Sensor Hub Transport Protocol

Document Number: 1000-3535
Document Revision: 1.7
Date: 02/16/2017

Hillcrest Laboratories, Inc. 15245 Shady Grove Road, Suite 400 Rockville, MD 20850

Table of Contents

List of Figures

1.0 Introduction

The Sensor Hub Transport Protocol (SHTP) is a low overhead, simple mechanism for managing communications between the host and the hub over SPI, I2C or other physical layers.

1.1 Intended Audience

This document is intended for application developers implementing products that use the Sensor Hub or other systems using the SHTP.

1.2 Scope

This document describes the features and use of the SHTP.

1.3 Revision History

Revision	Date	Description
1.7	02/16/2017	Update Notices.
1.6	01/20/2017	Remove unused references.
1.5	12/15/2016	Revise Figure 11 to show optional later deassertion of HINT. Added clarification to section 2.6 allowing later deassertions of HINT.
1.4	11/17/2016	Revised deassertion of HINT for UART protocol. Added requirement that the host delay between bytes in UART mode.
1.3	06/23/2016	Defined format for SHTP version number tag. Added TLV transmission at startup. Added Error List command. Added SHTP over UART specification and clarified/expanded synchronous protocol specifications.
1.2	07/13/2015	Updates to clarify operation with duplex interfaces.
1.1	03/27/2015	Minor updates following initial implementation.
1.0	02/25/2015	Initial issue

Figure 1: Document History

2.0 Theory of Operation

A transport mechanism moves cargo from one place to another with no knowledge of the cargo. The cargo must be examined at the destination to determine how to handle the cargo. The SHTP is a simple transport mechanism that exchanges cargoes between the host and the Sensor Hub. Moving cargo from the host to the hub is a write operation. Moving cargo from the hub to the host is a read operation.

2.1 Channels

Channels are used to identify a specific path between the host and hub through the SHTP. The SHTP accepts cargoes for a specific channel on one side and delivers them to that channel on the other side. The users of SHTP can use a channel for transporting a specific type of cargo or they can use it for any type of cargo.

Channels are assigned to users for their exclusive use. Both directions of a given channel are assigned to the same user.

2.2 Headers

The SHTP uses headers to convey information about the transport of cargo between the host and hub. The SHTP header precedes the cargo in all write and read operations.

2.2.1 SHTP Header

The SHTP header is shown in Figure 2.

Byte	Field
0	Length LSB
1	Length MSB
2	Channel
3	SeqNum

Figure 2: SHTP Header

Length	The MSB of the length field is used to indicate if a transfer is a continuation of a previous transfer. Bits 14:0 are used to indicate the total number of bytes in the cargo plus header, which may be spread over multiple messages. The bytes in the header field are counted as part of the length. A length of 65535 (0xFFFF) is reserved because a failed peripheral can too easily produce 0xFFFF. Therefore, the largest cargo that can be transported is 32766 minus the header bytes.
Channel	The channel number of the cargo. Channel 0 is the command channel and is used by the SHTP.
SeqNum	The sequence number of the cargo. The sequence number is a monotonically incrementing number that increments once for each cargo sent or cargo continuation sent. Each channel has its own sequence number. The sequence number is used to detect duplicate or missing cargoes and to associate segmented cargoes with each other.

2.3 Rules

2.3.1 General

The SHTP operates according to the following rules:

- Reads that terminate after the length field or partway through the cargo indicate that the host is not sure of how many bytes to read. When this happens the host examines the length field to determine how many bytes to read. The host then performs one or more additional reads to retrieve the remaining cargo. The hub sends the unread portion of the cargo in subsequent reads until the entire cargo is read. For these subsequent transfers, the hub sets the MSB of the length field. The length field indicates the remaining number of bytes to transfer (cargo and header). The channel for subsequent reads is the same is the channel for the initial read. The sequence number is incremented for each read. If the hub changes the channel number or sends a header with the MSB cleared before the previous cargo was completely transferred then the previous cargo is lost and the host begins receiving a new cargo.
- Reads that extend beyond the cargo length are padded with zeroes. This rule is needed for SPI (duplex) physical layers. A read cargo must be padded if the host is simultaneously writing a cargo that is longer than the read cargo. This behavior is safe for I2C (simplex) physical layers.
- A length of 0 indicates that there is no cargo. The remaining bytes in the header may also be set to 0. This situation represents a null header. Null headers are used for SPI physical layers or if the host performs a read when no cargo is available.
- Writes may also be segmented into several transfers. The host does this by sending the first part of the cargo with the MSB of the length field cleared and then terminating the transfer early. The host sends the remaining portion of the cargo in one or more writes with the MSB of the length field set. The length field indicates the remaining number of bytes to transfer (cargo and header). The sequence number for each write is incremented. The hub assembles the cargo from the multiple transfers. If the host changes the channel number or sends a header with the MSB cleared before the previous cargo was completely transferred then the previous cargo is lost and the hub begins receiving a new cargo.
- Writes that extend beyond the cargo length are padded with zeroes. Bytes beyond the cargo length are ignored. This rule is needed for SPI physical layers. A write cargo must be padded if the host is simultaneously reading a cargo that is longer than the write cargo. This behavior is safe for I2C layers.
- A length of 65535 is an error. The remaining header and cargo bytes are ignored. This type of error may occur if there is a failure in the SPI or I2C peripheral.
- At system startup, the hub must send its full advertisement message (see 5.2 and 5.3) to the host. It must not send any other data until this step is complete.

2.3.2 Hubs with Capability Limitations

Hardware or software capability restrictions or resource limitations may prevent a hub from either transferring a maximum length cargo or may limit the maximum number of bytes that can be exchanged in a single operation or transfer. Therefore a hub may impose implementation specific maximum cargo-plus-header sizes and maximum transfer sizes. These sizes may be different for the write and read directions. The maximum transfer size includes the SHTP header bytes. The following additional rules apply when these limitations are imposed:

- The maximum transfer length must always be less than or equal to the maximum cargo-plus-header length.
- The host may never attempt a transfer larger than the maximum transfer size. Doing so is a protocol violation. Since the conditions under which a maximum transfer size is imposed are implementation specific, recovery from this violation is implementation specific.
- A transfer of the maximum transfer length that does not transfer the entire cargo or complete the transfer of the cargo is treated as an early termination. The remaining cargo is transferred in subsequent reads or writes.

2.4 Writing Cargoes

When the host writes a cargo, it must transfer as much of the cargo as is permitted at a time: for example, if the maximum transfer length is 128 bytes and the host has 64 bytes to write, it should perform a single transfer of 64 bytes (not, for instance, 2 transfers of 32 bytes).

Note that this allows hubs that cannot support fragmented writes (due to resource limitations) to advertise identical maximum transfer length and maximum cargo length: in this case, all legal writes must be performed in a single non-fragmented transfer.

2.5 Reading Cargoes

Reading cargoes may only begin after the Sensor Hub asserts the host interrupt signal.

2.6 Host Interrupt and Timestamps

The host interrupt signal (HINT) is an edge triggered interrupt. When the Sensor Hub has a cargo to be read, it asserts HINT. After the read begins, it deasserts HINT. One transfer should be read for each assertion of HINT. If the host fails to respond to the assertion of HINT after a timeout, the hub may deassert and reassert HINT. If a cargo is broken up into multiple reads then the hub deasserts and reasserts HINT for each transfer. HINT may be deasserted at any time after the read begins, including after the transaction is complete.

Some applications may need to convey timestamp information from the hub to the host. The host should timestamp when HINT is asserted. Applications on the hub may then send time information relative to when HINT is asserted as part of their cargoes. The host may then reconstruct the time of events based on the timestamp of HINT and the relative time information provided in the cargoes. When a cargo is segmented into multiple transfers, only the first assertion of HINT should be used for timestamping.

3.0 Synchronous Protocols (SPI/I2C)

This section describes the high level behaviors for synchronous protocols (SPI and I2C) which implement SHTP.

3.1 Host-Hub handshaking

Some platforms may require uninterrupted execution of the host interface code to prevent race conditions or missed deadlines. The preferred mechanism for such a situation is:

1. Host asserts WAKE signal, requesting hub to become ready for a transfer.
2. Hub asserts HINT signal, indicating that it is ready for a transfer.
3. Hub deasserts HINT when transfer begins.

Figure 3 shows an example of this for SPI. The same flow should be observed for I2C implementations that need an explicit wakeup signal (e.g. that cannot be woken up from low power mode with an address match).

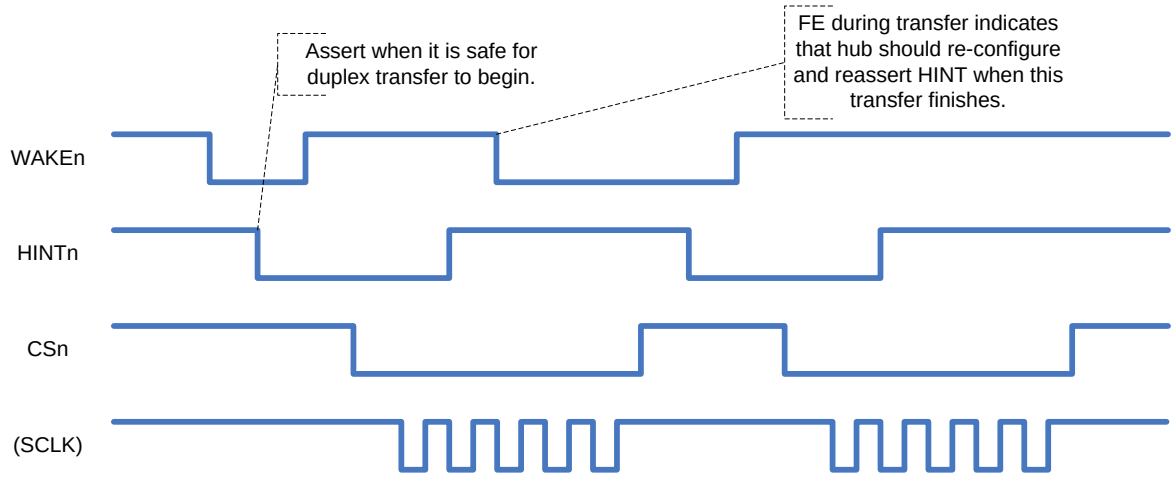


Figure 3: SPI Handshake sequence

3.2 SHTP over I2C

Repeated starts to the I2C address of an SHTP-enabled hub are not supported. Each SHTP transfer must end with a STOP condition.

3.3 SHTP over SPI

SHTP will support simultaneous read and write operations over SPI. If the platform limitations on transfer lengths differ for read and write operations, transfers which exceed the length of the smaller will be truncated. For example, if there is a 64-byte limit on write cargoes and a 128-byte limit on read cargoes, bytes 65-128 of the write cargo will be ignored in a 128-byte duplex transfer.

Note that the hub should never assert HINT if it is unable to perform a **duplex** transfer: if it has data for a host read, but does not have a buffer available for a host write, it may not assert HINT per the description in Section 3.1. If the host has asserted WAKE but it has no data to send, it may not assert HINT until it can safely provide a null header (all 0's).

3.4 Host Reads for Synchronous Protocols

The following diagrams explain the high-level operation of host reads when running SHTP over a synchronous protocol. The same flow should apply to both SPI and I2C implementations.

3.4.1 Host Read: Host-Side State Machine

A diagram of the state machine used by the host to manage reads is shown in Figure 4.

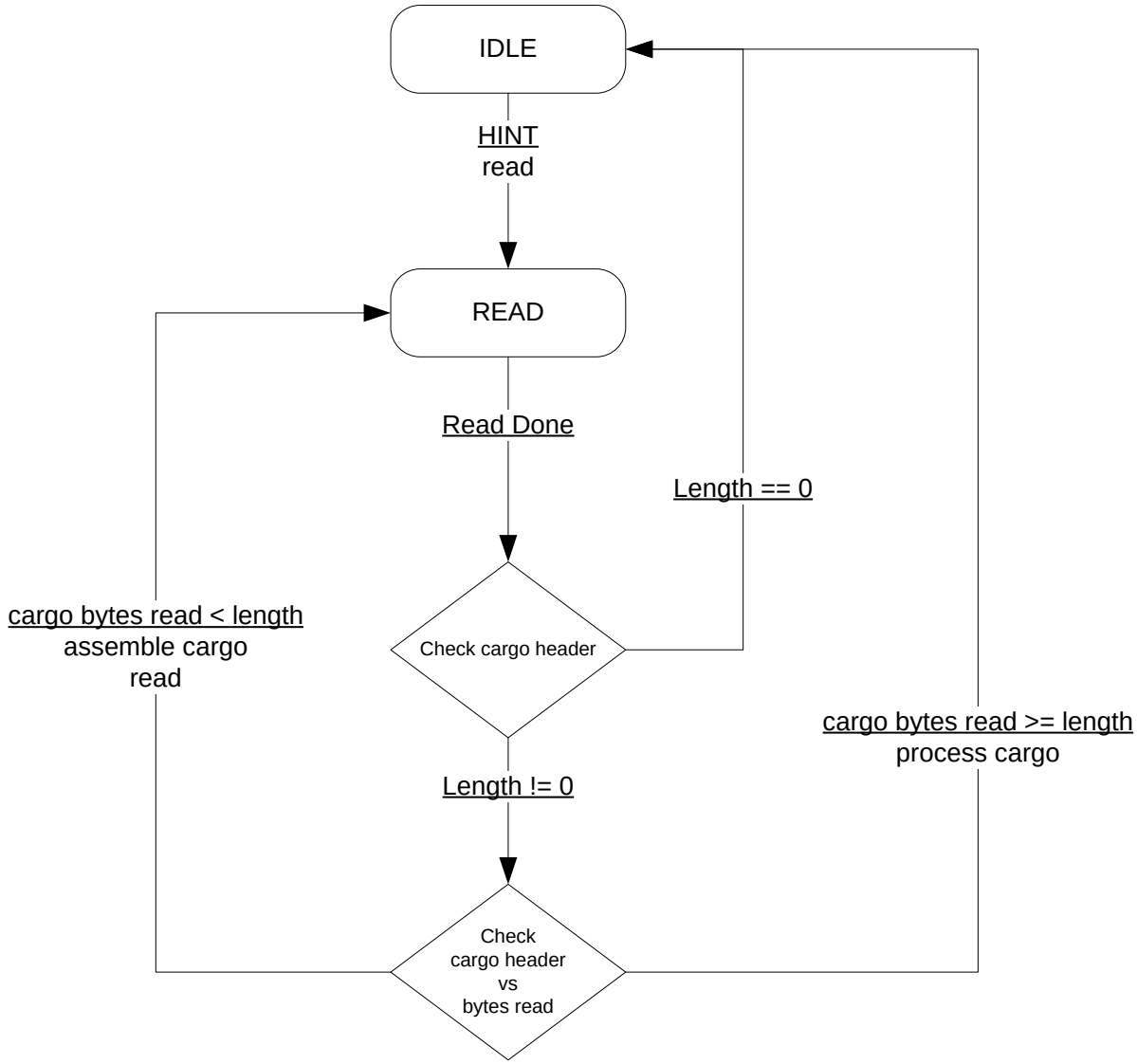


Figure 4: Host Read State Machine

The basic operation of the state machine is to read one cargo message each time the host interrupt signal is asserted. If the host knows what length to read then it does so. If it does not, it reads just the read cargo header to determine the length and then performs a full read. The host may also attempt to predict the length to read and read the predicted amount. If the read terminates early then the host performs additional reads as necessary.

3.4.2 Host Read: Hub-Side State Machine

A diagram of the hub read state machine is shown in Figure 5.

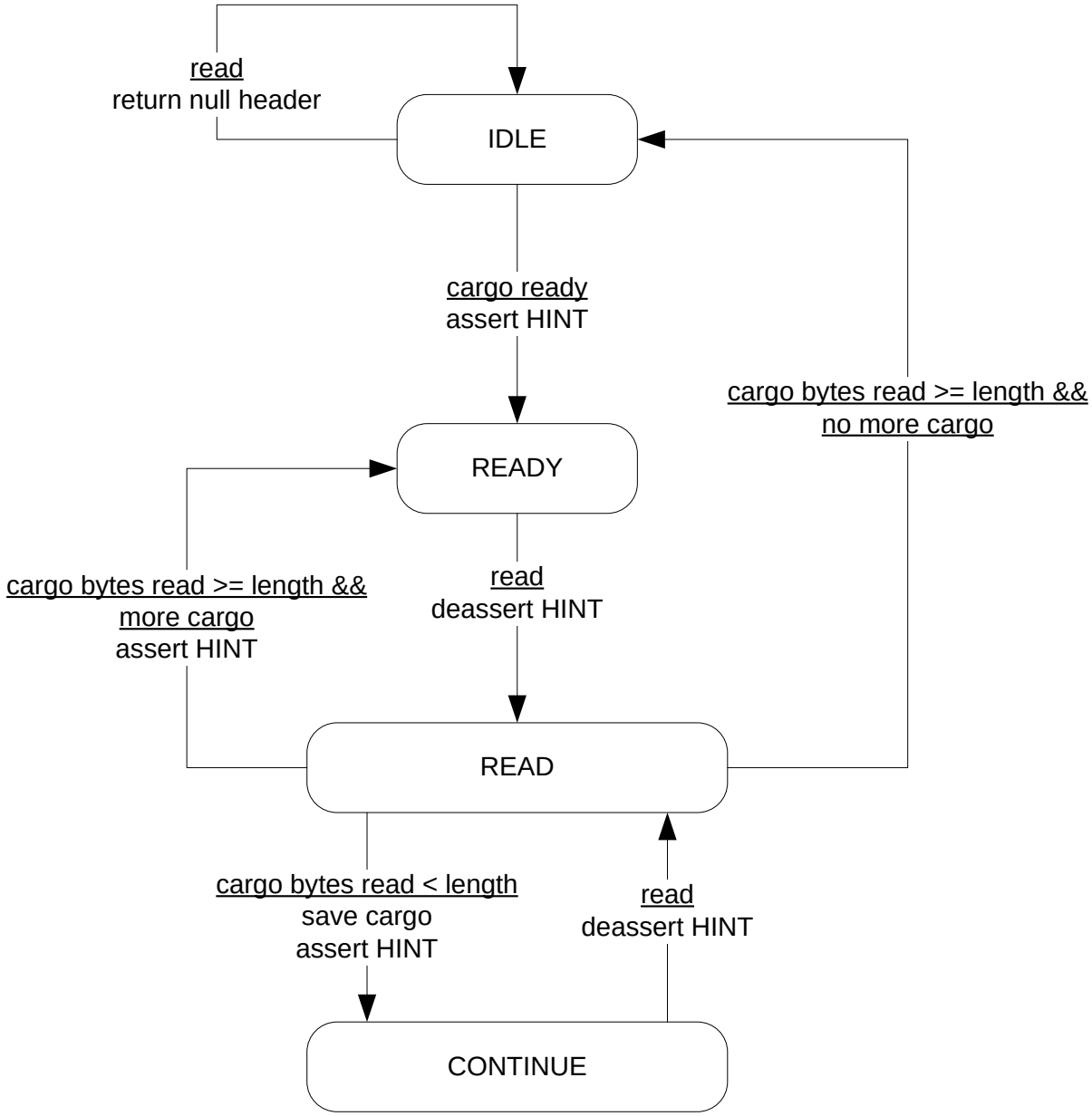


Figure 5: Hub Read State Machine

The basic operation of the state machine is to supply a read cargo to the host whenever it has one. If the host reads too few bytes then the hub supplies the remaining portion of the cargo during subsequent reads.

3.5 Host Writes for Synchronous Protocols

Host writes are straightforward under synchronous protocols. If handshaking is required, then it should be performed as described above. If handshaking is not required (e.g. because an I2C hub can use clock-stretching to signal the host when it is active and ready to receive a host write), then the host can simply write as much of its cargo as is dictated by the hub's capabilities and the description in section 2.3.2.

4.0 SHTP over UART

UART is a two wire (RX / TX) asynchronous protocol. In order to take advantage of message timestamping, a separate HINT signal is required. In order to get the lowest power consumption available from hubs that cannot receive UART data in low power modes, WAKE is required. If neither of these features is required, a host may communicate with a hub using the basic RX/TX signals.

4.1 Control Characters

In order to support the basic (two-wire) communication mode, we must reserve certain characters to delineate the start/end of each transfer. Per the RFC-1662 standard [1], a Flag Byte (start/end of frame) and control escape character are allocated:

Value	Meaning
0x7E	Flag Byte
0x7D	Control Escape

Figure 6: Control Characters

In the event that a data byte having one of these reserved control characters occurs, a control escape (0x7D) byte will be sent followed by the data byte XOR'ed with 0x20. e.g. the data byte 0x7E is sent as 0x7D, 0x5E.

4.2 Message Structure

Each SHTP over UART message begins with a flag byte, a "protocol ID" byte and any number of payload bytes. The message ends with another flag byte. Note that when two flag bytes are seen in a row, the recipient should treat this just as a "start of message" flag (e.g. disregard the 0-length message that would be indicated were the first byte a "start" and the second byte an "end").

Byte	Meaning
0	Flag (0x7E)
1	Protocol ID
…	Data
N	Flag (0x7E)

Figure 7: SHTP over UART message

The protocol ID byte indicates whether the (optional) payload to follow should be interpreted as an SHTP message (to be handled like any other SHTP message) or whether it is a control message for SHTP over UART.

Value	Meaning
0x00	SHTP over UART Control
0x01	SHTP

Figure 8: Protocol ID field

4.3 SHTP over UART Control Messages (Protocol ID = 0)

There are two SHTP over UART control messages which are used to establish when it is safe for the host to perform a host write.

1. Buffer Status Query (BSQ): Sent from host to hub to inquire as to its available buffer space for receiving host writes.

Byte	Value
0	Flag (0x7E)
1	Protocol ID = 0
2	Flag (0x7E)

Figure 9: Buffer Status Query (BSQ)

2. Buffer Status Notification (BSN): Sent from hub to host to announce how much space is available for it to receive host writes. Note that the space reported includes only the length of the un-escaped payload (i.e. it corresponds to the maximum value that can be put into an acceptable SHTP cargo header).

Byte	Meaning
0	Flag (0x7E)
1	Protocol ID = 0
2	Bytes available LSB
3	Bytes available MSB
4	Flag (0x7E)

Figure 10: Buffer Status Notification (BSN)

The hub must always be able to receive a BSQ when it is in an active mode (i.e. host writes should not interfere with the hub's ability to store/parse a BSQ).

4.4 Host Reads

A Host Read (Hub TX) can occur at any time: it's assumed that the host device has the capability to handle this.

The hub asserts HINT prior to the beginning of its initial flag byte transmission. HINT may be deasserted at any point after the initial flag byte character begins and must be deasserted before the next read begins.

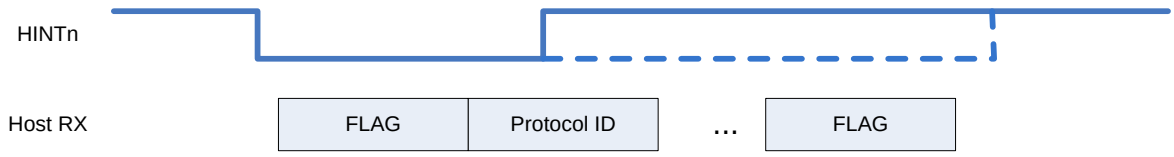


Figure 11: Host Read Sequence

4.5 Host Writes

In order to ensure correct operation with interrupt-driven hubs, the host must delay a minimum of 100 microseconds between bytes written to the hub.

If a host write operation begins while the hub is in a low-power mode or while the hub does not have local storage available for the data being written, the hub may ignore the entire write operation. In order to ensure that a host write will be properly received, the host should take care to follow the guidance provided by the Buffer Status Notifications (BSNs) it receives from the hub.

A BSN may be thought of as a token that grants the host permission to write up to some number of bytes until some expiration condition has occurred. A BSN is valid from the time that it is read by the host **until the time that the next host write completes**, as long as it is within a short timeout[1] of the **latest** of the following events:

1. WAKE deasserted
2. BSN received

Note that this implies that if WAKE is continuously asserted, BSNs have no expiration.

There are three ways that a host can obtain a BSN:

1. Assert WAKE: the hub MUST respond with a BSN as soon as it is able to do so.
2. Write a BSQ: the hub MUST respond with a BSN as soon as it is able to do so.
3. At the conclusion of a host write, if WAKE is still asserted the hub MUST send a BSN when it is next able to receive more than 0 bytes.

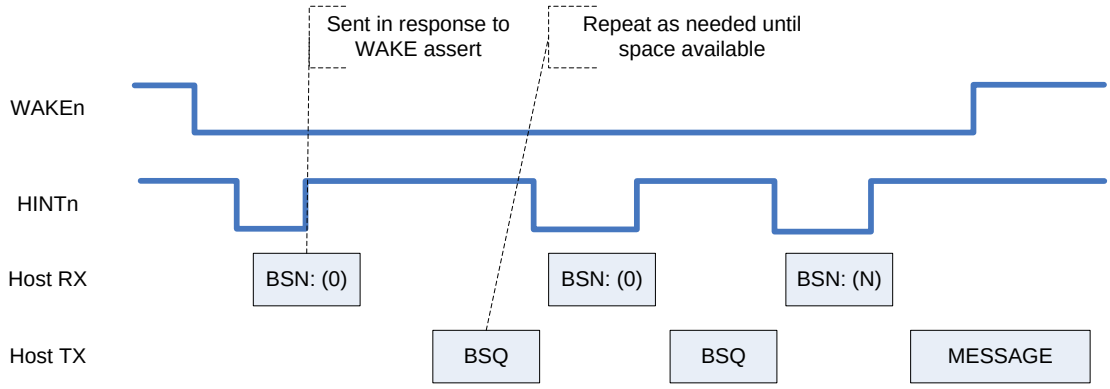
The following sections explain the flow control process under a variety of settings.

4.5.1 WAKE controlled by Host

This mechanism provides the ability for the lowest-power operation of the hub.

When the host wishes to begin a write, it asserts WAKE. It waits until it receives a BSN. If the BSN indicates space is available, it proceeds with the write. Otherwise, it leaves WAKE asserted and repeatedly sends BSQ's (optionally delaying between each one) until a BSN is received that indicates space is available.

Figure 12 demonstrates such a sequence where the host must send two BSQ's before it gets a BSN which indicates enough space is available to proceed.



[1] The timeout is implementation-dependent and specified by the "SHTP over UART Timeout" entry in the hub's SHTP advertisement.

Figure 12: Host Write: WAKE controlled by host

Note the host deasserts WAKE before its write concludes: had it been left asserted, the hub would have sent another BSN when it had finished handling the host.

Had the initial BSN reported sufficient space, the host write could have commenced immediately.

4.5.2 WAKE is always asserted

This mechanism can be used on systems where no WAKE signal is available to the host, but the hub requires a GPIO signal to remain in a mode where it can receive UART data. It compromises low-power operation, but provides the most responsive interface with reduced GPIO requirements.

When the host wishes to begin its first write, the hub's ability to accept it is unknown. It sends a BSQ and waits until a BSN is received. It continues sending BSQ's until it receives a BSN that indicates that sufficient space is available.

After each host write, the host must wait until it sees another BSN before it begins its next write (item 3 above: WAKE is always asserted). Since WAKE is never deasserted, these BSNs are valid until the next host write begins (no additional BSQs are required).

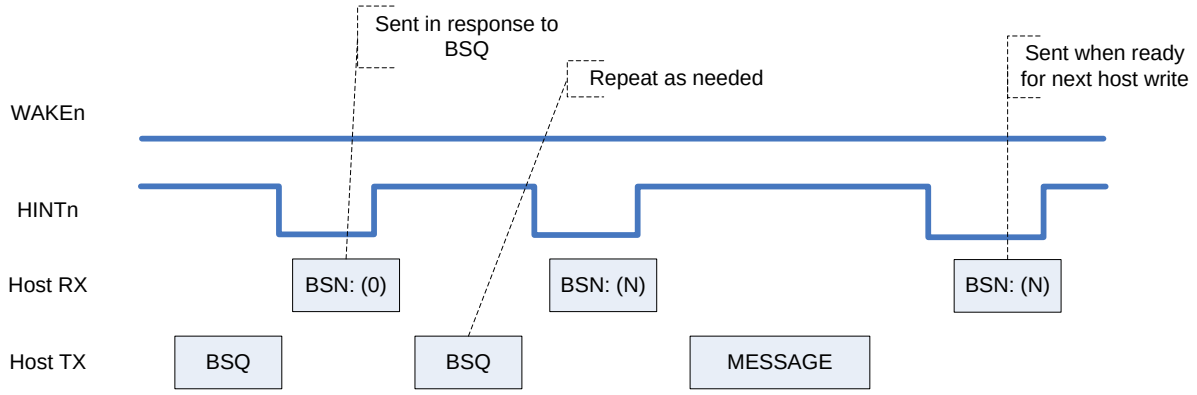


Figure 13: Host Write: WAKE always asserted

4.5.3 WAKE is always deasserted

This mechanism is only available to hubs that can be woken up from low-power mode by UART activity. It compromises responsiveness for low power with reduced GPIO requirements.

It is accepted that the first BSQ written to the hub while it is in low power mode may be lost (e.g. if it can only buffer a single byte at a time and the baud rate is fast enough that the hub's wakeup process may not have completed by the time the second byte has been written).

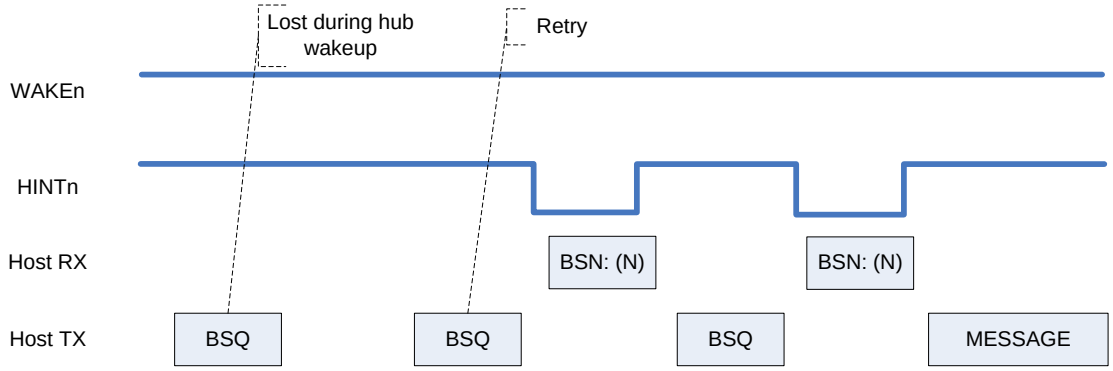


Figure 14: Host Write: WAKE always deasserted

5.0 Channel Usage

The command channel is reserved for use by the SHTP. All other channels are available for use by users of the SHTP.

5.1 Command Channel

The command channel is channel 0. Each command has an associated response. Commands are written by the host. Responses are read by the host. Write cargoes on channel 0 may consist of one or more commands. Read cargoes on channel 0 may consist of one or more responses.

The format of a command is shown in Figure 15.

Byte	Field
0	Command
1	P0
…	…
n	Pn-1

Figure 15: Command Channel Commands

Command	The command. Commands are one byte long. Each command has 0 to n-1 parameters. The number of parameters and their meaning are command dependent.
P0	Parameter 0 for the command
Pn-1	Parameter n-1 for the command

The format of a response is shown in Figure 15.

Byte	Field
0	Response
1	P0
…	…
n	Pn-1

Figure 16: Command Channel Responses

Response	The response. Responses are one byte long. Each response has 0 to n-1 parameters. The number of parameters and their meaning are response dependent.
P0	Parameter 0 for the response
Pn-1	Parameter n-1 for the response

5.1.1 Command 0: Get Advertisement

Command 0 is the advertise command. The advertise command tells the hub return an advertise response to the host. Parameter 0 is used to indicate if the advertise response should be for just the SHTP or if it should be for the entire hub. A value of 0 indicates that just the SHTP should respond. A value of 1 indicates that the response should be for the entire hub.

Other values are reserved. The purpose of this parameter is to allow the host to discover just the version of the SHTP without having to retrieve and parse a response for the entire hub.

5.1.1.1 Response 0: Advertisement

Response 0 is the advertise response. An advertise response contains information about the capabilities and limitation of the particular implementation of the SHTP on the hub and the allocation of channels to applications on the hub.

On system startup, the SHTP control application will send its full advertisement response, unsolicited, to the host.

5.1.2 Command 1: Send Error list

Request the hub to send its list of SHTP protocol errors accumulated since the last system reset.

5.1.2.1 Response 1: Error List

A response having command field set to 1 is sent whenever the hub encounters one or more recoverable error conditions in the SHTP protocol. It is sent automatically on error and does not require a corresponding host – hub command. New errors are added to the end of the list. Thus the length of the list will grow over time.

Note that if an error occurs that would normally be reported through this response, but an error list response is already pending, the new error will be dropped (reported to board support package internally).

The error list is not cleared until a system reset occurs.

Byte	Field
0	Response = 1
1	P0 = first error
…	…
N	Pn-1 = last error

Figure 17: Error List Response

Response	1 (Error list)
P0 - Pn-1	A list of 8-bit error codes.

The error codes in use are given below.

Error Code	Meaning
0	No error
1	Hub application attempted to exceed maximum read cargo length
2	Host write was too short (need at least a 4-byte header)
3	Host wrote a header with length greater than maximum write cargo length.
4	Host wrote a header with length less than or equal to header length (either invalid or no payload). Note that a length of 0 is permitted, indicating “no cargo.”
5	Host wrote beginning of fragmented cargo (transfer length was less than full cargo length), fragmentation not supported.
6	Host wrote continuation of fragmented cargo (continuation bit sent), fragmentation not supported.
7	Unrecognized command on control channel.

Error Code	Meaning
8	Unrecognized parameter to get-advertisement command.
9	Host wrote to unrecognized channel.
10	Advertisement request received while Advertisement Response was pending.
11	Host performed a write operation before the hub had finished sending its advertisement response.
12	Error list too long to send, truncated.

Figure 18: Error Codes

5.1.3 Commands/Responses 2 – 255

Commands 2 through 255 are reserved for future use.

5.2 Channel Advertisement

Channel advertisement has two general purposes. They are to convey details about the SHTP implementation to the host and convey channel assignments to the host. The host uses this information to determine how to communicate with the hub and how to associate channels with the users, or applications, that are using them.

Each application is assigned a globally unique identifier (GUID). This GUID is use as part of the advertisement response to associate information with a particular application.

The parameters of the advertise response carry information in tag-length-value (TLV) format. For consistency, even values with fixed lengths use the TLV format. For multi-byte numeric values, the bytes are ordered from LSB to MSB within the value field. Tags and lengths are each one byte. Within a TLV entry the tag appears first, followed by the length and then the value. The table of defined tags is shown in Figure 19.

Tags in the range of 0x80 through 0xff are free to be defined by applications as needed, tags in the range of 0x0B – 0x7F are reserved for future use as globally-defined tags.

Tag	Tag Name / Value
0	Reserved
1	GUID
2	MaxCargoPlusHeaderWrite
3	MaxCargoPlusHeaderRead
4	MaxTransferWrite
5	MaxTransferRead
6	NormalChannel
7	WakeChannel
8	AppName
9	ChannelName

Figure 19: Global Advertisement Tags

GUID	The GUID is assigned by Hillcrest and is unique across all applications on every hub. GUID 0 is assigned to the SHTP.
MaxCargoPlusHeaderWrite	Maximum length for a write cargo, including header bytes. This value is used for implementations that have some limit more restrictive than the limit of 2.2.1. This tag is used only for the SHTP GUID.

MaxCargoPlusHeaderRead	Maximum length for a read cargo, including header bytes. This value is used for implementations that have some limit more restrictive than the limit of 2.2.1. This tag is used only for the SHTP GUID.
MaxTransferWrite	Maximum length for a write transfer. This value is used for implementations that have some limit more restrictive than the limit of 2.2.1. This tag is used only for the SHTP GUID.
MaxTransferRead	Maximum length for a read transfer. This value is used for implementations that have some limit more restrictive than the limit of 2.2.1. This tag is used only for the SHTP GUID.
NormalChannel	A normal channel used by the application. Normal channels do not wake the host.
WakeChannel	A wake up channel used by the application. Wake up channels wake the host. This tag is used only for application GUIDs.
AppName	A user friendly string of the name of the application. The string must be null terminated.
ChannelName	A user friendly string of the name of this channel. The string must be null terminated. This is an optional tag and must appear immediately after a NormalChannel or a WakeChannel tag.

Tags are associated with the nearest preceding GUID tag. Unrecognized tags are ignored. An advertisement response consists of sets of a GUID tag followed by other tags. The sequence of a GUID tag followed by other tags repeats for each application. An example of the information conveyed in an advertise response is shown below.

GUID: 0
Version: 1.0.0
MaxCargoPlusHeaderWrite: 1024
MaxCargoPlusHeaderRead: 1024
MaxTransferWrite: 128
MaxTransferRead: 256
AppName: SHTP
NormalChannel: 0
ChannelName: control
GUID: 1
AppName: sensorhub
NormalChannel: 1
ChannelName: device
NormalChannel: 2
ChannelName: sensorhubControl
NormalChannel: 3
ChannelName: inputNormal
WakeChannel: 4
ChannelName: inputWake

If the MaxCargoPlusHeaderWrite tag is used without the MaxTransferWrite tag then the MaxTransferWrite value is assumed to equal the MaxCargoPlusHeaderWrite value. If the MaxCargoPlusHeaderRead tag is used without the MaxTransferRead tag then the MaxTransferRead value is assumed to equal the MaxCargoRead value.

5.3 SHTP Advertisement

The SHTP uses up to two application specific tag as part of its advertisement.

Tag	Tag Name / Value
0x80	SHTP Version
0x81	SHTP over UART Timeout

Figure 20: SHTP Advertisement Tags

Version	Required. Indicates version of the SHTP protocol. The format of the version field is a null terminated string of the version number. The format of the version number is <major>.<minor>.<patch>. The major, minor and patch fields are decimal numbers represented by the ASCII digits 0-9. These fields are separate by dots. An example version number is 2.3.1. The TLV encoding for this version is: 0x80 0x06 0x32 0x2E 0x33 0x2E 0x31 0x00 The version number fields may have more than one digit. Leading zeroes in a field are not allowed. Single zeroes are allowed. For example: 2.12.11 – valid 02.3.1 – invalid 2.0.1 – valid
SHTP over UART Timeout	Required for SHTP over UART, other protocols should not use. A 32-bit unsigned integer representing the amount of time (in milliseconds) after sending a Buffer Status Notification that the hub will remain active.

6.0 References

1. IETF, RFC 1662 https://tools.ietf.org/html/rfc1662#page-8

7.0 Notices

Information furnished by Hillcrest Laboratories, Inc. (Hillcrest) is believed to be accurate and reliable. However, Hillcrest assumes no responsibility for its use, nor for any infringement of patents or other rights of third parties that may result from its use.

Hillcrest reserves the right to make changes, corrections, modifications or improvements to this document at any time without notice. Information in this document supersedes and replaces all information previously supplied. Hillcrest makes no warranties, express or implied, regarding the information contained in this document.

Information in this document is provided solely to enable the use of Hillcrest products. “Typical” parameters provided by Hillcrest are not guaranteed, and can vary between applications and over time.

The product is designed primarily for use in consumer electronics and has not been evaluated for use in products or systems where failure or malfunction may result in personal injury, death, severe property damage or environmental damage, such as aerospace, life saving, life sustaining or military applications. Hillcrest assumes no liability for any claims or damages arising from information contained in this document, or from the use of products and services detailed herein. This exclusion includes, but is not limited to, claims or damages based on the infringement of patents, trademarks, copyrights and/or any other intellectual property rights. The product is provided by Hillcrest “As Is.” Hillcrest makes no representations or warranties regarding the product express or implied, including without limitation any implied warranties as to title, noninfringement, merchantability, or fitness for a particular purpose.

Freespace is a registered trademark of Hillcrest Laboratories, Inc. The Hillcrest Labs logo is a trademark of Hillcrest Laboratories, Inc. All other trademarks and copyrights are the property of their respective owners.

“Third party notices are available on our website at: http://hillcrestlabs.com/legal/third-party-attributions/”