icListen Streaming Telemetry

October 7, 2014

Ocean Sonics Ltd.
Hill House, 11 Lornevale Road,
Great Village, NS, B0M 1L0 Canada
Phone: +1 902 655 3000
www.OceanSonics.com

Table of Contents

Table of Figures

1 Introduction

This document applies to the **icListen AF/HF** v2.1 (Release 28) and below. If your **icListen AF/HF** is a higher firmware version than this, please contact Ocean Sonics for the latest revision of this document.

This document details the telemetry format for communicating with an **icListen** unit over the streaming channels. **icListen AF** and **icListen HF** support data streaming channels. All **icListen** models support a command & control channel.

Data may be streamed via the UDP or TCP network protocols. Each requested data type (wave data, FFT data, etc.) will be sent on a separate port. All stream messages are in network order (big endian), with the exception of the data body, which is the configured data format.

Data collection and processing settings cannot be configured, using the streaming channels. To accomplish this, the command and control channel should be used. For more information on the command and control channel telemetry, please see the *icListen Command & Control Telemetry* document.

2 Overview of Messages

This section contains a brief overview of the messages used by **icListen HF** stream channels.

2.1 Basic Message Structure

All messages transmitted over **icListen**'s streaming channels share a common structure:

Byte Offset:	Field:	# Bytes:
0	Msg Char	1
1	Sync (0x2A)	1
2	Payload Length	2
4	Payload	n

Figure 2-1: Basic Message Structure

All messages consist of a message type byte, a sync byte, and a 16 bit payload length (in bytes), followed by the payload. All message payloads must be 32bit aligned, and are in network order (big endian) unless otherwise noted.

2.2 Streaming Channel Messages

Message	***Code***	***Description***
Notify	'0' (0x30)	Sends important notifications
Data	'1' (0x31)	Data transmitted from icListen
Event Header	'2' (0x32)	Contains header information related to the following data messages
Start Stream	'3' (0x33)	Requests a data stream be opened
Stop Stream	'4' (0x34)	Requests a data stream be closed
Other Sensor Data	'5' (0x35)	Data from other sensors on the hydrophones (Temp, Humidity, etc)

2.3 Streamed Data Behavior

Streamed message types include an *Event Header* message and a *Data* message. The event header message will be sent once every second, on the second, and will contain data handling parameters. Multiple data messages will be sent between each event header message. This reduces the overhead of the header information, as it is only sent once a second. The message ordering is depicted below.

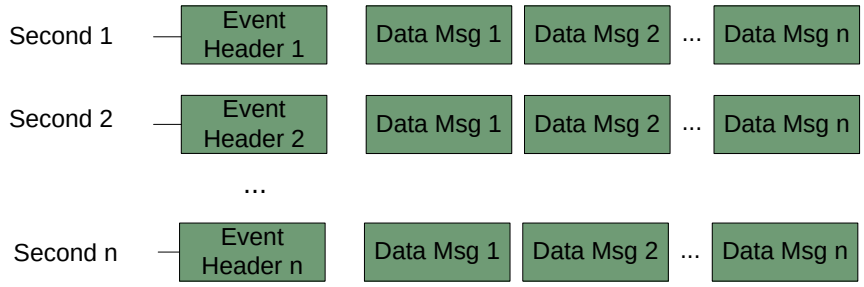

Figure 2-2: Message Ordering

The payloads of a both the event and data messages are broken into chunks, with formatting loosely based off of the RIFF standard. Each chunk starts with a type descriptor, a version number, a 16 bit payload length, followed by the payload. For more information on the chunks used to make up these messages, please refer to the *Message Chunk Details* section.

After a stream has been activated, it will remain active until either: the time the stream was requested for expires, a *Stop Stream* message is received, a *Stop All Streams* message is received over the command and control channel, or a connection loss has been detected.

2.4 Available Streaming Channels

icListen makes use of a different channels for each type of data to be streamed, and for each protocol which is used to stream data. Currently **icListen** supports streaming channels for spectrum data, and time series data, over TCP and UDP. Epoch and Other Sensor streams are available over TCP. The following table shows what streaming channels are available:

Streaming Channels:

Data Type	Protocol	Port
Time Series (Waveform)	UDP	51676
	TCP	51678
Spectrum (FFT)	UDP	51677
	TCP	51679
Epoch Messages	TCP	51680
Other Sensors	TCP	51681

2.5 Epoch Message Stream

As of Release 15, **icListen AF/HF** can stream epoch messages whenever an epoch event is triggered. The epoch stream is only available over TCP. For the epoch stream, connect to port 51680, and the unit will send epoch messages whenever an epoch trigger which is configured to send the epoch messages occurs and (as of Release 19) when a trigger ends.

The epoch messages are ASCII strings. An example string is as follows:

20120920T10-39-55 U:1219 Epoch:1 Seq:36430802944 Hz:10000-25000 > 100uPa 3sec

The format is:

[Time] U:[Serial #] Epoch:[Epoch #] Seq:[Sequence #] Hz:[Freq. Range] [> or <] [Amplitude] [Duration]

2.6 Other Sensor Data Stream

As of release 28, **icListen AF/HF** can stream data from other sensors on the hydrophone. Examples of sensors which can send data through this stream are temperature, humidity, battery details, time sync/PPS, accelerometer and magnetometer, and trigger status. The other sensor stream is only available over TCP. For the other sensor stream, connect to port 51681, and the unit will send a message whenever sensor data is updated.

3 Detailed Message Descriptions

This section gives detailed descriptions of all messages transmitted over the streaming data channel.

The only messages accepted into the **icListen** streaming port are those required to start and stop the streaming process:
Start Stream
Stop Stream

icListen will transmit the following messages over the streaming channels:
Notify
Event Header
Data
Other Sensor Data

3.1 Start Stream

Byte Offset:	Field:	# Bytes:
0	Type ‘3’ (0x33)	1
1	Sync (0x2A)	1
2	Payload Length	2
4	Duration	2
6	Requested Data	2

Figure 3-1: Start Message

Purpose: To start streaming data.

Description: This message requests that data be streamed from the port that receives this message. Each stream can be started by sending a Start message to the desired stream’s listening port. The data stream will be transmitted to the port which sent the start message. The “Duration” field sets the length of time (in minutes) that the stream will remain active. Setting a time of zero will cause the stream to remain active indefinitely, or until a *Stop Stream* message is received. For the Other Sensor data stream, the field of requested data is a bitmask of the types of data which is requested from the stream. A value of 0xFFFF for the requested data starts streams for all Other Sensor data. For waveform and spectrum data streams this field is ignored and should be set to 0.

Possible values for the requested data bitmask:

Code	***Data Type***
0x0001	Temperature/Humidity
0x0002	Heading Data
0x0004	Time Sync
0x0008	Battery
0x0010	Trigger Status

3.2 Stop Stream

Byte Offset:	Field:	# Bytes:
0	Type ‘4’ (0x34)	1
1	Sync (0x2A)	1
2	Payload Length	2

Figure 3-2: Stop Message

Purpose: To stop an active data stream.

Description: This message requests that an active data stream be terminated. When this message is received, **icListen** will stop streaming data from the channel that received the message, to the address that issued the message. If no data is currently being streamed to the address that sent this message, this command will have no effect. There is no payload for this message type.

3.3 Notify

Byte Offset:	Field:	# Bytes:
0	Type ‘0’ (0x30)	1
1	Sync (0x2A)	1
2	Payload Length	2
4	Notification Code	4

Figure 3-3: Notify Message

Purpose: To notify stream connection subscribers of important events.

Description: This message is transmitted from the **icListen** when important notifications must be made to any stream subscribers, such as error conditions. The payload of this message consists of a 32bit notification code.

Possible Notification Codes:

Code	Message
1	No Free Connection
2	Timeout
3	Stream Stopped
4	Invalid Start Message

No free connection: This is sent if a stream is connected to which has no remaining stream connections. The connection is terminated after this message is sent.

Timeout: If a stream connection was started with a set duration, this notification message is sent when that duration is over. The data which was being streamed will stop after this message, but the connection will remain opened.

Stream Stopped: This message is sent when data streams are stopped by a “Stop All Streams” command from the **icListen** Command and Control channel. As of **icListen AF/HF** release 28, the connection is terminated after this message is sent.

Invalid Start Message: This message is sent if a “Start Stream” message is sent with an invalid length. Data will not be streamed, but the connection will remain opened.

3.4 Event Header

Byte Offset:	Field:	# Bytes:
0	Type ‘2’ (0x32)	1
1	Sync (0x2A)	1
2	Payload Length	2
4	Event Key	m
4+m	Device Info	k
4+m+k	Status	p
4+m+k+p	Setup	i
4+m+k+p+i	Amplitude Scaling	j

Figure 3-4: Event Header Message

Purpose: To provide the necessary information for interpreting and synchronizing the following data messages.

Description: This message is transmitted from the **icListen** once per second, on the second boundary, from all active streaming channels. It provides the information required for processing the data messages that will follow.

The payload of this message contains an *Event Key Chunk*, *Device Info Chunk*, *Status Chunk*, and other chunks which vary based on what data is streamed from the channel. For more detail on the individual chunks, please refer to the *Message Chunk Details* section.

3.5 Data

Byte Offset:	Field:	# Bytes:
0	Type ‘1’ (0x31)	1
1	Sync (0x2A)	1
2	Payload Length	2
4	Event Key	m
4+m	Data	p

Figure 3-5: Data Message

Purpose: These messages contain the acoustic data read by **icListen**.

Description: Data messages consist of an *Event Key Chunk* (used to synchronize to an *Event Header* message), and a *Data Chunk* (which contains the actual data, and some information required for reading it). For more detail on these chunks, please refer to the *Message Chunk Details* section.

Multiple Data messages may be sent for each *Event Header* message.

3.6 Other Sensor Data

Byte Offset:	Field:	# Bytes:
0	Type ‘5’ (0x35)	1
1	Sync (0x2A)	1
2	Payload Length	2
4	Data Chunk 1	m
4+m	Data …	p
4+m+p…	Data Chunk n	i

Figure 3-6: Data Message

Purpose: These messages contain the data from other sensors in an **icListen**.

Description: Other Sensor Data messages consist of multiple data chunks from the sensors in an **icListen**. These sensors can include: Temperature, humidity, Battery details, time sync/PPS, Accelerometer and Magnetometer, and Trigger Status. This message is sent when new data is available for one or more of the sensors. The message will contain any updated sensor data, and does not always include all chunks.

4 Message Chunk Details

This section describes the format and use of each chunk type used to build *Event Header* and *Data* messages. The message chunks contained within *Event Header* messages will vary based on the form of data being transmitted. All message chunks use the same basic format:

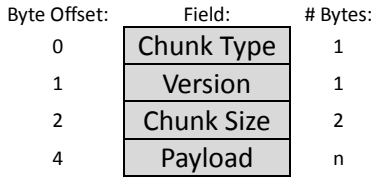

Figure 4-1: Chunk Structure

The "Chunk Type" and "Version" fields are used to determine the contents of a message chunk, and the "Chunk Size" field is used to indicate the length of the chunk payload in bytes.

Available chunk types include:

Message	***Code***	***Description***
Event Key	'A' (0x41)	Contains Time/Data Sequence Number, used to pair data with headers
Data Chunk	'B' (0x42)	Contains Waveform or Spectrum Data
Status	'C' (0x43)	Contains various sensor data, included with data header
Device Info	'D' (0x44)	Contains Device/firmware details
Wave Setup	'E' (0x45)	Contains the waveform data setup
FFT Setup	'F' (0x46)	Contains the spectrum data setup
Scaling	'G' (0x47)	Contains the details for scaling data counts to real world units
Temp/Humidity	'H' (0x48)	Contains Temperature and Humidity data
Heading	'I' (0x49)	Contains Accelerometer and Magnetometer data
Time Sync	'J' (0x50)	Contains time sync and PPS details
Battery	'K' (0x51)	Contains battery details and statuses
Trigger Status	'L' (0x52)	Contains the status of all epoch triggers

4.1 Event Key Chunk

Byte Offset:	Field:	# Bytes:
0	Type ‘A’ (0x41)	1
1	Version (0)	1
2	Chunk Size	2
4	UNIX Time	4
8	Seq Num High	4
12	Seq Num Low	4

Figure 4-2: Event Key Chunk

The event key chunk is chunk type ‘A’ (0x41). Currently only version 0 of this chunk exists. This chunk has 2 purposes. The first purpose is to detail the ordering of the data. This is done using the UNIX time value and a sample sequence number. The second purpose is to match each *Data* message with an *Event Header* message. The event key from an event header message will match exactly the event key in all data messages before the next header. This is used to provide data processing details to the data returned in the data messages. The sample sequence number is a 64-bit value which is incremented for every sample in the stream. This value is updated in the event key once a second, and will increase by the sample frequency in value.

4.2 Data Chunk

Byte Offset:	Field:	# Bytes:
0	Type ‘B’ (0x42)	1
1	Version (0)	1
2	Chunk Size	2
4	Sample #	4
8	# of channels	1
9	Data Format	1
10	# of Samples	2
12	Data	n
12+n	Padding	k

Figure 4-3: Data Chunk

The data chunk is chunk type ‘B’ (0x42). Currently only version 0 of this chunk exists. This chunk contains the position of the data, in samples, relative to the corresponding header message. This chunk also contains the details for parsing the data. These details include the number of interlaced data channels, data format, and the number of samples. These details are followed by the data samples.

Padding bytes are placed at the end of the message to maintain alignment to the nearest 32bit boundary.

Bit 7(highest bit) of the Data Format is set to 1 for little endian data, and 0 for big endian. The remaining bits represent the number of bytes per sample.

4.3 Status Chunk

Byte Offset:	Field:	# Bytes:
0	Type ‘C’ (0x43)	1
1	Version (0)	1
2	Chunk Size	2
4	Temperature	2
6	Humidity	2
8	Battery Charge	1
9	Battery Status	1
10	Data Epoch	1
12	Reserved (0)	1
14	Reserved (0)	12

Figure 4-4: Status Chunk

The status chunk is chunk type ‘C’ (0x43). Currently only version 0 of the chunk exists. This chunk is used to send the values of other sensors from the **icListen**. In version 0, the returned data contains: temperature (in tenths of degrees Celsius), humidity (in tenths of percent), charge percent, battery status, and the data epoch.

The data epoch is a monotonic counter which is incremented each time the ADC must be restarted. Any time the ADC is restarted the sequence number will be reset to 0. This may happen when data is sync’d to a new pulse per second (PPS) signal (if the device is configured to do so). The data epoch field is available on firmware release 28 and newer.

The available battery status codes are shown in the table below:

Status Code	Description
0	No battery status available (Charge % should be ignored when this status is present)
1	Battery is charging
2	Battery is discharging (no external power applied)
3	Battery is externally powered, but not charging (this will occur when the battery is fully charged)

*Note that the battery status and charge percentage are for the internal battery only.

4.4 Device Info Chunk

Byte Offset:	Field:	# Bytes:
0	Type ‘D’ (0x44)	1
1	Version (0)	1
2	Chunk Size	2
4	Model	1
5	Firmware Version	3
8	Serial Number	2
10	Phone Sensitivity	2

Figure 4-5: Device Chunk

The device chunk is chunk type ‘D’ (0x44). Currently only version 0 of the chunk exists. This chunk provides details about the connected **icListen** unit. This includes the model (5 = **icListen HF**, 7 = **icListen AF**), firmware version, serial number, and phone sensitivity (in tenths of dB re 1 µPa). This chunk is passed as part of the *Event Header* message.

4.5 Wave Setup Chunk

Byte Offset:	Field:	# Bytes:
0	Type ‘E’ (0x45)	1
1	Version (0)	1
2	Chunk Size	2
4	Sample Rate	4
8	Gain	2
10	Data Format	1
11	Reserved (0)	1

Figure 4-6: Wave Setup Chunk

The wave setup chunk is chunk type ‘E’ (0x45). Currently, only version 0 exists for this chunk. This chunk contains the setup parameters for a stream of wave data. This chunk includes the sample rate (in samples per second), gain (in dB), the data format, and followed one reserved byte.

Bit 7(highest bit) of the Data Format is set to 1 for little endian data, and 0 for big endian. The remaining bits represent the number of bytes per sample.

4.6 Spectrum Setup Chunk

Byte Offset:	Field:	# Bytes:
0	Type ‘F’ (0x46)	1
1	Version (0)	1
2	Chunk Size	2
4	Sample Rate	4
8	Num of Points	2
10	Reference Level	2
12	Processing Type	2
14	Samples Between FFTs	2
16	Accumulation Count	2
18	Weighting Factor	2

Figure 4-7: Spectrum Setup Chunk

The spectrum setup chunk is chunk type ‘F’ (0x46). Currently, only version 0 exists for this chunk. This chunk contains the setup parameters for spectrum (FFT) data. These parameters include the sample rate, reference level, number of points per FFT calculation, FFT processing type, Samples between FFTs, FFT Accumulation Count, and Weighting Factor. The reference level refers to the value (in dBV) of the minimum FFT count. (i.e. A count of 0 is equal to the reference level). The samples between FFTs field is the number of samples between the start of each FFT calculation. The accumulation count is the number of averaged FFT results. The Weighting factor is a setting for the exponential average “filtered” mode.

4.7 Amplitude Scaling Chunk

Byte Offset:	Field:	# Bytes:
0	Type ‘G’ (0x47)	1
1	Version (0)	1
2	Chunk Size	2
4	Channel #	1
5	Unit of Measure	1
6	Reserved (0)	1
7	Bytes Per Sample	1
8	Max Count	4
12	Min Count	4
16	Max Amplitude	4
20	Min Amplitude	4

Figure 4-8: Amplitude Scaling Chunk

The amplitude scaling chunk is chunk type ‘G’ (0x47). Currently, only version 0 exists for this command. This chunk contains the details for converting the returned values to real world units. There will be one of these chunks for each channel of returned data. This chunk includes the data channel number, the unit of measure of the data, and the bytes per sample. Also included are the max/min count values, which are the max/min values returned by the **icListen**. Finally there are the max/min amplitude values, which are the max/min value in the returned units of measure, which the max/min counts correspond. For example, for Spectrum (FFT) data, the min and max count values will be 0 and 255 respectively, while the min and max amplitude values will be roughly (dependent on reference level settings) -4 and 124 respectively.

4.8 Temperature/Humidity Chunk

Byte Offset:	Field:	# Bytes:
0	Type ‘H’ (0x48)	1
1	Version (0)	1
2	Chunk Size	2
4	Temperature	2
6	Humidity	2

Figure 4-9: Temperature/Humidity Chunk

The Temperature/Humidity chunk is chunk type ‘H’ (0x48). Currently, only version 0 exists for this command. This chunk contains the temperature in tenths of °C (ex. A value of 124 is 12.4°C) and the humidity in tenths of % (ex. A value of 201 is 20.1%).

4.9 Heading Chunk

Byte Offset:	Field:	# Bytes:
0	Type ‘I’ (0x49)	1
1	Version (0)	1
2	Chunk Size	2
4	Accel X	2
6	Accel Y	2
8	Accel Z	2
10	Mag X	2
12	Mag Y	2
14	Mag Z	2

Figure 4-10: Heading Chunk

The Heading chunk is chunk type ‘I’ (0x49). Currently, only version 0 exists for this command. This chunk contains accelerometer and magnetometer data from **icListen**.

4.10 Time Sync Chunk

Byte Offset:	Field:	# Bytes:
0	Type ‘J’ (0x50)	1
1	Version (0)	1
2	Chunk Size	2
4	Sync Status	1
5	System Time Status	1
6	Reserved (0)	2
8	Sync Offset	4
12	PPS Count	4
16	System Time	4

Figure 4-11: Time Sync Chunk

The Time Sync chunk is chunk type ‘J’ (0x50). Currently, only version 0 exists for this command. This chunk contains details for time synchronization and PPS handling.

Sync Status: This code indicates is the status of synchronization. Valid codes are:

0 = Sync input/output disabled
1 = Sync output enabled
2 = Sync output enabled (with PPS encoded time)
3 = Sync input enabled (falling edge), no PPS detected
4 = Sync input enabled (falling edge), syncing to detected PPS
5 = Sync input enabled (falling edge), synced to PPS
6 = Sync input enabled (rising edge), no PPS detected
7 = Sync input enabled (rising edge), syncing to detected PPS
8 = Sync input enabled (rising edge), synced to PPS

System Time Status: This code indicates how the system time was set. Valid codes are:

0 = Time value has not been set
1 = Time was manually set
2 = Time set from RTC on reboot
3 = Time was set from file system after reboot
128 = Time set from PPS encoded time (from unset source)
129 = Time set from PPS encoded time (from manually set source)
130 = Time set from PPS encoded time (from source set by RTC on reboot)
131 = Time set from PPS encoded time (from source set by file system on reboot)

Sync Offset: This is the measured time offset in nanoseconds, between the detected PPS edge, and this system’s second edge.

PPS Count: This is a monotonic counter of PPS inputs detected when configured for sync input. This value will be set to 0 if sync input is not enabled.

System Time: This is the UNIX time value of **icListen**’s system clock. UNIX time is defined as the number of seconds since 00:00:00 January 1, 1970 UTC.

4.11 Battery Chunk

Byte Offset:	Field:	# Bytes:
0	Type ‘K’ (0x51)	1
1	Version (0)	1
2	Chunk Size	2
4	Charging Status	1
5	Charge Percent	1
6	Charge (mAh)	2
8	Capacity	2
10	Battery Current	2
12	Battery Voltage	2
14	Battery Temperature	2
16	Board Voltage	2
18	Microprocessor Voltage	2

Figure 4-12: Battery Chunk

The Battery chunk is chunk type ‘K’ (0x51). Currently, only version 0 exists for this command. This chunk contains the details of the status and charge of the battery.

Charging Status: This is a code which indicates the charging state of the battery. Valid codes are:
- 0 = No Data Available
- 1 = Battery is Charging
- 2 = Battery is Discharging
- 3 = Battery is not Charging or discharging

Charge Percent: This is the estimated percentage of full capacity that the battery is charged to.
Charge: This is the estimated charge currently stored in the battery, measured in mAh.
Capacity: This is the capacity of the battery when fully charged, measured in mAh.
Battery Current: This is current flowing into the battery, measured in mA. A negative current indicates that current is flowing out of the battery (discharging).
Battery Voltage: This is the voltage measured on the battery terminals, measured in mV.
Battery Temperature: This is temperature of the battery, measured in tenths of °C (303 = 30.3 °C).
Board Voltage: This is regulated supply voltage to the **icListen**, measured in mV.
Microprocessor Voltage: This is the supply voltage to the main microprocessor, measured in mV.

The battery state can report a status code of 3 (NOT charging or discharging), for one of 3 reasons: the battery may be fully charged, the status may have been updated during the switch between charging and discharging, or the charging may have been stopped due to high battery temperature.

4.12 Trigger Status Chunk

Byte Offset:	Field:	# Bytes:
0	Type ‘L’ (0x52)	1
1	Version (0)	1
2	Chunk Size	2
4	Epoch 1 Status	1
5	Epoch 2 Status	1
6	Epoch 3 Status	1
7	Epoch 4 Status	1
8	Epoch 5 Status	1
9	Reserved (0)	3

Figure 4-13: Trigger Status Chunk

The Trigger Status chunk is chunk type ‘L’ (0x52). Currently, only version 0 exists for this command. This chunk contains the status of all epoch triggers in the **icListen**.

Possible statuses include:

Code	Trigger Statuses
0	No Signal, Event Inactive
1	Signal Present, Event Inactive
2	Signal Present, Event Active
3	No Signal, Event Active
4	Disabled

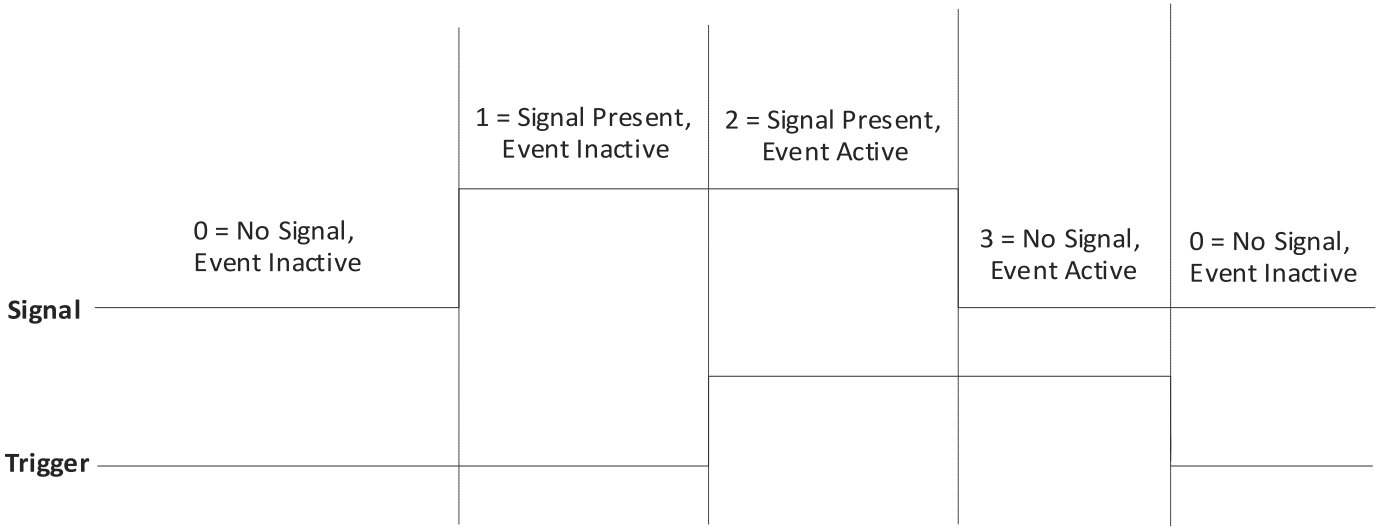

Figure 4-14: Trigger Status Example

The above figure shows a trigger going through all of the stages as a signal appears and disappears. It begins in state 0, where there is no signal and the event is inactive. When the desired signal appears, then it enters state 1, the signal is present but the event is still inactive as the signal has not been

present for long enough. After the signal has been present long enough to cause an event trigger the status changes to 2, which is signal present and event active. Once the signal disappears, the status changes to 3, this means that there is no signal but the event is still active. This status lasts for the duration of the configured event hold and then the status returns to 0. The status of 5 means that no event trigger is configured for that trigger number.