Embit Binary Interface

-

ZigBee Specific Documentation

embit s.r.l.

Document information

Versions & Revisions

Revision	Date	Author	Comments
1.0		A. Sala	First release
1.1	14/12/2012	C. Biagi	Minor fixes
1.2	12/03/2013	F. Montorsi	Added hyperlinks; minor fixes
1.3	18/03/2013	F. Montorsi	Minor fixes
1.4	15/04/2013	A. Sala	Minor fixes (on send data command)
1.5	19/04/2013	A. Sala	Added ATI command
1.6	26/02/2014	F. Montorsi	Expanded Introduction; more figures
1.7	14/03/2014	C. Biagi	Merged with “EBI ZigBee Sample Usage”
1.8	17/04/2014	F. Montorsi	Moved the sample usage guide to the top of the document and improved its contents. Moved EBI Bootloader commands to the EBI Bootloader guide

References

Ref	Version	Date	Author	Title
1	Rev. 1.9	2014	Embit	Embit Binary Interface Overview
2	Rev 02	2008	ZigBee Standards Organization	ZigBee Cluster Library Specification

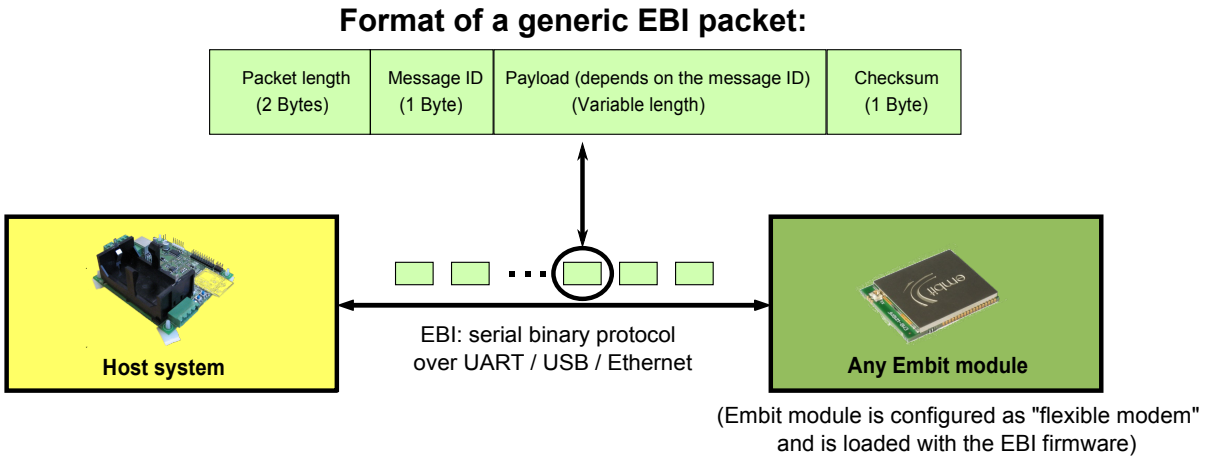
Index

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 support the ZigBee over-the-air protocol [2]. This document is intended as a reference manual.

It is important to specify that, although the EBI protocol abstracts and simplifies some aspects of ZigBee wireless networks, a good knowledge of ZigBee concepts is useful to understand how to use EBI-ZigBee.

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 following Chapter a usage example to get started with EBI-ZigBee is detailed step by step, and is useful to new users.

In Chapter 3 the list of commands specifically supported by EBI-ZigBee is provided, in the form of reference manual.

2 EBI ZigBee Usage Example

This chapter provides a guide useful to get started with EBI for ZigBee. The step by step instructions provided here will guide the user through the creation of a network; moreover, some test data will be exchanged between two Embit EMB-EVB boards.

The devices configured with this guide are not ZigBee compliant as they won't implement all the cluster and profiles required by ZigBee. The aim of this section is only to provide a very first successful experience when communicating wireless with EBI for ZigBee.

2.1 Getting started

To follow step by step this guide, you only need:

1. two EMB-EVB boards (mounting EBI ZigBee modules, like the EMB-ZRF2xx or the EMB-Z253x ones);
2. the Embit EBI ZigBee serial tester software, which is shipped in Embit evaluation kit's disks as `ebi-zigbee-serial-tester.exe`;
3. a PC connected to the two EMB-EVB boards (through USB cables) and running the Embit EBI ZigBee serial tester software;

This guide will provide a list of command to be sent sequentially to the boards in order to establish a communication. The format of these command will be as follow:

command description
payload content

where “payload content” will be the string to copy&paste in the Payload text field of the Embit EBI ZigBee serial tester software. For almost all commands listed later in this document, the Embit module will reply to the command it receives with some bytes in the format

00 05 xx 00 yy

which will appear in the Embit EBI ZigBee serial tester software; in general the “xx” field is a code identifying the message sent by the module to the PC (it can be ignored for now) and the “yy” field is a checksum (it is ignored in the following). The 00 field is the so-called “execution status byte” [1] and is important because it denotes that the command was successful; in case another value appears, then an error occurred and the user is advised to restart the step-by-step procedure from the beginning.

Please refer to the EBI documentation [1] while reading this document for further details on the format of the commands sent and the notifications received.

To get started, just connect the two EMB-EVB boards to a PC and open two instances of Embit EBI ZigBee serial tester software in order to setup each module. Connect the Embit EBI ZigBee serial tester software to the serial ports associated to each board. In the payload text field write the commands as indicated in the following guide pressing the “Send data” button (or ENTER key) to send the command.

In order to verify the connection of the boards to the PC, please send:

Get device information
01

The devices will reply with a command formatted as follows:

Device information response
00 0E 81 WW XX YY YY YY YY YY YY YY YY ZZ

where WW is the protocol in use (typically 20, 21, 22, 23 or 24 for ZigBee). XX is the module (see EBI guides for details). YY are the 8 bytes of the IEEE address (see the comments below). ZZ is the CRC of the packet.

If no reply is received please check the connection with the board, the baudrate set in Embit EBI ZigBee serial tester software (typically 9600 baud) and the hardware flow control (typically disabled).

2.2 Configuration of radio interfaces

2.2.1 Physical address

The first thing to do is to set an appropriate IEEE address on the device. The EMB-Z253x modules have a fixed preprogrammed IEEE address so this step can be skipped. For all other modules, please send:

Set physical address
20 XX XX XX XX XX XX XX XX

where XX are the bytes of the IEEE address to be set. The modules should have an associated IEEE address (printed on a tag or provided with different means). To be compliant to the IEEE 802.15.4 specification, a real IEEE address (allocated by the IEEEE association) shall be used.

2.2.2 Output power

The second thing to do is to set an appropriate output power according to the regulations (please check the appropriate documentation for details). As a starter let's set the output power to 0 dBm:

Set output power
10 00

Please repeat this command on any board involved in the network.

2.2.3 Operating channel

In order to set the operating channel, two methods are available. One can set the operating channel with the appropriate command and set the automated setting to exclude channel or one can set the channel mask to the channels to be used and set the automated settings to configure the channel automatically according to this mask. In this example we'll be using this second method by sending:

Set channel mask to channel 11 only
12 00 00 08 00

Please repeat this command on any board involved in the network.

2.2.4 Network address

The network address on each device can be set manually with a specific command. The coordinator will always reset its address to 0x0000 no matter which address is set; on the other board (router/end-device) we will set instead the 0x0001 address. To keep things as simple as possible, we will not define in this example session which one of the two boards will act as coordinator; we will rather ask EBI-802.15.4 to automatically elect one of the two boards as coordinator. For this reason, for now we can safely set the 0x0001 address on both boards:

Set network address
21 00 01

2.2.5 Auto options

The auto options come to help in creating and managing a network. In our example we'll set them to:

- role (be a coordinator if no network is found or a router otherwise);
- channel (select the best channel from the channel mask);
- network address (automatically allocate if not set on devices);
- associate children;

in practice, this is achieved sending:

Set auto options
24 59 00

Please repeat this command on any board involved in the network.

2.2.6 Endpoint

The endpoints are required in order to operate a device in a ZigBee network. A device without endpoints won't be able to communicate. These endpoints should be set as required by the ZigBee Cluster Library specifications [2]. To simplify here we'll be creating just one endpoint with one manufacturer-specific cluster. In a commercial application a proper configuration of the profile identifier, the device identifier and the clusters is required in order to be ZigBee compliant.

First of all, make sure that no endpoints have already been set (to avoid conflicts with the one which is going to be created):

Remove all endpoints
39 FF

(Note that if an execution status byte different than 00 is received, it can be safely ignored.) To create one endpoint with one cluster send the following payload:

Add endpoint
38 01 C0 00 C0 00 01 80 00 01 80 00

Please repeat these commands on any board involved in the network.

2.2.7 Saving parameters

Optionally, in order to avoid all these commands on the next power cycle we can send:

Save settings
08

Please repeat this command on any board involved in the network.

2.3 Initialization of the radio interfaces

The following commands must be sent to start the network and initiate any communication over the air. At every power cycle, if we have saved the parameters, the network must be started again by sending the following commands.

2.3.1 Stop network

To make sure that the start command will take effect as desired, we first stop any network process on the attached board:

Stop network
30

(Note that if an execution status byte different than 00 is received, it can be safely ignored.)

2.3.2 Start network

The network can now be restarted by issuing the command:

Start network
31

on the first EMB-EVB board; the radio module on such board will not find any personal area network (PAN) with the settings previously specified and will thus start the PAN as a *coordinator*. The acknowledge of the start network command execution may take some time but will be successful (execution status byte equal to 00). Now the same command can be executed on the second board: it will find then the network created by the first one and will join it as a *router*.

2.4 Exchange of data over-the-air

When the network is up and running, data can be exchanged between the two boards by using the following command:

Send data
50 00 00 FF FF C0 00 01 01 80 00 68 00 00 00 01 DD DD DD DD

where:

- “50” is the send command ID;
- “00 00” are the byte options;
- “FF FF” is the destination address; note that it can be set to:
  - 00 00 if the data should be sent to the coordinator;
  - the network address set before (e.g., 00 01) if the data should be sent to the router;
  - FF FF for a broadcast transmission (as in the example above);
- “C0 00” is the profile ID (as set before with the “add endpoint” command);

- “01” and “01” are the source and the destination endpoints (such indexes were set with the add endpoint command);
- “80 00” is the cluster ID;
- “68” is the frame control field indicating a manufacturer specific command;
- “00 00” is the manufacturer code (should be provided by the ZigBee alliance for ZigBee compliant code);
- “00” is the transaction number (should be increased at every transaction);
- “01” is the manufacturer specific command we are sending;
- “DD DD DD DD” is example data which will be sent over-the-air (it can be changed with any other data up to the packet size limit; to keep a safety margin we'll keep this data shorter than 40 bytes in this example).

Once the data has been sent, the destination device(s) will be notified; in practice, a line like

```
Received data notification
00 xx E0 xx xx xx xx xx xx xx xx xx xx xx xx xx xx 01 01 xx xx 00 01 DD DD DD DD xx
```

will appear on the Embit EBI ZigBee serial tester software instance attached to the module which received the data ('xx' denotes bytes which can be ignored for now). Such byte sequence is a “received data notification”; please refer to EBI documentation [1] for information on how this data is formatted. The payload bytes which have been sent (e.g., DD DD DD DD) are visible in such notification.

Note that in case you don't see such a received data notification in the Embit serial tester the following aspects should be checked:

- antenna connections: are the Embit modules mounted on the two EMB-EVB boards correctly connected to their external antennas? Note that if the two modules have an integrated antenna, then this check is not necessary;
- network formation: in case there was an error in the sequence of commands provided to the Embit modules, it may happen that both start the network as coordinators of two different PANs; such a case can be verified using a sniffer tool, like the EMB-Z2531PA-USB, for 2.4 GHz networks. Usually, the easiest way to proceed is to reset the two boards and restart the example session.

The “send data” command and “received data notification” above complete this usage example. In this session you have learned how to:

- use Embit serial tester software to quickly run EBI sessions on Embit modules;
- configure RF and PAN parameters on Embit modules;
- exchange data over-the-air.

For more details about advanced EBI commands and features, please refer to the following Chapter of this document.

3 EBI ZigBee Binary Commands

EBI binary commands allow to control each aspect of the network and the wireless module module behavior. They target embedded hosts that require advanced networking features or complex network topologies.

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-802.15.4 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-802.15.4 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).

3.1 Device information (0x01)

Direction: host → module.

Payload format: no payload.

3.1.1 Device information response (0x81)

Direction: host ← module.

Payload format:

Field	Wireless Protocol	Module	Unique ID
Length	1 Byte	1 Byte	8 Bytes

The "Wireless protocol" field identifies which protocol is implementing the module and is divided in two nibbles, the most significant 4 bits identify the protocol family, the least significant 4 bits identify the variant. Any nibble can be zero indicating unknown value. This is the list of all possibilities:

0x00 = Unknown
0x01 = Proprietary
0x10 = 802.15.4
0x20 = ZigBee
  0x21 = ZigBee 2004 (1.0)
  0x22 = ZigBee 2006
  0x23 = ZigBee 2007
  0x24 = ZigBee 2007-Pro
0x40 = Wireless M-Bus

The "Module" field is divided in three sub-fields as follows:

Field	Family	Model	Revision
Length	4 Bits	2 Bits	2 Bits

Any of these sub-fields can be zero indicating unknown information.

The list of all possibilities is the following:

0x00 = Unknown
0x10 = Reserved
0x20 = EMB-ZRF2xx
  0x24 = EMB-ZRF231xx
    0x25 = EMB-ZRF231
    0x26 = EMB-ZRF231PA
  0x28 = EMB-ZRF212xx
    0x29 = EMB-ZRF212
    0x2A = EMB-ZRF212PA
0x30 = EMB-Z253x
  0x34 = EMB-Z2530x
    0x35 = EMB-Z2530
    0x36 = EMB-Z2530PA
  0x38 = EMB-Z2531x
    0x39 = EMB-Z2531-USB
    0x3A = EMB-Z2531PA-USB
0x40 = EMB-WMBx
  0x44 = EMB-WMB169x
    0x45 = EMB- WMB169T
    0x46 = EMB- WMB169PA
  0x48 = EMB-WMB868x
    0x49 = EMB- WMB868

The “Unique ID” field identifies the module universally through its IEEE address.

Notes:

With this coding, the microcontroller family can be obtained by reading the first 4 bits of the “Module” field.

3.2 Device state (0x04)

Direction: host → module.

Payload format: no payload.

3.2.1 Device state response / Event notification (0x84)

Direction: host ← module.

Payload format:

Single byte indicating the device's state:

0x00 = Booting
0x01 = Inside bootloader
0x10 = Ready (startup operations completed successfully)
0x11 = Ready (startup operations failed)
0x20 = Offline
0x21 = Connecting
0x22 = Transparent mode startup
0x30 = Online
0x40 = Disconnecting
0x50 = Waking up from low power mode
0x51 = End of receiving window

Notes:

The module will send a notification after booting up with a Ready state and then will switch to Offline or Online state (depending on the result of the startup operations, see “Auto network creation” bit in the “Network automated settings” command).

The module will also send a notification when the Offline state is entered directly from the Online state (indicating that the device is orphan).

A “device state notification” might also indicate that the device exited the energy save mode due to an expiring timer (for more information, see the “energy save mode” command).

The “End of receiving window” is not used for ZigBee.

3.3 Reset (0x05)

Direction: host → module.

Payload format: no payload.

3.3.1 Reset response (0x85)

Direction: host ← module.

Payload format: single byte in the “execution status byte” format [1].

Notes:

Please wait for the “device state notification” message that comes at startup after receiving the confirmation in order to allow the module to perform the hardware reset and initialize everything again.

3.4 Firmware version (0x06)

Direction: host → module.

Payload format: no payload.

3.4.1 Firmware version response (0x86)

Direction: host ← module.

Payload format: 4 bytes identifying the firmware version.

3.5 Restore factory default settings (0x07)

Direction: host → module.

Payload format: no payload.

Notes:

The default settings are the following:

Parameter	Default value
Channel Mask	All zigbee channels (11 to 26)
Operating Channel	11
Transmission Power	+11 dBm
Serial Port Settings	9600 baud, 8, N, 1, no flow control
Network role	End device
Network identifier	0x0000000000000123
Energy Save Mode	0x2300 (Module always on, radio polling every 3 seconds)
Auto Parameters	Channel, Network address, Network ID, Association, Role (coordinator or router)
Endpoints	0 (none)

3.5.1 Restore factory default settings response (0x87)

Direction: host ← module.

Payload format: single byte in the “execution status byte” format [1].

Notes:

The module will turn off networking when executing this command and will perform a system reset right after sending this response. Please wait for the “device state notification” message that comes at startup after receiving the response in order to allow the module to perform the hardware reset and initialize everything again.

3.6 Save settings (0x08)

Direction: host → module.

Payload format: no payload.

Notes:

Saves the currently selected settings (operating channel, transmission power, serial port settings, addresses, etc) in the module's internal non-volatile memory.

Once the settings have been saved, they will be used by the module each time the module is (re)started, in place of the factory default settings. Note that the factory default settings can always be restored using EBI command ID 0x07.

3.6.1 Save settings response (0x88)

Direction: host ← module.

Payload format: single byte in the “execution status byte” format [1].

3.7 Serial port configuration (0x09)

Direction: host → module.

Payload format:

The payload is 2 bytes long formatted as follows:

Field	Baudrate	Flow control
Length	1 Byte	1 Byte

The “baudrate” field specifies the baudrate in use as follows:

Support on modules:	EMB-ZRF2xx	EMB-Z253x
0x00 = Maintain current speed	V	V
0x01 = 1200 baud/sec.	V	
0x02 = 2400 baud/sec.	V	V
0x03 = 4800 baud/sec.	V	V
0x04 = 9600 baud/sec.	V	V
0x05 = 19200 baud/sec.	V	V
0x06 = 38400 baud/sec.	V	V
0x07 = 57600 baud/sec.		V
0x08 = 115200 baud/sec.	V	V
0x09 = 230400 baud/sec.		V*
0x0A = 460800 baud/sec.		V*
0x0B = 921600 baud/sec.		

The “Flow control” field specifies the flow control mode in use:

0x00 = Flow control disabled
0x01 = Hardware flow control (modem mode)
0x02 = Hardware flow control (peer to peer mode)

Modem mode means host assert RTS and waits for CTS asserted from the module before sending data. Peer to peer mode means that both module and host will use RTS and CTS as FIFO full signal de-asserting their line when the input buffer are full (for example when using FTDI chips).

Notes:

Using high baudrates can introduce errors, especially with the speeds marked with an asterisk (*). Please keep the baudrate as low as possible when low data is exchanged.

The flow control mode affects the way the module wakes up from energy save mode. While in energy save mode, the module will not be able to receive data over the UART. If modem mode hardware flow control is used, the RTS will be asserted before sending data by the host UART and this will wake the device up from energy save mode. The device will then assert CTS and start receiving the data. This means that in modem mode, the device can serve commands also during energy save mode. The other hardware flow control modes will not be able to operate correctly during energy save mode, and so, energy save timeouts must be used.

3.7.1 Serial port configuration response (0x89)

Direction: host ← module.

Payload format: Single byte in the "execution status byte" format [1].

Notes:

If the execution is acknowledged with a "Success" response, the module will switch to the new settings immediately after the response has been sent, otherwise it will remain with current settings. Please wait at least 25 ms after the reception of the response to allow the module to switch to the new baudrate.

3.8 Output power (0x10)

Direction: host → module.

Payload format:

To retrieve the current output power, the packet is sent with an empty payload.

To set the output power of the module, the payload is a single signed byte indicating the output power to be used in dBm.

Accepted values:

EMB-Z253xPAx [-5, +20]
EMB-ZRF231PA [0, +20]

3.8.1 Output power response (0x90)

Direction: host ← module.

Payload format:

If getting the current output power, the payload is a single signed byte indicating the output power in use in dBm.

If setting the channel, the payload is a single byte in the “execution status byte” format [1].

3.9 Operating channel (0x11)

Direction: host → module.

Payload format:

To retrieve the current channel the packet is sent with an empty payload.

To set the channel the payload is a single unsigned byte indicating the channel to be used (accepted values: [11, 26]).

Notes:

The operating channel can only be changed when network is down.

3.9.1 Operating channel response (0x91)

Direction: host ← module.

Payload format:

If getting the current channel, the payload is a single byte indicating which channel is currently being used.

If setting the channel, the payload is a single byte in the “execution status byte” format [1].

3.10 Active channel mask (0x12)

Direction: host → module.

Payload format:

To retrieve the active channel mask, the packet is sent with an empty payload.

To set the channel the payload is a 32 bit unsigned value (most significant byte transmitted first) indicating the channel mask to be used. The least significant byte is channel 1, the most significant byte is channel 32. On 2.4 GHz, the 802.15.4 standard supports channel from 11 to 26.

Notes:

The channel mask can only be modified when the device is disconnected.

3.10.1 Active channel mask response (0x92)

Direction: host ← module.

Payload format:

If getting the current channel, the payload is a 32 bit unsigned value (most significant byte transmitted first) indicating the channel mask in use.

If setting the channel, the payload is a single byte in the “execution status byte” format [1].

3.11 Energy save (0x13)

Direction: host → module.

Payload format:

To retrieve the current energy save options, the packet is sent with an empty payload. To set the energy save options, the payload is as follow:

Field	Radio (RX) policy	Host interface (MCU) policy	Settings
Length	1 Byte	1 Byte	Variable

The "Radio (RX) policy" field specifies how the radio should be set in low power mode according to the following table:

Data reception	Support on modules: EMB-ZRF2xx	EMB-Z253x
0x00 = Always on (maximum power consumption)	V	V
0x01 = Always off (receive data only after sending some data)	V	V
0x02 = Follow MCU policy (receive when MCU awakes)	V	V
[0x11 : 0x20] = Every (# - 0x10) * 100 ms	V	V
[0x21 : 0x5C] = Every (# - 0x20) seconds	V	V

The "Host interface (MCU) policy" field specifies how the microcontroller should save energy according to the following list:

Host interface (MCU):	Support on modules: EMB-ZRF2xx	EMB-Z253x
0x00 = always on (maximum power consumption)	V	V
0x01 = always off (modem hardware flow control needed)	V	V
0x02 = enabled every "Wake up interval", disabled after "Sleep timeout" since the last UART activity.	V*	V
0x03 = enabled every "Sleep interval" - "Anticipation", disabled when "Sleep interval" fires (UART activity independent).		
0x04 = Wireless Mbus synchronous message		

* this setting is only partially supported

The “settings” field is optional and specifies additional settings specific for each Host interface policy. If not sent, the last settings used for the selected policies will be used. The length and format of this field is variable according to the following table:

Host interface policy	Settings field length	Fields	
0x00	0 bytes		
0x01	0 bytes		
0x02	6 bytes	Wake up interval (4 bytes)	Sleep timeout (2 bytes)
0x03	6 bytes	Sleep interval (4 bytes)	Wake up anticipation (2 bytes)

The “Wake up interval” specifies an interval in time (milliseconds) after which to enable the host interface. Accepted values are [20 : 0xFFFFFFFF] where 0xFFFFFFFF means never wake up (wake up only through outgoing UART activity or incoming UART activity if hardware flow control is set to modem mode).

The “Sleep timeout” specifies how many milliseconds to wait before entering sleep mode again. Accepted values are 0; [5 : 0xFFFF] where 0 means enter sleep-mode immediately and 0xFFFF means never enter sleep-mode again. If “Sleep timeout” is bigger than “Wake up interval” the device will always stay on (but will keep sending status notification when “Wake up interval” fires).

The “Sleep interval” specifies an interval in time (milliseconds) after which the device will be set into sleep mode.

The “Wake up anticipation” indicates when the device should wake up (in milliseconds before the expiration of the “sleep interval”). If “Wake up anticipation” is bigger than “Sleep interval” the device will always stay on sending status notification right after “Sleep interval” fires. Accepted values are in the [5 : 0xFFFF] range where 0xFFFF means never enter low power mode (keep sending periodic notifications).

Notes:

On EMB-Z253x the radio policy 0x00 (always on) is not available for end devices and will behave as radio policy 0x01 (always off).

For switching between radio policy 0x00 to other radio policies, the device must be disconnected.

The radio policy and the microcontroller policy are not to be confused and should be considered independent options.

If the UART hardware flow control is in modem mode, the host will be able to always wake up the device by pulling low the RTS (sending data). This method of waking up the device is the one to be preferred and will not interfere with the energy save timers.

If the UART hardware flow control is not in modem mode, the host will lose control of the device during the host interface sleep period. The timers must be used wisely (allowing enough time with host interface enabled to receive a full packet).

When sending this command the device will reset the timers as if the interval timers were just expired (entering sleep mode immediately for “host interface policy” 0x03, while staying awake for “Sleep timeout” for “host interface policy” 0x02 as an example).

To manually enter sleep-mode immediately use the “Enter sleep mode” command.

When the host interface is woken up from low power due to a timer expiring, a status notification will be sent to inform the host that it can send commands. No notification will be issued when entering low power mode.

If “Radio (RX) policy” is set to “Follow MCU policy”, the device will poll its parent for data only when waking up timers expire. The radio will not poll continuously during the awake period (will just send one poll at the beginning of the period) and so, data will only be received at the beginning of the wake up period.

The selected data polling interval (only for Radio (RX) policy different than always on) will be automatically increased to one poll every 100ms after sending or receiving over the air data (to allow fast download of responses and multiple queued packets).

Only end devices are allowed to sleep.

Notes for EMB-ZRF2xx:

On the EMB-ZRF2xx the sleep modes are dealt at stack level in a way that makes it difficult to predict the next poll event.

If low power modes are required, the suggested solution is to use modem flow control and MCU always asleep, the RX poll interval can be chosen as preferred.

If other energy-save modes are preferred please consider that intervals and timeouts cannot be assured in this architecture as explained in the following notes:

In this architecture, the node will always sleep for the required poll interval and then send a poll at wake up. If the device enters sleep mode, the elapsed time from the last poll event is not taken into account and the next poll event will happen in poll interval plus elapsed time since last poll event (the effective time between polls will therefore be in the “required poll interval” to 2 * “required poll interval” range). Also if using mcu policies that involve timers, consider that the time is approximative and it can vary up to twice as much as the required interval. This behaviour depends on the ratio between poll interval and sleep periods and also is affected by manually interrupting sleep mode by RTS or incoming packets over the air. Also changing intervals when network is running can amplify these effects.

3.11.1 Energy save response (0x93)

Direction: host ← module.

Payload format:

If getting the current energy save options, the payload is a single byte as described in the request.

If setting the energy save options, the payload is a single byte in the “execution status byte” format [1].

3.12 Force sleep (0x14)

Direction: host → module.

Payload format:

The command is used to manually enter sleep-mode immediately regardless of energy save timers. The command can be used with an empty payload for waking up at next scheduled instant (see Energy-save) or with a 32 bit unsigned integer indicating the time to spend in sleep mode (overriding the energy-save setting).

Field	Sleep-mode timeout
Length	0/4 Bytes

"Sleep-mode timeout": optional, in order to override the value specified in the "Energy save" parameter. Range [3 : 0xFFFFFFFF] where 0xFFFFFFFF means sleep forever.

Notes:

The device will enter sleep mode immediately after sending the response.

If the UART hardware flow control is not in modem mode, the host will not have a way to wake up the device (such as RTS switching in modem mode), in this case, to avoid losing control of the module, the sleep-mode timeout must be used wisely.

When exiting the forced sleep mode period, the normal behaviour specified with the Energy-save command will be restored.

If radio is always enabled, the device will not enter sleep mode.

Some combinations of "host interface policy" and "sleep-mode timeout" don't make sense and are not allowed such as:

Any sleep-mode timeout and host interface policy set to always off
Sleep-mode timeout not attached and host interface policy set to always on or always off
Sleep-mode timeout = 0xFFFFFFFF with host interface policy set to always on

If the current host interface energy-save policy has a "Wakeup anticipation" parameter, the sleep-mode timeout will be reduced by this anticipation.

Only end devices are allowed to sleep.

3.12.1 Force sleep response (0x94)

Direction: host ← module.

Payload format:

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

3.13 Force data poll (0x15)

Direction: host → module.

Payload format:

The command is used to manually send a data poll to the module's parent. If the parent have queued data for the node, it will be received right after the data poll. The packet has no payload.

Notes:

Only sleeping end devices are allowed to poll for data.

3.13.1 Force data poll response (0x95)

Direction: host ← module.

Payload format:

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

3.14 Physical address (0x20)

Direction: host → module.

Payload format:

To retrieve the physical address of a module, the packet is sent with an empty payload.

To set the physical address of a module, the payload is an 8 bytes field (sent most significant byte first) indicating the physical address to be used (any value accepted).

Notes:

The physical address might be only changed when device is disconnected.

The EMB-Z253x has an integrated physical address that has already been registered to the IEEE. Because of this, changing the physical address is not allowed for this device.

3.14.1 Physical address response (0xA0)

Direction: host ← module.

Payload format:

If getting the physical address, the payload is an 8 bytes field (sent most significant byte first) indicating the physical address of the node.

If setting the physical address, the payload is a single byte in the “execution status byte” format [1].

3.15 Network address (0x21)

Direction: host → module.

Payload format:

To retrieve the network address associated with the node, the packet is sent with an empty payload.

To set the network address to be used, the payload is a 2 bytes field (sent most significant byte first) indicating the network address to be used (accepted values: [0, 0xFFF7]).

Notes:

The network address might be only changed when device is disconnected.

3.15.1 Network address response (0xA1)

Direction: host ← module.

Payload format:

If getting the network address, the payload is a 2 bytes field (sent most significant byte first) indicating the network address in use.

If setting the network address, the payload is a single byte in the “execution status byte” format [1].

3.16 Network identifier (0x22)

Direction: host → module.

Payload format:

To retrieve the network identifier in use on the node, the packet is sent with an empty payload. Note that the EBI "network identifier" corresponds, in the context of IEEE 802.15.4 networks, to the Personal Area Network Identifier (PAN ID).

To set the network identifier, the payload is an 8 bytes field (sent most significant byte first) indicating the network identifier to be used (accepted values: [1, 0xFFFFFFFFFFFFFFFF]).

Notes:

The network identifier might be only changed when device is disconnected.

3.16.1 Network identifier response (0xA2)

Direction: host ← module.

Payload format:

If getting the network address, the payload is an 8 bytes field (sent most significant byte first) indicating the network identifier in use.

If setting the network address, the payload is a single byte in the "execution status byte" format [1].

3.17 Network role (0x23)

Direction: host → module.

Payload format:

To retrieve the selected network role, the packet is sent with an empty payload.

To set the network role, the payload is a single unsigned byte with the following meaning:

0x00 = Coordinator
0x01 = Router
0x02 = End Device

3.17.1 Network role response (0xA3)

Direction: host ← module.

Payload format:

If getting the network role, the payload is a single unsigned byte as specified in the request packet.

If setting the network role, the payload is a single byte in the “execution status byte” format [1].

3.18 Network automated settings (0x24)

Direction: host → module.

Payload format:

The payload can be empty for reading the current setting or formatted as follows for writing it:

Field	Auto network creation	Auto channel	Auto network address	Auto network identifier	Auto associate children	Auto restore network	Auto role	Reserved
Length	1 Bit	1 Bit	1 Bit	1 Bit	1 Bit	1 Bit	2 Bits	8 Bits

“Auto network creation”: if set, the module will invoke a network start at startup right after loading the data from NVM in order to automatically create a network.

“Auto channel”: if set, this bit will make the device pick the less busy channel (if creating a network) or the channel with the best network to associate to (if joining a network). All the channels possibilities must be included in the channel mask.

“Auto network address”: if set makes an end device or router get the network address from the parent they are joining to. In a coordinator this parameter will be ignored and the network address used will be one declared with the “set network address” command.

“Auto network identifier”: if set makes the end device or router join the best network available without caring about the PANID (perfect for the first join). In a coordinator this bit makes the coordinator pick a random network identifier.

“Auto associate children”: if set, the module will accept every association requests and will automatically allocate an address for the devices requiring it without asking or informing the host about it.

“Auto restore network”: if set, when a coordinator will be started, it will restore the previous network (if any) without doing the normal network formation procedure (useful in case of coordinator crashes). This will happen only if previous network had some status to be saved (joined devices, etc). This will only be considered by coordinators.

“Auto role”: if set to 1, the module will start a network as coordinator unless a network with the defined parameters is found. In that case, the node will join the network as a router. If set to 2, the module will start as coordinator or end device.

Notes:

If auto role and auto network identifier are both set, the device will join the first network it finds (no matter which network identifier it has) or will create a network with a random network identifier otherwise.

On the EMB-ZRF2xx and EMB-Z253x the children are always auto associated.

On the EMB-ZRF2xx the network restore option is not supported.

3.18.1 Network automated settings response (0xA4)

Direction: host ← module.

Payload format:

If getting the network automated settings, the payload is as specified in the request.

If setting the network automated settings, the payload is a single byte in the “execution status byte” format [1].

3.19 Network preferences (0x25)

Direction: host → module.

Payload format:

To retrieve the active network preferences, the packet is sent with an empty payload.

To set the network preferences, the payload is formatted as follows:

Field	Joining permitted (open network)
Length	1 Byte

"Joining permitted": time in seconds during which to accept new devices to join the network. If set to 0x00, the network will not accept joining requests from devices which were not part of the network. If set to 0xFF the network will open and never close again automatically.

Notes:

On the EMB-ZRF2xx, the maximum value for the "Joining permitted" field is 60 (or 0xFF).

3.20 Network security (0x26)

Direction: host → module.

Payload format:

For security reasons, the security settings cannot be read.

To set the security settings, the payload is formatted as follows:

Field	Options	Network key
Length	1 Byte	0/16 Bytes

The “Options” field specify the options to enable:

b7 (MSB) – Enable security
b6 – Keys are pre-shared (avoid sending keys over the air during joining)
b5 ↔ b1 – Reserved
b0 – Update network key with the one attached

The “Network key” field specify the network key to use and is only attached if the associated bit of the “Options” field is set.

Notes:

The security settings can be changed only when the network is down (offline).

The EMB-Z253x doesn't support choosing between security enabled or disabled at runtime. For this reasons two different firmwares are provided (one for operating with secured networks and the other for unsecured networks).

3.20.1 Network security response (0xA6)

Direction: host ← module.

Payload format:

If setting the network preferences, the payload is a single byte in the “execution status byte” format [1].

For security reasons, the security settings cannot be read.

3.21 Network stop (0x30)

Direction: host → module.

Payload format:

The packet has no payload

Notes:

On the EMB-Z253x modules, the network stop command is internally followed by a system reset to ensure that the stack is clean. Expect the device to reset, save NVM if needed before calling Network stop and wait until the "device state notification" informs that the device is ready to accept commands.

3.21.1 Network stop response (0xB0)

Direction: host ← module.

Payload format:

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

3.22 Network start (0x31)

Direction: host → module.

Payload format:

The packet has no payload.

Notes (network role is set to coordinator):

If the auto options include channel, a network scan on the active channel mask will be performed before creating the network and the most silent channel will be selected, otherwise, the network scan will be performed only on the selected channel to ensure that there is no network with the same PANID running already (the EMB-Z253x modules will not check for networks with same PANID).

If the channel mask is 0 and the auto options include channel, the operation will fail.

If a network with identical PANID exists in one of the channels in the channel mask, the operation will fail (the EMB-Z253x modules will not check for networks with same PANID).

If the network address of the device is set to 0xFFFF the operation will fail.

Notes (network role is set to router or end device):

If the auto options include channel, a network scan on the active channel mask will be performed and the best parent with the same PANID as the one set in the device will be selected for joining, otherwise, the network scan will be performed only on the selected channel.

If the channel mask is 0 and the auto options include channel, the operation will fail.

The network address set will be ignored by the device if no PANID is set (auto PANID). This is because for ZigBee standard, the coordinator allocates addresses to fresh joining devices.

Notes (network role is set to any of the Auto modes):

A network scan on the selected channel (or all channels in channel mask if the auto options include channel) will be performed and if an active coordinator with the selected PANID is found, the device will join.

A device reset (indicated by the status notification) after starting a network start command might indicate a network joining procedure failed during authentication. Consider checking the status notification also (in addition to the network start response) when using security.

3.22.1 Network start response / notification (0xB1)

Direction: host ← module.

Payload format:

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

Note:

It is strongly suggested to save the settings whenever the network is started correctly in order to remember all the parameters for the next network startup.

3.23 Network scan (0x32)

Direction: host → module.

Payload format:

The packet has two bytes of payload. The first sent byte has the following meaning:

0x00 = Perform an energy scan
0x01 = Perform a passive network scan and return all compatible networks
0x02 = Perform an active network scan and return all compatible networks

The second byte send over the serial interface indicates how much time to spend on each channel with the following formula (x is the value sent with this second byte):

timePerChannel = 15.75 ms * (1 << x);

A typical value of this byte for network discovery is 3. For energy measurements, the longer this time is, the more accurate the results will be.

Notes:

The network/energy scan will be performed on the channels selected with the channel mask. The time needed for this operation depends on the selected channel count and the time spent on each channel.

Typically, in 802.15.4 networks, the network scan operations will be performed with active scan (request for beacons).

This command is not available on the EMB-ZRF2xx and EMB-Z253x modules.

3.23.1 Network scan response (0xB2)

Direction: host ← module.

Payload format:

The payload have two different formats depending if the request was a network scan or an energy scan. Also, if the request cannot be processed, the payload will be a single executions status byte informing about the error.

Payload format for energy scan:

For energy scans, the response has two fields as follows:

Field	Execution status byte	Least noisy channel	Most noisy channel	Channel RSSI data
Length	1 Byte	1 Byte	1 Byte	32 Bytes

The “Execution status byte” specifies if the operation concluded successfully and further data is present.

The “Least noisy channel” field indicates which channel revealed the minimum RF power during the scan (or zero if no channel was selected).

The “Most noisy channel” field indicates which channel revealed the maximum RF power during the scan (or zero if no channel was selected).

The “Channel RSSI data” (most significant byte transmitted first) contains the RSSI acquired on each channel (least significant byte refers to channel 1). Only channels 11 to 26 are available for 802.15.4. Channels who are not selected will contain zeros.

Payload format for network scan:

Field	Execution status byte	Network count	Network data
Length	1 Byte	1 Byte	Variable

The “Execution status byte” specifies if the operation concluded successfully and further data is present.

The “Network count” field indicates how many networks were found.

The “Network data” is the concatenated data for each network formatted as follows:

Field	PANID	Channel	Rssi
Length	2 Bytes	1 Byte	1 Byte

3.24 Add endpoint (0x38)

Direction: host → module.

Payload format:

Field	Endpoint number	Profile ID	Device ID	Input cluster number	Input clusters	Output cluster number	Output clusters
Length	1 Byte	2 Byte	2 Byte	1 Byte	Variable	1 Byte	Variable

The “Endpoint number” field identifies the endpoint on the device. Range: [0x01, 0xEF].

The “Profile ID” field must be set according to ZigBee specifications (for example: Home Automation 0x0104).

The “Device ID” field must be set according to ZigBee specifications.

The “Input(Output) cluster number” indicate how many clusters are supported on the endpoint for each direction.

The “Input clusters”(“Output clusters”) fields are lists of the supported cluster (unsigned 16 bit integers).

3.24.1 Add endpoint response (0xB8)

Direction: host ← module.

Payload format:

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

3.25 Remove endpoint (0x39)

Direction: host → module.

Payload format:

The payload is a single byte indicating which endpoint to remove, or 0xFF to remove every endpoint.

3.25.1 Remove endpoint response (0xB9)

Direction: host ← module.

Payload format:

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

3.26 Associated addresses (0x40)

Direction: host → module.

Payload format:

The payload for this packet is the address of the device. The address can be a network address (16 bit) or an extended IEEE address (64 bit). The module will detect which address is sent based on the packet length and will respond.

3.26.1 Associated addresses response (0xC0)

Direction: host ← module.

Payload format:

If the payload is one byte long, this is an execution status response. Otherwise the payload is in the following format:

Field	Execution status byte	Network address	IEEE address
Length	1 Byte	2 Bytes	8 Bytes

The “Execution status byte” indicates if the data was found or retrieved correctly.

The “Network address” is the short address of the device.

The “IEEE address” is the extended address of the device.

Notes:

This packet will return a negative execution status response if the module fails to find an address association in its tables and with over the air requests.

3.27 Associating device (0x41)

Direction: host ← module.

Payload format:

Field	IEEE address	Capability information
Length	8 Byte	1 Byte

The “IEEE address” is the extended address of the device requiring association.

The “Capability information” show the functionalities implemented in the device (as specified in the IEEE 802.15.4 specification).

Notes:

The association can be performed automatically without this set of commands when the associated bit is set in the automatic options.

3.27.1 Associating device response (0xC1)

Direction: host → module.

Payload format:

Field	Execution status byte	Network address
Length	1 Byte	0/2 Bytes

The “Execution status byte” indicates if the device is to be associated (0) or rejected (anything else).

The “Network address” can be inserted or not. If present, the module will associate the specified address to the device during association, if not, it will pick an address automatically.

Notes:

The response must arrive to the module before 300 ms from request.

This command is not available on the EMB-ZRF2xx and EMB-Z253x module.

3.28 Send data (0x50)

Direction: host → module.

Payload format:

Field	Options	Custom channel	Custom power	Destination network identifier	Destination address	ZigBee data
Length	2 Bytes	0/1 Byte	0/1 Byte	0/2 Bytes	2/8 Bytes	Variable

The “Options” field indicates to the module which option to apply to the request:

b15 (MSb) – Send on specific channel
b14 – Send with specific output power
b13 – Send on specific network (for interpan messages)
b12 – Security enabled
b11 ↔ b2 - Reserved
b1 – Send with extended source address (instead of short network address)
b0 (LSb) – Send to extended destination address (instead of short network address)

The “Custom channel” field is only present if the bit 15 in the “options” field is set. It indicates on which channel this packet must be sent. On 2.4 GHz, the 802.15.4 standard supports channels from 11 to 26.

The “Custom power” field is only present if the bit 14 in the “options” field is set. It indicates which output power to use with this specific packet. The power is expressed in dBm in a signed integer as described earlier for the “Output power” command.

The “Destination network identifier” field is only present if the bit 13 in the “options” field is set. It indicates to which PANID the packet is addressed and can be used for interpan messages if the field is set to 0xFFFF.

The “Destination address” field is the 16 bit network address (most significant byte transmitted first) of the recipient device. The address can be its extended IEEE address (64 bit long, most significant byte transmitted first) if the bit 0 of the “Options” field is set.

The “ZigBee data” field is formatted as follows:

Field	Profile ID	Source Endpoint	Destination Endpoint	Cluster ID	ZigBee application payload
Length	2 Bytes	1 Byte	1 Byte	2 Byte	Variable

The “Profile ID” field specifies to which profile this packet belongs to.

The “Source Endpoint” field specifies from which endpoint the packet must be sent.

The “Destination Endpoint” field specifies to which endpoint the packet is sent.

The “Cluster ID” field specifies to which cluster the following command belongs.

The “ZigBee application payload” is the effective payload to be sent to the destination endpoint. If a ZCL application is targeted, this payload should implement the ZCL format as described in the ZigBee Cluster Library Specifications [2]. In details, the first bytes will be formatted as follows:

Field	Frame control field	Manufacturer code	Transaction sequence number	ZCL command	Data
Length	1 Byte	0/2 Bytes	1 Byte	1 Byte	Variable

For more information on these fields, please consult the ZigBee Cluster Library Specifications [2, chapter 2.3.1].

The “Data” field contains the data to be delivered to the destination endpoint. It's maximum length depends on the enabled options and selected command.

Notes:

The EMB-ZRF2xx doesn't support sending packets on different channels, with a different power or to a different network identifier.

The EMB-Z253x doesn't support specifying the security to use during packet send. The security setting used will be the one of the network.

The EMB-Z253x will not buffer broadcast messages for sleeping end-devices. Because of this, sleeping end devices will not receive broadcast messages.

The EMB-Z253x doesn't support sending with extended addresses (both sender and receiver). It doesn't support sending packets on different channels or with a different power.

3.28.1 Send data response (0xD0)

Direction: host ← module.

Payload format:

Field	Execution status byte	Retries	RSSI of the acknowledge
Length	1 Byte	0/1 Byte	0/1 Byte

Notes:

The “Retries” parameter is only present in some architectures.

The “RSSI of the acknowledge” field is only present if the “Execution status byte” indicates success and the packet was not a broadcast.

On the EMB-ZRF2xx the only field present is the Execution status byte

3.29 Received data (0x60)

This packet should not be sent and will be ignored by the module (use notification instead).

3.29.1 Received data notification (0xE0)

Direction: host ← module.

Payload format:

Field	Options	Rssi	Source network identifier	Destination network identifier	Source address	Destination address	ZigBee Data
Length	2 Byte	0/1 Byte	0/2 Bytes	0/2 Bytes	2/8 Bytes	2/8 Bytes	Variable

The "Options" field indicates to the host which fields are present in the packet and which options the received packet was implementing:

b15 (MSb) – Rssi attached
b14 – Sent from a different network identifier
b13 – Sent to a different network identifier
b12 – Security enabled
b11 ↔ b2 - Reserved
b1 – Extended source address (instead of short network address)
b0 (LSb) – Extended destination address (instead of short network address)

The "Rssi" field indicates the received signal strength in dBm (signed integer) and is only present if the associated bit in the "Options" field is set.

The "Source network identifier" field is only present if the bit 14 in the "Options" field is set. It indicates from which PANID the packet was sent.

The "Destination network identifier" field is only present if the bit 13 in the "Options" field is set. It indicates to which PANID the packet was sent (broadcast PANID for example).

The "Source address" field is the 16 bit network address (most significant byte transmitted first) of the sending device. The address can be its extended IEEE address (64 bit long, most significant byte transmitted first) if the source device sent it with extended address information instead of network address information. In this case, bit 1 of the "Options" field is set.

The "Destination address" field is the 16 bit network address (most significant byte transmitted first) of the recipient device. The address can be its extended IEEE address (64 bit long, most significant byte transmitted first) if the bit 0 of the "Options" field is set.

The "ZigBee data" field is formatted as follows:

Field	Profile ID	Source Endpoint	Destination Endpoint	Cluster ID	ZigBee application payload
Length	2 Bytes	1 Byte	1 Byte	2 Byte	Variable

The “Profile ID” field specifies to which profile this packet belongs to.

The “Source Endpoint” field specifies from which endpoint the packet must be sent.

The “Destination Endpoint” field specifies to which endpoint the packet is sent.

The “Cluster ID” field specifies to which cluster the following command belongs.

The “ZigBee application payload” is the effective payload to be sent to the destination endpoint. If a ZCL application is targeted, this payload should implement the ZCL format as described in the ZigBee Cluster Library Specifications. In details, the first bytes will be formatted as follows:

Field	Frame control field	Manufacturer code	Transaction sequence number	ZCL command	Data
Length	1 Byte	0/2 Bytes	1 Byte	1 Byte	Variable

For more information on these fields, please consult the ZigBee Cluster Library Specifications (chapter 2.3.1).

Notes:

The packets will be sent to the host only if they match the internal address filter (destination network identifier identical to the one set in the module or broadcast and destination network address identical to the one set in the module or broadcast).

On EMB-Z253x and EMB-ZRF2xx, the profile ID is not considered and always returned as 0xFFFF.

On EMB-ZRF2xx, the source and destination PAN IDs are not checked and will always return the PAN ID of which the device is part of.

3.30 Enter bootloader (0x70)

Direction: host → module.

Payload format:

The packet has no payload

This command enters in the bootloader from an application (when using bootloader software entering method).

A hardware bootloader entering method also exist and works as follows: assert RTS, reset module, the CTS line will be toggled 10 times every 50 ms by the module to indicate that the system is in the bootloader. During this time, the host can send a software “Enter bootloader” command to stop the procedure and stay in the bootloader. If no command is received after the 10 toggles (500 ms), the bootloader will jump to the application.

In order to speed up the boot of the module when the bootloader is not needed, please deassert RTS.

3.30.1 Enter bootloader response (0xF0)

Direction: host ← module.

Payload format:

The payload is a single byte in the “execution status byte” format [1]. When this packet is sent, the module is in the bootloader and ready to accept further commands.

4 Annex

4.1 Disclaimer of liability

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.

4.2 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.