Firmware Download Protocol

AirPrime HL75xx and HL854xx

SIERRA WIRELESS®

4116488
2.0
December 09, 2014

Important Notice

Due to the nature of wireless communications, transmission and reception of data can never be guaranteed. Data may be delayed, corrupted (i.e., have errors) or be totally lost. Although significant delays or losses of data are rare when wireless devices such as the Sierra Wireless modem are used in a normal manner with a well-constructed network, the Sierra Wireless modem should not be used in situations where failure to transmit or receive data could result in damage of any kind to the user or any other party, including but not limited to personal injury, death, or loss of property. Sierra Wireless accepts no responsibility for damages of any kind resulting from delays or errors in data transmitted or received using the Sierra Wireless modem, or for failure of the Sierra Wireless modem to transmit or receive such data.

Safety and Hazards

Do not operate the Sierra Wireless modem in areas where cellular modems are not advised without proper device certifications. These areas include environments where cellular radio can interfere such as explosive atmospheres, medical equipment, or any other equipment which may be susceptible to any form of radio interference. The Sierra Wireless modem can transmit signals that could interfere with this equipment. Do not operate the Sierra Wireless modem in any aircraft, whether the aircraft is on the ground or in flight. In aircraft, the Sierra Wireless modem **MUST BE POWERED OFF**. When operating, the Sierra Wireless modem can transmit signals that could interfere with various onboard systems.

Note: *Some airlines may permit the use of cellular phones while the aircraft is on the ground and the door is open. Sierra Wireless modems may be used at this time.*

The driver or operator of any vehicle should not operate the Sierra Wireless modem while in control of a vehicle. Doing so will detract from the driver or operator's control and operation of that vehicle. In some states and provinces, operating such communications devices while in control of a vehicle is an offence.

Limitations of Liability

This manual is provided "as is". Sierra Wireless makes no warranties of any kind, either expressed or implied, including any implied warranties of merchantability, fitness for a particular purpose, or noninfringement. The recipient of the manual shall endorse all risks arising from its use.

The information in this manual is subject to change without notice and does not represent a commitment on the part of Sierra Wireless. SIERRA WIRELESS AND ITS AFFILIATES SPECIFICALLY DISCLAIM LIABILITY FOR ANY AND ALL DIRECT, INDIRECT, SPECIAL, GENERAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES INCLUDING, BUT NOT LIMITED TO, LOSS OF PROFITS OR REVENUE OR ANTICIPATED PROFITS OR REVENUE ARISING OUT OF THE USE OR INABILITY TO USE ANY SIERRA WIRELESS PRODUCT, EVEN IF SIERRA WIRELESS AND/OR ITS AFFILIATES HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES OR THEY ARE FORESEEABLE OR FOR CLAIMS BY ANY THIRD PARTY.

Notwithstanding the foregoing, in no event shall Sierra Wireless and/or its affiliates aggregate liability arising under or in connection with the Sierra Wireless product, regardless of the number of events, occurrences, or claims giving rise to liability, be in excess of the price paid by the purchaser for the Sierra Wireless product.

Customer understands that Sierra Wireless is not providing cellular or GPS (including A-GPS) services. These services are provided by a third party and should be purchased directly by the Customer.

SPECIFIC DISCLAIMERS OF LIABILITY: CUSTOMER RECOGNIZES AND ACKNOWLEDGES SIERRA WIRELESS IS NOT RESPONSIBLE FOR AND SHALL NOT BE HELD LIABLE FOR ANY DEFECT OR DEFICIENCY OF ANY KIND OF CELLULAR OR GPS (INCLUDING A-GPS) SERVICES.

Patents

This product may contain technology developed by or for Sierra Wireless Inc.

This product includes technology licensed from QUALCOMM®.

This product is manufactured or sold by Sierra Wireless Inc. or its affiliates under one or more patents licensed from InterDigital Group and MMP Portfolio Licensing.

Copyright

Trademarks

Sierra Wireless®, AirPrime®, AirLink®, AirVantage®, WISMO® and the Sierra Wireless and Open AT logos are registered trademarks of Sierra Wireless, Inc. or one of its subsidiaries.

Watcher® is a registered trademark of NETGEAR, Inc., used under license.

Windows® and Windows Vista® are registered trademarks of Microsoft Corporation.

Macintosh® and Mac OS X® are registered trademarks of Apple Inc., registered in the U.S. and other countries.

QUALCOMM® is a registered trademark of QUALCOMM Incorporated. Used under license.

Other trademarks are the property of their respective owners.

Contact Information

Sales Desk:	Phone:	1-604-232-1488
	Hours:	8:00 AM to 5:00 PM Pacific Time
	Contact:	http://www.sierrawireless.com/sales
Post:	Sierra Wireless 13811 Wireless Way Richmond, BC Canada V6V 3A4	
Technical Support:	support@sierrawireless.com	
RMA Support:	repairs@sierrawireless.com	
Fax:	1-604-231-1109	
Web:	http://www.sierrawireless.com/	

Consult our website for up-to-date product descriptions, documentation, application notes, firmware upgrades, troubleshooting tips, and press releases: www.sierrawireless.com

Document History

Version	Date	Updates
1.0	October 23, 2014	Creation
2.0	December 09, 2014	Added section 7 FLS File Decoding Mechanism
		Updated: • 3 Synchronization Process • 6.2.1 Download Protocol Configuration • 6.2.3 Hardware Information • 6.3 Flash Information • 6.4 Security Information • 6.8.2 USB

Contents

List of Figures

1. Overview

1.1. Purpose

This document describes the protocol used to download firmware into the flash of an AirPrime HL75xx or HL854xx embedded module from an external processor (either a PC or a microprocessor).

1.2. Glossary

Term	Definition
Boot ROM	Built-in software executed after hardware reset or power up
EBL	External Boot Loader
Flash	High-speed writable/readable/erasable non-volatile memory component
PSI	Primary Signed Image
RAM	Random Access Memory
ROM	Read Only Memory

2. Firmware Download Overview

Firmware download is initiated when the firmware download tool from the host is run. Once this is run, the host will initiate the synchronization phase by sending "**AT**" to the modem.

At power up, or after reset, the modem will wait for 100 ms to receive the sync character "**AT**" from the host. If it receives "**AT**" in time, the synchronization phase will continue. After which the host will send the PSI to the modem. Once the PSI has been downloaded and executed in RAM, the host will send the EBL to the modem. And once the EBL has been downloaded and executed in flash, the firmware image data will be sent to the flash via EBL's protocol.

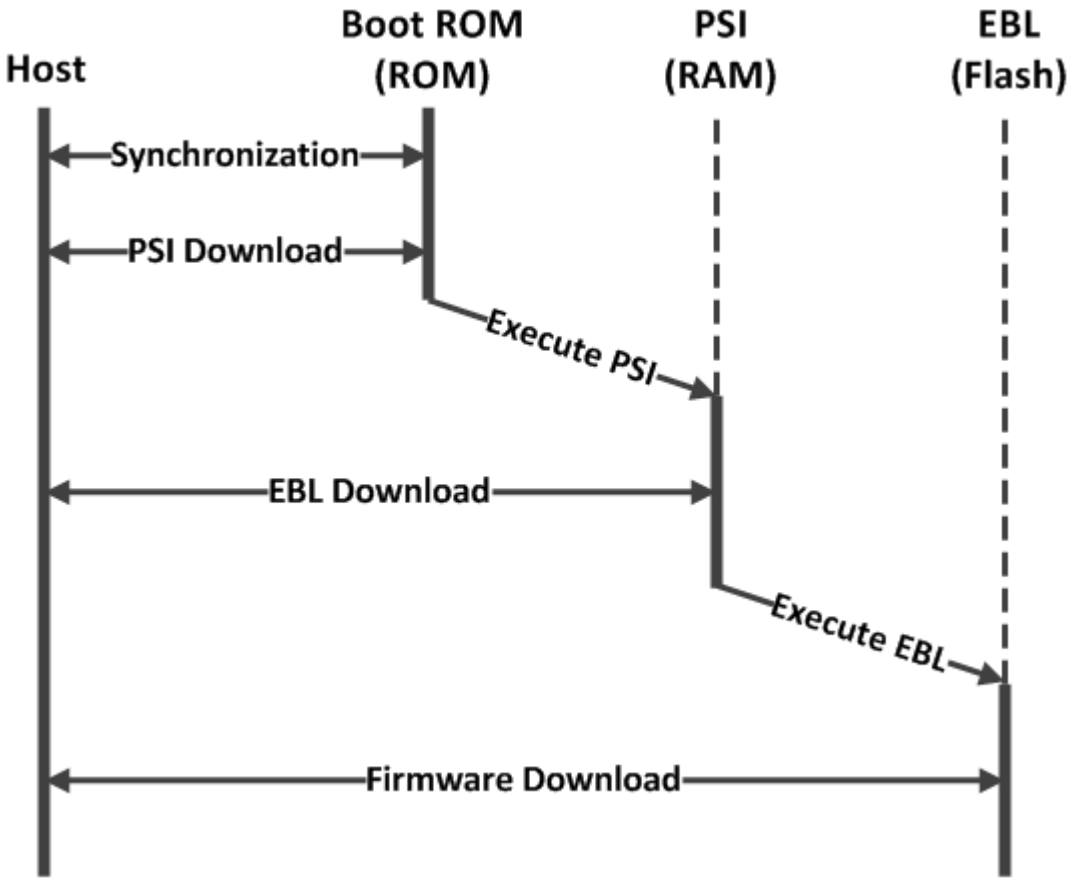


Figure 1. Firmware Download Overview

3. Synchronization Process

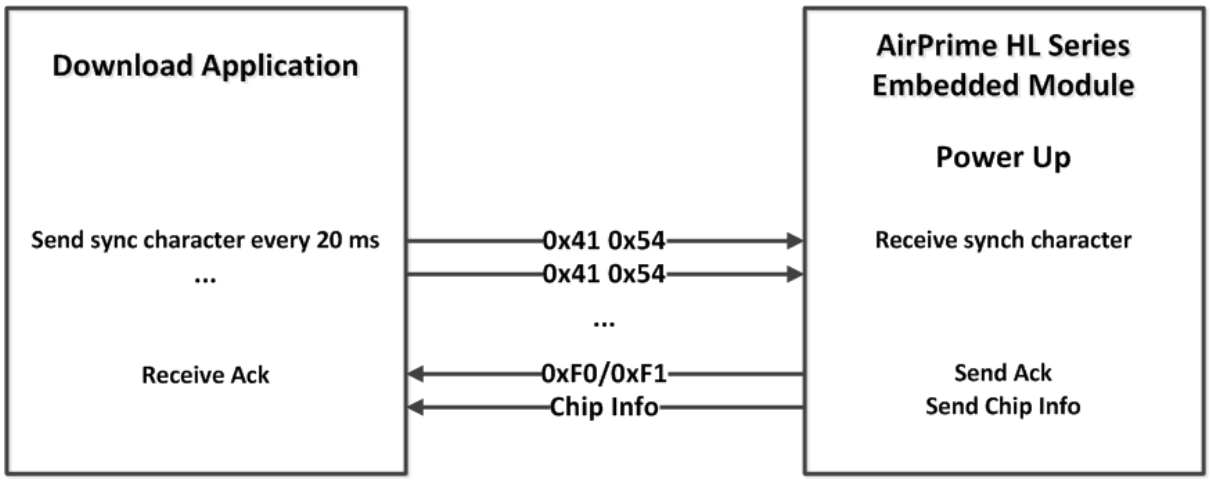


Figure 2. Synchronization Process

1. Host sends the sync character "**AT**" (**0x41 0x54**) to the modem every 20 ms.
2. Modem returns first acknowledgement frame (**0xF0** or **0xF1**) if two sync characters are received.
3. Modem sends chip information (27 bytes for HL75xx, 23 bytes for HL854xx), which includes chip ID, chip version, etc. Additional information regarding chip information parameters are listed below.

   For the HL75xx,

   <Size> (byte 0) = **0x1C**

   <Chip ID> (byte 1) = **0x54**

   <Boot core version> (byte 2) = **0x35**

   For the HL854xx,

   <Size> (byte 0) = **0x16**

   <Chip ID> (byte 1) = **0x51**

   <Boot core version> (byte 2) = **0x94**

4. PSI Download Process

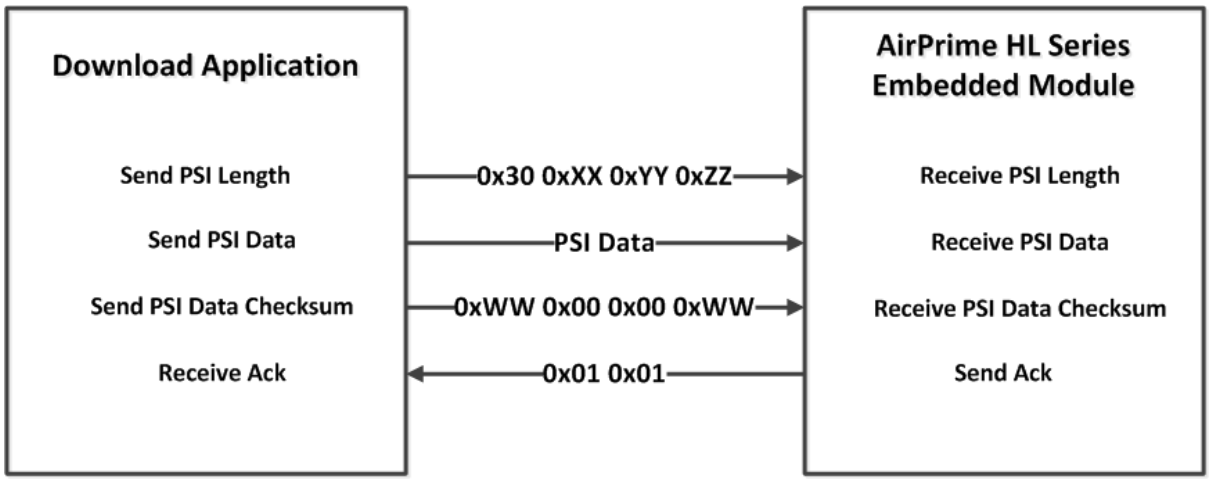


Figure 3. PSI Download Process

1. Host sends 4 bytes (described below) to the modem indicating the length (in bytes) of PSI.

   Transmit frame = **`0x30`** (1 byte) **`length`** (3 bytes) = total of 4 bytes

   E.g. If the length is **`0x18 0xA2 0x00`**, it means length = 41496 bytes (0x00A218 = 41496 in decimal).
2. Host sends (raw) PSI data of N bytes (where N = length sent in the previous command).
3. Host sends XOR checksum of PSI data.

   Transmit frame = **`checksum`** (1 byte) **`0x00 0x00 checksum`** (1 byte) = total of 4 bytes, duplicate checksum in the frame.

   E.g. **`0x60 0x00 0x00 0x60`** for checksum = 0x60
4. Modem returns 2 bytes as acknowledgment.

   Transmit frame = **`0x01 0x01`** (if the first byte is **`0x01`**, it means that the download was successful; but if the first byte is **`0xFF`**, it means that the download was unsuccessful.

5. EBL Download Process

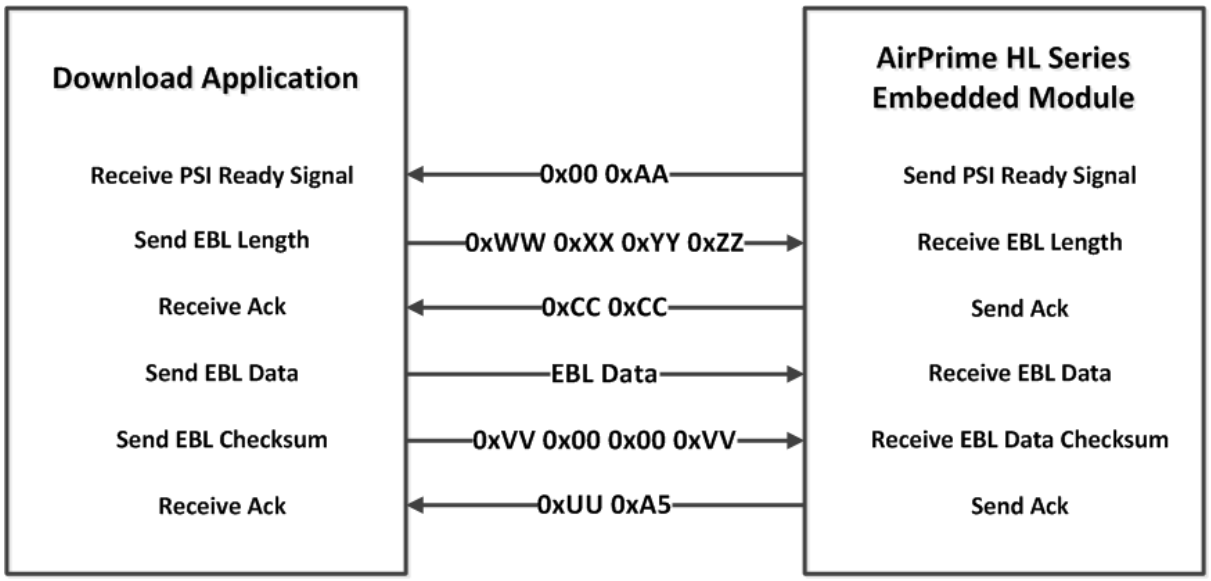


Figure 4. EBL Download Process

1. Modem sends PSI Ready signal.

   Transmit frame = **`0x00 0xAA`**

2. Host sends EBL length (in bytes).

   E.g. **`0x34 0xCE 0x03 0x00`** for 249396 (= 0x0003CE34) bytes

3. Modem confirms EBL length by sending an acknowledgement.

   Transmit frame = **`0xCC 0xCC`**

4. Host sends (raw) EBL data of N bytes (where N = EBL length).
5. Host sends XOR checksum of EBL data.

   Transmit frame = **`checksum`** (1 byte) **`0x00 0x00 checksum`** (1 byte) = total of 4 bytes, duplicate checksum in the frame.

6. Modem returns acknowledgement.

   Transmit frame = **`Chip ID`** (1 byte) **`result`** (1 byte); where **`Chip ID = 0x54`** for HL75xx, and **`Chip ID = 0x51`** for HL854xx, and **`result = 0xA5`** if the process was successful.

6. Firmware Download Process

6.1. Firmware Download Protocol Frame Structure

6.1.1. UART Protocol

SOT	TYPE	LENGTH	PAYLOAD	CRC	EOT

SOT (2 bytes) = **0x02 0x00**

TYPE (2 bytes) = **0xXX 0xYY** (bytes are swapped; actually represents **0xYYXX)**

LENGTH (2 bytes) = **0xXX 0xYY** (bytes are swapped; actually represents **0xYYXX**. Must be an even number, with maximum value = 0x800.)

PAYLOAD (N bytes where N = **LENGTH**) = Payload data

CRC (2 bytes) = **0xXX 0xYY** (bytes are swapped; actually represents **0xYYXX**. Sum-of-byte, SOT and EOT are **not** included in the checksum calculation.)

EOT (2 bytes) = **0x03 0x00**

6.1.2. USB Protocol

CRC	TYPE	LENGTH	PAYLOAD

CRC (2 bytes) = **0xXX 0xYY** (bytes are swapped; actually represents **0xYYXX**. This is the sum-of-byte of the **TYPE**, **LENGTH** and **PAYLOAD** data bytes.)

TYPE (2 bytes) = **0xXX 0xYY** (bytes are swapped; actually represents **0xYYXX**)

LENGTH (4 bytes) = **0xWW 0xXX 0xYY 0xZZ** (bytes are swapped; actually represents **0xZZYYXXWW**)

PAYLOAD (N bytes where N = **LENGTH**) = Payload data

6.1.3. Payload Data

The length of the payload data depends on the **LENGTH** defined in the header.

For example, if using UART protocol and **LENGTH = 0x04 0x00**, this means that the actual length is 0x04 and that the payload should be 4 bytes long.

6.2. Firmware Download

6.2.1. Download Protocol Configuration

Before downloading the firmware, the download protocol configuration needs to be set. The modem and the host exchange EBL versions and list of extended capabilities to use.

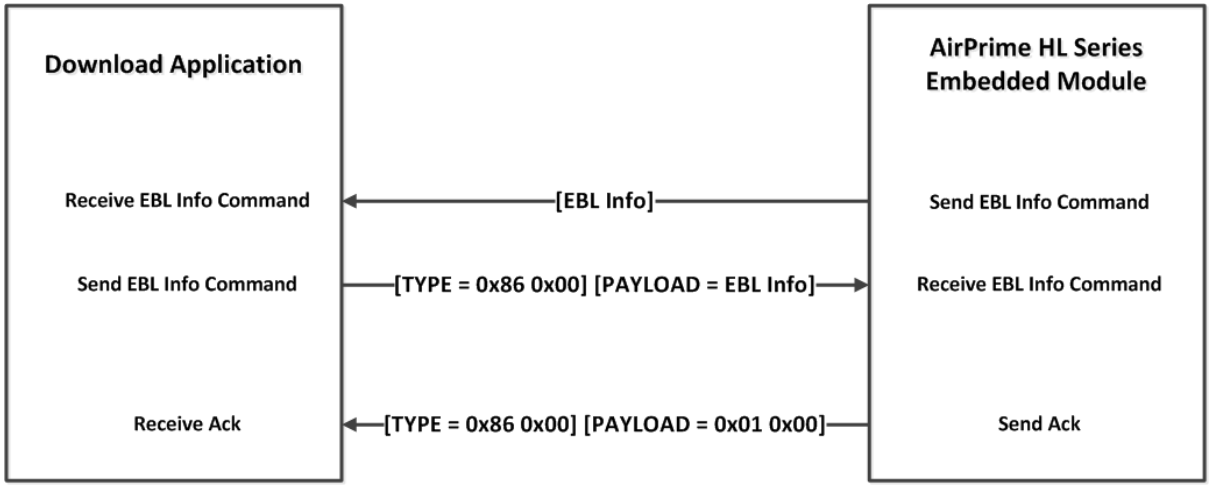


Figure 5. Download Protocol Configuration

1. When EBL starts running, the modem sends the EBL version string and the list of extended capabilities supported by the EBL to the host (76 bytes raw data).
2. Host sends back the EBL version string and the list of extended capabilities.

   Header: **`TYPE = 0x86`**, **`LENGTH = 0x48`**

   Payload: **`EBL version and the list of extended capabilities`** (same as the raw data in step 1 above).

   Additional information regarding EBL version and extended capabilities are listed below.

   For the HL75xx,

   `<EBL version>` (byte 12 to byte 43) =

   ```
   58 4D 4D 37                                     | XMM7
   31 36 30 5F 31 34 33 34 2E 35 30 30 5F 4D 31 53 | 160_1434.500_M1S
   31 00 00 00 00 00 00 00 00 00 00 00             | 1
   ```

   `<comm_format>` (byte 46) = **`0x00`** for USIF, **`0x01`** for USB

   For the HL854xx,

   `<EBL version>` (byte 12 to byte 43) =

   ```
   58 4D 4D 36                                     | XMM6
   32 36 30 5F 31 33 32 37 2E 31 30 30 5F 4D 31 53 | 260_1327.100_M1S
   31 00 00 00 00 00 00 00 00 00 00 00             | 1
   ```

   `<comm_format>` (byte 46) = **`0x00`** for USIF, **`0x01`** for USB

3. Modem sends acknowledgement frame.

   Header: **`TYPE = 0x86`**, **`LENGTH = 0x02`**

   Payload: **`0x01 0x00`**

6.2.2. Baud Rate

The command is sent by the host to change the baud rate of the modem.

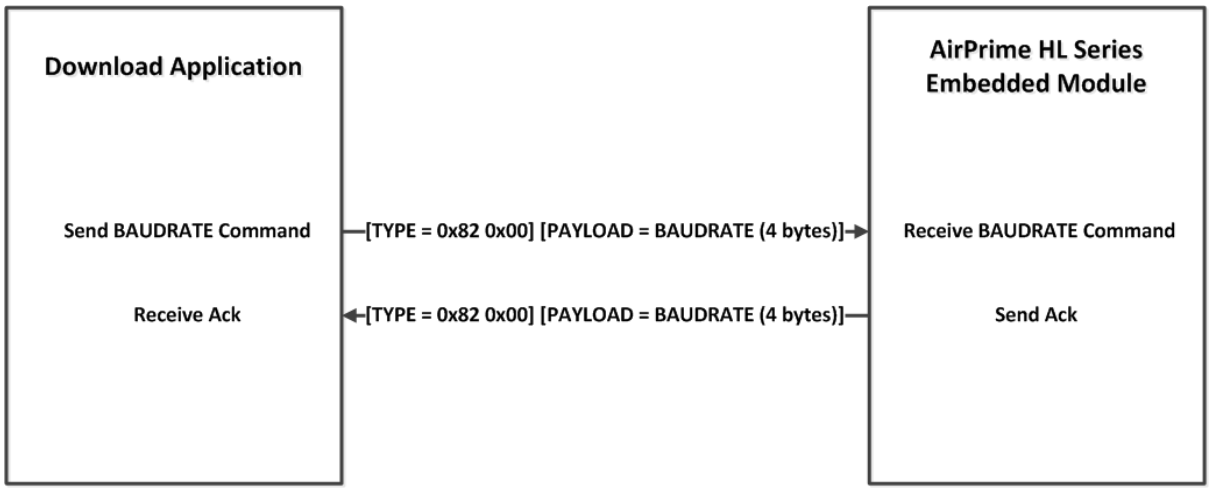


Figure 6. Baud Rate

1. Host sends BAUDRATE command.

   Header: **`TYPE = 0x82`**, **`LENGTH = 0x04`**

   Payload: **`baud rate`** (4 bytes)

   E.g. **`0x00 0x10 0x0E 0x00`** for 921600 (= 0x000E1000)

2. Modem sends acknowledgement frame with same baud rate.

   Header: **`TYPE = 0x82`**, **`LENGTH = 0x04`**

   Payload: **`baud rate`** (same as the baud rate in step 1 above)

3. Host should wait for 20 ms for other commands in order to ensure that baud rate has changed.

6.2.3. Hardware Information

The command is sent by the host to transfer hardware information to the modem.

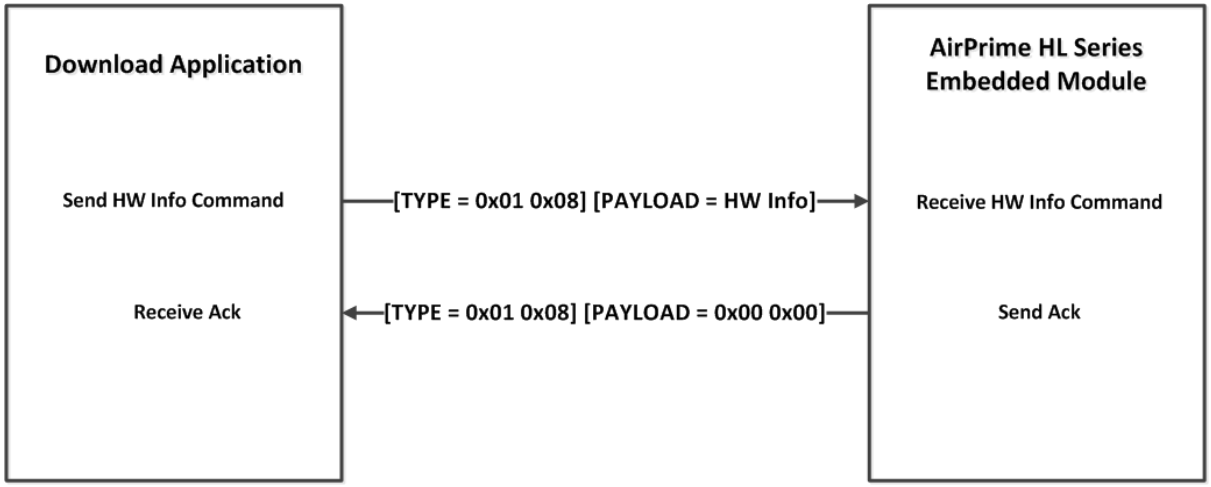


Figure 7. Hardware Information

1. Host sends HW Info command.

   Header: **`TYPE = 0x801`**, **`LENGTH = 0xAC`**

   Payload: **`Hardware information`** (platform, boot speed, etc.)

   For the HL75xx, **`Hardware information`** details are as follows:

   `<Platform ID>` (byte 0 to byte 3) = **`0x14 0x00 0x00 0x00`**

   `<Boot speed>` (byte 8 to byte 11) = **`0x00 0xC2 0x01 0x00`**
   (this value represents **`0x1C200`**, which is 115200 in decimal)

   For the HL854xx, **`Hardware information`** details are as follows:

   `<Platform ID>` (byte 0 to byte 3) = **`0x0E 0x00 0x00 0x00`**

   `<Boot speed>` (byte 8 to byte 11) = **`0x00 0xC2 0x01 0x00`**
   (this value represents **`0x1C200`**, which is 115200 in decimal)

2. Modem sends acknowledgement frame.

   Header: **`TYPE = 0x801`**, **`LENGTH = 0x02`**

   Payload: **`0x00 0x00`**

6.3. Flash Information

The command is sent by the host to request Flash information from the modem.

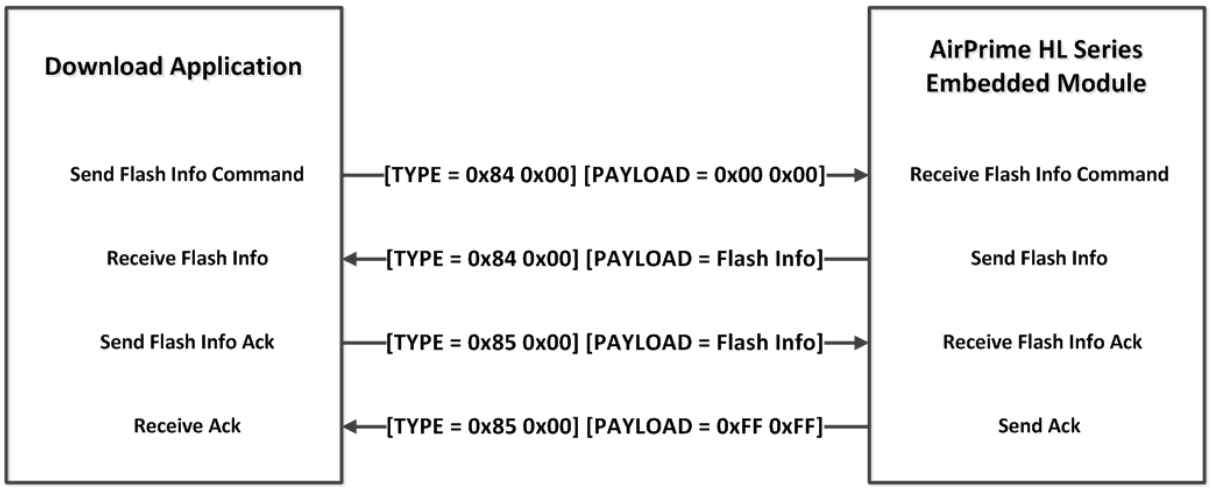


Figure 8. Flash Information

1. Host sends Flash Info command.

   Header: **`TYPE = 0x84`**, **`LENGTH = 0x02`**

   Payload: **`0x00 0x00`**
2. Modem returns flash information.

   Header: **`TYPE = 0x84`**, **`LENGTH = 0x100`**

   Payload: **`Flash information`** (manufacturer, etc.)

   For the HL75xx, **`<manufacturer>`** (byte 4 to byte 7) = **`0x2C 0x00 0xB1 0x00`**

   For the HL854xx, **`<manufacturer>`** (byte 4 to byte 7) = **`0xC8 0x00 0xA1 0x00`**
3. Host sends Flash Info acknowledgement.

   Header: **`TYPE = 0x85`**, **`LENGTH = 0x100`**

   Payload: **`Flash information`** (same as the flash information in step 2 above)
4. Modem sends acknowledgement frame.

   Header: **`TYPE = 0x85`**, **`LENGTH = 0x02`**

   Payload: **`0xFF 0xFF`**

6.4. Security Information

The command is sent by the host to verify the firmware to download. The payload contains the load map and the signature of the firmware image to download.

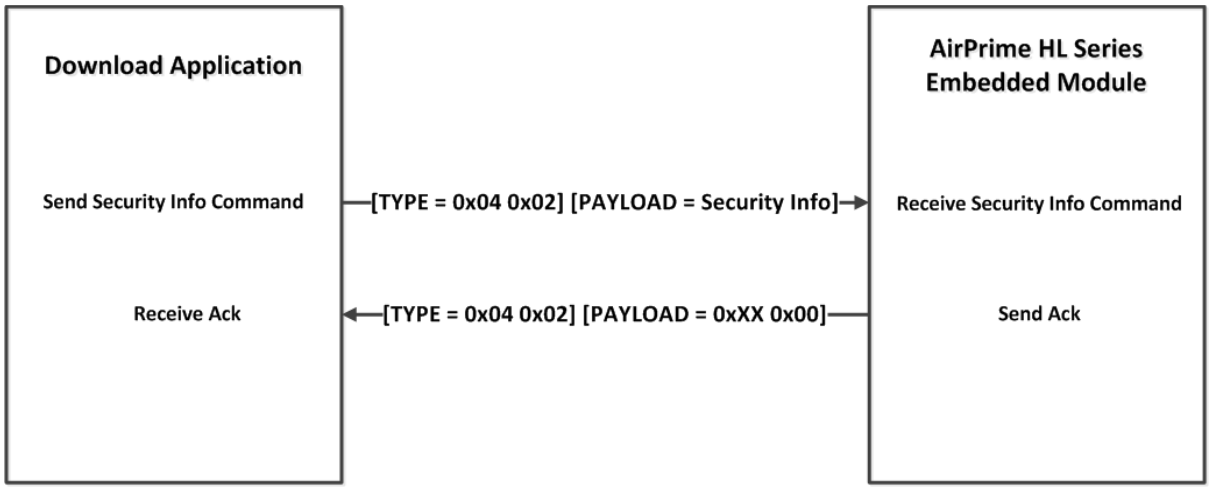


Figure 9. Security Information

1. Host sends Security Info command.

   Header: **`TYPE = 0x204`**, **`LENGTH = 0x800`**

   Payload: **`Load map and signature`** (2048 bytes)

   Additional information regarding load map and signature are listed below.

   `<Security data, signature and flags>` (byte 0 to byte 1919) = **`0xXX, 0xYY,`** …

   `<LoadMap>` (byte 1920 to byte 2047) is 128 bytes long and supports up to 8 load map sections (16 bytes each). The structure of each load map section (4x4 bytes) is:

```
typedef struct {
    U32 StartAddr;
    U32 TotalLength;
    U32 UsedLength;
    U32 ImageFlags;
}RegionStructType;
```

   The start address and end address used in sections 6.5 and 6.7 refer to the information specified here; start address = **`StartAddr`** in the load map and end address = **`StartAddr`** + **`UsedLength`** – 2.

2. Modem sends acknowledgement frame.

   Header: **`TYPE = 0x204`**, **`LENGTH = 0x02`**

   Payload: **`0xXX 0x00`** (where **`0xXX = 0x00`** means normal download, and **`0xXX = 0x01`** means the firmware image to be downloaded can be considered as identical to the installed firmware image; download is not needed).

6.5. Flash Erase

The command is sent by the host to specify the start address and the end address of the flash area to erase. After this command is acknowledged, erasing will be conducted in the background.

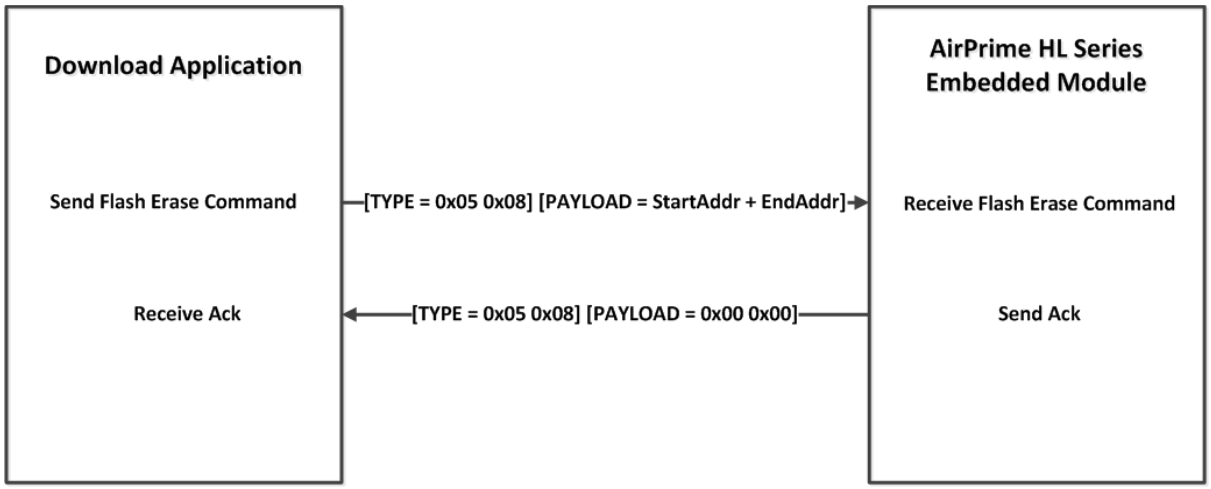


Figure 10. Flash Erase

1. Host sends Flash Erase command.

   Header: **`TYPE = 0x805`**, **`LENGTH = 0x08`**

   Payload: **`start address`** (4 bytes) and **`end address`** (4 bytes)

   E.g. **`0x00 0x00 0x0A 0x00 0xFE 0xFF 0x0B 0x00`** (where **`start address = 0x000A0000`**, and **`end address = 0x000BFFFE`**)

2. Modem sends acknowledgement frame.

   Header: **`TYPE = 0x805`**, **`LENGTH = 0x02`**

   Payload: **`0x00 0x00`**

6.6. Flash Erase Check

The command is sent by the host to check if the erase procedure has finished.

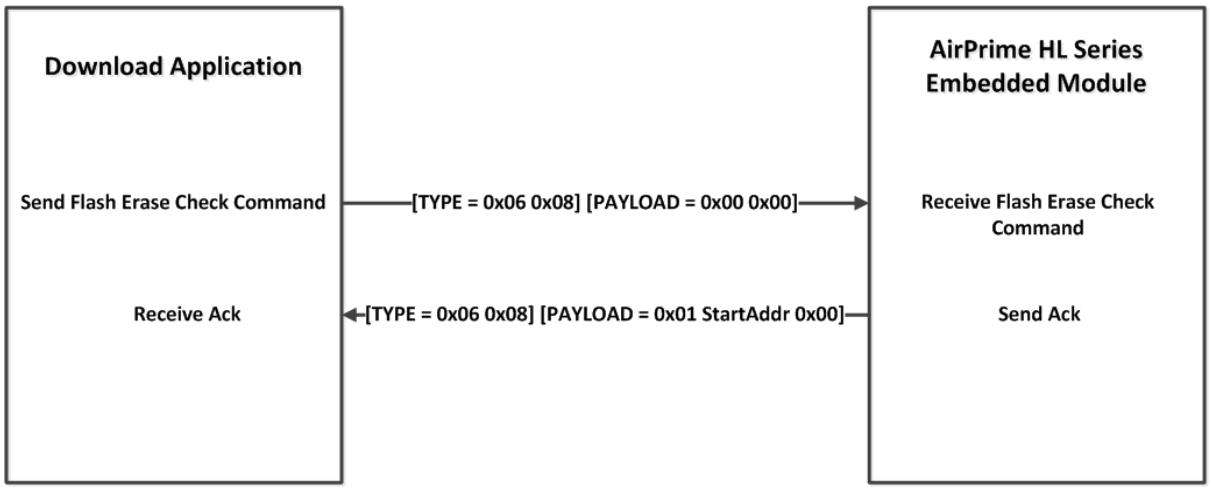


Figure 11. Flash Erase Check

1. Host sends Flash Erase Check command.

   Header: **`TYPE = 0x806`**, **`LENGTH = 0x02`**

   Payload: **`0x00 0x00`**

2. Modem sends acknowledgement frame.

   Header: **`TYPE = 0x806`**, **`LENGTH = 0x06`**

   Payload: **`state`** (1 byte), **`start address`** 4 bytes and **`0x00`** (where **`state = 0x01`** if erasing is finished, else erasing is not yet finished; and **`start address`** should be the same start address sent in step 1 of section 6.5 Flash Erase)

6.7. Flash Write Address

The command is sent by the host to set the address pointer for writing to the flash.

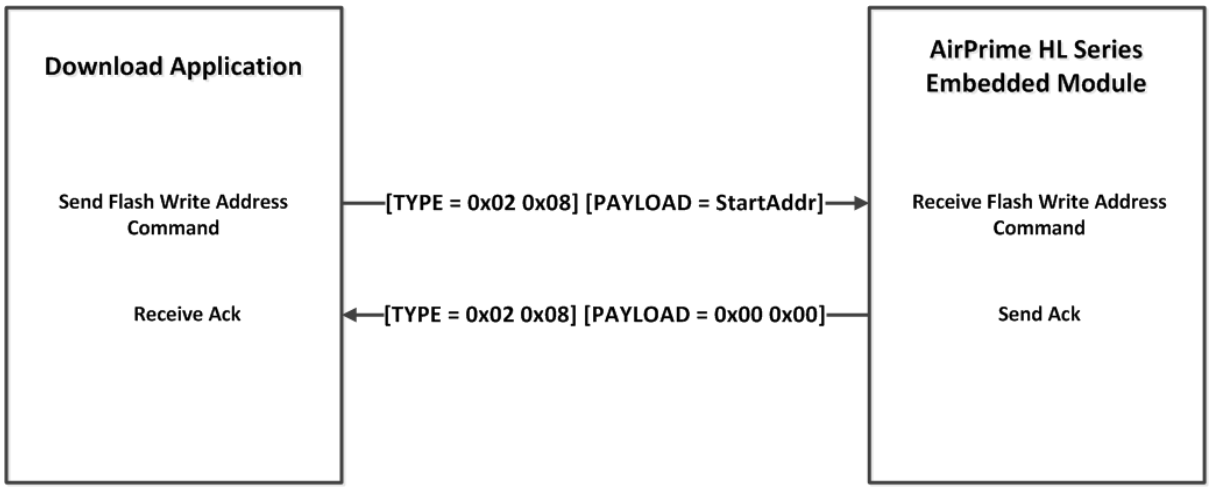


Figure 12. Flash Write Address

1. Host sends Flash Write Address command.

   Header: **`TYPE = 0x802`**, **`LENGTH = 0x04`**

   Payload: **`start address`**

   E.g. **`0x00 0x00 0x0A 0x00`** (where **`start address = 0x000A0000`**)

2. Modem sends acknowledgement frame.

   Header: **`TYPE = 0x802`**, **`LENGTH = 0x02`**

   Payload: **`0x00 0x00`**

6.8. Flash Write

Although USB and UART adopt different methods to send data streams, commands are sent by the host to write a chunk of data to the flash memory for both protocols.

Before the command is sent, make sure that the flash erase process of the target area has finished.

6.8.1. UART

The host truncates the firmware image into blocks and sends the blocks to the modem one by one. Each block has a maximum of 2048 bytes. The firmware image is fully downloaded when all blocks have been sent to the modem.

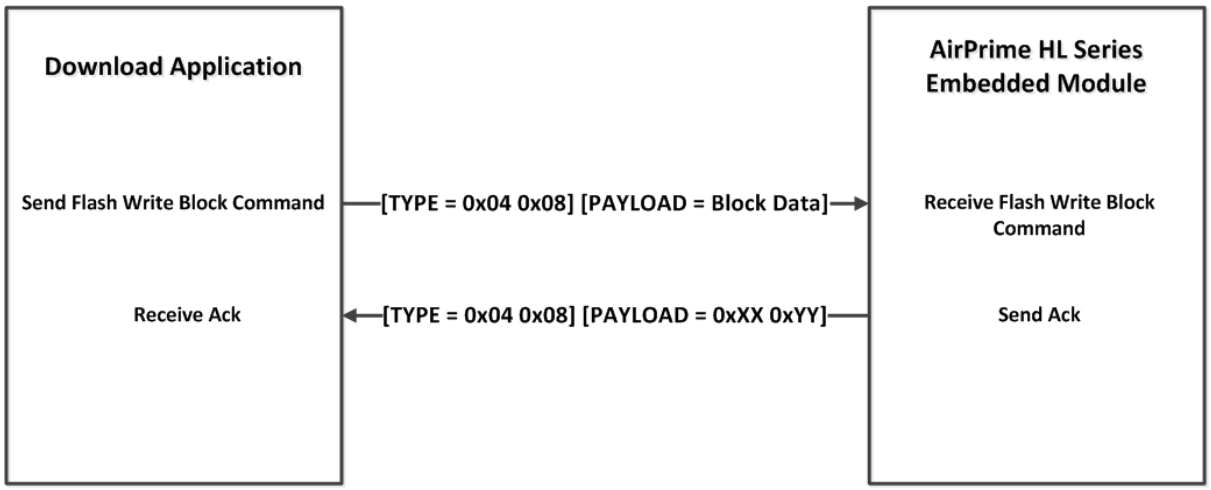


Figure 13. Flash Write – UART

1. Host sends Flash Write Block command.

   Header: **`TYPE = 0x804`**, **`LENGTH = 0x800`** (note that the length can be less than 0x800 if needed, i.e. the payload is less than 2048 bytes in the last block)

   Payload: **`Block data of firmware image`**

2. Modem sends acknowledgement frame for every block.

   Header: **`TYPE = 0x804`**, **`LENGTH = 0x02`**

   Payload: **`CRC for the received block`**

3. Repeat steps 1 and 2 above until all data blocks are sent.

6.8.2. USB

In this mode, larger chunks of data can be sent using the USB layer's handshake mechanism. The first package will have a payload of length 4 bytes, indicating the length of every chunk of data to be sent. The raw data will be sent to the modem immediately after this. The upper length limit for each chunk of data is 0x20000 (131073 bytes). Repeat this process if massive data is required to be sent chunk by chunk.

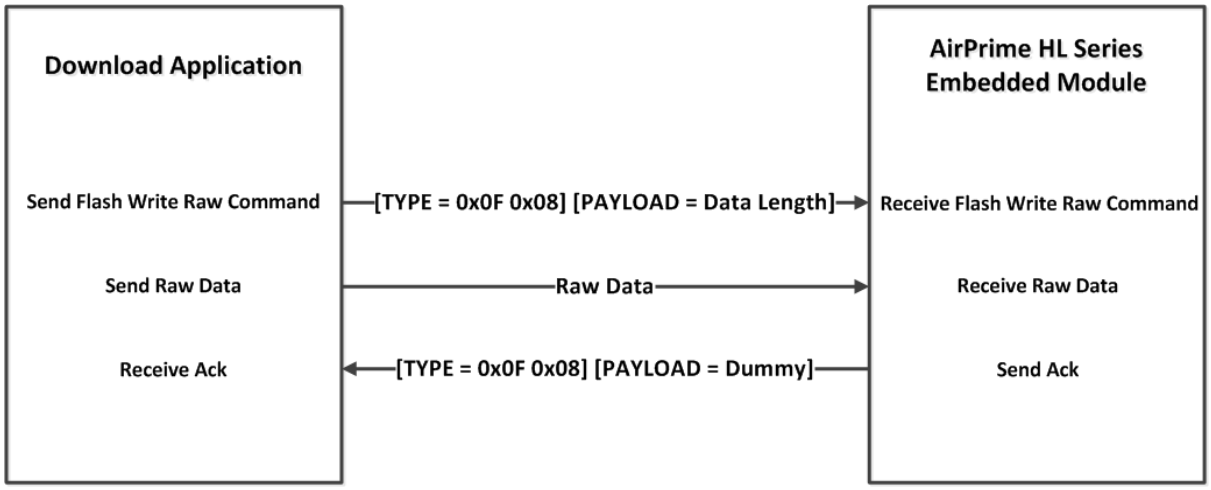


Figure 14. Flash Write – USB

1. Host sends Flash Write Raw command.

   Header: **`TYPE = 0x80F`**, **`LENGTH = 0x04`**

   Payload: **`data length`**

   E.g. **`0x00 0x00 0x02 0x00`** = 0x20000 (131072 bytes)
2. Host sends firmware image (raw data of N bytes where N = data length).
3. Modem sends acknowledgement frame.

   Header: **`TYPE = 0x80F`**, **`LENGTH = 0x04`**

   Payload: **`Dummy`** (4 bytes)

6.9. Firmware Checksum

The command is sent to request verification of the firmware image downloaded and check if it has been downloaded successfully.

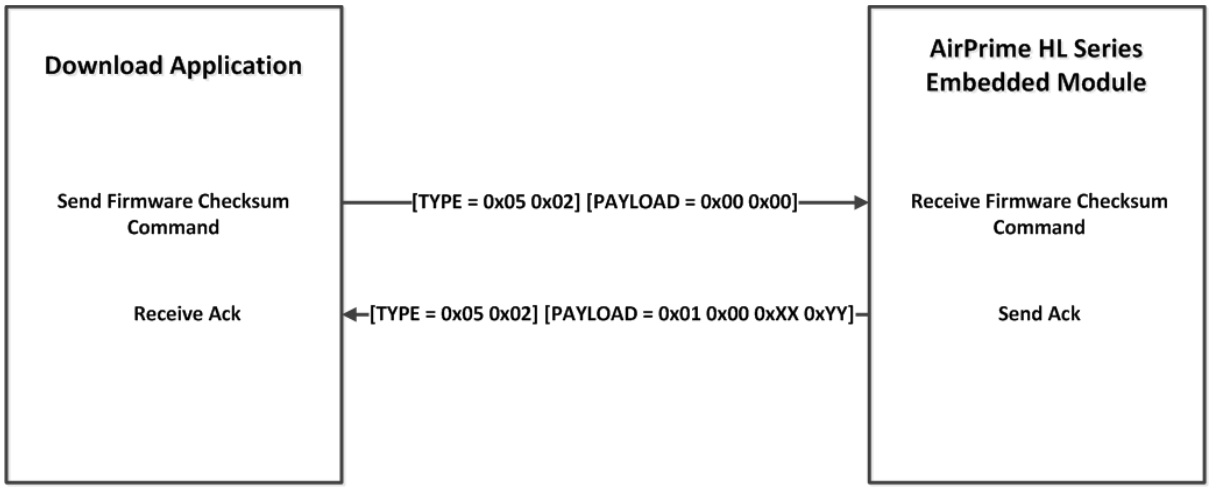


Figure 15. Firmware Checksum

1. Host sends Firmware Checksum command.

   Header: **`TYPE = 0x205`**, **`LENGTH = 0x02`**

   Payload: **`0x00 0x00`**

2. Modem sends acknowledgement frame.

   Header: **`TYPE = 0x205`**, **`LENGTH = 0x04`**

   USB Payload: **`0x01 0x00 0xXX 0xYY`** (where the firmware checksum = 0xYYXX)

   UART Payload: **`0x01 0x00 0x00 0x00`**

6.10. Modem Reset

The command is sent by host to force the target to reset. This will cause the protocol to shut down and the target will start-up in normal mode.

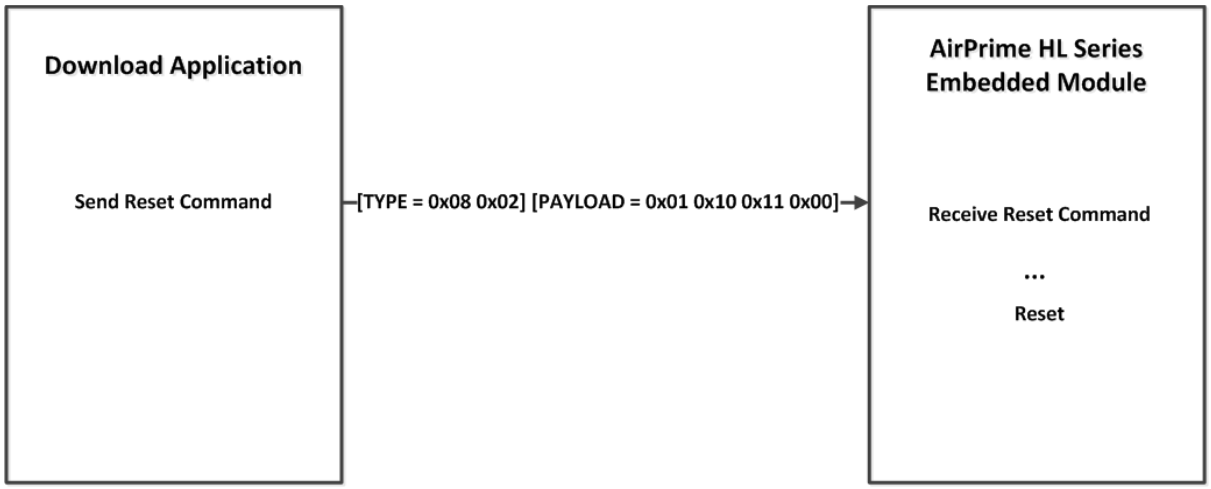


Figure 16. Modem Reset

Host sends Modem Reset command.

Header: **`TYPE = 0x208`**, **`LENGTH = 0x04`**

Payload: **`0x01 0x10 0x11 0x00`**

7. FLS File Decoding Mechanism

Several FLS files could be packed into one FLS file. The current HL854xx firmware release consists of six FLS files. The host needs to download these six files into the HL854xx Embedded Module one by one, according to their load map. The HL75xx firmware on the other hand, consists of a single FLS file from where six FLS files can be extracted from. The same flashing process used for the HL854xx should then also be used for the HL75xx.

The FLS file has a block by block structure where each element block contains a specific type of information and consists of an **element header** and its **element data**.

7.1. Element Header

Each element header takes 12 bytes in the form of 3 unsigned integer variables representing its type, size and UID as follows:

```
typedef struct {
   U32   Type;
   U32   Size;
   U32   UID;
}ElementHeaderStructType;
```

7.1.1. Type

Type indicates the type of information stored in a block. Each type of information (e.g. hardware information, security information) has its unique Type value. Some necessary values are specified below:

- Type = **0x0C** refers to flashing data (the actual data to be downloaded into ROM)
- Type = **0x0D** refers to hardware information (see section 6.2.3)
- Type = **0x0F** refers to security information (see section 6.4)
- Type = **0x10** refers to table of contents (used for the HL75xx to extract the embedded FLS files)
- Type = **0x12** refers to PSI data patch (see section 4)
- Type = **0x13** refers to EBL data patch (see section 5)
- Type = **0x02** refers to the last element block of FLS file

7.1.2. Size

Size represents the total length (in bytes) of the block, including the element header (12 bytes) and data (N bytes).

7.1.3. UID

The UID correspond to the table of contents (see section 7.2.2) used with the HL75xx. Each table of contents entry has one UID.

7.2. Element Data

7.2.1. Download Data (Type=0x0C)

Contained in the element data of Download Data (Type=0x0C) is a sub-block with DownloadData Header and Download Data. Download Data is the exact data to be downloaded. The block structure is as follows:

[Element Header (Type=0x0C)][DownloadData Header][Download Data]

7.2.1.1. DownloadData Header

```
typedef struct
{
     U32 LoadMapIndex;
     U32 CompressionAlgorithm;
     U32 CompressedLength;
     U32 CRC;
     U32 DataLength;
     U8  *Data;
     U32 DataOffset;
}DownloadDataStructType;
```

Where:

<DataLength> (byte 16 to byte 19) indicates the total length (in bytes) of the flashing data to be transmitted. This is identical with the **<UsedLength>** in load map.

<DataOffset> (byte 24 to byte 27) indicates the offset value with respect to the FLS file. (In the case of the HL75xx, this is the offset value of the packed FLS file, i.e. the full FLS file with its six embedded FLS packed inside.)

7.2.1.2. Download Data

The range of Download Data starts from **<DataOffset>**, and is **<DataLength>** bytes long.

7.2.2. Table of Contents (Type=0x10)

In the HL75xx, six FLS files are packed into one FLS file. In order to download the FLS files one by one, the host needs to extract the files according to the Table of Contents. Information regarding Table of Contents parameters are as follows:

<NoOfEntries> (byte 0 to byte 3) indicates how many entries are stored in the table of contents

<DataOffset> (byte 8 to byte 11) indicates the offset value with respect to the FLS file

7.2.2.1. TableOfContent Entry

Starting with **<DataOffset>**, several TableOfContent Entry sub-blocks are stored, with the total number of entries specified by **<NoOfEntries>**. Each TableOfContent Entry sub-block is 144 bytes, and contains the variables listed below.

<UID> (byte 0 to byte 3) matches with the **<UID>** in section 7.1.3. In the case of the HL75xx, there are six TableOfContent Entry sub-blocks, with **<UID>** ranging from 0 to 5, representing the unique ID for each packed FLS file.

<MemoryClass> (byte 4 to byte 7) and can have the following values:

0x01 = PSI

0x02 = SLB

0x04 = CODE

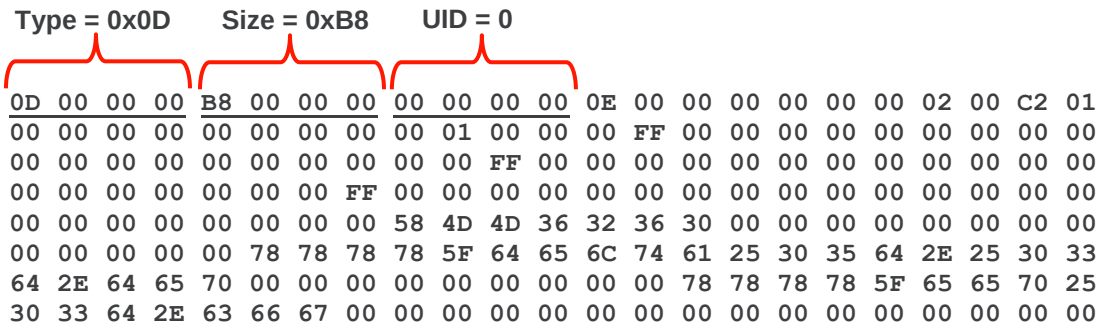
0x05 = CUST

<FileName> (byte 16 to byte 143) stores an ASCII string of the file name of each packed FLS file.

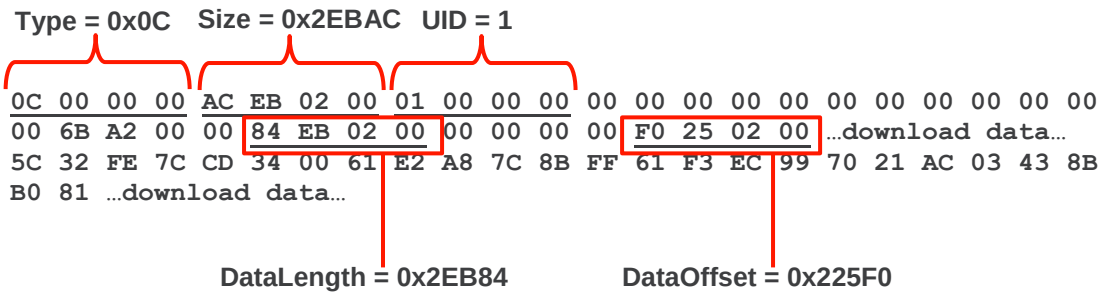
7.3. Examples

7.3.1. Normal Element Block (Hardware Information)

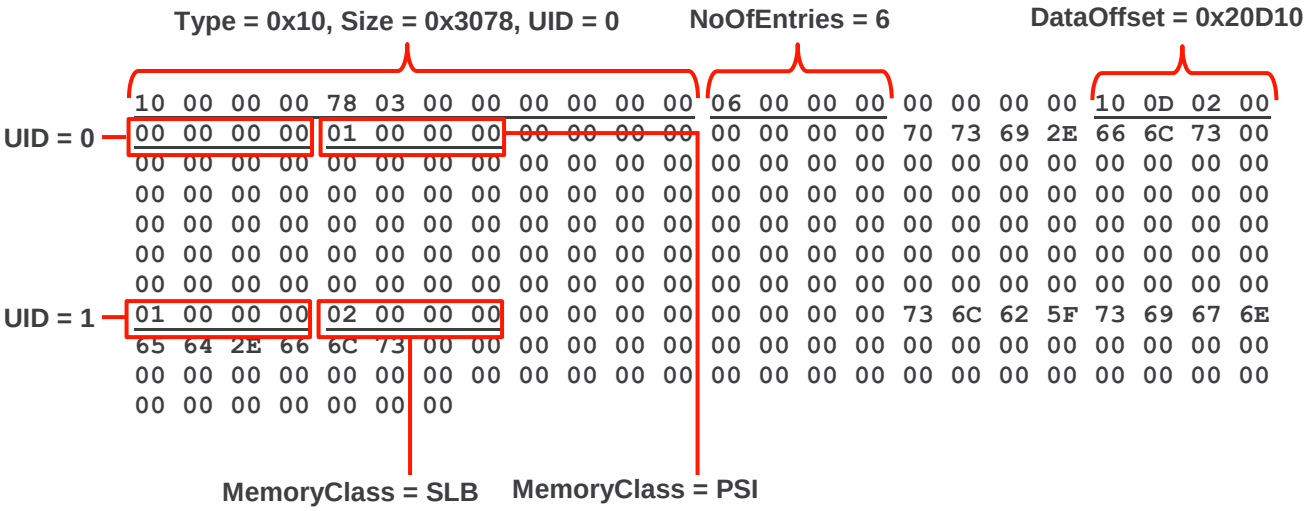
The following is an example of a block for hardware information trimmed from an FLS file. Blocks on PSI, EBL and security information have the same structure.



7.3.2. Download Data



7.3.3. Table of Contents



8. Appendix: HL75xx Download Flow Example

The following shows an actual data flow of an HL75xx firmware download over USB.

Host sends the sync character “AT”

```
41 54 41 54
```

Host receives acknowledgement

```
F1
```

Host receives chip information

```
1C 54 35 05 00 15 00 00 00 E0 10 0C 09 70 20 94 C1 48 E6 EC 2E
92 30 00 20 00 FF
```

Host sends PSI length

```
30 04 80 01
```

Host sends PSI data

```
72 35 00 EA 6C 69 48 55 02 A0 FE FF F4 01 9A 05
00 80 01 00 10 03 00 00 00 03 00 00 04 03 00 00
FF FF FF FF FF FF FF FF 00 7F 01 00 FF FF FF FF
00 00 08 00 FF FF F7 FF 00 03 08 00 43 4A 4B 54
```

…..suppressed…..

```
EC E6 C2 74 22 C4 10 A4 E4 FE F1 92 B9 DF 29 33
54 D4 75 91
```

Host sends PSI checksum

```
BF 00 00 BF
```

Host receives acknowledgement

```
01 01
```

Host receives PSI Ready signal

```
00 AA
```

Host sends EBL length

```
34 CE 03 00
```

Host receives acknowledgement

```
CC CC
```

Host sends EBL data

```
28 B4 25 67 80 84 9D 97 13 A2 35 38 9F CC 0A 8B
```

…..suppressed…..

```
36 08 DA 0C 08 09 A6 B8 72 0F 0A 4D 04 08 09 9E
F8 09 51 08   (249396 bytes)
```

Host sends EBL checksum

```
54 00 00 54
```

Host receives acknowledgement

```
54 A5
```

Host receives EBL Info command

```
BB 00 00 00 9A 05 00 00 F4 01 00 00 58 4D 4D 37
31 36 30 5F 31 34 33 34 2E 35 30 30 5F 4D 31 53
31 00 00 00 00 00 00 00 00 00 00 00 03 10 01 01
01 00 01 00 00 01 01 01 00 00 00 00 00 00 00 00
28 01 04 00 04 00 00 00 00 00 00 00
```

Host sends EBL Info command

```
56 08 86 00 48 00 00 00 BB 00 00 00 9A 05 00 00 F4 01 00 00 58
4D 4D 37
31 36 30 5F 31 34 33 34 2E 35 30 30 5F 4D 31 53
31 00 00 00 00 00 00 00 00 00 00 00 03 10 01 01
01 00 01 00 00 01 01 01 00 00 00 00 00 00 00 00
28 01 04 00 04 00 00 00 00 00 00 00
```

Host receives acknowledgement

```
89 00 86 00 02 00 00 00 01 00
```

Host sends BAUDRATE command

```
A4 00 82 00 04 00 00 00 00 10 0E 00
```

Host receives acknowledgement

```
A4 00 82 00 04 00 00 00 00 10 0E 00
```

Host request HW Info command

```
BF 1B 01 08 AC 00 00 00 14 00 00 00 00 00 00 02
00 C2 01 00 00 00 00 00 00 00 00 00 01 00 00 00
FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 FF 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 FF 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

…..suppressed…..

```
00 00 00 00 00 00 00 00
```

Host receives acknowledgement

```
03 08 01 08 02 00 00 00 00 00
```

Host sends Flash Info command

```
86 00 84 00 02 00 00 00 00 00
```

Host receives Flash Info command

```
61 02 84 00 00 01 00 00 00 00 00 00 2C 00 B1 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

…..suppressed…..

```
00 00 00 00 00 00 00 00
```

Host sends Flash Info acknowledgement

62 02 85 00 00 01 00 00 00 00 00 00 2C 00 B1 00

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

…..suppressed…..

00 00 00 00 00 00 00 00

Host receives acknowledgement

85 02 85 00 02 00 00 00 FF FF

Host sends Security Info command

E8 E6 04 02 00 08 00 00 BD 73 12 9F 90 19 99 44

3F BE 17 F1 BE 26 A0 F6 8A A0 D7 1B 9E 7C EF DA

…..suppressed…..

00 00 00 00 00 00 00 00

Host receives acknowledgement

07 02 04 02 02 00 00 00 01 00

Host sends Flash Erase command

1F 0A 05 08 08 00 00 00 00 00 0A 00 FE FF 0B 00

Host receives acknowledgement

07 08 05 08 02 00 00 00 00 00

Host sends Flash Erase Check command

08 08 06 08 02 00 00 00 00 00

Host receives acknowledgement

17 08 06 08 06 00 00 00 01 00 00 0A 00 00

Host sends Flash Write Address command

10 08 02 08 04 00 00 00 00 00 0A 00

Host receives acknowledgement

04 08 02 08 02 00 00 00 00 00

Host sends Flash Write Raw command

15 08 0F 08 04 00 00 00 00 00 02 00

Host sends firmware image

83 C1 9A 9D 76 5C 32 FE 7C CD

…..suppressed…..

83 C1 9A 9D 76 5C 32 FE 7C CD

Host receives acknowledgement

13 08 0F 08 04 00 00 00 00 00 00 00

The process of the host sending the Flash Write Raw command, sending the firmware image and then receiving acknowledgement is repeated until all chunks of data are sent.

Host sends Firmware Checksum command

07 02 05 02 02 00 00 00 00 00 00 00

Host receives acknowledgement

EC 02 05 02 04 00 00 00 01 00 85 5D

Host sends Modem Reset command

```
2E 02 08 02 04 00 00 00 01 10 11 00
```

Modem resets and the download process is finished.

SIERRA
WIRELESS®