

# SYSTEM COMMANDS

## 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. Where not included here, those for special options can be found in the options' dedicated Operator's Manuals.

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

### How They are Listed


The descriptions are listed in alphabetical order according to their *long* form. Thus the description of ATTENUATION, whose short form is ATTN, is listed before that of AUTO\_CALIBRATE, whose short form is ACAL. The two special indexes at the beginning of this section (*pages 3 to 8*) are designed as reference aids for quickly finding commands and queries. One lists the commands and queries in alphabetical order according to short form, while the other groups them according to subsystem or category.

### How They are Described

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

A short example illustrating a typical use is also presented.

### When Can They be Used?

All the commands and queries listed here can be used with the standard LSA1000, except when a particular option is required. The raised hand symbol  indicates a note on availability for a particular option or function.

## Command Notation

The following notation is used in the commands:

- < > Angular brackets enclose words that are used as placeholders, of which there are two types: the header path and the data parameter of a command.
- : = A colon followed by an equals sign separates a placeholder from the description of the type and range of values that may be used in a command instead of the placeholder.
- { } Braces enclose a list of choices, one of which one must be made.
- [ ] Square brackets enclose optional items.
- ... An ellipsis indicates that the items both to its left and right may be repeated a number of times.

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

```
<channel> : VOLT_DIV <v_gain>  
<channel> : = {C1, C2}  
<v_gain> : = 5.0 mV to 2.5 V
```

The first line shows the formal appearance of the command, with <channel> denoting the placeholder for the header path and <v\_gain> the placeholder for the data parameter specifying the desired vertical gain value. The second line indicates that either C1 or C2 must be chosen for the header path. And the third explains that the actual vertical gain can be set to any value between 5 mV and 2.5 V.

*Refer to Chapter 2 for an overview of the command functions and notation used.*

## Command Execution

Before attempting to execute a command or query, the LSA1000 scans it to verify its correctness and that sufficient information is given to perform the requested action.

Since interrogating the LSA1000 does not change its internal state, it may be queried at any time. The only exceptions to this are the queries \*CAL? and \*TST?, which both recalibrate the instrument.

# Commands & Queries Tabled By Short Form

Page No.	Short Form	Long Form	Subsystem (category)	What the Command/Query Does
12	ACAL	AUTO_CALIBRATE	MISCELLANEOUS	Enables or disables automatic calibration.
10	ALST?	ALL_STATUS?	STATUS	Reads and clears the contents of all status registers.
9	AOUT	ACQ_OUT	ACQUISITION	Sets the mode of the ACQ OUT signal
11	ARM	ARM_ACQUISITION	ACQUISITION	Changes acquisition state from "stopped" to "single".
13	BWL	BANDWIDTH_LIMIT	ACQUISITION	Enables/disables the bandwidth-limiting low-pass filter.
14	*CAL?	*CAL?	MISCELLANEOUS	Performs complete internal calibration of the instrument.
27	CFMT	COMM_FORMAT	COMMUNICATION	Selects the format for sending waveform data.
29	CHDR	COMM_HEADER	COMMUNICATION	Controls formatting of query responses.
30	CHLP	COMM_HELP	COMMUNICATION	Controls operational level of the RC Assistant.
31	CHL?	COMM_HELP_LOG?	COMMUNICATION	Returns the contents of the RC Assistant log.
17	CLM	CLEAR_MEMORY	FUNCTION	Clears the specified memory.
19	*CLS	*CLS	STATUS	Clears all status data registers.
18	CLSW	CLEAR_SWEEPS	FUNCTION	Restarts the cumulative processing functions.
15	CMGN?	CAL_MARGIN?	MISCELLANEOUS	Checks oil light margins and names margins and lights.
20	CMR?	CMR?	STATUS	Reads and clears the Command error Register (CMR).
22	COLR	COLOR	DISPLAY	Selects color of individual on-screen objects
25	COMB	COMBINE_CHANNELS	ACQUISITION	Controls the channel interleaving function.
26	COMS	COMBINE_SOURCE	ACQUISITION	Determines the input channel used for interleaving.
32	CONET	COMM_NET	COMMUNICATION	Specifies the LSA1000's network address.
33	CORD	COMM_ORDER	COMMUNICATION	Controls the byte order of waveform data transfers.
34	CRMS	CURSOR_MEASURE	CURSOR	Specifies the type of cursor/parameter measurement.
37	CRST	CURSOR_SET	CURSOR	Allows positioning of any one of eight cursors.
39	CRVA?	CURSOR_VALUE?	CURSOR	Returns trace values measured by specified cursors.
24	CSCH	COLOR_SCHEME	DISPLAY	Selects the display color scheme.
16	GSTS?	CAL_STATUS?	MISCELLANEOUS	Checks internal control margins, gives calibration status.
41	DATE	DATE	MISCELLANEOUS	Changes the date/time of the internal real-time clock.
42	DDR?	DDR?	STATUS	Reads, clears the Device Dependent Register (DDR).
43	DEF	DEFINE	FUNCTION	Specifies math expression for function evaluation.
49	DELF	DELETE_FILE	MASS STORAGE	Deletes files from mass storage.
50	DIR	DIRECTORY	MASS STORAGE	Creates and deletes file directories.
52	DISP	DISPLAY	DISPLAY	Controls the display screen.
40	DPNT	DATA_POINTS	DISPLAY	Controls bold/single pixel display of sample points.
53	DTJN	DOT_JOIN	DISPLAY	Controls the interpolation lines between data points.
54	DZOM	DUAL_ZOOM	DISPLAY	Sets horizontal magnification and positioning.
55	*ESE	*ESE	STATUS	Sets the Standard Event Status Enable register(ESE).
56	*ESR?	*ESR?	STATUS	Reads, clears the Event Status Register (ESR).
59	EXR?	EXR?	STATUS	Reads, clears the EXecution error Register (EXR).
62	FCR	FIND_CTR_RANGE	FUNCTION	Sets histogram center and width.
63	FCRD	FORMAT_CARD	MASS STORAGE	Formats a memory card.
65	FFLP	FORMAT_FLOPPY	MASS STORAGE	Formats a floppy disk.
67	FHDD	FORMAT_HDD	MASS STORAGE	Formats a removable hard disk.
61	FLNM	FILENAME	MASS STORAGE	Changes default filenames.
70	FRST	FUNCTION_RESET	FUNCTION	Resets a waveform-processing function.

Page No.	Short Form	Long Form	Subsystem (category)	What the Command/Query Does
69	FSCR	FULL_SCREEN	DISPLAY	Selects magnified view format for the grid.
71	GMOD	GAIN_MODE	ACQUISITION	Specifies the gain mode (gain range) of the front end.
72	GBWL	GLOBAL_BWL	ACQUISITION	Enables/disables global bandwidth-limiting.
73	GRID	GRID	DISPLAY	Specifies single-, dual- or quad-mode grid display.
74	HMAG	HOR_MAGNIFY	DISPLAY	Horizontally expands the selected math trace.
75	HPOS	HOR_POSITION	DISPLAY	Horizontally positions intensified zone's center.
77	*IDN?	*IDN?	MISCELLANEOUS	For identification purposes.
78	INE	INE	STATUS	Sets the Internal state change Enable register (INE).
79	INR?	INR?	STATUS	Reads, clears INternal state change Register (INR).
82	INSP?	INSPECT?	WAVEFORM TRANS.	Allows acquired waveform parts to be read.
81	INTS	INTENSITY	DISPLAY	Sets the grid or trace/text intensity level.
84	*IST?	*IST?	STATUS	Reads the current state of the IEEE 488.
85	MSIZ	MEMORY_SIZE	ACQUISITION	Selects max. memory length.
86	MZOM	MULTI_ZOOM	DISPLAY	Sets horizontal magnification and positioning.
87	OFST	OFFSET	ACQUISITION	Allows output channel vertical offset adjustment.
88	*OPC	*OPC	STATUS	Sets the OPC bit in the Event Status Register (ESR).
89	*OPT?	*OPT?	MISCELLANEOUS	Identifies LSA1000 options.
91	PACL	PARAMETER_CLR	CURSOR	Clears all current parameters in Custom, Pass/Fail.
92	PACU	PARAMETER_CUSTOM	CURSOR	Controls parameters with customizable qualifiers.
96	PADL	PARAMETER_DELETE	CURSOR	Deletes a specified parameter in Custom, Pass/Fail.
97	PAST?	PARAMETER_STATISTICS?	CURSOR	Returns current statistics parameter values.
98	PAVA?	PARAMETER_VALUE?	CURSOR	Returns current parameter, mask test values.
108	PECS	PER_CURSOR_SET	CURSOR	Positions independent cursors.
110	PECV?	PER_CURSOR_VALUE?	CURSOR	Returns values measured by cursors.
113	PELT	PERSIST_LAST	DISPLAY	Shows the last trace drawn in a persistence data map.
111	PERS	PERSIST	DISPLAY	Enables or disables the persistence display mode.
112	PECL	PERSIST_COLOR	DISPLAY	Controls color rendering method of persistence traces.
114	PESA	PERSIST_SAT	DISPLAY	Sets the color saturation level in persistence.
115	PESU	PERSIST_SETUP	DISPLAY	Selects display persistence duration.
101	PFCO	PASS_FAIL_CONDITION	CURSOR	Adds a Pass/Fail test condition or custom parameter.
103	PFCT	PASS_FAIL_COUNTER	CURSOR	Resets the Pass/Fail acquisition counters.
104	PFDO	PASS_FAIL_DO	CURSOR	Defines desired outcome, actions after Pass/Fail test.
106	PFMS	PASS_FAIL_MASK	CURSOR	Generates tolerance mask on a trace and stores it.
107	PFST?	PASS_FAIL_STATUS?	CURSOR	Returns the Pass/Fail test for a given line number.
116	*PRE	*PRE	STATUS	Sets the PaRallel poll Enable register (PRE).
118	RCLK	REFERENCE_CLOCK	ACQUISITION	Selects the system clock source.
117	REC	RECALL	WAVEFORM TRANS.	Recalls a file from mass storage to internal memory.
119	*RST	*RST	SAVE/RECALL	The *RST command initiates a device reset.
120	SEQ	SEQUENCE	ACQUISITION	Sets the conditions for the sequence mode acquisition.
122	*SRE	*SRE	STATUS	Sets the Service Request Enable register (SRE).
123	*STB?	*STB?	STATUS	Reads the contents of the IEEE 488.
126	STO	STORE	WAVEFORM TRANS.	Stores a trace in internal memory or mass storage.
125	STOP	STOP	ACQUISITION	Immediately stops signal acquisition.
127	STST	STORE_SETUP	WAVEFORM TRANS.	Controls the way in which traces are stored.
128	STTM	STORE_TEMPLATE	WAVEFORM TRANS.	Stores the waveform template to mass storage.

# The Commands and Queries

Page No.	Short Form	Long Form	Subsystem (category)	What the Command/Query Does
130	TDIV	TIME_DIV	ACQUISITION	Modifies the timebase setting.
129	TMPL?	TEMPLATE?	WAVEFORM TRANS.	Produces a complete waveform template copy.
131	TRA	TRACE	DISPLAY	Enables or disables the display of a trace.
133	TRDL	TRIG_DELAY	ACQUISITION	Sets the time at which the trigger is to occur.
132	*TRG	*TRG	ACQUISITION	Executes an ARM command.
134	TRLV	TRIG_LEVEL	ACQUISITION	Adjusts the trigger level of the specified trigger source.
135	TRMD	TRIG_MODE	ACQUISITION	Specifies the trigger mode.
136	TRSE	TRIG_SELECT	ACQUISITION	Selects the condition that will trigger acquisition.
137	TRSL	TRIG_SLOPE	ACQUISITION	Sets the trigger slope of the specified trigger source.
138	TRWI	TRIG_WINDOW	ACQUISITION	Sets window amplitude on current Edge trigger source.
139	*TST?	*TST?	MISCELLANEOUS	Performs an internal self-test.
142	VDIV	VOLT_DIV	ACQUISITION	Sets the vertical sensitivity.
140	VMAG	VERT_MAGNIFY	DISPLAY	Vertically expands the specified trace.
141	VPOS	VERT_POSITION	DISPLAY	Adjusts the vertical position of the specified trace.
143	VRNG	VOLT_RANGE	ACQUISITION	Sets the full-scale range in volts.
144	*WAI	*WAI	STATUS	Required by the IEEE 488.
145	WAIT	WAIT	ACQUISITION	Prevents new analysis until current is completed.
146	WF	WAVEFORM	WAVEFORM TRANS.	Transfers a waveform from controller to LSA1000.
148	WFSU	WAVEFORM_SETUP	WAVEFORM TRANS.	Specifies amount of waveform data to go to controller.
150	WFTX	WAVEFORM_TEXT	WAVEFORM TRANS.	Documents acquisition conditions.
151	XYAS?	XY_ASSIGN?	DISPLAY	Returns traces currently assigned to the XY display.
152	XYCO	XY_CURSOR_ORIGIN	CURSOR	Sets origin position of absolute cursor measurements.
153	XYCS	XY_CURSOR_SET	CURSOR	Allows positioning of XY voltage cursors.
155	XYCV?	XY_CURSOR_VALUE?	CURSOR	Returns the current values of the X vs Y cursors.
157	XYDS	XY_DISPLAY	DISPLAY	Enables or disables the XY display mode.
158	XYSA	XY_SATURATION	DISPLAY	Sets persistence color saturation level in XY display.

# Commands & Queries Tabled By Subsystem

## **ACQUISITION – Controlling Waveform Acquisition**

9	AOUT	ACQ_OUT	Sets the mode of the ACQ OUT signal.
11	ARM	ARM_ACQUISITION	Changes acquisition state from “stopped” to “single”.
13	BWL	BANDWIDTH_LIMIT	Enables or disables the bandwidth-limiting low-pass filter.
26	COMS	COMBINE_SOURCE	Determines the input channel used for interleaving.
25	COMB	COMBINE_CHANNELS	Controls the acquisition system’s channel-interleaving function.
72	GBWL	GLOBAL_BWL	Enables/disables global bandwidth-limiting.
71	GMOD	GAIN_MODE	Specifies the gain mode (gain range) of the front end.
85	MSIZ	MEMORY_SIZE	Allows selection of maximum memory length (M- and L-models only).
87	OFST	OFFSET	Allows vertical offset adjustment of the specified input channel.
118	RCLK	REFERENCE_CLOCK	Selects the system clock source.
120	SEQ	SEQUENCE	Sets the conditions for the sequence mode acquisition.
125	STOP	STOP	Immediately stops signal acquisition.
130	TDIV	TIME_DIV	Modifies the timebase setting.
133	TRDL	TRIG_DELAY	Sets the time at which the trigger is to occur.
132	*TRG	*TRG	Executes an ARM command.
134	TRLV	TRIG_LEVEL	Adjusts the level of the specified trigger source.
135	TRMD	TRIG_MODE	Specifies Trigger mode.
136	TRSE	TRIG_SELECT	Selects the condition that will trigger acquisition.
137	TRSL	TRIG_SLOPE	Sets the slope of the specified trigger source.
138	TRWI	TRIG_WINDOW	Sets the window amplitude in volts on the current Edge trigger source.
142	VDIV	VOLT_DIV	Sets the vertical sensitivity in volts/div.
143	VRNG	VOLT_RANGE	Sets the full-scale range in volts.
145	WAIT	WAIT	Prevents new command analysis until current acquisition completion.

## **COMMUNICATION – Setting Communication Characteristics**

27	CFMT	COMM_FORMAT	Selects the format to be used for sending waveform data.
29	CHDR	COMM_HEADER	Controls formatting of query responses.
30	CHLP	COMM_HELP	Controls operational level of the RC Assistant.
31	CHL?	COMM_HELP_LOG?	Returns the contents of the RC Assistant log.
32	CONET	COMM_NET	Specifies the LSA1000’s network address.
33	CORD	COMM_ORDER	Controls the byte order of waveform data transfers.

## **CURSOR – Performing Measurements**

34	CRMS	CURSOR_MEASURE	Specifies the type of cursor or parameter measurement for display.
37	CRST?	CURSOR_SET?	Allows positioning of any one of eight cursors.
39	CRVA?	CURSOR_VALUE?	Returns the values measured by the specified cursors for a given trace.
91	PACL	PARAMETER_CLR	Clears all current parameters in Custom and Pass/Fail modes.
92	PACU	PARAMETER_CUSTOM	Controls parameters with customizable qualifiers.
96	PADL	PARAMETER_DELETE	Deletes a specified parameter in Custom and Pass/Fail modes.
97	PAST?	PARAMETER_STATISTICS?	Returns current statistics values for the specified pulse parameter.
98	PAVA?	PARAMETER_VALUE?	Returns current value(s) of parameter(s) and mask tests.
108	PECS	PER_CURSOR_SET	Allows positioning of any one of six independent cursors.
110	PECV?	PER_CURSOR_VALUE?	Returns the values measured by specified cursors for a given trace.
101	PFCO	PASS_FAIL_CONDITION	Adds a Pass/Fail test condition or custom parameter to display.

# The Commands and Queries

103	PFCT	PASS_FAIL_COUNTER	Resets the Pass/Fail acquisition counters.
104	PFDO	PASS_FAIL_DO	Defines the desired outcome and actions following a Pass/Fail test.
106	PFMS	PASS_FAIL_MASK	Generates a tolerance mask around a chosen trace and stores it.
107	PFST?	PASS_FAIL_STATUS?	Returns the Pass/Fail test for a given line number.
152	XYCO	XY_CURSOR_ORIGIN	Sets position of origin for absolute cursor measurements on XY display.
153	XYCS	XY_CURSOR_SET	Allows positioning of any one of six independent XY voltage cursors.
155	XYCV?	XY_CURSOR_VALUE?	Returns current values of X vs Y cursors.

## DISPLAY – Displaying Waveforms

22	COLR	COLOR	Selects color of individual objects: traces, grids or cursors.
24	CSCH	COLOR_SCHEME	Selects the display color scheme..
40	DPNT	DATA_POINTS	Controls display of sample points in single display pixels or bold.
52	DISP	DISPLAY	Controls the oscilloscope display screen.
53	DTJN	DOT_JOIN	Controls the interpolation lines between data points.
54	DZOM	DUAL_ZOOM	Sets horiz. magnification and positioning for all expanded traces.
69	FSCR	FULL_SCREEN	Selects magnified view format for the grid.
73	GRID	GRID	Specifies grid display in single, dual or quad mode.
74	HMAG	HOR_MAGNIFY	Horizontally expands selected expansion trace.
75	HPOS	HOR_POSITION	Horizontally positions intensified zone's center on source trace.
81	INTS	INTENSITY	Sets grid or trace/text intensity level.
86	MZOM	MULTI_ZOOM	Sets horiz. magnification and positioning for all expanded traces.
111	PERS	PERSIST	Enables or disables the Persistence Display mode.
112	PECL	PERSIST_COLOR	Controls color rendering method of persistence traces.
113	PELT	PERSIST_LAST	Shows the last trace drawn in a persistence data map.
114	PESA	PERSIST_SAT	Sets the color saturation level in persistence.
115	PESU	PERSIST_SETUP	Selects display persistence duration in Persistence mode.
131	TRA	TRACE	Enables or disables the display of a trace.
140	VMAG	VERT_MAGNIFY	Vertically expands the specified trace.
141	VPOS	VERT_POSITION	Adjusts the vertical position of the specified trace.
151	XYAS?	XY_ASSIGN?	Returns the traces currently assigned to the XY display.
157	XYDS	XY_DISPLAY	Enables or disables the XY display mode.
158	XYSA	XY_SATURATION	Sets persistence color saturation level in XY display.

## FUNCTION – Performing Waveform Mathematical Operations

17	CLM	CLEAR_MEMORY	Clears the specified memory.
18	CLSW	CLEAR_SWEEPS	Restarts the cumulative processing functions.
43	DEF	DEFINE	Specifies the mathematical expression to be evaluated by a function.
62	FCR	FIND_CONTROL_RANGE	Sets histogram center and width.
70	FRST	FUNCTION_RESET	Resets a waveform processing function.

## MASS STORAGE – Creating and Deleting File Directories

49	DELF	DELETE_FILE	Deletes files from mass storage.
50	DIR	DIRECTORY	Creates and deletes file directories.
63	FCRD	FORMAT_CARD	Formats a memory card.
65	FFLP	FORMAT_FLOPPY	Formats a floppy disk.
67	FHDD	FORMAT_HDD	Formats a removable hard disk.
61	FLNM	FILENAME	Changes default filenames.

### **MISCELLANEOUS – Calibration and Testing**

12	ACAL	AUTO_CALIBRATE	Enables or disables automatic calibration.
14	*CAL?	*CAL?	Performs a complete internal calibration of the instrument.
15	CMGN?	CAL_MARGIN?	Checks oil light margins and names margins and lights.
16	CSTS?	CAL_STATUS?	Checks internal control margins, gives last calibration status.
41	DATE	DATE	Changes the date/time of the LSA1000's internal real-time clock.
77	*IDN?	*IDN?	Used for identification purposes.
89	*OPT?	*OPT?	Identifies LSA1000 options.
139	*TST?	*TST?	Performs an internal self-test.

### **SAVE / RECALL SETUP – Preserving and Restoring Settings**

119	*RST	*RST	The *RST command initiates a device reset.
-----	------	------	--

### **STATUS – Obtaining Status Information and Setting Up Service Requests**

10	ALST?	ALL_STATUS?	Reads and clears the contents of all (but one) of the status registers.
19	*CLS	*CLS	Clears all the status data registers.
20	CMR?	CMR?	Reads and clears the contents of the CoMmand error Register (CMR).
42	DDR?	DDR?	Reads and clears the Device-Dependent error Register (DDR).
55	*ESE	*ESE	Sets the standard Event Status Enable (ESE) register.
56	*ESR?	*ESR?	Reads and clears the Event Status Register (ESR).
59	EXR?	EXR?	Reads and clears the EXecution error Register (EXR).
78	INE	INE	Sets the INternal state change Enable register (INE).
79	INR?	INR?	Reads and clears the INternal state change Register (INR).
84	*IST?	*IST?	Individual STatus reads the current state of IEEE 488.
88	*OPC	*OPC	Sets to true the OPC bit (0) in the Event Status Register (ESR).
116	*PRE	*PRE	Sets the PaRallel poll Enable register (PRE).
122	*SRE	*SRE	Sets the Service Request Enable register (SRE).
123	*STB?	*STB?	Reads the contents of IEEE 488.
144	*WAI	*WAI	WAI to continue — required by IEEE 488.

### **WAVEFORM TRANSFER – Preserving and Restoring Waveforms**

82	INSP?	INSPECT?	Allows acquired waveform parts to be read.
117	REC	RECALL	Recalls a waveform file from mass storage to internal memories M1–4.
126	STO	STORE	Stores a trace in one of the internal memories M1–4 or mass storage.
127	STST	STORE_SETUP	Controls the way in which traces are stored.
128	STTM	STORE_TEMPLATE	Stores the waveform template in a mass-storage device.
129	TMPL?	TEMPLATE?	Produces a copy of the template describing a complete waveform.
146	WF	WAVEFORM	Transfers a waveform from the controller to the LSA1000.
148	WFSU	WAVEFORM_SETUP	Specifies amount of waveform data for transmission to controller.
150	WFTX	WAVEFORM_TEXT	Documents the conditions under which a waveform has been acquired.



# The Commands and Queries

## ACQUISITION

## ACQ\_OUT, AOUT Command/Query

### DESCRIPTION

The ACQ\_OUT command specifies the operation of the ACQ OUT signal. In STD mode, the ACQ OUT signal is a hardware-generated pulse that occurs at the end of every acquisition segment. In SEQ mode, behavior is identical to STD except in sequence mode. In sequence mode, the pulse is software-generated at the end of a sequence acquisition only (i.e. after the acquisition of the last segment only).

### COMMAND SYNTAX

```
Acq_OUT <mode>  
<mode> := { STD, SEQ }
```

### QUERY SYNTAX

```
Acq_OUT?
```

### RESPONSE FORMAT

```
Acq_OUT <mode>
```

# The Commands and 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:

**ALST?**

Response message:

**ALST TB,000000,ESR,000052,INR,000005,DDR,000000,  
CMR,000004,EXR,000024,URR,000000**

### **RELATED COMMANDS**

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

# The Commands and Queries

<b>ACQUISITION</b>	<b>ARM_ACQUISITION, ARM Command</b>
<b>DESCRIPTION</b>	The ARM_ACQUISITION command enables the signal acquisition process by changing the acquisition state (trigger mode) from “stopped” to “single”.
<b>COMMAND SYNTAX</b>	<code>ARM_acquisition</code>
<b>EXAMPLE</b>	The following command enables signal acquisition: <code>ARM</code>
<b>RELATED COMMANDS</b>	STOP, *TRG, TRIG_MODE, WAIT

# The Commands and Queries

## MISCELLANEOUS

## AUTO\_CALIBRATE, ACAL Command/Query

### DESCRIPTION

The AUTO\_CALIBRATE command is used to enable or disable the automatic calibration of the instrument. At power-up, auto-calibration is turned ON, i.e. all input channels are periodically calibrated for the current input amplifier and timebase settings.

The automatic calibration may be disabled by issuing the command ACAL OFF. Whenever it is convenient, a \*CAL? query may be issued to fully calibrate the LSA1000. When the LSA1000 is returned to local control, the periodic calibrations are resumed.

The response to the AUTO\_CALIBRATE? query indicates whether auto-calibration is enabled.

### COMMAND SYNTAX

**Auto\_CALibrate** <state>  
<state> : = {ON, OFF}

### QUERY SYNTAX

**Auto\_CALibrate?**

### RESPONSE FORMAT

**Auto\_CALibrate** <state>

### EXAMPLE

The following instruction disables auto-calibration:

**ACAL OFF**

### RELATED COMMANDS

\*CAL?, CAL\_STATUS

# The Commands and Queries

## ACQUISITION

## BANDWIDTH\_LIMIT, BWL Command/Query

<b>DESCRIPTION</b>	BANDWIDTH_LIMIT enables or disables the bandwidth-limiting low-pass filter. When Global_BWL (see page 72) is on the BWL command applies to all channels; when off, the command is used to set the bandwidth individually for each channel. The response to the BANDWIDTH_LIMIT? Query indicates whether the bandwidth filters are on or off.
<b>COMMAND SYNTAX</b>	<b>BandWidth_Limit &lt;mode&gt;</b>  Or, alternatively, to choose the bandwidth limit of an individual channel or channels when Global_BWL is <i>off</i> : <b>BandWidth_Limit &lt;channel&gt;,&lt;mode&gt;[,&lt;channel&gt;,&lt;mode&gt; [,&lt;channel&gt;,&lt;mode&gt;[,&lt;channel&gt;,&lt;mode&gt;]]]</b> <b>&lt;mode&gt; : = {OFF, ON, 200MHZ}</b> <b>&lt;channel&gt; : = {C1, C2}</b>
<b>QUERY SYNTAX</b>	<b>BandWidth_Limit?</b>
<b>RESPONSE FORMAT</b>	When Global_BWL is <i>on</i> , or if Global_BWL is <i>off</i> and all channels have the same bandwidth limit, the response is: <b>BandWidth_Limit &lt;mode&gt;</b>  Or, alternatively, if at least two channels have their bandwidth limit filters set differently from one another, the response is: <b>BandWidth_Limit &lt;channel&gt;,&lt;mode&gt;[,&lt;channel&gt;,&lt;mode&gt; [,&lt;channel&gt;,&lt;mode&gt;[,&lt;channel&gt;,&lt;mode&gt;]]]</b>
<b>EXAMPLE</b>	The following turns on the bandwidth filter for all channels: <b>BWL ON</b>  The following turns the bandwidth filter on for Channel 1 only (the first instruction turns <i>off</i> Global_BWL): <b>GBWL OFF</b> <b>BWL C1,ON</b>
<b>RELATED COMMANDS</b>	GLOBAL_BWL

# The Commands and Queries

## MISCELLANEOUS

**\*CAL?**  
Query

### DESCRIPTION

The \*CAL? query cause the LSA1000 to perform an internal self-calibration and generates a response that indicates whether or not the instrument completed the calibration without error. This internal calibration sequence is the same as that which occurs at power-up. At the end of the calibration, after the response has indicated how the calibration terminated, the instrument returns to the state it was in just prior to the calibration cycle.

### QUERY SYNTAX

\*CAL?

### RESPONSE FORMAT

\*CAL <diagnostics>  
<diagnostics> : = 0 or other  
0 = Calibration successful

### EXAMPLE

The following instruction forces a self-calibration:

\*CAL?

Response message (if no failure): \*CAL 0

### RELATED COMMANDS

AUTO\_CALIBRATE, CAL\_STATUS

The \*CAL? query performs an internal calibration and returns an integer denoting any errors.

<u>Bit</u>	<u>Value</u>	<u>Description</u>
0	1	C1 failure
1	2	C2 failure
2	4	C3 failure
3	8	C4 failure
4	16	TDC failure
5	32	Trigger failure
6	64	Other failure
7	128	reserved

# The Commands and Queries

MISCELLANEOUS

CAL\_MARGIN?, CMGN?

Query

## DESCRIPTION

The CAL\_MARGIN? query checks the margins on all the device-specific internal adjustments and responds with a list of all the names and margins. Margin values close to 0% or 100% are bad. Ideal margin values are around 50%.

## QUERY SYNTAX

**Cal\_MarGin** [<tag>, <margin>[, <tag>, <margin>[,...]]]  
<margin>: = <percentage in range 0–100>  
<tag>:= tag string denoting what the margin is for.

## EXAMPLE

The following instruction reads all of the margins:

The query and response

CMGN?

```
CMGN C1_A_DELAY,50.2 PCT,C1_A_GAIN_MATCH,50.2
PCT,C1_A_OFST_MATCH,50.2 PCT,C1_B_DELAY,50.2
PCT,C1_B_GAIN_MATCH,50.2 PCT,C1_B_OFST_MATCH,50.2
PCT,C1_GAIN,50 PCT,C1_OFFSET,50
PCT,C1_TRIG_THRESH1,50.2 PCT,C1_TRIG_THRESH2,50.2
PCT,C2_A_DELAY,50.2 PCT,C2_A_GAIN_MATCH,50.2
PCT,C2_A_OFST_MATCH,50.2 PCT,C2_B_DELAY,50.2
PCT,C2_B_GAIN_MATCH,50.2 PCT,C2_B_OFST_MATCH,50.2
PCT,C2_GAIN,50 PCT,C2_OFFSET,50
PCT,C2_TRIG_THRESH1,50.2 PCT,C2_TRIG_THRESH2,50.2
PCT,PS_+12_VCO,40.4 PCT,PS_+3.3,47.3 PCT,PS_+5VCO,50
PCT,PS_+5V_FE,44.9 PCT,P_+6V,62.8 PCT,PS_-2V,50.7
PCT,PS_-3.8VCO,48.9 PCT,PS_-4.5V_ADC,48 PCT,PS_-
5V_FEC,42.9 PCT,PS_-5V_FED,46.9 PCT,PS_-5V_MSH,38.8
PCT,PS_-6VC,47.8 PCT,PS_-6VD,43.7 PCT,PS_-VT,51.7
PCT,PS_12V_FAN,53.6 PCT,PS_2V5,44.4 PCT,PS_3V3,51.3
PCT,PS_VCC,37.8 PCT,PS_VEE,35.8 PCT
```

## RELATED COMMANDS

AUTO\_CALIBRATE, \*CAL?

# The Commands and Queries

MISCELLANEOUS

CAL\_STATUS?, CSTS?

Query

**DESCRIPTION**

The CAL\_STATUS? query checks the margins on all the internal controls and the status of the most recent calibration and returns an integer denoting any errors or warnings.

Bit	Value	Description
0-7	0..255	Same as for *CAL? query
8	256	Calibration recommended. This bit is set when automatic calibrations are disabled (using the AUTO_CALIBRATE command) and a temperature change or time period has elapsed that would ordinarily trigger an automatic calibration.
9	512	Margin violations detected. This bit is set when one or more of the internal controls used to maintain the calibration of the unit is currently within 10% of the end of its adjustment range. A margin violation may accompany a calibration failure (as reported in bits 0-7 or by the *CAL? command) but does not by itself necessarily indicate an internal hardware failure or that the unit is not able to perform to specifications. A margin violation not accompanied by a calibration failure indicates that the calibration passed conditionally. The unit should still function, but the warning indicates that some internal adjustments are near the end of their range. The criteria for a margin violation are intentionally more strict than the criteria for reporting a calibration failure. Thus, even properly functioning hardware may occasionally trigger a margin violation either due to an anomalous reading during a calibration or due to factors such as temperature transients (such as while the unit is warming up after power on). Frequent and persistent margin violations can be caused by extreme operating conditions (for example, extreme temperature), drift in the hardware that is exceeding the unit's ability to compensate internally, or some other hardware failure.

**QUERY SYNTAX**

Ca1\_sTatus?

**RESPONSE FORMAT**

Ca1\_sTatus <diagnostics>

<diagnostics>: = 0 to 1023

0 = Calibration status OK.

**EXAMPLE**

The following instruction reads the calibration status:

CSTS?

Response message

CSTS 256

Indicates that a calibration is recommended

**RELATED COMMANDS**

AUTO\_CALIBRATE, CAL?, CAL\_MARGIN?



# The Commands and Queries

<b>FUNCTION</b>	<b>CLEAR_MEMORY, CLM Command</b>
<b>DESCRIPTION</b>	The CLEAR_MEMORY command clears the specified memory. Data previously stored in this memory are erased and memory space is returned to the free memory pool.
<b>COMMAND SYNTAX</b>	<code>CLear_Memory &lt; memory&gt;</code> <code>&lt;memory&gt; := {M1, M2, M3, M4}</code>
<b>EXAMPLE</b>	The following command clears the memory M2. <code>CLM M2</code>
<b>RELATED COMMANDS</b>	STORE

# The Commands and Queries

<b>FUNCTION</b>	<b>CLEAR_SWEEPS, CLSW Command</b>
<b>DESCRIPTION</b>	The CLEAR_SWEEPS command restarts the cumulative processing functions: summed or continuous average, extrema, FFT power average, histogram, pulse parameter statistics, pass/fail counters, and persistence.
<b>COMMAND SYNTAX</b>	<code>CLear SWeeps</code>
<b>EXAMPLE</b>	The following example will restart the cumulative processing: <code>CLSW</code>
<b>RELATED COMMANDS</b>	DEFINE, INR

# The Commands and Queries

## 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:

**\*CLS**

### RELATED COMMANDS

ALL\_STATUS, CMR, DDR, \*ESR, EXR, \*STB, URR

# The Commands and Queries

<b>STATUS</b>	<b>CMR? Query</b>
<b>DESCRIPTION</b>	The CMR? query reads and clears the contents of the CoMmand error Register (CMR) — <i>see table next page</i> — which specifies the last syntax error type detected by the instrument.
<b>QUERY SYNTAX</b>	CMR?
<b>RESPONSE FORMAT</b>	CMR <value> <value> : = 0 to 13
<b>EXAMPLE</b>	The following instruction reads the contents of the CMR register: CMR? Response message: CMR 0
<b>RELATED COMMANDS</b>	ALL_STATUS?, *CLS

# The Commands and Queries

## ADDITIONAL INFORMATION

Command Error Status Register Structure (CMR)	
Value	Description
1	Unrecognized command/query header
2	Illegal header path
3	Illegal number
4	Illegal number suffix
5	Unrecognized keyword
6	String error
7	GET embedded in another message
10	Arbitrary data block expected
11	Non-digit character in byte count field of arbitrary data block
12	EOI detected during definite length data block transfer
13	Extra bytes detected during definite length data block transfer

# The Commands and Queries

**DISPLAY**

**COLOR, COLR**  
Command/Query

## DESCRIPTION

The COLOR command is used to select the color of an individual display object such as text, trace, grid or cursor.

The response to the COLOR? query indicates the color assigned to each display object, whether or not it is currently displayed.

*Note: This command is only effective if the color scheme (CSCH) is chosen from U1...U4.*

This command is included for used with programs such as ScopeExplorer.

## COMMAND SYNTAX

COLoR <object, color>[,...<object>,<color>]

<object> := {BACKGND, C1, C2, C3, C4, TA, TB, TC, TD, GRID, TEXT, CURSOR, NEUTRAL, WARNING},

<color> := { WHITE, CYAN, YELLOW, GREEN, MAGENTA, BLUE, RED, LTGRAY, GRAY, SLGRAY, CHGRAY, DKCYAN, CREAM, SAND, AMBER, OLIVE, LTGEEN, JADE, LMGREEN, APGREEN, EMGREEN, GRGREEN, OCSPRAY, ICEBLUE, PASTBLUE, PALEBLUE, SKYBLUE, ROYLBLUE, DEEPBLUE, NAVY, PLUM, PURPLE, AMETHYST, FUCHSIA, RASPBRY, NEONPINK, PALEPINK, PINK, VERMIL, ORANGE, CERISE, KHAKI, BROWN, BLACK}

## QUERY SYNTAX

COLoR?

## RESPONSE FORMAT

COLoR <object>,<color>[,...<object>,<color>]

## EXAMPLE

The following instruction selects color scheme U1, and then red as the color of Channel 1:

```
CSCH U1
COLR C1,RED
```

## RELATED COMMANDS

COLOR\_SCHEME, PERSIST\_COLOR

# The Commands and Queries

## ADDITIONAL INFORMATION

Notation			
<color>	Color	<color>	Color
WHITE	White	OCSPRAY	Ocean Spray
CYAN	Cyan	ICEBLUE	Ice Blue
YELLOW	Yellow	PASTBLUE	Pastel Blue
GREEN	Green	PALEBLUE	Pale Blue
MAGENTA	Magenta	SKYBLUE	Sky Blue
BLUE	Blue	ROYLBLUE	Royal Blue
RED	Red	DEEPBLUE	Deep Blue
LTGRAY	Light Gray	NAVY	Navy
GRAY	Gray	PLUM	Plum
SLGRAY	Slate Gray	PURPLE	Purple
CHGRAY	Charcoal Gray	AMETHYST	Amethyst
DKCYAN	Dark Cyan	FUCHSIA	Fuchsia
CREAM	Cream	RASPB	Raspberry
SAND	Sand	NEONPINK	Neon Pink
AMBER	Amber	PALEPINK	Pale Pink
OLIVE	Olive	PINK	Pink
LTGREEN	Light Green	VERMIL	Vermillion
JADE	Jade	ORANGE	Orange
LMGREEN	Lime Green	CERISE	Cerise
APGREEN	Apple Green	KHAKI	Khaki
EMGREEN	Emerald Green	BROWN	Brown
GRGREEN	Grass Green	BLACK	Black
<object>	Display Object	<object>	Display Object
BACKGND	Background	CURSOR	cursors
C1..C4	Channel Traces	WARNING	Warning Messages
TA..TD	Function Traces	NEUTRAL	Neutral color
GRID	Grid lines	OVERLAYS	Menu background color (Full Screen)

# The Commands and Queries

**DISPLAY**

**COLOR\_SCHEME, CSCH**  
Command/Query

**DESCRIPTION**

The COLOR\_SCHEME command is used to select the color scheme for the display.

The response to the COLOR\_SCHEME? query indicates the color scheme in use.

This command is included for used with programs such as ScopeExplorer.

**COMMAND SYNTAX**

Color\_SCHeme <scheme>

<scheme > := {1, 2, 3, 4, 5, 6, 7, U1, U2, U3, U4}

**QUERY SYNTAX**

Color\_SCHeme?

**RESPONSE FORMAT**

Color\_SCHeme <scheme>

**EXAMPLE**

The following instruction selects the user color scheme U2:

```
CSCH U2
```

**RELATED COMMANDS**

COLOR, PERSIST\_COLOR



# The Commands and Queries

## ACQUISITION

## COMBINE\_CHANNELS, COMB Command/Query

<b>DESCRIPTION</b>	The COMBINE_CHANNELS command controls the channel interleaving function of the acquisition system. The COMBINE_CHANNELS? query returns the interleaving function's current status.
<b>COMMAND SYNTAX</b>	COMBine_channels <state> <state> := {1, 2}
<b>QUERY SYNTAX</b>	COMBine_channels?
<b>RESPONSE FORMAT</b>	COMB <state>
<b>EXAMPLE</b>	The following engages the interleaving function: COMB 2
<b>RELATED COMMANDS</b>	COMBINE_SOURCE

# The Commands and Queries

## ACQUISITION

## COMBINE\_SOURCE, COMS Command/Query

<b>DESCRIPTION</b>	The COMBINE_SOURCE command controls which input channel is used for interleaving. The COMBINE_SOURCE? query returns the current setting.
<b>COMMAND SYNTAX</b>	<code>COMbine_Source &lt;channel&gt;</code> <code>&lt;channel&gt; := {C1, C2}</code>
<b>QUERY SYNTAX</b>	<code>COMbine_Source?</code>
<b>RESPONSE FORMAT</b>	<code>COMS &lt;channel&gt;</code>
<b>EXAMPLE</b>	The following sets the interleaved source to Channel 1: <code>COMS C1</code>
<b>RELATED COMMANDS</b>	COMBINE_CHANNELS

# The Commands and Queries

## COMMUNICATION

## COMM\_FORMAT, CFMT Command/Query

**DESCRIPTION** The COMM\_FORMAT command selects the format the LSA1000 uses to send waveform data. The available options allow the block format, the data type and the encoding mode to be modified from the default settings.

The COMM\_FORMAT? query returns the currently selected waveform data format.

**COMMAND SYNTAX** `Comm_ForMaT <block_format>,<data_type>,<encoding>`  
`<block_format> := {DEF9, IND0, OFF}`  
`<data_type> := {BYTE, WORD}`  
`<encoding> := {BIN, HEX}`  
(ETHERNET uses both encoding forms)  
Initial settings (i.e. after power-on) are:  
**DEF9, WORD, BIN**

**QUERY SYNTAX** `Comm_ForMaT?`

**RESPONSE FORMAT** `Comm_ForMaT <block_format>,<data_type>,<encoding>`

**EXAMPLE** The following code redefines the transmission format of waveform data. The data will be transmitted as a block of indefinite length. Data will be coded in binary and represented as 8-bit integers.

```
CFMT IND0,BYTE,BIN
```

### ADDITIONAL INFORMATIONBLOCK FORMAT

**DEF9:** Uses the IEEE 488.2 definite length arbitrary block response data format. The digit 9 indicates that the byte count consists of 9 digits. The data block directly follows the byte count field.

For example, a data block consisting of three data bytes would be sent as:

```
WF DAT1,#9000000003<DAB><DAB><DAB>
```

where <DAB> represents an eight-bit binary data byte.

# The Commands and Queries

**IND0:** Uses the IEEE 488.2 indefinite length arbitrary block response data format.

A <NL^END> (new line with EOI) signifies that block transmission has ended.

The same data bytes as above would be sent as:

**WF DAT1,#0**<DAB><DAB><DAB><NL^END>

**OFF:** Same as IND0. In addition, the data block type identifier and the leading #0 of the indefinite length block will be suppressed. The data presented above would be sent as:

**WF** <DAB><DAB><DAB><NL^END>

*Note: The format OFF does not conform to the IEEE 488.2 standard and is only provided for special applications where the absolute minimum of data transfer may be important.*

## DATA TYPE

**BYTE:** Transmits the waveform data as eight-bit signed integers (one byte).

**WORD:** Transmits the waveform data as 16-bit signed integers (two bytes).

*Note: The data type BYTE transmits only the high-order bits of the internal 16-bit representation. The precision contained in the low-order bits is lost.*

## ENCODING

**BIN:** Binary encoding

**HEX:** Hexadecimal encoding (bytes are converted to two hexadecimal ASCII digits (0, ...9, A, ...F))

**RELATED COMMANDS**      WAVEFORM

# The Commands and Queries

## COMMUNICATION

## COMM\_HEADER, CHDR Command/Query

### DESCRIPTION

The COMM\_HEADER command controls the way the LSA1000 formats responses to queries. The instrument provides three response formats: LONG format, in which responses start with the long form of the header word; SHORT format, where responses start with the short form of the header word; and OFF, for which headers are omitted from the response and suffix units in numbers are suppressed. Until the user requests otherwise, the SHORT response format is used.

This command does not affect the interpretation of messages sent to the LSA1000. Headers may be sent in their long or short form regardless of the COMM\_HEADER setting.

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

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

### COMMAND SYNTAX

`Comm_HeaDeR <mode>`

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

*Note: The default mode, i.e. the mode just after power-on, is SHORT.*

### QUERY SYNTAX

`Comm_HeaDeR?`

### RESPONSE FORMAT

`Comm_HeaDeR <mode>`

### EXAMPLE

The following code sets the response header format to SHORT:

```
CHDR SHORT
```

### RELATED COMMANDS

COMM\_HELP\_LOG

# The Commands and Queries

**COMMUNICATION**

**COMM\_HELP, CHLP**  
Command/Query

**DESCRIPTION**

The COMM\_HELP command controls the level of operation of the diagnostics utility Remote Control Assistant, which assists in debugging remote control programs. Selected when using the instrument's front-panel via the "UTILITIES" and "SPECIAL MODES" menus, Remote Control Assistant can log all message transactions occurring between the external controller and the LSA1000. The log may be viewed at any time in the provided menu on the screen and has four levels to choose from:

- OFF** Don't assist at all.
- EO** Log detected Errors Only (default after power-on).
- FD** Log the Full Dialog between the controller and the LSA1000.

**COMMAND SYNTAX**

`Comm_HeLP <level>`  
`<level> : = { OFF, EO, FD }`

The default level (i.e. the level just after power-on) is EO.

**QUERY SYNTAX**

`Comm_HeLP?`

**RESPONSE FORMAT**

`Comm_HeLP <level>`

**EXAMPLE (GPIB)**

After sending this command, all the following commands and responses will be logged:

`CHLP FD`

**RELATED COMMANDS**

`COMM_HELP_LOG`

# The Commands and Queries

## COMMUNICATION

## COMM\_HELP\_LOG?, CHL? Query

<b>DESCRIPTION</b>	The COMM_HELP_LOG query returns the current contents of the log generated by the Remote Control Assistant (see <i>CHLP description</i> ). If the optional parameter CLR is specified, the log will be cleared after the transmission. Otherwise, it will be kept.
<b>QUERY SYNTAX</b>	<code>Comm_HeLP_Log? [CLR]</code>
<b>RESPONSE FORMAT</b>	<code>Comm_Help_Log</code> <string containing the logged text>
<b>EXAMPLE (GPIB)</b>	The following code reads the remote control log and prints it: <code>CHL?</code>
<b>RELATED COMMANDS</b>	COMM_HELP

# The Commands and Queries

## COMMUNICATION

## COMM\_NET, CONET Command/Query

<b>DESCRIPTION</b>	The COMM_NET command specifies the network address of the instrument. The COMM_NET? query returns the current network address.
<b>COMMAND SYNTAX</b>	<code>COmm_NET &lt;subaddress&gt;</code> <code>&lt;subaddress&gt; := {IP, "X.X.X.X.", MASK, "X.X.X.X.", GATEWAY, "X.X.X.X"}</code>
<b>QUERY SYNTAX</b>	<code>COmm_NET?</code>
<b>RESPONSE FORMAT</b>	<code>COmm_NET &lt;subaddress&gt;</code>
<b>EXAMPLE</b>	<p style="text-align: center;"><u>WARNING</u></p> <p>This command acts immediately. The software used to send the command will need to be initialized with the new address before continuing.</p> <p>The query and response:</p> <pre>CONET? IP,"172.25.1.2",MASK,"255.255.0.0",GATEWAY,"172.25.0.1"</pre> <p>This command changes the IP:</p> <pre>COMM_NET IP,"172.28.11.77"</pre>



# The Commands and Queries

## COMMUNICATION

## COMM\_ORDER, CORD Command/Query

### DESCRIPTION

The COMM\_ORDER command controls the byte order of waveform data transfers. Waveform data may be sent with the most significant byte (MSB) or the least significant byte (LSB) in the first position. The default mode is the MSB first.

COMM\_ORDER applies equally to the waveform's descriptor and time blocks. In the descriptor some values are 16 bits long ("word"), 32 bits long ("long" or "float"), or 64 bits long ("double"). In the time block all values are floating values, i.e. 32 bits long. When "COMM\_ORDER HI" is selected, the most significant byte is sent first. When "COMM\_ORDER LO" is specified, the least significant byte is sent first.

The COMM\_ORDER? query returns the byte transmission order currently in use.

### COMMAND SYNTAX

Comm\_ORDer <mode>

<mode> := {HI, LO}

*Note: The initial mode, i.e. the mode after power-on, is HI.*

### QUERY SYNTAX

Comm\_ORDer?

### RESPONSE FORMAT

Comm\_ORDer <mode>

### EXAMPLE

The order of transmission of waveform data depends on the data type. The following table illustrates the different possibilities.

Type	CORD HI	CORD LO
Word	<MSB><LSB>	<LSB><MSB>
Long/Float	<MSB><byte2><byte3><LSB>	<LSB><byte3><byte2><MSB>
Double	<MSB><byte2>...<byte7><LSB>	<LSB><byte7>...<byte2><MSB>

### RELATED COMMANDS

WAVEFORM

# The Commands and Queries

**CURSOR**

**CURSOR\_MEASURE, CRMS**  
Command/Query

## DESCRIPTION

The CURSOR\_MEASURE command specifies the type of cursor or parameter measurement to be displayed, and is the main command for displaying parameters and pass/fail.

The CURSOR\_MEASURE? query indicates which cursors or parameter measurements are currently displayed.

Notation	
<b>ABS</b>	absolute reading of relative cursors
<b>CUST</b>	custom parameters
<b>FAIL</b>	pass/fail: fail
<b>HABS</b>	horizontal absolute cursors
<b>HPAR</b>	standard time parameters
<b>HREL</b>	horizontal relative cursors
<b>OFF</b>	cursors and parameters off
<b>PARAM</b>	synonym for VPAR
<b>PASS</b>	pass/fail: pass
<b>SHOW</b>	custom parameters (old form)
<b>STAT</b>	parameter statistics
<b>VABS</b>	vertical absolute cursors
<b>VPAR</b>	standard voltage parameters
<b>VREL</b>	vertical relative cursors

*Note: The PARAM mode is turned OFF when the XY mode is ON.*

## COMMAND SYNTAX

**CuRsor\_MeaSure** <mode>[,<submode>]

<mode> := {**CUST**, **FAIL**, **HABS**, **HPAR**, **HREL**, **OFF**, **PARAM**,  
**PASS**, **SHOW**, **VABS**, **VPAR**, **VREL**}

<submode> := {**STAT**, **ABS**}

*Note 1: The keyword STAT is optional with modes CUST, HPAR, and VPAR. If present, STAT turns parameter statistics on. Absence of STAT turns parameter statistics off.*

# The Commands and Queries

*Note 2: The keyword ABS is optional with mode HREL. If it is present, ABS chooses absolute amplitude reading of relative cursors. Absence of ABS selects relative amplitude reading of relative cursors.*

**QUERY SYNTAX**            **CuRsor\_MeaSure?**

**RESPONSE FORMAT**        **CuRsor\_MeaSure <mode>**

**EXAMPLE**                    The following command switches on the vertical relative cursors:

**CRMS VREL**

The following command determines which cursor is currently turned on:

**CRMS?**

Example of response message:

**CRMS OFF**

**RELATED COMMANDS**        **CURSOR\_SET, PARAMETER\_STATISTICS,**  
**PARAMETER\_VALUE, PASS\_FAIL\_CLEAR,**  
**PASS\_FAIL\_CONDITION, PASS\_FAIL\_DELETE,**  
**PASS\_FAIL\_MASK,**

**ADDITIONAL INFORMATION** To turn off the cursors, parameter measurements or Pass/Fail tests, use:

**CURSOR\_MEASURE OFF**

To turn on a cursor display, use one of the following four forms:

**CURSOR\_MEASURE HABS**

**CURSOR\_MEASURE HREL**

**CURSOR\_MEASURE VABS**

**CURSOR\_MEASURE VREL**

# *The Commands and Queries*

To turn on parameter measurements without statistics, use one of the following three forms:

```
CURSOR_MEASURE CUST  
CURSOR_MEASURE HPAR  
CURSOR_MEASURE VPAR
```

To turn on parameter statistics, add the keyword **STAT** to the above three forms.

To turn on Pass or Fail tests on parameter or mask tests, use:

```
CURSOR_MEASURE PASS  
CURSOR_MEASURE FAIL
```

Use the command:

```
PASS_FAIL_CONDITION
```

to select parameters in the Custom mode, and to modify the test conditions in the Pass/Fail mode.

# The Commands and Queries

**CURSOR**

**CURSOR\_SET, CRST**  
Command/Query

## DESCRIPTION

The CURSOR\_SET command allows the user to position any one of the eight independent cursors at a given screen location. The positions of the cursors can be modified or queried even if the required cursor is not currently displayed on the screen.

When setting a cursor position, a trace must be specified, relative to which the cursor will be positioned.

The CURSOR\_SET? query indicates the current position of the cursor(s). The values returned depend on the grid type selected.

*Note: To change only the trace without repositioning the cursors, the CURSOR\_SET command may be given with no argument (for example, . TB:CRST).*

Notation			
<b>HABS</b>	horizontal absolute	<b>PREF</b>	parameter reference
<b>HDIF</b>	horizontal difference	<b>VABS</b>	vertical absolute
<b>HREF</b>	horizontal reference	<b>VDIF</b>	vertical difference
<b>PDIF</b>	parameter difference	<b>VREF</b>	vertical reference

## COMMAND SYNTAX

`<trace> : CuRsor_sEt <cursor>,<position>[,<cursor>,<position>,<cursor>,<position>]`

`<trace> : = {TA, TB, TC, TD, C1, C2}`

`<cursor> : = {HABS, VABS, HREF, HDIF, VREF, VDIF, PREF, PDIF}`

`<position> : = 0 to 10 DIV (horizontal)`

`<position> : = -29.5 to 29.5 DIV (vertical)`

*Note 1: The suffix DIV is optional.*

# The Commands and Queries

*Note 2: Parameters are grouped in pairs. The first parameter specifies the cursor to be modified and the second one indicates its new value. Parameters may be grouped in any order and may be restricted to those items to be changed.*

## QUERY SYNTAX

`<trace> : CuRsor_SeT? [<cursor>,...<cursor>]`

`<cursor> : = {HABS, VABS, HREF, HDIF, VREF, VDIF, PREF, PDIF, ALL}`

## RESPONSE FORMAT

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

If <cursor> is not specified, ALL will be assumed. If the position of a cursor cannot be determined in a particular situation, its position will be indicated as UNDEF.

## EXAMPLE

The following command positions the VREF and VDIF cursors at +3 DIV and -7 DIV respectively, using Trace A as a reference:

**TA:CRST VREF,3DIV,VDIF,-7DIV**

## RELATED COMMANDS

CURSOR\_MEASURE, CURSOR VALUE,  
PARAMETER\_VALUE, PER\_CURSOR\_SET,  
XY\_CURSOR\_SET

# The Commands and Queries

## CURSOR

## CURSOR\_VALUE?, CRVA?

Query

### DESCRIPTION

The CURSOR\_VALUE? query returns the values measured by the specified cursors for a given trace. (The PARAMETER\_VALUE? query is used to obtain measured waveform parameter values.)

Notation			
<b>HABS</b>	horizontal absolute	<b>VABS</b>	vertical absolute
<b>HREL</b>	horizontal relative	<b>VREL</b>	vertical relative

### QUERY SYNTAX

<trace> : CuRsor\_VAlue? [<mode>, ...<mode>]

<trace> := {TA, TB, TC, TD, C1, C2}

<mode> := {HABS, HREL, VABS, VREL, ALL}

### RESPONSE FORMAT

<trace> : CuRsor\_VAlue HABS, <abs\_hori>, <abs\_vert>

<trace> : CuRsor\_VAlue HREL, <delta\_hori>, <delta\_vert>, <absvert\_ref>, <absvert\_dif>

<trace> : CuRsor\_VAlue VABS, <abs\_vert>

<trace> : CuRsor\_VAlue VREL, <delta\_vert>

For horizontal cursors, both horizontal as well as vertical values are given. For vertical cursors only vertical values are given.

*Note: If <mode> is not specified or equals ALL, all the measured cursor values for the specified trace are returned. If the value of a cursor cannot be determined in the current environment, the value UNDEF will be returned.*

### EXAMPLE

The following query reads the measured absolute horizontal value of the cross-hair cursor (HABS) on Channel 2:

**C2:CRVA? HABS**

Response message:

**C2:CRVA HABS,34.2E-6 S, 244 E-3 V**

### RELATED COMMANDS

CURSOR\_SET, PARAMETER\_VALUE, PER\_CURSOR\_VALUE, XY\_CURSOR\_VALUE

# The Commands and Queries

**DISPLAY**

**DATA\_POINTS, DPNT**  
Command/Query

**DESCRIPTION**

The DATA\_POINTS command is used to control whether the waveform sample points are shown as single display pixels or are made bold.

The response to the DATA\_POINTS? query indicates whether the waveform sample points are being displayed as single pixels or in bold face.

This command is included for used with programs such as ScopeExplorer.

**COMMAND SYNTAX**

Data\_PoiNTs <state>  
<state> := {NORMAL, BOLD}

**QUERY SYNTAX**

Data\_PoiNTs?

**RESPONSE FORMAT**

DataPoiNTs <state>

**EXAMPLE**

The following instruction highlights the waveform sample points:

**DPNT BOLD**



# The Commands and Queries

## MISCELLANEOUS

## DATE Command/Query

### DESCRIPTION

The DATE command changes the date/time of the LSA1000's internal real-time clock.

The DATE? query returns the current date/time setting.

### 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

*Note: It is not always necessary to specify all the DATE parameters. Only those parameters up to and including the parameter to be changed need be specified in order to change the "year" setting, specify day, month and year together with the required settings. The time settings will remain unchanged. To change the "second" setting, all the DATE parameters must be specified with the required settings.*

### QUERY SYNTAX

**DATE?**

### RESPONSE FORMAT

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

### EXAMPLE

This instruction will change the date to January 1, 1997 and the time to 1:21:16 p.m. (13:21:16 in 24-hour notation):

**DATE 1 , JAN , 1997 , 13 , 21 , 16**

# The Commands and Queries

## STATUS

## DDR? Query

### DESCRIPTION

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

Bit	Bit Value	Description
15...14		0 Reserved
13	8192	1 Timebase hardware failure detected
12	4096	1 Trigger hardware failure detected
11	2048	0 Reserved
10	1024	0 Reserved
9	512	1 Channel 2 hardware failure detected
8	256	1 Channel 1 hardware failure detected
7	128	1 External input overload condition detected
6...4		0 Reserved
3	8	0 Reserved
2	4	0 Reserved
1	2	1 Channel 2 overload condition detected
0	1	1 Channel 1 overload condition detected

### QUERY SYNTAX

DDR?

### RESPONSE FORMAT

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

### EXAMPLE

The following instruction reads the contents of the DDR register:

DDR

Response message:

DDR 0

### RELATED COMMANDS

ALL\_STATUS, \*CLS

# The Commands and Queries

## FUNCTION

## DEFINE, DEF Command/Query

### DESCRIPTION

The DEFINE command specifies the mathematical expression to be evaluated by a function. This command is used to control all functions in the standard instruments and WPOX processing packages.

### COMMAND SYNTAX

<function> : **DEFine EQN**, '<equation>'  
[, <param\_name>, <value>, ...]

*Note 1: Parameters are grouped in pairs. The first in the pair names the variable to be modified, <param\_name>, while the second one gives the new value to be assigned. Pairs can be given in any order and restricted to the variables to be changed.*

*Note 2: Space (blank) characters inside equations are optional.*

*Note 3: The LSA1000 only calculates functions that have a consumer. That is, either the function trace must be on (see the TRACE command, page 131) or the function must be used by another function or a parameter.*

### QUERY SYNTAX

<function> : **DEFine?**

### RESPONSE FORMAT

<function> : **DEFine EQN**, '<equation>'  
[, **MAXPTS**, <max\_points>]  
[, **SWEEPS**, <max\_sweeps>][, **WEIGHT**, <weight>][, **BITS**, <bits>]

<param_name>	<value>	Description
EQN	'<equation>'	Function equation as defined below
MAXPTS	<max_points>	Maximum number of points to compute
SWEEPS	<max_sweeps>	Maximum number of sweeps
<b>Parameters To Support Additional Functions in WP01</b>		
WEIGHT	<weight>	Continuous Average weight
BITS	<bits>	Number of ERES bits
<b>Parameters To Support Additional Functions in WP02</b>		
WINDOW	<window_type>	FFT window function

# The Commands and Queries

Parameters To Support Additional Functions in WP03 or DDM		
MAXBINS	<bins>	Number of bins in histogram
MAX_EVENTS	<max_values>	Maximum number of values in histogram
CENTER	<center>	Horizontal center position for histogram display.
WIDTH	<width>	Width of histogram display
VERT	<vert_scale>	Vertical scaling type
Parameters To Support Additional Functions in PRML		
LENGTH	<length>	Number of points to use from first waveform
START	<start>	Starting point in second waveform
Function Equations And Names Available On All Models		
<source>		Identity
+<source>		Identity
-<source>		Negation
<source1> + <source2>		Addition
<source1> - <source2>		Subtraction
<source1><source2>		Multiplication
<source1>/<source2>		Ratio
AVGS(<source>)		Average Summed
SINX(<source>)		Sin(x)/x interpolator
ZOOMONLY (<extended_source>)		Zoom only (No Math)
Extended Functions Available On Instruments With WP01 Processing Firmware		
ABS(<source>)		Absolute Value
AVGC(<source>)		Continuous Average
DERI(<source>)		Derivative
ERES(<source>)		Enhanced Resolution
EXP(<source>)		Exponential (power of e)
EXP10(<source>)		Exponential (power of 10)
EXTR(<source>)		Extrema (Roof and Floor)
FLOOR(EXTR(<source>))		Floor (Extrema source only)
INTG(<source>[+,-] <addend>]		Integral

# The Commands and Queries

LN(<source>)	Logarithm base e	
LOG10(<source>)	Logarithm base 10	
RESC([+,-][<multiplier>*<source>[+,-]<addend>])	Rescale	
ROOF(EXTR(<source>))	Roof (Extrema source only)	
1/<source>	Reciprocal	
SQR(<source>)	Square	
SQRT(<source>)	Square Root	
<b>FFT Functions Available on Instruments with WP02 Processing Firmware</b>		
<i>Note: The source waveform must be a time-domain signal, single segment.</i>		
FFT(<source>)	Fast Fourier Transform (complex result)	
REAL(FFT(<source>))	Real part of complex result	
IMAG(FFT(<source>))	Imaginary part of complex result	
MAG(FFT(<source>))	Magnitude of complex result	
PHASE(FFT(<source>))	Phase angle (degrees) of complex result	
PS(FFT(<source>))	Power spectrum	
PSD(FFT(<source>))	Power density	
RESC([+,-][<multiplier>]<source>[+,-]<addend>])	Rescale	
<b>Power Average Functions Available on Instruments with WP02 Processing Firmware</b>		
<i>Note: The source waveform must be another function defined as a Fourier transform.</i>		
MAG(AVGP(<function>))	PS(AVGP(<function>))	PSD(AVGP(<function>))
<b>Function Equations and Names Available on Instruments with WP03 or DDM Firmware</b>		
HIST(<custom_line>)	Histogram of parameter on custom line	
<b>Function Equations and Names Available on Instruments with PRML Firmware</b>		
CORR(<source1>,<source2>)	Cross Correlation	

## Source values

*Note: The numbers in CUST1, CUST2, CUST3, CUST4, and CUST5 refer to the line numbers of the selected custom parameters.*

<sourceN> := {TA, TB, TC, TD, M1, M2, M3, M4, C1, C2}

<function> := {TA, TB, TC, TD}

<custom\_line> := {CUST1, CUST2, CUST3, CUST4, CUST5}

# The Commands and Queries

<extended\_source> := {C1, C2, TA, TB, TC, TD, M1, M2, M3, M4}

## Values to define number of points/sweeps

<max\_points> := 50 to 10 000 000

<max\_sweeps> := 1 to 1000 (For standard instruments)

<max\_sweeps> := 1 to 1 000 000 (For WP01 only)

<max\_sweeps> := 1 to 50 000 (WP02 Power Spectrum only)

## Values for Rescale Function

<addend> := 0.0 to 1e15

<multiplier> := 0.0 to 1e15

## Values for Summation Average and ERES

<weight> := {1, 3, 7, 15, 31, 63, 127, 255, 511, 1023}

<bits> := {0.5, 1.0, 1.5, 2.0, 2.5, 3.0}

## Values for FFT window function

<window\_type> := {BLHA, FLTP, HAMM, HANN, RECT}

FFT Window Function Notation	
LHA	Blackman–Harris window
FLTP	Flat Top window
HABMM	Hamming window
HANN	von Hann window
RECT	Rectangular window

## Values for WP03 histogramming

<max\_bins> := {20, 50, 100, 200, 500, 1000, 2000}

<max\_events> := 20 to 2e9 (in a 1–2–5 sequence)

<center> := -1e15 to 1e15

<width> := 1e-30 to 1e30 (in a 1–2–5 sequence)

<vert\_scale> := {LIN, LOG, CONSTMAX}

# The Commands and Queries

Histogram Notation	
<b>LIN</b>	Use linear vertical scaling for histogram display
<b>LOG</b>	Use log vertical scaling for histogram display
<b>CONSTMAX</b>	Use constant maximum linear scaling for histogram display

## Values for PRML correlation

<length> : = 0 to 10 divisions

<start> : = 0 to 10 divisions



## AVAILABILITY

SWEEPS is the maximum number of sweeps (Average and Extrema only).

*Note: The pair SWEEPS,<max\_sweeps> applies only to the summed averaging (AVGS).*

## EXAMPLE

The following instruction defines Trace A to compute the summed average of Channel 1 using 5000 points over 200 sweeps:

```
TA:DEF EQN, 'AVGS(C1)', MAXPTS, 5000, SWEEPS, 200
```

## WP01 EXAMPLE

The following instruction defines Trace A to compute the product of Channel 1 and Channel 2, using a maximum of 10 000 input points:

```
TA:DEF EQN, 'C1*C2', MAXPTS, 10000
```

## WP02 FFT EXAMPLE

The following instruction defines Trace A to compute the Power Spectrum of the FFT of Channel 1. A maximum of 1000 points will be used for the input. The window function is Rectangular.

```
TA:DEF EQN, 'PS(FFT(C1))', MAXPTS, 1000, WINDOW, RECT
```

# The Commands and Queries

## WP02 PS EXAMPLE

The following instruction defines Trace B to compute the Power Spectrum of the Power Average of the FFT being computed by Trace A, over a maximum of 244 sweeps.

```
TB:DEF EQN, 'PS(AVGP(TA))', SWEEPS, 244
```

## WP03 EXAMPLE

The following command defines Trace C to construct the histogram of the all rise time measurements made on source Channel 1. The rise time measurement is defined on custom line 2. The histogram has a linear vertical scaling and the rise time parameter values are binned into 100 bins.

```
PACU 2,RISE,C1  
TC:DEF EQN, 'HIST(CUST2)', VERT, LIN, MAXBINS, 100
```

## RELATED COMMANDS

FIND\_CTR\_RANGE, FUNCTION\_RESET, INR?,  
PARAMETER\_CUSTOM, PARAMETER\_VALUE?,  
PASS\_FAIL\_CONDITION, TRACE



# The Commands and Queries

## 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> := {CARD<sup>Ⓛ</sup>, FLPY<sup>Ⓛ</sup>, HDD<sup>Ⓛ</sup>}

<filename> := An alphanumeric string of up to eight characters, followed by a dot and an extension of up to three characters.



### AVAILABILITY

<device> : CARD available only when MC01 option is fitted.

<device> : FLPY available only when FD01 option is fitted.

<device> : HDD available only when HD01 option is fitted.

### EXAMPLE

The following command deletes a front-panel setup from the memory card:

**DELF** DISK,CARD,FILE,`P001.PNL`

### RELATED COMMANDS

DIRECTORY, FORMAT\_CARD, FORMAT\_FLOPPY,  
FORMAT\_HDD

# The Commands and Queries

## MASS STORAGE

## DIRECTORY, DIR Command/Query

### DESCRIPTION

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

The query response consists of a double-quoted string containing a DOS-like listing of the directory. If no mass storage device is present, or if it is not formatted, the string will be empty.

### COMMAND SYNTAX

**DIRectory** DISK, <device>, ACTION, <action>, '<<directory>'

### QUERY SYNTAX

**DIRectory?** DISK, <device> [, '<<directory>']

<device> := {CARD<sup>Ⓛ</sup>, FLPY<sup>Ⓛ</sup>, HDD<sup>Ⓛ</sup>}

<action> := {CREATE, DELETE, SWITCH}

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

*Note: the query DIRectory\_list? is also accepted for backward compatibility but may not be supported in the future.*

### RESPONSE FORMAT

**DIRectory** DISK, <device> "<directory>"

<directory> := A variable length string detailing the file content of the memory card, floppy disk or hard disk.



### AVAILABILITY

<device> : CARD available only when MC01 option is fitted.

<device> : FLPY available only when FD01 option is fitted.

<device> : HDD available only when HD01 option is fitted.

# *The Commands and Queries*

## **EXAMPLE**

The following asks for a listing of the directory of the memory card:

**DIR? DISK,CARD**

Response message:

```
DIR "  
Directory          LECROY          1 DIR of 04-MAY-1998  
 10:46:20 on Memory Card  
SC1000    2859    19-DEC-1994 16:33:06  
SC1001    2859    19-DEC-1994 16:34:32  
TEST5     002     20359     12-MAY-1998  
          13:34:12  
3 File(s) 1948672 bytes free  
"
```

# The Commands and Queries

## DISPLAY

## DISPLAY, DISP Command/Query

### DESCRIPTION

The DISPLAY command controls the display screen of programs such as ScopeExplorer. When the user is remotely controlling the instrument and does not need to use the display map feature, it can be useful to switch off the display map update via the DISPLAY OFF command. This improves instrument response time, since the waveform graphic generation procedure is suppressed.

The response to the DISPLAY? query indicates the display state of the instrument.

*Note: When the display has been set to OFF, the real-time clock and the message field update. However, the waveforms and associated texts remain unchanged.*

### COMMAND SYNTAX

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

### QUERY SYNTAX

DISPlay?

### RESPONSE FORMAT

DISPlay <state>

### EXAMPLE

The following instruction turns off the display generation.

**DISP OFF**

# The Commands and Queries

**DISPLAY**

**DOT\_JOIN, DTJN**  
Command/Query

**DESCRIPTION**

The DOT\_JOIN command controls the interpolation lines between data points. This command is included for use with programs such as ScopeExplorer.

**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:

**DTJN OFF**

# The Commands and Queries

**DISPLAY**

**DUAL\_ZOOM, DZOM**  
Command/Query

**DESCRIPTION**

By setting DUAL\_ZOOM ON, the horizontal magnification and positioning controls are applied to all expanded traces simultaneously. This command is useful if the contents of all expanded traces are to be examined at the same time.

The DUAL\_ZOOM? query indicates whether multiple zoom is enabled or not.

*Note: This command has the same effect as MULTI\_ZOOM.*

This command is included for used with programs such as ScopeExplorer.

**COMMAND SYNTAX**

Dual\_Zoom <mode>  
<mode> := {ON, OFF}

**QUERY SYNTAX**

Dual\_Zoom?

**RESPONSE FORMAT**

Dual\_Zoom <mode>

**EXAMPLE**

The following instruction turns dual zoom on:

DZOM ON

**RELATED COMMANDS**

HOR\_MAGNIFY, HOR\_POSITION, MULTI\_ZOOM

# The Commands and Queries

<b>STATUS</b>	<b>*ESE</b> Command/Query
<b>DESCRIPTION</b>	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. <i>For an overview of the ESB defined events refer to the ESR table on page 58.</i>  The *ESE? query reads the contents of the ESE register.
<b>COMMAND SYNTAX</b>	<b>*ESE</b> <value> <value> : = 0 to 255
<b>QUERY SYNTAX</b>	<b>*ESE?</b>
<b>RESPONSE FORMAT</b>	<b>*ESE</b> <value>
<b>EXAMPLE</b>	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.  <b>*ESE 72</b>
<b>RELATED COMMANDS</b>	<b>*ESR</b>

# The Commands and Queries

## STATUS

**\*ESR?**  
Query

<b>DESCRIPTION</b>	The *ESR? query reads and clears the contents of the Event Status Register (ESR). The response represents the sum of the binary values of the register bits 0 to 7. The table below gives an overview of the ESR register structure.
<b>QUERY SYNTAX</b>	<b>*ESR?</b>
<b>RESPONSE FORMAT</b>	<b>*ESR</b> <value> <value> : = 0 to 255
<b>EXAMPLE</b>	The following instruction reads and clears the contents of the ESR register: <b>*ESR</b> Response message: <b>*ESR 0</b>
<b>RELATED COMMANDS</b>	ALL_STATUS, *CLS, *ESE



# The Commands and Queries

## ADDITIONAL INFORMATION

Standard Event Status Register (ES)				
Bit	Bit Value	Bit Name	Description	Note
15...8			0 reserved by IEEE 488.2	
7	128	PON	1 Power off-to-ON transition has occurred	(1)
6	64	URQ	1 User ReQuest has been issued	(2)
5	32	CME	1 CoMmand parser Error has been detected	(3)
4	16	EXE	1 EXecution Error detected	(4)
3	8	DDE	1 Device specific Error occurred	(5)
2	4	QYE	1 QuerY Error occurred	(6)
1	2	RQC	0 Instrument never requests bus control	(7)
0	1	OPC	0 OPeration Complete bit not used	(8)

# The Commands and Queries

## 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. LSA1000 in local state) or a semantic error. The EXE bit has an associated Execution Error Register (EXR) which specifies the error code. Refer to query EXR? for further details.*
- (5) *The Device specific Error (DDE) is set true (1) whenever a hardware failure has occurred at power-up, or execution time, such as a channel overload condition, a trigger or a timebase circuit defect. The origin of the failure may be localized via the DDR? or the self test \*TST? query.*
- (6) *The Query Error bit (QYE) is set true (1) whenever (a) an attempt is made to read data from the Output Queue when no output is either present or pending, (b) data in the Output Queue has been lost, (c) both output and input buffers are full (deadlock state), (d) an attempt is made by the controller to read before having sent an <END>, (e) a command is received before the response to the previous query was read (output buffer flushed).*
- (7) *The ReQuest Control bit (RQC) is always false (0), as the LSA1000 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 LSA1000 starts processing a command only when the previous command has been entirely executed.*

# The Commands and Queries

<b>STATUS</b>	<b>EXR?</b> Query
<b>DESCRIPTION</b>	The EXR? query reads and clears the contents of the EXecution error Register (EXR). The EXR register specifies the type of the last error detected during execution. <i>Refer to the table next page for further details.</i>
<b>QUERY SYNTAX</b>	<b>EXR?</b>
<b>RESPONSE FORMAT</b>	<b>EXR</b> <value> <value> : = 21 to 64
<b>EXAMPLE</b>	The following instruction reads the contents of the EXR register: <b>EXR</b> Response message (if no fault): <b>EXR 0</b>
<b>RELATED COMMANDS</b>	ALL_STATUS, *CLS

# The Commands and Queries

## ADDITIONAL INFORMATION

Execution Error Status Register Structure (EXR)	
Value	Description
21	Permission error. The command cannot be executed in local mode.
22	Environment error. The instrument is not configured to correctly process a command.
23	Option error. The command applies to an option which has not been installed.
24	Unresolved parsing error.
25	Parameter error. Too many parameters specified.
26	Non-implemented command.
30	Hex data error. A non-hexadecimal character has been detected in a hex data block.
31	Waveform error. The amount of data received does not correspond to descriptor indicators.
32	Waveform descriptor error. An invalid waveform descriptor has been detected.
33	Waveform text error. A corrupted waveform user text has been detected.
34	Waveform time error. Invalid RIS or TRIG time data has been detected.
35	Waveform data error. Invalid waveform data have been detected.
36	Panel setup error. An invalid panel setup data block has been detected.
50	No mass storage present when user attempted to access it. *
51	Mass storage not formatted when user attempted to access it. *
53	Mass storage was write protected when user attempted to create, or a file, to delete a file, or to format the device. *
54	Bad mass storage detected during formatting. *
55	Mass storage root directory full. Cannot add directory. *
56	Mass storage full when user attempted to write to it. *
57	Mass storage file sequence numbers exhausted (999 reached). *
58	Mass storage file not found. *
59	Requested directory not found. *
61	Mass storage filename not DOS compatible, or illegal filename. *
62	Cannot write on mass storage because filename already exists. *


---

\* For LSA1000s fitted with floppy disk (FD01), memory card (MCO1) or hard disk (HD01) options.

# The Commands and Queries

## MASS STORAGE

## FILENAME, FLNM Command/Query

<b>DESCRIPTION</b>	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.
<b>COMMAND SYNTAX</b>	<pre>FiLeNaMe TYPE, &lt;type&gt;, FILE, '&lt;filename&gt;'</pre> <p>&lt;type&gt; := {C1, C2, TA, TB, TC, TD, SETUP, HCOPIY }</p> <p>&lt;filename&gt; := For C1 to TD, an alphanumeric string of up to eight characters forming a legal DOS filename. Up to five characters for SETUP and HCOPIY.</p> <p><i>Note: No extension can be specified, as this is automatically assigned by the LSA1000.</i></p>
<b>QUERY SYNTAX</b>	<pre>FiLeNaMe? TYPE, &lt;type&gt;</pre> <p>&lt;type&gt; := {ALL, C1, C2, TA, TB, TC, TD, SETUP, HCOPIY}</p>
<b>RESPONSE FORMAT</b>	<pre>FiLeNaMe TYPE, &lt;type&gt;, FILE, "&lt;filename&gt;"[, TYPE, &lt;type&gt;, FILE, "&lt;filename&gt;"...]</pre>
 <b>AVAILABILITY</b>	Only available on LSA1000s fitted with the MC01, FD01 or HD01 options.
<b>EXAMPLE</b>	The following command designates channel 1 waveform files to be "TESTPNT6.xxx" where xxx is a numeric extension assigned by the LSA1000: <pre>FLNM TYPE, C1, FILE, 'TESTPNT6'</pre>
<b>RELATED COMMANDS</b>	DIRECTORY, FORMAT_CARD, FORMAT_FLOPPY, FORMAT_HDD, DELETE_FILE

# The Commands and Queries

## FUNCTION

## FIND\_CTR\_RANGE, FCR Command

### DESCRIPTION

The FIND\_CTR\_RANGE command automatically sets the center and width of a histogram to best display the accumulated events.

### COMMAND SYNTAX

<function> : **Find\_Ctr\_Range**  
<function> : = {**TA,TB,TC,TD**}



### AVAILABILITY

Command only available on LSA1000s fitted with the WP03 or DDM options.

### EXAMPLE

Assuming that Trace A (TA) has been defined as a histogram of one of the custom parameters, the following example will determine the best center and width and then rescale the histogram:

**TA:FCR**


### RELATED COMMANDS

DEFINE, PACU

# The Commands and Queries

## MASS STORAGE

## FORMAT\_CARD, FCRD Command/Query

<b>DESCRIPTION</b>	The <code>FORMAT_CARD</code> command formats the memory card according to the PCMIA/JEIDA standard with a DOS partition. The <code>FORMAT_CARD?</code> query returns the status of the card.
<b>COMMAND SYNTAX</b>	<code>Format_CaRD</code>
<b>QUERY SYNTAX</b>	<code>Format_CaRD?</code>
<b>RESPONSE FORMAT</b>	<code>Format_CaRD &lt;card_status&gt;[,&lt;read/write&gt;,&lt;free_space&gt;,&lt;card_size&gt;,&lt;battery_status&gt;]</code>  <card_status> := {NONE, BAD, BLANK, DIR_MISSING, OK} <read/write> := {WP, RW} <free_space> := A decimal number giving the number of bytes still available on the card <card_size> := A decimal number giving the total number of bytes on the card. <battery_status> := {BAT_OK, BAT_LOW, BAT_BAD}
 <b>AVAILABILITY</b>	Command available only on instruments fitted with the MC01 option.
<b>EXAMPLE</b>	The following code will first format a memory card and then verify its status:  <code>FCRD</code> <code>FCRD?</code>  Response message:  <code>FCRD OK,RW,130048,131072,BAT_OK</code>
<b>RELATED COMMANDS</b>	DIRECTORY

# The Commands and Queries

## ADDITIONAL INFORMATION

Notation	
BAD	Bad card after formatting
BAT_BAD	Bad battery or no battery
BAT_LOW	Battery should be replaced
BAT_OK	Battery is in order
BLANK	Current directory empty
DIR_MISSING	No subdirectory present. The directory "LECROY1_DIR" will be automatically created with the next "store" command
NONE	No card
OK	Card is correctly formatted
RW	Read/Write authorized
WP	Write protected



# The Commands and Queries

## MASS STORAGE

## FORMAT\_FLOPPY, FFLP Command/Query

### DESCRIPTION

The FORMAT\_FLOPPY command formats a floppy disk in the Double Density or High Density format.

The FORMAT\_FLOPPY? query returns the status of the floppy disk.

### COMMAND SYNTAX

**Format\_FLoPpy** [<type>]

<type> := {DD, HD}

If no argument is supplied, HD is used by default.

### QUERY SYNTAX

**Format\_FLoPpy?**

### RESPONSE FORMAT

**Format\_FloPpy** <floppy\_status>[,<read/write>,<free\_space>,<floppy\_size>]

<floppy\_status> := {NONE, BAD, BLANK, DIR\_MISSING, OK}

<read/write> := {WP, RW}

<free\_space> := A decimal number giving the number of bytes still available on the floppy.

<floppy\_size> := A decimal number giving the total number of bytes on the floppy.



### AVAILABILITY

Command only available on LSA1000s fitted with the FD01 option.

### EXAMPLE

The following code will first format a floppy in the Double Density (720 kB) format and then verify its status:

```
FFLP DD
```

```
FFLP?
```

Response message:

```
FFLP OK,RW,728064,737280,
```

### RELATED COMMANDS

DIRECTORY

# The Commands and Queries

## ADDITIONAL INFORMATION

Notation	
<b>BAD</b>	Bad floppy after formatting
<b>BLANK</b>	Current directory empty
<b>DD</b>	Double Density 720 kB formatted
<b>DIR_MISSING</b>	No subdirectory present. The directory "LECROY1_DIR" will be automatically created with the next "store" command.
<b>HD</b>	High Density 1.44 MB formatted
<b>NONE</b>	No floppy
<b>OK</b>	Floppy is correctly formatted
<b>RW</b>	Read/Write authorized
<b>WP</b>	Write protected

# The Commands and Queries

## MASS STORAGE

## FORMAT\_HDD, FHDD Command/Query

### DESCRIPTION

The FORMAT\_HDD command formats the removable hard disk according to the PCMIA/JEIDA standard with a DOS partition.

The FORMAT\_HDD? query returns the status of the hard disk.

### COMMAND SYNTAX

**Format\_HDD** <type>

<type> := {**QUICK**, **FULL**}

If no argument is supplied, QUICK will be used.

### QUERY SYNTAX

**Format\_HDD?**

### RESPONSE FORMAT

**Format\_HDD** <hdd\_status>[,<read/write>,<free\_space>,<hdd\_size>]

<hdd\_status> := {**NONE**, **BAD**, **BLANK**, **DIR\_MISSING**, **OK**}

<read/write> := {**WP**, **RW**}

<free\_space> := A decimal number giving the number of byte still available on the hard disk

<hdd\_size> := A decimal number giving the total number of bytes on the hard disk.



### AVAILABILITY

Command only available only on instruments fitted with the HD01 option.

### EXAMPLE

The following code will first format a hard disk and then verify its status:

```
FHDD  
FHDD?
```

Response message:

```
FHDD OK,RW,3076096,105744896
```

### RELATED COMMANDS

DIRECTORY

# The Commands and Queries

## ADDITIONAL INFORMATION

Notation	
<b>BAD</b>	Bad hard disk after formatting
<b>BLANK</b>	Current directory empty
<b>DIR_MISSING</b>	No subdirectory present. The directory "LECROY1_DIR" will be automatically created with the next "store" command
<b>NONE</b>	No hard disk
<b>OK</b>	Hard disk is correctly formatted
<b>RW</b>	Read/Write authorized
<b>WP</b>	Write protected

# The Commands and Queries

**DISPLAY**

**FULL\_SCREEN, FSCR**  
Command/Query

**DESCRIPTION**

The FULL\_SCREEN command is used to control whether the currently selected grid style is displayed in normal presentation format or with a full-screen grid. In Full Screen format, the waveform display areas are enlarged to the maximum possible size.

The response to the FULL\_SCREEN? query indicates whether or not the display is operating in Full Screen presentation format.

This command is included for used with programs such as ScopeExplorer.

**COMMAND SYNTAX**

**FullSCReen** <state>  
<state> := {ON, OFF}

**QUERY SYNTAX**

**FullSCReen?**

**RESPONSE FORMAT**

**FullSCReen** <state>

**EXAMPLE**

The following instruction enables the Full Screen presentation format:

**FSCR ON**

# The Commands and Queries

<b>FUNCTION</b>	<b>FUNCTION_RESET, FRST Command</b>
<b>DESCRIPTION</b>	The FUNCTION_RESET command resets a waveform processing function. The number of sweeps will be reset to zero and the process restarted.
<b>COMMAND SYNTAX</b>	<function> : <b>F</b> unction_ <b>R</b> e <b>S</b> e <b>T</b>
<b>EXAMPLE</b>	<function> := { <b>TA, TB, TC, TD</b> }  Assuming that Trace A (TA) has been defined as the summed average of Channel 1, the following instruction will restart the averaging process:  <b>TA:FRST</b>
<b>RELATED COMMANDS</b>	DEFINE, INR

# The Commands and Queries

## ACQUISITION

## GAIN\_MODE, GMOD Command/Query

**DESCRIPTION:** The GAIN\_MODE command specifies the gain mode (gain range) of the front end.

**COMMAND SYNTAX:** <channel>:Gain\_MODE <mode>  
<channel> := {C1, C2}  
<mode> := { LOW, MED, HI, AUTO }

**QUERY SYNTAX:** <channel>:Gain\_MODE?

**RESPONSE FORMAT:** <channel>:Gain\_MODE <mode>

**REMARKS:** The GAIN\_MODE command specifies the gain mode (gain range) of the front end.

The front end has three supported gain modes, the ratios of the ranges supported by the low, medium and high gain modes are 1, 2 and 5 respectively.

When changing gain modes, the current vertical gain (VDIV) setting is adapted by multiplying by the ratio of the old gain mode and the new. Thus, if the gain mode was LOW and the VDIV was 100mV/div, if the gain mode is changed to HIGH, the VDIV is multiplied by 1/5 to yield a new VDIV setting of 20mV/div.

**RELATED COMMANDS:** VOLT\_DIV, VOLT\_RANGE

# The Commands and Queries

## ACQUISITION

## GLOBAL\_BWL, GBWL Command/Query

<b>DESCRIPTION</b>	This turns on or off the Global Bandwidth Limit. When activated, the Bandwidth Limit applies to all channels; when deactivated, a Bandwidth Limit can be set individually for each channel (see <i>BWL, page 13</i> ). The response to the GLOBAL_BWL? query indicates whether the Global Bandwidth Limit is on or off.
<b>COMMAND SYNTAX</b>	<code>Global_BWL &lt;mode&gt;</code> <code>&lt;mode&gt; := {OFF, ON}</code>
<b>QUERY SYNTAX</b>	<code>Global_BWL?</code>
<b>RESPONSE FORMAT</b>	<code>Global_BWL &lt;mode&gt;</code>
<b>EXAMPLE</b>	The following instruction deactivates the Global Bandwidth Limit, allowing a Bandwidth Limit to be set individually for each channel (using the BWL command syntax for individual channels):  <code>GBWL OFF</code>
<b>RELATED COMMANDS</b>	BANDWIDTH_LIMIT



# The Commands and Queries

**DISPLAY**

**GRID**  
Command/Query

**DESCRIPTION**

The GRID command specifies whether the display is in single (1), dual (2), quad (4), XY or octal (8) grid mode.

The GRID? query returns the grid mode currently in use.

This command is included for used with programs such as ScopeExplorer.

**COMMAND SYNTAX**

GRID <grid>

<grid> := {SINGLE, DUAL, QUAD, OCTAL, XYONLY<sup>↑</sup>}

**QUERY SYNTAX**

GRID?

**RESPONSE FORMAT**

GRID <grid>



**AVAILABILITY**

<grid> := XYONLY available only when XY Display used.

**EXAMPLE**

The following instruction sets the screen display to dual grid mode:

**GRID DUAL**

**RELATED COMMANDS**

COLOR, INTENSITY, FULL\_SCREEN

# The Commands and Queries

**DISPLAY**

**HOR\_MAGNIFY, HMAG**  
Command/Query

## DESCRIPTION

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

If multiple zoom is enabled, the magnification factor for all expansion traces is set to the specified factor. 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 ( see table on page 124) 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.

This command is included for used with programs such as ScopeExplorer.

## COMMAND SYNTAX

```
<exp_trace> : Hor_MAGnify <factor>  
<exp_trace> : = {TA, TB, TC, TD}  
<factor> : = 1 to 20000
```

## QUERY SYNTAX

```
<exp_source> : Hor_MAGnify?
```

## RESPONSE FORMAT

```
<exp_source> : Hor_MAGnify <factor>
```

## EXAMPLE

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

```
TA:HMAG 5
```

## RELATED COMMANDS

DUAL\_ZOOM, MULTI\_ZOOM

# The Commands and Queries

**DISPLAY**

**HOR\_POSITION, HPOS**  
Command/Query

## DESCRIPTION

The HOR\_POSITION command horizontally positions the geometric center of the intensified zone on the source trace. Allowed positions range from division 0 through 10. If the source trace was acquired in sequence mode, horizontal shifting will only apply to a single segment at a time.

If the multiple zoom is enabled, the difference between the specified and the current horizontal position of the specified trace is applied to all expanded traces. 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.

If the sources of expanded traces are sequence waveforms, and the multiple zoom is enabled, the difference between the specified and the current segment of the specified trace is applied to all expanded traces. If this would cause the segment of any expanded trace to go outside the range of the number of source segments, the difference is adapted and then applied to the traces.

The VAB bit (bit 2) in the STB register ( see table on page 124) 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.

*Note: Segment number 0 has the special meaning "Show All Segments Unexpanded".*

This command is included for used with programs such as ScopeExplorer.

## COMMAND SYNTAX

`<exp_trace> : Hor_Position <hor_position>,<segment>`

`<exp_trace> := {TA, TB, TC, TD}`

`<hor_position> := 0 to 10 DIV`

`<segment> := 0 to max segments`

*Note 1: The suffix DIV is optional.*

# The Commands and Queries

*Note 2: The segment number is only relevant for waveforms acquired in sequence mode; it is ignored in single waveform acquisitions. When the segment number is set to 0, all segments will be shown.*

**QUERY SYNTAX** <exp\_trace> : Hor\_POSition?

**RESPONSE FORMAT** <exp\_trace> : Hor\_POSition <hor\_position>[,<segment>]

*Note 3: The segment number is only given for sequence waveforms.*

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

**TA:HPOS 3**

**RELATED COMMANDS** DUAL\_ZOOM, MULTI\_ZOOM

# The Commands and Queries

## MISCELLANEOUS

**\*IDN?**  
Query

### DESCRIPTION

The \*IDN? query is used for identification purposes. The response consists of four different fields providing information on the manufacturer, the LSA1000 model, the serial number and the firmware revision level.

### QUERY SYNTAX

**\*IDN?**

### RESPONSE FORMAT

**\*IDN LECROY,<model>,<serial\_number>,<firmware\_level>**

<model> : = A six- or seven-character model identifier

<serial\_number> : = A nine- or 10-digit decimal code

<firmware\_level> : = two digits giving the major release level followed by a period, then one digit giving the minor release level followed by a period and a single-digit update level (xx.y.z)

### EXAMPLE

This example issues an identification request to the LSA1000:

**\*IDN?**

Response message:

**\*IDN LECROY,LSA1000,LSA100000000,01.0.0**

# The Commands and Queries

<b>STATUS</b>	<b>INE</b> Command/Query
<b>DESCRIPTION</b>	The INE command sets the Internal state change Enable register (INE). This command allows one or more events in the INR register to be reflected in the INB summary message bit (bit 0) of the STB register. <i>For an overview of the INR defined events, refer to the table next page.</i>  The INE? query reads the contents of the INE register.
<b>COMMAND SYNTAX</b>	<b>INE</b> <value>  <value> : = 0 to 65535
<b>QUERY SYNTAX</b>	<b>INE?</b>
<b>RESPONSE FORMAT</b>	<b>INE</b> <value>
<b>EXAMPLE</b>	The following instruction allows the INB bit to be set whenever a screen dump has finished (bit 1, i.e. decimal 2), or a waveform has been acquired (bit 0, i.e. decimal 1), or both of these. Summing these two values yields the INE mask 2+1=3.  <b>INE 3</b>
<b>RELATED COMMANDS</b>	INR

# The Commands and Queries

## STATUS

## INR? Query

### DESCRIPTION

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

Internal State Register Structure (INR)		
Bit	Bit Value	Description
15...14		0 Reserved for future use
13	8192	1 Trigger is ready
12	4096	1 Pass/Fail test detected desired outcome
11	2048	1 Waveform processing has terminated in Trace D
10	1024	1 Waveform processing has terminated in Trace C
9	512	1 Waveform processing has terminated in Trace B
8	256	1 Waveform processing has terminated in Trace A
7	128	1 A memory card, floppy or hard disk exchange has been detected <sup>Ⓢ</sup>
6	64	1 Memory card, floppy or hard disk has become full in "AutoStore Fill" mode <sup>Ⓢ</sup>
5	32	0 Reserved for LeCroy use
3	8	1 A time-out has occurred in a data block transfer
2	4	1 A return to the local state is detected
0	1	1 A new signal has been acquired

### QUERY SYNTAX

INR?

### RESPONSE FORMAT

INR <state>

<state> : = 0 to 65535

# *The Commands and Queries*



## **AVAILABILITY**

These bits only available on LSA1000s fitted with MC01, FD01, or HD01 options.

## **EXAMPLE**

The following instruction reads the contents of the INR register:

**INR?**

Response message:

**INR 1026**

i.e. waveform processing in Function C and a screen dump have both terminated.

## **RELATED COMMANDS**

ALL\_STATUS, \*CLS, INE



# The Commands and Queries

**DISPLAY**

**INTENSITY, INTS**  
Command/Query

## DESCRIPTION

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

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

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

This command is included for used with programs such as ScopeExplorer.

## COMMAND SYNTAX

**INTensity** GRID,<value>, **TRACE**,<value>

<value> : = 0 to 100 [PCT]

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

*Note 2: The suffix PCT is optional.*

## QUERY SYNTAX

**INTensity?**

## RESPONSE FORMAT

**INTensity** TRACE,<value>,**GRID**,<value>

## EXAMPLE

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

**INTS** GRID,75

# The Commands and Queries

## WAVEFORM TRANSFER

## INSPECT?, INSP?

Query

### DESCRIPTION

The INSPECT? query allows the user to read parts of an acquired waveform in intelligible form. The command is based on the explanation of the format of a waveform given by the template (use the query TEMPLATE? to obtain an up-to-date copy).

**Any logical block of a waveform can be inspected using this query by giving its name enclosed in quotes as the first (string) parameter (see the template).**

The special logical block named WAVEDESC may also be inspected in more detail. By giving the name of a variable in the block WAVEDESC, enclosed in quotes as the first (string) parameter, it is possible to inspect only the actual value of that variable. See Chapter 4 for more on INSPECT?.

Notation	
BYTE	raw data as integers (truncated to 8 (m.s.b. <sup>†</sup> ))
FLOAT	normalized data (gain, offset applied) as floating point numbers (gives measured values in volts or units)
WORD	raw data as integers (truncated to 16 m.s.b.)

### QUERY SYNTAX

<trace> : INSPECT? `<string>'[, <data\_type>]

<trace> := {TA, TB, TC, TD, M1, M2, M3, M4, C1, C2}

<string> := A valid name of a logical block or a valid name of a variable contained in block WAVEDESC (see the Template).

<data\_type> := {BYTE, WORD, FLOAT}

*Note: The optional parameter <data\_type> applies only for inspecting the data arrays. It selects the representation of the data. The default <data\_type> is FLOAT.*

---

<sup>†</sup> most significant bits

# *The Commands and Queries*

## **RESPONSE FORMAT**

`<trace> : INSPect "<string>"`

`<string> : =` A string giving name(s) and value(s) of a logical block or a variable.

## **EXAMPLES (GPIB)**

The following instruction reads the value of the timebase at which the last waveform in Channel 1 was acquired:

`C1:INSP? `TIMEBASE``

Response message:

`C1:INSP `TIMEBASE: 500 US/DIV``

The following command reads the entire contents of the waveform descriptor block:

`C1:INSP? `WAVEDESC``

## **RELATED COMMANDS**

TEMPLATE, WAVEFORM\_SETUP

# The Commands and Queries

## STATUS

**\*IST?**  
Query

### DESCRIPTION

The \*IST? (Individual S**T**atus) query reads the current state of the IEEE 488.1-defined "ist" local message. The "ist" individual status message is the status bit sent during a parallel poll operation.

### QUERY SYNTAX

**\*IST?**

### RESPONSE FORMAT

**\*IST** <value>  
<value> := 0 or 1

### EXAMPLE

The following instruction cause the contents of the IST bit to be read:

**\*IST?**

Response message

**\*IST 0**

### RELATED COMMANDS

\*PRE

# The Commands and Queries

## ACQUISITION

## MEMORY\_SIZE, MSIZ Command/Query

<b>DESCRIPTION</b>	<p>MEMORY_SIZE allows selection of the maximum memory length used for acquisition. Reducing the number of data points results in faster throughput.</p> <p>The MEMORY_SIZE? query returns the current maximum memory length used to capture waveforms. When the optional suffix NUM is used with the query, the response will be returned in standard numeric format.</p>
<b>COMMAND SYNTAX</b>	<p><b>Memory_SIZE</b> &lt;size&gt;</p> <p>&lt;size&gt; : = {500, 1000, 2500, 5000, 10K, 25K, 50K, 100K, 250K, 500K, 1M, 2M, 4M, 8M}</p> <p><i>Note: The instrument will adapt to the closest valid &lt;size&gt; or numerical &lt;value&gt; according to available channel memory.</i></p>
<b>QUERY SYNTAX</b>	<p><b>Memory_SIZE?</b></p>
<b>RESPONSE FORMAT</b>	<p><b>Memory_SIZE</b> &lt;size&gt;</p>
<b>EXAMPLE</b>	<p>The following will set the LSA1000 to acquire at most 10 000 data samples per acquisition:</p> <p><b>MSIZ 10K</b></p>
<b>RELATED COMMANDS</b>	<p>TDIV, COMB?</p>

# The Commands and Queries

**DISPLAY**

**MULTI\_ZOOM, MZOM**  
Command/Query

**DESCRIPTION**

By setting MULTI\_ZOOM ON, the horizontal magnification and positioning controls apply to all expanded traces simultaneously. This command is useful if the contents of all expanded traces are to be examined at the same time.

The MULTI\_ZOOM? query indicates whether multiple zoom is enabled or not.

*Note: This command has the same effect as DUAL\_ZOOM.*

This command is included for used with programs such as ScopeExplorer.

**COMMAND SYNTAX**

**Multi\_Zoom** <mode>  
<mode> := {**ON**, **OFF**}

**QUERY SYNTAX**

**Multi\_Zoom?**

**RESPONSE FORMAT**

**Multi\_Zoom** <mode>

**EXAMPLE**

The following example turns the multiple zoom on:  
**MZOM ON**

**RELATED COMMANDS**

**HOR\_MAGNIFY**, **HOR\_POSITION**, **DUAL\_ZOOM**

# The Commands and Queries

## 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 LSA1000 is set to the closest possible value and the VAB bit (bit 2) in the STB register is set.

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

### COMMAND SYNTAX

<channel> : **OFFseT** <offset>

<channel> := {C1, C2}

<offset> := *Refer to model's specifications.*

*Note: The suffix V is optional.*

### QUERY SYNTAX

<channel> : **OFFseT?**

### RESPONSE FORMAT

<channel> : **OFFseT** <offset>

### EXAMPLE

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

**C2:OFST -3V**

# The Commands and Queries

## STATUS

**\*OPC**  
Command/Query

### DESCRIPTION

The \*OPC (Operation Complete) command sets to true the OPC bit (bit 0) in the standard Event Status Register (ESR). This command has no other effect on the operation of the LSA1000 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 LSA1000 only responds to the query when the previous command has been entirely executed.

### COMMAND SYNTAX

\*OPC

### QUERY SYNTAX

\*OPC?

### RESPONSE FORMAT

\*OPC 1

### RELATED COMMANDS

\*WAI



# The Commands and Queries

## MISCELLANEOUS

**\*OPT?**  
Query

### DESCRIPTION

The \*OPT? query identifies LSA1000 options, i.e. additional firmware or hardware options. The response consists of a series of response fields listing all the installed options.

### QUERY SYNTAX

**\*OPT?**

### RESPONSE FORMAT

**\*OPT** <option\_1>,<option\_2>,...,<option\_N>

<option\_n> : = A three- or four-character ASCII string

*Note: If no option is present, the character 0 will be returned*

### EXAMPLE

The following queries the installed options:

**\*OPT?**

If, for example, the waveform processing options WP01 and WP02 are installed, the response will be returned as:

**\*OPT WP01,WP02**

Response message if no options are installed:

**\*OPT 0**

# The Commands and Queries

**ADDITIONAL INFORMATION** *(not all options are necessarily available)*

<b>Notation</b>	
<b>CKTR</b>	CKTRIG Clock-Trigger-Ext. ref. Option
<b>DDFA</b>	Disk Drive Failure Analysis Option
<b>DDM</b>	Disk Drive Measurements Option
<b>ECL</b>	External Trigger -- ECL levels
<b>FD01</b>	Floppy Disk Option
<b>HD01</b>	Hard Disk Option
<b>JTA</b>	Jitter and Timing Analysis Option
<b>ORM</b>	Optical Recording Measurements Option
<b>PMT</b>	Power Measurement Tools
<b>PRML</b>	PRML Measurements Option
<b>MC01</b>	Memory Card Option
<b>TTL</b>	External Trigger -- TTL levels
<b>WP01</b>	Waveform Processing Option WP01
<b>WP02</b>	Waveform Processing Option WP02
<b>WP03</b>	Waveform Processing Option WP03

# The Commands and Queries

**CURSOR**

**PARAMETER\_CLR, PAEL  
Command**

**DESCRIPTION**

The PARAMETER\_CLR command clears all the current parameters from the five-line list used in the Custom and Pass/Fail modes.

*Note: This command has the same effect as the command PASS\_FAIL\_CONDITION, given without any arguments.*

**COMMAND SYNTAX**

`PAparameter_Clear`

**RELATED COMMANDS**

PARAMETER\_DELETE, PARAMETER\_VALUE,  
PASS\_FAIL\_CONDITION

# The Commands and Queries

**CURSOR**

**PARAMETER\_CUSTOM, PACU**  
Command/Query

**DESCRIPTION**

The PARAMETER\_CUSTOM command controls the parameters that have customizable qualifiers, (for example, DTLEV or RLEV) and may also be used to assign any parameter for histogramming.

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

**COMMAND SYNTAX**

**P**Parameter\_Custom <line>,<parameter>,<qualifier>[,<qualifier>,...]

<line> := 1 to 5

<parameter> := {a parameter from the table below or any parameter listed in the PAVA? command}

<qualifier> := Measurement qualifier(s) specific to each <param>. See below.

<param>	definition	<qualifier> list
<b>Standard Custom Parameters (all parameters listed are not necessarily available)</b>		
<b>DDL</b>	delta delay	<source1>,<source2>
<b>DTLEV</b>	delta time at level	<source1>,<slope1>,<level1>,<source2>,<slope2>,<level2>,<hysteresis>
<b>FLEV</b>	fall at level	<source>,<high>,<low>
<b>PHASE</b>	phase difference	<source1>,<edge1>,<level1>,<source2>,<edge2>,<level2>,<hysteresis>,<angular unit>
<b>RLEV</b>	rise at level	<source>,<low>,<high>
<b>TLEV</b>	time at level	<source>,<slope>,<level>,<hysteresis>
<b>Parameters available on instruments equipped with WP03 or DDM processing firmware</b>		
<b>FWXX</b>	full width at xx% of max	<source>,<threshold>
<b>PCTL</b>	percentile	<source>,<threshold>
<b>XAPK</b>	x position at peak	<source>,<rank>

## The Commands and Queries

<b>&lt;param&gt;</b>	<b>definition</b>	<b>&lt;qualifier&gt; list</b>
<b>Parameters available on instruments equipped with DDM processing firmware</b>		
<b>LBASE</b>	local base	<source>,<hysteresis>
<b>LBSEP</b>	local baseline separation	<source>,<hysteresis>
<b>LMAX</b>	local maximum	<source>,<hysteresis>
<b>LMIN</b>	local minimum	<source>,<hysteresis>
<b>LNUM</b>	number of local events	<source>,<hysteresis>
<b>LPP</b>	local peak to peak	<source>,<hysteresis>
<b>LTBE</b>	local time between events	<source>,<hysteresis>
<b>LTBP</b>	local time between peaks	<source>,<hysteresis>
<b>LTBT</b>	local time between troughs	<source>,<hysteresis>
<b>LTMN</b>	local time at minima	<source>,<hysteresis>
<b>LTMX</b>	local time at maxima	<source>,<hysteresis>
<b>LTOT</b>	local time over threshold	<source>,<hysteresis>,<threshold>
<b>LTPT</b>	local time peak to trough	<source>,<hysteresis>
<b>LTPP</b>	local time trough to peak	<source>,<hysteresis>
<b>LTUT</b>	local time under threshold	<source>,<hysteresis>,<threshold>
<b>NBPH</b>	narrow band phase	<source>,<freq>
<b>NBPW</b>	narrow band power	<source>,<freq>
<b>OWRITE</b>	overwrite	<source 1>,<source 2>,<freq>
<b>PW50</b>	pulse width 50	<source>,<hysteresis>
<b>PW50NEG</b>	pulse width 50 for troughs	<source>,<hysteresis>
<b>PW50POS</b>	pulse width 50 for peaks	<source>,<hysteresis>
<b>RES</b>	resolution	<source 1>,<source 2>,<hysteresis>
<b>TAA</b>	track average amplitude	<source>,<hysteresis>
<b>TAANEG</b>	track average amplitude for troughs	<source>,<hysteresis>
<b>TAAPOS</b>	track average amplitude for peaks	<source>,<hysteresis>
<b>Parameters available on instruments equipped with PRML processing firmware</b>		
<b>ACSN</b>	auto correlation signal to noise	<source>,<length>
<b>NLTS</b>	non-linear transition shift	<source>,<length>,<delay>

# The Commands and Queries

**Where:**

<sourceN> := {C1, C2, TA, TB, TC, TD}  
<slopeN> := {POS, NEG, FIRST}  
<levelN>, <low>, <high> := 1 to 99 if level is specified in percent (PCT), or  
<levelN>, <low>, <high> := Level in <sourceN> in the units of the waveform.  
<delay> := -100 PCT to 100 PCT  
<freq> := 10 to 1e9 Hz (Narrow Band center frequency).  
<hysteresis> := 0.01 to 8 divisions  
<length> := 1e-9 to 0.001 seconds  
<rank> := 1 to 100  
<threshold> := 0 to 100 percent  
<angular unit> = {PCT, DEG, RAD}

**QUERY SYNTAX** PParameter\_CUstom? <line>

**RESPONSE FORMAT** PParameter\_Custom  
<line>,<parameter>,<qualifier>[,<qualifier>,...]

**EXAMPLE 1** DTLEV

**Command Example** PACU 2,DTLEV,C1,POS,345E-3,C2,NEG,-789E-3

**Query/Response Examples** PACU? 2 returns:  
PACU 2,DTLEV,C1,POS,345E-3,C2,NEG,-789E-3  
PAVA? CUST2 returns:  
C2:PAVA CUST2,789 NS

**EXAMPLE 2** DDLY

**Command Example** PACU 2,DDLY,C1,C2

**Query/Response Examples** PACU? 2 returns:  
PACU 2,DDLY,C1,C2  
PAVA? CUST2 returns:  
C2:PAVA CUST2,123 NS

# *The Commands and Queries*

## **EXAMPLE 3**

### **RLEV**

#### **Command Example**

PACU 3,RLEV,C1,2PCT,67PCT

#### **Query/Response Examples**

PACU? 3 returns:  
PACU 3,RLEV,C1,2PCT,67PCT  
PAVA? CUST3 returns:  
C1:PAVA CUST3,23 MS

## **EXAMPLE 4**

### **FLEV**

#### **Command Example**

PACU 3,FLEV,C1,345E-3,122E-3

#### **Query/Response Examples**

PACU? 3 returns:  
PACU 3,FLEV,C1,345E-3,122E-3  
PAVA? CUST3 returns:  
C1:PAVA CUST3,23 MS

## **RELATED COMMANDS**

PARAMETER\_DELETE, PARAMETER\_VALUE,  
PASS\_FAIL\_CONDITION

# The Commands and Queries

**CURSOR**

**PARAMETER\_DELETE, PADL**  
Command

## DESCRIPTION

The PARAMETER\_DELETE command deletes a parameter at a specified line from the list of parameters used in the Custom and Pass/Fail modes.

Notation		
1	line 1	of Custom or Pass/Fail display
2	line 2	of Custom or Pass/Fail display
3	line 3	of Custom or Pass/Fail display
4	line 4	of Custom or Pass/Fail display
5	line 5	of Custom or Pass/Fail display

## COMMAND SYNTAX

**PARAMETER\_DELETE** <line>

<line> := {1, 2, 3, 4, 5}

*Note: This command has the same effect as the command PASS\_FAIL\_CONDITION <line>, given without any further arguments.*

## EXAMPLE

The following instruction deletes the third test condition in the list:

```
PADL 3
```

## RELATED COMMANDS

PARAMETER\_CLR, PARAMETER\_VALUE,  
PASS\_FAIL\_CONDITION



# The Commands and Queries

**CURSOR**

**PARAMETER\_STATISTICS?, PAST?**

Query

## DESCRIPTION

The `PARAMETER_STATISTICS?` query returns the current values of statistics for the specified pulse parameter mode and the result type, for all five lines of the pulse parameters display.

Notation	
<b>AVG</b>	average
<b>CUST</b>	custom parameters
<b>HIGH</b>	highest value
<b>HPAR</b>	horizontal standard parameters
<b>LOW</b>	lowest value
<b>PARAM</b>	parameter definition for each line
<b>SIGMA</b>	sigma (standard deviation)
<b>SWEEPS</b>	number of sweeps accumulated for each line
<b>VPAR</b>	vertical standard parameters

## QUERY SYNTAX

`PARAMETER_STATISTICS? <mode>, <result>`

`<mode> := {CUST, HPAR, VPAR}`

`<result> := {AVG, LOW, HIGH, SIGMA, SWEEPS, PARAM}`

*Note: If keyword PARAM is specified, the query returns the list of the five pairs <parameter\_name>,<source>.*

## EXAMPLE

The following query reads the average values of the five standard vertical parameters:

```
PAST? VPAR, AVG
```

## RESPONSE FORMAT

```
PAST VPAR, AVG, 13V, 26V, 47V, 1V, 0V
```

## RELATED COMMANDS

`PARAMETER_VALUE`

# The Commands and Queries

## CURSOR

## PARAMETER\_VALUE?, PAVA?

Query

### DESCRIPTION

The PARAMETER\_VALUE query returns the current value(s) of the pulse waveform parameter(s) and mask tests for the specified trace. Traces do not need to be displayed or selected to obtain the values measured by the pulse parameters or mask tests.

Standard Parameters					
<b>ALL</b>	all parameters	<b>DUTY</b>	duty cycle	<b>OVSP</b>	positive overshoot
<b>AMPL</b>	amplitude	<b>FALL</b>	falltime	<b>PER</b>	period
<b>AREA</b>	area	<b>FALL82</b>	fall 80 to 20%	<b>PKPK</b>	peak-to-peak
<b>BASE</b>	base	<b>FREQ</b>	frequency	<b>PNTS</b>	points
<b>CMEAN</b>	mean for cyclic waveform	<b>FRST</b>	first point	<b>RISE</b>	risetime
<b>CMEDI</b>	median for cyclic waveform	<b>LAST</b>	last point	<b>RISE28</b>	rise 20 to 80%
<b>CRMS</b>	root mean square for cyclic part of waveform	<b>MAX</b>	maximum	<b>RMS</b>	root mean square
<b>CSDEV</b>	standard deviation for cyclic part of waveform	<b>MEAN</b>	mean	<b>SDEV</b>	standard deviation
<b>CYCL</b>	cycles	<b>MEDI</b>	median value	<b>TOP</b>	top
<b>DLY</b>	delay	<b>MIN</b>	minimum	<b>WID</b>	width
<b>DUR</b>	duration of acquisition	<b>OVSN</b>	negative overshoot		
Parameters Available on Instruments with WP03 or DDM Processing Firmware					
<b>AVG</b>	average of distribution	<b>HMEDI</b>	median of a histogram	<b>PKS</b>	number of peaks
<b>DATA</b>	data values	<b>HRMS</b>	histogram rms value	<b>RANGE</b>	range of distribution
<b>FWHM</b>	full width at half max	<b>HTOP</b>	histogram top value	<b>SIGMA</b>	sigma of distribution
<b>HAMPL</b>	histogram amplitude	<b>LOW</b>	low of distribution	<b>TOTP</b>	total population
<b>HBASE</b>	histogram base	<b>MAXP</b>	maximum population		
<b>HIGH</b>	high of histogram	<b>MODE</b>	mode of distribution		
Custom Parameters Defined using PARAMETER_CUSTOM Command <sup>‡</sup>					
CUST1	CUST2	CUST3	CUST4	CUST5	

<sup>‡</sup> The numbers in the terms CUST1, CUST2, CUST3, CUST4 and CUST5 refer to the line numbers of the selected custom parameters.

# The Commands and Queries

Parameter Computation States			
<b>AV</b>	averaged over several (up to 100) periods	<b>OF</b>	signal partially in overflow
<b>GT</b>	greater than given value	<b>OK</b>	deemed to be determined without problem
<b>IV</b>	invalid value (insufficient data provided)	<b>OU</b>	signal partially in overflow and underflow
<b>LT</b>	less than given value	<b>PT</b>	window has been period truncated
<b>NP</b>	no pulse waveform	<b>UF</b>	signal partially in underflow
Mask Test Names			
<b>ALL_IN</b>	all points of waveform inside mask (TRUE = 1, FALSE = 0)	<b>SOME_IN</b>	some points of waveform inside mask (TRUE = 1, FALSE = 0)
<b>ALL_OUT</b>	all points of waveform outside mask (TRUE = 1, FALSE = 0)	<b>SOME_OUT</b>	some points of waveform outside mask (TRUE = 1, FALSE = 0)

## QUERY SYNTAX

`<trace> : PArAmeter_VAlue? [<parameter>, ..., <parameter>]`  
`<trace> := {TA, TB, TC, TD, C1, C2}`  
`<parameter> := See table of parameter names, previous page.`

### Alternative forms of query for mask tests:

`<trace> : PArAmeter_VAlue? <old_mask_test>`  
`<trace> : PArAmeter_VAlue? <mask_test>, <mask>`  
`<mask_test> := {ALL_IN, SOME_IN, ALL_OUT, SOME_OUT}`  
`<mask> := {TA, TB, TC, TD}`

## RESPONSE FORMAT

`<trace> : PArAmeter_VAlue <parameter>, <value> ,`  
`<state> [ , ..., <parameter>, <value>, <state> ]`  
`<value> := A decimal numeric value`  
`<state> := {OK, AV, PT, IV, NP, GT, LT, OF, UF, OU}`

*Note: If <parameter> is not specified, or is equal to ALL, all the standard voltage and standard time parameters followed by their values and states are returned.*

# *The Commands and Queries*

## **EXAMPLE**

The following query reads the risetime of Trace B (TB):

```
TB:PAVA? RISE
```

Response message:

```
TB:PAVA RISE,3.6E-9S,OK
```

## **RELATED COMMANDS**

CURSOR\_MEASURE, CURSOR\_SET,  
PARAMETER\_CUSTOM, PARAMETER\_STATISTICS

# The Commands and Queries

**CURSOR**

**PASS\_FAIL\_CONDITION, PFCO**  
Command/Query

## DESCRIPTION

The PASS\_FAIL\_CONDITION command adds a Pass/Fail test condition or a custom parameter at the specified line on the Pass/Fail or Custom Parameter display.

The PASS\_FAIL\_CONDITION? query indicates the current Pass/Fail test setup or the current selection of custom parameters at the specified line.

*Note 1: Up to five test conditions (or custom parameters) can be specified at five different display lines on the screen. The command PASS\_FAIL\_CONDITION deals with one line at a time.*

### Notation

GT	greater than	LT	lower than
----	--------------	----	------------

## COMMAND SYNTAX

### Pass\_Fail\_Condition

[<line>, <trace>, <parameter>[, <rel\_op> [, <ref\_value>]]]

<line> := {1, 2, 3, 4, 5}

<trace> := {TA, TB, TC, TD, C1, C2}

<parameter> := See tables of parameter names on pages 92 and 98.

<rel\_op> := {GT, LT}

<ref\_value> := -1e15 to +1e15

*Note 2: The PFCO command with no arguments (i.e. "PFCO") deletes all conditions. The PFCO command with a single argument (i.e. "PFCO <line>") deletes the condition at <line>.*

*Note 3: Old mask test keywords ALLI and ANYO imply testing of <trace> against the mask waveform TD. Old mask test keywords INSIDE and OUTSIDE are equivalent to ALL\_IN and SOME\_OUT; they are only supported for compatibility with former versions.*

Alternative form of command for mask tests:

Pass\_Fail\_Condition [<line>, <trace>, <mask\_test>, <mask>]

<mask\_test> := {ALL\_IN, SOME\_IN, ALL\_OUT, SOME\_OUT}

<mask> := {TA, TB, TC, TD}

# *The Commands and Queries*

## **QUERY SYNTAX**

**PFCO?** <line>

## **RESPONSE FORMAT**

**PFCO** <line>,<trace>,<parameter>,<rel\_op>,<ref\_value>

Alternative form of response for mask tests:

**PFCO** <line>,<trace>,<mask\_test>,<mask>

## **EXAMPLE**

The following instruction sets the first test condition in the list to be “frequency on Channel 1 lower than 10 kHz”:

**PFCO 1,C1,FREQ,LT,10000**

## **RELATED COMMANDS**

CURSOR\_MEASURE, CURSOR\_SET,  
PASS\_FAIL\_COUNTER, PASS\_FAIL\_DO, PASS\_FAIL\_MASK,  
PARAMETER\_VALUE

# The Commands and Queries

<b>CURSOR</b>	<b>PASS_FAIL_COUNTER, PFCT</b> Command/Query
<b>DESCRIPTION</b>	The PASS_FAIL_COUNTER command resets the Passed/Failed acquisitions counters. The PASS_FAIL_COUNTER? query returns the current counts.
<b>COMMAND SYNTAX</b>	<code>Pass_Fail_CounTer</code>
<b>QUERY SYNTAX</b>	<code>Pass_Fail_CounTer?</code>
<b>RESPONSE FORMAT</b>	<code>Pass_Fail_CounTer &lt;pass/fail&gt;, &lt;value&gt;, OF, &lt;value&gt;</code> <value> := 0 to 999999 <pass/fail> := {PASS, FAIL}
<b>EXAMPLE</b>	The following query reads the counters: <code>PFCT?</code> Response message: <code>PFCT PASS, 8, OF, 9</code>
<b>RELATED COMMANDS</b>	CURSOR_MEASURE, CURSOR_SET, PASS_FAIL_DO, PASS_FAIL_MASK, PARAMETER_VALUE



# The Commands and Queries

**CURSOR**

**PASS\_FAIL\_DO, PFDO**  
Command/Query

## DESCRIPTION

The PASS\_FAIL\_DO command defines the desired outcome and the actions that have to be performed by the LSA1000 after a Pass/Fail test. The PASS\_FAIL\_DO? query indicates which actions are currently selected.

Notation	
<b>BEEP</b> 	emit a beep
<b>PULS</b> 	emit a pulse on the CAL connector
<b>SCDP</b>	make a hard copy
<b>STO</b>	store in memory or on storage media
<b>STOP</b>	stop acquisition

## COMMAND SYNTAX

**Pass\_Fail\_DO** [<outcome>[,<act>[,<act>...]]

<outcome> := {**PASS**,**FAIL**}

<act> := {**STOP**, **SCDP**, **STO**}

*Note 2: The PFDO command with no arguments (i.e. "PFDO") deletes all actions.*

*Note 3: The STO command performs the store operation.*

*Note 4: After every pass or fail detected, the instrument sets the INR bit 12.*

## QUERY SYNTAX

**Pass\_Fail\_DO?**



# The Commands and Queries

## RESPONSE FORMAT

`Pass_Fail_DO [<pass_fail>[,<act>[,<act>...]]]`



## AVAILABILITY

The BEEP command is accepted only on models equipped with the CLBZ hardware option.

The PULS command is accepted only on models equipped with the CKIO software option.

## EXAMPLE

This following instruction forces the LSA1000 to stop acquiring when the test passes:

```
PFDO PASS,STOP
```

## RELATED COMMANDS

BUZZER, CURSOR\_MEASURE, CURSOR\_SET, INR,  
PARAMETER\_VALUE, PASS\_FAIL\_COUNTER,  
PASS\_FAIL\_MASK

# The Commands and Queries

**CURSOR**

**PASS\_FAIL\_MASK, PFMS**  
Command

**DESCRIPTION**

The PASS\_FAIL\_MASK command generates a tolerance mask around a chosen trace and stores the mask in the selected memory.

**COMMAND SYNTAX**

**Pass\_Fail\_MaSk** [<trace>[,<htol>[,<vtol>[,<mask>]]]]

<trace> := {TA, TB, TC, TD, M1, M2, M3, M4, C1, C2}

<htol> := 0.0 to 5.0

<vtol> := 0.0 to 4.0

<mask> := {M1, M2, M3, M4}

*Note: if any arguments are missing, the previous settings will be used.*

The alternative form of command:

**Pass\_Fail\_MaSk INVT** [,<mask>]

inverts the mask in the selected mask memory. If <mask> is missing, M4 is implied.

**EXAMPLE**

The following instruction generates a tolerance mask around the Channel 1 trace and stores it in M2:

**PASS\_FAIL\_MASK C1,0.2,0.3,M2**

**RELATED COMMANDS**

PASS\_FAIL\_DO, PARAMETER\_VALUE

# The Commands and Queries

**CURSOR**

**PASS\_FAIL\_STATUS?, PFST?**

Query

**DESCRIPTION**

The PASS\_FAIL\_STATUS query returns the status of the pass/fail test for a given line number.

**QUERY SYNTAX**

`Pass_Fail_Status? <line>`

`<line> := {1, 2, 3, 4, 5}`

**RESPONSE FORMAT**

`Pass_Fail_Status <line>,<state>`

`<state> := {TRUE, FALSE}`

**EXAMPLE**

The following queries the state of the pass/fail test condition specified for line 3.

`PFST? 3`

**RELATED COMMANDS**

PASS\_FAIL\_DO, PASS\_FAIL\_CONDITION,  
PARAMETER\_VALUE

# The Commands and Queries

**CURSOR**

**PER\_CURSOR\_SET, PECS**  
Command/Query

## DESCRIPTION

The PER\_CURSOR\_SET command allows the user to position any one of the six independent cursors at a given location. The position of the cursor can be modified or queried.

The PER\_CURSOR\_SET? query indicates the current position of the cursor(s).

The vertical cursor positions are the same as those controlled by the CURSOR\_SET command.

Notation			
<b>HABS</b>	horizontal absolute	<b>VABS</b>	vertical absolute
<b>HDIF</b>	horizontal difference	<b>VDIF</b>	vertical difference
<b>HREF</b>	horizontal reference	<b>VREF</b>	vertical reference

## COMMAND SYNTAX

```
<trace> : PEr_Cursor_Set <cursor> ,  
<position>[ ,<cursor> ,<position> ,... ,<cursor> ,<position>
```

```
trace> := {TA, TB, TC, TD, C1, C2}
```

```
<cursor> := {HABS, HDIF, HREF, VABS, VDIF, VREF}
```

```
<position> := 0 to 10 DIV (horizontal), -29.5 to 29.5 DIV  
(vertical)
```

*Note 1: The suffix DIV is optional.*

*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 in any order and be restricted to those variables to be changed.*

## QUERY SYNTAX

```
<trace> : PEr_Cursor_Set? <cursor>[ ,<cursor> ,... ,<cursor>]
```

```
<cursor> := {HABS, HDIF, HREF, VABS, VDIF, VREF, ALL}
```

*Note 3: If <cursor> is not specified, ALL will be assumed. If the position of a cursor cannot be determined in a particular situation, its position will be indicated as UNDEF.*

# *The Commands and Queries*

## **RESPONSE FORMAT**

**PER\_Cursor\_Set** <cursor>,<position>[,<cursor>,<position>,...,  
<cursor>,<position>

## **EXAMPLE**

The following code positions the HREF and HDIF cursors at +2.6 DIV and +7.4 DIV respectively, using Channel 2 as a reference:

**C2:PECS HREF,2.6 DIV,HDIF,7.4DIV**

## **RELATED COMMANDS**

CURSOR\_MEASURE, CURSOR\_SET, PERSIST,  
PER\_CURSOR\_VALUE,

# The Commands and Queries

## CURSOR

## PER\_CURSOR\_VALUE?, PECV?

Query

### DESCRIPTION

The PER\_CURSOR\_VALUE? query returns the values measured by the cursors specified below while in Persistence Mode.

Notation			
<b>HABS</b>	horizontal absolute	<b>VABS</b>	vertical absolute
<b>HREL</b>	horizontal relative	<b>VREL</b>	vertical relative

### QUERY SYNTAX

<trace> : **PER\_Cursor\_Value?** <cursor>[,<cursor>, ..., <cursor>]

<trace> := {**TA, TB, TC, TD, C1, C2**}

<cursor> := {**HABS, HREL, VABS, VREL, ALL**}

*Note: If <cursor> is not specified, ALL will be assumed.*

### RESPONSE FORMAT

<trace> : **PER\_Cursor\_Value** <cursor>,  
<value>[,<cursor>,<value>, ..., <cursor>,<value>]

### EXAMPLE

The following code returns the value measured with the vertical relative cursor on Channel 1:

**C1:PECV? VREL**

Response message:

**C1:PECV VREL,56 MV**

### RELATED COMMANDS

CURSOR\_MEASURE, PERSIST, PER\_CURSOR\_SET

# The Commands and Queries

**DISPLAY**

**PERSIST, PERS**  
Command/Query

**DESCRIPTION**

The PERSIST command enables or disables the persistence display mode.

This command is included for used with programs such as ScopeExplorer.

**COMMAND SYNTAX**

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

**QUERY SYNTAX**

**PERSist?**

**RESPONSE FORMAT**

**PERSist** <mode>

**EXAMPLE**

The following code turns the persistence display ON:

```
PERS ON
```

**RELATED COMMANDS**

PERSIST\_COLOR, PERSIST\_LAST, PERSIST\_SAT,  
PERSIST\_SETUP

# The Commands and Queries

**DISPLAY**

**PERSIST\_COLOR, PECL**  
Command/Query

**DESCRIPTION**

The PERSIST\_COLOR command controls the color rendering method of persistence traces.

The response to the PERSIST\_COLOR? query indicates the color rendering method, Analog Persistence or Color Graded Persistence.

This command is included for used with programs such as ScopeExplorer.

**COMMAND SYNTAX**

**P**ersist\_\*\*C\*\*oLor <state>

<state> := {ANALOG, COLOR\_GRADED}

**QUERY SYNTAX**

**P**ersist\_\*\*C\*\*oLor?

**RESPONSE FORMAT**

**P**ersist\_\*\*C\*\*oLor <state>

**EXAMPLE**

The following instruction sets the persistence trace color to an intensity-graded range of the selected trace color:

**PECL ANALOG**

**RELATED COMMANDS**

COLOR, COLOR\_SCHEME, PERSIST, PERSIST\_LAST, PERSIST\_SAT, PERSIST\_SETUP



# The Commands and Queries

**DISPLAY**

**PERSIST\_LAST, PELT**  
Command/Query

**DESCRIPTION**

The PERSIST\_LAST command controls whether or not the last trace drawn in a persistence data map is shown.

The response to the PERSIST\_LAST? query indicates whether the last trace is shown within its persistence data map.

This command is included for used with programs such as ScopeExplorer.

**COMMAND SYNTAX**

**P**ersist\_**L**as**T** <state>  
<state> := {**ON**, **OFF**}

**QUERY SYNTAX**

**P**ersist\_**L**as**T**?

**RESPONSE FORMAT**

**P**ersist\_**L**as**T** <state>

**EXAMPLE**

The following instruction ensures the last trace is visible within its persistence data map:

**PELT ON**

**RELATED COMMANDS**

PERSIST, PERSIST\_COLOR, PERSIST\_SAT, PERSIST\_SETUP

# The Commands and Queries

**DISPLAY**

**PERSIST\_SAT, PESA**  
Command/Query

## DESCRIPTION

The PERSIST\_SAT command sets the level at which the color spectrum of the persistence display is saturated.

The level is specified in terms of percentage (PCT) of the total persistence data map population. A level of 100 PCT corresponds to the color spectrum being spread across the entire depth of the persistence data map. At lower values, the spectrum will saturate (brightest value) at the specified percentage value. The PCT is optional.

The response to the PERSIST\_SAT? query indicates the saturation level of the persistence data maps.

This command is included for used with programs such as ScopeExplorer.

## COMMAND SYNTAX

**P**ersist\_sat <trace>,<value> [<trace>,<value>]

<trace> := { C1, C2, C3, C4, TA, TB, TC, TD, ALL}

<value> := 0 to 100 PCT

*Note: The suffix PCT is optional.*

## QUERY SYNTAX

**P**ersist\_sat?

## RESPONSE FORMAT

**P**ersist\_sat <trace>,<value>

## EXAMPLE

The following instruction sets the saturation level of the persistence data map for channel 3 to be 60%, i.e. 60% of the data points will be displayed with the color spectrum, with the remaining 40% saturated in the brightest color:

**PESA C3,60**

## RELATED COMMANDS

PERSIST, PERSIST\_COLOR, PERSIST\_PERS,  
PERSIST\_SETUP

# The Commands and Queries

**DISPLAY**

**PERSIST\_SETUP, PESU**  
Command/Query

**DESCRIPTION**

The PERSIST\_SETUP command selects the persistence duration of the display, in seconds, in persistence mode. In addition, the persistence can be set either to all traces or only the top two on the screen.

The PERSIST\_SETUP? query indicates the current status of the persistence.

This command is included for used with programs such as ScopeExplorer.

**COMMAND SYNTAX**

**P**ersist\_ SetUp <time>,<mode>

<time> := {0.5, 1, 2, 5, 10, 20, infinite}

<mode> := {TOP2, ALL}

**QUERY SYNTAX**

**P**ersist\_ SetUp?

**RESPONSE FORMAT**

**P**ersist\_ SetUp <time>,<mode>

**EXAMPLE**

The following instruction sets the variable persistence at 10 seconds on the top two traces:

**PESU 20, TOP2**

**RELATED COMMANDS**

PERSIST, PERSIST\_COLOR, PERSIST\_PERS, PERSIST\_SAT

# The Commands and Queries

**STATUS**

**\*PRE**  
**Command/Query**

**DESCRIPTION**

The \*PRE command sets the PaRallel poll Enable register (PRE). The lowest eight bits of the Parallel Poll Register (PPR) are composed of the STB bits. The \*PRE command allows the user to specify which bit(s) of the parallel poll register will affect the 'ist' individual status bit.

The \*PRE? query reads the contents of the PRE register. The response is a decimal number which corresponds to the binary sum of the register bits.

**COMMAND SYNTAX**

**PRE** <value>  
<value> : = 0 to 65 535

**QUERY SYNTAX**

**\*PRE?**

**RESPONSE FORMAT**

**\*PRE** <value>

**EXAMPLE**

The following instruction will cause the 'ist' status bit to become 1 as soon as the MAV bit (bit 4 of STB, i.e. decimal 16) is set. This yields the PRE value 16.

**\*PRE 16**

**RELATED COMMANDS**

**\*IST**

# The Commands and Queries

## 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 M4. *Note that only waveforms stored in BINARY format can be recalled.*

### COMMAND SYNTAX

<memory> : RECALL DISK, <device>, FILE, '*filename*'

<memory> : = {M1, M2, M3, M4, ALL}

<device> : = {CARD , FLPY , HDD 

<filename> : = An alphanumeric string of up to eight characters, followed by a dot and an extension of up to three digits.



### AVAILABILITY

This command is available only when the FD01, HD01 or MC01 option is fitted.

<device> : CARD only available when MC01 Option is fitted.

<device> : FLPY only available when FD01 Option is fitted.

<device> : HDD only available when HD01 Option is fitted.

### EXAMPLE

The following recalls a waveform file called "SC1.001" from the memory card into Memory M1:

```
M1:REC DISK,CARD,FILE, 'SC1.001'
```

### RELATED COMMANDS

FUNCTION\_STATE, STORE, INR?

# The Commands and Queries

## ACQUISITION

## REFERENCE\_CLOCK, RCLK Command/Query

<b>DESCRIPTION</b>	The REFERENCE_CLOCK selects the system clock source, allowing the LSA1000 to be phase-synchronized to an external reference clock input.
<b>COMMAND SYNTAX</b>	<code>Reference_CLoCK &lt;state&gt;</code> <code>&lt;state&gt;: = {INT, EXT}</code>
<b>QUERY SYNTAX</b>	<code>Reference_CLoCK?</code>
<b>RESPONSE FORMAT</b>	<code>Reference_CLoCK &lt;state&gt;</code>
<b>EXAMPLE</b>	The following command sets the LSA1000 to use the external reference clock. <code>RCLK EXT</code>
<b>RELATED COMMANDS</b>	*CAL, *RCL

# The Commands and Queries

## SAVE/RECALL SETUP

**\*RST**  
Command

<b>DESCRIPTION</b>	The *RST command initiates a device, recalls the default setup, and causes a calibration to be performed.
<b>COMMAND SYNTAX</b>	*RST
<b>EXAMPLE</b>	This example resets the LSA1000: *RST
<b>RELATED COMMANDS</b>	*CAL, *RCL

# The Commands and Queries

## ACQUISITION

## SEQUENCE, SEQ Command/Query

### DESCRIPTION

The SEQUENCE command sets the conditions for the sequence mode acquisition. The response to the SEQUENCE? query gives the conditions for the sequence mode acquisition. The argument <max\_size> can be expressed either as numeric fixed point, exponential or using standard suffixes. When the optional suffix NUM is used with the query, the response will be returned in standard numeric format.

### COMMAND SYNTAX

**SE**quence <mode>[,<segments>[,<max\_size>]]

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

<segments> := the right-hand column in the table below:

Max. memory length per channel	Max. number of segments
10 k	50
25 k	50
50 k	200
100 k	500
200 k	500
250 k	500
500 k	2000
1 M	2000
2 M	2000
4 M	2000

<max\_size> := {50, 100, 250, 500, 1000, 2500, 5K, 10K, 25K, 50K, 100K, 250K, 500K, 1M}

Or, alternatively, in standard numeric format:

= {...10e+3, 10.0e+3,...11e+3,...}, for example.

*Note: The instrument will adapt the requested <max\_size> to the closest valid value.*

### QUERY SYNTAX

**SE**quence? [NUM]

### RESPONSE FORMAT

**SE**quence <mode> ,<segments> ,<max\_size>



# *The Commands and Queries*

<mode> := {ON, OFF}

## **EXAMPLE (GPIB)**

The following sets the segment count to 43, the maximum segment size to 250 samples, and turns the sequence mode ON:

**SEQ ON, 43, 250**

## **RELATED COMMANDS**

TRIG\_MODE

# The Commands and Queries

**STATUS**

**\*SRE**  
**Command/Query**

## DESCRIPTION

The \*SRE command sets the Service Request Enable register (SRE). This command allows the user to specify which summary message bit(s) in the STB register will generate a service request. Refer to the table on page 124 for an overview of the available summary messages.

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

The \*SRE? query returns a value that, when converted to a binary number, represents the bit settings of the SRE register. Note that bit 6 (MSS) cannot be set and its returned value is always zero.

## COMMAND SYNTAX

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

## QUERY SYNTAX

**\*SRE?**

## RESPONSE FORMAT

**\*SRE <value>**

## EXAMPLE

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

**\*SRE 17**

# The Commands and Queries

## STATUS

**\*STB?**  
Query

### DESCRIPTION

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

The response to a \*STB? query is identical to the response of a serial poll except that the MSS summary message appears in bit 6 in place of the RQS message. *Refer to the table on page 124 for further details of the status register structure.*

### QUERY SYNTAX

\*STB?

### RESPONSE FORMAT

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

### EXAMPLE

The following reads the status byte register:

\*STB?

Response message:

\*STB 0

### RELATED COMMANDS

ALL\_STATUS, \*CLS, \*PRE, \*SRE

# The Commands and Queries

## ADDITIONAL INFORMATION

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

### Notes

- (1) The Master Summary Status (MSS) indicates that the instrument requests service, whilst the Service Request status — when set — specifies that the LSA1000 issued a service request. Bit position 6 depends on the polling method:  
 Bit 6 = MSS if an \*STB? query is received  
 = RQS if serial polling is conducted
- (2) Example: If SRE=10 and STB=10 then MSS=1. If SRE=010 and STB=100 then MSS=0.
- (3) The Event Status Bit (ESB) indicates whether or not one or more of the enabled IEEE 488.2 events have occurred since the last reading or clearing of the Standard Event Status Register (ESR). ESB is set if an enabled event becomes true (1).
- (4) The Message AVailable bit (MAV) indicates whether or not the Output queue is empty. The MAV summary bit is set true (1) whenever a data byte resides in the Output queue.
- (5) The Value Adapted Bit (VAB) is set true (1) whenever a data value in a command has been adapted to the nearest legal value. For instance, the VAB bit would be set if the timebase is redefined as 2.5  $\mu$ s/div since the adapted value is 2  $\mu$ s/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.

# The Commands and Queries

## ACQUISITION

## STOP Command

### DESCRIPTION

The STOP command immediately stops the acquisition of a signal. If the trigger mode is AUTO or NORM, it will change to trigger mode STOPPED to prevent further acquisition.

### COMMAND SYNTAX

STOP

### EXAMPLE

The following stops the acquisition process:

STOP

### RELATED COMMANDS

ARM\_ACQUISITION, TRIG\_MODE, WAIT

# The Commands and Queries

## WAVEFORM TRANSFER

## STORE, STO Command

<b>DESCRIPTION</b>	The STORE command stores the contents of the specified trace into one of the internal memories M1 to M4.
<b>COMMAND SYNTAX</b>	<pre>store [&lt;trace&gt;,&lt;dest&gt;] &lt;trace&gt; := {TA, TB, TC, TD, C1, C2, ALL_DISPLAYED} &lt;dest&gt; := {M1, M2, M3, M4}</pre> <p><i>Note: If the STORE command is sent without any argument, all traces currently enabled in the Store Setup will be stored. This setup can be modified using the STORE_SETUP command.</i></p>
<b>EXAMPLE</b>	<p>The following command stores the contents of Trace A (TA) into Memory 1 (M1):</p> <pre>STO TA,M1</pre> <p>The following command executes the storage operation currently defined in the Storage Setup (see command STORE_SETUP, page 127):</p> <pre>STO</pre>
<b>RELATED COMMANDS</b>	STORE_SETUP, RECALL

# The Commands and Queries

## WAVEFORM TRANSFER

## STORE\_SETUP, STST Command/Query

### DESCRIPTION

The STORE\_SETUP command controls the way in which traces will be stored. A single trace or all displayed traces may be enabled for storage. This applies to auto-storing or to the STORE, STO command.

The STORE\_SETUP? query returns the current mode of operation of Autostore, the current trace selection, and the current destination.

*Note that only waveforms stored in BINARY format can be recalled.*

### COMMAND SYNTAX

```
Store_Setup [<trace>, <dest>] [, AUTO, <mode>] [, FORMAT, <type>]
<trace> := {TA, TB, TC, TD, C1, C2, ALL_DISPLAYED}
<dest> := {M1, M2, M3, M4}
<mode> := {OFF, WRAP, FILL}
<type> := {BINARY, SPREADSHEET, MATHCAD, MATLAB }
```

### QUERY SYNTAX

```
Store_Setup?
```

### RESPONSE FORMAT

```
Store_Setup <trace>, <dest>, AUTO, <mode>
```

### EXAMPLE

The following selects Channel 1 to be stored and enables an “autostore” to the M1 memory until no more space is left in the memory.

```
STST C1, M1, AUTO, FILL
```

### RELATED COMMANDS

STORE, INR

# The Commands and Queries

## WAVEFORM TRANSFER

## STORE\_TEMPLATE, STTM Command

### DESCRIPTION

The STORE\_TEMPLATE command stores the instrument's waveform template on a mass-storage device. A filename is automatically generated in the form of "LECROYvv.TPL" where "vv" is the two-digit revision number.

*Note: For revision 2.1, for example, the file name generated will be LECROY21.TPL.*

*Refer to Chapter 4 for more on the waveform template, and Appendix B for a copy of the template itself.*

### COMMAND SYNTAX

SToRe\_TeMplate DISK, <device>

<device> : = {CARD , FLPY , HDD 



### AVAILABILITY

This command is available only when the FD01, HD01 or MC01 option is fitted.

<device> : CARD only available when MC01 option is fitted.

<device> : FLPY only available when FD01 option is fitted

<device> : HDD only available when HD01 option is fitted.

### EXAMPLE

The following code stores the current waveform template on the memory card for future reference:

**STTM DISK, CARD**

### RELATED COMMANDS

TEMPLATE



# The Commands and Queries

## WAVEFORM TRANSFER

## TEMPLATE?, TMPL? Query

### DESCRIPTION

The TEMPLATE? query produces a copy of the template which formally 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. *Refer to Chapter 4 for more on the waveform template, and Appendix A for a copy of the template itself.*

### QUERY SYNTAX

TeMPLate?

### RESPONSE FORMAT

TeMPLate "<template>"

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

### RELATED COMMANDS

INSPECT

# The Commands and Queries

## ACQUISITION

## TIME\_DIV, TDIV Command/Query

### DESCRIPTION

The TIME\_DIV command modifies the timebase setting. The new timebase setting may be specified with suffixes: NS for nanoseconds, US for microseconds, MS for milliseconds, S for seconds, or KS for kiloseconds. An out-of-range value causes the VAB bit (bit 2) in the STB register (see *table on page 124*) to be set.

The TIME\_DIV? query returns the current timebase setting.

### COMMAND SYNTAX

**Time\_DIV** <value>

<value> : = Refer to model's specifications.

*Note: The suffix S (seconds) is optional.*

### QUERY SYNTAX

**Time\_DIV?**

### RESPONSE FORMAT

**Time\_DIV** <value>

### EXAMPLE

The following sets the time base to 500  $\mu$ sec/div:

**TDIV 500US**

The following sets the time base to 2 msec/div:

**TDIV 0.002**

### RELATED COMMANDS

TRIG\_DELAY, TRIG\_MODE

# The Commands and Queries

**DISPLAY**

**TRACE, TRA**  
Command/Query

**DESCRIPTION**

The TRACE command enables or disables the display of a trace. An environment error ( see table on page 60) is set if an attempt is made to display more than four waveforms.

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

**COMMAND SYNTAX**

<trace> : **TRAcE** <mode>

<trace> : = {**C1, C2, TA, TB, TC, TD**}

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

**QUERY SYNTAX**

<trace> : **TRAcE?**

**RESPONSE FORMAT**

<trace> : **TRAcE** <mode>

**EXAMPLE**

The following command displays Trace A (TA):

**TA:TRA ON**

**RELATED COMMANDS**

DEFINE

# The Commands and Queries

## ACQUISITION

## \*TRG Command

### DESCRIPTION

The \*TRG command executes an ARM command.

*Note: The \*TRG command is the equivalent of the 488.1 GET (Group Execute Trigger) message.*

### COMMAND SYNTAX

**\*TRG**

### EXAMPLE

The following command enables signal acquisition:

**TRG**

### RELATED COMMANDS

ARM\_ACQUISITION, STOP, WAIT

# The Commands and Queries

## 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 (displayed at the left-hand edge of the screen).

The command expects positive trigger delays to be expressed as a percentage of the full horizontal screen. 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  $-10\ 000\ \text{div X time/div}$  and 100% is specified, the trigger time will be set to the nearest limit and the VAB bit (bit 2) will be set in the STB register.

The response to the TRIG\_DELAY? query indicates the trigger time with respect to the first acquired data point. Positive times are expressed as a percentage of the full horizontal screen and negative times in seconds.

### COMMAND SYNTAX

**TRig\_DeLay** <value>

<value> := 0.00 PCT to 100.00 PCT (pretrigger)  
-20 PS to -10 MAS (post-trigger)

*Note: The suffix is optional. For positive numbers the suffix PCT is assumed. For negative numbers the suffix S is assumed. MAS is the suffix for Ms (megaseconds), useful only for extremely large delays at very slow timebases.*

### QUERY SYNTAX

**TRig\_DeLay?**

### RESPONSE FORMAT

**TRig\_DeLay** <value>

### EXAMPLE

The following command sets the trigger delay to -20 s (post-trigger):

```
TRDL - 20S
```

### RELATED COMMANDS

TIME\_DIV, TRIG\_COUPLING, TRIG\_LEVEL, TRIG\_LEVEL\_2, TRIG\_MODE, TRIG\_SELECT, TRIG\_SLOPE, TRIG\_WINDOW

# The Commands and Queries

## ACQUISITION

## TRIG\_LEVEL, TRLV Command/Query

### DESCRIPTION

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

The TRIG\_LEVEL? query returns the current trigger level.

### COMMAND SYNTAX

<trig\_source> : TRig\_LeVel <trig\_level>

<trig\_source> := {C1, C2}

<trig\_level> := *Refer to model's specifications.*

*Note: The suffix V is optional.*

### QUERY SYNTAX

<trig\_source> : TRig\_LeVel?

### RESPONSE FORMAT

<trig\_source> : TRig\_LeVel <trig\_level>

### EXAMPLE

The following code adjusts the trigger level of Channel 2 to -3.4 V:

```
C2:TRLV -3.4V
```

### RELATED COMMANDS

TRIG\_COUPLING, TRIG\_DELAY, TRIG\_LEVEL\_2,  
TRIG\_MODE, TRIG\_SELECT, TRIG\_SLOPE, TRIG\_WINDOW

# The Commands and Queries

## ACQUISITION

## TRIG\_MODE, TRMD Command/Query

<b>DESCRIPTION</b>	The TRIG_MODE command specifies the trigger mode. The TRIG_MODE? query returns the current trigger mode.
<b>COMMAND SYNTAX</b>	TRig_MoDe <mode> <mode> := {AUTO, NORM, SINGLE, STOP}
<b>QUERY SYNTAX</b>	TRig_MoDe?
<b>RESPONSE FORMAT</b>	TRig_MoDe <mode>
<b>EXAMPLE</b>	The following selects the normal mode:  TRMD NORM
<b>RELATED COMMANDS</b>	ARM_ACQUISITION, STOP, TRIG_SELECT, TRIG_COUPLING, TRIG_LEVEL, TRIG_SLOPE, TRIG_WINDOW

# The Commands and Queries

## ACQUISITION

## TRIG\_SELECT, TRSE Command/Query

### DESCRIPTION

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

The TRIG\_SELECT? query returns the current trigger condition.

### Trigger Notation

EDGE	Edge	WIND	Window
------	------	------	--------

### COMMAND SYNTAX

**TRig\_Select** <trig\_type>,SR,<source>,HT,<hold\_type>

<trig\_type> := {EDGE, WIND<sup>☞</sup>}

<source> := {C1, C2, EX}

<hold\_value> := {OFF} — OFF is the only hold type available

*Note: The suffix S (seconds) is optional.*

### QUERY SYNTAX

**TRig\_Select?**

### RESPONSE FORMAT

**TRig\_Select** <trig\_type>,SR,<source>,HT,OFF



### AVAILABILITY

<trig\_type> : WIND not available with EX as source.

### EXAMPLE

The following sets up the trigger system to trigger on Channel 1:

**TRSE EDGE,SR,C1**

### RELATED COMMANDS

TRIG\_COUPLING, TRIG\_DELAY, TRIG\_LEVEL, TRIG\_MODE, TRIG\_SLOPE, TRIG\_SLOPE, TRIG\_WINDOW



# The Commands and Queries

## ACQUISITION

## TRIG\_SLOPE, TRSL Command/Query

### DESCRIPTION

The TRIG\_SLOPE command sets the trigger slope of the specified trigger source. An environment error (see *table on page 60*) will be generated when an incompatible TRSL order is received — for example, if the current trigger type is EDGE, a slope of WIN or WOUT may not be specified.

The TRIG\_SLOPE? query returns the trigger slope of the selected source.

### COMMAND SYNTAX

<trig\_source> : TRig\_SLoPe <trig\_slope>  
<trig\_source> := {C1, C2, EX}  
<trig\_slope> := {NEG, POS, WOUT<sup>Ⓢ</sup>, WIN<sup>Ⓢ</sup>}

### QUERY SYNTAX

<trig\_source> : TRig\_SLoPe?

### RESPONSE FORMAT

<trig\_source> : TRig\_SLoPe <trig\_slope>



### AVAILABILITY

<trig\_source> := WOUT and WIN not available with EX as source.

### EXAMPLE

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

```
C2:TRSL NEG
```

### RELATED COMMANDS

TRIG\_COUPLING, TRIG\_DELAY, TRIG\_LEVEL, TRIG\_MODE, TRIG\_SELECT, TRIG\_SLOPE, TRIG\_WINDOW

# The Commands and Queries

## ACQUISITION

## TRIG\_WINDOW, TRWI Command/Query

### DESCRIPTION

The TRIG\_WINDOW command sets the window amplitude in volts on the specified trigger source. The window is centered around the Edge trigger level.

The TRIG\_WINDOW? query returns the current window amplitude.

### COMMAND SYNTAX

`<trig_source>:TRig_WIndow <value>`

`<source>: = {C1, C2}`

`<value> : = 0 to 1 V (maximum range)`

*Note: The suffix V is optional.*

### QUERY SYNTAX

`<trig_source> : TRig_WIndow?`

### RESPONSE FORMAT

`<trig_source> : TRig_WIndow <trig_level>`

### EXAMPLE

The following command adjusts the window size to +0.5 V on Channel 1:

```
C1:TRWI 0.5V
```

### RELATED COMMANDS

TRIG\_COUPLING, TRIG\_DELAY, TRIG\_LEVEL, TRIG\_MODE, TRIG\_SELECT, TRIG\_SLOPE

# The Commands and Queries

## MISCELLANEOUS

**\*TST?**  
Query

### DESCRIPTION

The \*TST? query performs an internal self-test, the response indicating whether the self-test has detected any errors. The self-test includes testing the hardware of all channels, the timebase and the trigger circuits.

Hardware failures are identified by a unique binary code in the returned <status> number. A "0" response indicates that no failures occurred.

### QUERY SYNTAX

\*TST?

### RESPONSE FORMAT

\*TST <status>

<status> : = 0 self-test successful

### EXAMPLE

The following causes a self-test to be performed:

\*TST?

Response message (if no failure):

\*TST 0

### RELATED COMMANDS

\*CAL

# The Commands and Queries

**DISPLAY**

**VERT\_MAGNIFY, VMAG**  
Command/Query

**DESCRIPTION**

The VERT\_MAGNIFY command vertically expands the specified trace. The command is executed even if the trace is not displayed.

The maximum magnification allowed depends on the number of significant bits associated with the data of the trace.

The VERT\_MAGNIFY? query returns the magnification factor of the specified trace.

This command is included for used with programs such as ScopeExplorer.

**COMMAND SYNTAX**

<trace> : **Vert\_MAGnify** <factor>

<trace> : = {**TA, TB, TC, TD**}

<factor> : = 4.0E-3 to 50 (maximum)

**QUERY SYNTAX**

<trace> : **Vert\_MAGnify?**

**RESPONSE FORMAT**

<trace> : **Vert\_MAGnify** <factor>

**EXAMPLE**

The following command enlarges the vertical amplitude of Trace A by a factor of 3.45 with respect to its original amplitude:

**TA:VMAG 3.45**

**RELATED COMMANDS**

VERT\_POSITION

# The Commands and Queries

**DISPLAY**

**VERT\_POSITION, VPOS**  
Command/Query

**DESCRIPTION**

The VERT\_POSITION command adjusts the vertical position of the specified 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 trace.

This command is included for use with programs such as ScopeExplorer.

**COMMAND SYNTAX**

<trace> : Vert\_POSITION <display\_offset>

<trace> : = {TA, TB, TC, TD}

<display\_offset> : = -5900 to +5900 DIV

*Note: The suffix DIV is optional. The limits depend on the current magnification factor, the number of grids on the display, and the initial position of the trace.*

**QUERY SYNTAX**

<trace> : Vert\_POSITION?

**RESPONSE FORMAT**

<trace> : Vert\_POSITION <display\_offset>

**EXAMPLE**

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

**TA:VPOS 3DIV**

**RELATED COMMANDS**

VERT\_MAGNIFY

# The Commands and Queries

## ACQUISITION

## VOLT\_DIV, VDIV Command/Query

### DESCRIPTION

The VOLT\_DIV command sets the vertical sensitivity in Volts/div. The VAB bit (bit 2) in the STB register (*see table on page 124*) 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}

<v\_gain> := *Refer to model's specifications.*

*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:

```
C1:VDIV 50MV
```

### RELATED COMMANDS

GAIN\_MODE, VOLT\_RANGE

# The Commands and Queries

## ACQUISITION

## VOLT\_RANGE, VRNG Command/Query

### DESCRIPTION

The VOLT\_RANGE command sets the full-scale range in volts. The VAB bit (bit 2) in the STB register (see *table on page 124*) is set if an out-of-range value is entered.

The VOLT\_RANGE query returns the full-scale range of the specified channel.

### COMMAND SYNTAX

<channel> : Volt\_RaNGe <v\_range>

<channel> := {C1, C2}

<v\_range> := *Refer to model specifications.*

*Note: The suffix V is optional.*

### QUERY SYNTAX

<channel> : Volt\_RaNGe?

### RESPONSE FORMAT

<channel> : Volt\_RaNGe <v\_range>



### AVAILABILITY

Command is only available if model supports adjustable range.

### EXAMPLE

The following sets the full-scale range of Channel 1 to 50 mV:

```
C1:VRNG 50MV
```

### RELATED COMMANDS

GAIN\_MODE, VOLT\_DIV

# The Commands and Queries

**STATUS**

**\*WAI  
Command**

**DESCRIPTION**

The \*WAI (WAIt to continue) command, required by the IEEE 488.2 standard, has no effect on the instrument, as the LSA1000 only starts processing a command when the previous command has been entirely executed.

**COMMAND SYNTAX**

\*WAI

**RELATED COMMANDS**

\*OPC



# The Commands and Queries

## ACQUISITION

## WAIT Command

### DESCRIPTION

The WAIT command prevents the instrument from analyzing new commands until the LSA1000 has completed the current acquisition.

### COMMAND SYNTAX

WAIT

### EXAMPLE

```
send: "TRMD SINGLE"  
loop {send: "ARM; WAIT; C1: PAVA? MAX"  
      read response  
      process response  
      }
```

This example finds the maximum amplitudes of several signals acquired one after another. ARM starts a new data acquisition. The WAIT command ensures that the maximum is evaluated for the newly acquired waveform.

"C1: PAVA? MAX" instructs the instrument to evaluate the maximum data value in the Channel 1 waveform.

### RELATED COMMANDS

\*TRG

# The Commands and Queries

## WAVEFORM TRANSFER

## WAVEFORM, WF Command/Query

### DESCRIPTION

A WAVEFORM command transfers a waveform from the controller to the LSA1000, whereas a WAVEFORM? query transfers a waveform from the LSA1000 to the controller.

The WAVEFORM command stores an external waveform back into the LSA1000's internal memory. 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).

*For further information on the structure of the waveform refer to Chapter 4.*

*Note 1: Only complete waveforms queried with "WAVEFORM? ALL" can be restored into the LSA1000.*

The WAVEFORM? query instructs the LSA1000 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 2: The format of the waveform data depends on the current settings specified by the last WAVEFORM\_SETUP command, the last COMM\_ORDER command, and the last COMM\_FORMAT command.*

### COMMAND SYNTAX

<memory> : WaveForm ALL <waveform\_data\_block>

<memory> : = {M1, M2, M3, M4}

<waveform\_data\_block> : = Arbitrary data block (see Chapter 5).

### QUERY SYNTAX

<trace> : WaveForm? <block>

<trace> : = {TA, TB, TC, TD, M1, M2, M3, M4, C1, C2}

<block> : = {DESC, TEXT, TIME, DAT1, DAT2, ALL}

*Note 3: If no parameter is given ALL will be assumed.*

# The Commands and Queries

## RESPONSE FORMAT

<trace> : WaveForm <block> , <waveform\_data\_block>

*Note 4: It may be convenient to disable the response header if the waveform is to be restored. Refer to command COMM\_HEADER for further details.*

## EXAMPLE

The following reads the block DAT1 from Memory 1:

**M1:WF? DAT1**

## RELATED COMMANDS

INSPECT, COMM\_FORMAT, COMM\_ORDER,  
FUNCTION\_STATE, TEMPLATE, WAVEFORM\_SETUP,  
WAVEFORM\_TEXT

# The Commands and Queries

## WAVEFORM TRANSFER

## WAVEFORM\_SETUP, WFSU Command/Query

### DESCRIPTION

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

Notation			
<b>FP</b>	first point	<b>NP</b>	number of points
<b>SP</b>	sparsing		

**Sparsing (SP):** The sparsing parameter defines the interval between data points. For example:

SP = 0            sends all data points  
SP = 1            sends all data points  
SP = 4            sends every 4th data point

**Number of points (NP):** The number of points parameter indicates how many points should be transmitted. For example:

NP = 0            sends all data points  
NP = 1            sends 1 data point  
NP = 50           sends a maximum of 50 data points  
NP = 1001        sends a maximum of 1001 data points

**First point (FP):** The first point parameter specifies the address of the first data point to be sent. For example:

FP = 0            corresponds to the first data point  
FP = 1            corresponds to the second data point  
FP = 5000        corresponds to data point 5001

The WAVEFORM\_SETUP? query returns the transfer parameters currently in use.

### COMMAND SYNTAX

**WaveForm\_SetUp** SP, <sparsing>, NP, <number>, FP, <point>

*Note 1: After power-on, all values are set to 0 (i.e. entire waveforms will be transmitted without sparsing).*

*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.*

# *The Commands and Queries*

**QUERY SYNTAX**

WaveForm\_SetUp?

**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:

**WFSU SP,3,FP,200**

**RELATED COMMANDS**

INSPECT, WAVEFORM, TEMPLATE

# The Commands and Queries

## WAVEFORM TRANSFER

## WAVEFORM\_TEXT, WFTX Command/Query

<b>DESCRIPTION</b>	<p>The WAVEFORM_TEXT command is used to document the conditions under which a waveform has been acquired. The text buffer is limited to 160 characters.</p> <p>The WAVEFORM_TEXT? query returns the text section of the specified trace.</p>
<b>COMMAND SYNTAX</b>	<p>&lt;trace&gt; : WaveForm_Text '&lt;text&gt;'</p> <p>&lt;trace&gt; := {TA, TB, TC, TD, M1, M2, M3, M4, C1, C2}</p> <p>&lt;text&gt; := An ASCII message (max. 160 characters long)</p>
<b>QUERY SYNTAX</b>	<p>&lt;trace&gt; : WaveForm_Text?</p>
<b>RESPONSE FORMAT</b>	<p>&lt;trace&gt; : WaveForm_Text "&lt;text&gt;"</p>
<b>EXAMPLE</b>	<p>The following documents Trace A (TA):</p> <p><b>TA:WFTX 'Averaged pressure signal. Experiment carried out Jan.15, 98'</b></p>
<b>RELATED COMMANDS</b>	<p>INSPECT, WAVEFORM, TEMPLATE</p>

# The Commands and Queries

**DISPLAY**

**XY\_ASSIGN?, XYAS?**

Query

**DESCRIPTION**

The XY\_ASSIGN? query returns the traces currently assigned to the XY display. If there is no trace assigned to the X-axis and/or the Y-axis the value UNDEF will be returned instead of the trace name.

This command is included for use with programs such as ScopeExplorer.

**QUERY SYNTAX**

**XY\_Assign?**

**RESPONSE FORMAT**

**XY\_Assign** <X\_source>,<Y\_source>

<X\_source> := {UNDEF, TA, TB, TC, TD, C1, C2}

<Y\_source> := {UNDEF, TA, TB, TC, TD, C1, C2}

**EXAMPLE**

The following query finds the traces assigned to the X-axis and the Y-axis respectively:

**XYAS?**

Example of response message:

**XYAS C1,C2**

**RELATED COMMANDS**

TRACE

# The Commands and Queries

## **CURSOR**

## **XY\_CURSOR\_ORIGIN, XYCO** Command/Query

### **DESCRIPTION**

The XY\_CURSOR\_ORIGIN command sets the position of the origin for XY absolute cursor measurements.

Absolute cursor values may be measured either with respect to the point (0,0) volts (OFF) or with respect to the center of the XY grid (ON).

The XY\_CURSOR\_ORIGIN query returns the current assignment of the origin for absolute cursor measurements.

### **COMMAND SYNTAX**

`XY_Cursor_Origin <mode>`

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

### **QUERY SYNTAX**

`XY_Cursor_Origin?`

### **RESPONSE FORMAT**

`XY_Cursor_Origin <mode>`

### **EXAMPLE**

The following command sets the origin for absolute cursor measurements to the center of the XY grid.

`XYCO ON`

### **RELATED COMMANDS**

`XY_CURSOR_VALUE`



# The Commands and Queries

**CURSOR**

**XY\_CURSOR\_SET, XYCS**  
Command/Query

## DESCRIPTION

The XY\_CURSOR\_SET command allows the user to position any one of the six independent XY voltage cursors at a given location. The positions of the cursors can be modified or queried even if the required cursor is not currently displayed or if the XY display mode is OFF.

The XY\_CURSOR\_SET? query indicates the current position of the cursor(s).

The CURSOR\_SET command is used to position the time cursors.

Notation	
<b>XABS</b>	vertical absolute on X axis
<b>XREF</b>	vertical reference on X axis
<b>XDIF</b>	vertical difference on X axis
<b>YABS</b>	vertical absolute on Y axis
<b>YREF</b>	vertical reference on Y axis
<b>YDIF</b>	vertical difference on Y axis

## COMMAND SYNTAX

**XY\_Cursor\_Set**

<cursor>, <position>[, <cursor>, <position>, ...

<cursor>, <position>]

<cursor> := {XABS, XREF, XDIF, YABS, YREF, YDIF}

<position> := -4 to 4 DIV

*Note 1: The suffix DIV is optional.*

*Note 2: Parameters are grouped in pairs. The first of the pair names the cursor to be modified, whilst the second indicates its new value. Pairs may be given in any order and may be restricted to those items to be changed.*

## QUERY SYNTAX

**XY\_Cursor\_Set?** [<cursor>, ...<cursor>]

<cursor> := {XABS, XREF, XDIF, YABS, YREF, YDIF, ALL}

*Note: If <cursor> is not specified, ALL will be assumed.*

# The Commands and Queries

## RESPONSE FORMAT

**XY\_Cursor\_Set**  
<cursor>,<position>[,<cursor>,<position>... ,  
<cursor>,<position>]

## EXAMPLE

The following command positions the XREF and YDIF at +3 DIV and -2 DIV respectively.

**XYCS XREF,3DIV,YDIF,-2DIV**

## RELATED COMMANDS

XY\_CURSOR\_VALUE, CURSOR\_MEASURE, CURSOR\_SET

# The Commands and Queries

**CURSOR**

**XY\_CURSOR\_VALUE?, XYCV?**

Query

## DESCRIPTION

The XY\_CURSOR\_VALUE? query returns the current values of the X versus Y cursors. The X versus Y trace does not need to be displayed to obtain these parameters, but valid sources must be assigned to the X and Y axes.

Notation	
<cursor type> := [HABS, HREL, VABS, VREL]	
<cursor type>_X	X
<cursor type>_Y	Y
<cursor type>_RATIO	+ Y/+ X
<cursor type>_PROD	+ Y*+ X
<cursor type>_ANGLE	arc tan(+ Y/+ X)
<cursor type>_RADIUS	sqrt(+ X*+ X + + Y*+ Y)

## QUERY SYNTAX

XY\_Cursor\_Value? [<parameter>,...<parameter>]

<parameter> := {HABS\_X, HABS\_Y, HABS\_RATIO, HABS\_PROD, HABS\_ANGLE, HABS\_RADIUS, HREL\_X, HREL\_Y, HREL\_RATIO, HREL\_PROD, HREL\_ANGLE, HREL\_RADIUS, VABS\_X, VABS\_Y, VABS\_RATIO, VABS\_PROD, VABS\_ANGLE, VABS\_RADIUS, VREL\_X, VREL\_Y, VREL\_RATIO, VREL\_PROD, VREL\_ANGLE, VREL\_RADIUS, ALL}

*Note: If <parameter> is not specified or equals ALL, all the measured cursor values are returned. If the value of a cursor could not be determined in the current environment, the value UNDEF will be returned. If no trace has been assigned to either the X axis or the Y axis, an environment error will be generated.*

## RESPONSE FORMAT

XY\_Cursor\_Value <parameter>,<value>[, ...<parameter>,<value>]

<value> := A decimal value or UNDEF

# *The Commands and Queries*

## **EXAMPLE**

The following query reads the ratio of the absolute horizontal cursor, the angle of the relative horizontal cursor, and the product of the absolute vertical cursors:

```
XYCV? HABS_RATIO,HREL_ANGLE,VABS_PROD
```

## **RELATED COMMANDS**

```
CURSOR_MEASURE, CURSOR_VALUE,  
XY_CURSOR_ORIGIN
```

# The Commands and Queries

## DISPLAY

## XY\_DISPLAY, XYDS Command/Query

### DESCRIPTION

The XY\_DISPLAY command enables or disables the XY display mode. When off, the scope is in standard display mode.

The XY\_DISPLAY? query returns the current mode of the XY display.

### COMMAND SYNTAX

`XY_Display <mode>`

### QUERY SYNTAX

`XY_Display?`

### RESPONSE FORMAT

`XY_Display <mode>`

### EXAMPLE

The following turns the XY display ON:  
`XYDS ON`

### RELATED COMMANDS

GRID

# The Commands and Queries

**DISPLAY**

**XY\_SATURATION, XYSA**  
Command/Query

**DESCRIPTION**

The XY\_SATURATION command sets the level at which the color spectrum of the persistence display is saturated in XY display mode. The level is specified in terms of percentage (PCT) of the total persistence data map population. A level of 100 PCT corresponds to the color spectrum being spread across the entire depth of the persistence data map. At lower values, the spectrum will saturate (brightest value) at the specified percentage value. The PCT is optional.

The response to the XY\_SAT? query indicates the saturation level of the persistence data maps.

**COMMAND SYNTAX**

**XY\_SAturation** <trace>,<value> [<trace>,<value>]

<trace> := { C1, C2, TA, TB, TC, TD, ALL}

<value> := 0 to 100 PCT

*Note: The suffix PCT is optional.*

**QUERY SYNTAX**

**XY\_SAturation?**

**RESPONSE FORMAT**

**XY\_SAturation** <trace>,<value>

**EXAMPLE**

The following sets the saturation level of the XY persistence data map for channel 3 to be 60%, i.e. 60% of the data points will be displayed with the color spectrum, with the remaining 40% saturated in the brightest color:

**XYSa C3,60**

**RELATED COMMANDS**

PERSIST\_SAT