26

DATA and SETup Commands

Introduction

The DATA and SETup commands are SYSTem commands that allow you to send and receive block data between the HP 1660 series and a controller. Use the DATA instruction to transfer acquired timing and state data, and the SETup instruction to transfer instrument configuration data. This is useful for:

- Re-loading to the logic analyzer
- Processing data later
- Processing data in the controller

This chapter explains how to use these commands.

The format and length of block data depends on the instruction being used, the configuration of the instrument, and the amount of acquired data. The length of the data block can be up to 409,760 bytes in the HP 1660A.

The SYSTem:DATA section describes each part of the block data as it will appear when used by the DATA instruction. The beginning byte number, the length in bytes, and a short description is given for each part of the block data. This is intended to be used primarily for processing of data in the controller.

Do not change the block data in the controller if you intend to send the block data back into the logic analyzer for later processing. Changes made to the block data in the controller could have unpredictable results when sent back to the logic analyzer.

Data Format

To understand the format of the data within the block data, there are four important things to keep in mind.

- Data is sent to the controller in binary form.
- Each byte, as described in this chapter, contains 8 bits.
- The first bit of each byte is the MSB (most significant bit).
- Byte descriptions are printed in binary, decimal, or ASCII depending on how the data is described.

For example, the first ten bytes that describe the section name contain a total of 80 bits as follows:

```
           Byte 1                                                             Byte 10
         ┌────┴────┐                                                        ┌────┴────┐
Binary   0100 0100 0100 0001 0101 0100 0100 0001 0010 0000 ...  0010 0000
         |         |
        MSB       LSB

Decimal  68 65 84 65 32 32 32 32 32 32

  ASCII  DATA space space space space space space
```

:SYSTem:DATA

Command	`:SYSTem:DATA <block_data>`

The SYSTem:DATA command transmits the acquisition memory data from the controller to the HP 1660-series logic analyzer.

The block data consists of a variable number of bytes containing information captured by the acquisition chips. The information will be in one of three formats, depending on the type of data captured. The three formats are glitch, transitional, conventional timing or state. Each format is described in the "Acquisition Data Description" section later in this chapter. Since no parameter checking is performed, out-of-range values could cause instrument lockup; therefore, care should be taken when transferring the data string into the logic analyzer.

The <block_data> parameter can be broken down into a <block_length_specifier> and a variable number of <section>'s.

The <block_length_specifier> always takes the form #8DDDDDDDD. Each D represents a digit (ASCII characters "0" through "9"). The value of the eight digits represents the total length of the block (all sections). For example, if the total length of the block is 14522 bytes, the block length specifier would be "#800014522".

Each <section> consists of a <section header> and <section data>. The <section data> format varies for each section. For the DATA instruction, there is only one <section>, which is composed of a data preamble followed by the acquisition data. This section has a variable number of bytes depending on configuration and amount of acquired data.

`<block_data>`	`<block_length_specifier><section>`
`<block_length_specifier>`	`#8<length>`
`<length>`	The total length of all sections in byte format (must be represented with 8 digits)
`<section>`	`<section header><section data>`
`<section_header>`	16 bytes, described in "Section Header Description," on page 26-6.
`<section_data>`	Format depends on the specific section.

Example

```
OUTPUT XXX;":SYSTEM:DATA" <block_data>
```

The total length of a section is 16 (for the section header) plus the length of the section data. So when calculating the value for `<length>`, don't forget to include the length of the section headers.

Query

```
:SYSTem:DATA?
```

The SYSTem:DATA query returns the block data to the controller. The data sent by the SYSTem:DATA query reflect the configuration of the machines when the last run was performed. Any changes made since then through either front-panel operations or programming commands do not affect the stored configuration.

Returned Format

```
[:SYSTem:DATA] <block_data><NL>
```

Example

See "Transferring the logic analyzer acquired data" on page 27-17 in chapter 27, "Programming Examples" for an example.

Section Header Description

The section header uses bytes 1 through 16 (this manual begins counting at 1; there is no byte 0). The 16 bytes of the section header are as follows:

Byte Position	
1	10 bytes - Section name ("DATA space space space space space space" in ASCII for the DATA instruction).
11	1 byte - Reserved
12	1 byte - Module ID (0010 0000 binary or 32 decimal for the HP 1660 series)
13	4 bytes - Length of section in number of bytes that, when converted to decimal, specifies the number of bytes contained in the section.

Section Data

For the SYSTem:DATA command, the <section data> parameter consists of two parts: the data preamble and the acquisition data. These are described in the following two sections.

Data Preamble Description

The block data is organized as 160 bytes of preamble information, followed by a variable number of bytes of data. The preamble gives information for each analyzer describing the amount and type of data captured, where the trace point occurred in the data, which pods are assigned to which analyzer, and other information. The values stored in the preamble represent the captured data currently stored in this structure and not the current analyzer configuration. For example, the mode of the data (bytes 21 and 49) may be STATE with tagging, while the current setup of the analyzer is TIMING.

The preamble (bytes 17 through 176) consists of the following 160 bytes:

17	2 bytes - Instrument ID (always 1660 decimal for HP 1660 series)
19	1 byte - Revision Code
20	1 byte - number of acquisition chips used in last acquisition

The next 40 bytes are for Analyzer 1 Data Information.

Byte Position

21 1 byte - Machine data mode, one of the following decimal values:
- −1 = off
- 0 = state data without tags
- 1 = state data with each chip assigned to a machine (2kB memory) and either time or state tags
- 2 = state data with unassigned pod used to store tag data (4kB memory)
- 8 = state data at half channel (8kB memory with no tags)
- 10 = conventional timing data at full channel
- 11 = transitional timing data at full channel
- 12 = glitch timing data
- 13 = conventional timing data at half channel
- 14 = transitional timing data at half channel

22 1 byte - Unused.

23 2 bytes - List of pods in this analyzer, where a binary 1 indicates that the corresponding pod is assigned to this analyzer

bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8
unused	unused	always 1	unused	unused	unused	unused	Pod 8[1]

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Pod 7[1]	Pod 6[2]	Pod 5[2]	Pod 4[3]	Pod 3[3]	Pod 2	Pod 1	unused

1 – also unused in the HP 1661A, HP 1662A, and HP 1663A
2 – also unused in the HP 1662A and HP 1663A
3 – also unused in the HP 1663A

Example

`xx10 0000 0001 111x` indicates pods 1 through 4 are assigned to this analyzer (x = unused bit).

25 1 byte - This byte returns which chip is used to store the time or state tags when an unassigned pod is available to store tag data. This chip is available in state data mode with an unassigned pod and state or time tags on. Byte 21 = 2 in this mode.

`Byte Position`

26 1 byte - Master chip for this analyzer. This decimal value returns which chip's time tag data is valid in a non-transitional mode; for example, state with time tags.

5 - pods 1 and 2	2 - pods 7 and 8[3]
4 - pods 3 and 4[1]	1 - unused
3 - pods 5 and 6[2]	0 - unused
	– 1 - no chip

1 – also unused in the HP 1663A
2 – also unused in the HP 1662A and HP 1663A
3 – also unused in the HP 1661A, HP 1662A, and HP 1663A

27 6 bytes - Unused

33 8 bytes - A decimal integer representing sample period in picoseconds (timing only).

Example The following 64 bits in binary would equal 8,000 picoseconds or, 8 nanoseconds:

```
00000000 00000000 00000000 00000000 00000000 00000000 00011111 01000000
```

41 8 bytes - Unused

49 1 byte - Tag type for state only in one of the following decimal values:
0 = off
1 = time tags
2 = state tags

50 1 byte - Unused

51 8 bytes - A decimal integer representing the time offset in picoseconds from when this analyzer is triggered and when this analyzer provides an output trigger to the IMB or port out. The value for one analyzer is always zero and the value for the other analyzer is the time between the triggers of the two analyzers.

59 2 bytes - Unused

Byte Position

61 40 bytes - The next 40 bytes are for Analyzer 2 Data Information. They are organized in the same manner as Analyzer 1 above, but they occupy bytes 61 through 100.

101 26 bytes - Number of valid rows of data (starting at byte 177) for each pod. The 26 bytes of this group are organized as follows:
Bytes 1 and 2 - Unused
Bytes 3 and 4 - Unused.
Bytes 5 and 6 - Unused.
Bytes 7 and 8 - Unused.
Bytes 9 and 10 - Unused.
Bytes 11 and 12 contain the number of valid rows of data for pod 8 of the HP 1660A only. Unused in the other HP 1660-series logic analyzers.
Bytes 13 and 14 contain the number of valid rows of data for pod 7 of the HP 1660A only. Unused in the other HP 1660-series logic analyzers
Bytes 15 and 16 contain the number of valid rows of data for pod 6 of the HP 1660A and HP 1661A only.
Bytes 17 and 18 contain the number of valid rows of data for pod 5 of the HP 1660A and HP 1661A only.
Bytes 19 and 20 contain the number of valid rows of data for pod 4 of the HP 1660A, HP 1661A, and HP 1662A only.
Bytes 21 and 22 contain the number of valid rows of data for pod 3 of the HP 1660A, HP 1661A, and HP 1662A only.
Bytes 23 and 24 contain the number of valid rows of data for pod 2 of all models of the HP1660-series logic analyzers.
Bytes 25 and 26 contain the number of valid rows of data for pod 1 of all models of the HP1660-series logic analyzers.

Byte Position

127 26 bytes - Row of data containing the trigger point. This byte group is organized in the same way as the data rows (starting at byte 101 above). These binary numbers are base zero numbers which start from the first sample stored for a specific pod. For example, if bytes 151 and 152 contained a binary number with a decimal equivalent of +1018, the data row having the trigger is the 1018th data row on pod 1. There are 1018 rows of pre-trigger data as shown below.

row 0
row 1
.
.
.
row 1017
row 1018 – trigger row

153 24 bytes - Unused

Acquisition Data Description

The acquisition data section consists of a variable number of bytes depending on which logic analyzer you are using, the acquisition mode and the tag setting (time, state, or off). The data is grouped in 18-byte rows for the HP 1660A, in 14-byte rows for the HP 1661A, in 10-byte rows for the HP 1662A, and in 6-byte rows for the HP 1663A.

The number of rows for each pod is stored in byte positions 101 through 126. The number of bytes in each row can be determined by the value stored in byte position 20 which contains the number of acquisition chips in the instrument. For example, if the value in byte position 20 is 4, the instrument is an HP 1660A. Values 3, 2, and 1 represent the HP 1661A, 1662A, and 1663A respectively.

Byte Position

	clock lines	Pod 8[1]	Pod 7[1]	pod 6[2]	pod 5[2]	pod 4[3]	pod 3[3]	pod 2	pod 1[4]
177	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes
195	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes
•	•	•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•	•	•
(x)	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes

1 – unused in the HP 1661A, HP 1662A, and HP 1663A
2 – also unused in the HP 1662A and HP 1663 A
3 – also unused in the HP 1663A
4 – The headings are not a part of the returned data.

Row (x) is the highest number of valid rows specified by the bytes in byte positions 101 through 126 in all modes and when neither analyzer is in glitch mode. In the glitch mode, row (x) is the larger of:

1. The highest number of valid rows specified by the bytes in byte positions 101 through 126; or,
2. 2048 + the highest number of valid rows for the pods assigned to the timing analyzer, when one or more glitches are detected.

The clock-line bytes for the HP 1660A, which also includes 2 additional data lines (D), are organized as follows:

```
xxxx xxPN xxDD MLKJ
```

The clock-line bytes for the HP 1661A and HP 1662A are organized as follows:

```
xxxx xxxx xxxx MLKJ
```

The clock-line bytes for the HP 1663A are organized as follows:

```
xxxx xxxx xxxx xxKJ
```

Time Tag Data Description

The time tag data starts at the end of the acquired data. Each data row has an 8-byte time tag for each chip (2-pod set). The starting location of the time tag data is immediately after the last row of valid data (maximum data byte + 1). If an analyzer is in a non-transitional mode, the master chip (byte 26) is the only chip with valid time-tag data. The time tag data is a decimal integer representing time in picoseconds for both timing and state time tags. For state tags in the state analyzer, tag data is a decimal integer representing the number of states.

Time Tag Block (for the HP 1660A)

Byte 1 through 8 (64 bits starting with the MSB) - First sample tag for pods 1 and 2.

Byte 9 through 16 (64 bits starting with the MSB) - Second sample tag for pods 1 and 2.

.

.

.

Byte (w) through (w + 7) (64 bits starting with the MSB) - Last sample tag for pods 1 and 2.

Byte (w + 8) through (w + 15) (64 bits starting with the MSB) - First sample tag for pods 3 and 4.

Byte (w + 16) through (w + 23) (64 bits starting with the MSB) - Second sample tag for pods 3 and 4.

.

.

.

Byte (x) through (x+ 7) (64 bits starting with the MSB) - Last sample tag for pods 3 and 4.

Byte (x + 8) through (x + 15) (64 bits starting with the MSB) - First sample tag for pods 5 and 6.

Byte (x + 16) through (x + 23) (64 bits starting with the MSB) - Second sample tag for pods 5 and 6.

.

.

.

Byte (y) through (y+ 7) (64 bits starting with the MSB) - Last sample tag for pods 5 and 6.

Byte (y + 8) through (y + 15) (64 bits starting with the MSB) - First sample tag for pods 7 and 8.

Byte (y + 16) through (y + 23) (64 bits starting with the MSB) - Second sample tag for pods 7 and 8.

.

.

.

Byte (z) through (z+ 7) (64 bits starting with the MSB) - Last sample tag for pods 7 and 8.

Glitch Data Description

In the glitch mode, each pod has two bytes assigned to indicate where glitches occur in the acquired data. For each row of acquired data there will be a corresponding row of glitch data. The glitch data is organized in the same way as the acquired data. The glitch data is grouped in 18-byte rows for the HP 1660A. The number of rows is stored in byte positions 101 through 126. The starting byte of the glitch data is an absolute starting point regardless of the number of rows of acquired data.

A binary 1 in the glitch data indicates a glitch was detected. For example, if a glitch occurred on bit 1 of pod 8 in data row 1 of an HP 1660A, bytes 37043 and 37044 would contain:

```
 Byte 37043   Byte 37044
 ┌────┴────┐ ┌────┴────┐
 0000 0000 0000 0010
 |                 |
Bit 15            Bit 1
```

Byte Position

	clock lines	Pod 8[1]	Pod 7[1]	pod 6[2]	pod 5[2]	pod 4[3]	pod 3[3]	pod 2	pod 1[4]
37041	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes
37059	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes
•	•	•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•	•	•
(x)	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes

1 – unused in the HP 1661A, HP 1662A, and HP 1663A
2 – also unused in the HP 1662A and HP 1663 A
3 – also unused in the HP 1663A
4 – The headings are not a part of the returned data.

SYSTem:SETup

Command

```
:SYStem:SETup <block_data>
```

The SYStem:SETup command configures the logic analyzer module as defined by the block data sent by the controller. The length of the configuration data block can be up to 350,784 bytes in the HP 1660A.

There are four data sections which are always returned. These are the strings which would be included in the section header:

```
"CONFIG    "
"DISPLAY1  "
"BIG_ATTRIB"
"RTC_INFO  "
```

Additionally, the following sections may also be included, depending on what's available:

```
"SYMBOLS A "
"SYMBOLS B "
"INVASM A  "
"INVASM B  "
"COMPARE   "
```

With the exception of the RTC_INFO section, the block data is not described. However, the RTC_INFO section contains the real-time clock time of the acquired data in the data block. This time information can be meaningful to some measurements.

`<block_data>`	`<block_length_specifier><section>`
`<block_length_specifier>`	`#8<length>`
`<length>`	The total length of all sections in byte format (must be represented with 8 digits)
`<section>`	`<section_header><section_data>[<section_data>...]`
`<section_header>`	16 bytes in the following format: 10 bytes for the section name 1 byte reserved 1 byte for the module ID code (32 for the HP 1660-series logic analyzer) 4 bytes for the length of section data in number of bytes that, when converted to decimal, specifies the number of bytes contained in the section. The RTC_INFO section is described in the "RTC_INFO Section Description."
`<section_data>`	Format depends on the section.

The total length of a section is 16 (for the section header) plus the length of the section data. So when calculating the value for <length>, don't forget to include the length of the section headers.

Example `OUTPUT XXX;"SETUP" <block_data>`

Query `:SYStem:SETup?`

The SYStem:SETup query returns a block of data that contains the current configuration to the controller.

Returned Format `[:SYStem:SETup] <block_data><NL>`

Example See "Transferring the logic analyzer configuration" on page 27-14 in Chapter 27, "Programming Examples" for an example.

RTC_INFO Section Description

The RTC_INFO section contains the real time of the acquired data. Because the time of the acquired data is important to certain measurements, this section describes how to find the real-time clock data.

Because the number of sections in the SETup data block depends on the logic analyzer configuration, the RTC_INFO section will not always be in the same location within the block. Therefore, the section must be found by name. Once the section is found, you can find the time by using the description in the following section:

```
#8<block_length>...[<section_name><section_length>
<section_data>]...
```

`<block_length>`	Total length of all sections
`<section_name>`	10 bytes - Section name. **"RTC_INFO space space"**
`<section_length>`	4 bytes - Length of section. 8 bytes, decimal, for RTC_INFO section.
`<section_data>`	10 bytes - Contains the real-time clock data described as follows:
Byte Position	
1	1 byte - Year. A decimal integer that, when added to 1990, defines the year. For example, if this byte has a decimal value of 2, the year is 1992.
2	1 byte - Month. An integer from 1 to 12.
3	1 byte - Day. An integer from 1 to 31.
4	1 byte - Unused
5	1 byte - Hour. An integer from 1 to 23.
6	1 byte - Minute. An integer from 1 to 59.
7	1 byte - Second. An integer from 1 to 59.
8	1 byte - Unused.