**:AUToscale**

Enable the waveform auto setting function. The oscilloscope will automatically adjust the vertical scale, horizontal timebase, and trigger mode according to the input signal to realize optimum waveform display. Equivalent to pressing the AUTO key on the front panel.

**:CLEar**

Clear all the waveforms on the screen. If the oscilloscope is in the RUN state, waveform will still be displayed. Equivalent to pressing the CLEAR key on the front panel.

**:RUN, :STOP**

The run command starts the oscilloscope and the :STOP command stops the oscilloscope. Equivalent to pressing the RUN/STOP key on the front panel. When the waveform record function is enabled or during the playback of the waveform, these commands are invalid.

**:SINGle**

Set the oscilloscope to the single trigger mode. This command is equivalent to SINGLE key on the front panel and sending the :TRIGger:SWEep SINGle command. In the single trigger mode, the oscilloscope triggers once when the trigger conditions are met and then stops. Invalid in record or playback mode.

**:TFORce**

Generate a trigger signal forcefully. Only applicable to the normal and signal trigger modes and is equivalent to pressing FORCE key in the trigger control area on the front panel.

**:ACQuire Commands**

:ACQuire:AVERages 128 /\*Set the number of averages to 128\*/

:ACQuire:AVERages? /\*The query returns 128\*/

**:ACQuire:MDEPth**

Set or query the memory depth of the oscilloscope (namely the number of waveform points that can be stored in a single trigger sample). The default unit is pts (points).

:ACQuire:MDEPth 12000 /\*Set the memory depth to 12000pts\*/

:ACQuire:MDEPth? /\*The query returns 12000\*/

**:ACQuire:TYPE**

Set or query the acquisition mode of the oscilloscope.

:ACQuire:TYPE AVERages /\*Select the average acquisition mode\*/

:ACQuire:TYPE? /\*The query return AVER\*/

**:ACQuire:SRATe?**

Query the current sample rate. The default unit is Sa/s. Sample rate is the sample frequency of the oscilloscope, namely the waveform points sampled per second. Memory Depth = Sample Rate x Waveform Length.

**:CALibrate Commands**

**:CALibrate:QUIT**

Exit the self-calibration at any time.

**:CALibrate:STARt**

The oscilloscope starts to execute self-calibration. The self-calibration operation can make the oscilloscope quickly reach its optimum working state to obtain the most accurate measurement values. During the self-calibration, all the channels of the oscilloscope must be disconnected from the inputs.

**:CHANnel<n> Commands**

These are used to set or query the vertical system parameters of the analog channels, such as the bandwidth limit, coupling, vertical scale, and vertical offset.

**:CHANnel<n>:BWLimit**

Set or query the bandwidth limit parameter of the specified channel.

:CHANnel1:BWLimit 20M /\*Enable the 20MHz bandwidth limit\*/

:CHANnel1:BWLimit? /\*The query returns 20M\*/

**:CHANnel<n>:COUPling**

Set or query the coupling mode of the specified channel.

AC: the DC components of the signal under test are blocked.

DC: the DC and AC components of the signal under test can both pass the channel.

GND: the DC and AC components of the signal under test are both blocked.

:CHANnel1:COUPling AC /\*Select the AC coupling mode\*/

:CHANnel1:COUPling? /\*The query returns AC\*/

**:CHANnel<n>:DISPlay**

Enable or disable the specified channel or query the status of the specified channel.

:CHANnel1:DISPlay ON /\*Enable CH1\*/

:CHANnel1:DISPlay? /\*The query returns 1\*/

**:CHANnel<n>:INVert**

Enable or disable the waveform invert of the specified channel or query the status of the waveform invert of the specified channel.

:CHANnel1:INVert ON /\*Enable the waveform invert of CH1\*/

:CHANnel1:INVert? /\*The query returns 1\*/

**:CHANnel<n>:OFFSet**

Set or query the vertical offset of the specified channel. The defaust unit is V.

:CHANnel1:OFFSet 0.01 /\*Set the vertical offset of CH1 to 10mV\*/

:CHANnel1:OFFSet? /\*The query returns 1.000000e-02\*/

**:CHANnel<n>:RANGe**

Set or query the vertical range of the specified channel. The default unit is V.

:CHANnel1:RANGe 8 /\*Set the vertical range of CH1 to 8V\*/

:CHANnel1:RANGe? /\*The query returns 8.000000e+00\*/

**:CHANnel<n>:TCAL**

Set or query the delay calibration time of the specified channel to calibrate the zero offset of the corresponding channel. The default unit is s.

:CHANnel1:TCAL 0.00000002 /\*Set the delay calibration time to 20ns\*/

:CHANnel1:TCAL? /\*The query returns 2.000000e-08\*/

**:CHANnel<n>:SCALe**

Set or query the vertical scale of the specified channel. The default unit is V.

:CHANnel1:SCALe 1 /\*Set the vertical scale of CH1 to 1V\*/

:CHANnel1:SCALe? /\*The query returns 1.000000e+00\*/

**:CHANnel<n>:PROBe**

Set or query the probe ratio of the specified channel. Refers to multiply the signal sampled with the specified ratio and then display the result. (the actual amplitude of the signal will not be affected).

:CHANnel1:PROBe 10 /\*Set the probe ratio of CH1 to 10X\*/

:CHANnel1:PROBe? /\*The query returns 1.000000e+01\*/

**:CHANnel<n>:UNITs**

Set or query the amplitude display unit of the specified channel. VOLT, WATT, AMP, or UNKN.

:CHANnel1:UNITs VOLTage /\*Set the amplitude display unit of CH1 to V\*/

:CHANnel1:UNITs? /\*The query returns VOLT\*/

**:CHANnel<n>:VERNier**

Enable or disable the fine adjustment of the vertical scale of the specified channel, or query the fine adjustment status of the vertical scale of the specified channel.

:CHANnel1:VERNier ON /\*Enable the fine adjustment function of the vertical scale of CH1\*/

:CHANnel1:VERNier? /\*The query returns 1\*/

**:CURSor Commands**

The :CURSor commands are used to measure the X-axis value (such as time) and Y-axis value (such as voltage) of the waveform displayed on the screen.

**:CURSor:MODE**

Set or query the cursor measurement mode. OFF, MANual, TRACk, AUTO, XY.

:CURSor:MODE MANual /\*Enable the manual cursor measurement mode\*/

:CURSor:MODE? /\*The query returns MAN\*/

**:CURSor:MANual:TYPE**

Set or query the cursor type in manual cursor measurement mode. X: a vertical solid line (cursor A) and a vertical dotted line (cursor B) and are usually used to measure the time parameters. Y: a horizontal sold line (cursor A) and a horizontal dotted line (cursor B) and are usually used to measure the voltage parameters. When the channel source of manual cursor measurement is set to LA, the cursor type cannot be set to Y.

:CURSor:MANual:TYPE Y /\*select the Y type cursors\*/

:CURSor:MANual:TYPE? /\*The query returns Y\*/

**:CURSor:MANual:SOURce**

Set or query time channel source of the manual cursor measurement mode. CHAN1, CHAN2, CHAN3, CHAN4, MATH, LA

:CURSor:MANual:SOURce CHANnel2 /\*Set the channel source to CH2\*/

:CURSor:MANual:SOURce? /\*The query returns CHAN2\*/

**:CURSor:MANual:TUNit**

Set or query the horizontal unit in the manual cursor measurement mode. S, HZ, DEGR, PERC

:CURSor:MANual:TUNit DEGRee /\*Set the horizontal unit to degree\*/

:CURSor:MANual:TUNit? /\*The query returns DEGR\*/

**:CURSor:MANual:VUNit**

Set or query the vertical unit in the manual cursor measurement mode. PERC or SOUR: unit of the current source.

:CURSor:MANual:VUNit PERCent /\*Set the system to express AY, BY and BY-AY in the measurement results in percentage\*/

:CURSor:MANual:VUNit? /\*The query returns PERC\*/

**:CURSor:MANual:AX**

Set or query the horizontal position of cursor A in the manual cursor measurement mode.

The horizontal and vertical positions of the cursor are defined by the pixel coordinate of the screen. The pixel coordinate of the screen ranges from (0,0) to (600,400). (0,0) is located at the left top corner of the screen and (600,400) is located at the right bottom corner.

:CURSor:MANual:AX 200 /\*Set the horizontal position of cursor A to 200\*/

:CURSor:MANual:AX? /\*The query returns 200\*/

**:CURSor:MANual:BX**

Set or query the horizontal position of cursor B in the manual cursor measurement mode.

:CURSor:MANual:BX 200 /\*Set the horizontal position of cursor B to 200\*/

:CURSor:MANual:BX? /\*The query returns 200\*/

**:CURSor:MANual:AY**

Set or query the vertical position of cursor A in the manual cursor measurement mode.

:CURSor:MANual:AY 200 /\*Set the vertical position of cursor A to 200\*/

:CURSor:MANual:AY? /\*The query returns 200\*/

**:CURSor:MANual:BY**

Set or query the vertical position of cursor B in the manual cursor measurement mode.

:CURSor:MANual:BY 200 /\*Set the vertical position of cursor B to 200\*/

:CURSor:MANual:BY? /\*The query returns 200\*/

**:CURSor:MANual:AXValue?**

Query the X value of cursor A in the manual cursor measurement mode. The unit depends on the horizontal unit currently selected.

**:CURSor:MANual:AYValue?**

Query the Y value of cursor A in the manual cursor measurement mode. The unit depends on the horizontal unit currently selected.

**:CURSor:MANual:BXValue?**

Query the X value of cursorB in the manual cursor measurement mode. The unit depends on the horizontal unit currently selected.

**:CURSor:MANual:BYValue?**

Query the Y value of cursorB in the manual cursor measurement mode. The unit depends on the horizontal unit currently selected.

**:CURSor:MANual:XDELta?**

Query the difference between the X values of cursor A and cursor B (BX-AX) in the manual cursor measurement mode. The unit depends on the horizontal unit currently selected.

**:CURSor:MANual:IXDELta?**

Query the recriprocal of the absoulte value of the difference between the X values of cursor A and cursor B (1/|dX|) in the manual cursor measurement mode. The unit depends on the horizontal unit currently selected.

**:CURSor:MANual:YDELta?**

Query the difference between the XY values of cursor A and cursor B (BY-AY) in the manual cursor measurement mode. The unit depends on the horizontal unit currently selected.

**:CURSor:TRACk Commands**

**:CURSor:TRACk:SOURce1**

Set or query the channel source of cursor A in the track cursor measurement mode. OFF, CHAN1, CHAN2, CHAN3, CHAN4 or MATH.

:CURSor:TRACk:SOURce1 CHANnel2 /\*Set the channel source to CH2\*/

:CURSor:TRACk:SOURce1? /\*The query returns CHAN2\*/

**:CURSor:TRACk:SOURce2**

Set or query the channel source of cursor B in the track cursor measurement mode. OFF, CHAN1, CHAN2, CHAN3, CHAN4 or MATH.

:CURSor:TRACk:SOURce2 CHANnel2 /\*Set the channel source to CH2\*/

:CURSor:TRACk:SOURce2? /\*The query returns CHAN2\*/

**:CURSor:TRACk:AX**

Set or query the horizontal position of cursor A in the track cursor measurement mode.

:CURSor:TRACk:AX 200 /\*Set the horizontal position of cursor A to 200\*/

:CURSor:TRACk:AX? /\*The query returns 200\*/

**:CURSor:TRACk:BX**

Set or query the horizontal position of cursor B in the track cursor measurement mode.

:CURSor:TRACk:BX 200 /\*Set the horizontal position of cursor B to 200\*/

:CURSor:TRACk:BX? /\*The query returns 200\*/

**:CURSor:TRACk:AY**

Set or query the vertical position of cursor A in the track cursor measurement mode.

:CURSor:TRACk:AY 200 /\*Set the vertical position of cursor A to 200\*/

:CURSor:TRACk:AY? /\*The query returns 200\*/

**:CURSor:TRACk:BY**

Set or query the vertical position of cursor B in the track cursor measurement mode.

:CURSor:TRACk:BY 200 /\*Set the vertical position of cursor B to 200\*/

:CURSor:TRACk:BY? /\*The query returns 200\*/

**:CURSor:TRACk:AXValue?**

Query the X value of cursor A in the track cursor measurement mode. The default unit is s.

**:CURSor:TRACk:AYValue?**

Query the Y value of cursor A in the track cursor measurement mode. The unit is the same as the channel unit currently selected.

**:CURSor:TRACk:BXValue?**

Query the X value of cursor B in the track cursor measurement mode. The default unit is s.

**:CURSor:TRACk:BYValue?**

Query the Y value of cursor B in the track cursor measurement mode. The unit is the same as the channel unit currently selected.

**:CURSor:TRACk:XDELta?**

Query the difference between the X values of cursor A and cursor B (BX-AX) in the track cursor measurement mode. The default unit is s.

**:CURSor:TRACk:YDELta?**

Query the difference between the Y values of cursor A and cursor B (BY-AY) in the track cursor measurement mode. The unit is the same as the channel unit currently selected.

**:CURSor:TRACk:IXDELTA?**

Query the reciprocal of the absolute value of the difference between the X values of cursor A and cursor B (1/|dX|) in the track cursor measurement mode. The default unit is Hz.

**:CURSor:AUTO Commands**

**:CURSor:AUTO:ITEM**

The auto cursor function can measure 37 waveform parameters. Using this command, you can select the parameters to be measured by the auto cursor from the five parameters enabled last or query the parameters currently measured by the auto cursor.

The 37 waveform parameters: Period, Frequency, Rise Time, Fall Time, +Width, -Width, +Duty, -Duty, +Pulses, -Pulses, +Edges, -Edges, tVmax, tVmin, +Rate, +Rate, Delay 1→2, Delay 1→2, Phase 1→2, Phase 1→2, Vmax, Vmin, Vpp, Vtop, Vbase, Vamp, Vupper, Vmid, Vlower, Vavg, Vrms, Overshoot, Preshoot, Area, Period Area, Period, Vrms, Variance.

:CURSor:AUTO:ITEM ITEM3 /\*Use auto cursor to measure ITEM3\*/

:CURSor:AUTO:ITEM? /\*The query returns ITEM3\*/

**:CURSor:AUTO:AX?**

Query the horizontal position of cursor A in auto cursor measurement.

**:CURSor:AUTO:BX?**

Query the horizontal position of cursor B in auto cursor measurement.

**:CURSor:AUTO:AY?**

Query the vertical position of cursor A in auto cursor measurement.

**:CURSor:AUTO:BY?**

Query the vertical position of cursor B in auto cursor measurement.

**:CURSor:AUTO:AXValue?**

Query the X value of cursor A in auto cursor measurement. The unit depends on the horizontal unit currently selected.

**:CURSor:AUTO:AYValue?**

Query the Y value of cursor A in auto cursor measurement. The unit depends on the vertical unit currently selected.

**:CURSor:AUTO:BXValue?**

Query the X value of cursor B in auto cursor measurement. The unit depends on the horizontal unit currently selected.

**:CURSor:AUTO:BYValue?**

Query the Y value of cursor B in auto cursor measurement. The unit depends on the vertical unit currently selected.

**:CURSor:XY Commands**

The :CURSor:XY commands can only be used when the horizontal timebase mode is XY.

**:CURSor:XY:AX**

Set or query the horizontal position of cursor A in the XY cursor measurement mode.

:CURSor:XY:AX 200 /\*Set the horizontal position of cursor A to 200\*/

:CURSor:XY:AX? /\*The query returns 200\*/

**:CURSor:XY:BX**

Set or query the horizontal position of cursor B in the XY cursor measurement mode.

:CURSor:XY:BX 200 /\*Set the horizontal position of cursor B to 200\*/

:CURSor:XY:BX? /\*The query returns 200\*/

**:CURSor:XY:AY**

Set or query the vertical position of cursor A in the XY cursor measurement mode.

:CURSor:XY:AY 200 /\*Set the vertical position of cursor A to 200\*/

:CURSor:XY:AY? /\*The query returns 200\*/

**:CURSor:XY:BY**

Set or query the vertical position of cursor B in the XY cursor measurement mode.

:CURSor:XY:BY 200 /\*Set the vertical position of cursor B to 200\*/

:CURSor:XY:BY? /\*The query returns 200\*/

**:CURSor:XY:AXValue?**

Query the X value of cursor A in the XY cursor measurement mode. The unit depends on the amplitude unit of the corresponding channel.

**:CURSor:XY:AYValue?**

Query the Y value of cursor A in the XY cursor measurement mode. The unit depends on the amplitude unit of the corresponding channel.

**:CURSor:XY:BXValue?**

Query the X value of cursor B in the XY cursor measurement mode. The unit depends on the amplitude unit of the corresponding channel.

**:CURSor:XY:BYValue?**

Query the Y value of cursor B in the XY cursor measurement mode. The unit depends on the amplitude unit of the corresponding channel.

**:DECoder Commands**

The :DECoder commands are used to execute decoding settings and operations.

**:DECoder<n>:MODE**

Set or query the decoder type. PARaller, UART, SPI, and IIC correspond to parallel decoding, RS232 decoding, SPI decoding, and I2C decoding respectively.

:DECoder1:MODE SPI /\*Set the decoder type to SPI\*/

:DECoder1:MODE? /\*The query returns SPI\*/

**:DECoder<n>:DISPlay**

Turn on or off the decoder or query the status of the decoder.

:DECoder1:DISPlay ON /\*Turn on Decoder 1\*/

:DECoder1:DISPlay? /\*The query returns 1\*/

**:DECoder<n>:FORMat**

Set or query the bus display format. HEX, ASC, DEC, BIN, or LINE

:DECoder1:FORMat HEX /\*Set the bus display format to hexadecimal\*/

:DECoder1:FORMat? /\*The query returns HEX\*/

**:DECoder<n>:POSition**

Set or query the vertical position of the bus on the screen.

:DECoder1:POSition 300 /\*Set the vertical position of the bus to 300\*/

:DECoder1:POSition? /\*The query returns 300\*/

**:DECoder<n>:THREshold:CHANnel1**

**:DECoder<n>:THREshold:CHANnel2**

**:DECoder<n>:THREshold:CHANnel3**

**:DECoder<n>:THREshold:CHANnel4**

Set or query the threshold level of the specified analog channel.

:DECoder1:THREshold:CHANnel4 1.5 /\*Set the threshold level of CH4 to 1.5V\*/

**:DECoder<n>:THREshold:AUTO**

Turn on or off the auto threshold function of the analog channels, or query the status of the auto threshold function of the analog channels.

:DECoder1:THREshold:AUTO OFF /\*Turn off the auto threshold function\*/

:DECoder1:THREshold:AUTO? /\*The query returns 0\*/

**:DECoder<n>:CONFig:LABel**

Turn on or off the label display function, or query the status of the label display function.

:DECoder1:CONfig:LABel ON /\*Turn on the label display function\*/

:DECoder1:CONFig:LABel? /\*The query returns 1\*/

**:DECoder<n>:CONFig:LINE**

Turn on or off the buss display function, or query the status of the bus display function.

**:DECoder<n>:CONFig:FORMat**

Turn on or off the format display function, or query the status of the format display function.

**:DECoder<n>:CONFig:ENDian**

Turn on or off the endian display function in serial bus decoding, or query the status of the endian display function is serial bus decoding.

**:DECoder<n>:CONFig:WIDth**

Turn on or off the width display function, or query the status of the width display function.

**:DECoder<n>:CONFig:SRATe?**

Query the current digital sample rate.

**:DECoder<n>:UART Commands**

These are used to set the RS232 decoding parameters.

**:DECoder<n>:UART:TX**

Set or query the TX channel source of RS232 decoding.

:DECoder1:UART:TX CHAN2 /\*Set the TX channel source to CH2\*/

:DECoder1:UART:TX? /\*The query returns CHAN2\*/

**:DECoder<n>:UART:RX**

Set or query the RX channel source of RS232 decoding.

:DECoder1:UART:RX CHAN4 /\*Set the RX channel source to CH4\*/

:DECoder1:UART:RX? /\*The query returns CHAN4\*/

**:DECoder<n>:UART:POLarity**

Set or query the polarity of RS232 decoding. NEGative, POSitive.

**:DECoder<n>:UART:ENDian**

Set or query the endian of RS232 decoding. LSB, MSB

**:DECoder<n>:UART:BAUD**

Set or query the baud rate of RS232 decoding. The default unit is bps (baud per second). 110 to 20M. Default 9600

**:DECoder<n>:UART:WIDTh**

Set or query the width of each frame of data in RS232 decoding. 5 to 8. Default 8.

**:DECoder<n>:UART:STOP**

Set or query the stop bit after each frame of data in RS232 decoding. 1 | 1.5 | 2. Default 1

**DECoder<n>:UART:PARity**

Set or query the even-odd check mode of the data transmission in RS232 decoding. NONE, EVEN, ODD. Default NONE

**:DECoder<n>:IIC Commands**

These commands are used to set the I2C decoding parameters.

**:DECoder<n>:IIC:CLK**

Set or query the signal source of the clock channel in I2C decoding. Default CHANnel1

**:DECoder<n>:IIC:DATA**

Set or query the signal source of the data channel in I2C decoding. Default CHANnel2

**:DECoder<n>:IIC:ADDRess**

Set or query the address mode of I2C decoding. NORMal, RW. Default NORMal

**:DECoder<n>:SPI Commands**

These are used to set the SPI decoding parameters.

**:DECoder<n>:SPI:CLK**

Set or query the signal source of the clock channel in SPI decoding. Default CHANnel1.

**:DECoder<n>:SPI:MISO**

Set or query the MISO channel source in SPI decoding. Default OFF.

**:DECoder<n>:SPI:MOSI**

Set or query the MOSI channel source in SPI decoding. Default CHANnel2.

**:DECoder<n>:SPI:CS**

Set or query the CS channel source in SPI decoding. Default CHANnel3.

**:DECoder<n>:SPI:SELect**

Set or query the CS polarity in SPI decoding. NCS, CS. Default NCS.

**:DECoder<n>:SPI:MODE**

Set or query the frame synchronization mode of SPI decoding. CS, TIMeout. Default TIMeout.

**:DECoder<n>:SPI:TIMeout**

Set or query the timeout time in the timeout mode of SPI decoding. The default unit is s. The timeout time should be greater that the maximum pulse width of the clock and lower tht the idle time between frames.

**:DECoder<n>:SPI:POLarity**

Set or query the polarity of the SDA data line in SPI decoding. NEGative, POSitive. Default POSitive.

**:DECoder<n>:SPI:EDGE**

Set or query the clock type when the instrument samples the data line in SPI decoding. RISE, FALL. Default RISE.

**:DECoder<n>:SPI:ENDian**

Set or query the endian of the SPI decoding data. LSB, MSB. Default MSB.

**:DECoder<n>:SPI:WIDTh**

Set or query the number of bits of each frame of data in SPI decoding. 8 to 32. Default 8.

**:DECoder<n>:PARallel Commands**

These are used to set the parallel decoding parameters.

**:DECoder<n>:PARallel:CLK**

Set or query the CLK channel source of parallel decoding. Default CHANnel1.

**:DECoder<n>:PARallel:EDGE**

Set or query the edge type of the clock channel when the instrument samples the data channel in parallel decoding. RISE, FALL, BOTH. Default RISE.

**:DECoder<n>:PARallel:WIDTh**

Set or query the data width (namely the number of bits of each frame of data) of the parallel bus. 1 to 16. Default 8.

**:DECoder<n>:PARallel:BITX**

Set or query the data bit that requires a channel source on the parallel bus. 0 to (data width -1). Default 0.

**:DECoder<n>:PARallel:SOURce**

Set or query the channel source of the data bit currently selected. Default: related to the bit selected.

**:DECoder<n>:PARallel:POLarity**

Set or query the data polarity of parallel decoding. NEGative, POSitive. Default POSitive.

**:DECoder<n>:PARallel:NREJect**

Turn on or off the noise rejection function of parallel decoding, or query the status of the noise rejection function of parallel decoding. ON, OFF. Default OFF.

**:DECoder<n>:PARallel:NRTime**

Set or query the noise rejection time of parallel decoding. The default unit is s. 0.00s to 100ms. Default 0.00s.

**:DECoder<n>:PARallel:CCOMpensation**

Set or query the clock compensation time of parallel decoding. The default unit is s. -100ms to 100ms. Default 0.00s.

**:DECoder<n>:PARallel:PLOT**

Turn on or off the curve function of parallel decoding, or query the status of the curve function of parallel decoding. ON, OFF. Default OFF.

**:DISPlay Commands**

These can be used to set the waveform display mode, persistence time, waveform intensity, screen grid type and grid brightness.

**:DISPlay:CLEar**

Clear all the waveforms on the screen. If the oscilloscope is in the RUN state, waveform will still be displayed. This command is equivalent to pressing CLEAR key on the front panel.

**:DISPlay:DATA?**

Read the data stream of the image currently displayed on the screen and set the color, invert display, and format of the image acquired. The command is sent from the PC to the instrument through the VISA interface. The instrument responds to the command and directly returns the data stream of the image currently displayed to the buffer are of the PC. Make sure that the buffer is large enough to receive the data stream. The returned data stream contains the TMC data header which should be removed to make the data stream a standard image data stream. The terminator “\n’(0X0A) at the end of the data should be removed.

**:DISPlay:TYPE**

Set or query the display mode of the waveform on the screen. VECTors, DOTS. Default VECTors.

**:DISPlay:GRADing:TIME**

Set or query the persistence time. The default unit is s. Default MIN

**:DISPlay:WBRightness**

Set or query the waveform brightness. 0 to 100. Default 60.

**:DISPlay:GRID**

Set or query the grid type of screen display. FULL, HALF, NONE. Default FULL.

**:DISPlay:GBRightness**

Set or query the brightness of the screen grid. 0 to 100. Default 50.

**:ETABle Commands**

These commands are used to set the parameters related to the decoding event table.

**:ETABle<n>:DISP**

Turn on or off the decoding event table, or query the status of the decoding event table.

**:ETABle<n>:FORMat**

Set or query the data display format of the event table. HEX, ASCii, DEC. Default HEX

**:ETABle<n>:VIEW**

Set or query the display mode of the event table. PACKage, DETail, PAYLoad. Default PACKage.

**:ETABLe<n>:COLumn**

Set or query the current column of the event table. DATA, TX, TX, MISO, MOSI

**:ETABLe<n>:ROW**

Set or query the current row of the event table.

**:ETABLe<n>:SORT**

Set or query the display type of the decoding results in the event table. ASCend, DESCend. Default ASCend.

**:ETABLe<n>:DATA?**

Read the current event table data.

**:FUNCtion Commands**

These are used to set the waveform recording and playback parameters.

**:FUNCtion:WRECord:FEND**

Set or query the end frame of waveform recording. Default 5000.

**:FUNCtion:WRECord:FMAX?**

Query the maximum number of frames can be recorded currently.

**:FUNCtion:WRECord:FINTerval**

Set or query the time interval between frames in waveform recording. The default unit is s. 100ns to 10s. Default 100ns.

**:FUNCtion:WRECord:PROMpt**

Turn on or off the sound prompt when the recording finishes, or query the status of the sound prompt when the recording finishes.

**:FUNCtion:WRECord:OPERate**

Start or stop the waveform recording, or query the status of the waveform recording. RUN, STOP.

**:FUNCtion:WRECord:ENABle**

Turn on or off the waveform recording function, or query the status of the waveform recording function.

**:FUNCtion:WREPlay:FSTart**

Set or query the start frame of waveform playback.

**:FUNCtion:WREPlay:FEND**

Set or query the end frame of waveform playback.

**:FUNCtion:WREPlay:FMAX?**

Query the maximum number of frames can be played, namely the maximum number of frames recorded.

**:FUNCtion:WREPlay:FINTerval**

Set or query the time interval between frames in waveform playback. The default unit is s.

**:FUNCtion:WREPlay:MODE**

Set or query the waveform playback mode. REPeat, SINGLe. Default SINGLe

**:FUNCtion:WREPlay:DIRection**

Set or query the waveform playback direction. FORWard, BACKward.

**:FUNCtion:WREPlay:OPERate**

Start, pause, or stop the waveform playback, or query the status of the waveform playback. PLAY, PAUSe, STOP.

**:FUNCtion:WREPlay:FCURrent**

Set or query the current frame in waveform playback.

**IEEE488.2 Common Commands**

These are some common commands used for querying the basic information of the instrument or executing the basic operations. These commands usually start with ‘\*’ and the keyword of the command is usually 3-char long.

**\*CLS :** Clear all the event registers and clear the error queue.

**\*ESE :** Set or query the enable register for the standard event status register set.

**\*ESR? :** Query and clear the event register for the standard event status register.

**\*IDN? :** Query the ID string of the instrument.

**\*OPC :** Used to set the Operation Complete bit (bit 0) in the standard event status register to 1 after the current operation is finished. The \*OPC? Command is used to query whether the current operation is finished.

**\*RST :** Restore the instrument to the default state.

**\*SRE :** Set or query the enable register for the status byte register set.

**\*STB? :** Query the event register for the status byte register. The value of the status byte register is set to 0 after this command is executed.

**\*TST? :** Perform a self-test and then return the self-test results.

**\*WAI :** Wait for the operation to finish.

**:LAN Commands**

**:LAN:DHCP**

Turns on or off the DHCP configuration mode; or queries the on/off status of the current DHCP configuration mode.

**:LAN:AUToip**

Turns on or off the Auto IP configuration mode; or queries the on/off status of the current Auto IP configuration mode.

**:LAN:GATeway**

Sets or queries the default gateway.

**:LAN:DNS**

Sets or queries the DNS address.

**:LAN:MAC?**

Queries the MAC address of the instrument. The address format displayed on the oscilloscope is as follows: 00-19-AF-30-00-00

**:LAN:MANual**

Turns on or off the static IP configuration mode; or quesries the on/off status of the static IP configuration mode.

**:LAN:INITiate**

Initiates the network parameters. Before running the command, confirm that the oscilloscope has been connected to the network properly.

**:LAN:IPADdress**

Sets or queries the IP address of the instrument.

**:LAN:SMASk**

Sets or queries the subnet mask.

**:LAN:STATus?**

Queries the current network configuration status.

**:LAN:VISA?**

Queries the VISA address of the instrument.

**:LAN:APPLy**

Applies the network configuration.

**:MATH Commands**

These are used to set the operations between the waveforms of multiple channels.

**:MATH:DISPlay**

Enable or disable the math operation function or query the math operation status.

**:MATH:OPERator**

Set or query the operator of the math operation.

**:MATH:SOURce1**

Set or query the source or source A of algebraic operation/functional operation/the outer layer operation of compound operation.

**:MATH:SOURce2**

Set or query source B of algebraic operation/the outer layer operation of compound operation.

**:MATH:LSOUrce1**

Set or query source A of logic operation.

**:MATH:LSOUrce2**

Set or query source B of logic operation.

**:MATH:SCALe**

Set or query the vertical scale of the operation result. The unit depends on the operatior currently selected and the unit of the source.

**:MATH:OFFSet**

Set or query the vertical offset of the operation result. The unit depends on the operator currently selected and the unit of the source.

**:MATH:INVert**

Enable or disable the inverted display mode of the operation result, or query the inverted display mode status of the operation result.

**:MATH:RESet**

Sending this command, the instrument adjusts the vertical scale of the operation result to most proper value according to the current operator and the horizontal timebase of the source.

**:MATH:FFT:SOURce**

Set or query the source of FFT operation/filter.

**:MATH:FFT:WINDow**

Set or query the window function of the FFT operation. RECTangle, BLACKman, HANNing, HAMMing, FLATtop, TRIangle. Default RECTangle.

**:MATH:FFT:SPLit**

Enable or disable the half-screen display mode of the FFT operation, or query the status of the half display mode of the FFT operation.

**:MATH:FFT:UNIT**

Set or query the vertical unit of the FFT operation result.

**:MATH:FFT:HSCale**

Set or query the horizontal scale of the FFT operation result. The default unit is Hz.

**:MATH:FFT:HCENter**

Set or query the center frequency of the FFT operation result, namely the frequency relative to the horizontal center of the screen. The default unit is Hz.

**:MATH:FFT:MODE**

Set or query the FFT mode. TRACe, MEMory.

**:MATH:FILTer:TYPE**

Set or query the filter type. LPASs, HPASs, BPASs, BSTOP.

**:MATH:FILTer:W1**

Set or query the cutoff frequency of the low pass/high pass filter or cutoff frequency 1 of the band pass/band stop filter. The default unit is Hz.

**:MATH:FILTer:W2**

Set or query the cutoff frequency 2 of the band pass/band stop filter. The default unit is Hz.

**:MATH:OPTion:STARt**

Set or query the start point of the waveform math operation.

**:MATH:OPTion:END**

Set or query the end point of the waveform math operation.

**:MATH:OPTion:INVert**

Enable or disable the inverted display mode of the operation result, or query the inverted display mode status of the operation result.

**:MATH:OPTion:SENSitivity**

Set or query the sensitivity of the logic operation. The default unit is div (namely the current vertical scale).

**:MATH:OPTion:DISTance**

Set or query the smoothing window width of differential operation (diff).

**:MATH:OPTion:ASCale**

Enable or disable the auto scale setting of the operation result or query the status of the auto scale setting.

**:MATH:OPTion:THREshold1**

Set or query the threshold level of source A in logic operations. The default unit is V.

**:MATH:OPTion:THREshold2**

Set or query the threshold level of source B in logic operations. The default unit is V.

**:MATH:OPTion:FX:SOURce1**

Set or query source A of the inner layer operation of compound operation.

**:MATH:OPTion:FX:SOURce2**

Set or query source B of the inner layer operation of compound operation.

**:MATH:OPTion:FX:OPERator**

Set or query the operator of the inner layer operation of compound operation. ADD, SUBTract, MULTiply, DIVision.

**:MASK Commands**

The :MASK commands are used to set and query the pass/fail parameters.

**:MASK:ENABle**

Enable or disable the pass/fail test or query the status of the pass/fail test.

**:MASK:SOURce**

Set or query the source of the pass/fail test.

**:MASK:OPERate**

Run or stop the pass/fail test, or query the status of the pass/fail test.

**:MASK:MDISplay**

Enable or disable the statistic information when the pass/fail test is enabled, or query the status of the statistic information.

**:MASK:SOOutput**

Turn the ‘Stop on Fail’ function on or off, or query the status of the ‘Stop on Fail’ function.

**:MASK:OUTPut**

Enable or disable the sound prompt when failed waveforms are detected, or query the status of the sound prompt.

**:MASK:X**

Set or query the horizontal adjustment parameter in the pass/fail test mask. The default unit is div.

**:MASK:Y**

Set or query the vertical adjustment parameter in the pass/fail test mask. The default unit is div.

**:MASK:CREate**

Create the pass/fail test mask using the current horizontal adjustment parameter and vertical adjustment parameter.

**:MASK:PASSed?**

Query the number of passed frames in the pass/fail test.

**:MASK:FAILed?**

Query the number of failed frames in the pass/fail test.

**:MASK:TOTal?**

Query the total number of frames in the pass/fail test.

**:MASK:RESet**

Reset the numbers of passed frames and failed frames as well as the total number of frames in the pass/fail test to 0.

**:MEASure Commands**

The device supports the auto measurement of the following 37 waveform parameters and provides the statistic function for the measurement results. The :MEASure commands are used to set and query the measurement parameters.

**:MEASure:SOURce**

Set or query the source of the current measurement parameter.

**:MEASure:COUNter:SOURce**

Set or query the source of the frequency counter, or disable the frequency counter.

**:MEASure:COUNter:VALue?**

Query the measurement result of the frequency counter. The default unit is Hz.

**:MEASure:CLEar**

Clear one or all of the last five measurement items enabled.

**:MEASure:RECover**

Recover the measurement item which has been cleared.

**:MEASure:ADISplay**

Enable or disable the all measurement function, or query the status of the all measurement function.

**:MEASure:AMSource**

Set or query the source(s) of the all measurement function.

**:MEASure:SETup:MAX**

Set or query the upper limit of the threshold (expressed in the percentage of amplitude) in time, delay, and phase measurements.

**:MEASure:SETup:MID**

Set or query the middle point of the threshold (expressed in the percentage of amplitude) in time, delay, and phase measurements.

**:MEASure:SETup:MIN**

Set or query the lower limit of the threshold (expressed in the percentage of amplitude) in time, delay, and phase measurements.

**:MEASure:STATistic:DISPlay**

Enable or disable the statistic function, or query the status of the statistic function.

**:MEASure:STATistic:MODE**

Set or query the statistic mode.

**:MEASure:STATistic:RESet**

Clear the history data and make statistic again.

**:MEASure:STATistic:ITEM**

Enable the statistic function of any waveform parameter of the specified source, or query the statistic result of any waveform parameter of the specified source.

**:MEASure:ITEM**

Measure any waveform parameter of the specified source, or query the measurement result of any waveform parameter of the specified source.

**:REFerence Commands**

These are used to set the reference waveform parameters.

**:REFerence:DISPlay**

Enable or disable the REF function, or query the status of the REF function.

**:REFerence<n>:ENABle**

Enable or disable the specified reference channel, or query the status of the specified reference channel.

**:REFerence<n>:SOURce**

Set or query the source of the current reference channel.

**:REFerence<n>:VSCale**

Set or query the vertical scale of the specified reference channel. The unit is the same as the unit of the source.

**:REFerence<n>:VOFFset**

Set or query the vertical offset of the specified reference channel. The unit is the same as the unit of the source.

**:REFerence<n>:RESet**

Reset the vertical scale and vertical offset of the specified reference channel to their default values.

**:REFerence<n>:CURRent**

Select the current reference channel.

**:REFerence<n>:SAVe**

Store the waveform of the current reference channel to the internal memory as reference waveform.

**:REFerence<n>:COLor**

Set or query the display color of the current reference channel.

**:STORage Commands**

The :STORage commands are used to set the related parameters when storing images.

**:STORage:IMAGE:TYPE**

Set or query the image type when storing images. PNG, BMP8, BMP24, JPEG, TIFF.

**:STORage:IMAGe:INVERT**

Turn on or off the invert function when storing image; or query the status of the invert function.

**:STORage:IMAGe:COLor**

Set the image color when storing images to color (ON) or intensity graded color (OFF); or query the image color when storing images.

**:SYSTem Commands**

The :SYSTem commands are used to set the system related parameters, such as the sound and language.

**:SYSTem:AUTOscale**

Enable or disable the AUTO key on the front panel, or query the status of the AUTO key.

**:SYSTem:BEEPer**

Enable or disable the beeper, or query the status of the beeper.

**:SYSTem:ERRor[:NEXT]?**

Query and delete the last system error message.

**:SYSTem:GAM?**

Query the number of grids in the horizontal direction of the instrument screen.

**:SYSTem:LANGuage**

Set or query the system language.

**:SYSTem:LOCKed**

Enable or disable the keyboard lock function, or query the status of the keyboard lock function.

**:SySTem:PON**

Set or query the system configuration to be recalled when the oscilloscope is powered on again after power-off.

**:SYSTem:OPTion:INSTall**

Install a option.

**:SYSTem:OPTion:UNINSTall**

Uninstall the options installed.

**:SYSTem:RAM?**

Query the number of analog channels of the instrument.

**:SYSTem:SETup**

Import the setting parameters of the oscilloscope to restore the oscilloscope to the specified setting.

**:TIMebase Commands**

These are used to set the horizontal parameters, such as enabling the delayed sweep and setting the horizontal timebase mode.

**:TIMebase:DELay:ENABle**

Enable or disable the delayed sweep, or query the status of the delayed sweep.

**:TIMebase:DELay:OFFSet**

Set or query the delayed timebase offset. The default unit is s.

**:TIMebase:DELay:SCALe**

Set or query the delayed timebase scale. The default unit is s/div.

**:TIMebase[:MAIN]:OFFSet**

Set or query the main timebase offset. The default unit is s.

**:TIMebase[:MAIN]:SCALe**

Set or query the main timebase scale. The default unit is s/div.

**:TIMebase:MODE**

Set or query the mode of the horizontal timebase.

**:TRIGger Commands**

These are used to set the trigger system of the oscilloscope.

**:TIRGger:MODE**

Select or query the trigger type.

**:TRIGger:COUPling**

Select or query the trigger coupling type.

**:TRIGger:STATus?**

The query returns TD, WAIT, RUN, AUTO, or STOP.

**:TRIGger:SWEep**

Set or query the trigger mode.

**:TRIGger:HOLDoff**

Set or query the trigger holdoff time. The default unit is s.

**:TRIGger:NREJect**

Enable or disable noise rejection, or query the status of noise rejection.

**:TRIGger:POSition?**

Query the position in the internal memory that corresponds to the waveform trigger position.

**:TRIGger:EDGe:SOURce**

Set or query the trigger source in edge trigger.

**:TRIGger:EDGe:SLOPe**

Set or query the edge type in edge trigger.

**:TRIGger:EDGe:LEVel**

Set or query the trigger level in edge trigger. The unit is the same as the current amplitude unit of the signal source selected.

**:TRIGger:PULSe:SOURce**

Set or query the trigger source in pulse width trigger.

**:TRIGger:PULSe:WHEN**

Set or query the trigger condition in pulse width trigger.

**:TRIGger:PULSe:WIDTh**

Set or query the pulse width in pulse width trigger. The default unit is s.

**:TRIGger:PULSe:UWIDth**

Set or query the upper pulse width in pulse width trigger. The default unit is s.

**:TRIGger:PULSe:LWIDth**

Set or query the lower pulse width in pulse width trigger. The default unit is s.

**:TRIGger:PULSe:LEVel**

Set or query the trigger level in pulse width trigger. The unit is the same as the current amplitude unit.

**:TRIGger:SLOPe:SOURce**

Set or query the trigger source in slope trigger.

**:TRIGger:SLOPe:WHEN**

Set or query the trigger condition in slope trigger.

**:TRIGger:SLOPe:TIME**

Set or query the time value in slope trigger. The default unit is s.

**:TRIGger:SLOPe:TUPPer**

Set or query the upper limit of the time in slope trigger. The default unit is s.

**:TRIGger:SLOPe:TLOWer**

Set or query the lower limit of the time in slope trigger. The default unit is s.

**:TRIGger:SLOPe:WINDow**

Set or query the vertical window type in slope trigger.

**:TRIGger:SLOPe:ALEVel**

Set or query the upper limit of the trigger level in slope trigger. The unit is the same as the current amplitude unit.

**:TRIGger:SLOPe:BLEVel**

Set or query the lower limit of the trigger level in slope trigger. The unit is the same as the current amplitude unit.

**:TRIGger:VIDeo**

Select or query the trigger source in video trigger.

**:TRIGger:VIDeo:POLarity**

Select or query the video polarity in video trigger.

**:TRIGger:VIDeo:MODE**

Set or query the sync type in video trigger.

**:TRIGger:VIDeo:LINE**

Set or query the line number when the sync type in video trigger is LINE.

**:TRIGger:VIDeo:STANdard**

Set or query the video standard in video trigger.

**:TRIGger:VIDeo:LEVel**

Set or query the trigger level in video trigger. The unit is the same as the current amplitude unit.

**:TRIGger:PATTern:PATTern**

Set or query the pattern of each channel in pattern trigger.

**:TRIGger:PATTern:LEVel**

Set or query the trigger level of the specified channel in pattern trigger. The unit is the same as the current amplitude unit.

**:TRIGger:DURATion:SOURce**

Set or query the trigger source in duration trigger.

**:TRIGger:DURATion:TYPe**

Set or query the pattern of each channel in duration trigger.

**:TRIGger:DURATion:WHEN**

Set or query the trigger condition in duration trigger.

**:TRIGger:DURATion:TUPPer**

Set or query the duration time upper limit in duration trigger. The default unit is s.

**:TRIGger:DURATion:TLOWer**

Set or query the duration time lower limit in duration trigger. The default unit is s.

**:TRIGger:TIMeout:SOURce**

Set or query the trigger source in timeout trigger.

**:TRIGger:TIMeout:SLOPe**

Set or query the edge type in timeout trigger.

**:TRIGger:TIMeout:TIMe**

Set or query the timeout time in timeout trigger. The default unit is s.

**:TRIGger:RUNT:SOURce**

Set or query the trigger source in runt trigger.

**:TRIGger:RUNT:POLarity**

Set or query the pulse polarity in runt trigger.

**:TRIGger:RUNT:WHEN**

Set or query the qualifier in runt trigger.

**:TRIGger:RUNT:WUPPer**

Set or query the pulse width upper limit in runt trigger. The default unit is s.

**:TRIGger:RUNT:WLOWer**

Set or query the pulse width lower limit in runt trigger. The default unit is s.

**:TRIGger:RUNT:ALEVel**

Set or query the trigger level upper limit in runt trigger. The unit is the same as the current amplitude unit.

**:TRIGger:RUNT:BLEVel**

Set or query the trigger level lower limit in runt trigger. The unit is the same as the current amplitude unit.

**:TRIGger:WINDows:SOURce**

Set or query the trigger source in windows trigger.

**:TRIGger:WINDows:SLOPe**

Set or query the windows type in windows trigger.

**:TRIGger:WINDows:POSition**

Set or query the trigger position in windows trigger.

**:TRIGger:WINDows:TIMe**

Set or query the hold time in windows trigger.

**:TRIGger:WINDows:ALEVel**

Set or query the trigger level upper limit in windows trigger. The unit is the same as the current amplitude.

**:TRIGger:WINDows:BLEVel**

Set or query the trigger level lower limit in windows trigger. The unit is the same as the current amplitude unit.

**:TRIGger:DELay:SA**

Set or query the trigger source in delay trigger.

**:TRIGger:DELay:SLOPA**

Set or query the edge type of edge A in delay trigger.

**:TRIGger:DELay:SB**

Set or query the trigger source B in delay trigger.

**:TRIGger:DELay:SLOPB**

Set or query the edge type of edge B in delay trigger.

**:TRIGger:DELay:TYPe**

Set or query the delay type in delay trigger.

**:TRIGger:DELay:TUPPer**

Set or query the upper limit of the delay time in delay trigger. The default unit is s.

**:TRIGger:DELay:TLOWer**

Set or query the lower limit of the delay time in delay trigger. The default unit is s.

**:TRIGger:SHOLd:DSrc**

Set or query the data source in setup/hold trigger.

**:TRIGger:SHOLd:CSrc**

Set or query the clock source in setup/hold trigger.

**:TRIGger:SHOLd:SLOPe**

Set or query the edge type in setup/hold trigger.

**:TRIGger:SHOLd:PATTern**

Set or query the data type in setup/hold trigger.

**:TRIGger:SHOLd:TYPe**

Set or query the setup type in setup/hold trigger.

**:TRIGger:SHOLd:STIMe**

Set or query the setup time in setup/hold trigger. The default unit is s.

**:TRIGger:SHOLd:HTIMe**

Set or query the hold time in setup/hold trigger. The default unit is s.

**:TRIGger:NEDGe:SOURce**

Set or query the trigger source in Nth edge trigger.

**:TRIGger:NEDGe:SLOPe**

Set or query the edge type in Nth edge trigger.

**:TRIGger:NEDGe:IDLE**

Set or query the idle time in Nth edge trigger. The default unit is s.

**:TRIGger:NEDGe:EDGE**

Set or query the number of edges in Nth edge trigger.

**:TRIGger:NEDGe:LEVel**

Set or query the trigger level in Nth edge trigger. The unit is the same as the current amplitude unit.

**:WAVeform Commandsa**

These are used to read the waveform data and its related settings. :WAVeform:MODE is used to set the reading mode of the waveform data.

**:WAVeform:SOURce**

Set or query the channel of which the waveform data will be read.

**:WAVeform:FORMat**

Set or query the return format of the waveform data.

**:WAVeform:DATA?**

Read the waveform data. See the manual for detailed procedure.

**:WAVeform:XINCrement?**

Query the difference between two neighboring points of the specified channel source in the X direction.

**:WAVeform:XORigin?**

Query the start time of the waveform data of the channel source currently selected in the X direction.

**:WAVeform:XREFerence?**

Query the reference time of the specified channel source in the X direction.

**:WAVeform:YINCrement?**

Query the waveform increment of the specified channel source in the Y direction. The unit is the same as the current amplitude unit.

**:WAVeform:YORigin?**

Query the vertical offset relative to the vertical reference position of the specified channel source in the Y direction.

**:WAVeform:YREFerence?**

Query the vertical reference position of the specified channel source in the Y direction.

**:WAVeform:STARt**

Set or query the start point of waveform data reading.

**:WAVeform:STOP**

Set or query the stop point of waveform data reading.

**:WAVeform:PREamble?**

Query the return all the waveform parameters.

# VISA LIBRARY SUMMARY

**VISA:** Virtual Instrument Software Architecture.

VISA template provides all the basic resource control services to applications.

**Life Cycle Control:** VISA controls the life cycle of sessions, find lists, and events. Once an application has finished using any of them, it can use viClose() to free up all the system resources in it.

**Characteristic Control:** Resources can have attributes. The operation for modifying attributes is viSetAttribute() and the operation that retrieves it is viGetAttribute().

**Asynchronous Operation Control:** Resources can have asynchronous operations. Instead of waiting for the actual job to be done, they register the job to be done and return immediately. viTerminate() can be used to abort the operation.

**Access Control:** Applications can open multiple sessions to a VISA resource simultaneously. VISA defines a locking mechanism to restrict accesses to resources for such special circumstances. The operation used to acquire a lock on a resource is viLock(), and the operation to relinquish the lock is viUnlock().

Applications using VISA access resources by opening sessions to them. The primary method of communication to resources is by invoking operations. A VISA system also allows information exchange through events.

**viClose(vi) :** Close the specified session, event, or find list. In this process all the data structures that that had been allocated for the specified vi are freed. Returns Type ViStatus. Either a completion code or an error code.

**viGetAttribute(vi, attribute, attrState) :** Retrieve the state of an attribute. Attribute: Session, event, or find list attribute for which the state query is made. attrState: The state of the queried attribute for a specified resource. The interpretation of the returned value is defined by the individual resource.

**viSetAttribute(vi, attribute, attrState) :** Set the state of an attribute.

**viStatusDesc(vi, status, desc) :** Return a user-readable description of the status code passed to the operation.

**viTerminate(vi, degree, jobId) :** Request a VISA session to terminate normal execution of an operation. The jobId parameter is a unique value generated from each call to an asynchronous operation. If a user passes VI\_NULL as a jobId value to viTerminate(), a VISA implementation should abort any calls in the current process executing on the specified vi.

**viLock(vi, lockType, timeout, requestedKey, accessKey) :** Establish an access mode to the specified resource. This operation is used to obtain a lock on the specified resource. The caller can specify the type of lock requested-exclusive or shared lock-and the length of time the operation will suspend while waiting to acquire the lock before timing out. This operation can also be used for sharing and nesting locks.

**viUnlock(vi) :** Relinquish a lock for the specified resource.

**Event Handling and Processing**

VISA event model provides two different ways for an application to receive event notification. First is place all of the occurences of a specified event type in a session-based queue. There is one event queue per event type per session. The application can receive the event occurences later by dequeing them with the viWaitOnEvent() operation. The other method is to call the application directly, invoking a function that the application installed prior to enabling the event. A callback handler is invoked on every occurrence of the specified event.

Queuing mechanism is generally useful for non-critical events that do not need immediate servicing. The callback mechanism is useful when immediate responses are needed. They can both be enabled at the same time. The viEnableEvent() operation can be used to enable a session to respond to a specified event type either the queuing mechanism, the callback mechanism, or both.

**Queuing Mechanism**

This mechanism gives an application the flexibility to receive events only when it requests them. An application retrieves the event information by using the viWaitOnEvent() operation. Two different state: Disabled or Queuing. A session can transition between these two states using the viEnableEvent() or viDisableEvent(). The retained events can be dequeued at any time using the viWaitOnEvent() operation. An application can explicitly clear (flush) the event queue for a specified event type using the viDiscardEvents() operation.

**Callback Mechanism**

It also allows applications to install functions that can be called back when a particular event type is received. The viInstallHandler() operation can be used to install handlers to receive specified event types. The handlers are invoked on every occurrence of the specified event, once the session is enabled for the callback mechanism. One handler must be installed before a session can be enabled for sensing using the callback mechanism.

viEnableEvent(vi, eventType, mechanism, context)

viDisableEvent(vi, eventType, mechanism)

viDiscardEvents(vi, eventType, mechanism)

viWaitOnEvent(vi, inEventType, timeout, outEventType, outContext)

viInstallHandler(vi, eventType, handler, userHandle)

viUninstallHandler(vi, eventType, handler, userHandle)

**VISA Resource Management**

Includes, but not limited to, the assignment of unique resource addresses, unique resource IDs, and operation invocation. Much of the work is done by the VISA Resource Manager.

**Access:** It allows the opening of sessions to resources established on request by applications. Applications can request this service using viOpen().

**Search:** These services are used to find a resource in order to establish a communication link to it. The search is based on a description string. Applications can request this service by using the viFindRsrc() and viFindNext() operations. A session to the Default Resource Manager can be returned from the viOpenDefaultRM() function.

In order to open a session to a device resource or any other type of resource with VISA, it is essential to be able to uniquely identify a resource in the system. An address string must uniquely identify a VISA resource. The address string is used in viOpen().

**viOpenDefaultRM(sesn) :** Return a session to the Default Resource Manager resource. This function must be called before any VISA operations can be invoked. The first call to this function initializes the VISA system, including the Default Resource Manager resource, and also returns a session to that resource. Subsequent calls to this function return unique sessions to the same Default Resource Manager resource.

**viOpen(sesn, rsrcName, accessMode, timeout, vi) :** Open a session to the specified device. It returns a session identifier that can be used to call any other operations of that device.

**viParseRsrc(sesn, rsrcName, intfType, intfNum) :** Parse a resource string to get the interface information. It should succeed for all strings returned by viFindRsrc() and recognized by viOpen(). This operation is useful if you want to know what interface a given resource descriptor would use without actually opening a session to it.

**viParseRsrcEx(sesn, rsrcName, intfType, intfNum, rsrcClass, unaliasedExpandedRsrcName, aliasIfExists) :** Parse a resource string to get extended interface information. The calue returned in unaliasedExpandedRsrcName should in most cases be identical to the VISA-defined canonical resource name. The value returned in aliasIfExists allows programmatic access to user-defined aliases.

**Search Services**

VISA provides the ability to search and locate resources regardless of where the resource is residing. To be able to locate a resource in a VISA system, it is essential to be able to uniquely identify the given resource throughout the system.

**Resource Regular Expression**

A regular expression is a string consisting of ordinary characters as well as special characters. A regular expression is used for specifying patterns to match in a given string.

**viFindRsrc(sesn, expr, findList, retcnt, instrDesc) :** Query a VISA system to locate the resources associated with a specified interface. This operation matches the value specified in the expr parameter with the resources available for a particular interface. On successful completion, it returns the first resource found in the list and returns a count to indicate if there was more resources found for the designated interface. This function also returns a handle to a find list. This handle points to the list of resources and it must be used as an input to viFindNext(). When this handle is no longer needed, it should be passed to viClose().

**viFindNext(findList, instrDesc) :** Return the next resource found during a previous call to viFindRsrc().

**VISA Resource Classes**

If a VISA implementation supports the USB interface (VI\_INTF\_USB), then it shall implement the resource type INSTR.

**Instrument Control Resource**

This section describes the resource that is provided to encapsulate the various operations of a device (reading, writing, triggering and so on).

**INSTR Resource Overview**

This resource lets a controller interact with the device associated with this resource, by providing the controller with services to send blocks of data to the device, request blocks of data from the device, send the device clear command to the device, trigger the device, and find information about the device’s status.

**Basic I/O Services**

The Read service lets a controller request blocks of data from the device that is associated with this resource. How the returned data is interpreted depends on how the device has been programmed.

The Write service lets a controller send blocks of data to the device associated with this resource. The device can interpret the data as necessary.

The trigger service provides monitoring and controll access to the trigger capabilities of the device associated with the resource. Assertion of both software and hardware triggers is handled by using the viAssertTrigger() operation.

The Status/Service Request Service allows the controller to service request made by the other service requesters in a system. In this role of a service provider, it can procure the device status information. Applications can use the viReadSTB() operation to manually obtain the status information. If the resource cannot obtain the status information from the requester in the actual timeout period, timeout is returned.

The Clear service lets a controller send the device clear command to the device it is associated with, as specified by the interface regulations and the type of device.

**Formatted I/O Services**

These perform formatted and buffered I/O for devices. A formatted write operation writes to a buffer, while a formatted read operation reads from a buffer. Buffering improves system performance by making it possible to transfer large blocks of data to and from devices. The system provides separate read and write buffers that can be disabled or have their sizes modified by a user application, via the viSetBuf() operation.

**INSTR Resource Events**

This resource defines the following events for communication with applications.

**VI\_EVENT\_SERVICE\_REQ :** Notification that a service request was received from the device.

**VI\_EVENT\_VXI\_SIGP :** Notification that a VXIbus signal or VXIbus interrupt was received from the device.

**VI\_EVENT\_TRIG :** Notification that a trigger interrupt was received from the device. For VISA, the only triggers that can be sensed are VXI hardware triggers on the assertion edge (SYNC and ON trigger protocols only).

**VI\_EVENT\_IO\_COMPLETION :** Notification that an asynchronous operation has completed.

**VI\_EVENT\_VXI\_VME\_INTR :** Notification that a VXIbus interrupt was received from the device.

**VI\_EVENT\_USB\_INTR :** Notification that a vendor-specific USB interrupt was received from the device.

**VI\_EVENT\_PXI\_INTR :** Notification that a PCI Interrupt was received from the device.

**VISA Resource-Specific Operations**

**Basic I/O Services**

**viRead(vi, buf, count, retCount) :** Read data from device synchronously. The data read is to be stored in the buffer represented by buf. This operation returns only when the transfer terminates. Only one synchronous read operation can occur at any one time.

**viReadAsync(vi, buf, count, jobId) :** Read data from device asynchronously. The data read is to be stored in the buffer represented by buf. This operation normally returns before the transfer terminates. An I/O Completion event will be posted when the transfer is actually completed. The operation returns jobId, which you can use with either viTerminate() to abort the operation or with an I/O Completion event to identify which asynchronous read operation completed.

**viReadToFile(vi, fileName, count, retCount) :** Read data synchronously, and store the transferred data in a file. The file specified in fileName is opened in binary write-only mode. If the value of VI\_ATTR\_FILE\_APPEND\_EN is VI\_FALSE, any existing contents are destroyed, otherwise, the file contents are preserved. The data read is written to the file. This operation returns only when the transfer terminates. This operation is useful for storing raw data to be processed later.

**viWrite(vi, buf, count, retCount) :** Write data to device synchronously. The data to be written is in the buffer represented by buf. This operation returned only when the transfer terminates. Only one synchronous write operation can occur at any time.

**viWriteAsync(vi, buf, count, jobId) :** Write data to device asynchronously. The data to be written is in buffer represented by buf. This operation normally returns before the transfer terminates. An I/O Completion event will be posted when the transfer is actually completed. The operation returns jobId, which you can use with either viTerminate() to abort the operation or with an I/O Completion event to identify which asynchronous write operation completed.

**viWriteFromFile(vi, fileName, count, retCount) :** Take data from a file and write it out synchronously. The file specified in fileName is opened in binary read-only mode, and the data (up to end-of-file or number of bytes specified in count) is read. The data is then written to the device. This operation returns only when the transfer terminates. This operation is useful for sending data that was already processed and/or formatted.

**viAssertTrigger(vi, protocol) :** Assert software or hardware trigger dependent on the interface type. For a session to a USB instrument, this function send the TRIGGER message ID on the Bulk-OUT pipe.

**viReadSTB(vi, status) :** Read a status byte of the service request. For a session to a USB instrument, this function sends the READ\_STATUS\_BYTE command on the control pipe.

**viClear(vi) :** Clear a device. For USB INSTR sessions, VISA must send the INITIATE\_CLEAR and CHECK\_CLEAR\_STATUS commands on the control pipe.

**Formatted I/O Services**

**viSetBuf(vi, mask, size) :** Set the size for the formatted I/O and/or serial communication buffer(s). The mask parameter specifies which buffer to set the size of. The mask parameter can specify multiple buffers b bit-Oring any of the following values together.

**viFlush(vi, mask) :** Manually flush the specified buffers associated with formatted I/O operations and/or serial communication.

**viPrintf(vi, writeFmt, arg1, arg2,…) :** Convert, format, and send the parameters arg1, arg2, … to the device as specified by the format string. Before sending the data, the operation formats the arg characters in the parameter list as specified in the writeFmt string. The viWrite() operation performs the actual low-level I/O to the device. As a result, you should not use the viWrite() and viPrintf() operations in the same session.

**viVPrintf(vi, writeFmt, params) :** Convert, format, and send params to the device as specified by the format string. This operation is similar to viPrintf(), except that the ViVAList parameters list provides the parameters rather that separate arg parameters.

**viSPrintf(vi, buf, writeFmt, arg1, arg2, …) :** Same as viPrintf(), except the data is written to a user-specified buffer rather than the device.

**viVSPrintf(vi, buf, writeFmt, params) :** Same as viVPrintf(), except that the data is written to a user-specified buffer rather that a device.

**viBufWrite(vi, buf, count, retCount) :** Similar to viWrite(), except the data is written to the formatted I/O write buffer rather than directly to the device.

**viScanf(vi, readFmt, arg1, arg2, …) :** Read, convert, and format data using the format specifier. Store the formatted data in the arg1, arg2 parameters.

**viVScanf(vi, readFmt, params) :** Read, convert, and format data using the format specifier. Store the formatted data in params.

**viVSScanf(vi, buf, readFmt, params) :** Same as viVScanf(), except that the data is read from a user-specified buffer instead of a device.

**viBufRead(vi, buf, count, retCount) :** Similar to viRead(), except that the operation uses the formatted I/O read buffer for holding data read from the device. This operation can intermix with the viScanf() operation, but use with the viRead() operation is discouraged.

**viQueryf(vi, writeFmt, readFmt, arg1, arg2, …) :** Perform a formatted write and read through a single operation invocation. This operation provides a mechanism of “Send, then receive” typical to a command sequence from a commander device. It is a combination of the viPrintf() and viScanf() operations.

**viVQueryf(vi, writeFmt, readFmt, params) :** Perform a formatted write and read through a single operation invocation.

**viUsbControlOut(vi, bmRequestType, bRequest, wValue, wIndex, wLength, buf) :** Send arbitrary data to the USB device on the control port. The user must be aware of how to use each parameter based on the relevant USB base or class specification, or based on a vendor-specific request definition.

**viUsbControlIn(vi, bmRequestType, bRequest, wValue, wIndex, wLength, buf, retCnt) :** Request arbitrary data from the USB device on the control port. The user must be aware of how to use each parameter based on the relevant USB base or class specification, or based on a vendor-specific request definition.