Embit Binary Interface

-

Bootloader Guide

embit s.r.l.

Document information

Versions & Revisions

Revision	Date	Author	Comments
1.0	2013-04-18	A. Sala	First release
1.1	2013-04-29	C. Biagi	Minor fixes
1.2	2014-03-14	C. Biagi	Added update procedure, minor fix
1.3	06/05/2014	F. Montorsi	Moved the documentation of binary commands supported by the EBI bootloader from the EBI-WMBus/802.15.4/ZigBee guides to this document
1.4	21/05/2014	F. Montorsi	Add a few comments about the EMB-ZRF2xx family

References

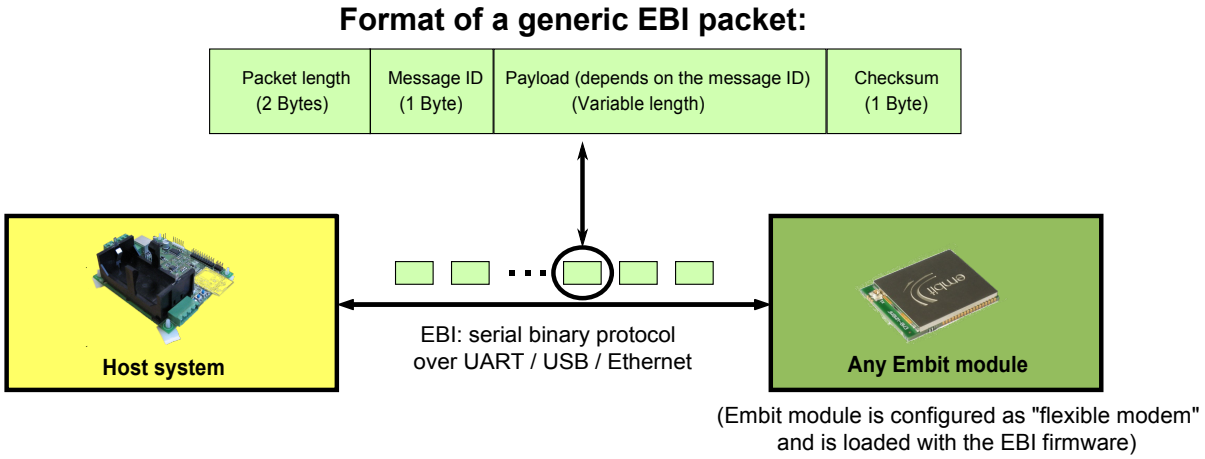
Ref	Version	Date	Author	Title
1	Rev. 2.0	2014	Embit	Embit Binary Interface Overview

Index

1 Introduction

This document is an extension of the “Embit Binary Interface Overview” document [1] and describes the EBI protocol for the Embit wireless modules that implements the bootloader. This document is intended as a reference manual.

Note that in this document the term “host” and the term “module” refer to the customer system hosting the Embit wireless module and the Embit wireless module itself, respectively. An overview of the interaction between the “host” and the “module”, using the EBI protocol, is shown in the following figure:



In the next section, the description of the bootloader and the firmware update is provided.

2 The EBI Bootloader

The bootloader provided with the EBI is an advanced tool capable to update the firmware of the module using the serial communication over the UART port. The bootloader is the first code that executes when the module is powered on and, unless a specific voltage is applied to a module hardware pin (more details below), it does nothing and starts the application stored on the module (usually an EBI variant, e.g., EBI-WMBus, or the customer's own firmware application). For this reason, the bootloader is usually “invisible” to both the firmware application and the MCU interfaced with the Embit module; indeed, to be able to exploit the bootloader to perform the firmware upgrade over UART, it is necessary to force the module's MCU to “enter” the bootloader, as explained in the next Sections.

2.1 Bootloader code area

Usually, the MCU of Embit modules has a specific bootloader area on its FLASH memory, where the EBI bootloader is pre-loaded. However, care has to be taken to avoid removing the EBI bootloader (if the customer wants to preserve the ability to perform firwmare upgrade over UART!); indeed when developing custom firmware applications, it is possible to erase the EBI bootloader by performing a full chip erase or overwriting the bootloader area on the FLASH memory.

2.1.1 EMB-WMBx family

On EMB-WMBx modules the bootloader is stored in the specific BSL FLASH sector of the MSP430 MCU (0x1000 – 0x1800, i.e. the first 2 kB of the FLASH memory);

2.1.2 EMB-Z2530PA and EMB-Z2531PA-USB

The bootloader is typically written in a special bootloader section which ensure that the application can run in its own section without interfering. On the EMB-Z253x the program memory is not separated in sections and so, the bootloader is written on the first 0x800 bytes (2 kB) of the flash. The application will have to take into account for this with the linkerscripts in order to avoid overwriting the bootloader or not working properly. All the EBI versions are already compatible with the bootloader.

2.1.3 EMB-ZRF2xx family

On EMB-ZRF2xx modules the bootloader is stored in a specific “Boot Section” of the Atmel ATxMega MCU, at the end of the FLASH memory dedicated to the firmware application (for 256 kB ATxMega MCUs the boot section starts at 0x20FFF and is 8 kB large);

2.2 Bootloader entering modes

There are two ways to “enter” the bootloader and start the firmware update of the device:

- Hardware entering mode (by RTS / CTS pin signaling)
- Software entering mode (by means of EBI commands)

The hardware entering mode is useful whenever the customer has loaded his own firmware application in the firmware application area

available and can be used on Embit modules where the firmware application is not present; the software mode can only be used on modules with EBI programmed.

2.3 Hardware entering mode

When the bootloader starts, the RTS line is checked and if it is asserted (usually active low), the CTS output line is toggled continuously at a fast rate for a fraction of a second to indicate that the bootloader is ready to receive commands.

The host device will notice the CTS toggling, indicating that the module is ready to start the update process. If no EBI command is received within about 500 ms, and an application is loaded on the module, the bootloader automatically call the application firmware.

To force the bootloader to stop the launch of the application firmware, the host should send the EBI command “Enter bootloader”, then the bootloader is able to receive further commands.

2.4 Software entering mode

If the module has EBI programmed, the bootloader can be entered with special EBI commands without having to deal with the RTS / CTS lines. This is only possible on modules with EBI programmed.

2.5 Update procedure

Once started the bootloader, the update procedure should take the following steps:

1) Send “*Enter Bootloader*” to ensure the stop of the launch of the application already present on the module.
2) Send “*Set Bootloader Options*” (optional step, otherwise defaults value will be used).
3) Send “*Erase memory*”. The current application firmware is erased.
4) Send multiple “*Write memory chunk*”, until all memory content will be programmed.

5) Send "*Commit firmware*" to finalize the update. If this command is not sent, the application will not boot. Note that is necessary to check the return value of the response, because in some cases the update process can not be successful and it must be restarted.

3 PC host application

The host application is to be run on a Windows PC with "*Microsoft .NET Framework 2.0*" or newer installed. Once opened, by clicking "*Browse...*" the hex file containing the firmware to be flashed can be selected. The the proper serial port associated to the module must be selected. By leaving everything to "Auto" the application will try entering the bootloader with a software mode in the different baudrates supported by the module. If this operation fails, the hardware mode will be tried (again with the different baudrates). The user is required to press RESET in order to enter the bootloader when in hardware entering mode.

If the hardware entering mode is preferred, it can be selected on the *combobox* before pressing "*Program*".

Once in the bootloader, the application will retrieve the fastest supported speed of the UART and perform the firmware download with verification (by means of CRC).

4 EBI Bootloader Binary Commands

This chapter provides details on the format of the payload for each different packet. As detailed in [1], the generic packet format for EBI packets is:

Field	Packet length	Message ID	Payload (specific data for each message ID)	Checksum
Length	2 Bytes	1 Byte	Variable	1 Byte

In the following sections the "Message ID" for each EBI-Bootloader command is provided, in hexadecimal format, at the end of the section name; note that the message IDs in this document match the message IDs reported in [1].
The "***Payload format***" paragraphs provide the EBI-Bootloader specification for the variable-length "Payload" field.
Finally, the "***Direction***" paragraphs identify whether the packets are commands sent to the module (host → module) or are replies/notifications sent to the host (host ← module).

4.1 Set bootloader options (0x71)

Direction: host → module.

Valid when: always.

Payload format:

The payload is one byte long formatted as follows:

Field	Chunk size
Length	1 Byte

The “Chunk size” field indicates the size of the memory chunks on which to operate, specified in 2^n bytes:

Data reception	Support on modules: EMB-ZRF2xx	EMB-Z253x	EMB-WMBx
0x04 = 16 Bytes chunk		V	V
0x05 = 32 Bytes chunk		V	V
0x06 = 64 Bytes chunk		V	V
0x07 = 128 Bytes chunk		V	V
0x08 = 256 Bytes chunk		V	V
0x09 = 512 Bytes chunk		V	V

4.1.1 Set bootloader options response (0xF1)

Direction: host ← module.

Payload format:

The payload is a single byte in the “execution status byte” format [1].

4.2 Erase memory (0x78)

Direction: host → module.

Valid when: always.

Payload format:

Erase the non volatile memory of the device.

The payload is one byte long formatted as follows:

Field	Reserved	User memory	Program memory
Length	6 Bits	1 Bit	1 Bit

"User memory": deletes saved settings (all values are set to factory defaults).

"Program memory": delete all code memory.

Notes:

On EMB-Z253x modules, the memory can be only deleted all at a time.

4.2.1 Erase memory response (0xF8)

Direction: host ← module.

Payload format:

The payload is a single byte in the "execution status byte" format [1].

4.3 Write memory chunk (0x7A)

Direction: host → module.

Valid when: always.

Payload format:

The payload is formatted as follows:

Field	Options	Address	Data	Checksum
Length	1 Byte	4 Bytes	X Bytes	2 Bytes

“Options”: left for future uses (for now always zero).

“Address”: absolute value of the base address of the memory to be written.

“Data”: data payload to be written (data length must match the chunk size set in the bootloader options).

“Checksum”: sum of all data and address bytes.

4.3.1 Write memory response (0xFA)

Direction: host ← module.

Payload format:

The payload is a single byte in the “execution status byte” format [1].

4.4 Read memory chunk (0x7B)

Direction: host → module.

Valid when: always.

Payload format:

Reads a chunk of program memory.

The payload is formatted as follows:

Field	Options	Address
Length	1 Byte	4 Bytes

“Options”: left for future uses (for now always zero).

“Address”: absolute value of the base address of the memory to be read.

4.4.1 Read memory response (0xFB)

Direction: host ← module.

Payload format:

The payload is formatted as follows:

Field	Options	Address	Data	Checksum
Length	1 Byte	4 Bytes	X Bytes	2 Bytes

“Options”: left for future uses (for now always zero).

“Address”: absolute value of the base address of the read memory.

“Data”: read data.

“Checksum”: sum of all data and address bytes.

4.5 Commit firmware (0x7F)

Direction: host → module.

Valid when: always.

Payload format:

The packet has no payload.

Notes:

Inform the bootloader that the downloaded firmware is complete and should be enabled (otherwise the firmware update will not complete, holding the module in bootloader mode or with an old firmware).

4.5.1 Commit firmware response (0xFF)

Direction: host ← module.

Payload format:

The payload is a single byte in the “execution status byte” format [1]. If this response indicates successful operation, after the next reboot, the module will start the application (the reboot can be also forced with the reset command).

5 Annex

5.1 Disclaimer

The information provided in this and other documents associated to the product might contain technical inaccuracies as well as typing errors. Regulations might also vary in time. Updates to these documents are performed periodically and the information provided in these manuals might change without notice. The user is required to ensure that the documentation is updated and the information contained is valid. Embit reserves the right to change any of the technical/functional specifications as well as to discontinue manufacture or support of any of its products without any written announcement.

5.2 Trademarks

Embit is a registered trademark owned by Embit s.r.l.

All other trademarks, registered trademarks and product names are the sole property of their respective owners.