UG101: UART-EZSP Gateway Protocol Reference

This document describes the protocol used by the UART-based EmberZNet Serial Protocol (UART-EZSP) to reliably carry commands and responses between a host processor and a Network Co-Processor (NCP).

KEY FEATURES

- Compares the benefits of communications using UART and SPI.
- Describes the EZSP-UART general frame format.
- Discusses the various types of frames.
- Describes ASH protocol operations.
- Reviews the characteristics of the CONNECTED and FAILED states.
- Summarizes configuration parameters.

Table of Contents

1. Overview

Silicon Labs designed the Asynchronous Serial Host (ASH) protocol for network gateway systems in which the host processor is running a standard operating system such as embedded Linux or Windows. This document describes ASH version 2. For information on ASH v3, refer to *UG115: ASH v3 Protocol Reference*.

The host communicates with a Network Co-Processor (NCP) that runs EmberZNet PRO. The NCP is a chip from the Silicon Labs portfolio of wireless mesh chips which is programmed with Silicon Labs-supplied EmberZNet Serial Protocol (EZSP) UART or SPI NCP firmware.

EZSP-UART using ASH may be a better choice than the SPI protocol if:

- The Host processor is running a large-scale multi-tasking operating system that can benefit from the protocol's tolerance for Host response delays.
- The serial connection between the Host and the Network Co-Processor is slightly noisy due to a long or unshielded cable.
- The Host does not have enough GPIO pins available to implement the SPI interface.

ASH may not be a good choice in other systems for the following reasons:

- On some platforms, ASH does not allow the Network Co-Processor to sleep and thus cannot be used on sleepy devices.
- Command execution may take longer due to the lower UART speed and the need for messages to be acknowledged.
- The Host processor must supply more RAM and program memory, and perform more processing.

The ASH protocol is a data-link layer protocol below EZSP and above the serial device (or UART) driver. The following figure shows the ASH layer's relationship to other software layers in the Host and Network Co-Processor.
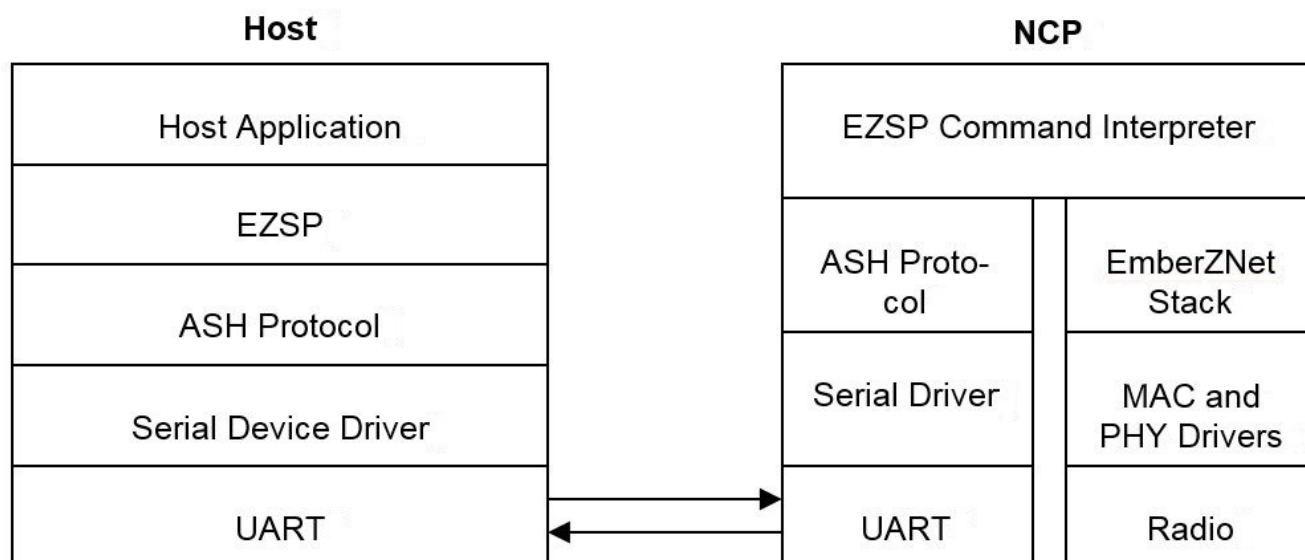

Figure 1.1. ASH Layer Diagram

The ASH protocol includes features for reliable and efficient communication between the host processor and the Network Co-Processor:

- Frame-based protocol with only four (4) bytes overhead per message
- All frames are validated using a 16-bit cyclical redundancy check (CRC)
- Byte stuffing provides binary transparency, so any binary data can be sent and received
- Sliding window acknowledgement performs better than simple stop and wait protocols
- Command responses can be given priority over callbacks to prevent host receive queue overflow
- Acknowledgement timeouts adapt to actual timings

Silicon Labs supplies EZSP-UART for the NCP as a program image, and for the host as C-language software in source form that is compatible with both Linux and Windows. The ASH host software is ready to compile and run on most PCs, and can usually be adapted to a particular embedded host processor with a few simple edits.

2. Frames

A *frame* is the smallest unit transmitted between the Host and the NCP. A frame is composed of a Control Byte, possibly a Data Field (depending on the frame type), a 2-byte CRC, and a Flag Byte.

The following figure illustrates the general format used by all ASH frame types.
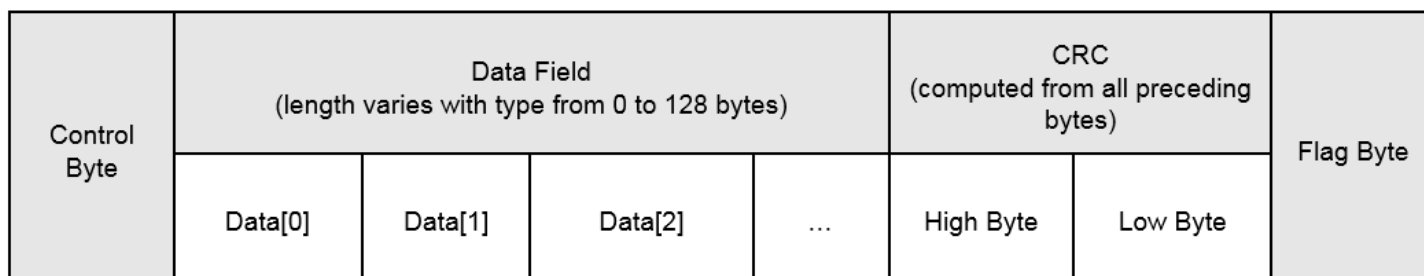
Control Byte	Data Field (length varies with type from 0 to 128 bytes)				CRC (computed from all preceding bytes)		Flag Byte
	Data[0]	Data[1]	Data[2]	…	High Byte	Low Byte	

Figure 2.1. General ASH Frame Format

2.1 Control Byte

The Control Byte identifies the type of frame and contains various bit fields as shown in the following table. Subsequent sections provide detailed information on the different frame Control Bytes.

Table 2.1. Frame Control Bytes

Frame Type	Data Field Size (Bytes)	Sent By	Control Byte Value (Bit Field)							
			b7	b6	b5	b4	b3	b2	b1	b0
DATA	varies: 3-128	NCP, Host	0	frmNum			reTx	ackNum		
ACK	0	NCP, Host	1	0	0	res	nRdy	ackNum		
NAK	0	NCP, Host	1	0	1	res	nRdy	ackNum		
RST	0	Host	1	1	0	0	0	0	0	0
RSTACK	2	NCP	1	1	0	0	0	0	0	1
ERROR	2	NCP	1	1	0	0	0	0	1	0

- frmNum - DATA frame's 3-bit sequence number
- ackNum - acknowledges receipt of DATA frames up to, but not including, ackNum
- reTx - set to 1 in a retransmitted DATA frame; 0 otherwise
- nRdy – host sets to 1 if to inhibit the NCP from sending callbacks frames to the host (always 0 in frames sent by the NCP)
- res - reserved for future use: may be either 0 or 1

2.2 Data Field

The Data Field carries information for DATA, RSTACK, and ERROR frame types. The length of the Data Field is two (2) bytes for RSTACK and ERROR frame types. For DATA frames, the length of the Data Field is variable and sized to fit the EZSP frame being sent.

2.3 CRC

A 16-bit CRC is computed on all bytes in a frame preceding the CRC, excluding bytes added by byte stuffing. The standard CRC-CCITT ($g(x) = x16 + x12 + x5 + 1$) is initialized to 0xFFFF. The most significant byte precedes the least significant byte (big-endian mode).

2.4 Flag Byte

A Flag Byte, 0x7E, marks the end of a frame. Consecutive Flag Bytes after the first Flag Byte are ignored.

2.5 Frame Lengths

The length of a frame is at least four bytes—Control Byte, two CRC bytes, and a Flag Byte—plus the length of the Data Field, if any:
- RST, ACK, and NAK frames are 4 bytes long (no Data Field).
- RSTACK and ERROR frames are 6 bytes long (2-byte Data Field).
- DATA frames range from 7 through 132 bytes long (3- to 128-byte Data Field).

2.6 Frame Numbers

DATA frames are numbered using 3-bit sequence numbers that count up from 0 to 7 and then wrap back to 0. They are used to acknowledge properly received frames and allow detection of a missing frame. The frame number is assigned when a frame is first sent, and stays the same if the frame is retransmitted.

3. Frame Formats

This section describes each type of ASH frame, including its format and purpose, the notation used in documentation, and some examples.

3.1 RST Frame Format

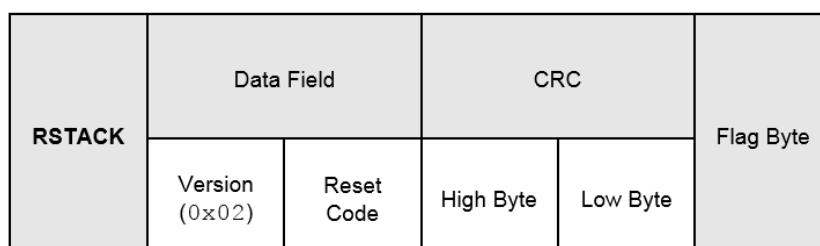The following figure illustrates the format of the RST frame.
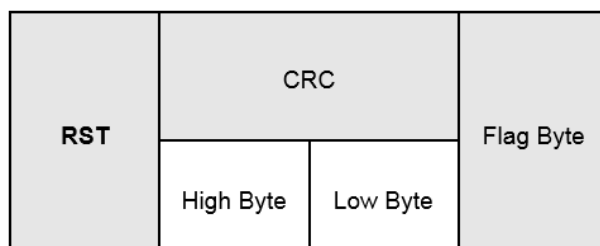
Figure 3.1. RST Frame Format

- Purpose: Requests the NCP to perform a software reset (valid even if the NCP is in the FAILED state).
- Notation used in documentation: RST()
- Example: `C0 38 BC 7E`

3.2 RSTACK Frame Format

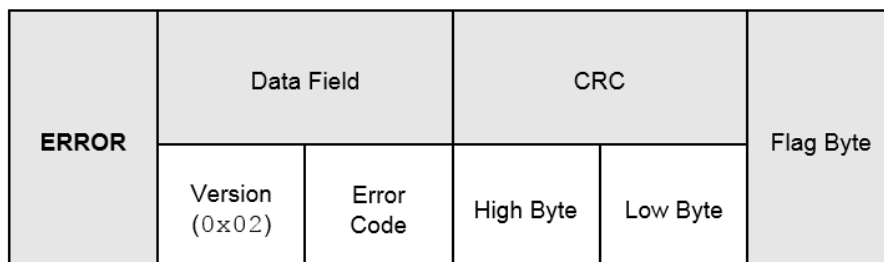The following figure illustrates the format of the RSTACK frame.

Figure 3.2. RSTACK Frame Format

- Purpose: Informs the Host that the NCP has reset and the reason for the reset.
- Data field:
  - Version: Fixed at 0x02 in this version of ASH
  - Reset Code: See Table 6.1 Reset and Error Codes on page 18
  - Notation used in documentation: RSTACK(V, C)
    - V: version
    - C: reset code
- Example: `C1 02 02 9B 7B 7E`

3.3 ERROR Frame Format

The following figure illustrates the format of the ERROR frame.

Figure 3.3. ERROR Frame Format

- Purpose: Informs the Host that the NCP detected a fatal error and is in the FAILED state.
- Data field:
  - Version: Fixed at 0x02 in this version of ASH
  - Error Code: See Table 6.1 Reset and Error Codes on page 18
  - Notation used in documentation: ERROR(V, C)
    - V: version
    - C: reset code
- Example: `C2 01 52 FA BD 7E`

3.4 DATA Frame Format

The following figure illustrates the format of the DATA frame.
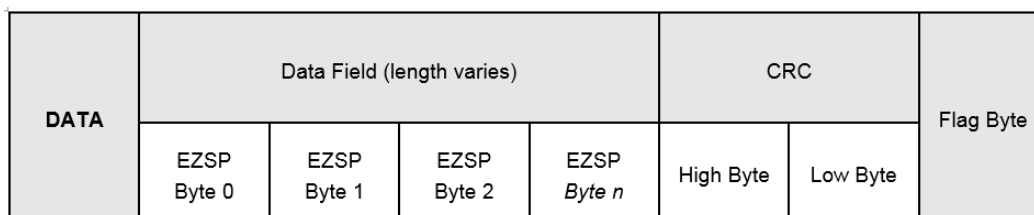
Figure 3.4. DATA Frame Format

- Purpose: Carries all EZSP frames.
- Data Field:
  - EZSP frame: exclusive-OR'ed with a pseudo-random sequence (see the section Sending frames)
- Notation used in documentation: DATA(F, A, R)
  - F: frame number (frmNum)
  - A: acknowledge number (ackNum)
  - R: retransmit flag (reTx)
- Examples — without pseudo-random sequence applied to Data Field:
  - EZSP "version" command: 00 00 00 02
  - DATA(2, 5, 0) = 25 00 00 00 02 1A AD 7E
  - EZSP "version" response: 00 80 00 02 02 11 30
  - DATA(5, 3, 0) = 53 00 80 00 02 02 11 30 63 16 7E
- Examples — with pseudo-random sequence applied to Data Field:
  - EZSP "version" command: 00 00 00 02
  - DATA(2, 5, 0) = 25 42 21 A8 56 A6 09 7E
  - EZSP "version" response: 00 80 00 02 02 11 30
  - DATA(5, 3, 0) = 53 42 A1 A8 56 28 04 82 96 23 7E

3.5 ACK Frame Format

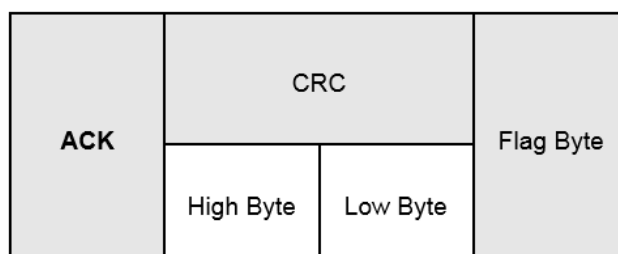The following figure illustrates the format of the ACK frame.

Figure 3.5. ACK Frame Format

- Purpose: Acknowledges receipt of a valid DATA frame.
- Notation used in documentation: ACK(A)+/-
  - A: acknowledge number (ackNum)
  - +/-: not ready flag (nRdy); "+" = "0" = "ready"; "-" = "1" = "not ready"
- Examples:
  - ACK(1)+ :81 60 59 7E
  - ACK(6)– : 8E 91 B6 7E

3.6 NAK Frame Format

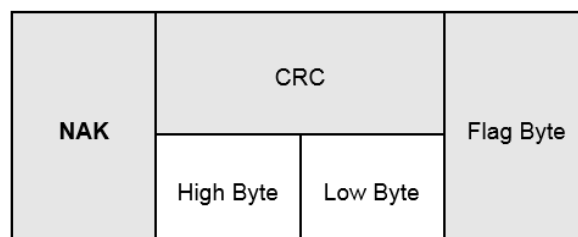The following figure illustrates the format of the NAK frame.

Figure 3.6. NAK Frame Format

- Purpose: Indicates receipt of a DATA frame with an error or that was discarded due to lack of memory.
- Notation used in documentation: NAK(A)+/-
  - A: acknowledge number (ackNum)
  - +/-: not ready flag (nRdy); "+" = "0" = "ready"; "-" = "1" = "not ready"
- Examples:
  - ACK(6)+ : A6 34 DC 7E
  - ACK(5)- : AD 85 B7 7E

4. Protocol Operations

This section describes how the ASH protocol sends and receives frames, including the method used to achieve binary data transparency.

4.1 Sending Frames

Before ASH sends a DATA frame, it performs this series of steps:

1. The Control Byte is added before the Data Field. The frmNum field is set to the last frame transmitted plus one, and the ackNum field is set to the number of the next frame expected to be received. The reTx flag is clear.
2. The Data Field is exclusive-OR'ed with a pseudo-random sequence (see Data Randomization).
3. The two-byte CRC of the Control Byte plus the Data Field is computed and appended after the Data Field.
4. The frame is byte stuffed to remove reserved byte values (see Reserved Bytes and Byte Stuffing).
5. A Flag Byte is added after the CRC.

If a DATA frame is retransmitted, the process is the same except for step 1. The frmNum field retains the same value as when the frame was first transmitted, and the reTx bit is set. The ackNum is the current value as in normal transmission.

Other frame types omit step 2 and have differently formatted Control Bytes, but otherwise they use the same process.

4.2 Reserved Bytes and Byte Stuffing

ASH reserves certain byte values for special functions. If bytes with these values happen to occur within a frame, ASH uses a process known as byte stuffing to replace those bytes so they have non-reserved values. Byte stuffing is performed on the entire ASH frame except for the Flag Byte. The receiver reverses the process to recover the original frame contents.

The following table lists the byte values that are reserved in ASH.

Table 4.1. ASH Reserved Byte Values

Value	Special Function
0x7E	Flag Byte: Marks the end of a frame. When a Flag Byte is received, the data received since the last Flag Byte or Cancel Byte is tested to see whether it is a valid frame.
0x7D	Escape Byte: Indicates that the following byte is escaped. If the byte after the Escape Byte is not a reserved byte, bit 5 of the byte is complemented to restore its original value. If the byte after the Escape Byte is a reserved value, the Escape Byte has no effect.
0x11	XON: Resume transmission. Used in XON/XOFF flow control. Always ignored if received by the NCP.
0x13	XOFF: Stop transmission. Used in XON/XOFF flow control. Always ignored if received by the NCP.
0x18	Substitute Byte: Replaces a byte received with a low-level communication error (e.g., framing error) from the UART. When a Substitute Byte is processed, the data between the previous and the next Flag Bytes is ignored.
0x1A	Cancel Byte: Terminates a frame in progress. A Cancel Byte causes all data received since the previous Flag Byte to be ignored. Note that as a special case, RST and RSTACK frames are preceded by Cancel Bytes to ignore any link startup noise.

To byte stuff, an Escape Byte is sent first, followed by the reserved byte being escaped with bit 5 of the reserved byte inverted. When a frame is received, byte stuffing is reversed to restore the original data.

Note: The Escape, Substitute, and Cancel Bytes are not the same as the ESC, SUB, and CAN ASCII control characters.

The following list illustrates some examples of possible byte stuffing:

- Flag Byte 7E is sent as 7D 5E
- XON Byte 11 is sent as 7D 31
- XOFF Byte 13 is sent as 7D 33
- Substitute Byte 18 is sent as 7D 38
- Cancel Byte 1A is sent as 7D 3A
- Escape Byte 7D is sent as 7D 5D

The byte value 0xFF has a special meaning when it occurs between ASH frames in ASH versions that allow sleeping by the NCP. (Within a frame, 0xFF is not special and does not need to be escaped.) 0xFF is sent by the Host to wake the NCP, and is echoed back when the NCP wakes up. The NCP can also wake the Host by sending 0xFF, although the Host does not echo it back to the NCP.

EmberZNet Serial Protocol (EZSP) on the NCP may also send a 0xFF byte to inform the host that it has a callback pending while it is operating in the synchronous (polled) callback mode.

4.3 Data Randomization

If a DATA frame contains many reserved bytes, byte stuffing can nearly double the length of the frame. To make this less likely, a DATA frame's Data Field is exclusive-OR'ed with the output of a pseudo-random sequence before byte stuffing. The Data Field is restored by exclusive-OR'ing again at the receiver.

The pseudo-random sequence is reinitialized at the start of every Data Field and is generated by an 8-bit linear feedback shift register described by the following bitwise equations:

- $rand0 = 0x42$
- if bit 0 of $randi$ is 0, $randi+1 = randi >> 1$
- if bit 0 of $randi$ is 1, $randi+1 = (randi >> 1)$ ^ $0xB8$

The sequence starts {0x42, 0x21, 0xA8, 0x54, 0x2A, …}.

4.4 Receiving Frames

When data is received, first any byte stuffing is reversed by inverting bit 5 of bytes following Escape Bytes. The Control Byte, the first byte received in a frame, is saved, and if the buffer memory is available, the following Data Field bytes are stored. Regardless of buffer availability, the frame's length and CRC are computed as data is input. If a Cancel Byte or Substitute Byte is received, the bytes received so far are discarded. In the case of a Substitute Byte, subsequent bytes will also be discarded until the next Flag Byte.

Normally, a Flag Byte marks the end of a frame, which is then validated. To be valid, a frame must:
- Have a correct CRC.
- Have a Control Byte corresponding to a valid frame type:
    - On the Host: DATA, ACK, NAK, RSTACK, or ERROR
    - On the NCP: DATA, ACK, NAK, or RST
- Have a valid Data Field length for the frame type.
- Have a valid ackNum if in the CONNECTED state and the frame type is DATA, ACK, or NAK.

A frame that fails any of these criteria is discarded. If ASH is in the CONNECTED state and the Reject Condition is not already set, the Reject Condition is set and a NAK is sent. This is also done if a Substitute Byte was received within a frame.

5. CONNECTED State

The Host initializes the ASH protocol by resetting the NCP through either the nRESET pin or sending the RST frame. When the NCP is fully booted and ready to interact with the Host, the NCP will send the RSTACK frame to the Host.

Due to possible I/O buffering, it is important to note that the Host could receive several valid or invalid frames after triggering a reset of the NCP. The Host must discard all frames and errors until a valid RSTACK frame is received. The Host must also allow a certain amount of time to receive the RSTACK frame; the parameter T_RSTACK_MAX defines this timeout. If the Host does not receive the RSTACK frame within the timeout period, the Host should retry the reset process up to five times in case of noise interfering with the RST or RSTACK frames.

When a RSTACK frame is received, the Host must perform a final verification of the Version number provided in the RSTACK's Data Field. If the Version number being supplied by the NCP is compatible with the Host's version number for ASH, the Host and the NCP have successfully transitioned into the CONNECTED state and can begin to exchange EZSP messages.

5.1 Frame Numbers

ASH uses 3-bit frame numbers to track reception of DATA frames and detect when a frame is lost in transmission. The bit fields labeled frmNum in the Control Byte of DATA frames are numbered sequentially by the sender, 0 through 7 and back to 0, and the receiver expects to get consecutively numbered frames.

Both the NCP and the Host maintain two frame numbers, because the frame numbers for data sent from the Host to the NCP are independent of those sent from the NCP to the Host. Thus the frmNum field in a DATA frame belongs to the sequence number in one direction, and the ackNum field in a DATA, ACK, or NAK frame belongs to the sequence number in the reverse direction.

The Host and the NCP maintain a sliding window for DATA frames, and this window must be smaller than the 8 frames allowed by the 3-bit frame number field. The effect of the sliding window is that a sender may transmit multiple frames before receiving an acknowledgement, and the window's smaller size means that neither side has to worry about the frame numbers wrapping and mistakenly acknowledging the wrong frame.

5.2 Acknowledgements and Frame Numbers

The ackNum field in DATA, ACK, and NAK frames acknowledges received DATA frames. Note that ackNum is the number of the next frame the receiver expects and it is one greater than the last frame received.

The NCP discards any frames it receives that have an invalid ackNum value. A valid ackNum is a number between the last received ackNum and the last transmitted frmNum plus one, where both limits are inclusive.

The maximum number of frames a sender can transmit without them being acknowledged is the window size, which is specified by the parameter TX_K. When a sender has TX_K unacknowledged frames, it may not send any more, although it may retransmit frames if needed.

5.3 Piggybacked Acknowledgements

The NCP, but not the Host, may "piggyback" acknowledgements on a DATA frame. This is more efficient than sending a separate ACK frame, but if there is no DATA frame ready to send, the NCP will send an ACK frame instead. The NCP delays sending an ACK after it receives a DATA frame. This delay increases the likelihood that an acknowledgement is piggybacked on a DATA frame or multiple acknowledgements are sent with one ACK frame. The ACK delay parameter, T_TX_ACK_DELAY, is the artificial delay that specifies how long to wait before sending an ACK frame when there is no DATA frame to send.

The Host may not piggyback acknowledgments and should promptly send an ACK frame when it receives a DATA frame. The NCP will normally reject DATA frames received with piggybacked ACKs because it is not able to buffer them.

5.4 Sending DATA Frames

A given DATA frame can be sent if there is no higher priority ACK or NAK frame to be transmitted, the number of frames already transmitted without being acknowledged is not at the window size limit, and the receiver is ready to accept the frame. When first sent, the frmNum bit field is assigned the next consecutive frame number, and its ackNum bit field is set to the frame number of the next frame expected to be received. The Data Field is then exclusive-OR'ed with a pseudo-random sequence. Finally, the entire frame, excluding the Flag Byte, is byte stuffed.

After the DATA frame has been written to the UART buffer, it is saved on an ASH queue so that it can be retransmitted if required. When a frame is acknowledged, it is deleted from the queue. If an acknowledgement is not received in time, the frame will be retransmitted.

5.5 Receiving DATA Frames

The Data Field of a DATA frame is only passed up to EZSP if it is valid, in sequence, and there is enough memory to buffer the data. Before passing the data to EZSP, the data is exclusive-OR'ed with the pseudo-random sequence to restore the original contents. The ackNum field is always processed, because this information is separate from the Data Field or frmNum. Even if the DATA frame is discarded due to being a duplicate, out of sequence, or lack of memory, the ackNum field is still valid and will be processed to update the acknowledged frames.

5.6 DATA frame Acknowledgement timing

The NCP adjusts the time allowed to receive an acknowledgement for a DATA frame based on how long it has taken to receive previous acknowledgements. In general, the NCP waits four times the rolling average of measured acknowledgement times before timing out. Note that a DATA frame may be acknowledged by the ackNum bit field in an ACK, NAK, or DATA frame.

Three parameters govern computing the DATA frame acknowledgement timeout: t_rx_ack: T_RX_ACKINIT, T_RX_ACK_MIN, and T_RX_ACK_MAX. These are the initial, the minimum, and the maximum values, respectively, that are assigned to t_rx_ack.

At startup, t_rx_ack is set to T_RX_ACK_INIT. Whenever an acknowledgement is received, t_rx_ack is set to 7/8 of its current value plus 1/2 of the measured time for the acknowledgement. Therefore, over time t_rx_ack equals four times the time-weighted average of actual acknowledgement times. If a DATA frame acknowledgement is not received within the current timeout value, then t_rx_ack is doubled. The value of t_rx_ack is always limited to the range from T_RX_ACK_MIN to T_RX_ACK_MAX.

If a frame is not acknowledged within t_rx_ack, it times out and the NCP will retransmit the frame. If there are enough consecutive timeouts, the NCP enters the ERROR state. The maximum number of consecutive timeouts is given by the parameter ACK_TIMEOUTS.

5.7 DATA Frame Flow Control

The Host uses DATA frame flow control to throttle the transmission of DATA frames from the NCP, but the NCP does not control Host transmissions to it. NCP DATA frames are either responses to EZSP commands or callbacks that are sent without being requested by the Host. DATA frame flow control applies only to callback DATA frames, and does not stop the NCP from sending command response DATA frames.

When the Host is not ready to accept additional callback DATA frames because it does not have enough free buffers to store them, it instructs the NCP to pause callback DATA frame transmission by sending an ACK or NAK frame with the nRdy (not ready) flag set to 1. When callback DATA frame transmissions are paused, the NCP is still allowed to transmit ACK and NAK frames, transmit command response DATA frames, and retransmit callback DATA frames if required to recover from errors. When the Host is ready again, it sends an ACK or NAK with nRdy set to 0.

After it receives an ACK or NAK with nRdy set, the NCP may not transmit a callback DATA frame for a period of time defined by the parameter T_REMOTE_NOTRDY. As long as the Host remains in the not ready state, the Host sends an ACK or NAK frame with the nRdy flag set at intervals of time defined by the parameter T_LOCAL_NOTRDY. The NCP may resume sending callback DATA frames if it does not receive another ACK or NAK with the nRdy flag set after the period T_REMOTE_NOTRDY has elapsed. This prevents the NCP being held off indefinitely if an ACK or NAK with the nRdy flag cleared is lost.

DATA frame flow control does not eliminate the need for low-level (RTS/CTS or XON/XOFF) flow control. Low-level flow control helps ensure that serial data is not lost at the UART due to overruns or overflows, while DATA frame flow control operates at the ASH level and prevents loss of DATA frames due to lack of buffer memory.

5.8 Data Exchange Example

The following figure illustrates a simple timeline providing an example of basic DATA and ACK frame usage and the sequence numbering involved.
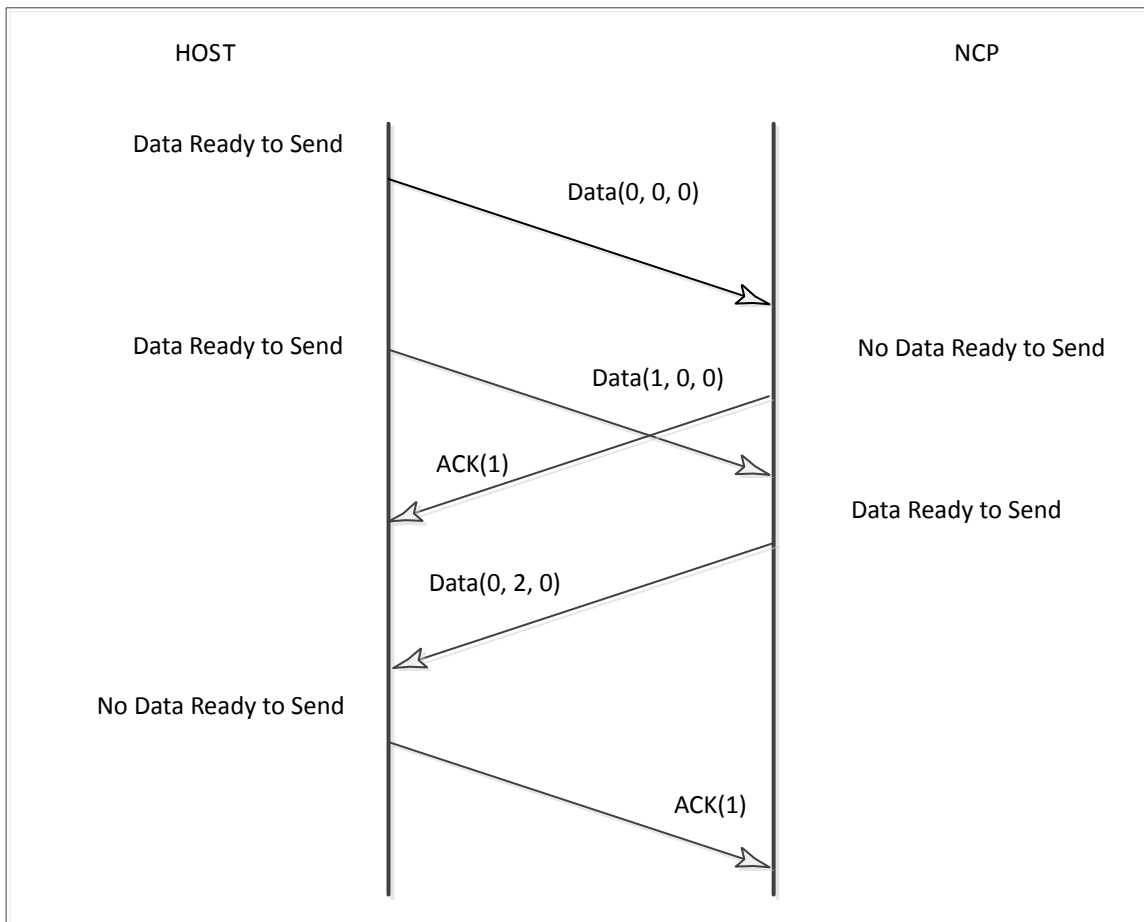
Figure 5.1. Two-Way Data Exchange with Acknowledgements

1. The Host sends a DATA frame with frmNum 0 to the NCP.
2. The NCP has no data ready to send, so it replies with an ACK frame with ackNum 1, the next frame the NCP expects to receive.
3. The Host has more data to send, so it sends DATA frame with frmNum 1 before it receives the ACK frame. The Host now has two unacknowledged frames outstanding.
4. The NCP has data to send, so it sends its DATA frame with frmNum 0 and ackNum 2 piggybacked on the DATA frame as an acknowledgement of the Host's DATA frame 1.
5. The Host has no further data to send, so it sends just an ACK for the frame the Host just received.

5.9 Reject Condition and NAK Frames

ASH sets the Reject Condition after receiving a DATA frame with any of the following attributes:

- Has an incorrect CRC.
- Has an invalid control byte.
- Is an invalid length for the frame type.
- Contains a low-level communication error (e.g., framing, overrun, or overflow).
- Has an invalid ackNum.
- Is out of sequence.
- Was valid, but had to be discarded due to lack of memory to store it.

A frame is out of sequence if its frmNum is not one greater than the last (non-retransmitted) frame received. (Remember that sequence numbers are 3 bits and the modulo 8 operation means 0 is the number after 7.) Retransmitted frames are never considered out of sequence.

If the Reject Condition is set when it is currently clear, a NAK frame is sent. The Reject Condition is cleared as soon as a valid, in-sequence DATA frame is received and can be buffered.

When a NAK is received, the receiver of the NAK begins retransmitting if it is not already doing so, assuming it has transmitted any frames that have not been acknowledged. If a new DATA frame is in the process of being sent, the new DATA frame transmission is cancelled by sending a Cancel Byte because it would be discarded as out of sequence anyway, and retransmission is started more quickly this way.

5.10 Retransmitting

A DATA frame can be retransmitted for one of two reasons:

* The DATA frame was not acknowledged within the allowed time.
* A NAK was received indicating that the DATA frame was not received correctly.

Retransmission begins with the oldest unacknowledged frame, and all retransmitted frames are flagged by setting the reTx flag in the Control Byte. A retransmitted frame keeps the same frmNum as when first transmitted, but the ackNum bit field may differ if the sender has received more DATA frames since it was first transmitted.

If the sender receives an acknowledgement for a frame due to be retransmitted, that frame will not be retransmitted. When done re-transmitting, the sender resumes sending new DATA frames again.

Upon receiving a retransmitted frame, the receiver sends an immediate ACK. The receiver does not set the Reject Condition or send a NAK when a retransmitted frame is received. If the retransmitted frame was previously received correctly, the Data Field is discarded and the ackNum is processed as usual.

In the example shown in the following figure, the second DATA frame sent by the NCP is lost, and this is detected by the Host when it receives an out-of-sequence frame. At this point the Host sends a NAK and sets its Reject Condition. Meanwhile, another DATA frame is being sent by the NCP, and when the Host receives this, it is also out of sequence; however, since the Reject Condition is already set, the Host does not send another NAK.

When the NCP receives the NAK, the NCP begins retransmitting starting from its oldest unacknowledged frame. When the Host receives the first in-sequence frame, it clears the Reject Condition and sends an ACK. Finally, the NCP finishes sending all of the unacknowledged frames and the NCP clears the Retransmit Condition, resuming transmission of new data frames.

In the simple example illustrated in the following figure, the Host did not send any DATA frames to the NCP, but it could have been sending DATA frames throughout the error recovery process. ASH data transfer, error detection, and error recovery operate independently and simultaneously in both directions.
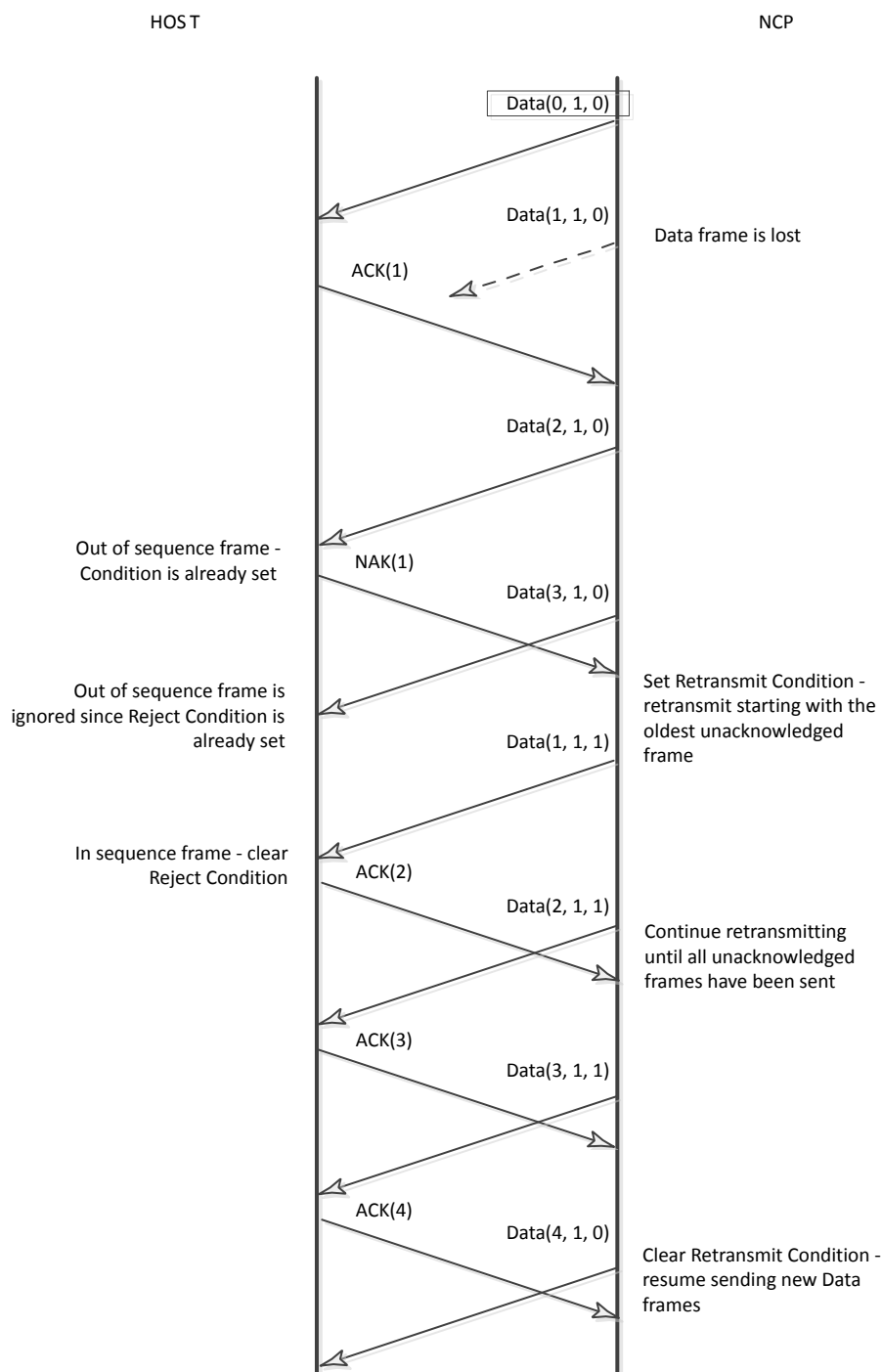
Figure 5.2. NAK Error Recovery

6. FAILED State

The NCP enters the FAILED state if it detects one of the following errors:
* An abnormal internal reset due to an error, failed assertion, or fault.
* Exceeding the maximum number of consecutive acknowledgement timeouts.

When the NCP enters the FAILED state, the NCP sends an ERROR frame containing a reset or error code and will reply to all subsequent frames received, except RST, with an ERROR frame. To reinitialize the ASH protocol, the Host must reset the NCP by either asserting the nRESET pin or sending the RST frame. The following table lists the codes returned by the NCP in the Reset Code byte of a RSTACK frame or in the Error Code byte of an ERROR frame. Silicon Labs wireless mesh chips can detect numerous reset fault causes beyond those in the table. When sent to the host, these new reset codes have 0x80 added to the value returned by their HAL's reset code.

Table 6.1. Reset and Error Codes

Code	Meaning
0x00	Reset: Unknown reason
0x01	Reset: External
0x02	Reset: Power-on
0x03	Reset: Watchdog
0x06	Reset: Assert
0x09	Reset: Boot loader
0x0B	Reset: Software
0x51	Error: Exceeded maximum ACK timeout count
0x80	Chip-specific error reset code

7. Configuration Parameters

Several parameters stored as manufacturing tokens control the operation of the ASH protocol. The values are optimized for operation at 115,200 bps using RTS/CTS flow control, or 57,600 bps when using XON/XOFF software flow control.

Silicon Labs has tuned the ASH protocol to operate best with the configurations listed in the following two tables. While you can change these configurations through manufacturing tokens, doing so may degrade performance. To learn how to change the configurations, contact customer support at https://www.silabs.com/community.

Table 7.1. Timing Parameters

Parameter	Value	Units	Description
T_RX_ACK_INIT	1.6	sec	Initial value of t_rx_ack, the maximum time the NCP waits to receive acknowledgement of a DATA frame
T_RX_ACK_MIN	0.4	sec	Minimum value of t_rx_ack
T_RX_ACK_MAX	3.2	sec	Maximum value of t_rx_ack
T_TX_ACK_DELAY	20	msec	Delay before sending a non-piggybacked acknowledgement
T_REMOTE_NOTRDY	1.0	sec	Time from receiving an ACK or NAK with the nRdy flag set after which the NCP resumes sending callback frames to the host without requiring an ACK or NAK with the nRdy flag clear

Table 7.2. Other Parameters

Parameter	Value	Description
BAUD_RATE	115200 (RST/CTS) / 57600 (XON/XOFF)	Transmit and receive baud rate
TX_K	5	Maximum number of DATA frames the NCP can transmit without having received acknowledgements
ACK_TIMEOUTS	4	Maximum number of consecutive timeouts allowed while waiting to receive an ACK before going to the FAILED state. The value 0 prevents the NCP from entering the error state due to timeouts.
RANDOMIZE	enabled	Enables randomization of received and transmitted DATA frame Data Fields; can be disabled for debugging

Smart. Connected. Energy-Friendly.

IoT Portfolio
www.silabs.com/products

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

www.silabs.com