PROGRAMMING MANUAL

Digital Storage Oscilloscope

MODEL: 2550 Series (2552, 2553, 2554, 2555, 2556, 2557, 2558, 2559)
TABLE OF CONTENTS

Using Status Registers .......................................................... 3
About these Commands & Queries ........................................ 3
  How they are listed? ................................................................ 3
  How they are described? ........................................................ 3
Command Notation .................................................................. 4
Table of Commands & Queries ............................................... 5
Commands & Queries ............................................................. 11
Index .................................................................................... 13
Using Status Registers

A wide range of status registers allows the oscilloscope’s internal processing status to be determined quickly at any time. These registers and the instrument’s status reporting system are designed to comply with IEEE 488.2 recommendations. Following an overview, starting this page, each of the registers and their roles are described.

Related functions are grouped together in common status registers. Some, such as the Status Byte Register (STB) or the Standard Event Status Register (ESR), are required by the IEEE 488.2 Standard. Other registers are device-specific, and include the Command Error Register (CMR) and Execution Error Register (EXR). Those commands associated with IEEE 488.2 mandatory status registers are preceded by an asterisk <*>.

About these Commands & Queries

This section lists and 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.
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.

sd

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:

\[ <\text{channel}> : \text{VOLT\_DIV} <\text{v\_gain}> \]
\[ <\text{channel}> : = \{\text{C1, C2, C3, C4}\} \]
\[ <\text{v\_gain}> : = 2 \text{ mV to 5 V} \]

The first line shows the formal appearance of the command, with \(<\text{channel}>\) denoting the placeholder for the header path and \(<\text{v\_gain}>\) 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 5 V.
# Table of Commands & Queries

<table>
<thead>
<tr>
<th>Short Form</th>
<th>Long Form</th>
<th>Subsystem</th>
<th>What the Command Query Does</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALST?</td>
<td>ALL_STATUS?</td>
<td>STATUS</td>
<td>Reads and clears the contents of all status registers.</td>
</tr>
<tr>
<td>ARM</td>
<td>ARM_ACQUISITION</td>
<td>ACQUISITION</td>
<td>Changes acquisition state from “stopped” to “single”.</td>
</tr>
<tr>
<td>ATTN</td>
<td>ATTENUATION</td>
<td>ACQUISITION</td>
<td>Selects the vertical attenuation factor of the probe.</td>
</tr>
<tr>
<td>ACAL</td>
<td>AUTO_CALIBRATE</td>
<td>MISCELLANEOUS</td>
<td>Enables or disables automatic calibration.</td>
</tr>
<tr>
<td>ASET</td>
<td>AUTO_SETUP</td>
<td>ACQUISITION</td>
<td>Adjusts vertical, time and trigger parameters.</td>
</tr>
<tr>
<td>AUTTS</td>
<td>AUTO_TYPESET</td>
<td>ACQUISITION</td>
<td>Selects the display type automatic setup.</td>
</tr>
<tr>
<td>AVGA</td>
<td>AVERAGE_ACQUIRE</td>
<td>ACQUISITION</td>
<td>Selects the average time of average acquisition.</td>
</tr>
<tr>
<td>BWL</td>
<td>BANDWIDTH_LIMIT</td>
<td>ACQUISITION</td>
<td>Enables/disables the bandwidth-limiting low pass filter.</td>
</tr>
<tr>
<td>BUZZZ</td>
<td>BUZZER</td>
<td>MISCELLANEOUS</td>
<td>Controls the built-in piezo-electric buzzer.</td>
</tr>
<tr>
<td>*CAL?</td>
<td>*CAL?</td>
<td>MISCELLANEOUS</td>
<td>Performs complete internal calibration of instrument.</td>
</tr>
<tr>
<td>CHDR</td>
<td>COMM_HEADER</td>
<td>COMMUNICATION</td>
<td>Controls formatting of query responses.</td>
</tr>
<tr>
<td>*CLS</td>
<td>*CLS</td>
<td>STATUS</td>
<td>Clears all status data registers.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRST?</td>
<td>CURSOR_SET?</td>
<td>CURSOR</td>
<td>Allows positioning of one of eight cursors.</td>
</tr>
<tr>
<td>CRVA?</td>
<td>CURSOR_VALUE?</td>
<td>CURSOR</td>
<td>Returns trace values measured by specified cursors.</td>
</tr>
<tr>
<td>CRAU</td>
<td>CURSOR_AUTO</td>
<td>CURSOR</td>
<td>Changes the cursor mode to auto mode.</td>
</tr>
<tr>
<td>CSVS</td>
<td>CSV_SAVE</td>
<td>SAVE/RECALL</td>
<td>Saves specified wave data of CSV format to USB device.</td>
</tr>
<tr>
<td>COUN</td>
<td>COUNTER</td>
<td>FUNCTION</td>
<td>Enables or disables the cymometer to display on the screen.</td>
</tr>
<tr>
<td>CYMT</td>
<td>CYMOMETER</td>
<td>FUNCTION</td>
<td>Returns the current cymometer value while displaying on the screen.</td>
</tr>
<tr>
<td>DATE</td>
<td>DATE</td>
<td>MISCELLANEOUS</td>
<td>Changes the date/time the internal real-time clock.</td>
</tr>
<tr>
<td>DDR?</td>
<td>DDR?</td>
<td>STATUS</td>
<td>Clears the Device Dependent Register (DDR).</td>
</tr>
<tr>
<td>DEF</td>
<td>DEFINE?</td>
<td>FUNCTION</td>
<td>Specifies math expression for function evaluation.</td>
</tr>
<tr>
<td>DELF</td>
<td>DELETE_FILE</td>
<td>MASS STORAGE</td>
<td>Deletes files from mass storage.</td>
</tr>
<tr>
<td>DIR</td>
<td>DIRECTORY</td>
<td>MASS STORAGE</td>
<td>Creates and deletes file directories.</td>
</tr>
<tr>
<td>DTJN</td>
<td>DOT_JOIN</td>
<td>DISPLAY</td>
<td>Controls the interpolation lines between data points.</td>
</tr>
<tr>
<td>*ESE</td>
<td>*ESE</td>
<td>STATUS</td>
<td>Sets the Standard Event Status Enable register (ESE).</td>
</tr>
<tr>
<td>Function</td>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>FILTS</td>
<td>FILT_SET</td>
<td>Selects the type of filter and sets the limit value filter.</td>
<td></td>
</tr>
<tr>
<td>FFTW</td>
<td>FFT_WINDOW</td>
<td>Selects the window of FFT.</td>
<td></td>
</tr>
<tr>
<td>FFTZ</td>
<td>FFT_ZOOM</td>
<td>Selects the zoom in/out times of FFT trace.</td>
<td></td>
</tr>
<tr>
<td>FFTS</td>
<td>FFT_SCALE</td>
<td>Selects the vertical scale of FFT trace.</td>
<td></td>
</tr>
<tr>
<td>FFTF</td>
<td>FFT_FULLSCREEN</td>
<td>Enables or disables to display the FFT trace screen.</td>
<td></td>
</tr>
<tr>
<td>GRDS</td>
<td>GRID_DISPLAY</td>
<td>Selects the type of grid.</td>
<td></td>
</tr>
<tr>
<td>GCSV</td>
<td>GET_CSV WAVEFORMTRANS</td>
<td>Specifies waveform data format to controller.</td>
<td></td>
</tr>
<tr>
<td>HMAG</td>
<td>HOR_MAGNIFY</td>
<td>Horizontally expands selected expansion trace.</td>
<td></td>
</tr>
<tr>
<td>HPOS</td>
<td>HOR_POSITION</td>
<td>Horizontally positions intensified zone’s center.</td>
<td></td>
</tr>
<tr>
<td>HCSU</td>
<td>HARDCOPY_SETUP</td>
<td>Configures the hard-copy driver.</td>
<td></td>
</tr>
<tr>
<td>INTS</td>
<td>INTENSITY</td>
<td>Sets the grid or trace/intensity level.</td>
<td></td>
</tr>
<tr>
<td>ILVD</td>
<td>INTERLEAVED</td>
<td>Enables/disables random interleaved sampling (RIS).</td>
<td></td>
</tr>
<tr>
<td>INR?</td>
<td>INR?</td>
<td>Reads, clears INternal state change Register (INR).</td>
<td></td>
</tr>
<tr>
<td>INVS</td>
<td>INVERT_SET</td>
<td>Invert the trace or the math waveform of specified source.</td>
<td></td>
</tr>
<tr>
<td>LOCK</td>
<td>LOCK</td>
<td>Lock keyboard.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Enables or disables to</td>
<td></td>
</tr>
<tr>
<td>Key</td>
<td>Command</td>
<td>Category</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-----------</td>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>OFST</td>
<td>OFFSET</td>
<td>ACQUISITION</td>
<td>Allows output channel vertical offset adjustment.</td>
</tr>
<tr>
<td>*OPC</td>
<td>*OPC</td>
<td>STATUS</td>
<td>Sets the OPC bit in the Event Status Register (ESR).</td>
</tr>
<tr>
<td>PACL</td>
<td>PARAMETER_CLR</td>
<td>CURSOR</td>
<td>Clears all current parameters in Custom Pass/Fail.</td>
</tr>
<tr>
<td>PACU</td>
<td>PARAMETERCUSTOM</td>
<td>CURSOR</td>
<td>Controls parameters with customizable qualifiers.</td>
</tr>
<tr>
<td>PAVA?</td>
<td>PARAMETER_VALUE?</td>
<td>CURSOR</td>
<td>Returns current parameter mask test values.</td>
</tr>
<tr>
<td>PDET</td>
<td>PEAK_DETECT</td>
<td>ACQUISITION</td>
<td>Switches the peak detector ON and OFF.</td>
</tr>
<tr>
<td>PERS</td>
<td>PERSIST</td>
<td>DISPLAY</td>
<td>Enables or disables the persistence display mode.</td>
</tr>
<tr>
<td>PESU</td>
<td>PERSIST SETUP</td>
<td>DISPLAY</td>
<td>Selects display persistence duration.</td>
</tr>
<tr>
<td>PNSU</td>
<td>PANEL SETUP</td>
<td>SAVE/RECALL</td>
<td>Complements the *SAV/*RST command.</td>
</tr>
<tr>
<td>PFDS</td>
<td>PF_DISPLAY</td>
<td>FUNCTION</td>
<td>Enables or disables to display the test and the message options of pass/fail.</td>
</tr>
<tr>
<td>PFST</td>
<td>PF_SET</td>
<td>FUNCTION</td>
<td>Sets the X mask and trigger mask setting.</td>
</tr>
<tr>
<td>PFSL</td>
<td>PF_SAVELOAD</td>
<td>SAVE/RECALL</td>
<td>Saves or recalls the created mask setting.</td>
</tr>
<tr>
<td>PFCT</td>
<td>PF_CONTROL</td>
<td>FUNCTION</td>
<td>Selects the “operate”, “output” and the “stop output” which are the options of pass/fail.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Creates the mask of the test data.</td>
</tr>
<tr>
<td>Code</td>
<td>Description</td>
<td>Action</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>---------------------------</td>
<td>---------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>RCPN</td>
<td>RECALL_PANEL</td>
<td>Save/Recall</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Recalls a front-panel setup from mass storage.</td>
<td></td>
</tr>
<tr>
<td>*RST</td>
<td>*RST</td>
<td>Save/Recall</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The *RST command initiates a device reset.</td>
<td></td>
</tr>
<tr>
<td>REFS</td>
<td>REF_SET</td>
<td>Function</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sets the reference waveform and its options.</td>
<td></td>
</tr>
<tr>
<td>*SAV</td>
<td>*SAV</td>
<td>Save/Recall</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Stores current state in volatile internal memory.</td>
<td></td>
</tr>
<tr>
<td>SCDP</td>
<td>SCREEN_DUMP</td>
<td>Hard Copy</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Causes a screen dump controller.</td>
<td></td>
</tr>
<tr>
<td>SCSV</td>
<td>SCREEN_SAVE</td>
<td>Display</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Controls the automatic screen saver.</td>
<td></td>
</tr>
<tr>
<td>*SRE</td>
<td>*SRE</td>
<td>Status</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sets the Service Request Enable register (SRE).</td>
<td></td>
</tr>
<tr>
<td>*STB?</td>
<td>*STB?</td>
<td>Status</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reads the contents of IEEE 488.</td>
<td></td>
</tr>
<tr>
<td>STOP</td>
<td>STOP</td>
<td>Acquisition</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Immediately stops signal acquisition.</td>
<td></td>
</tr>
<tr>
<td>STO</td>
<td>STORE</td>
<td>WaveformTrans</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Stores a trace in internal memory or mass storage.</td>
<td></td>
</tr>
<tr>
<td>STPN</td>
<td>STORE_PANEL</td>
<td>Save/Recall</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Stores front-panel setup in mass storage.</td>
<td></td>
</tr>
<tr>
<td>STST</td>
<td>STORE_SETUP</td>
<td>WaveformTrans</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Controls the way in which traces are stored.</td>
<td></td>
</tr>
<tr>
<td>SAST</td>
<td>SAMPLE_STATUS</td>
<td>Acquisition</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Return the acquisition status of the scope.</td>
<td></td>
</tr>
<tr>
<td>SARA</td>
<td>SAMPLE_RATE</td>
<td>Acquisition</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Return the sample rate of the scope.</td>
<td></td>
</tr>
<tr>
<td>SANU</td>
<td>SAMPLE_NUM</td>
<td>Acquisition</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Return the number of sampled points available from last acquisition at the trigger position.</td>
<td></td>
</tr>
<tr>
<td>SKEW</td>
<td>SKEW</td>
<td>Acquisition</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sets the skew of specific trace.</td>
<td></td>
</tr>
<tr>
<td>SET50</td>
<td>SETTO%50</td>
<td>Function</td>
<td></td>
</tr>
</tbody>
</table>
|      |                           | Sets the trigger level or trigger source to the center.
<table>
<thead>
<tr>
<th>Command</th>
<th>Command</th>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRCP</td>
<td>TRIG_COUPLING</td>
<td>ACQUISITION</td>
<td>Sets the coupling mode of the specified trigger source.</td>
</tr>
<tr>
<td>TRDL</td>
<td>TRIG_DELAY</td>
<td>ACQUISITION</td>
<td>Sets the time at which the trigger is to occur.</td>
</tr>
<tr>
<td>TRLV</td>
<td>TRIG_LEVEL</td>
<td>ACQUISITION</td>
<td>Adjusts the trigger level of the specified trigger source.</td>
</tr>
<tr>
<td>TRMD</td>
<td>TRIG_MODE</td>
<td>ACQUISITION</td>
<td>Selects the trigger mode.</td>
</tr>
<tr>
<td>TRSE</td>
<td>TRIG_SELECT</td>
<td>ACQUISITION</td>
<td>Sets the trigger slope of the specified trigger source.</td>
</tr>
<tr>
<td>TRSL</td>
<td>TRIG_SLOPE</td>
<td>ACQUISITION</td>
<td>Sets the unit of specification.</td>
</tr>
<tr>
<td>UNIT</td>
<td>UNIT</td>
<td>ACQUISITION</td>
<td>Adjusts the vertical position of the FFT trace.</td>
</tr>
<tr>
<td>VPOS</td>
<td>VERT_POSITION</td>
<td>DISPLAY</td>
<td>Sets the vertical sensitivity.</td>
</tr>
<tr>
<td>VDIV</td>
<td>VOLT_DIV</td>
<td>ACQUISITION</td>
<td>Enables or disables the XY display.</td>
</tr>
<tr>
<td>VTCL</td>
<td>VERTICAL</td>
<td>ACQUISITION</td>
<td>Gets the waveform from the instrument.</td>
</tr>
<tr>
<td>WF</td>
<td>WAVEFORM</td>
<td>WAVEFORMTRANS</td>
<td>Specifies amount of waveform data to go to the controller.</td>
</tr>
<tr>
<td>WFSU</td>
<td>WAVEFORM_SETUP</td>
<td>WAVEFORMTRANS</td>
<td>Prevents new analysis until current has been completed.</td>
</tr>
<tr>
<td>WAIT</td>
<td>WAIT</td>
<td>ACQUISITION</td>
<td>Enables or disables to display the XY format.</td>
</tr>
<tr>
<td>XYDS</td>
<td>XY_DISPLAY</td>
<td>DISPLAY</td>
<td></td>
</tr>
</tbody>
</table>
 Commands & Queries

**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?
**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
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**

\(<\text{channel}>: \text{ATTeNuation} <\text{attenuation}>\)

\(<\text{channel}> : = \{\text{C1, C2, C3, C4}\}\)

\(<\text{attenuation}> : = \{1, 5, 10, 50, 100, 500, 1000\}\)

**QUERY SYNTAX**

\(<\text{channel}>: \text{ATTeNuation}\?\)

**RESPONSE FORMAT**

\(<\text{channel}>: \text{ATTeNuation} <\text{attenuation}>\)

**EXAMPLE**

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

Command message:

\(\text{C1:ATTN 100}\)
**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, the *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?
The AUTO_SETUP command attempts to identify the waveform type and automatically adjusts control 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
```
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
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}

QUERY SYNTAX
AVERAGE_ACQUIRE?

RESPONSE FORMAT
AVERAGE_ACQUIRE <time>

EXAMPLE
The following turns the average times of average acquisition 16:

Command message:
AVGA 16
**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):

```
COMMAND: BandWidth_Limit C1, ON, C2, ON, C3, ON, C4, ON
```

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

```
COMMAND: BandWidth_Limit C1, ON
```
**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
The *CAL? query causes 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
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:

<table>
<thead>
<tr>
<th>COMM_HEADER</th>
<th>RESPONSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>LONG</td>
<td>C1:VOLT_DIV 200E-3 V</td>
</tr>
<tr>
<td>SHORT</td>
<td>C1:VDIV 200E-3 V</td>
</tr>
<tr>
<td>OFF</td>
<td>200E-3</td>
</tr>
</tbody>
</table>

**COMMAND SYNTAX**

Comm_HeaDeR <mode>

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

**QUERY SYNTAX**

Comm_HeaDeR?

**RESPONSE FORMAT**

Comm_HeaDeR <mode>
**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
**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)

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Unrecognized command/query header</td>
</tr>
<tr>
<td>2</td>
<td>Invalid character</td>
</tr>
<tr>
<td>3</td>
<td>Invalid separator</td>
</tr>
<tr>
<td>4</td>
<td>Missing parameter</td>
</tr>
<tr>
<td>5</td>
<td>Unrecognized keyword</td>
</tr>
<tr>
<td>6</td>
<td>String error</td>
</tr>
<tr>
<td>7</td>
<td>Parameter cannot allowed</td>
</tr>
<tr>
<td>8</td>
<td>Command String Too Long</td>
</tr>
<tr>
<td>9</td>
<td>Query cannot allowed</td>
</tr>
<tr>
<td>10</td>
<td>Missing Query mask</td>
</tr>
<tr>
<td>11</td>
<td>Invalid parameter</td>
</tr>
<tr>
<td>12</td>
<td>Parameter syntax error</td>
</tr>
<tr>
<td>13</td>
<td>Filename too long</td>
</tr>
</tbody>
</table>
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
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, GN

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**

**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.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HREL</td>
<td>Selected tract-cursor mode</td>
</tr>
<tr>
<td>VREL</td>
<td>Selected manual-cursor mode and set to voltage type</td>
</tr>
<tr>
<td>AUTO</td>
<td>Selected auto mode</td>
</tr>
<tr>
<td>OFF</td>
<td>Cursors and parameters off</td>
</tr>
</tbody>
</table>

**COMMAND SYNTAX**

CuRsor_MeaSure <mode>

<mode>={ OFF,HREL,VREL,AUTO}

**QUERY SYNTAX**

CuRsor_MeaSure?

**RESPONSE FORMAT**

CuRsor_MeaSure <mode>

**EXAMPLE**

The following command determines cursor function is turned off:

Command message:

CRMS OFF
**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.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HREF</td>
<td>The time value of curA under Track cursor mode</td>
</tr>
<tr>
<td>HDIF</td>
<td>The time value of curB under Track cursor mode</td>
</tr>
<tr>
<td>VREF</td>
<td>The volt-value of curA under manual cursor mode</td>
</tr>
<tr>
<td>VDIF</td>
<td>The volt-value of curB under manual cursor mode</td>
</tr>
<tr>
<td>TREF</td>
<td>The time value of curA under manual cursor mode</td>
</tr>
<tr>
<td>TDIF</td>
<td>The time value of curB under manual cursor mode</td>
</tr>
</tbody>
</table>

**COMMANDSYNTAX**

```
<trace>:CuRsr_SeT<cursor>,<position>[,<cursor>,<position>,<cursor> ,<position>]
<trace> = {C1, C2, C3, C4}
<cursor> = {HREF,HDF,HREF,VREF,VDIF,TREF,TDIF}
<position> = 0.1 to 11.9(or 17.9) DI (horizontal of track, the range of the
```
<cursor> := { HREF, HDIF, 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**

**DESCRIPTION**

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

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HREL</td>
<td>the cursor value under track cursor mode</td>
</tr>
<tr>
<td>VREL</td>
<td>the delta volt-value under manual cursor mode</td>
</tr>
</tbody>
</table>

**QUERY SYNTAX**

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

**RESPONSE FORMAT**

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

**EXAMPLE**

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

Command message:
C2:CRVA? VREL

Response message:
C2:CuRsor_Value VREL 1.00V

**RELATED COMMANDS**

CURSOR_SET, PARAMETER_VALUE
**CURSOR**

**DESCRIPTION**

The CURSOR_AUTO command changes the cursor mode to auto mode.

**COMMAND SYNTAX**

CRAU

**EXAMPLE**

The following code changes the cursor mode to auto mode:

Command message:
CRAU
**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 DD,<DD>,SAVE,<state>
```

The option DD is the data depth which is saved as the maximum data depth. The meaning of MAX is saved as the maximum data depth. The meaning of DIS is saved as the date depth which is displayed on the screen.

<DD>: = {MAX, DIS}

<save>: = {OFF, ON}

**QUERY SYNTAX**

```
CSV_SAVE?
```

**RESPONSE FORMAT**

```
CSV_SAVE DD, <DD>, SAVE, <state>
```

**EXAMPLE**

The following command sets the save data depth as the maximum and “para” save to off

Command message:

```
CSV_SAVE DD,MAX,SAVE,OFF
```
**FUNCTION**

**DESCRIPTION**

The COUNTER command enables or disables the cymometer display on the screen of instrument.

The response to the COUNTER? query indicates whether the cymometer is displayed on the screen of instrument.

**COMMAND SYNTAX**

COUNTER <state>

<state> : = {ON, OFF}

**QUERY SYNTAX**

COUNTER?

**RESPONSE FORMAT**

COUNTER <state>

**EXAMPLE**

The following command enables the cymometer display

Command message:

COUN ON
FUNCTION CYMOMETER, CYMT

QUERY SYNTAX

CYMOMETER?

RESPONSE FORMAT

CYMOMETER <option>

EXAMPLE

The following instruction returns the value of the counter which displays on the screen of the instrument.

Response message:

CYMT 10Hz
**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**

**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**

**RESPONSE FORMAT**

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

**EXAMPLE**
The following instruction reads the contents of the DDR register:

Command message:
DD?

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

<table>
<thead>
<tr>
<th>Function Equations</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;source1&gt; + &lt;source2&gt;</td>
</tr>
<tr>
<td>&lt;source1&gt; - &lt;source2&gt;</td>
</tr>
<tr>
<td>&lt;source1&gt;*&lt;source2&gt;</td>
</tr>
<tr>
<td>&lt;source1&gt;/&lt;source2&gt;</td>
</tr>
<tr>
<td>FFT(source x)</td>
</tr>
</tbody>
</table>

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 be up to eight characters.

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

Command message:
DELF DISK, UDSK, FILE, ‘/SETUP/001.SE’

**RELATED COMMANDS**
DIRECTORY
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 on a USB memory device:

Command message:

```
DIR? DISK, UDSK
```

Response message:

```
DIRectory DISK, UDSK,"A:
```
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**

**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_\text{= 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**

**DESCRIPTION**

The *ESR?* query reads and clears the content 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)

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

The EXR? query reads and clears the content 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)

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

**DESCRIPTION**

Causes the instrument to make one acquisition.

**COMMAND SYNTAX**

FoRce_TRigger

**EXAMPLE**

Either of the following pairs of instruction make one acquisition:

Command message1:
TRMD SINGLE;ARM;FRTR

Command message2:
TRMD STOP;ARM;FRTR
**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
## DESCRIPTION

The FILTER command enables or disables filter on the specified trace.

The response to the FILTER? query indicates whether the filter of specified trace is enabled.

### COMMAND SYNTAX

```plaintext
<channel>:FILTER <state>
<channel> : = {C1,C2,C3,C4}
<state> : = {ON,OFF}
```

### QUERY SYNTAX

```plaintext
<channel>:FILTER?
```

### RESPONSE FORMAT

```plaintext
<channel>:FILTER <state>
```

### EXAMPLE

The following command enables the filter of channel 1:

Command message:
C1:FILT ON

### RELATED COMMANDS

FILTS
**FUNCTION**

**FILT_SET, FILTS**  
Command /Query

**DESCRIPTION**  
The FILT_SET command selects the specified type of filter, and sets the limit value of filter. The response to the FILT_SET? query indicates current parameter of the filter.

**COMMAND SYNTAX**  
```
<channel>: FILT_SET TYPE,<type>,<limit>,<limit_value>
```

- `<channel>` : = {C1,C2,C3,C4}
- `<type>` : = {LP,HP,BP,BR}
- LP is lowpass, HP is highpass, BP is bandpass, BR is bandreject  
- `<limit>` : = {UPPLIMIT,LOWLIMIT}  
- if seted the `<limit>`,the `<type>` must be related

**QUERY SYNTAX**  
```
<channel>: FILT_SET?
```

**RESPONSE FORMAT**  
```
<channel>:FILTER TYPE,<type>,<limit>,<limit_value>
```

**EXAMPLE**  
The following command changes the type of filter to bandpass, and sets the upplimit to 200 KHz and the lowlimit to 100 KHz:

Command message:  
C1:FILTS TYPE,BP,UPPLIMIT,200KHz,LOWLIMIT,100KHz
The FFT_WINDOW command selects the window of FFT (Fast Fourier Transform algorithm).

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

COMMAND SYNTAX
FFT_SCALE <scale>
<s 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 whether 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
**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
DESCRIPTION
indicates current waveform of CSV format.

The response to the GET_CSV? query

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

QUERY SYNTAX
GET_CSV? DD,<DD>,SAVE,<state>

The option DD is the data depth of the CSV format waveform. The option SAVE is that if the waveform data have parameters.

<DD>: = {MAX, DIS} the meaning of MAX is that the CSV waveform’s depth is maximum. The meaning of DIS is that CSV waveform’s depth is the data which is displayed on the screen.

<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 the maximum depth of waveform data with parameters information.

Command message:
GET_CSV? DD,MAX,SAVE,ON
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 50,000,000 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:

```
DISPLAY
HOR_MAGNIFY, HMAG
Command /Query
```
The HOR_POSITION command horizontally positions the geometric center of the intensified zone on the source trace. Allowed positions range from division -9 to 9. 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>: = -9 to 9 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 source trace.
**HARD COPY**

**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
*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
B&K Precision, 2553,SN#, 3.01.01.22
**DISPLAY**

**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 while 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 level through the INTENSITY command:

`DISPLAY INTENSITY, INTS`
**ACQUISITION**

**INTERLEAVED, ILVD**

**Command /Query**

**DESCRIPTION**

The INTERLEAVED command enables or disables random interleaved sampling (RIS) for timebase settings where both single shot and RIS mode are available.

The response to the INTERLEAVED? Query indicates whether the oscilloscope is in RIS mode.

**COMMAND SYNTAX**

InterLeaVeD <mode>

<mode> : = {ON, OFF}

**QUERY SYNTAX**

InterLeaVeD?

**RESPONSE FORMAT**

InterLeaVeD <mode>

**EXAMPLE**

The following instructs the oscilloscope to use RIS mode:

Command message:
ILVD ON

**RELATED COMMANDS**

TIME_DIV, TRIG_MODE
STATUS

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 1 bit.

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

The response to the INVERTSET? query indicates whether the specified waveform 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
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**

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

**QUERY SYNTAX**

```plaintext
LOCK?
```

**RESPONSE FORMAT**

```plaintext
LOCK < status >
```

**EXAMPLE**

The following instruction enables the functions of the panel keys:

Command message:

```
LOCK ON
```
**DESCRIPTION**

The MENU command enables or disables the display of the menu.

The response to the MENU? query indicates whether the menu is displayed.

**COMMAND SYNTAX**

```
MENU <status>
```

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

**QUERY SYNTAX**

```
MENU?
```

**RESPONSE FORMAT**

```
MENU <status>
```

**EXAMPLE**

The following instruction enables the display of the menu:

Command message:
```
MENU 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 25. The math waveform will be displayed 1 grid up to the vertical center of the screen. Namely one grid is 25.

**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 25
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**

**MEASURE_DELY, MEAD**

Command / Query

**DESCRIPTION**
The MEASURE_DELY command selects the type of delay measure.

The response to the MEASURE_DELY? query indicates the type of delay measure.

**COMMAND SYNTAX**

MEASURE_DELY
SOURCE,<mode>,TYPE,<type>

<mode>::= {C1-C2, C1-C3, C1-C4, C2-C3, C2-C4, C3-C4}

<type>::= {PHA,FRR,FRF,FFF,LRR,LRF,LFR,LFF},
The PHA is phase, the others are the same as the specified type of the instrument’s delay measure

**QUERY SYNTAX**

MEASURE_DELY?

**RESPONSE FORMAT**

MEASURE_DELY
SOURCE,<mode>,TYPE,<type>

**EXAMPLE**
The following instruction sets the type of delay measure to phase between C1 and C2.

Command message:
MEAD SOURCE,C1-C2,TYPE,PHA
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>: OFST <offset>
<channel> : = {C1, C2, C3,C4}
<offset> : = See specifications.

QUERY SYNTAX
<channel>: OFST?

RESPONSE FORMAT
<channel>: OFST <offset>

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

Command message:
C2: OFST -3V
**STATUS**

**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**

**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
**PARAMETER_CLR, PACL**

**Command**

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

**COMMAND SYNTAX**

```plaintext
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 }
<qualifier> := 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, C
Query/Response Examples PACU? 2 returns PACU 2, PKPK, C1
PAVA? CUST2 returns:
C2: PAVA CUST2, 160.00mV

**RELATED COMMANDS**

COMMENTS PARAMETER_CLR, CURSOR
DESCRIPTION

The PARAMETER_VALUE query returns the measurement values.

Parameters Available on All Models

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

QUERY SYNTAX

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

RESPONSE FORMAT

PA VAl
CURSOR
PARAMETER_VALUE?, PAVA?
RELATED COMMANDS

CURSOR_MEASURE, CURSOR_SET,
PARAMETER_CUSTOM
**ACQUISITION**

**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
```
**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
**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, 2, 5, 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 oscilloscope’s current panel setup in 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**

**DESCRIPTION**
The PF_SAVELOAD command saves or recall the created mask setting.

**COMMAND SYNTAX**

```plaintext
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 SYNTAX
PF_DATADIS?

RESPONSE FORMAT
PF_DATADIS
FAIL,<num>,PASS,<num>,total,<num>

EXAMPLE
The following instruction returns the number of the fail, pass and total number that the screen is showing:

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

RELATED COMMANDS
PACL
**SAVE/RECALL SETUP**

**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 EEXECution 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
**WAVEFORM TRANSFER**

**RECALL, REC Command**

**DESCRIPTION**

The RECALL command recalls a waveform file from the current directory on mass storage into any or all of the internal memories M1 to M10 (or M20 in the CFL series).

**COMMAND SYNTAX**

<memory>: RECall DISK, <device>, FILE, ‘<filename>’

- <memory> := {M1~M10} (or M1~M20 in the CFL series)
- <device> := {UDSK}
- <filename> := A waveform file under a legal DOS path. A filename-string of up to eight characters, with the extension “.DAV”. (This can include the ‘/’ character to define the root directory.)

**EXAMPLE**

The following recalls a waveform file called “C1WF.DAV” from the memory card into Memory M1:

Command message:

M1: REC DISK, UDSK FILE, ‘C1WF.DAV’

**RELATED COMMANDS**

STORE, INR?
**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**

<table>
<thead>
<tr>
<th>Description</th>
<th>The *RST command initiates a device reset. The *RST sets recalls the default setup.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>COMMAND SYNTAX</strong></td>
<td>*RST</td>
</tr>
<tr>
<td><strong>EXAMPLE</strong></td>
<td>This example resets the oscilloscope:</td>
</tr>
<tr>
<td>Command message:</td>
<td>*RST</td>
</tr>
<tr>
<td><strong>RELATED COMMANDS</strong></td>
<td>*CAL, *RCL</td>
</tr>
</tbody>
</table>
**FUNCTION**

**REF_SET, REFS**

**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,C1OFF,C2OFF,C3OFF,C4OFF}
If the trace is closed, the specified trace will be CxOFF, (x is 1,2,3,4), the closed trace couldn’t be saved or set
<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:
**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>

<p>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
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
**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 save:

Command message:

SCSV YES
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) is set in the STB register:

*SRE 17

STATUS
*SRE
Command/Query
**STATUS**

**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)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Value</th>
<th>Bit Name</th>
<th>Description</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>128</td>
<td>DIO7</td>
<td>0 reserved for future use</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>MSS/RQS</td>
<td>at least 1 bit in STB masked by SRE is 1 service is requested</td>
<td>(1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MSS=1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>RQS=1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>ESB</td>
<td>1 an ESR enabled event has occurred</td>
<td>(3)</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>MAV</td>
<td>1 output queue is not empty</td>
<td>(4)</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>DIO3</td>
<td>0 reserved</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>VAB</td>
<td>1 a command data value has been adapted</td>
<td>(5)</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>DIO1</td>
<td>0 reserved</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>INB</td>
<td>1 an enabled INternal state change has occurred</td>
<td>(6)</td>
</tr>
</tbody>
</table>

### 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.
2. Bit position 6 depends on the polling method:
   - Bit 6 = MSS if an *STB? Query is received
   - = RQS if serial polling is conducted
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.
ACQUISITION

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
**STORE, STO**

**Command**

**DESCRIPTION**

The STORE command stores the contents of the specified trace into one of the internal memories M1 to M10 (or M20 in the CFL series) or to the current directory in a USB memory device.

**COMMAND SYNTAX**

\[ \text{STOre } [<\text{trace}>], [<\text{dest}>] \]

\(<\text{trace}>: = \{TA, TB, TC, TD, C1, C2, C3, C4, ALL\_DISPLAYED}\)

\(<\text{dest}>: = \{M1\sim M10 (or M20 in the CFL series), UDSK}\)

Note: If the STORE command is sent without an 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 Memory 1 (M1):

Command message:
STO C1, M1

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

Command message:
STO ALL\_DISPLAYED, UDSK
**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.SET'

**RELATED COMMANDS**

*SAV, RECALL_PANEL, *RCL
**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**

```plaintext
STore_Setup [<trace>, <dest>]
<trace> := {C1,C2,C3,C4,ALL_DISPLAYED}
<dest> := {M1~M10(or M20 in the CFL series), UDSK}
```

**QUERY SYNTAX**

```plaintext
STore_Setup?
```

**RESPONSE FORMAT**

```plaintext
STore_Setup <trace>, <dest>
```

**EXAMPLE**

The following command selects Channel 1 to be stored.

Command message:

STST C1, UDSK

**RELATED COMMANDS**

STORE, INR
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
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
**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
The SET50 command sets the trigger level of the specified trigger source to the centre of the signal amplitude.

**EXAMPLE**

The following command sets the trigger level of the specified trigger source to the centre of the signal amplitude.

Command message:

SET50
**ACQUISITION**

**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
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(not every type has this value),2.5NS,5NS,10NS,25NS,50NS,100NS,250NS,500NS,1US,2.5US,5US,10US,25US,50US,100US,250US,500US,1MS,2.5MS,5MS,10MS,25MS,50MS,100MS,250MS,500MS,1S,2.5S,5S,10S,25S,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
**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
**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**

\[ <\text{trace}>: \text{TRAce} <\text{mode}> \]

\[ <\text{trace}> : = \{C1, C2, C3, C4, TA, TB, TC, TD\} \]

\[ <\text{mode}> : = \{ON, OFF\} \]

**QUERY SYNTAX**

\[ <\text{trace}>: \text{TRAce}? \]

**RESPONSE FORMAT**

\[ <\text{trace}>: \text{TRAce} <\text{mode}> \]

**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
**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 3) 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):

ACQUISITION
**ACQUISITION**

**TRIG_LEVEL, TRLV**

**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>: = -6DIV* volt/div to 6DIV * 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_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...
The TRIG_SELECT command is used to set the trigger type and the type’s option HT which is an option of the TRIG_SELECT command is related to the TRSL command. The TRSL command could set the <trig_slope>. The HT’s polarity will also be changed.

The TRIG_SELECT? query returns the current trigger type.

**COMMAND SYNTAX**

TRig_SelEct <trig_type>,SR,<source>,HT,<hold_type>,HV,<hold_value>

TRig_SelEct<trig_type>,SR,<source>,CHAR,<characteristicse>,POL,<polarity>,SYNC,<sync type>,LINE,<line>

TRig_SelEct INTV,SR,<source>,VERT,<vertical>

**OPTION**

<trig_type>: = {EDGE, GLIT, INTV, TV,}
GLIT means pulse trigger, INTV means slope trigger and TV means video trigger.

**Options:** SR HT HV POL CHAR SYNC LINE VERT

HT,<hold_type> is used to set pulse type.

<hold_type> := {TI, PS, PL, PE, IS, IL, IE}
TI means holdoff, PS means that the pulse width is smaller than the set value. PL means that the pulse width is larger than the set value. PE means that the pulse width is equal with the set value. I
SR, `<source>`: is used to set the trigger’s channel. If you want to set the other option, you must set it.
`<source>`: = {C1, C2, C3, C4, EX, EX5}

CHAR, `<characteristics>`: is used to set the standard. If you want to set it, the `<trig_type>` must be set to TV.
`<characteristics>`: = {NTSC, PALSEC}

SYNC, `<sync_type>`: is used to set sync. If you want to set it, you must set `<trig_type>` to TV.
`<sync_type>`: = {AL, LN, OF, EF}
AL means all lines; LN means line num; OF means odd field; EF means even field

LINE, `<line>`: is used to set the line num. If you want to set it, the SYNC must be set to LINENUM.

POL, `<polarity>`: is used to set polarity. If you want to set it, you must set `<trig_type>` to TV.
`<polarity>`: = {PO, NE}
PO means positive. NE means negative.

VERT, `<vertical>`: is used to set vertical. If you want to set it, you must set `<trig_type>` to INTV.
`<vertical>`: = {UP, DOWN, BOTH}

**QUERY SYNTAX**

```
TRig_SelEct?
```

**RESPONSE FORMAT**

```
TRig_SelEct <mode>, the other options
```

**EXAMPLE**

The following sets the trigger type to video, the trigger source to C1, the standard to NTSC, the polarity to positive, the sync to line num, and the vertical to both.

```
TRig_SelEct Video, C1, NTSC, PO, LN, BOTH
```
ACQUISITION

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, LINE}
<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

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
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

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
**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 (or 5V in the CFL series)

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
**ACQUISITION**

**DESCRIPTION**

The VERTICAL command controls the vertical position of the slope trigger line. It is related to the TRSE command. The VERT option of the TRSE command changes the controlling type of the slopes trigger line.

When the slope trigger lines are both controlled, the vertical position of the slope trigger line is the sum up one’s position.

The VERTICAL query returns the vertical position of the slope trigger line.

**COMMAND SYNTAX**

```
<channel>: VERTICAL <pos>
```

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

<pos>: = the position is related to the screen vertical center. For example, if we set the vertical position of the slope trigger line to 25, it will be displayed 1 grid up to the screen vertical center. Namely one grid is 25.

**QUERY SYNTAX**

```
<channel> : VERTICAL ?
```

**RESPONSE FORMAT**

```
<channel>: VERTICAL <pos>
```

**EXAMPLE**

The following command sets the vertical position of the slope trigger line to 25 that what is the distance from the up of centre about 1 grid:

Command message:
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 user text (TEXT)
3. the time (TIME) descriptor
4. the data (DAT1) block, and, optionally,
5. a second block of data (DAT2)

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: The format of the waveform data depends on the current settings specified by the last WAVEFORM_SETUP command.

**QUERY SYNTAX**

```
<trace>: WaveForm?
<trace> : = { C1,C2,C3,C4}
```
Note:
Offset data factor is a 4 byte floating point number starting at address 0xA0.
Amplitude scale factor data is a 4 byte floating point number starting at address 0x9C.
Waveform descriptor block starts off from “WAVEDESC” in the return data. The size of the
descriptor is 0x16e – 0x15 + 1.
All waveform data are represented in two’s complement binary. It must be converted to
decimal and apply to the linear equation formula \( y = mx + b \), where \( x \) is the data in decimal
value, \( m \) is the amplitude scale factor, and \( b \) is the offset data factor.

For detailed description, see the end of the document.
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.

Note: This command currently only support NP (Number of points).

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FP</td>
<td>first point</td>
</tr>
<tr>
<td>SP</td>
<td>sparsing</td>
</tr>
<tr>
<td>NP</td>
<td>number of points</td>
</tr>
</tbody>
</table>

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 waveform.
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

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 to 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
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,
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,
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,
PFDATEDIS, 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,
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,
Explanation of the formats of waveforms and their descriptors on the DSO

A descriptor and/or a waveform consists of one or several logical data blocks whose formats are explained below.

Usually, complete waveforms are read: at the minimum they consist of
the basic descriptor block WAVEDESC
a data array block.

Some more complex waveforms, e.g. Extrema data or the results of a Fourier transform, may contain several data array blocks.

When there are more blocks, they are in the following sequence:
the basic descriptor block WAVEDESC
auxiliary data array block
data array block

In the following explanation, every element of a block is described by a single line in the form

<byte position>   <variable name>: <variable type> ; <comment>

where

<byte position> = position in bytes (decimal offset) of the variable, relative to the beginning of the block.

<variable name> = name of the variable.

<variable type> = string up to 16-character name
                   terminated with a null byte
byte 08-bit signed data value
word 16-bit signed data value
long 32-bit signed data value
float 32-bit IEEE floating point value
                   with the format shown below
                   31  30 .. 23   22 ... 0   bit position
                   s   exponent   fraction
                   where
                   s = sign of the fraction
                   exponent = 8 bit exponent e
                   fraction = 23 bit fraction f
                   and the final value is
                   \((-1)**s * 2**(e-127) * 1.f\)
double 64-bit IEEE floating point value
                   with the format shown below
                   63  62 .. 52   51 ... 0   bit position
                   s   exponent   fraction
                   where
                   s = sign of the fraction
                   exponent = 11 bit exponent e
                   fraction = 52 bit fraction f
                   and the final value is
                   \((-1)**s * 2**(e-1023) * 1.f\)
enum enumerated value in the range 0 to N
                   represented as a 16-bit data value.
                   The list of values follows immediately.
                   The integer is preceded by an _.
time_stamp double precision floating point number, for the number of seconds and some bytes for minutes, hours, days, months and year.
                   double seconds  (0 to 59)
                   byte minutes    (0 to 59)
                   byte hours      (0 to 23)
byte  days    (1 to 31)
byte  months (1 to 12)
word  year    (0 to 16000)
word  unused

There are 16 bytes in a time field.

data byte, word or float, depending on the read-out mode reflected by the WAVEDESC variable COMM_TYPE, modifiable via the remote command COMM_FORMAT.

text arbitrary length text string (maximum 160)

unit_definition a unit definition consists of a 48 character ASCII string terminated with a null byte for the unit name.

==========================================================================

DESC: BLOCK

Explanation of the wave descriptor block WAVEDESC;

< 0>   DESCRIPTOR_NAME: string ; the first 8 chars are always WAVEDESC
< 16>  TEMPLATE_NAME: string
< 32>  COMM_TYPE: enum          ; chosen by remote command COMM_FORMAT
_0      byte
_1      word
endenum
< 34>  COMM_ORDER: enum
_0      HIFIRST
_1      LOFIRST
endenum

The following variables of this basic wave descriptor block specify the block lengths of all blocks of which the entire waveform (as it is currently being read) is composed. If a block length is zero, this block is (currently) not present.

Blocks and arrays that are present will be found in the same order as their descriptions below.

BLOCKS :
< 36>  WAVE_DESCRIPTOR: long    ; length in bytes of block WAVEDESC
< 40>  USER_TEXT: long          ; length in bytes of block USERTEXT
< 44>  RES_DESC1: long          ;

ARRAYS :
< 48>  TRIGTIME_ARRAY: long     ; length in bytes of TRIGTIME array
< 52>  RIS_TIME_ARRAY: long     ; length in bytes of RIS_TIME array
< 56>  RES_ARRAY1: long         ; an expansion entry is reserved
< 60>  WAVE_ARRAY_1: long       ; length in bytes of 1st simple data array. In transmitted waveform, represent the number of transmitted bytes in accordance with the NP parameter of the WFSU remote command and the used format (see COMM_TYPE).
< 64>  WAVE_ARRAY_2: long       ; length in bytes of 2nd simple data array
RES_ARRAY2: long
RES_ARRAY3: long ; 2 expansion entries are reserved

The following variables identify the instrument

INSTRUMENT_NAME: string
INSTRUMENT_NUMBER: long
TRACE_LABEL: string ; identifies the waveform.
RESERVED1: word
RESERVED2: word ; 2 expansion entries

The following variables describe the waveform and the time at which the waveform was generated.

WAVE_ARRAY_COUNT: long ; number of data points in the data array. If there are two data arrays (FFT or Extrema), this number applies to each array separately.
PNTS_PER_SCREEN: long ; nominal number of data points on the screen
FIRST_VALID_PNT: long ; count of number of points to skip before first good point
FIRST_VALID_POINT = 0 ; for normal waveforms.
LAST_VALID_PNT: long ; index of last good data point in record before padding (blanking) was started.
LAST_VALID_POINT = WAVE_ARRAY_COUNT - 1 ; except for aborted sequence and rollmode acquisitions
FIRST_POINT: long ; for input and output, indicates the offset relative to the beginning of the trace buffer.
Value is the same as the FP parameter of the WFSU remote command.
SPARSING_FACTOR: long ; for input and output, indicates the sparsing into the transmitted data block.
Value is the same as the SP parameter of the WFSU remote command.
SEGMENT_INDEX: long ; for input and output, indicates the index of the transmitted segment.
Value is the same as the SN parameter of the WFSU remote command.
SUBARRAY_COUNT: long ; for Sequence, acquired segment count, between 0 and NOM_SUBARRAY_COUNT
SWEEPS_PER_ACQ: long ; for Average or Extrema, number of sweeps accumulated else 1
POINTS_PER_PAIR: word ; for Peak Detect waveforms (which always include data points in DATA_ARRAY_1 and min/max pairs in DATA_ARRAY_2).
Value is the number of data points for
PAIR_OFFSET: word ; for Peak Detect waveforms only
; Value is the number of data points by
; which the first min/max pair in
; DATA_ARRAY_2 is offset relative to the
; first data value in DATA_ARRAY_1.

VERTICAL_GAIN: float

VERTICAL_OFFSET: float ; to get floating values from raw data :
; VERTICAL_GAIN * data - VERTICAL_OFFSET

MAX_VALUE: float ; maximum allowed value. It corresponds
; to the upper edge of the grid.

MIN_VALUE: float ; minimum allowed value. It corresponds
; to the lower edge of the grid.

NOMINAL_BITS: word ; a measure of the intrinsic precision
; of the observation: ADC data is 8 bit
; averaged data is 10-12 bit, etc.

NOM_SUBARRAY_COUNT: word ; for Sequence, nominal segment count
; else 1

HORIZ_INTERVAL: float ; sampling interval for time domain
; waveforms

HORIZ_OFFSET: double ; trigger offset for the first sweep of
; the trigger, seconds between the
; trigger and the first data point

PIXEL_OFFSET: double ; needed to know how to display the
; waveform

VERTUNIT: unit_definition ; units of the vertical axis

HORUNIT: unit_definition ; units of the horizontal axis

HORIZ_UNCERTAINTY: float ; uncertainty from one acquisition to
; the next, of the horizontal offset in
; seconds

TRIGGER_TIME: time_stamp ; time of the trigger

ACQ_DURATION: float ; duration of the acquisition (in sec)
; in multi-trigger waveforms.
; (e.g. sequence, RIS, or averaging)

RECORD_TYPE: enum
_0 single_sweep
_1 interleaved
_2 histogram
_3 graph
_4 filter_coefficient
_5 complex
_6 extrema
_7 sequenceObsolete
_8 centeredRIS
_9 peak_detect
denum

PROCESSING_DONE: enum
_0 no_processing
_1 fir_filter
_2 interpolated
The following variables describe the basic acquisition conditions used when the waveform was acquired.

```
<320>       RESERVED5: word           ; expansion entry
<322>       RIS_SWEEPS: word         ; for RIS, the number of sweeps
                 ; else 1

<324>       TIMEBASE: enum
         _0       1_ns/div
         _1       2.5_ns/div
         _2       5_ns/div
         _3       10_ns/div
         _4       25_ns/div
         _5       50_ns/div
         _6       100_ns/div
         _7       250_ns/div
         _8       500_ns/div
         _9       1_us/div
        _10       2.5_us/div
        _11       5_us/div
        _12       10_us/div
        _13       25_us/div
        _14       50_us/div
        _15       100_us/div
        _16       250_us/div
        _17       500_us/div
        _18       1_ms/div
        _19       2.5_ms/div
        _20       5_ms/div
        _21       10_ms/div
        _22       25_ms/div
        _23       50_ms/div
        _24       100_ms/div
        _25       250_ms/div
        _26       500_ms/div
        _27       1_s/div
        _28       2.5_s/div
        _29       5_s/div
        _30       10_s/div
        _31       25_s/div
        _32       50_s/div
      _100     EXTERNAL
  endenum

<326>       VERT_COUPLING: enum
          _0       DC_50_Ohms
          _1       ground
          _2       DC_1MOhm
          _3       ground
          _4       AC_1MOhm
  endenum

<328>       PROBE_ATT: float

<332>       FIXED_VERT_GAIN: enum
          _0       2_mV/div
          _1       5_mV/div
          _2       10_mV/div
          _3       20_mV/div
          _4       50_mV/div
```
DAT1: ARRAY

Explanation of the data array DAT1. This is an optional secondary data array for special types of waveforms, and it has not been implemented in current DSO, so when you query it, it will always return 'ALL'.

< 0>     MEASUREMENT: data        ; the actual format of a data is given in the WAVEDESC descriptor by the COMM_TYPE variable.

/00     ENDARRAY

DAT2: ARRAY

Explanation of the data array DAT2. This main data array is always present. It is the only data array for waveforms. The data item is repeated for each acquired or computed data point of the first data array of any waveform.

< 0>     MEASUREMENT: data        ; the actual format of a data is given in the WAVEDESC descriptor by the COMM_TYPE variable.

/00     ENDARRAY

ALL: BLOCK

Explanation of the ALL. This data is identical to DESC block, followed by DAT1 and DAT2 array. ALL is an accepted alias name for the combined arrays DESC, DAT1 and DAT2.
< 0> MEASUREMENT_1: data ; data in DATA_ARRAY_1.
< 0> MEASUREMENT_2: data ; data in DATA_ARRAY_2.
/00 ENDARRAY

000000 ENDTTEMPLATE