Understanding Waveforms

This chapter covers the reading and writing of waveforms in remote control, and attempts to explain their structure and content.

Basic Structure

Waveforms can be divided into two basic entities. One is the basic data array: the raw data values from the LSA1000's ADCs (Analog–to–Digital Converters) in the acquisition. The other is the accompanying descriptive information, such as vertical and horizontal scale, and time of day, necessary for a full understanding of the information contained in the waveform.

This information can be accessed by remote control using the INSPECT? Query, which interprets it in an easily understood ASCII text form. It can be more rapidly transferred using the WAVEFORM? query, or written back into the instrument with the WAVEFORM command.

The LSA1000 itself contains a data structure, or *template*, which provides a detailed description of how the waveform's information is organized.

Waveforms can also be stored in pre-formatted ASCII output, for popular spreadsheets and math processing packages, using the STORE and STORE_SETUP commands.

Waveform Template

This gives a detailed description of the form and content of the logical data blocks of a waveform, and is provided as a reference. Although a sample template is given elsewhere in this manual (*see Appendix A*), it is suggested that the TEMPLATE? query and the actual instrument template be used. The template may change as the instrument's firmware is enhanced, and it will help provide backward compatibility for the interpretation of waveforms.

Logical Data Blocks

A waveform normally contains a waveform descriptor block and a data array block. However, in more complicated cases, one or more other blocks will be present.

- Waveform Descriptor block (WAVEDESC): This block includes all the information necessary to reconstitute the display of the waveform from the data. This includes:
  - hardware settings at the time of acquisition
  - the exact time of the event
  - the kinds of processing that have been performed
  - the name and serial number of the instrument
  - the encoding format used for the data blocks
  - miscellaneous constants.
- Optional User-provided Text block (USERTEXT): The WFTX command can be used to put a title or description of a waveform into this block. The WFTX? query command gives an alternative way to read it. This text block can hold up to 160 characters. They can be displayed in the TEXT + TIMES status menu as four lines of 40 characters.
- First data array block (SIMPLE or DATA_ARRAY_1): This is the basic integer data of the waveform. It can be raw or corrected ADC data or the integer result of waveform processing.
- Second data array block (DATA_ARRAY_2): This second data array is needed to hold the results of processing functions such as the Extrema (WP01 option) or Complex FFT (WP02 option). In such cases, the data arrays contain:

	Extrema	FFT
DATA_ARRAY_1	Roof trace	Real part
DATA_ARRAY_2	Floor trace	Imaginary part

***Note:** The Template also describes an array named DUAL. This is simply a way to allow the INSPECT? command to examine the two data arrays together.*

Using the INSPECT? Query

The query INSPECT? is a simple way to examine the contents of a waveform in remote control.

Usable on both the data and descriptive parts, its most basic form is:

INSPECT? "name**"**

where the template gives the name of a descriptor item or data block. The answer is returned as a single string, but may span many lines. Some typical dialogue:

question **C1:INSPECT? "VERTICAL_OFFSET"**
response **C1:INSP "VERTICAL_OFFSET: 1.5625e-03"**
question **C1:INSPECT? "TRIGGER_TIME"**
response **C1:INSP "TRIGGER_TIME: Date = FEB 17, 1994, Time = 4: 4:29.5580"**

INSPECT? can also be used to provide a readable translation of the full waveform descriptor block with:

INSPECT? "WAVEDESC"

The template dump will give details of the interpretation of each of the parameters. INSPECT? is also used to examine the measured data values of a waveform using:

INSPECT? "SIMPLE"

For example, for an acquisition with 52 points:

```
INSPECT? "SIMPLE"
C1:INSP "
 0.0005225  0.0006475 -0.00029   -0.000915   2.25001E-0 0.000835
 0.0001475 -0.0013525 -0.00204   -4E-05      0.0011475  0.0011475
-0.000915  -0.00179   -0.0002275  0.0011475  0.001085  -0.00079
-0.00179   -0.0002275  0.00071    0.00096   -0.0003525 -0.00104
 0.0002725  0.0007725  0.00071   -0.0003525 -0.00129   -0.0002275
 0.0005225  0.00046   -0.00104   -0.00154    0.0005225  0.0012725
 0.001335  -0.0009775 -0.001915  -0.000165   0.0012725  0.00096
-0.000665  -0.001665  -0.0001025  0.0010225  0.00096   -0.0003525
-0.000915   8.50001E-0 0.000835   0.0005225
 "
```

These numbers are the fully converted measurements in volts. Of course, when the data block contains thousands of items the string will contain a great many lines.

Depending on the application, the data may be preferred in its raw form as either a BYTE (8 bits) or a WORD (16 bits) for each data value. In this case the relations given below must be used in association with WAVEFORM? to interpret the measurement. It might then say:

```
INSPECT? "SIMPLE",BYTE
```

The examination of data values for waveforms with two data arrays can be performed as follows:

INSPECT? "DUAL"	to get pairs of data values on a single line
INSPECT? "DATA_ARRAY_1"	to get the values of the first data array
INSPECT? "DATA_ARRAY_2"	to get the values of the second data array.

Finally…

INSPECT? is useful, but it is also a rather verbose way to send information. As a query form only, INSPECT? cannot be used to send a waveform back into the LSA1000. Users who require this capability or speed or both should instead use the WAVEFORM query or commands. It is possible to examine just a part of the waveform or a sparsed form of it, using the WAVEFORM_SETUP command covered later in this chapter.

Programmers might find it convenient, too, to combine the capabilities of the inspect facility with the waveform query command in order to construct files containing a plain text version of the waveform descriptor together with the full waveform in a format suitable for retransmission to the instrument. This can be done for a waveform in a memory location by sending the command

```
MC:INSPECT? "WAVEDESC";WAVEFORM?
```

and putting the response directly into a disk file.

WAVEFORM?, Related Commands, and Blocks

Using the WAVEFORM? query is an effective way to transfer waveform data using the block formats defined in the IEEE-488.2 standard. Responses can then be downloaded back into the instrument using the WAVEFORM command.

All of a waveform's logical blocks can be read with the single query:

```
C1:WAVEFORM?
```

This is the preferred form for most applications due to its completeness. Time and space are the advantages when reading many waveforms with the same acquisition conditions, or when the interest is only in large amounts of raw integer data.

And any single block can be chosen for reading with a query such as:

```
C1:WAVEFORM? DAT1
```

The description In the *System Commands* section provides the various block names.

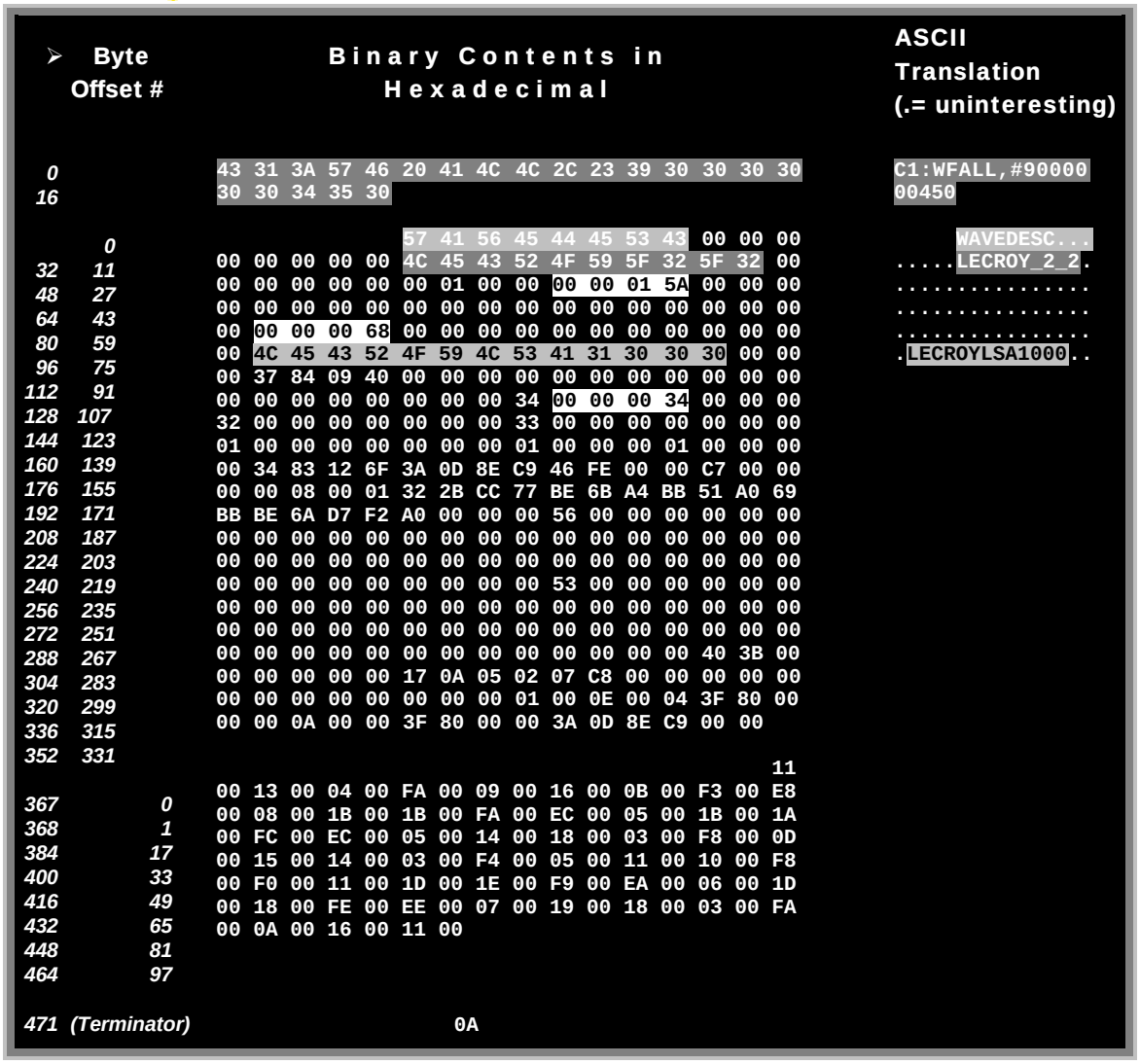
Note: A waveform query response can easily be a block containing over 16 million bytes if it is in binary format and twice as much if the HEX option is used.

Interpreting the Waveform Descriptor

The binary response to a query of the form:

`C1:WAVEFORM?` or **`C1:WAVEFORM? ALL`**

can be placed in a disk file and then dumped to show the following hexadecimal and ASCII form:



➢ Byte Offset #		Binary Contents in Hexadecimal	ASCII Translation (.= uninteresting)
0		43 31 3A 57 46 20 41 4C 4C 2C 23 39 30 30 30 30	C1:WFALL,#90000
16		30 30 34 35 30	00450
	0	57 41 56 45 44 45 53 43 00 00 00	WAVEDESC...
32	11	00 00 00 00 00 4C 45 43 52 4F 59 5F 32 5F 32 00	.....LECROY_2_2.
48	27	00 00 00 00 00 00 01 00 00 00 00 01 5A 00 00 00	..............Z...
64	43	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....
80	59	00 00 00 00 68 00 00 00 00 00 00 00 00 00 00 00	....h...........
96	75	00 4C 45 43 52 4F 59 4C 53 41 31 30 30 30 00 00	.LECROYLSA1000..
112	91	00 37 84 09 40 00 00 00 00 00 00 00 00 00 00 00	.7..@...........
128	107	00 00 00 00 00 00 00 00 34 00 00 00 34 00 00 00	........4...4...
144	123	32 00 00 00 00 00 00 00 33 00 00 00 00 00 00 00	2.......3.......
160	139	01 00 00 00 00 00 00 00 01 00 00 00 01 00 00 00	....
176	155	00 34 83 12 6F 3A 0D 8E C9 46 FE 00 00 C7 00 00	.4..o:...F......
192	171	00 00 08 00 01 32 2B CC 77 BE 6B A4 BB 51 A0 69	.....2+.w.k..Q.i
208	187	BB BE 6A D7 F2 A0 00 00 00 56 00 00 00 00 00 00	..j......V......
224	203	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....
240	219	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....
256	235	00 00 00 00 00 00 00 00 00 53 00 00 00 00 00 00	.........S......
272	251	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....
288	267	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....
304	283	00 00 00 00 00 00 00 00 00 00 00 00 00 40 3B 00	.............@;.
320	299	00 00 00 00 00 17 0A 05 02 07 C8 00 00 00 00 00	....
336	315	00 00 00 00 00 00 00 00 01 00 0E 00 04 3F 80 00	.............?..
352	331	00 00 0A 00 00 3F 80 00 00 3A 0D 8E C9 00 00	.....?...:....
			11
367	0	00 13 00 04 00 FA 00 09 00 16 00 0B 00 F3 00 E8	
368	1	00 08 00 1B 00 1B 00 FA 00 EC 00 05 00 1B 00 1A	
384	17	00 FC 00 EC 00 05 00 14 00 18 00 03 00 F8 00 0D	
400	33	00 15 00 14 00 03 00 F4 00 05 00 11 00 10 00 F8	
416	49	00 F0 00 11 00 1D 00 1E 00 F9 00 EA 00 06 00 1D	
432	65	00 18 00 FE 00 EE 00 07 00 19 00 18 00 03 00 FA	
448	81	00 0A 00 16 00 11 00	
464	97		
471	(Terminator)	0A	

Here, in order to illustrate the contents of the logical blocks, the relevant parts (see explanations next page) have been separated. In addition, to facilitate counting, the corresponding Byte Offset numbering has been restarted each time a new block begins. The ASCII translation, only part of which is shown, has been similarly split and highlighted, showing how its parts correspond to the binary contents, highlighted in the same fashion.

On the preceding page… The first 10 bytes translate into ASCII and resemble the simple beginning of a query response. These are followed by the string **"#9000000450**", the beginning of a binary block in which nine ASCII integers are used to give the length of the block (450 bytes). The waveform itself starts immediately after this, at Byte number 21. The very first byte is Byte #0, as it is for the first byte in each block (at the head of each of the three Byte Offset columns illustrated).

The first object is a DESCRIPTOR_NAME, a string of 16 characters with the value **WAVEDESC**.

Then, 16 bytes after the beginning of the descriptor (or at Byte #37, counting from the very start and referring to the numbers in the first Byte Offset column), we find the beginning of the next string: the TEMPLATE_NAME with the value **LECROY_2_2**.

Several other parameters follow. The INSTRUMENT_NAME, 76 bytes from the descriptor start (Byte #97), is easily recognizable. On the preceding line, at 38 bytes after the descriptor (Byte #59), a four-byte-long integer gives the length of the descriptor:

WAVE_DESCRIPTOR = 00 00 01 5A (hex) = 346.

At 60 bytes from the descriptor start (or Byte #81) we find another four-byte integer giving the length of the data array:

WAVE_ARRAY_1 = 00 00 00 68 (hex) = 104.

And at 116 bytes after the descriptor (Byte #137), yet another four-byte integer gives the number of data points:

WAVE_ARRAY_COUNT = 00 0000 34 (hex) = 52.

Now we know that the data will start at 346 bytes from the descriptor's beginning (Byte #367), and that each of the 52 data points will be represented by two bytes. The waveform has a total length of 346 + 104, which is the same as the ASCII string indicated at the beginning of the block. The final 0A at Byte #471 is the NL character associated with the message terminator <NL><EOI>.

As the example was taken using an instrument with an eight-bit ADC, we see the eight bits followed by a 0 byte for each data point. However, for many other kinds of waveform this second byte will not be zero and will contain significant information. The

data is coded in signed form (two's complement) with values ranging from –32768 = 8000 (hex) to 32767 = 7FFF (hex). If we had chosen to use the BYTE option for the data format the values would have been signed integers in the range –128 = 80 (hex) to 127 = 7F (hex). These ADC values are mapped to the display grid in the following way:

- 0 is located on the grid's center axis
- 127 (BYTE format) or 32767 (WORD format) is located at the top of the grid
- –128 (BYTE format) or –32768 (WORD format) is located at the bottom of the grid.

Interpreting Vertical Data

Now that we know how to decipher the data it would be useful to convert it to the appropriate measured values.

The vertical reading for each data point depends on the vertical gain and the vertical offset given in the descriptor. For acquisition waveforms this corresponds to the volts/div and voltage offset selected after conversion for the data representation being used. The template tells us that the vertical gain and offset can be found at bytes 156 and 160 respective of the descriptor start and that they are stored as floating point numbers in the IEEE 32-bit format. An ASCII string giving the vertical unit is to be found in VERTUNIT, Byte #196. The vertical value is given by the relationship:

value = VERTICAL_GAIN × data – VERTICAL_OFFSET

In the case of the data shown above we find:

VERTICAL_GAIN =	2.44141e–07 from the floating point number 3483 126f at byte 177
VERTICAL_OFFSET =	0.00054 from the floating point number 3A0D 8EC9 at byte 181
VERTICAL_UNIT =	V = volts from the string 5600 ... at byte 217

and therefore:

since data[4] = FA00 = 64000 from the hexadecimal word FA00 at byte 371. Overflows the maximum. 16 bit value of 32767, so must be a negative value. Using the two's complement conversion $64000 - 2^{16} = -1536$

value[4] = –0.000915 V as stated in the inspect command.

If the computer or the software available is not able to understand the IEEE floating point values, a description is to be found in the template.

The data values in a waveform may not all correspond to measured points. FIRST_VALID_PNT and LAST_VALID_PNT give the necessary information. The descriptor also records the SPARSING_FACTOR, the FIRST_POINT, and the SEGMENT_INDEX to aid interpretation if the options of the WAVEFORM_SETUP command have been used.

For waveforms such as the extrema and the complex FFT there will be two arrays — one after the other — for the two of the result.

Calculating a Data Point's Horizontal Position

Each vertical data value has a corresponding horizontal position, usually measured in time or frequency units. Every data value has a position, i, in the original waveform, with i = 0 corresponding to the first data point acquired. The descriptor parameter HORUNIT gives a string with the name of the horizontal unit.

Single-Sweep Waveforms

$x[i] = HORIZ\_INTERVAL \times i + HORIZ\_OFFSET$

For acquisition waveforms this time is from the trigger to the data point in question. It will be different from acquisition to acquisition since the HORIZ_OFFSET is measured for each trigger.

In the case of the data shown above this means:

HORIZ_INTERVAL = 1e–08 from the floating point number 322b cc77 at byte 194

HORIZ_OFFSET =	–5.149e–08 from the double precision floating point number be6b a4bb 51a0 69bb at byte 198
HORUNIT = S =	seconds from the string 5300 ... at byte 262.

This gives:

x[0] = –5.149e–08 S
x[1] = –4.149e–08 S.

WAVEFORM Commands

Waveforms that have been read in their entirety with the WAVEFORM? query can be sent back into the instrument using WAVEFORM and other, related commands. Since the descriptor contains all of the necessary information, care need not be taken with any of the communication format parameters. The instrument can learn all it needs to know from the waveform.

***Note:** Waveforms can only be sent back to memory traces (M1, M2, M3, M4). This means possibly removing or changing the prefix (C1 or CHANNEL_1) in the response to the WF? query. See the System Commands for examples.*

When synthesizing waveforms for display or comparison, in order to ensure that the descriptor is coherent, read out a waveform of the appropriate size and then replace the data with the desired values.

There are many ways to use WAVEFORM and related commands to simplify or speed up work. Among them:

- **Partial Waveform Readout:** The WAVEFORM_SETUP command allows specification of a short part of a waveform for readout, as well as selection of a sparsing factor for reading only every n'th data point.
- **Byte Swapping:** The COMM_ORDER command allows the swapping of the two bytes of data presented in 16-bit word format (can be in the descriptor or in the the data/time arrays), when sending the data over the remote-control ports. This allows easier data interpretation, depending on the computer system used:
  - Intel-based computers — the data should be sent with the LSB first, and the command should be CORD LO.

- Motorola-based computers — the data should be sent with the MSB first (CORD HI). This is the default at power-up.

- **Data Length, Block Format, and Encoding**: The COMM_FORMAT command gives control over these parameters. If the extra precision of the lower order byte of the standard data value is not needed, the BYTE option allows a saving of a factor of two on the amount of data to be transmitted or stored. If the computer being used is unable to read binary data, the HEX option allows a response form where the value of each byte is given by a pair of hexadecimal digits.

- **Data-Only Transfers**: The COMM_HEADER OFF mode enables a response to WF? DAT1 with the data only (the C1:WF DAT1 will disappear).

  If COMM_FORMAT OFF,BYTE,BIN has also been specified, the response will be mere data bytes (the #90000nnnnn will disappear).

High-Speed Waveform Transfer

Several important factors need to taken into account for achieving maximum continuous-data-transfer rates from server to client.

The single most important of these is the limiting of work done in the computer. This effectively means avoiding writing data to disk wherever possible, as well as minimizing operations such as per-data-point computations and reducing the number of calls to the IO system. Ways of doing this include:

- Reducing the number of points to be transferred and the number of data bytes per point. The pulse parameter capability and the processing functions can save a great deal of computing and a lot of data transfer time if employed creatively.
- Attempting to overlap waveform acquisition with waveform transfer. The LSA1000 is capable of transferring an already acquired or processed waveform after a new acquisition has been started. If the instrument is obliged to wait for triggers, overlapping waveform acquisition with waveform transfer will considerably increase the total time that the instrument will be able to acquire events (live time).

Example

The desirable type of command is:

`ARM; WAIT;C1:WF?` to wait for the event, transfer the data, and then start a new acquisition.

This line can be “looped” in the program as soon as it has finished reading the waveform.

4 Waveform Structure