UG115: ASHv3 Protocol Reference

This document describes the protocol used by the UART-based ASHv3 to reliably carry commands and responses between a host processor and a network co-processor (NCP).

Proprietary is supported on all EFR32FG devices. For others, check the device's data sheet under Ordering Information > Protocol Stack to see if Proprietary is supported. In Proprietary SDK version 2.7.n, Connect is not supported on EFR32xG22.

KEY POINTS

- Introduces ASHv3.
- Lists the files used by the Host and NCP for UART communications.
- Describes the ASHv3 protocol.
- Presents the ASHv3 frame formats.
- Provides examples of ASHv3 communications.
- Includes the default pin configurations for NCP UART.

1. ASHv3 Introduction

ASHv3 is the third revision of the Asynchronous Serial Host (ASH). It is a reliable and efficient UART communication protocol that is used to facilitate NCP and Host communication. ASHv3 features session synchronization, a retry mechanism, and CRC error detection. ASHv3 operates on the same level as the UART and interfaces directly with it.

The following figure illustrates how ASHv3 and UART communicate.

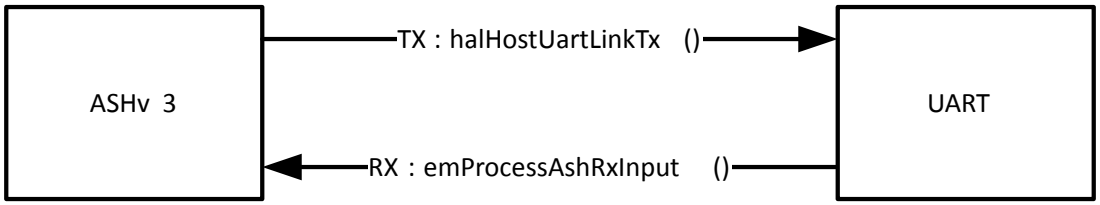


Figure 1.1. ASHv3-UART Communication

- TX: halHostUartLinkTx()

  ASHv3 transmits bytes to the UART using halHostUartLinkTx().
- RX: emProcessAshRxInput()

  The UART sends received bytes to ASHv3 using emProcessAshRxInput().

2. UART Layer Files

The Host and NCP have shared and specific files for their UART communication layers, as shown in the following table. The details are in the following subsections.

Table 2.1. UART Layer Files

UART Layer Files	Silicon Labs Connect
Common Files for Host and NCP	hal/micro/generic/ash-v3.c
	plugins/command-interpreter/command-interpreter2-binary.c
	plugins/host-ncp/binary-management.c
Host-Specific Files	plugins/host-ncp/serial-link-host.c
	plugins/host-ncp/host-buffer-management.c
	plugins/host-ncp/host-stream.c
NCP-Specific Files	plugins/host-ncp/ncp-link.c
	plugins/host-ncp/serial-link-ncp.c

3. ASHv3 Protocol

The following subsections describe the ASHv3 protocol in more detail.

3.1 Design

The application sends and receives a stream of bytes. The application writes and reads data in blocks, but the block boundaries are not preserved in ASHv3 frames. ASHv3 framing is independent of the upward interface; the data is treated as a stream.

The ASHv3 protocol uses a single DMA (Direct Memory Access) operation to send a frame. The maximum size of a DMA buffer used in ASHv3 is 64 bytes. Obtaining good throughput requires allowing two frames in flight; having more does not help. ASHv3 is responsible for ACKing and retrying frames over the wire.

ASHv3 uses the SC1 serial port only. Developers are not allowed to designate another serial port.

3.2 Frame Counters

ASHv3 uses 3-bit frame numbers to track reception of frames and detect when a frame is lost in transmission. The bit fields labeled OFC (Outgoing Frame Counter) and AFC (ACK/NACK Frame Counter) in the Control Byte are numbered sequentially 1 through 7 and back to 1 by the sender; the receiver expects to receive 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. As a result, the OFC field in a DATA frame belongs to the sequence number in one direction, and the AFC field in ACK or NACK frames belongs to the sequence number in the reverse direction.

The maximum number of frames a sender can transmit without them being acknowledged by the receiver is two.

3.3 Message Payload

ACK, NACK, and RESET ACK frames can contain application payloads. The maximum payload size is 57 bytes.

3.4 Resets

To reset the UART link, each device sends a RESET frame and its peer responds with a RESET ACK frame. A RESET packet has OFC/AFC [1/0]. A RESET ACK frame with no payload has OFC/AFC [1/1]. A RESET ACK with non-empty payload has OFC/AFC [2/1].

3.5 Retransmission

ASHv3 retains transmitted frames until a matching ACK is received. Frames are retransmitted if a NACK is received or, in the case when an ACK is lost, after 500 milliseconds.

3.6 ACKs, NACKs, and RESET ACKs

ASHv3 will NACK any frame that has any of the following conditions:

- Incorrect CRC
- Invalid outgoing frame counter
- Invalid payload length
- Does not start with ASH FLAG (0x7E)
- The payload ends in an escape byte
- RESET packet with non-empty payload
- RESET with an outgoing frame counter not equal to 1
- RESET with an ACK/NACK frame counter not equal to 0

ACK, NACK, and RESET ACK frames can carry data. As such, there is no explicit 'DATA' frame type. The ACK/NACK counter has the OFC from the last correctly received frame. A NACK frame is sent if a corrupt frame is received. An ACK is sent whenever a frame that contains data is received correctly. If an ACK or NACK has no payload, the receiving side ignores its OFC, and no ACK is sent in response.

Note: A RESET ACK's OFC is recorded whether or not its payload is populated. An ACK, NACK, or RESET ACK with an empty payload must not have an incremented OFC (it must have the same OFC as the last frame with a non-empty payload).

After sending a RESET frame, all incoming ACK and NACK frames are ignored until a RESET ACK is received. Additional incoming RESET frames are answered with a matching RESET ACK.

3.7 Waking

When using the UART to wake up another device, the byte 0xFF (ASH WAKEUP) can be sent between frames. This only applies between frames, so 0xFF does not need to be escaped within a frame. Any number of wake bytes can be sent between the end of one frame and the ASH FLAG at the beginning of the next frame.

3.8 Header

Each ASHv3 frame begins with a 4-byte header defined as follows:

1st byte (0)	2nd byte (1)	3rd byte (2)	4th byte (3)
ASH FLAG	HEADER ESCAPE	CONTROL	PAYLOADLENGTH

The ASH FLAG byte indicates the beginning of an ASHv3 frame. It has the value 0x7E.

The HEADER ESCAPE byte contains escape data for the CONTROL byte and the payload length PAYLOAD LENGTH byte. The HEADER ESCAPE byte is encoded as follows:

7	6	5	4	3	2	1	0
A	B	Reserved					

where:

- A = Toggles whether the control byte is escaped. Set to '1' if true; otherwise, set to '0'.
- B = Toggles whether the payload length byte is escaped. Set to '1' if true; otherwise, set to '0'.

The CONTROL byte has the following syntax:

7	6	5	4	3	2	1	0
T		OFC			AFC		

where:

- T = type (see section 3.9 Message Types)
- OFC = Outgoing Frame Counter
- AFC = ACK/NACK Frame Counter

3.9 Message Types

The following table lists the ASHv3 message types.

Table 3.1. ASHv3 Message Types

Message Type	Value
ASH RESET	0
ASH RESET ACK	1
ASH ACK	2
ASH NACK	3

RESET and RESET ACK messages are used for synchronization. ACK messages are used for acknowledgement and NACK messages are used to indicate RX failure.

3.10 Reserved Bytes and Byte Stuffing

ASHv3 reserves certain byte values for special functions. If these values happen to occur within a frame, ASHv3 uses a process known as byte stuffing to replace those bytes so they have non-reserved values. Byte stuffing is performed on the entire ASHv3 frame. The receiver reverses the process to recover the original frame contents.

The following table lists the byte values that are reserved in ASHv3.

Table 3.2. ASHv3 Reserved Byte Values

Value	Special Function
0x7E	Flag Byte: Marks the beginning of a 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.
0xF8	Reserved Byte

To use byte stuffing, first an Escape Byte is sent, 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.

The following list illustrates some examples of 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
- Escape Byte 7D is sent as 7D 5D
- 0xF8 is sent as 7D D8

The byte value 0xFF has a special meaning when it occurs between ASHv3 frames in ASHv3 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.

The following bytes must be escaped when they are included in an ASHv3 packet:

	Bits							
Byte Name	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ASH FLAG	0	1	1	**1**	1	1	1	0
ASH ESC	0	1	1	**1**	1	1	0	1
ASH XON	0	0	0	**1**	0	0	0	1
ASH XOFF	0	0	0	**1**	0	0	1	1
Reserved	1	1	1	**1**	1	0	0	0

Note: All the bytes have bit 4 set to 1. Bit 4 is bold above for illustration.

3.11 CRC Expansion

A CRC-16 (two bytes) is created by CRCing all bytes in the ASHv3 frame, except the CRC itself. The CRC is then expanded to three bytes when it is sent so that none of its bytes needs to be escaped.

The steps to expand the 2-byte CRC, A, into three bytes, B, follow. 'A' refers to the CRC-16 and 'B' refers to the expanded 3-byte version of it. A1[4] refers to A, byte 1, bit 4 and B3[7] refers to B, byte 3, bit 7 (see the following figure for an illustration). Use the same notation for the others.

1. Copy A1[4] to B3[7].
2. Set A1[4] to 0.
3. Copy A1 to B1.
4. Copy A2[4] to B3[6].
5. Set A2[4] to 0.
6. Copy A2 to B2.

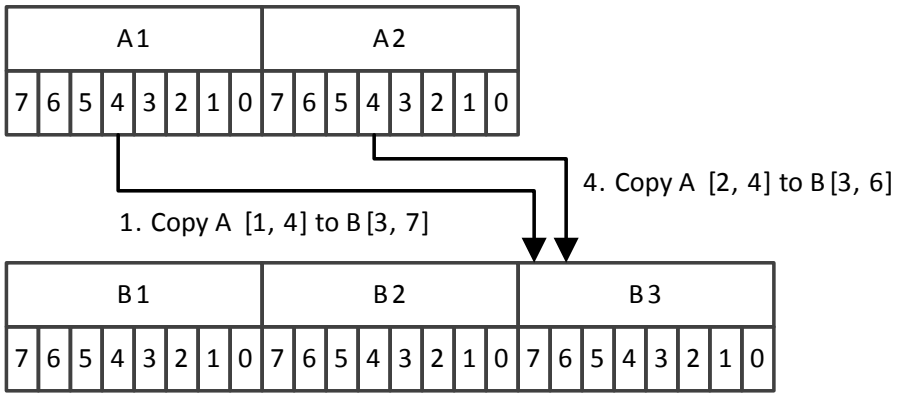


Figure 3.1. Expanding 2-byte CRC (A) into 3 bytes (B)

The expanded CRC is placed at the end of an ASHv3 frame. Note that the three bytes of the expanded CRC are not included in the payload length.

4. Frame Formats

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

4.1 RESET Frame Format

The following table illustrates the format of a RESET frame.

Table 4.1. RESET Full Frame Format

ASH FLAG (0x7E)	Header Escape (0)	Control Byte (0x08) • Type: 0 • OFC: 1 • AFC: 0	Payload Length (0)	CRC 0 (0x69)	CRC 1 (0x86)	CRC 2 (0)

4.2 RESET ACK Frame Format

The following table illustrates the format of a RESET ACK frame with empty payload.

Table 4.2. RESET ACK Full Frame Format

ASH FLAG (0x7E)	Header Escape (0)	Control Byte (0x49) • Type: 1 • OFC: 1 • AFC: 0	Payload Length (0)	CRC 0 (0x47)	CRC 1 (0x6B)	CRC 2 (0xC0)

Note: A RESET ACK with a non-empty payload will have OFC 2.

4.3 ACK Frame Format

The following table illustrates the format of an ACK frame header.

Table 4.3. ACK Frame Header Format

ASH FLAG (0x7E)	Header Escape	Control Byte • Type 2	Payload Length (N)	Data Byte 0	Data Byte 1	,,,,	Data Byte N	CRC 0	CRC 1	CRC 2

4.4 NACK Frame Format

The following table illustrates the format of a NACK frame header. It differs from an ACK frame by only the type in the control byte.

Table 4.4. NACK Frame Header Format

ASH FLAG (0x7E)	Header Escape	Control Byte • Type: 3	Payload Length (N)	Data Byte 0	Data Byte 1	,,,,	Data Byte N	CRC 0	CRC 1	CRC 2

5. Examples

The following sub-sections provide examples of ASHv3 communications.

5.1 Link Synchronization

Both sides send RESET and RESET_ACK packets during link synchronization. The order of the two packet types is not guaranteed.

Node	Type	Outgoing Frame Counter	ACK/NACK Frame Counter
Host	RESET	1	0
NCP	RESET	1	0
Host	RESET_ACK	1	1
NCP	RESET_ACK	1	1

5.2 Link Synchronization with Data

The NCP's RESET ACK has data and an incremented Outgoing Frame Counter. The Host then ACKs it.

Node	Type	Outgoing Frame Counter	ACK/NACK Frame Counter
Host	RESET	1	0
NCP	RESET	1	0
Host	RESET_ACK	1	1
NCP	RESET_ACK	2	1
Host	ACK	1	2

5.3 Sending Data

The Host sends a data packet to the NCP, and the NCP ACKs it.

Node	Type	Outgoing Frame Counter	ACK/NACK Frame Counter	Payload Length
Host	ACK	5	3	40
NCP	ACK	3	5	0

5.4 Sending Data Both Ways

The Host sends a data packet to the NCP and the NCP ACKs it. The NCP's ACK contains data and an incremented Outgoing Frame Counter. The Host sends an empty ACK in response to the NCP's message.

Node	Type	Outgoing Frame Counter	ACK/NACK Frame Counter	Payload Length
Host	ACK	5	3	40
NCP	ACK	4	5	30
Host	ACK	5	4	0

5.5 The NCP NACKs a Frame

The NCP's ACK/NACK Frame Counter has the value of the last understood Outgoing Frame Counter from the Host, which is 2. The host then retries the message.

Steps:

1. The Host sends a data packet.
2. The NCP NACKs the data packet from step 1.
3. The Host resends the data packet from step 1 in response to the NACK.
4. The NCP ACKs the data packet from step 3.

Step	Node	Type	Outgoing Frame Counter	ACK/NACK Frame Counter	Payload Length
1	Host	ACK	3	5	40
2	NCP	NACK	5	2	0
3	Host	ACK	3	5	40
4	NCP	ACK	5	3	0

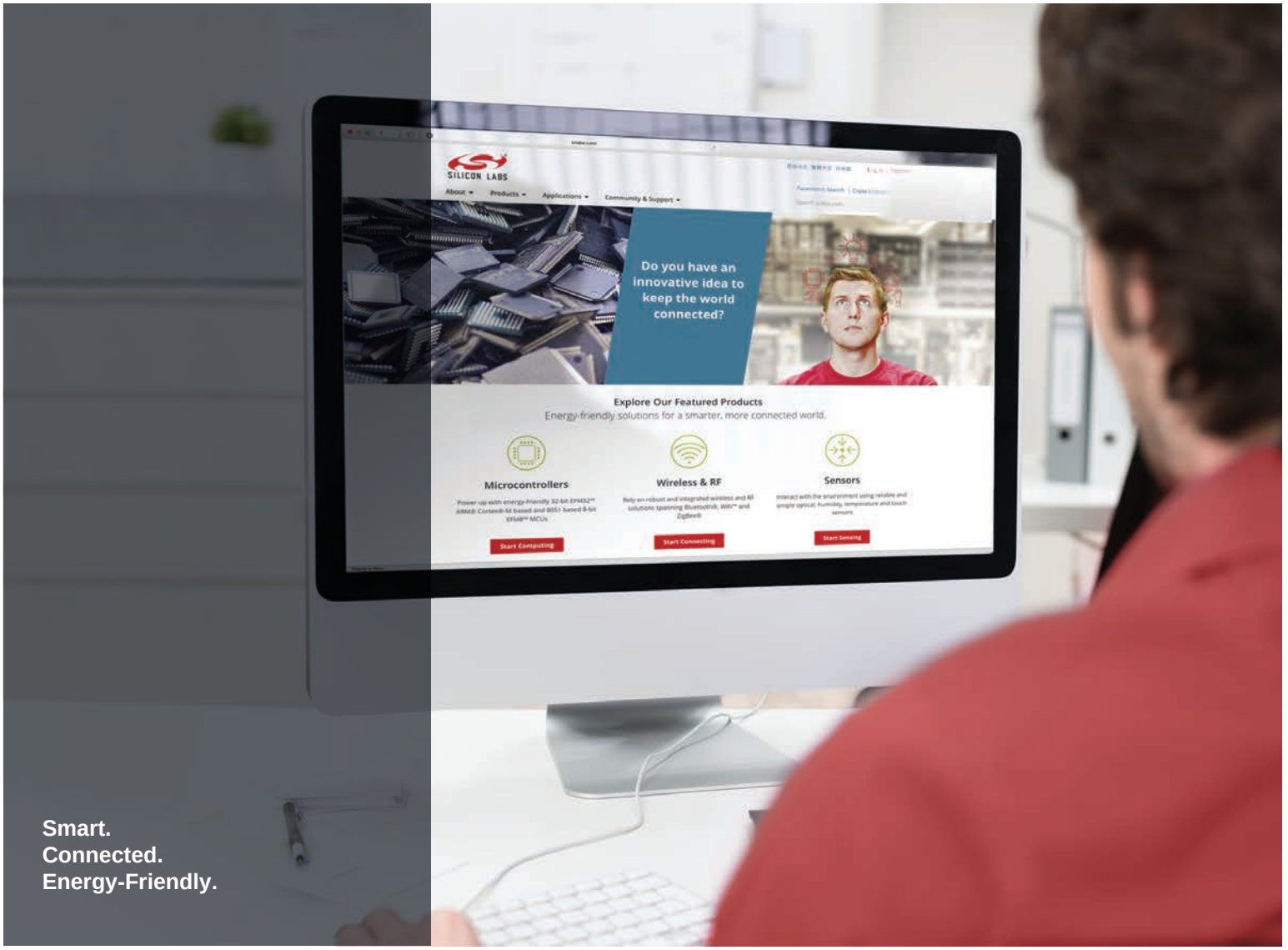
5.6 The NCP NACKs a Frame and Sends Data

The NCP NACKs a frame from the Host, and the NACK contains a payload.

Steps:

1. The Host sends a data packet.
2. The NCP NACKs the data packet from step 1. The NACK contains data.
3. The Host resends the data packet from step 1 in response to the NACK.
4. The Host ACKs the data packet from step 2.
5. The NCP ACKs the data packet from step 3.

Step	Node	Type	Outgoing Frame Counter	ACK/NACK Frame Counter	Payload Length
1	Host	ACK	3	5	40
2	NCP	NACK	6	2	30
3	Host	ACK	3	5	40
4	Host	ACK	3	6	0
5	NCP	ACK	6	3	0



Products
www.silabs.com/products

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

SILICON LABS

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

http://www.silabs.com