNTAX

<exp_trace> : Hor_MAGnify？

RESPONSE FORMAT

<exp_trace>: Hor_MAGnify <factor>

EXAMPLE

The following instruction horizontally magnifies Trace A (TA) by a factor of 5:

Command message:
TA: HMAG 5.00

RELATED COMMANDS

HPOS

DISPLAY

HOR_POSITION, HPOS

Command /Query

DESCRIPTION

The HOR_POSITION command horizontally positions the geometric center of the intensified zone on the source trace. Allowed positions range from division -7 to 7. If this would cause the horizontal position of any expanded trace to go outside the left or right screen boundaries, the difference of positions is adapted and then applied to the traces.

The VAB bit (bit 2) in the STB register is set if a value outside the legal range is specified.

The HOR_POSITION query returns the position of the geometric center of the intensified zone on the source trace.

COMMAND SYNTAX

<exp_trace>: Hor_POSition <hor_position>
<exp_trace>: = {TA, TB, TC, TD}
<hor_position>: = -7 to 7 DIV(The range of the value is related to the size of the screen). the range of the <hor_position> is related to the magnification factors of command HMAG. While the range after magnifying beyond the screen could display, it will be adjusted to the proper value.

QUERY SYNTAX

<exp_trace>: Hor_POSition?

RESPONSE FORMAT

<exp_trace>: Hor_POSition <hor_position>

EXAMPLE

The following instruction positions the center of the intensified zone on the trace currently viewed by Trace A (TA) at division 3:

Command message:
TA: HPOS 3

RELATED COMMANDS

HMAG

HARD COPY

HARDCOPY_SETUP, HCSU

Command /Query

DESCRIPTION

The HARDCOPY_SETUP command configures the instrument's hard-copy driver.

COMMAND SYNTAX

HCSU PSIZE, <page_size>,
ISIZE, <image_size>,
FORMAT, <format>, BCKG,
<bckg>, PRTKEY, <printkey>

<page_size> :={ DEFAULT}
<printkey>:={SAVE,PRINT}
<format> : = {PORTRAIT, LANDSCAPE}
<bckg> : = {BLACK, WHITE}
<image_size>:={DEFAULT,A4,LETTER}.

QUERY SYNTAX

HCSU?

RESPONSE FORMAT

HCSU PSIZE, <page_size>, ISIZE, <image_size>, FORMAT, <format>, BCKG, <bckg>, PRTKEY, <printkey>

EXAMPLE

The following example selects PORTRAIT format, sets the size of the image to "6*8CM":

Command message:
HCSU ISIZE, 6*8CM, FORMAT, PORTRAIT

RELATED COMMANDS

SCDP

MISCELLANEOUS

*IDN?

Query

DESCRIPTION

The *IDN? query is used for identification purposes``. The response consists of four different fields providing information on the manufacturer, the scope model, the serial number and the firmware revision level.

QUERY SYNTAX

*IDN?

RESPONSE FORMAT

*IDN SIGLENT, <model>, <serial_number>, <firmware_level>

<model> : = A eleven characters model identifier

<serial_number> : = A 14-digit decimal code

<firmware_level> : = similar to k.xx.yy.zz

EXAMPLE

This example issues an identification request to the scope:

Command message:
*IDN?

Response message:
*IDN
SIGLENT SDS1102CML,SDS00002110025,
3.01.01.22

DISPLAY

INTENSITY, INTS

Command /Query

DESCRIPTION

The INTENSITY command sets the intensity level of the grid or the trace.

The intensity level is expressed as a percentage (PCT). A level of 100 PCT corresponds to the maximum intensity whilst a level of 0 PCT sets the intensity to its minimum value.(The minimum value of the trace is 30 PCT)

The response to the INTENSITY? Query indicates the grid and trace intensity levels.

COMMAND SYNTAX

INTenSity GRID, <value>, TRACE, <value>
<value> : = 0(or 30) to 100 [PCT]

Note 1: Parameters are grouped in pairs. The first of the pair names the variable to be modified, whilst the second gives the new value to be assigned. Pairs may be given in any order and be restricted to those variables to be changed.

Note 2: The suffix PCT is optional.

QUERY SYNTAX

INTenSity?

RESPONSE FORMAT

INTenSity TRACE, <value>, GRID, <value>

EXAMPLE

The following instruction enables remote control of the intensity, and changes the grid intensity level to 75%:

Command message:
INTS GRID, 75

STATUS INR?

Query

DESCRIPTION

The INR? query reads and clears the contents of the INternal state change Register(INR). The INR register (table below) records the completion of various internal operations and state transitions.

Note : This command only supports 0 bit and 13 bit.

Internal State Register Structure (INR)

Bit	Bit Value	Description	
15…14		0	Reserved for future use
13	8192	1	Trigger is ready
12	4096	1	Pass/Fail test detected desired outcome
11	2048	1	Waveform processing has terminated in Trace D
10	1024	1	Waveform processing has terminated in Trace C
9	512	1	Waveform processing has terminated in Trace B
8	256	1	Waveform processing has terminated in Trace A
7	128	1	A memory card, floppy or hard disk exchange has been detected
6	64	1	Memory card, floppy or hard disk has become full in “AutoStore Fill” mode
5	32	0	Reserved for LeCroy use
4	16	1	A segment of a sequence waveform has been acquired
3	8	1	A time-out has occurred in a data block transfer
2	4	1	A return to the local state is detected
1	2	1	A screen dump has terminated
0	1	1	A new signal has been acquired

QUERY SYNTAX

INR?

RESPONSE FORMAT

INR <value>
<value> : = 0 to 65535

EXAMPLE

If we send INR? query after have triggered the INR register:

Command message1:

INR?

Response message1:
INR 8913

If we send INR? query while the instrument didn't trigger, the INR register:

Command message2:
INR?

Response message2:
INR 8912

If we send INR? query after have sent a INR? query and the mode of the instrument is STOP The INR register:

Command message3:
INR?

Response message3:
INR 0

If we send INR? query while there is no and then make the instrument triggered. Finally we send another INR? query
the INR register:

Command message4:
INR?

Response message4:
INR 1

RELATED COMMANDS ALL_STATUS? ,*CLS

DISPLAY

INVERTSET, INVS

Command /Query

DESCRIPTION

The INVERTSET command inverts the specified traces or the waveform of math.

The response to the INVERTSET? query indicates whether the specified waveform is invert.

COMMAND SYNTAX

<trace>:INVERTSET < state >
< trace > : = {C1,C2,C3,C4,MATH}
< state >:= {ON,OFF}

QUERY SYNTAX

<trace>:INVERTSET?

RESPONSE FORMAT

<trace>:INVERTSET < state >

EXAMPLE

The following instruction inverts the trace of channel 1:

Command message:
C1:INVS ON

MISCELLANEOUS

LOCK, LOCK

Command /Query

DESCRIPTION

The LOCK command enables or disables the panel keyboard of the instrument.

When any command or query is executed in either local or remote state, the functions of the panel keys except “FORCE” are not available. When the panel keyboard of the instrument is locked , press “FORCE” key can enable the panel keyboard function.

The LOCK? query returns the status of the panel keyboard of the instrument.

COMMAND SYNTAX

```
LOCK < status >
<status>:= {ON,OFF}
```

QUERY SYNTAX

```
LOCK?
```

RESPONSE FORMAT

```
LOCK < status >
```

EXAMPLE

The following instruction enables the functions of the panel keys:

Command message:
LOCK ON

ACQUISITION

MATH_VERT_POS, MTVP

Command /Query

DESCRIPTION

The MATH_VERT_POS command controls the vertical position of the math waveform with specified source.

The FFT waveform isn't included. But we have another command which called VPOS to control its vertical position.

The response to the MATH_VERT_POS? query indicates the value of the vertical position of the math waveform.

COMMAND SYNTAX

MATH_VERT_POS <position>
<position>:= the position is related to the position of the screen center. For example, if we set the position of MTVP to 50. The math waveform will be displayed 1 grid up to the vertical center of the screen. Namely one grid is 50.

QUERY SYNTAX

MATH_VERT_POS?

RESPONSE FORMAT

MATH_VERT_POS < position >

EXAMPLE

The following instruction changes the vertical position of the math waveform to 1 grid up to the screen vertical centre:

Command message:
MTVP 50

ACQUISITION

MATH_VERT_DIV, MTVD

Command /Query

DESCRIPTION

The MATH_VERT_DIV command controls the vertical sensitivity of the math waveform of specified source. We can only set the value of existing

The FFT waveform isn't included.

The response to the MATH_VERT_DIV? query indicates the specified scale of math waveform of specified source.

COMMAND SYNTAX

MATH_VERT_DIV < scale >
< scale >:= 1PV/div ~ 100V/div.

QUERY SYNTAX

MATH_VERT_DIV?

RESPONSE FORMAT

MATH_VERT_DIV < scale >

EXAMPLE

The following instruction changes the vertical sensitivity of the math waveform of specified source to 1V/div:

Command message:
MTVD 1V

FUNCTION

MEMORY_SIZE, MSIZ

Command /Query

DESCRIPTION

The MEMORY_SIZE command sets the maximal depth of memory.

The response to the MEMORY_SIZE? query the maximal depth of memory.

COMMAND SYNTAX

MEMORY_SIZE <size>
<size>:= {7K, 14K, 70K, 140K, 700K, 1.4M,7M,14M}

QUERY SYNTAX

MEMORY_SIZE?

RESPONSE FORMAT

MEMORY_SIZE <size>

EXAMPLE

The following instruction sets the maximal depth of memory to 14M.

Command message:
MSIZ 14M

ACQUISITION

OFFSET, OFST

Command /Query

DESCRIPTION

The OFFSET command allows adjustment of the vertical offset of the specified input channel. The maximum ranges depend on the fixed sensitivity setting.

If an out-of-range value is entered, the oscilloscope is set to the closest possible value and the VAB bit (bit 2) in the STB register is set.

The OFFSET? query returns the offset value of the specified channel.

COMMAND SYNTAX

<channel>: OFfSeT <offset>
<channel> : = {C1, C2, C3,C4}
<offset> : = See the SDS1000X specifications.

QUERY SYNTAX

<channel>: OFfSeT?

RESPONSE FORMAT

<channel>: OFfSeT <offset>

EXAMPLE

The following command sets the offset of Channel 2 to -3 V:

Command message:
C2: OFST -3V

STATUS

*OPC

Command /Query

DESCRIPTION

The *OPC (OPeration Complete) command sets to true the OPC bit (bit 0) in the standard Event Status Register (ESR). This command has no other effect on the operation of the oscilloscope because the instrument starts parsing a command or query only after it has completely processed the previous command or query.

The *OPC? query always responds with the ASCII character “1” because the oscilloscope only responds to the query when the previous command has been entirely executed.

COMMAND SYNTAX

*OPC

QUERY SYNTAX

*OPC?

RESPONSE FORMAT

*OPC 1

MISCELLANEOUS

*OPT
Query

DESCRIPTION

The *OPT? query identifies oscilloscope options: installed software or hardware that is additional to the standard instrument configuration. The response consists of a series of response fields listing all the installed options.

QUERY SYNTAX

*OPT?

RESPONSE FORMAT

*OPT <option>

NOTE: If no option is present, the character 0 will be returned.

EXAMPLE： The following instruction queries the installed options:

```
*OPT?
Return: *OPT RS232,NET,USBTMC
```

CURSOR

PARAMETER_CLR, PACL

Command

DESCRIPTION The PARAMETER_CLR command clears the P/F test counter and starts it again at 0.

COMMAND SYNTAX PArameter_CLr

RELATED COMMANDS PARAMETER_VALUE PFDD

CURSOR

PARAMETER_CUSTOM, PACU

Command /Query

DESCRIPTION

The PARAMETER_CUSTOM command controls the parameters that have customizable qualifiers.

Note: The measured value of a parameter setup with PACU may be read using PAVA?

COMMAND SYNTAX

PArameter_CUstom <line>,
<parameter>,<qualifier><line> : = 1 to 5
<parameter> : ={PKPK, MAX, MIN, AMPL, TOP, BASE, CMEAN, MEAN, RMS, CRMS, OVSN, FPRE, OVSP, RPRE, PER, FREQ, PWID, NWID, RISE, FALL, WID, DUTY, NDUTY,PHASE,FRR,FRF,FFR,FFF,LRR,LRF,LFR,LFF }
<qualifier> : = {C1,C2,C3,C4,C1-C2,C1-C3,C1-C4,C2-C3,C2-C4,C3-C4}
Measurement qualifier specific to each(source option)

QUERY SYNTAX

PArameter_CUstom? <line>

RESPONSE FORMAT

PArameter_Custom <line>, <parameter>, <qualifier>

EXAMPLE

Command Example PACU 2, PKPK, C1
Query/Response Examples PACU? 2 returns:
PACU 2, PKPK, C1
PAVA? CUST2 returns:
C2: PAVA CUST2, 160.00mV

RELATED

COMMANDS PARAMETER_CLR, PARAMETER_VALUE

CURSOR PARAMETER_VALUE?, PAVA?

Query

DESCRIPTION

The PARAMETER_VALUE query returns the measurement values.

Parameters Available on All Models			
ALL	all parameters	NDUTY	negative duty cycle
AMPL	amplitude	NWID	negative width
BASE	base	OVSN	negative overshoot
CMEAN	mean for cyclic waveform	OVSP	positive overshoot
CRMS	root mean square for cyclic part of waveform	PKPK	peak-to-peak
DUTY	duty cycle	PER	period
FALL	falltime	RPRE	(Vmin-Vbase)/ Vamp before the waveform rising transition
FREQ	frequency	PWID	positive width
FPRE	(Vmin-Vbase)/ Vamp before the waveform falling transition	RMS	root mean square
MAX	maximum	RISE	risetime
MIN	minimum	TOP	top
MEAN	mean	WID	width

Custom Parameters Defined using PARAMETER_CUSTOM Command				
CUST1	CUST2	CUST3	CUST4	CUST5

QUERY SYNTAX

<trace>: PArameter_VAlue? [<parameter>, ... , <parameter>]
<trace>: = { C1, C2, C3, C4}
<parameter> : = See table of parameter names on previous table.

RESPONSE FORMAT

<trace>: PArameter_VAlue <parameter>, <value> [, ... , <parameter>,<value>]

EXAMPLE

The following query reads the risetime of Channel 2

Command message:
C2: PAVA? RISE

Response message:
C2: PAVA RISE, 3.6E-9S

RELATED COMMANDS

CURSOR_MEASURE, CURSOR_SET, PARAMETER_CUSTOM

ACQUISITION

PEAK_DETECT, PDET

Command /Query

DESCRIPTION

The PEAK_DETECT command switches ON or OFF the peak detector built into the acquisition system.

The PEAK_DETECT? query returns the current status of the peak detector.

COMMAND SYNTAX

Peak_DETect <state>
<state> : = {ON, OFF}

QUERY SYNTAX

Peak_DETect?

RESPONSE FORMAT

PDET <state>

EXAMPLE

The following instruction turns on the peak detector:

Command message:
PDET ON

DISPLAY

PERSIST, PERS

Command /Query

DESCRIPTION

The PERSIST command enables or disables the persistence display mode.

COMMAND SYNTAX

PERSist <mode>
<mode> : = {ON, OFF}

QUERY SYNTAX

PERSist?

RESPONSE FORMAT

PERSist <mode>

EXAMPLE

The following code turns the persistence display ON:

Command message:
PERS ON

RELATED COMMANDS

PERSIST_SETUP

DISPLAY

PERSIST_SETUP, PESU

Command /Query

DESCRIPTION

The PERSIST_SETUP command selects the persistence duration of the display, in seconds,in persistence mode.

The PERSIST_SETUP? query indicates the current status of the persistence.

COMMAND SYNTAX

PErsist_SetUp <time>
<time>：={1，5，10，30,Infinite}

QUERY SYNTAX

PErsist_SetUp?

RESPONSE FORMAT

PErsist_SetUp <time>

EXAMPLE

The following instruction sets the variable persistence at 5 Seconds:

Command message:
PESU 5

RELATED COMMANDS

PERSIST

SAVE/RECALL SETUP

PANEL_SETUP, PNSU

Command /Query

DESCRIPTION

The PANEL_SETUP command complements the *SAV or *RST commands. PANEL_SETUP allows you to archive panel setups in encoded form on external storage media.Only setup data read by the PNSU? query can be recalled into the oscilloscope.

COMMAND SYNTAX

PaNel_SetUp <setup>
<setup> : = A setup previously read by PNSU?

QUERY SYNTAX

PaNel_SetUp?

RESPONSE FORMAT

PaNel_SetUp <setup>

EXAMPLE

The following instruction saves the scilloscope's current panel setupin the file PANEL.SET:

Command message:
PNSU?

RELATED COMMANDS

*RCL, *SAV

FUNCTION

PF_DISPLAY, PFDS

Command /Query

DESCRIPTION

The PF_DISPLAY command enables or disables to turn the test and display the message in the pass/fail option.

The response to the PF_DISPLAY? query indicates whether the test is enabled and the message of pass/fail is displayed

COMMAND SYNTAX

PF_DISPLAY TEST,<state>,DISPLAY,<state>
<state> : = {ON, OFF}

QUERY SYNTAX

PF_DISPLAY TEST?

RESPONSE FORMAT

PF_DISPLAY TEST <state>,DISPLAY,<state>

EXAMPLE

The following instruction enables to turn on the test and display the message of pass/fail:

Command message:
PFDS TEST,ON,DISPLAY,ON

FUNCTION

PF_SET, PFST

Command /Query

DESCRIPTION

The PF_SET command sets the X mask and the Y mask of the mask setting in the pass/fail option.

The response to the PF_ SET? query indicates the value of the X mask and the Y mask.

COMMAND SYNTAX

PF_ SET XMASK, <div>,YMASK, <div>
<div> : = 0.04div~4.0div

QUERY SYNTAX

PF_ SET?

RESPONSE FORMAT

PF_ SET XMASK, <div>,YMASK, <div>

EXAMPLE

The following instruction sets the X mask to 0.4div and the Y mask to 0.5div of the mask setting in the pass/fail option:

Command message:
PFST XMASK,0.4,YMASK,0.5

RELATED COMMANDS

PFSL PFST

SAVE/RECALL

PF_SAVELOAD, PFSL

Command

DESCRIPTION

The PF_SAVELOAD command saves or recalls the created mask setting.

COMMAND SYNTAX

PF_ SAVELOAD LOCATION, <location>,ACTION, <action>

The <location> means to save the created mask setting to the internal memories or the external memories.

<location> : = {IN,EX}
IN means to save the mask setting to the internal memories while EX means the external memories.
<action> := {SAVE,LOAD}
SAVE means to save the mask setting while LOAD means recall the stored mask setting.

EXAMPLE

The following instruction saves the mask setting to the internal memories:

Command message:
PFSL LOCATION,IN,ACTION,SAVE

RELATED COMMANDS

PFCM

FUNCTION PF_CONTROL, PFCT

Command /Query

DESCRIPTION

The PF_CONTROL command controls the pass/fail controlling options: “operate”, “output” and the “stop on output”.

See instrument’s Operator Manual for these options

The response to the PF_ CONTROL? query indicates the controlling options of the pass/fail.

COMMAND SYNTAX

PF_ CONTROL TRACE,<trace>,CONTROL,<control>,OUTPUT,<output>,OUTPUTSTOP,<state>
<trace> : = {C1,C2,C3,C4}
<control> : = {START,STOP}
<output> : = {FAIL,PASS}
<state> : = {ON,OFF}

QUERY SYNTAX

PF_ CONTROL?

RESPONSE FORMAT

PF_ CONTROL TRACE,<trace>,CONTROL,<control>, OUTPUT,<output>,OUTPUTSTOP,<state>

EXAMPLE

The following instruction sets source to channel 1, “operate” to “start”, “output” to “pass” and “stop on output” to “off”:

Command message:

```
PFCT TRACE,C1,CONTROL,START,
OUTPUT,PASS,OUTPUTSTOP,OFF
```

FUNCTION **PF_CREATEM, PFCM**

Command

DESCRIPTION

The PF_CREATEM command creates the mask of the pass/fail.

COMMAND SYNTAX

PF_ CREATEM

EXAMPLE

The following instruction creates the mask of the pass/fail.:

Command message:
PFCM

RELATED COMMANDS

PFSL PFST

FUNCTION PF_DATADIS, PFDD

Query

DESCRIPTION

The PF_DATADIS? query returns the number of the fail ,pass and total number that the screen showing.

QUERY SYNTAX

PF_ DATADIS?

RESPONSE FORMAT

PF_ DATADIS
FAIL,<num>,PASS,<num>,total,<num>

EXAMPLE

The following instruction returns the number of the message display of the pass/fail:

Command message:
PFDD FAIL,0,PASS,0,TOTAL,0

RELATED COMMANDS

PACL

SAVE/RECALL SETUP

*RCL
Command

DESCRIPTION

The *RCL command sets the state of the instrument, using one of the ten non-volatile panel setups, by recalling the complete front-panel setup of the instrument. Panel setup 0 corresponds to the default panel setup.

The *RCL command produces the opposite effect of the *SAV command.

If the desired panel setup is not acceptable, the EXecution error status Register (EXR) is set and the EXE bit of the standard Event Status Register (ESR) is set.

COMMAND SYNTAX

```
*RCL <panel_setup>
<panel_setup>:= 0 to 20
```

EXAMPLE

The following recalls the instrument setup previously stored in panel setup 3:

Command message:

```
*RCL 3
```

RELATED COMMANDS

PANEL_SETUP, *SAV, EXR

SAVE/RECALL SETUP

RECALL_PANEL, RCPN

Command

DESCRIPTION

The RECALL_PANEL command recalls a front-panel setup from the current directory on mass storage.

COMMAND SYNTAX

ReCall_PaNel DISK, <device>, FILE, ‘<filename>’

<device> : = {UDSK}

<filename>： = A waveform file under a legal DOS path . A filename-string of up to eight characters, with the extension “.SET”. (This can include the ‘/’ character to define the root directory.)

EXAMPLE

The following recalls the front-panel setup from file SEAN. SET in a USB memory device:

Command message:
RCPN DISK, UDSK, FILE,‘SEAN. SET’

RELATED COMMANDS

PANEL_SETUP, *SAV, STORE_PANEL, *RCL

SAVE/RECALL SETUP

*RST
Command

DESCRIPTION

The *RST command initiates a device reset.
The *RST sets recalls the default setup.

COMMAND SYNTAX

*RST

EXAMPLE

This example resets the oscilloscope:

Command message:
*RST

RELATED COMMANDS

*CAL, *RCL

FUNCTION

REF_SET, REFS

Command /Query

DESCRIPTION

The REF_SET command sets the reference waveform and its options.

The response to the REF_ SET? query indicates whether the specified reference waveform is turned on.

COMMAND SYNTAX

REF _ SET TRACE,<trace>REF,<ref>,state,
<state>,SAVE,DO

<trace> : = {C1,C2,C3,C4,MATH}

<ref> : = {RA,RB,RC,RD}

The Rx(x is A,B,C,D) is that which one can be stored or displayed

<state> := {ON,OFF}

The state enables or disables to display the specified reference waveform.

If the command syntax have the option that SAVE,DO, means that the specified trace will be saved to the specified reference waveform.

QUERY SYNTAX

REF _ SET? REF,<ref>

RESPONSE FORMAT

REF _ SET REF,<ref>,STATE,<state>

EXAMPLE

The following instruction saves the channel 1 waveform to the REFA, and turns on REFA:

Command message:

```
REFS TRACE,C1,REF,RA,
STATE,ON,SAVE,DO
```

SAVE/RECALL SETUP

***SAV**
Command

DESCRIPTION

The *SAV command stores the current state of the instrument in internal memory. The *SAV command stores the complete front-panel setup of the instrument at the time the command is issued.

COMMAND SYNTAX

*SAV <panel_setup>
<panel_setup>: = 1 to 20

EXAMPLE

The following saves the current instrument setup in Panel Setup 3:

Command message:
*SAV 3

RELATED COMMANDS

PANEL_SETUP, *RCL

HARD COPY

SCREEN_DUMP, SCDP

Command

DESCRIPTION

The SCREEN_DUMP command is used to obtain the screen information of image format .

COMMAND SYNTAX

SCreen_DumP

EXAMPLE

The following command transfers the screen information of image format to the controller

Command message:
SCDP

DISPLAY

SCREEN_SAVE, SCSV

Command /Query

DESCRIPTION

The SCREEN_SAVE command controls the automatic Screen Saver, which automatically shuts down the internal color monitor after a preset time.

The response to the SCREEN_SAVE? query indicates whether the automatic screen saver feature is on or off.

Note: When the screen save is in effect, the oscilloscope is still fully functional.

COMMAND SYNTAX

SCreen_SaVe <enabled>
<enabled> : = {YES, NO}

QUERY SYNTAX

SCreen_SaVe?

RESPONSE FORMAT

SCreen_SaVe <enabled>

EXAMPLE

The following enables the automatic screen saver:

Command message:
SCSV YES

STATUS

*SRE

Command /Query

DESCRIPTION

The *SRE command sets the Service Request Enable register (SRE). This command allows the user to specify which summary message bit(s) in the STB register will generate a service request.

A summary message bit is enabled by writing a '1' into the corresponding bit location. Conversely, writing a '0' into a given bit location prevents the associated event from generating a service request (SRQ). Clearing the SRE register disables SRQ interrupts.

The *SRE? query returns a value that, when converted to a binary number, represents the bit settings of the SRE register.

Note: that bit 6 (MSS) cannot be set and its returned value is always zero.

COMMAND SYNTAX

*SRE <value>
<value> : = 0 to 255

QUERY SYNTAX

*SRE?

RESPONSE FORMAT

*SRE <value>

EXAMPLE

The following instruction allows an SRQ to be generated as soon as the MAV summary bit (bit 4, i.e. decimal 16) or the INB summary bit (bit 0, i.e. decimal 1) in the STB register, or both, are set. Summing these two values yields the SRE mask 16+1 = 17.

Command message:
*SRE 17

STATUS *STB?

Query

DESCRIPTION

The *STB? query reads the contents of the 488.1 defined status register (STB), and the Master Summary Status (MSS). The response represents the values of bits 0 to 5 and 7 of the Status Byte register and the MSS summary message.

The response to a *STB? Query is identical to the response of a serial poll except that the MSS summary message appears in bit 6 in place of the RQS message.

QUERY SYNTAX

*STB?

RESPONSE FORMAT

*STB <value>
<value> : = 0 to 255

EXAMPLE

The following reads the status byte register:

Command message:
*STB?

Response message:
*STB 0

RELATED COMMANDS

ALL_STATUS, *CLS, *SRE

ADDITIONAL INFORMATION

Status Byte Register (STB)				
Bit	Bit Value	Bit Name	Description	Note
7	128	DIO7	0 reserved for future use	

6	64	MSS/RQS
		MSS=1
		RQS=1
			at least 1 bit in STB masked by SRE is 1
			service is
			requested
				(1)
				(2)

5	32	ESB	1 an ESR enabled event has occurred	(3)
4	16	MAV	1 output queue is not empty	(4)
3	8	DIO3	0 reserved	
2	4	VAB	1 a command data value has been adapted	(5)
1	2	DIO1	0 reserved	
0	1	INB	1 an enabled INternal state change has occurred	(6)

Notes

(1) The Master Summary Status (MSS) indicates that the instrument requests service, whilst the Service Request status — when set — specifies that the oscilloscope issued a service request. Bit position 6 depends on the polling method:
Bit 6 = MSS if an *STB? Query is received
= RQS if serial polling is conducted

(2) Example: If SRE=10 and STB=10 then MSS=1. If SRE=010 and STB=100 then MSS=0.

(3) The Event Status Bit (ESB) indicates whether or not one or more of the enabled IEEE 488.2 events have occurred since the last reading or clearing of the Standard Event Status Register (ESR). ESB is set if an enabled event becomes true (1).

(4) The Message AVailable bit (MAV) indicates whether or not the Output queue is empty. The MAV summary bit is set true (1) whenever a data byte resides in the Output queue.

(5) The Value Adapted Bit (VAB) is set true (1) whenever a data value in a command has been adapted to the nearest legal value. For instance, the VAB bit would be set if the timebase is redefined as 2 µs/div since the adapted value is 2.5 µs/div.

(6) The INternal state Bit (INB) is set true (1) whenever certain enabled internal states are entered. For further information, refer to the INR query.

STOP

Command

DESCRIPTION

The STOP command immediately stops the acquisition of a signal. If the trigger mode is AUTO or NORM.

COMMAND SYNTAX

STOP

EXAMPLE

The following stops the acquisition process:

Command message:
STOP

RELATED COMMANDS

ARM_ACQUISITION, TRIG_MODE, WAIT

WAVEFORM TRANSFER

STORE, STO

Command

DESCRIPTION

The STORE command stores the contents of the specified trace into the current directory in a USB memory device.

COMMAND SYNTAX

STOre <trace>
<trace>: = {TA, TB, TC, TD, C1, C2, C3, C4,ALL_DISPLAYED}
<dest>: ={ UDSK}

Note: If the STORE command is sent without any argument, and the current trace isn't enabled, the current trace will be enabled and stored in the Store Setup. This setup can be modified using the STORE_SETUP command.

EXAMPLE

The following command stores the contents of Channel 1(C1) into USB memory device:

Command message:

```
STO C1, UDSK
```

The following command stores all currently displayed waveforms onto the USB memory device:

Command message:

```
STO ALL_DISPLAYED, UDSK
```

RELATED COMMANDS

STORE_SETUP, RECALL

SAVE/RECALL SETUP

STORE_PANEL, STPN

Command

DESCRIPTION

The STORE_PANEL command stores the complete front-panel setup of the instrument, at the time the command is issued, into a file on the specified-DOS path directory in a USB memory device.

COMMAND SYNTAX

STore_PaNel DISK, <device>, FILE, ‘<filename>’

<device>：={UDSK}

< directory >：=A legal DOS path or filename. A filename -string of up to 8 characters, with the extension “.SET”. (This can include the ‘/’ character to define the root directory.)

EXAMPLE

The following code saves the current instrument setup to root directory of the USB memory device in a file called “SEAN.SET”:

Command message:

```
STore_PaNel DISK,UDSK,FILE,'SEAN.SET'
```

The following code saves the current instrument setup to specified-directory of the USB memory device in a file called “SEAN.SET”:

Command message:

```
STore_PaNel DISK,UDSK,FILE,'/AAA/SEAN'
```

RELATED COMMANDS

*SAV, RECALL_PANEL, *RCL

WAVEFORM TRANSFER STORE_SETUP, STST

Command /Query

DESCRIPTION

The STORE_SETUP command controls the way in which traces will be stored. A single trace or all displayed traces may be enabled for storage.

COMMAND SYNTAX

STore_SeTup [<trace>, <dest>]
<trace> : = {C1,C2,C3,C4,ALL_DISPLAYED }
<dest>: ={ UDSK}

QUERY SYNTAX

STore_SeTup?

RESPONSE FORMAT

STore_SeTup <trace>, <dest>

EXAMPLE

The following command selects Channel 1 to be stored.

Command message:
STST C1, UDSK

RELATED COMMANDS

STORE, INR

ACQUISITION

SAMPLE_STATUS, SAST

Query

DESCRIPTION

The SAST? query the acquisition status of the scope.

QUERY SYNTAX

SAST?

RESPONSE FORMAT

SAST < status >

EXAMPLE

The following command reads the acquisition status of the scope.

Command message:
SAST?

Response message:
SAST trig'd

ACQUISITION SAMPLE_RATE, SARA

Query

DESCRIPTION

The SARA? query returns the sample rate of the scope.

QUERY SYNTAX

SARA?

RESPONSE FORMAT

SARA <value>

EXAMPLE

The following command reads the sample rate of the scope.

Command message:
SARA?

Response message:
SARA 500.0kSa

ACQUISITION

SAMPLE_NUM, SANU

Query

DESCRIPTION

The SANU? query returns the number of sampled points available from last acquisition and the trigger position.

QUERY SYNTAX

SANU? <channel>

RESPONSE FORMAT

SANU <value>

EXAMPLE

The following command reads the number of sampled points available from last acquisition from the Channel 2.

Command message:
SANU? C2

Response message:
SANU 6000

ACQUISITION

SKEW, SKEW

Command

DESCRIPTION

The SKEW command sets the skew value of the specified trace.

The response to the SKEW? query indicates the skew value of the specified trace.

COMMAND SYNTAX

```
<trace>:SKEW <skew>
<trace> : = {C1,C2,C3,C4 }
<skew>: = it is a value about time.
```

QUERY SYNTAX

```
<trace>:SKEW?
```

RESPONSE FORMAT

```
<trace>:SKEW <skew>
```

EXAMPLE

The following command sets channel 1 skew value to 3ns

Command message:

```
C1:SKEW 3NS
```

ACQUISITION

SINXX_SAMPLE, SXSA

Command /Query

DESCRIPTION

The SINXX_SAMPLE command sets the way of interpolation.

The response to the SINXX_SAMPLE? query indicates the way of interpolation.

COMMAND SYNTAX

SINXX_SAMPLE, <state>
<state> : = {ON,OFF}
ON means sine interpolation, and OFF means linear interpolation

QUERY SYNTAX

SINXX_SAMPLE?

RESPONSE FORMAT

SINXX_SAMPLE <state>

EXAMPLE

The following instruction sets the way of the interpolation to sine interpolation:

Command message:
SXSA ON

ACQUISITION

TIME_DIV, TDIV

Command /Query

DESCRIPTION

The TIME_DIV command modifies the timebase setting. The new timebase setting may be specified with suffixes: NS for nanoseconds, US for microseconds, MS for milliseconds, S for seconds, or KS for kiloseconds. An out-of-range value causes the VAB bit (bit 2) in the STB register to be set.

The TIME_DIV? query returns the current timebase setting.

COMMAND SYNTAX

Time_DIV <value>
<value>:={1NS,2NS,5NS,10NS,20NS,50NS,100NS,200NS,500NS,1US,2US,5US,10US,20US,50US,100US,200US,500US,1MS,2MS,5MS,10MS,20MS,50MS,100MS,200MS,500MS,1S,2S,5S,10S,20S,50S}

QUERY SYNTAX

Time_DIV?

RESPONSE FORMAT

Time_DIV <value>

EXAMPLE

The following sets the time base to 500 μs /div:

Command message:
TDIV 500US

RELATED COMMANDS

TRIG_DELAY, TRIG_MODE

WAVEFORM TRANSFER TEMPLATE, TMPL

Query

DESCRIPTION

The TEMPLATE? query produces a copy of the template that describes the various logical entities making up a complete waveform. In particular, the template describes in full detail the variables contained in the descriptor part of a waveform.

QUERY SYNTAX

TeMPLate?

RESPONSE FORMAT

TeMPLate "<template>"
<template> : = A variable length string detailing the structure of a waveform.

RELATED COMMANDS

WF

DISPLAY

TRACE, TRA

Command /Query

DESCRIPTION

The TRACE command enables or disables the display of a trace. An environment error is set if an attempt is made to display more than four waveforms.

The TRACE? query indicates whether the specified trace is displayed or not.

COMMAND SYNTAX

<trace>: TRAce <mode>
<trace> : = {C1, C2, C3, C4, TA, TB, TC, TD}
<mode> : = {ON, OFF}

QUERY SYNTAX

<trace>: TRAce?

RESPONSE FORMAT

<trace>: TRAce <mode>

EXAMPLE

The following command displays Channel 1 (C1):

Command message:
C1: TRA ON

ACQUISITION

*TRG
Command

DESCRIPTION

The *TRG command executes an ARM command.

COMMAND SYNTAX

*TRG

EXAMPLE

The following command enables signal acquisition:

Command message:

```
*TRG
```

RELATED COMMANDS

ARM_ACQUISITION, STOP, WAIT

ACQUISITION

TRIG_COUPLING, TRCP

Command /Query

DESCRIPTION

The TRIG_COUPLING command sets the coupling mode of the specified trigger source.

The TRIG_COUPLING? query returns the trigger coupling of the selected source.

COMMAND SYNTAX

<trig_source>: TRig_CouPling <trig_coupling>
<trig_source>: = {C1, C2, C3, C4, EX, EX5, LINE}
<trig_coupling>: = {AC,DC,HFREJ,LFREJ}

QUERY SYNTAX

<trig_source>: TRig_CouPling?

RESPONSE FORMAT

<trig_source>: TRig_CouPling <trig_coupling>

EXAMPLE

The following command sets the coupling mode of the trigger source Channel 2 to AC:

Command message:
C2: TRCP AC

RELATED COMMANDS

TRIG_COUPLING, TRIG_DELAY, TRIG_LEVEL, TRIG_MODE, TRIG_SELECT, TRIG_SLOPE

ACQUISITION TRIG_DELAY, TRDL

Command /Query

DESCRIPTION

The TRIG_DELAY command sets the time at which the trigger is to occur with respect to the first acquired data point.

This mode is called pre-trigger acquisition, as data are acquired before the trigger occurs. Negative trigger delays must be given in seconds. This mode is called post-trigger acquisition, as the data are acquired after the trigger has occurred.

If a value outside the range, the trigger time will be set to the nearest limit and the VAB bit (bit 2) will be set in the STB register. The response to the TRIG_DELAY? query indicates the trigger time with respect to the first acquired data point.

COMMAND SYNTAX

TRig_DeLay <value>
<value>： = the range of value is related to the timebase.

Note: The suffix S is optional and assumed.

QUERY SYNTAX

TRig_DeLay?

RESPONSE FORMAT

TRig_DeLay <value>

EXAMPLE

The following command sets the trigger delay to -2ms (posttrigger):

Command message:
TRDL -2MS

RELATED COMMANDS

TIME_DIV, TRIG_COUPLING, TRIG_LEVEL, TRIG_MODE, TRIG_SELECT, TRIG_SLOPE

ACQUISITION

TRIG_LEVEL, TRLV

Command /Query

DESCRIPTION

The TRIG_LEVEL command adjusts the trigger level of the specified trigger source. An out-of-range value will be adjusted to the closest legal value and will cause the VAB bit (bit 2) in the STB register to be set.

The TRIG_LEVEL? query returns the current trigger level.

COMMAND SYNTAX

<trig_source>: TRig_LeVel <trig_level>
<trig_source>: = {C1, C2, C3, C4, EX, EX5}
<trig_level>: = -4.5DIV* volt/div to 4.5DIV * volt/div

Note: The suffix V is optional and assumed.

QUERY SYNTAX

<trig_source>: TRig_LeVel?

RESPONSE FORMAT

<trig_source>: TRig_LeVel <trig_level>

EXAMPLE

The following code adjusts the trigger level of Channel 3 to 52.00mv:

Command message:

```
C3:TRig_LeVel 52.00mv
```

RELATED COMMANDS

TRIG_COUPLING, TRIG_DELAY, TRIG_MODE, TRIG_SELECT, TRIG_SLOPE

ACQUISITION

TRIG_LEVEL2, TRLV2

Command /Query

DESCRIPTION

The TRIG_LEVEL command adjusts the second trigger
level of the specified trigger source. If want to use this command . The trigger type must have two trigger lines. An out-of-range value will be adjusted to the closest legal value and will cause the VAB bit (bit 2) in the STB register to be set.

The TRIG_LEVEL? query returns the current trigger level.

COMMAND SYNTAX

<trig_source>: TRig_LeVel2 <trig_level>
<trig_source>: = {C1, C2, C3, C4, EX, EX5}
<trig_level>: = -4.5DIV* volt/div to 4.5DIV * volt/div

Note: The suffix V is optional and assumed.

QUERY SYNTAX

<trig_source>: TRig_LeVel2?

RESPONSE FORMAT

<trig_source>: TRig_LeVel <trig_level>

EXAMPLE

The following code adjusts the trigger level of Channel 3 to 52.00mv:

Command message:
C3:TRig_LeVel 52.00mv

RELATED COMMANDS

TRIG_COUPLING, TRIG_DELAY, TRIG_MODE, TRIG_SELECT, TRIG_SLOPE

ACQUISITION

TRIG_MODE, TRMD

Command /Query

DESCRIPTION

The TRIG_MODE command specifies the trigger mode.

The TRIG_MODE? query returns the current trigger mode.

NOTE: STOP is a part of the option of this command, but is not a trigger mode of the instrument

COMMAND SYNTAX

TRig_MoDe <mode>
<mode>: = {AUTO, NORM, SINGLE,STOP}

QUERY SYNTAX

TRig_MoDe？

RESPONSE FORMAT

TRig_MoDe <mode>

EXAMPLE

The following selects the normal mode:

Command message:
TRMD NORM

RELATED COMMANDS

ARM_ACQUISITION, STOP, TRIG_SELECT, TRIG_COUPLING, TRIG_LEVEL, TRIG_SLOP

ACQUISITION **TRIG_SELECT, TRSE**

Command /Query

DESCRIPTION

The TRIG_SELECT command selects the condition that will trigger the acquisition of waveforms. Depending on the trigger type, additional parameters must be specified. These additional parameters are grouped in pairs. The first in the pair names the variable to be modified, while the second gives the new value to be assigned. Pairs may be given in any order and restricted to those variables to be changed.

The TRIG_SELECT? query returns the current trigger condition.

Trigger Notation			
EDGE	Edge	PS	Pulse smaller
GLIT	Glitch	SR	Source
HV	Hold value	TI	Time
HT	Hold type	TV	TV
IL	Interval larger	CHAR	Characteristics
INTV	Interval	LPIC	Lines per picture
IS	Interval smaller	LINE	Line
PL	Pulse larger		

NOTE: The command is unclear and needs more explanation.

COMMAND SYNTAX

For all but TV Trigger TRig_SElect <trig_type>,SR,<source>,QL,<source>,HT,<hold_type>,HV,<hold_value>

<trig_type> : = { EDGE, GLIT, INTV}

<source> : = {C1, C2, C3, C4, LINE, EX, EX5 }

<hold_type> : = {TI, PS, PL, P2,IS,

IL,I2,OFF,EV}

<hold_value> : = See instrument Operator's Manual for valid values

QUERY SYNTAX TRig_SElect?

RESPONSE FORMAT TRig_Select <trig_type>, SR, <source>, HT, <hold_type>, HV, <hold_value>

EXAMPLE The following selects the EDGE trigger with Channel 1 as trigger source. Hold type and hold-value are chosen as "time" and 1.43US:

Command message:

```
TRSE EDGE, SR, C1, HT, TI, HV, 1.43US
```

TV COMMAND SYNTAX TRig_SElect TV, SR, <source>, FLDC,<field_count>,FLD,<field>,CHAR, <characteristics>, IPIC,<ipic>,ILAC,<ilace>,LINE, <line>

<trig_type>: = {TV}
<source> : = {C1, C2, C3,C4 }
<field_count>: = {1,2,4,8}
<field>:=1 to field_count
<characteristics> : = {NTSC, PALSEC,720P/50,720P/60,1080P/50,1080P/60,1080I/50,1080I/60,CUSTOM}
<lpic>:=1 to 1500
<ilace>:= {1,2,4,8}
<line> : = 1 to 525 (PALSEC)
1 to 625(NTSC)

QUERY SYNTAX TRig_SElect?

RESPONSE FORMAT TRig_SElect TV, SR, <source>, CHAR, <characteristic>, LINE, <line>

EXAMPLE The following sets up the trigger system to trigger on the line 17, of the PAL/SECAM TV signal applied to the external input.

Command message:
TRSE TV, SR, EX, CHAR, PALSEC, LINE, 17

RELATED COMMANDS TRIG_COUPLING, TRIG_DELAY, TRIG_LEVEL, TRIG_MODE, TRIG_SLOPE

ACQUISITION TRIG_SLOPE, TRSL

Command /Query

DESCRIPTION

The TRIG_SLOPE command sets the trigger slope of the specified trigger source.

The TRIG_SLOPE? query returns the trigger slope of the selected source.

COMMAND SYNTAX

<trig_source>: TRig_SLope <trig_slope>
<trig_source>: = {C1, C2, C3, C4, EX,EX5 }
<trig_slope>: = {NEG, POS,WINDOW}

QUERY SYNTAX

<trig_source> : TRig_Slope?

RESPONSE FORMAT

<trig_source>: TRig_SLope <trig_slope>

EXAMPLE

The following sets the trigger slope of Channel 2 to negative:

Command message:
C2: TRSL NEG

RELATED COMMANDS

TRIG_COUPLING, TRIG_DELAY, TRIG_LEVEL, TRIG_MODE, TRIG_SELECT, TRIG_SLOPE

ACQUISITION TRIG_WINDOW, TRWI

Command /Query

DESCRIPTION

The TRIG_WINDOW command sets the relative height of the two trigger line of the trigger window type.

The TRIG_WINDOW? query returns relative height of the two trigger line of the trigger window type.

COMMAND SYNTAX

TRig_WIndow <value>
< value >: -4.5DIV* volt/div to 4.5DIV * volt/div

QUERY SYNTAX

TRig_WIndow?

RESPONSE FORMAT

TRig_WIndow < value >

EXAMPLE

The following sets the relative height of the two trigger line of the trigger window type to 2V:

Command message:
TRWI 2V

RELATED COMMANDS TRIG_LEVEL, TRIG_LEVEL2, TRIG_SELECT

ACQUISITION TRIG_PATTERN, TRPA

Command /Query

DESCRIPTION The TRIG_PATTERN command sets the condition of the pattern trigger.

The TRIG_ PATTERN? query returns the condition of the pattern trigger.

COMMAND SYNTAX TRig_PAttern <source>,<status>
[,<source>,<status>][,<source>,<status>][,<source>,
<status>],STATE,<condition>
< source >: ={C1, C2, C3, C4}
<status>:={X,L,H}
< condition >:= {AND, OR, NAND, OR}

QUERY SYNTAX TRig_PAttern?

RESPONSE FORMAT TRig_Pattern
<source>,<status>,<source>,<status>,<source>,<status>,<source>,<status>

EXAMPLE The following sets the channel 2 and channel 3 to low and the condition to AND:

Command message:
TRPA C2,L,C3,L,STATE,AND

RELATED COMMANDS TRIG_LEVEL, TRIG_LEVEL2, TRIG_SELECT

ACQUISITION

UNIT, UNIT

Command /Query

DESCRIPTION

The UNIT command sets the unit of the specified trace.

The UNIT query returns the unit of the specified trace.

COMMAND SYNTAX

<channel>: UNIT <type>
<channel>： = {C1, C2, C3, C4}
<type>： = {V,A}

QUERY SYNTAX

<channel> : UNIT？

RESPONSE FORMAT

<channel>: UNIT <type>

EXAMPLE

The following command sets the unit of the channel 1 to V:

Command message:
C1: UNIT V

DISPLAY

VERT_POSITION, VPOS

Command /Query

DESCRIPTION

The VERT_POSITION command adjusts the vertical position of the specified FFT trace on the screen. It does not affect the original offset value obtained at acquisition time.

The VERT_POSITION? query returns the current vertical position of the specified FFT trace.

COMMAND SYNTAX

```
<trace>: Vert_POSITION <display_offset>
<trace>: = {TA, TB, TC, TD}
<display_offset>：=-40 DIV to 40 DIV
```

Note: The suffix DIV is optional.

QUERY SYNTAX

```
<trace>: Vert_POSition?
```

RESPONSE FORMAT

```
<trace>: Vert_POSITION <display_offset>
```

EXAMPLE

The following shifts FFT Trace A (TA) upwards by +3 divisions relative to the position at the time of acquisition:

Command message:
TA: VPOS 3DIV

ACQUISITION

VOLT_DIV, VDIV

Command /Query

DESCRIPTION

The VOLT_DIV command sets the vertical sensitivity in Volts/div. The VAB bit (bit 2) in the STB register is set if an out-of-range value is entered.

The VOLT_DIV query returns the vertical sensitivity of the specified channel.

COMMAND SYNTAX

<channel>: Volt_DIV <v_gain>
<channel>：= {C1, C2, C3, C4}
<v_gain>：= 2mV to 10V
Note: The suffix V is optional.

QUERY SYNTAX

<channel> : Volt_DIV？

RESPONSE FORMAT

<channel>: Volt_DIV <v_gain>

EXAMPLE

The following command sets the vertical sensitivity of channel 1 to 50 mV/div:

Command message:
C1: VDIV 50MV

WAVEFORM TRANSFER

WAVEFORM, WF

Query

DESCRIPTION

A WAVEFORM? Query transfers a waveform from the oscilloscope to the controller.

A waveform consists of several distinct entities:

1. the descriptor (DESC)

2. the auxiliary data (DAT1) block3. the main data (DAT2) block

The WAVEFORM? Query instructs the oscilloscope to transmit a waveform to the controller. The entities may be queried independently. If the "ALL" parameter is specified,all four or five entities are transmitted in one block in the order enumerated above.

Note:1. The format of the waveform data depends on the current settings specified by the last WAVEFORM_SETUP command.
2.The format of the waveform data can be seen by the TEMPLATE? Query.

QUERY SYNTAX

<trace>: WaveForm?
<trace> : = { C1,C2,C3,C4}

RESPONSE FORMAT

<trace>: WaveForm <waveform_data_block>

EXAMPLE

The following command reads waveform data block of Channel 2:
Command message:
C2: WF?

RELATED COMMANDS

WAVEFORM_SETUP

WAVEFORM TRANSFER

WAVEFORM_SETUP, WFSU

Command /Query

DESCRIPTION

The WAVEFORM_SETUP command specifies the amount of data in a waveform to be transmitted to the controller. The command controls the settings of the parameters listed below.

Notation			
FP	first point	NP	number of points
SP	sparsing		

Sparsing (SP): The sparsing parameter defines the interval between data points. For example:
SP = 0 sends all data points
SP = 1 sends all data points
SP = 4 sends every 4th data point

Number of points (NP): The number of points parameter indicates how many points should be transmitted. For example:
NP = 0 sends all data points
NP = 1 sends 1 data point
NP = 50 sends a maximum of 50 data points
NP = 1001 sends a maximum of 1001 data points

First point (FP): The first point parameter specifies the address of the first data point to be sent. For waveforms acquired in sequence mode, this refers to the relative address in the given segment. For example:
FP = 0 corresponds to the first data point
FP = 1 corresponds to the second data point
FP = 5000 corresponds to data point 5001

The WAVEFORM_SETUP? query returns the transfer parameters currently in use.

COMMAND SYNTAX

WaveForm_SetUp SP, <sparsing>, NP, <number>, FP, <point>

QUERY SYNTAX

WaveForm_SetUp？

Note 1: After power-on, SP is set to 4, NP is set to 1000, and FP is set to 0.

Note 2: Parameters are grouped in pairs. The first of the pair names the variable to be modified, whilst the second gives the new value to be assigned. Pairs may be given in any order and may be restricted to those variables to be changed.

RESPONSE FORMAT

WaveForm_SetUp SP, <sparsing>, NP, <number>, FP, <point>

EXAMPLE

The following command specifies that every 3rd data point (SP=3) starting at address 200 should be transferred:

Command message:
WFSU SP, 3, FP, 200

RELATED COMMANDS

WAVEFORM

ACQUISITION

WAIT, WAIT

Command

DESCRIPTION

The WAIT command prevents the instrument from analyzing new commands until the oscilloscope has completed the current acquisition.

The instrument will be waiting for trigger or the limit time over (if we set it) or the device time out when we sent this command

COMMAND SYNTAX

WAIT <time>

Note : This command have two ways to use. One sets the limited time, another one doesn't set the limited time.

EXAMPLE

If we move the trigger level of the source to the position where the trace isn't triggered. Then we send an ARM command to set the trigger mode to single. Finally we send the WAIT command. The instrument will be waiting for triggering until the time over (if we set it) or time out.

If we move the trigger level of the source, and the instrument is triggered. Then we send an ARM command to set the trigger mode to single. Finally we send the WAIT command. The WAIT command will be finished if we send a FRTR for triggering.

Command message:
WAIT

DISPLAY

XY_DISPLAY, XYDS

Command /Query

DESCRIPTION

The XY_DISPLAY command enables or disables the display the XY format

The response to the XY_DISPLAY? query indicates whether the XY format display is enabled.

COMMAND SYNTAX

XY_DISPLAY <state>
<state>：= {ON, OFF}

QUERY SYNTAX

XY_DISPLAY？

RESPONSE FORMAT

XY_DISPLAY <state>

EXAMPLE

The following command enables to display the XY format:

Command message:
XYDS

Programming Examples

This chapter give some examples for the programmer. In these examples you can see how to use the ni-visa lib and the commands which have been described before this chapter to control our devices. By the examples' guide, you can develop more functions application as you want. This example is developed by Visual Studio project.

Main topics of this part:

●Example of Vc++

●Example of VB

●Example of MATLAB

●Example of LabVIEW

Example of VC++

Environment: Win7 32bit system, Visual Studio

The functions of this example: use the NI-VISA, to control the device with USBTMC or TCP/IP access to do a write and read.

Follow the steps to finish the example:

1、Open Visual Studio, create a new VC++ win32 project.

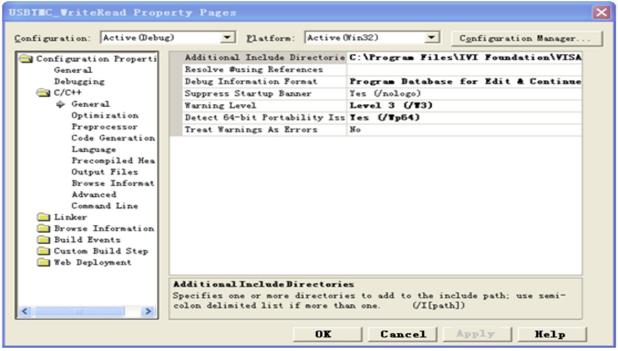
2、Set the project environment to use the NI-VISA lib, there are two ways to use NI-VISA, static or automatic:

2.1 Static: find files: visa.h, visatype.h, visa32.lib in NI-VISA install path. Copy them to your project, and add them into project. In the projectname.cpp file, add the follow two lines:

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

2.2 Automatic:

Set the .h file include directory, the NI-VISA install path, in our computer we set the path is : C:\Program Files\IVI Foundation \VISA\WinNT\include. Set this path to project---properties---c/c++---General---Additional Include Directories: See the picture.



USBTMC_WriteRead Property Pages

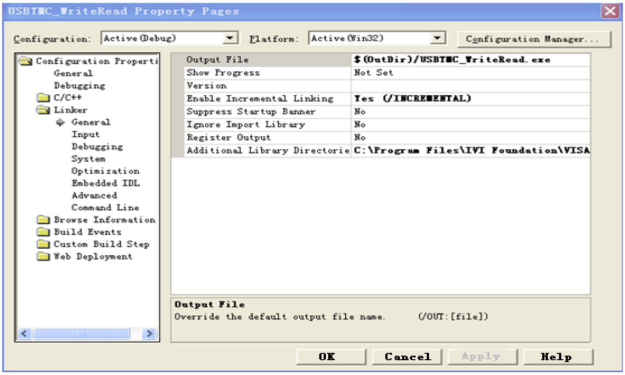
Configuration: Active (Debug)   Platform: Active (Win32)

Additional Include Directorie	C:\Program Files\IVI Foundation\VISA
Resolve #using References	
Debug Information Format	Program Database for Edit & Continue
Suppress Startup Banner	Yes (/nologo)
Warning Level	Level 3 (/W3)
Detect 64-bit Portability Iss	Yes (/Wp64)
Treat Warnings As Errors	No

Additional Include Directories
Specifies one or more directories to add to the include path; use semi-colon delimited list if more than one. (/I[path])

Set lib path set lib file:

Set lib path: the NI-VISA install path, in our computer we set the path is : C:\Program Files\IVI Foundation\VISA\WinNT \lib\msc. Set this path to project---properties---Linker---General---Additional Library Directories: as seen in the pictures below.

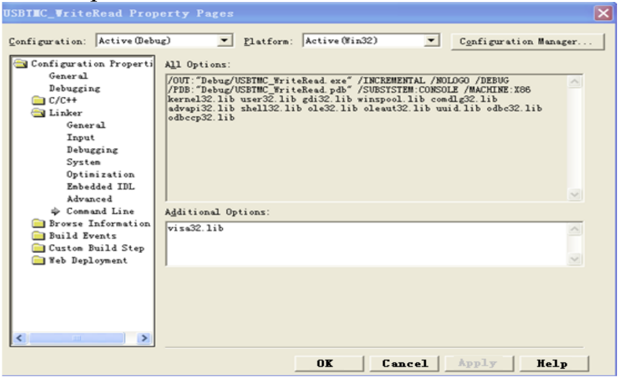


Configuration: Active(Debug)   Platform: Active(Win32)   Configuration Manager...

Output File	$(OutDir)/USBTMC_WriteRead.exe
Show Progress	Not Set
Version	
Enable Incremental Linking	Yes (/INCREMENTAL)
Suppress Startup Banner	No
Ignore Import Library	No
Register Output	No
Additional Library Directorie	C:\Program Files\IVI Foundation\VISA

Output File
Override the default output file name.   (/OUT:[file])

Set lib file:project---properties---Linker---Command Line---Additional Options: visa32.lib



Configuration: Active(Debug)   Platform: Active(Win32)   Configuration Manager...

All Options:

```
/OUT:"Debug/USBTMC_WriteRead.exe" /INCREMENTAL /NOLOGO /DEBUG
/PDB:"Debug/USBTMC_WriteRead.pdb" /SUBSYSTEM:CONSOLE /MACHINE:X86
kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib
advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib
odbccp32.lib
```

Additional Options:

```
visa32.lib
```

Include visa.h file: In the projectname.cpp file:

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

3、Add codes:

3.1 USBTMC access code:

Write a function Usbtmc_test.

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

```
ViSessioninstr;

ViUInt32numInstrs;

ViFindListfindList;

ViUInt32retCount;

ViUInt32writeCount;

ViStatusstatus;

CharinstrResourceString[VI_FIND_BUFLEN];

Unsignedcharbuffer[100];

Charstringinput[512];

Inti;

/** First we must call viOpenDefaultRM to get the manager

* handle.  We will store this handle in defaultRM.*/

status=viOpenDefaultRM (&defaultRM);

if (status<VI_SUCCESS)

{

printf ("Could not open a session to the VISA Resource
Manager!\n");

returnstatus;

}

/* 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.\nHit enter to
continue.");
fflush(stdin);
getchar();
viClose (defaultRM);
returnstatus;
}
/** 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<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,retCount, buffer);
status = viClose (instr);
}
/** Now we will close the session to the instrument using
* viClose. This operation frees all system resources.                */
status = viClose (defaultRM);
        return 0;
}
```

3.2 TCP/IP access code:

Write a function TCP_IP_Test.

```
IntTCP_IP_Test(char *pIP)
```

```
{
CharoutputBuffer[VI_FIND_BUFLEN];
ViSessiondefaultRM, instr;
ViStatusstatus;
ViUInt32count;
ViUInt16portNo;
/* 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 */
Charhead[256] ="TCPIP0::";
Chartail[] =":INSTR";
Charresource [256];
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 ("\ndata read from device: %*s\n", 0,outputBuffer);
status = viClose (instr);
status = viClose (defaultRM);
return 0;
}
```

Example of VB

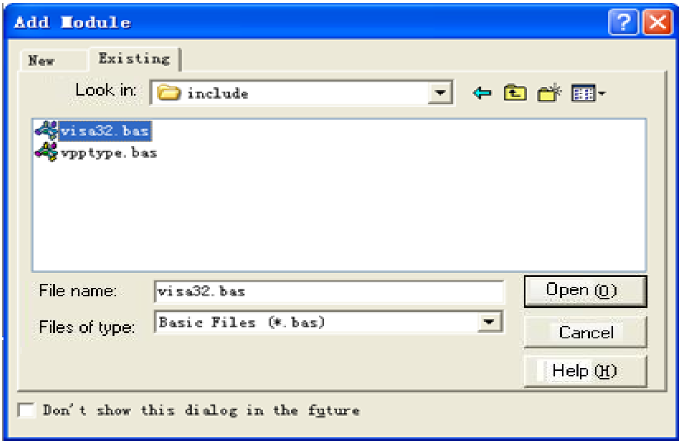
Environment: Win7 32bit system, Microsoft Visual Basic 6.0

The function of this example: Use the NI-VISA, to control the device with USBTMC and TCP/IP access to do a write and read.

Follow the steps to complete the example:

1、Open Visual Basic, build a standard application program project (Standard EXE)

2、Set the project environment to use the NI-VISA lib, Click the Existing tab of Project>>Add Module. Search for the visa32.bas file in the include folder under the NI-VISA installation path and add the file.



Look in: include

File name: visa32.bas

Files of type: Basic Files (*.bas)

□ Don't show this dialog in the future

This allows the VISA functions and VISA data types to be used in a program.

3、Add codes:

3.1、USBTMC access code:

Write a function Usbtmc_test.

```
Private Function Usbtmc_test() As Long
' 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 As Long
Dim instrsesn As Long
Dim numInstrs As Long
Dim findList As Long
Dim retCount As Long
Dim writeCount As Long
Dim status As Long
Dim instrResourceString As String * VI_FIND_BUFLEN
Dim buffer As String * MAX_CNT
Dim i As Integer
    ' 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
        Debug.Print "Could not open a session to the VISA Resource Manager!"
        Usbtmc_test = status
ExitFunction
End If

    ' 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
        Debug.Print "An error occurred while finding resources."
        viClose (defaultRM)
        Usbtmc_test = status
Exit Function
End If

 ' 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)
End If
        status = viOpen(defaultRM, instrResourceString, VI_NULL,
VI_NULL, instrsesn)
If (status < VI_SUCCESS) Then
            Debug.Print "Cannot open a session to the device ", i + 1
            GoTo NextFind
End If

        ' 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
            Debug.Print "Error writing to the device."
            status = viClose(instrsesn)
            GoTo NextFind
End If

        ' 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
            Debug.Print "Error reading a response from the device.", i + 1
Else
            Debug.Print i + 1, retCount, buffer
End If
        status = viClose(instrsesn)
NextFind:
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
End Function
```

3.2、TCP/IP access code:

Write a function TCP_IP_Test.

```
Private Function TCP_IP_Test(ip As String) As Long
Dim outputBuffer As String * VI_FIND_BUFLEN
Dim defaultRM As Long
Dim instrsesn As Long
Dim status As Long
Dim count As Long

' First we will need to open the default resource manager.
status = viOpenDefaultRM (defaultRM)
If (status < VI_SUCCESS) Then
        Debug.Print "Could not open a session to the VISA Resource
Manager!"
        TCP_IP_Test = status
        Exit Function
End If

' Now we will open a session via TCP/IP device
    status = viOpen(defaultRM, "TCPIP0::" + ip + "::INSTR",
VI_LOAD_CONFIG, VI_NULL, instrsesn)
 If (status < VI_SUCCESS) Then
        Debug.Print "An error occurred opening the session"
        viClose (defaultRM)
    TCP_IP_Test = status
Exit Function
End If


  status = viWrite(instrsesn, "*IDN?", 5, count)
 If (status < VI_SUCCESS) Then
    Debug.Print "Error writing to the device."
End If
  status = viRead(instrsesn, outputBuffer, VI_FIND_BUFLEN, count)
If (status < VI_SUCCESS) Then
    Debug.Print "Error reading a response from the device.", i + 1
Else
    Debug.Print "read from device:", outputBuffer
End If
  status = viClose(instrsesn)
  status = viClose(defaultRM)
  TCP_IP_Test = 0
End Function
```

Example of MATLAB

Environment: Win7 32bit system, MATLAB R2010b

The function of this example: Use the NI-VISA, to control the device with USBTMC and TCP/IP access to do a write and read.

Follow the steps to complete the example:

1、Open MATLAB, modify the **current directory**. In this demo, the current directory is modified to D:\USBTMC_TCPIP_Demo.

2、Click **File>>New>>Script** in the Matlab interface to create an empty M file

3、Add codes:

3.1 USBTMC access code:

Write a function Usbtmc_test.

```
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::0xF4EC::0xEE38::0123456789::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
```

3.2 TCP/IP access code:

Write a function TCP_IP_Test.

```
function TCP_IP_test( IPstr )
% 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::',IPstr,'::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
```

Example of MATLAB

Environment: Win7 32bit system, MATLAB R2010b

The function of this example: Use the NI-VISA, to control the device with USBTMC or TCP/IP access to do a write and read.

Follow the steps to complete the example:

Open MATLAB, modify the current directory. In this demo, the current directory is modified to D:\USBTMC_TCPIP_Demo.

Click File>>New>>Script in the Matlab interface to create an empty M file

Add codes:

USBTMC access code:

Write a function Usbtmc_test.

```
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::0xF4EC::0xEE38::0123456789::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
```

3.2 TCP/IP access code:

Write a function TCP_IP_Test.

```
function TCP_IP_test( IPstr )
% 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::',IPstr,'::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
```

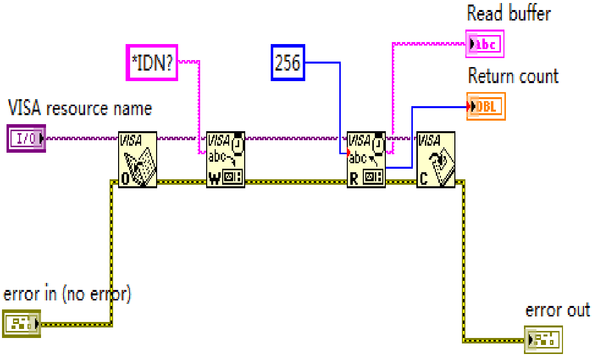
Example of LabVIEW

Environment: Win7 32bit system, LabVIEW 2011

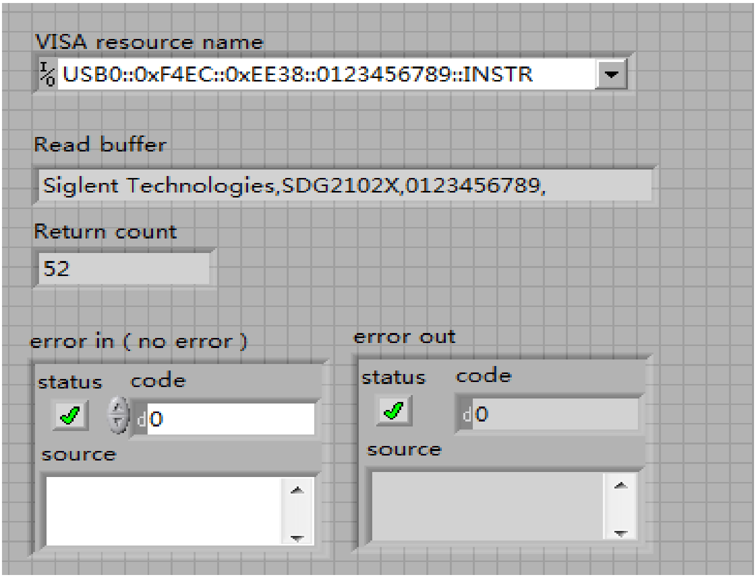
The functions of this example: use the NI-VISA, to control the device with USBTMC and TCP/IP access to do a write and read.

Follow the steps to complete the example:

1、Open LabVIEW, create a VI file.
2、Add controls. Right-click in the **Front Panel** interface, select and add **VISA resource name**, error in, error out and some indicators from the Controls column.
3、Open the **Block Diagram** interface. Right-click on the **VISA resource name** and you can select and add the following functions from VISA Palette from the pop-up menu: **VISA Write**, **VISA Read**, **VISA Open** and **VISA Close**.
4、Connect them as shown in the figure below



5、Select the device resource from the VISA Resource Name list box and run the program.



VISA resource name
USB0::0xF4EC::0xEE38::0123456789::INSTR

Read buffer
Siglent Technologies,SDG2102X,0123456789,

Return count
52

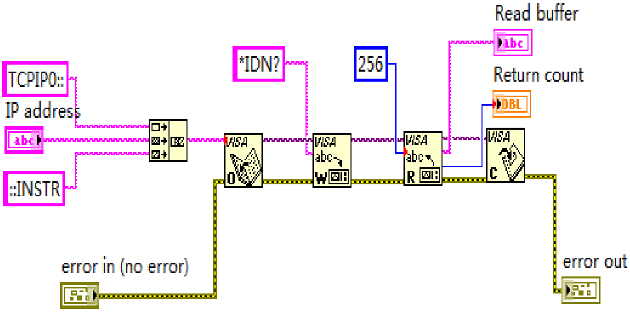
error in (no error)
status ✓ code 0
source

error out
status ✓ code 0
source

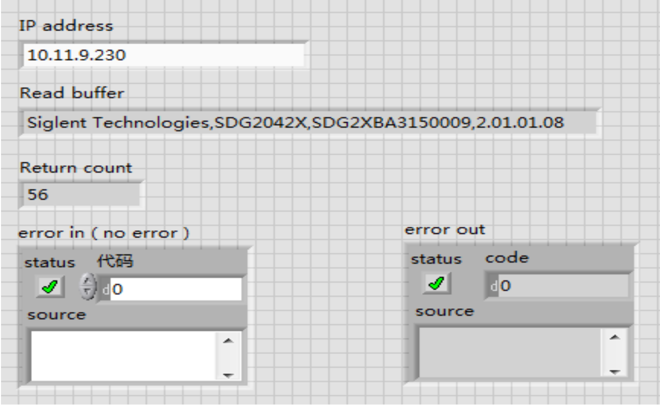
In this example, the VI opens a VISA session to a USBTMC device, writes a command to the device, and reads back the response. In this example, the specific command being sent is the device ID query. Check with your device manufacturer for the device command set. After all communication is complete, the VI closes the VISA session.

6、Communicating with the device via TCP/IP is similar to USBTMC. But you need to change VISA Write and VISA Read Function to Synchronous I/O. The LabVIEW default is asynchronous I/O. Right-click the node and select Synchronous I/O Mod>>Synchronous from the shortcut menu to write or read data synchronously.

7、Connect them as shown in the figure below



8、Input the IP address and run the program.


IP address
10.11.9.230

Read buffer
Siglent Technologies,SDG2042X,SDG2XBA3150009,2.01.01.08

Return count
56

error in (no error)
status 代码
✓ d0
source

error out
status code
✓ d0
source

Index

A
ALL_STATUS?, ALST?, Query,
ARM_ACQUISITION, ARM, Command,
ATTENUATION, ATTN,Command/Query,
AUTO_CALIBRATE, ACAL,Command/Query,
AUTO_SETUP, ASET,Command,
AUTO_TYPESET, AUTTS, Command/Query,
AVERAGE_ACQUIRE, AVGA, Command/Query,

B
BANDWIDTH_LIMIT, BWL,Command/Query,
BUZZER, BUZZ, Command,

C
CAL?, Query,
CLS, Command,
CMR?, Query,
COMM_NET, CONET,Command/Query,
COUPLING, CPL,Command/Query,
CURSOR_SET, CRST,Command/Query,
CURSOR_VALUE?, CRVA?,Query,
CURSOR_AUTO, CRAU, Command,
CSV_SAVE, CSVS, Command/Query,
COUNTER, COUN, Command/Query,
CYMOMETER, CYMT, Query,

D
DATE, Command/Query,
DDR?, Query,
DEFINE, DEF, Command/Query,
DELETE_FILE, DELF,Command,
DIRECTORY, DIR,Command/Query,
DOT_JOIN, DTJN,Command/Query,

E
ESE, Command/Query,
ESR?, Query,
EXR?, Query,

F

FILENAME, FLNM,Command/Query,
FORMAT_VDISK, FVDISK, Query,
FILTER, FILT, Command/Query,
FILT_SET, FILTS, Command/Query,
FFT_WINDOW, FFTW, Command/Query,
FFT_ZOOM, FFTZ, Command/Query,
FFT_SCALE, FFTS, Command/Query,
FFT_FULLSCREEN, FFTF, Command/Query,

G

GRID_DISPLAY, GRDS, Command/Query,
GCSV, GET_CSV, Query,

H

HARDCOPY_SETUP, HCSU,
HOR_MAGNIFY, HMAG,Command/Query,
HOR_POSITION, HPOS,Command/Query,

I

IDN?, Query,
INTENSITY, INTS,Command/Query,
INTERLEAVED, ILVD,Command/Query,
INR, INR, Query,
INVERT_SET, INVS, Command/Query,

L

LOCK, Command/Query,

M

MENU, MENU, Command/Query,
MATH_VERT_POS, MTVP, Command/Query,
MATH_VERT_DIV, MTVD, Command/Query,
MEASURE_DELY, MEAD, Command/Query,

O

OFFSET, OFST,Command/Query,
OPC, Command/Query,

P

PARAMETER_CLR, PACL,Command,
PARAMETER_CUSTOM, PACU,Command/Query,
PARAMETER_VALUE?, PAVA?,Query,

PEAK_DETECT, PDET,Command/Query,
PERSIST, PERS,Command/Query,
PERSIST_SETUP, PESU,Command/Query,
PANEL_SETUP,PNSU, Command/Query,
PF_DISPLAY, PFDS, Command/Query,
PF_SET, PFST, Command/Query,
PF_SAVELOAD, PFSL, Command,
PF_CONTROL, PFCT, Command/Query,
PF_CREATEM, PFCM, Command,
PF_DATEDIS, PFDD, Query,

R
RCL, Command,
RECALL, REC, Command,
RECALL_PANEL, RCPN,Command,
RST, Command,
REF_SET, REFS, Command/Query,

S
SAV, Command,
SCREEN_DUMP, SCDP,Command/Query,
SRE, Command/Query,
STB? Query,
STOP, Command,
STORE, STO, Command,
STORE_PANEL, STPN,Command,
STORE_SETUP, STST,Command/Query,
SAMPLE_STATUS, SAST/ Query,
SAMPLE_RATE, SARA/ Query,
SAMPLE_NUM, SANU/ Query,
SKEW, SKEW, Command,
SETTO%50, SET50, Command,
SINXX_SAMPLE, SXSA, Command/Query,

T
TIME_DIV, TDIV,Command/Query,
TRACE, TRA,Command/Query,
TRG, Command,
TRIG_COUPLING, TRCP,Command/Query,
TRIG_DELAY, TRDL,Command/Query,
TRIG_LEVEL, TRLV,Command/Query,
TRIG_MODE, TRMD,Command/Query,
TRIG_SELECT, TRSE,Command/Query,
TRIG_SLOPE, TRSL,Command/Query,

U
UNIT, UNIT, Command/Query,

V
VOLT_DIV, VDIV,Command/Query,
VERTICAL, VTCL, Command/Query,

W
WAIT, Command,
WAVEFORM,WF,Command/Query,
WAVEFORM_SETUP,WFSU,Command/Query,

X
XY_DISPLAY, XYDS, Command/Query,