Embit Binary Interface - IEEE 802.15.4-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	13/03/2013	F. Montorsi	Added hyperlinks; minor fixes
1.3	25/03/2013	A. Sala	Minor fixes
1.4	31/07/2013	A. Sala	Introduced peripheral commands
1.5	15/11/2013	A. Sala	Introduced EMB-Z2538PA platform. Corrections on AT commands.
1.6	26/02/2014	F. Montorsi	Expanded Introduction; more figures
1.7	17/04/2014	F. Montorsi	Added Usage Example
2.0	23/05/2014	F. Montorsi	Added usage example, added EMB-ZRF212B-specific comments, simplified “Energy save” command, aligned with firmware versions

References

Ref	Version	Date	Author	Title
1	Rev. 2.0	2014	Embit	Embit Binary Interface Overview
2	Rev. 1.3	2014	Embit	Embit Bootloader Guide
3	IEEE Std 802.15.4™-2006	2006	IEEE Standard for Information technology	Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)

Index

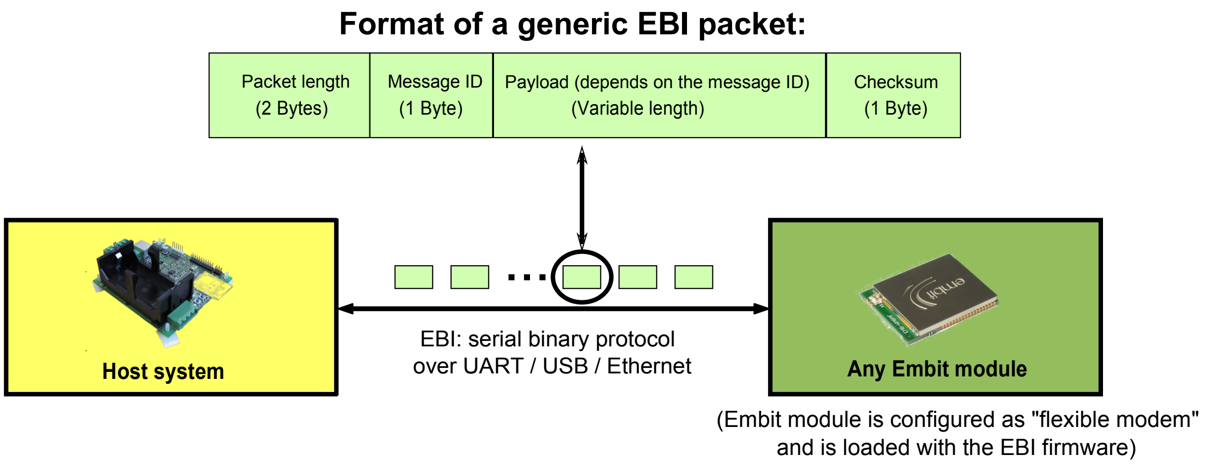
1 Introduction

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 IEEE 802.15.4 over-the-air protocol [3]. This document is intended as a reference manual, although it also provides a simple usage example. For information regarding the module firmware upgrade using the bootloader, please refer to the document "Embit Bootloader Guide" [2].

This document refers to the EBI-802154 firmware version "01 40 50 7A" and upwards.

It is important to specify that, although the EBI protocol abstracts and simplifies some aspects of 802.15.4 wireless networks, a good knowledge of IEEE 802.15.4 concepts is useful to understand how to use EBI-802.15.4.

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:


Format of a generic EBI packet:
Packet length (2 Bytes)
Message ID (1 Byte)
Payload (depends on the message ID) (Variable length)
Checksum (1 Byte)
Host system
EBI: serial binary protocol over UART / USB / Ethernet
Any Embit module
(Embit module is configured as "flexible modem" and is loaded with the EBI firmware)

In the following Chapter a usage example to get started with EBI-802.15.4 is detailed step by step, and is useful to new users.

In Chapter 3 the list of commands specifically supported by EBI-802.15.4 is provided, in the form of reference manual.

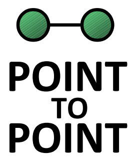
2 EBI 802.15.4 Network Structure

This chapter provides some information on the type of network topologies supported by EBI-802.15.4, how to exchange data in such networks and how IEEE 802.15.4 concepts are mapped in EBI-802.15.4.

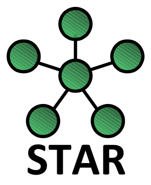
2.1 Network Topologies Supported

EBI-802.15.4 supports the same network topologies supported by IEEE 802.15.4, that is:

- ***Point-to-point networks***:
  this is the simplest type of network and it consists of only two devices: a coordinator (also known as "concentrator") and an end-device that must be associated with the coordinator.



- ***Star networks***:
  this is an extension of the point-to-point network consisting of a coordinator and multiple end-devices; each end-device must associate with the coordinator and may send or receive data to/from the coordinator.



Note that EBI-802.15.4 does not explicitly support *routing* of packets: just as in IEEE 802.15.4 networks the communication is from coordinator to end-devices or viceversa, without multi-hop communication.

Finally note that EBI-802.15.4 may support fairly large star networks depending on the Embit module employed:

- EMB-ZRF2xx: up to 128 devices
- EMB-Z2538PA: up to 128 devices
- EMB-Z2530PA / EMB-Z2531PA-USB: up to 32 devices

2.2 Low Power Aspects

In EBI-802.15.4 it is required that the coordinator of the network never turns off its radio, so that it is always ready to receive/send data from/to end-devices. Indeed the coordinator of an IEEE 802.15.4 network typically is not battery-powered and thus it has no strong energy consumption constraints.

The end-devices of the network instead are usually battery-powered and thus support different low-power features via the "energy save" (0x13) command. In particular, the end devices are supposed to "sleep" (i.e., disable their radio) for most of the time and then wake-up at periodic intervals to send a "data poll" request (0x15) to the coordinator of the network; if the network coordinator has data queued for the end-device it will reply to the "data poll" request. Note that such low-power policy entails that:

1. when the host MCU on the coordinator-side sends data to the Embit module acting as the coordinator for delivery to some end-device, such data will be delivered to

the end-device only when the end-device wakes up and sends the “data poll” request[1]; depending on the settings used in the “energy save” command, **this may entail a latency in coordinator → end-device data delivery of several seconds;**

2. when the host MCU on the end-device-side sends data to the Embit module acting as the end-device for delivery to the coordinator, such data will be immediately delivered to (and possibly received by) the coordinator, which is always on. For this reason, **there is no latency in the end-device → coordinator data delivery**.

To better understand low-power aspects in EBI-802.15.4 it is strongly suggested to read the low-power policy of the IEEE 802.15.4 standard [3]. In particular EBI-802.15.4 will use “direct transmission” or “indirect transmission” depending on the low-power modes enabled on the two devices exchanging data.

2.3 IEEE 802.15.4 Compliance

EBI-802.15.4 creates a *beacon-less* IEEE 802.15.4-compliant wireless network.

Since EBI protocol abstracts some concepts from the underlying over-the-air protocol (see [1]) it may be useful to define how the IEEE 802.15.4 concepts are mapped in EBI-802.15.4:

EBI-802.15.4 term	Corresponding IEEE 802.15.4 term
Physical address	IEEE MAC address (8 bytes long)
Network identifier	Personal Area Network Identifier (PAN ID)
Coordinator	Coordinator (fully-functional device, FFD)
Router	fully-functional device (FFD)
End device	fully-functional device (FFD)

In particular note that in the context of EBI-802.15.4 the “physical address” and the “IEEE address” are synonyms.

1 Note that when exiting the low-power mode the end-device will automatically send a “data poll request” to the concentrator; see the “energy save” (0x13) command for more information.

3 EBI 802.15.4 Usage Example

This chapter provides a guide useful to get started with EBI for 802.15.4. 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.

3.1 Getting started

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

1. two EMB-EVB boards (mounting Embit modules programmed with EBI-802.15.4, like the EMB-ZRF2xx or the EMB-Z253x ones);
2. the Embit EBI 802.15.4 serial tester software, which is shipped in Embit evaluation kit's disks as `ebi-802154-serial-tester.exe`;
3. a PC connected to the two EMB-EVB boards (through USB cables) and running the Embit EBI 802.15.4 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”
command payload content

where “command payload content” will be the string to copy&paste in the Payload text field of the Embit EBI 802.15.4 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 802.15.4 serial tester software (see Img.1); numbers are expressed in hex base. In this specific case, the 00 05 indicates the message length of 5 bytes; 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 indicates the content of the response coming from the module. It depends on the specific Command ID (refer to the specific section of this document), and can be interpreted in two ways:

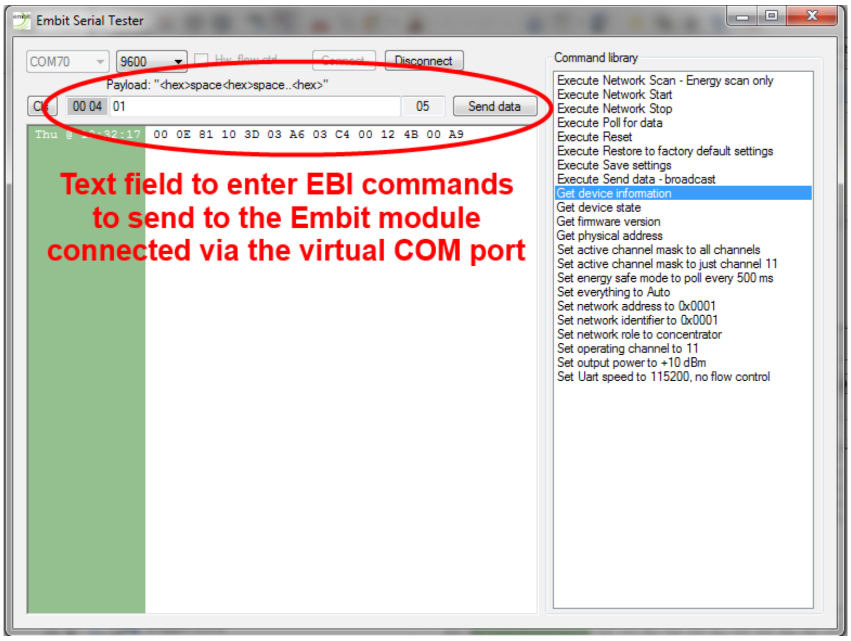
- if a command was sent, the 00 field is the so-called “execution status byte” [1] and is important because it denotes that the command was successful (e.g., Network start); in case another value appears, then an error occurred;
- if a value was requested from the module (e.g., the network address), the value is returned.

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 802.15.4 serial tester software in order to setup each module. Select the virtual serial port to each program instance and press “Connect” button to initiate the communication.

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.

Following is a screenshot of the application, with highlighted the area where the EBI commands can be entered:


Embit Serial Tester
COM70
9600
Hw flow ctrl
Connect
Disconnect
Payload: "<hex>space<hex>space..<hex>"
Cls
00 04
01
05
Send data
Text field to enter EBI commands to send to the Embit module connected via the virtual COM port
Command library

```
Thu @ 1[illegible]:32:17  00 0E 81 10 3D 03 A6 03 C4 00 12 4B 00 A9
```

Execute Network Scan - Energy scan only
Execute Network Start
Execute Network Stop
Execute Poll for data
Execute Reset
Execute Restore to factory default settings
Execute Save settings
Execute Send data - broadcast
Get device information
Get device state
Get firmware version
Get physical address
Set active channel mask to all channels
Set active channel mask to just channel 11
Set energy safe mode to poll every 500 ms
Set everything to Auto
Set network address to 0x0001
Set network identifier to 0x0001
Set network role to concentrator
Set operating channel to 11
Set output power to +10 dBm
Set Uart speed to 115200, no flow control

Img. 1: Serial Tester Window

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

> “01 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 (10 for 802.15.4). XX is the module (see Chapter 3 for details). YY are the 8 bytes of the IEEE address (see the comments below). ZZ is the checksum of the packet.

If no reply is received please check the connection with the board, the baudrate set in Embit EBI 802.15.4 serial tester software (default 9600 baud) and the hardware flow control (typically disabled).

3.2 Quick example

In this example the shortest command sequence to get the module working is illustrated.

3.2.1 Set role

In order to set up a network, it is necessary to elect one board as the network concentrator. So, on the first board send the command:

```
“23 Network role – Set concentrator”
23 00
```

The second board must be set as an “end device” using the command:

```
“23 Network role – Set end device”
23 02
```

3.2.2 Start network

The network can now be restarted by issuing the command:

```
“31 Network start”
31
```

on the first EMB-EVB board (concentrator). 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 an end device and return execution status byte equal to 00. If no coordinator is found, an error is returned (execution status byte equal to 01).

3.2.3 Exchange data over-the-air

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

```
“50 Send data - broadcast”
50 00 00 FF FF 01 02 03 04 05 06 07 08
```

This command will send the payload “01 02 03 04 05 06 07 08” to all devices on the network, using the broadcast network address “FF FF”.

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 01 02 03 04 05 06 07 08 xx
```

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

3.3 Advanced example

3.3.1 Stop network

To make sure that the start command will take effect as desired (some commands can be executed only when the network is stopped), we first stop any network process on the attached board:

```
“30 Network stop”
30
```

Note that if the module is already in stop state (e.g., it is just powered on or resetted), an execution status byte different than 00 is received. It can be safely ignored.

3.3.2 Physical address

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

```
“20 Physical address - Set”
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.

3.3.3 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:

```
“10 Output power - Set +10 dBm”
10 00
```

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

3.3.4 Set role

In order to set up a network, it is necessary to elect one board as the network coordinator. So, on the first board send the command:

```
“23 Network role – Set concentrator”
23 00
```

This command also set its network address to 0x0000 and its energy mode to “00 00” (this means that its radio is always in reception).

The second board must be set as an “end device” using the command:

```
“23 Network role – Set end device”
23 02
```

The end device normally employs the Indirect Data Transfer (as defined in IEEE 802.15.4 specifications). This means that the end device wakes up periodically to poll the concentrator, asking for incoming data, and then going back in sleep mode. The network address of the end device will be assigned by the concentrator during the association phase at the network start.

3.3.5 Operating channel

In order to set the operating channel, two methods are available:

- set a selection of enabled channels and let the concentrator select automatically the best channel;
- select a particular channel and do not enable the automatic channel selection.

In this example, the first method is used. For simplicity, only the channel 11 is enabled by sending:

```
“12 Active channel mask – Set ch 11”
12 00 00 08 00
```

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

Note: 00 00 08 00 (big endian order) equals to '1' shifted left 11 times.

3.3.6 Network ID (PAN ID)

The network ID (PAN ID) on each device can be set manually with the associated command.

```
“22 Network id – Set 0001”
22 00 01
```

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

3.3.7 Network automated settings

The network automated settings help creating and managing a network. In our example we'll set them to:

- channel (select the best channel from the channel mask);
- associate children (any new end node will be associated automatically by the concentrator)

In practice, this is achieved sending:

```
“24 Network automated settings - Set”
24 48 00
```

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

3.3.8 Set energy save mode

In this optional step it is possible to enable an energy save mode policy on the end device. To reduce the power consumption, the end device can automatically and periodically enter and exit sleep mode. It it possible to configure the sleep time and the wake time using the “Set energy mode” command.

During the sleep period, the microcontroller is in sleep mode and it is not possible to communicate with it. At the wakeup, it signal its state sending the event notification “0x50 = Waking up from low power mode” (see the section “event notification 0x84”). After this notification the module executes a “DATA POLL” to the concentrator, requesting data that it was not possible to receive during the module sleep time.

After the event notification it is possible to communicate with the module for the wake time previously set.

To use the energy save mode send:

```
“13 Energy save - Set”
13 02 02 00 00 07 D0 03 E8
```

on the end node board. In this case the timing set are:

• Radio mode:	02 = Follow MCU policy
• MCU mode:	02 = enable every wakeup interval
• Wake up interval:	00 00 07 D0 = 2000 ms sleep
• Sleep interval:	03 E8 = 1000 ms awake

so the end node will sleep for 2 seconds, wakeup (polling the concentrator and sending the event notification), remaining awake for 1 second waiting for further commands, and then go back to sleep.

3.3.9 Saving parameters

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

```
“08 Save settings”
08
```

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

Note: the UART configuration parameters are saved to the flash, so please be careful when saving “non standard” configurations (at the next poweron you must use the previously saved settings, otherwise you won't be able to communicate).

3.3.10 Initialization of the radio interfaces

In the next sections the command described will start the radio interfaces of the Embit modules and initiate communications over the air. Note that at every power cycle, even if the previous parameters have been saved, the network must be started again, with the commands detailed below.

3.3.11 Start network

The network can now be restarted by issuing the command:

```
“31 Network start”
31
```

on the first EMB-EVB board (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 an end device and return execution status byte equal to 00. If no coordinator is found, an error is returned (execution status byte equal to 01).

Note: network addresses are assigned by the concentrator. The concentrator address is fixed to 00 00; end nodes address are assigned starting form 00 01. In this example only one end node is present, so its address will be 00 01. For more complex networks, it is possible to manage the address using the commands "Network Address" on the end node and using the command "Associated Device List".

3.3.12 Exchange data over-the-air

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

```
"50 Send data - broadcast"
50 00 00 FF FF 01 02 03 04 05 06 07 08
```

where:

- "50" is the send command ID;
- "00 00" are the send 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 end device;
  - FF FF for a broadcast transmission (as in the example above);
- "01 02 03 04 05 06 07 08" is example data which will be sent over-the-air (it can be changed with any other data up to the packet size limit).

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 01 02 03 04 05 06 07 08 xx
```

will appear on the Embit EBI 802.15.4 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 Chapter 3 for information on how this data is formatted. Note that the payload bytes which have been sent over-the-air using the other Embit serial tester instance (in this example: 01 02 03 04 05 06 07 08) are visible in such notification message.

Note: the concentrator is configured with reception always enabled, so it is able to receive both direct messages and broadcast messages. The end device usually employs energy save mode, so it steadily turns on and off the power section, polling concentrator for data. Consequently broadcast messages cannot be received by end devices. To communicate with the end note you should use always explicit addressing:

```
"50 Send data – to addr 0001"
50 00 00 00 01 01 02 03 04 05 06 07 08
```

Note: 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.

3.3.13 Conclusion

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.

4 EBI 802.15.4 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).

4.1 Device information (0x01)

Direction: host → module.

Valid when: always.

Payload format: no payload.

4.1.1 Device information response (0x81)

Direction: host ← module.

Payload format:

Field	EBI Protocol	Embit Module	Embit UUID
Length	1 Byte	1 Byte	8 Bytes

The “EBI protocol” field is divided in two sub-fields of 4 bits as follows:

Field	EBI Protocol Family	EBI Protocol Variant
Length	4 Bits	4 Bits

Any nibble can be zero indicating unknown value. This is the list of all possible values for the “EBI protocol” field:

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 “Embit Module” field is divided in three sub-fields as follows:

Field	Embit Module 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 possible values for the “Embit module” field is:

- 0x00 = Unknown
- 0x10 = Reserved
- 0x20 = EMB-ZRF2xx
  - 0x24 = EMB-ZRF231xx
    - **0x26 = EMB-ZRF231PA**
  - 0x28 = EMB-ZRF212xx
    - **0x29 = EMB-ZRF212B**
- 0x30 = EMB-Z253x
  - 0x34 = EMB-Z2530x
    - **0x36 = EMB-Z2530PA**
  - 0x38 = EMB-Z2531x
    - **0x3A = EMB-Z2531PA-USB**
  - 0x3C = EMB-Z2538x
    - **0x3D = EMB-Z2538PA**
- 0x40 = EMB-WMBx
  - 0x44 = EMB-WMB169x
    - **0x45 = EMB-WMB169T**
    - **0x46 = EMB-WMB169PA**
  - 0x48 = EMB-WMB868x
    - **0x49 = EMB-WMB868**

The “Embit UUID” field contains the so-called Embit Universally Unique Identifier (known as “UUID” or just “UID”) that is an 8-bytes sequence identifying the module universally.

For the **EMB-Z253x** family (i.e., EMB-Z2530PA, EMB-Z2538PA, EMB-Z2531PA-USB) the Embit UUID coincides with the concept of physical address (see the “Physical Address (0x20)” command for more information) and also with the unique IEEE address of the module (all such identifiers are 8 bytes long).

For the **EMB-ZRF2xx** family (i.e., EMB-ZRF231PA, EMB-ZRF212B) the UUID is a unique sequence of 8 bytes which is NOT necessarily a valid IEEE address (even if it is used as default IEEE/physical address); this is due to the fact that the devices of the EMB-ZRF2xx family do not have a physical/IEEE address hardwired in their memory. For such a reason the physical address of the module on the EMB-ZRF2xx family must be set after start up with the “Physical Address (0x20)” command. Embit provides a valid physical/IEEE address on the printed label stick on the module.

Finally, for the **EMB-WMBx** family (i.e., EMB-WMB169PA, EMB-WMB868) the UUID is a unique sequence of 8 bytes which identifies the device but that does not necessarily coincides with the physical address; this is due to the fact that the devices of the EMB-WMBx family do not have a physical address hardwired in their memory. For such a reason the physical address of the module on the EMB-WMBx family must be set after start up with the “Physical Address (0x20)” command. Embit provides a valid physical address on the printed label stick on the module. Note that for the EBI-WMBus variant employed on EMB-WMBx devices there is no concept of IEEE address, since Wireless M-Bus does not employ such type of addressing.

4.2 Device state (0x04)

Direction: host → module.

Valid when: always.

Payload format: no payload.

4.2.1 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 (0x10 or 0x11) 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 (see associated command).

The "End of receiving window" is not used for 802.15.4.

4.3 Reset (0x05)

Direction: host → module.

Valid when: always.

Payload format: no payload.

4.3.1 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.

4.4 Firmware version (0x06)

Direction: host → module.

Valid when: always.

Payload format: no payload.

4.4.1 Firmware version response (0x86)

Direction: host ← module.

Payload format: 4 bytes identifying the firmware version.

4.5 Restore factory default settings (0x07)

Direction: host → module.

Valid when: network is stopped.

Payload format: no payload.

Notes:

The default settings are the following:

Parameter	Default value for EMB-Z253x and EMB-ZRF231PA modules	Default value for EMB-ZRF212B modules
Active channel mask	All IEEE 802.15.4 2.4 GHz channels (11 to 26)	All IEEE 802.15.4 868 / 915 MHz channels (0 to 10)
Operating channel	11	0
Transmission power	+11 dBm	+10 dBm
Serial port settings	9600 baud, 8, N, 1, no hardware flow control	
Network role	End device	
Network identifier	0x0001	
Network address	0x0000	
Network automated settings	0x4800 (Auto channel, Auto children association)	
Network preferences	0x01 (Network is open to children association)	
Energy save mode	0x0000 (MCU always on, radio always in reception)	
Physical address	For **EMB-Z253x** family: the physical address is read-only and consists of a valid IEEE 802.15.4 address For **EMB-ZRF2xx** family: the physical address default value is the Embit UUID, which is NOT a valid IEEE 802.15.4 address and needs to be changed to a proper value before use; see “Device Information” for more details.	

4.5.1 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.

4.6 Save settings (0x08)

Direction: host → module.

Valid when: network is stopped.

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.

4.6.1 Save settings response (0x88)

Direction: host ← module.

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

4.7 Serial port configuration (0x09)

Direction: host → module.

Valid when: always.

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	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 behavior of the RTS and CTS pins of the Embit module:

0x00 = Flow control disabled
0x01 = Hardware flow control enabled in modem mode

The “modem mode” hardware flow control means that, to communicate with the Embit module, the host MCU should assert the RTS line and then wait for the CTS line assertion (transition from logic high to logic low) from the Embit module before sending data over the UART interface.

Notes:

The use of high baudrate values may introduce errors on the UART communications, especially with the speed values marked with an asterisk (*). Please keep the baudrate as low as possible when few data bytes are 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 MCU and this will wake up the Embit module from energy save mode. The module 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.

4.7.1 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.

4.8 Output power (0x10)

Direction: host → module.

Valid when: network is stopped.

Payload format:

To retrieve the current output power, the packet must be 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-Z2530PA	[-5, +20]
EMB-Z2531PA-USB	[-5, +20]
EMB-Z2538PA	[+8, +20]
EMB-ZRF231PA	[+5, +20]
EMB-ZRF212B	[-25, +11]

4.8.1 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].

4.9 Operating channel (0x11)

Direction: host → module.

Valid when: network is stopped.

Payload format:

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

To set the channel the payload is a single unsigned byte indicating the channel to be used (note that such channels are those indicated in the IEEE 802.15.4 standard [3] for channel page #0):

Channel [hex]	Channel [dec]	Freq. [MHz]	Valid for:
0x00	0	868,3	EMB-ZRF212B
0x01	1	906	
0x02	2	908	
0x03	3	910	
0x04	4	912	
0x05	5	914	
0x06	6	916	
0x07	7	918	
0x08	8	920	
0x09	9	922	
0x0A	10	924	
0x0B	11	2405	EMB-Z253x family, EMB-ZRF231PA
0x0C	12	2410	
0x0D	13	2415	
0x0E	14	2420	
0x0F	15	2425	
0x10	16	2430	
0x11	17	2435	
0x12	18	2440	
0x13	19	2445	
0x14	20	2450	
0x15	21	2455	
0x16	22	2460	
0x17	23	2465	
0x18	24	2470	
0x19	25	2475	
0x1A	26	2480	

Notes:

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

The selected channel must be enabled in the current channel mask, otherwise the command with fail with the “not supported” execution status byte [1]. Use the “Active channel mask” first to select an appropriate channel mask.

4.9.1 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].

4.10 Active channel mask (0x12)

Direction: host → module.

Valid when: network is stopped.

Payload format:

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

To set the channel mask the payload is a 32 bit unsigned value (most significant byte transmitted first):

Field	Channel mask
Length	4 Bytes

The “Channel mask” field indicates to the module which channels the module should take into account when it receives the commands:

- operating channel (0x11)
- network start (0x31)
- network scan (0x32)

In particular each channel is flagged as “active” by a bit inside the channel mask. The least significant bit is channel 0, the most significant bit is channel 31. At 2.4 GHz, the IEEE 802.15.4 standard supports channel from 11 to 26; at 868/915 MHz bands the IEEE 802.15.4 standard supports channels from 0 to 10.

See the command “Operating channel (0x11)” for more details about how to change the operating channel.

Notes:

The channel mask can only be modified when the network is stopped.

If the current operating channel is not included in the channel mask which is provided with the “active channel mask” command, the operating channel will be moved to the first channel available in the channel mask which has been provided.

If the given channel mask contains channels not supported by the current device (e.g., a channel mask “activating” channel 0 is provided to a 2.4 GHz module), then this command will fail with the “unsupported value” execution status byte [1].

4.10.1 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].

4.11 Energy save (0x13)

Direction: host → module.

Valid when: network is stopped.

Payload format:

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

Field	Module sleep policy	Time intervals (optionals)
Length	2 Bytes	Variable

The "Module sleep policy" field specifies how the module should save power:

Module sleep policy: Support on modules:	EMB-ZRF2xx	EMB-Z253x
0x0000 = both MCU core and radio are always on (maximum power consumption)	V	V
0x0201 = keep both MCU and radio in sleep mode, wakeup using RTS pin	V	V
0x0202 = keep both MCU and radio in sleep mode, wakeup periodically, every "Wake up interval"	V	V

The "Time intervals" field is optional and can be provided to the module only when "Module sleep policy" is 0x0202. It is used to set the "Wake up interval" and the "Sleep timeout" and is formatted as follows:

Field	Wake up interval	Sleep timeout
Length	4 Bytes	2 Bytes

The "Wake up interval" specifies (in milliseconds) how often the module will wake up from sleep mode. Accepted values are [20 : 0xFFFFFFFF] where 0xFFFFFFFF means never wake up (wake up only through outgoing UART activity or incoming UART activity if UART hardware flow control is set to "modem mode"). Note that the module will automatically send a "data poll request" to the concentrator every time it wakes up from sleep, in order to receive any queued data eventually available on the concentrator side.

The “Sleep timeout” specifies how many milliseconds to wait, after waking up, 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).

Examples:

How to disable energy save mode (reception always enabled):

```
Energy save mode
23 00 00
```

How to set automatic module wakeup interval (of both MCU and radio) every 5 seconds (0x1388 ms) with automatic return to sleep after 500 ms (0x01F4 ms):

```
Energy save mode
23 02 02 00 00 13 88 01 F4
```

(Note that such wakeup setting is valid only for end devices.)

Notes regarding command validity:

The energy save mode is effective at the network start (when the network is stopped, the MCU is always running and capable to listen EBI UART commands).

Concentrators shall use energy save mode 00 00; indeed setting the network role to “concentrator” automatically resets energy save mode to 00 00.

To switch between energy save modes, the device must be disconnected (network stopped).

Notes regarding low-power in IEEE 802.15.4 networks:

Direct communication (also known as “direct transmission” in IEEE 802.15.4 networks) is allowed only when energy save mode 00 00 is employed.

When energy save mode is employed (different to 00 00), the communication with the end device is achieved using “indirect transmission”. See Chapter 2 or the IEEE 802.15.4 standard [3] and Chapter 2 of this document for further information.

If the “Wakeup interval” is too large, the concentrator could discard a packet intended for the sleeping end device before the wakeup of the end node.

Notes regarding wakeup from low-power mode:

If the UART hardware flow control is in modem mode, the host MCU will be able to wake up the sleeping module by pulling low the RTS (sending data). This method of waking up the device is the one to be preferred. If the UART hardware flow control is not in modem mode, the host won't be able to communicate with the module during the module sleep period.

When the module is exits low power mode due to an expiring timer (i.e., at the end of a “wakeup interval”), it will send a device state notification over UART to inform the host MCU that it can send commands to the module again. Moreover, the end device will automatically poll the concentrator for data, to retrieve data intended for the end device eventually queued on the concentrator. Note that the module will send just one “data poll request” immediately after exiting low-power and so, if available, data from concentrator will be received at the beginning of the wake up period only.

Notes regarding entering low-power mode:

To manually enter sleep-mode immediately use the “Force sleep” (0x14) command.

No notification are sent over UART when the device enters low power mode.

When sending the energy save command the device will reset the timers as if the interval timers were just expired.

4.11.1 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].

4.12 Force sleep (0x14)

Direction: host → module.

Valid when: network is stopped.

Payload format:

The command is used to manually enter sleep-mode immediately regardless of energy save timers set via the “energy save (0x13)” command. The command can be used with an empty payload for waking up at next scheduled instant (see “energy-save” command) or with a 32 bit unsigned integer indicating the time to spend in sleep mode (overriding the energy save “sleep timeout” setting).

Field	Sleep timeout
Length	0/4 Bytes

“Sleep timeout”: optional value; can be used to override the value specified in the “energy save” parameter. Range [3 : 0xFFFFFFFF] where 0xFFFFFFFF means sleep forever.

Notes regarding entering low-power mode:

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

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

Only end devices are allowed to sleep.

Notes regarding wakeup from low-power mode:

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 behavior specified with the “energy-save” command will be restored.

4.12.1 Force sleep response (0x94)

Direction: host ← module.

Payload format:

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

4.13 Force data poll (0x15)

Direction: host → module.

Valid when: network is started.

Payload format:

The command is used to manually force a module configured as “end-device” to send a “data poll request” to the PAN concentrator. If the concentrator has data queued for the end device, the concentrator will be send the data and the “end-device” module will generate a “Received data notification”.

This packet has no payload.

Notes:

Only modules configured as “end devices” and configured to sleep (via the “energy save” command) are allowed to poll for data. This command is not valid for modules running as concentrators.

4.13.1 Force data poll response (0x95)

Direction: host ← module.

Payload format:

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

4.14 Physical address (0x20)

Direction: host → module.

Valid when: network is stopped.

Payload format:

To retrieve the physical address of a module, the packet must be 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:

In EBI-802.15.4, the physical address coincides with the IEEE address used in the IEEE 802.15.4 over-the-air packets.

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. Indeed for EMB-Z253x modules the values of:

- Embit UUID (see the "Device information (0x01)" command);
- physical address;
- IEEE address

are all coincident.

4.14.1 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 module.

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

4.15 Network address (0x21)

Direction: host → module.

Valid when: network is started.

Payload format: no payload.

Notes:

This packet can be used to retrieve the network address associated with the module. Note that the network address is always allocated by the coordinator of the IEEE 802.15.4 network and cannot be explicitly set (although on the coordinator side the host MCU is allowed to define the IEEE address ↔ network address mapping via the “associating device” notification).

In EBI-802.15.4 concentrators always have network address 0x0000.

4.15.1 Network address response (0xA1)

Direction: host ← module.

Payload format:

The payload is a 2 bytes field (sent most significant byte first) indicating the network address in use. Note that the address is 0x0000 if the module has started the network as coordinator; otherwise it is the value decided by the network coordinator.

4.16 Network identifier (0x22)

Direction: host → module.

Valid when: network is stopped.

Payload format:

To retrieve the network identifier in use on the module, this packet must be 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 a 2 bytes field (sent most significant byte first) indicating the network identifier to be used (accepted values: [1, 0xFFFE]).

4.16.1 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].

4.17 Network role (0x23)

Direction: host → module.

Valid when: network is stopped.

Payload format:

To retrieve the selected network role, the packet must be 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

Note that setting the network role may fail if the current network automated settings have some auto-role policy enabled.

Note:

In the standard IEEE 802.15.4 there is no definition of router (see Chapter 2 for more information); for this reason in EBI-802.15.4 routers behave like end devices.

Setting the network role to “concentrator” also sets the energy save mode to 0x00 0x00 (continuous reception) and the network address to 0x0000.

4.17.1 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].

4.18 Network automated settings (0x24)

Direction: host → module.

Valid when: network is stopped.

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	Reserved	Auto network identifier	Auto associate children	Reserved	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 channel possibilities must be included in the channel mask.

“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 role”: if set to 0, the module will start with the network role currently set (see EBI command ID “Network 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 module will join the network as a router. If set to 2, the module will start as coordinator or end device.
Warning: use an “Auto role” value different from 0 may lead to unexpected behavior; the “Auto role” setting is not supported on EMB-ZRF2xx platforms.

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.

Default setting is 0x4800 (i.e., “Auto channel” bit set and “Auto associate children” bit set).

4.18.1 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].

4.19 Network preferences (0x25)

Direction: host → module.

Valid when: always.

Payload format:

To retrieve the active network preferences, the packet must be 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": if set to 0x00, the network will not accept joining requests from devices which were not part of the network. If set to 0x01 the network is open and joining requests will be accepted; note that if the "Auto associate children" bit of the "Network automated settings" is set then the new nodes will be automatically enrolled in the network.
Note: this value only has effect when the module has started the network as coordinator.

4.19.1 Network preferences response (0xA5)

Direction: host ← module.

Payload format:

If getting the network preferences, the payload is as specified in the request packet. Some modules might not support this get version of this command and will return an empty payload to inform about the impossibility to retrieve the setting.

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

4.20 Network stop (0x30)

Direction: host → module.

Valid when: network is started.

Payload format:

The packet has no payload

4.20.1 Network stop response (0xB0)

Direction: host ← module.

Payload format:

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

4.21 Network start (0x31)

Direction: host → module.

Valid when: network is stopped.

Payload format:

The packet has no payload.

Notes valid when the network role is set to "coordinator":

If the network automated settings include channel, a network scan on the channels included in the active channel mask will be performed before creating the network and the less noisy 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 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).

Notes valid when the network role is set to "router" or "end device":

If the network automated settings 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 PANID is set to all 0xFF, the device will join the strongest network (without caring about the PANID).

Notes valid when the automated network settings specify that network role will be automatically chosen:

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

4.21.1 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 store all the parameters for the next network startup.

4.22 Network scan (0x32)

Direction: host → module.

Valid when: network is stopped.

Payload format:

Field	Scan type	Time per channel
Length	1 Byte	1 Byte

The packet has two bytes of payload.

The "Scan type" field 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 "Time per channel" field indicates how much time to spend on each channel with the following formula; a value 0x00 indicates a very fast and possibly inaccurate scan; a value of 0x07 is the maximum value and indicates a slow and accurate scan. A typical value of the "Time per channel" for network discovery is 3, which translates to a network scan of duration ~150ms per each channel. 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; e.g., selecting 15 channels and providing the maximum scan duration value (0x07) may result in up to 30 seconds of scan activity.

Typically, in 802.15.4 networks, the network scan operations are performed using the active scan option (i.e., the device will actively request over-the-air for beacons from coordinators listening on that channel).

4.22.1 Network scan response (0xB2)

Direction: host ← module.

Payload format:

The payload has two different formats depending on the "Scan type" field provided in the request. However, if the scan request cannot be processed, the payload will be a single execution status byte describing the error [1].

Payload format for response to "Scan type" values 0x00 (energy scan):

For energy scans, the response payload is composed by 4 fields as follows:

Field	Execution status byte	Least noisy channel number	Most noisy channel number	Channel energy levels
Length	1 Byte	1 Byte	1 Byte	27 Bytes

The “Execution status byte” field specifies if the network scan operation was completed successfully and whether the remaining fields are present.

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

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

The “Channel energy levels” field contains the energy values acquired on the scanned radio channels (from channel 0 to channel 26); each i-th byte is an unsigned byte that indicates the amount of energy detected on the i-th channel. Note that the energy values are not expressed in dBm so that they are useful only to retrieve a *relative* indication of the energy in the various channels: a channel having energy level 0x10 contains less energy than a channel having energy level 0x20.

Note that at 2.4 GHz, the IEEE 802.15.4 standard supports channel from 11 to 26, so that for 2.4 GHz modules the first 11 bytes (for channel from 0 to 10) will all be zero; the viceversa holds for sub-GHz modules supporting only channels from 0 to 10.

Finally, the energy levels for channels that are not part of the active channel mask (see command 0x12) are zero.

Payload format for response to “Scan type” values 0x01 (passive scan) or 0x02 (active scan):

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

The “Execution status byte” field specifies if the network scan operation was completed successfully and whether the remaining fields are present.

The “Network count” field indicates how many IEEE 802.15.4 networks were found during the scan operation.

The “Network data” is the concatenated data for each IEEE 802.15.4 network formatted as follows:

Field	Network ID	Channel	RSSI
Length	2 Bytes	1 Byte	1 Byte

For each IEEE 802.15.4 network found:

- the “Network ID” field indicates the PAN ID of the network that has been found;
- the “Channel” field is a number from 0 to 26 indicating the channel on which the network has been found;
- the “RSSI” field is a signed integer indicating the RSSI level in dBm for the network that has been found.

4.23 Address translation (0x40)

Direction: host → module.

Valid when: network is started.

Payload format:

This command can be used to retrieve the association/mapping between a network address (2 bytes long) and an extended IEEE address (8 bytes long) or viceversa (similarly to the "ARP" requests of the TCP/IP protocol). This command can be sent only to a module that started the network as coordinator and results in a lookup inside the coordinator's internal table of associated devices.

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 with the translated/mapped address.

4.23.1 Address translation 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.

4.24 Associating device (0x41)

Direction: host ← module.

Valid when: network is started.

Payload format:

Field	IEEE address	Capability information
Length	8 Byte	1 Byte

This is a notification message sent from the module to the host whenever the module has started the network as coordinator and an other device is trying to associate with it.

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

The “Capability information” field describes the features implemented in the device which is trying to associate (as specified in the IEEE 802.15.4 specification [3]).

Notes:

The association can be performed automatically without this set of commands when the “auto associate children” bit is set in the “network automated settings”.

4.24.1 Associating device response (0xC1)

Direction: host → module.

Payload format:

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

This packet must be sent from the host to the module, to indicate if the device attempting the association is allowed to associate or not.

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

The “Network address” field is optional. If present, the concentrator will assign to the device attempting association the network address provided in this command. If this field is not present, the concentrator will automatically create a new network address for the associating device.

Notes:

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

4.25 Associated device list (0x42)

Direction: host → module.

Valid when: network is started.

Payload format:

This packet has no payload. It is valid only when the module has started the network as coordinator and can be used by the host to retrieve the list of devices that have been associated so far.

4.25.1 Associated device list response (0xC2)

Direction: host ← module.

Payload format:

Field	Execution status byte	Number of addresses	Addresses
Length	1 Byte	1 Byte	Variable

The “Execution status byte” field indicates whether the list of associated devices could be retrieved correctly (0) or not (anything else). Note that sending the “Associated addresses” command to a node that has started the network as router or end-device will result in an “Execution status byte” equal to 0x02.

The “Number of addresses” field indicates the number of devices that associated so far with the module, and defines the length of the following field.

The “Addresses” field contains a list of concatenated network addresses (2 bytes for each device), so that its length is equal to 2 * “Number of addresses”. Each network address is sent with the most significant byte first. If “Number of addresses” is zero, this field is not present.

4.26 Send data (0x50)

Direction: host → module.

Valid when: network is started.

Payload format:

Field	Options	Custom channel	Custom power	Destination address	Application data
Length	2 Bytes	0/1 Byte	0/1 Byte	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 ↔ b2 - Reserved
b1 – Send with extended source address (i.e., use IEEE address instead of short network address)
b0 (LSb) – Send to extended destination address (i.e., use IEEE address instead of short network address)

The "Custom channel" field is only present if the 15th bit in the "Options" field is set. It indicates on which channel this packet must be sent. On 2.4 GHz modules, the IEEE 802.15.4 standard supports channels from 11 to 26. On sub-GHz modules, the IEEE 802.15.4 standard supports channels from 0 to 10.

The "Custom power" field is only present if the 14th bit in the "Options" field is set. It indicates which output power to use when transmitting this specific packet. The power is expressed in dBm in a signed integer as described in the "Output power" command description. Note that such custom power setting overrides the currently selected output power only temporarily for the transmission of this specific packet.

The "Destination address" field is the 16th 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 "Application data" field contains the data bytes to be delivered to the destination device. Its maximum length is 116 bytes. EBI-802.15.4 will deliver these bytes "as is" to the recipient device; if the radio module of the destination device is running EBI-802.15.4, it will trigger a "Received data notification" on its side if it is in the communication range.

Notes:

The over-the-air transmission will always be done requesting an acknowledge from the destination device unless the destination address is the broadcast address (0xFFFF).

4.26.1 Send data response (0xD0)

Direction: host ← module.

Payload format:

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

The “Execution status byte” field indicates if the data transmission was successful (0x00) or no; the values 0x02 (generic error) or 0x03 (timeout) are commonly returned when the data could not be delivered to the destination device.

The “Retries” field indicates the number of transmission retries (after first primary attempt) carried out while attempting to deliver data to the destination device. Usually on IEEE 802.15.4 networks up to 3 attempts are done before giving up.

The “RSSI of the acknowledge” field (signed integer) provides the RSSI level (in dBm) of the acknowledge received from the destination device. This field is valid only if the “Execution status byte” indicates success and the packet was not transmitted in broadcast; in other cases this field is set to zero. In case the RSSI of the acknowledge could not be retrieved, this field is set to zero.

4.27 Received data (0x60)

This packet should not be sent by the host to the module (if sent, it is ignored).

4.27.1 Received data notification (0xE0)

Direction: host ← module.

Valid when: network is started.

Payload format:

Field	Options	RSSI	Source network identifier	Destination network identifier	Source address	Destination address	Application 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 “Application data” field contains the payload sent by the originating device.

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).

4.28 Enter bootloader (0x70)

Direction: host → module.

Valid when: network is stopped; module is not running the bootloader.

Payload format:

The packet has no payload

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

Note:

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.

The bootloader, once started, always uses the default UART configuration (9600 baud, 8,N,1 no flow control).

For further information refers to [2].

4.28.1 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.

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.