Embit Binary Interface

-

WMBus Specific Documentation

embit s.r.l.

Document information

Versions & Revisions

Revision	Date	Author	Comments
1.0	22/01/2010	A. Sala	Initial version
1.1	22/01/2010	C. Biagi	Minor edits
1.2	14/12/2012	C. Biagi	Minor edits
1.3	14/03/2013	F. Montorsi	Added hyperlinks and references.
1.4	26/04/2013	C. Biagi	Updated channel list
1.5	16/05/2013	C. Biagi	Removed sections on unsupported sections
1.6	05/06/2013	F. Montorsi	Added notes for send command
1.7	19/06/2013	C. Biagi	Added sample appendix added Network Security
1.8	27/08/2013	C. Biagi	Added Network preferences command
1.9	14/03/2014	C. Biagi	Fixed typo in channel table, updated usage example
2.0	17/04/2014	F. Montorsi	Moved the usage example on top, to align it to other EBI reference manuals; moved EBI Bootloader commands to the EBI Bootloader guide
2.1	08/05/2014	C. Biagi	Reference to bootloader guide; updated quick start guide; update energy save mode description
2.2	30/05/2014	C. Biagi	updated quick start guide

References

Ref	Version	Date	Author	Title
1	Rev. 2.0	2014	Embit	Embit Binary Interface Overview
2	Rev. 1.4	2014	Embit	Embit Bootloader Guide
3	EN 13757-4	2013	CEN	Communication systems for meters and remote reading of meters — Part 4: Wireless meter readout (Radio meter reading for operation in SRD bands)
4	prEN 13757-5	2013	CEN	Communication systems for meters and remote reading of meters - Part 5: Wireless relaying

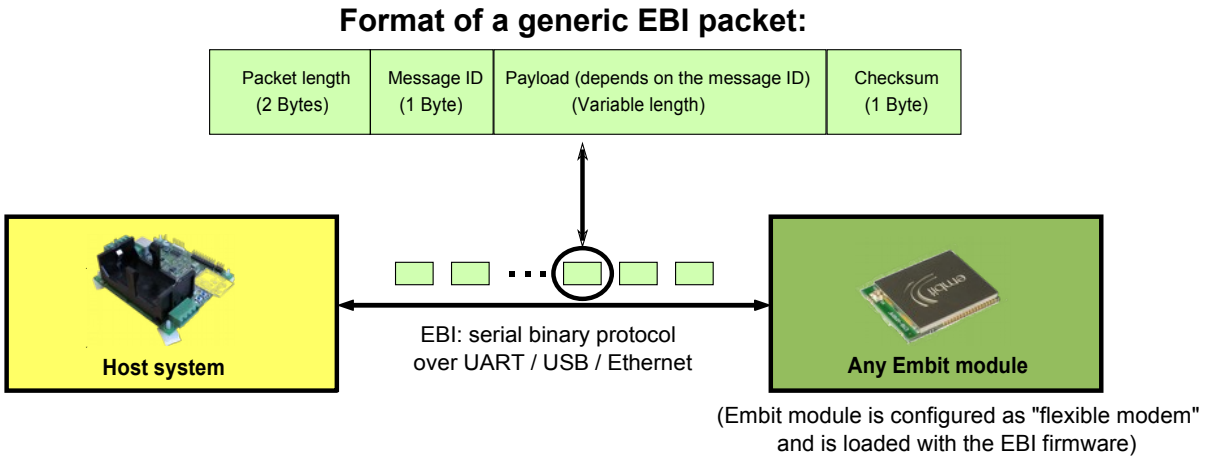
Index

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 Wireless M-Bus over-the-air protocol [3]. This document is intended as a reference manual.

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

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-WMBus is detailed step by step, and is useful to new users.

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

2 EBI WMBus Network Structure

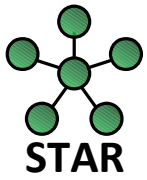
This chapter provides some information on the type of network topologies supported by EBI-WMBus, how to exchange data in such networks and how Wireless M-Bus concepts are mapped in EBI-WMBus.

2.1 Network Topologies Supported

EBI-WMBus supports the same network topologies supported by Wireless M-Bus, that is:

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

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



Note that EBI-WMBus does not support *routing* of packets: just as in Wireless M-Bus networks the communication is from the concentrator to end-devices or viceversa.
Generally speaking, multi-hop communications are not supported although it is important to point out a special feature of the WMBus protocol: exploiting a specific bit field of WMBus packets, it is possible to implement a simple **single-hop repeater** [4]. Although EBI-WMBus does not explicitly accounts for the presence of such type of repeaters, they can be used in EBI-WMBus networks to build tree networks with 2 hierarchy levels:

Please note that repeater functionalities are out of the scope of EBI-WMBus. Please contact Embit for further information.

2.2 Low Power Aspects

In EBI-WMBus it is required that the concentrator of the network never turns off its radio, so that it is always ready to receive/send data from/to end-devices. Indeed the concentrator of a Wireless M-Bus network is typically 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 meters are supposed to “sleep” (i.e., disable their radio) for most of the time and then wake-up at periodic intervals to send a radio packet to the concentrator of the network; then if the concentrator has data queued for the meter it will transmit such data. Note that such low-power policy entails that:

1. The concentrator is able to transmit data to the meter only during a short time period after the reception of a radio packet from the meter. The response to the meter must employ the “send data” command timing option “*0x04 = Send packet after Response delay since last reception (Other device specific)*” in order to synchronize correctly to the receive window of the meter. This may entail a latency in concentrator → meter data delivery of an amount of time depending on the meter transmission period.
2. When the host MCU on the meter-side sends data over UART to its Embit module for over-the-air delivery to the concentrator, such data will be immediately delivered to (and possibly received by) the concentrator, which is always on. For this reason, there is no latency in the meter → concentrator data delivery.

To better understand low-power aspects in EBI-WMBus it is strongly suggested to read the low-power policy of the Wireless M-Bus standard EN 13757-4:2013 [3].

2.3 Wireless M-Bus Compliance

EBI-WMBus creates an EN 13757-4:2013 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 Wireless M-Bus concepts are mapped in EBI-WMBus:

EBI-WMBus term	Corresponding Wireless M-Bus term
Physical address	Manufacturer meter address
Network identifier	Soft address
Concentrator	Other device
Router	-
End device	Meter

Notes about Wireless M-Bus addresses

The first and second byte of the Wireless M-Bus “address field” compose the “M-field”, that shall contain a unique User/Manufacturer ID of the sender. The 15 least significant bits of these two bytes shall be formed from a three letter ISO 646 code (A…Z) as specified in Clause 5.6 of EN 13757-3:2012. The unique User/Manufacturer code is presently administered by:

Flag Association Ltd
http://dlms.com/organization/flagmanufacturesids/index.html

The most significant bit of the M-field indicates:

- If it is zero, then the address A shall be a unique (hard coded) manufacturer meter address of 6 bytes.
- If it is different from zero, then the 6 byte address shall be unique at least within the maximum transmission range of the system (soft address).

Notes about WMBus-protocol features automatically handled by EBI-WMBus

To simplify usage of EBI-WMBus, a certain number of features are automatically handled:

1. Automatic timing management: EBI-WMBus ensures that the correct receiving windows, response delays and FAC timings are respected;
2. Automatic L-field computation: EBI-WMBus automatically prepends the Wireless M-Bus L-field to the bytes sent over UART with the “send data” command;
3. Automatic CRC computation & insertion upon transmission: Wireless M-Bus requires several CRC fields to be inserted in over-the-air packets; EBI-WMBus transparently computes and inserts such CRC fields;
4. Automatic CRC check & filtering upon reception: EBI-WMBus transparently checks the received data against the CRC fields and transparently removes them; note that in case of reception errors no “received data notifications” are generated.

3 EBI WMBus Usage Example

This chapter provides a guide useful to get started with EBI for Wireless M-Bus. 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 aim of this section is only to provide a very first successful experience when communicating wireless with EBI for WMBus.

3.1 Getting Started

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

1. two EMB-EVB boards (mounting EBI WMBus modules, like the EMB-WMB868);
2. the Embit EBI WMBus serial tester software, which is shipped in Embit evaluation kit's disks as `ebi-wmbus-serial-tester.exe`;
3. a PC connected to the two EMB-EVB boards (through USB cables) and running the Embit EBI WMBus 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-WMBus 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-WMBus 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.

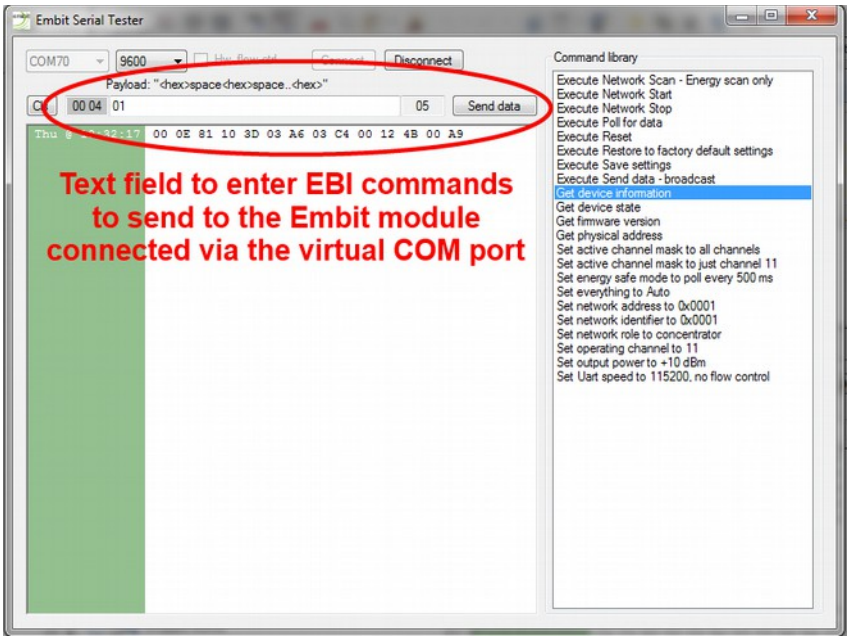
Please refer to the EBI documentation [1] while reading this document for further details on the format of the commands sent and the notifications received.

To get started, just connect the two EMB-EVB boards to a PC and open two instances of Embit EBI-WMBus 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:



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 (40 for Wireless M-Bus). 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-WMBus serial tester software (default 9600 baud) and the hardware flow control (typically disabled).

3.2 Simple Operation

3.2.1 Start the radio section

Start the network (initialize the radio section):

"31 Network Start"
31

The devices will reply with a command formatted as follows:

Execution Status Byte: Success
00 05 B1 00 B6

At this point, the board start reception, according to the default energy save mode (continuous reception). Repeat this command on the second board.

3.2.2 Send data

Send a data packet "Execute Send Data – Immediately":

"50 Send data – Immediately"
50 00 00 D0 D1 D2 D3

The field "50 00 00" indicates "no options" (default value); "D0 D1 D2 D3" is the data payload.

The device replies with:

Execution Status Byte: Success
00 05 D0 00 D5

The other (second) board device receive the packet, and send an asynchronous notification:

Received Data Notification
00 19 E0 80 0F EC 00 95 F5 0E 0D 44 00 00 00 00 00 00 00 00 D0 D1 D2 D3 A3

Note that the sending address is "00 00 00 00 00 00 00 00" because no address was previously set. The last bytes are the payload "D0 D1 D2 D3".

3.2.3 Configure the network address

To complete the basic configuration, a Wireless M-Bus address should be set.

Send the command “Set the network address to 80 00 A1 A2 A3 A4 A5 A6”:

“21 Network address - Set 8000A1A2A3A4A5A6”
21 80 00 A1 A2 A3 A4 A5 A6

The device replies with:

Execution Status Byte: Success
00 05 A1 00 D5

At this point it is possible to send data, and the correct source address will be automatically inserted:

“50 Send data - Immediately”
50 00 00 D0 D1 D2 D3

The device replies with:

Execution Status Byte: Success
00 05 D0 00 D5

The other device receive the packet, and send an EBI asynchronous notification:

Received Data Notification
E0 80 0F EC 03 63 85 8C 0D 44 80 00 A1 A2 A3 A4 A5 A6 D0 D1 D2 D3

Please note that the notification includes the fields:

- “EC” = RSSI level 20 dBm
- “03 63 85 8C” = timestamp od received packet
- “0D” = L-field
- “44” = C-field
- “80 00 A1 A2 A3 A4 A5 A6” = Address
- “D0 D1 D2 D3” = Payload

3.3 Advanced Operation

In the real application it is useful to save to non volatile memory the module configuration, in order to avoid the need of send the configuration each boot. The other advantage is that, if a power loss occur, the module reboots immediately reloading the right configuration. A sample session showing how to use the configuration save command follows.

Configure the module with the Wireless M-Bus parameters:

- Address “80 00 A1 A2 A3 A4 A5 A6”
- Role “other device”
- Radio channel “1” (169.406250 MHz)
- Output power +27 dBm
- Energy save mode: receive window after each transmission

“21 Network address - Set 8000A1A2A3A4A5A6”
21 80 00 A1 A2 A3 A4 A5 A6

“23 Network role - Set meter”
23 00

"11 Operating channel – Set 01d"
11 01

"10 Output power – Set +27 dBm"
10 1B

Set energy save behavior:

"13 Energy save mode – Set TRX wnd, MCU off"
13 03 01

At boot, start network automatically so there is no need to send "Start network" each boot:

"24 Network automated settings – Set network start"
24 80 00

Save all the settings into the non volatile memory:

"08 Save settings"
08

At this point it is possible to communicate with :

"50 Send data - Immediately"
50 00 00 D0 D1 D2 D3

After a reset, the module restarts automatically and it is able to send data immediately:

"05 Reset"
05

"50 Send data - Immediately"
50 00 00 D0 D1 D2 D3

The "Advanced operation" session above completes 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 WMBus 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-WMBus 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-WMBus 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	UID
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 “UID” field identifies the EMB-WMBx module.

Notes:

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

4.2 Device state (0x04)

Direction: host → module.

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 radio 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 (see associated command).

The “End of receiving window” event is only sent if energy save mode requires so.

4.3 Reset (0x05)

Direction: host → module.

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.

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.

Payload format: no payload.

Notes:

The default settings are the following:

Parameter	Default value
Operating Channel	1 (169 MHz) 13 (868 MHz)
Transmission Power	+15 dBm
Serial Port Settings	9600 baud, 8, N, 1, no flow control
Physical Address [hex]	UID
Network Address [hex]	00 00 00 00 00 00 00 00
Energy Save Mode [hex]	00 00 (Module and radio always on)
Auto Parameters [hex]	0 (none)
C field [hex]	44

Note:

The default physical address (UID) is not an EN 13757-4:2013 compliant manufacturer address. The user shall assign a valid address in production, using the right 3 letter code as specified in section “Wireless M-Bus compliance”.

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.

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.

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-WMBx
0x00 = Maintain current speed	V
0x01 = 1200 baud/sec.	V
0x02 = 2400 baud/sec.	V
0x03 = 4800 baud/sec.	V
0x04 = 9600 baud/sec.	V
0x05 = 19200 baud/sec.	V
0x06 = 38400 baud/sec.	V
0x07 = 57600 baud/sec.	V
0x08 = 115200 baud/sec.	V
0x09 = 230400 baud/sec.	
0x0A = 460800 baud/sec.	
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)

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:

Using high baudrates can introduce errors on UART communications, 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 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.

Payload format:

To get 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-WMB169PA	[-7, +27]
EMB-WMB868	[-7, +15]

Note:

Check local regulation compliance before using. Actual Rf output power may be different from the setting (it could be affected by supply voltage and module temperature).

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.

Payload format:

To get 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 according to the following table:

Channel number	WMBus mode	WMBus ch. name	Center Freq. [MHz]	Modulation	Data Rate [kbps]
1 (0x01)	N	N1a, N2a (CEPT 1a)	169,40625	GFSK	4,8
2 (0x02)	N	N1b, N2b (CEPT 1b)	169,41875	GFSK	4,8
3 (0x03)	N	N1c, N2c (CEPT 2a)	169,43125	GFSK	2,4
4 (0x04)	N	N1d, N2d (CEPT 2b)	169,44375	GFSK	2,4
5 (0x05)	N	N1e, N2e (CEPT 3a)	169,45625	GFSK	4,8
6 (0x06)	N	N1f, N2f (CEPT 3b)	169,46875	GFSK	4,8
7 (0x07)	N	N2g (CEPT 0)	169,43750	4-GFSK	19,2
13 (0x0D)	reserved	-	868,030	FSK	4,8
14 (0x0E)	reserved	-	868,090	FSK	4,8
15 (0x0F)	reserved	-	868,150	FSK	4,8
16 (0x10)	reserved	-	868,210	FSK	4,8
17 (0x11)	reserved	-	868,270	FSK	4,8
18 (0x12)	reserved	-	868,330	FSK	4,8
19 (0x13)	reserved	-	868,390	FSK	4,8
20 (0x14)	reserved	-	868,450	FSK	4,8
21 (0x15)	reserved	-	868,510	FSK	4,8
22 (0x16)	reserved	-	868,570	FSK	4,8
23 (0x17)	S (short preamble)	S	868,300	FSK	16,384
24 (0x18)	S (long preamble)	S	868,300	FSK	16,384
25 (0x19)	T	Meter channel	868,950	FSK	66,666

26 (0x1A)	T	Other channel	868,300	FSK	16,384
27 (0x1B)	R	R2 Ch n°0	868,030	FSK	2,4
28 (0x1C)	R	R2 Ch n°1	868,090	FSK	2,4
29 (0x1D)	R	R2 Ch n°2	868,150	FSK	2,4
30 (0x1E)	R	R2 Ch n°3	868,210	FSK	2,4
31 (0x1F)	R	R2 Ch n°4	868,270	FSK	2,4
32 (0x20)	R	R2 Ch n°5	868,330	FSK	2,4
33 (0x21)	R	R2 Ch n°6	868,390	FSK	2,4
34 (0x22)	R	R2 Ch n°7	868,450	FSK	2,4
35 (0x23)	R	R2 Ch n°8	868,510	FSK	2,4
36 (0x24)	R	R2 Ch n°9	868,570	FSK	2,4
37 (0x25)	C	Meter channel	868,950	FSK	100
38 (0x26)	C	Other channel	869,525	GFSK	50

Notes:

The operating channel can be changed at any time, except during the transmission.

Important:

Before starting using this product, please carefully check regulations regarding limits that apply. For every radio channel, power, duty cycle, and application specific restriction are to be respected. Check “ERC Recommendation 70-03” for further information.

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 Energy save (0x13)

Direction: host → module.

Payload format:

To get 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	Reserved
Length	1 Byte	1 Byte	-

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-WMBx
0x00 =	Always on	V
0x01 =	Always off	V
0x02 =	Receive window after transmission (whose duration is defined by WMBUS standard [3]); a notification (received data notification 0xE0) is generated if a packet is received during this receive window.	V
0x03 =	Receive window after transmission (whose duration is defined by WMBUS standard [3]); a notification will be generated if a packet is received (just like mode 0x02); however, even if no packet is received, a notification (device state notification, 0x84, with code 0x51) is generated to indicate the end of the receiving window.	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-WMBx
0x00 =	Always on (UART interface always enabled)	V
0x01 =	Always off (UART modem hardware flow control needed)	V

The “Host interface (MCU) policy” “always on” keep the reception on the UART interface always possible. Using “always off”, the communication is possible only when the RTS

signal is asserted (it acts like a wakeup signal). This behavior permits the minimum consumption of the module.

Notes:

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.

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).

4.10.1 Energy save response (0x93)

Direction: host ← module.

Payload format:

If getting the current energy save options, the payload is composed by two bytes as described in the request.

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

4.11 Physical address (0x20)

Direction: host → module.

Payload format:

To get 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 indicating the physical address to be used (any value accepted). This parameter must be initialized according to the EN-13757 format.

Notes:

The physical address is to be sent most significant byte first.

The module doesn't feature a hardcoded physical address and so, this field must be set by the application before using it. The default physical address (UID) is not an EN 13757-4:2013 compliant manufacturer address. The user shall assign a valid address in production, using the right 3 letter code as specified in section “Wireless M-Bus compliance”.

4.11.1 Physical address response (0xA0)

Direction: host ← module.

Payload format:

If getting the physical address, the payload is an 8 bytes field 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].

4.12 Network address (0x21)

Direction: host → module.

Payload format:

To get the network address the packet is sent with an empty payload.

To set the network address to be used, the payload is an 8 bytes field indicating the network address to be used.

This parameter must be initialized according to the EN-13757 format (manufacturer and address fields).

4.12.1 Network address response (0xA1)

Direction: host ← module.

Payload format:

If getting the network address, the payload is an 8 bytes field indicating the network address in use.

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

4.13 Network role (0x23)

Direction: host → module.

Payload format:

To get 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 = Meter
0x01 = Other device

4.13.1 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].

4.14 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	Reserved
Length	1 Bit	15 Bit

"Auto network creation": if set, the module will invoke a "Network start command" at startup right after loading the data from non volatile memory.

4.14.1 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].

4.15 Network preferences (0x25)

Direction: host → module.

Payload format:

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

Field	C-field value	Reserved
Length	1 Byte	Variable (0 Byte)

"C-field value": This is the value of the C-field, when it is automatically inserted in the packet by the "Send data" command.

4.15.1 Network preferences response (0xA5)

Direction: host ← module.

Payload format:

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

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

4.16 Network security (0x26)

Direction: host → module.

Payload format:

For security reasons, the security settings cannot be read. For setting the security settings, the payload is formatted as follows:

Field	Options	Encrypted payload Offset	Network key
Length	1 Byte	0/1 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 – Use AES-CTR encryption
b4 ↔ 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. In EMB-WMBx modules only pre-shared AES-CTR encryption is supported (Option field equal to 0xE1).

The "offset" field specifies the address of the start of the encrypted part of the payload. If it is set to zero, all the payload is encrypted.

Notes:

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

4.16.1 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.

4.17 Network stop (0x30)

Direction: host → module.

Payload format:

The packet has no payload

4.17.1 Network stop response (0xB0)

Direction: host ← module.

Payload format:

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

4.18 Network start (0x31)

Direction: host → module.

Payload format:

The packet has no payload.

4.18.1 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.

4.19 Send data (0x50)

Direction: host → module.

Payload format:

Field	Options	Custom channel	Custom power	Timing options	L field	C field	Address field	Data (CI field is first byte)
Length	2 Bytes	0/1 Byte	0/1 Byte	0/1 Byte	0/1 Byte	0/1 Byte	0/8 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 using physical address instead of network address
b12 ↔ b7 - Reserved
b6 - Use “Slow Response Delay” instead of the default “Fast Response Delay”
b5 - Use “Frame Format B” (as specified in EN13757-4)
b4 – Postponed transmission
b3 – Reserved
b2 – L field included
b1 – C field included
b0 (LSb) – Address field included

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. Depending on the Wireless M-Bus mode, different channels can be selected (see “Set Channel” command).

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 “Timing options” field specifies the settings for a postponing the packet transmission. The field is only present if bit 4 of the options field is set. The accepted values are as follows:

0x00 = Send packet immediately.
0x01 = Send packet at next UART input activity.
0x02 = Send packet after tFAC since the previous transmission (Meter specific).
0x03 = Reserved
0x04 = Send packet after Response delay since last reception (Other device specific).

The “L field” is only present if the bit 2 in the “options” field is set. It is the L field that will be used on this packet (if bit 3 in “options” is zero, the L field will be automatically calculated).

The “C field” is the C field to use with this packet. If not included, the default C field will be used.

The “Address” field is the 64 bit address (most significant byte transmitted first) to be inserted in the WMBus packet. Any address (of the *sender*) can be used. Note that the

WMBus does not explicitly support destination addresses and this field, if present, should be reserved for the *sender* address. If the field is not present, the module will use one of its previously set addresses (physical address or network address) depending on the status of bit 13 of the “options” field.

The “Data” field contains the data to be sent (starting from the CI field). The maximum field length is 246 Bytes for *Frame A* and 242 Bytes for *Frame B*.

***Important*:**

Before starting using this function, please carefully check regulations regarding limits that apply. For every radio channel, power, duty cycle, and application specific restriction shall be respected. Check “ERC Recommendation 70-03” for further information.

4.19.1 Send data response (0xD0)

Direction: host ← module.

Payload format:

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

4.20 Received data (0x60)

This packet should not be sent and will be ignored by the module.

4.20.1 Received data notification (0xE0)

Direction: host ← module.

Payload format:

Field	Options	RSSI	Timestamp	L field	C field	Address	Data (CI field is first byte)
Length	2 Bytes	0/1 Byte	0/4 Bytes	0/1 Byte	0/1 Bytes	0/8 Bytes	Variable

The “Options” field indicates to the host which fields are present in the packet and a few other properties of the radio message received (e.g., frame format type):

b15 (MSb) – RSSI attached
b14 ↔ b5 - Reserved
b4 – Frame Format B detected (as specified in EN13757-4)
b3 – Timestamp attached
b2 – L field attached
b1 - C field attached
b0 (LSB) – Address attached

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 “Timestamp” field indicates the time instant during which the first bit of the radio packet has been demodulated and “received”. It is expressed in 1/32768th of second, starting from zero since the module was powered up and the boot has finished.

The “L field” field is only present if the bit 3 in the “Options” field is set. It is the L field of the WMBus packet (not included as default).

The “C field” field is only present if the bit 2 in the “Options” field is set. It is the C field of the WMBus packet.

The “Address” field is the 64 bits address (most significant byte transmitted first) of the sender as specified in the WMBus standard.

The “Transparent header length” field indicates the length of the following field (“Transparent header”) and is only present if the bit 0 in the “Options” field is set.

The “Transparent header” field includes the bytes of the header as received from the radio. This field is only present if the bit 0 in the “Options” field is set and is used for those modes that have an atypical frame header (multiple address fields, etc).

The “Data” field contains the received data (starting from the CI field). For more information on the data field, please consult the EN-13757 specification.

4.21 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.

For further information refers to [2].

4.21.1 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.