eZeeNet™ Software 1.6

SerialNet™ Reference Manual

AT-Command Set

Doc. P-EZN-452~01 v.1.9 March 2007

Disclaimer

MeshNetics believes that all information is correct and accurate at the time of issue. MeshNetics reserves the right to make changes to this product without prior notice. Please visit MeshNetics website for the latest available version.

MeshNetics does not assume any responsibility for the use of the described product or convey any license under its patent rights.

MeshNetics warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with MeshNetics standard warranty. Testing and other quality control techniques are used to the extent MeshNetics deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

Trademarks

MeshNetics®, ZigBit, eZeeNet, ZigBeeNet, SensiLink, as well as MeshNetics and ZigBit logos are trademarks of MeshNetics Ltd.

All other product names, trade names, trademarks, logos or service names are the property of their respective owners.

Technical Support

Please e-mail your product-related questions to support@meshnetics.com. If you like to speak to one of our product specialists you can call the phone number listed below.

Development Support

Software customization services can be provided on terms and conditions mutually agreed by MeshNetics and end-user.

Contact Information

MeshNetics

9 Dmitrovskoye Shosse, Moscow 127434, Russia

Tel: +7 (495) 725 8125

Office hours: 8:00am – 5:00pm (Central European Time)

Fax: +7 (495) 725 8116

E-mail: support@meshnetics.com

www.meshnetics.com

Table of Contents

List of Figures

List of Tables

1. Introduction

SerialNet is software bundled with the ZigBit Evaluation/Development Kit (ZEK/ZDK), a solution from MeshNetics that helps in deployment of Wireless Sensor Networks (WSN). ZEK/ZDK is based on the ultra-compact low-power high sensitivity ZigBit OEM module [8] and eZeeNet software [9], which contains 802.15.4 MAC and ZigBee NWK layers enabling wireless network connectivity with a simplified programming interface.

SerialNet offers control over the most of ZigBit functionality through any communication interface using a standardized AT-command set (Hayes-like command set).

The SerialNet allows user application to easily extend the set of supported functions by adding extra S-registers or AT-commands. This service gives unique capability of over-the-air remote control without writing any special user-defined code. It also enables commissioning procedures, and makes debugging and testing easier. This technology enables wireless module configuration during OEM mass-production process, thus providing flexible commissioning mechanism for installation and maintenance of ZigBit-based devices, simplifying maintenance & network monitoring at the same time.

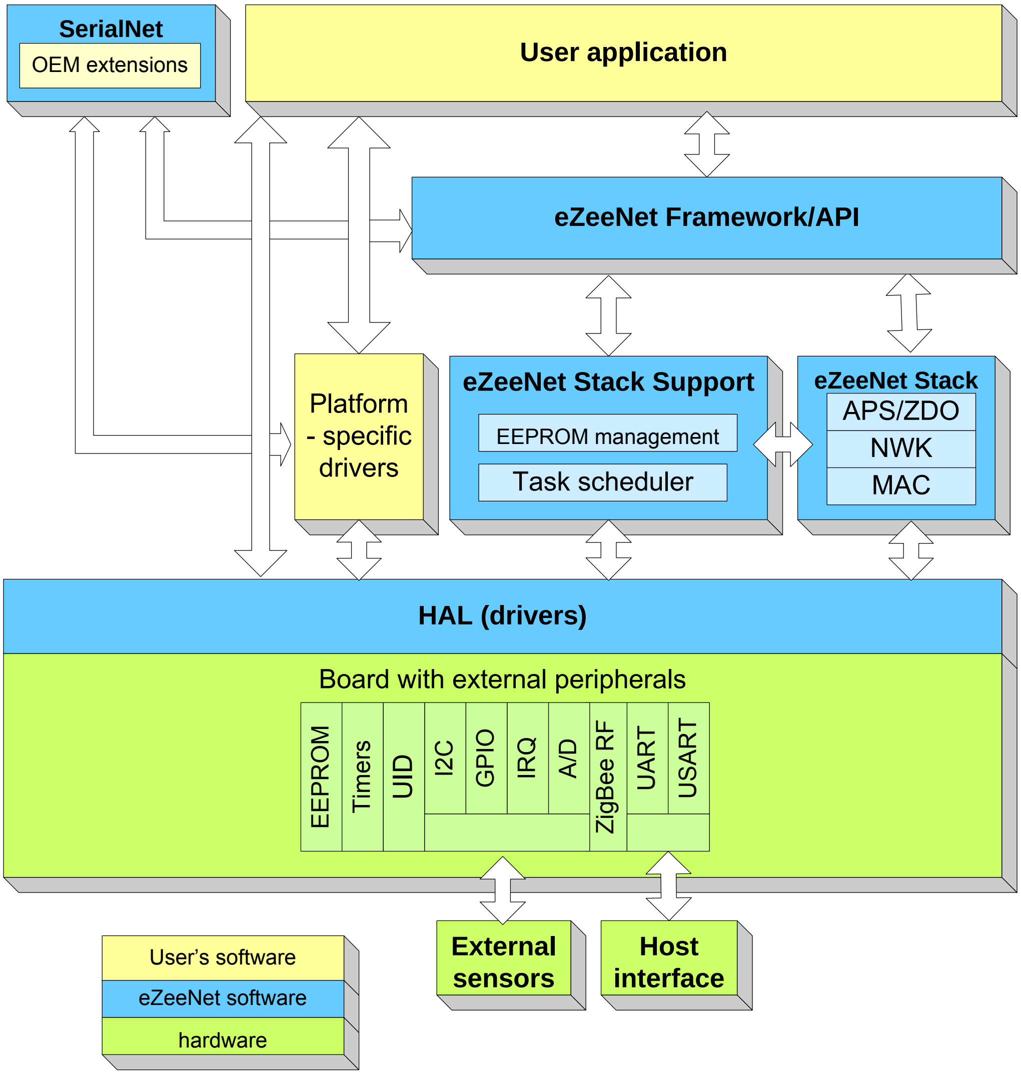

Figure 1. Simplified diagram of the eZeeNet software

SerialNet running on the ZigBit OEM Module provides the following advantages:

- ZigBit module can be connected to the host as communication processor; furthermore, host may use ZigBit spare HW interfaces to connect extra sensors
- user application may use a simpler S-register mapping, instead of the event-driven API programming
- OEM user extensions can easily increase the module functionality
- ZigBit module and user's parameters can be easily accessed over-the-air without specifically dedicated protocol thus opening the way to network management and further upgrades

The document presents the description of the SerialNet AT-Command language. The command set bases on wireless extensions of V.250 command set [3]. The command set includes 52 commands and more than 40 S-registers. It is applicable to eZeeNet Software delivered with the ZDK and ZEK packages.

Related Documents

[1] ZigBee Document 053474r08, February 17, 2006

[2] Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs). IEEE Std 802.15.4™-2003.

[3] Serial asynchronous automatic dialing and control. ITU-T Recommendation V.250, 05/99

[4] International Reference Alphabet (IRA) (Formerly International Alphabet No. 5 or IA5). Information Technology – 7-Bit Coded Character Set for Information Interchange, CCIT Recommendation T.50, 09/92.

[5] Procedure for the Allocation of CCITT Defined Codes for Non-Standard Facilities. CCIT Recommendation T.35, 1991.

[6] General Structure of Signals of International Alphabet No. 5. Code for Character Oriented Data Transmission over Public Telephone Networks. ITU-T Recommendation V.4

[7] IEEE Std 802.15.4-2003 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 (LR-WPANs)

[8] ZigBit™ OEM Module. Product Datasheet. MeshNetics Doc. M-251~01

[9] eZeeNet™ IEEE802.15.4/ZigBee Software. Product Datasheet. MeshNetics Doc. M-251~02

[10] 8-bit AVR Microcontroller with 64K/128K/256K Bytes In-System Programmable Flash ATmega 640/V, ATmega 1280/V, ATmega 1281/V, ATmega 2560/V, ATmega 2561/V. www.atmel.com

Abbreviations and Acronyms

ARQ	Automatic Repeat-reQuest
ASCII	American Standard Code for Information Interchange
BS	Backspace character
CCITT	Consultative Committee on International Telephony and Telegraphy.
CR	Carriage Return
CRE	Coordinator/Router/End-Device (meaning any of those)
CTS	Clear To Send
DCE	Data Communication Equipment,
DTR	Data Terminal Ready
EEPROM	Electrically Erasable Programmable Read Only Memory
GPIO	General Purpose Input/Output
I2C or I^2C	Inter-Integrated Circuit, pronounced I-squared-C
ID	Identifier
IEEE	Institute of Electrical and Electronics Engineers
ITU	International Telecommunications Union
LED	Light Emitting Diode
LF	Line Feed character
LQI	Link Quality Indicator
LSB	Least Significant Bit
MAC	Medium Access Control (Sublayer)
MCU	MultiController Unit/Multi-Chip Unit
NWK	Network layer
OEM	Original Equipment Manufacturer
PAN	Personal Area Network
PHY	PHYsical Layer
R	Read-only parameter
RSSI	Received Signal Strength Indicator
RTS	Request To Send
RW	Read-write parameter
RX	Receiver
TBD	To Be Defined
TX	Transmitter
UART	Universal Asynchronous Receiver Transmitter
USART	Universal Synchronous/Asynchronous Receiver/Transmitter
ZDO	ZigBee Device Object

2. AT-Commands

2.1. Conventions

To be distinguished from the rest, the definitions of commands directed to the module are denoted in `Courier` while the module responses are given in **`Bold Courier`** font. Angle brackets enclose the mandatory parameters. Square brackets contain optional parameters.

2.2. Overview

The AT-Command Protocol is widely used in communications between variable equipment. It is utilized in multiple applications due to simplicity, text parameter representation, automatic rate adjustment for COM port, an easy mechanism for self-recovery in case of error and due to its inherent flexibility.

The term *module* will be used throughout the document implying the ZigBit module [8] controlled by a *host* equipment (PC) using AT-commands. When necessary the term *node* will be used in reference to the module's role in the network (End-Device, Router or Coordinator).

The Protocol implements the following principles. The host sends commands to the module, which replies with text messages (*information responses*), and each of the messages is terminated by a result code (which is mostly **`OK`** or **`ERROR`**). Each command is prefixed by the AT string followed by the chained commands to be executed consecutively. In case of any command executed incorrectly, the command sequence is interrupted and the **`ERROR`** result code is returned. *Information responses* for any command are returned in an easily recognizable string format. Each command in a sequence may be of different syntax, depending on if it is used to execute an action, to read or to write parameter(s) or it is used to test valid parameter range. There is the *standard set* of commands (for instance `E, V, Z` etc.), but it is extended for the majority of commands. According to the V.250 wireless protocol standard recommendations, the *extension commands* are prefixed by the `+W` characters.

Illustrating example:

	Command/Response	Comment
Command to module	`ATE1V1+WTXPWR=-4+WLQI2+WRSSI2S104?`	Turn echo on (`E1`), enable verbose response, set TX level to -4 dBm, request for LQI and RSSI for link with node 2, request for AD4 pin
Information responses	**`+WLQI:254`**	LQI value is 254
	`+WRSSI:-80`	RSSI is -80 dBm
	`125`	Analog voltage on AD4 pin corresponds to the 125 code
Result code	**`OK`**	Execution is completed successfully

The important feature of AT-command set is the capability to request execution of particular function over the air via `ATR` command (see 3.9.2). This allows to transfer the AT-command to the remote node, run execution and redirect the execution output to the originator. Thus, the remote node can be monitored, commissioning can be performed and the corresponding parameters can be set.

The SerialNet AT-Command Protocol will be detailed in the Section 3.1, following closely to the V.250 recommendation adapted specifically for wireless networks.

Sections 2.3, 2.4, and 2.5 present all the referencing information on the Protocol implementation. Quick overview below will help you navigate this document easier.

Section 2.3 summarizes basic specifications of AT-commands grouped into functional categories. These specifications include:

- Node type applicable to a command
- S-register corresponding to a command (if any)
- Command syntax forms applicable
- A command name itself
- Availability of the command in different software packages
- A reference to the clause with detailed description

Section 2.4 explains both verbose and numeric forms of the result codes with the corresponding parameter(s), if any.

Section 2.5 is a functional representation of S-registers with the corresponding commands.

Each command is defined in Sections 3.2 – 3.9 with explanation of the following descriptors:

- A command syntax
- S-register corresponding (if any) and its read/write attribute
- Result codes
- Example
- Default value
- Persistence (settings are mostly stored in EEPROM)
- Node type to which the command is applicable
- Products supporting the command.

Section 2.6 contains more complex examples that can be run on the Evaluation or Development Kits.

2.3. Command Summary

Implemented in SerialNet, the AT-commands fall into the following categories:

- Network, node management and networking parameters
- Data transmission
- Generic control
- Host interface control
- Hardware control
- Remote management.

The first four of the above categories simply map most of the API functions, as well as add some functions for easier software and hardware identification. There are extra commands that allow rebooting the module safely or reloading factory default parameters.

Hardware control functions allow to configure/read/write GPIO pins, to read A/D and to manage other peripherals. That permits extra sensor interfaces for host MCU, if needed. CTS line management included in SerialNet simplifies power management of the external peripherals or the host processor because this circuit becomes high while the ZigBit module is entering the sleeping state.

A user can extend the functionality of SerialNet commands with the OEM extensions that map the parameters to additional S-registers. Furthermore, the OEM extensions can implement specific commands to initiate sophisticated processing for sensor readings, data exchange and so on.

Remote management functions include the password-protected AT-commands that come from originating node to a target node. The received AT-command sequences are executed on the destination node, as if they would come from a serial port. The execution results are sent back to the originating node in the form as if they are returned from UART, thus enabling conventional processing of AT-commands and responses by the host processor. User's OEM extensions are accessible through remote execution service as well. Remote execution service is protected by 32-bit password that can be set over-the-air during the node installation or manufacturing.

Remote management function is an important tool that allows to organize commissioning procedures on PC, using commercial off-the-shelf terminal software. The Evaluation/Development Kit can be used as a hardware platform to connect ZigBee network to PC.

Table 1. Command Summary

Function	Node type (C/R/E)	S-register	Action syntax	Parameter set syntax	Parameter read syntax	Parameter test syntax	Command	Persistence	Reference
Network management									
PAN ID	CRE	20, 21		x	x	x	`+WPANID`	x	3.2.1
Active channel	CRE	22			x		`+WCHAN`		3.2.2
Channel mask	CRE	23		x	x	x	`+WCHMASK`	x	3.2.3
Leave the network	CRE		x				`+WLEAVE`		3.2.4
Start/Join to network	CRE		x				`+WJOIN`		3.2.5
Request for networking status	CRE				x		`+WNWK`		3.2.6

Function	Node type (C/R/E)	S-register	Action syntax	Parameter set syntax	Parameter read syntax	Parameter test syntax	Command	Persistence	Reference
Request for parent address	CR				x		`+WPARENT`		3.2.7
Request for children addresses	RE				x		`+WCHILDREN`		3.2.8
Automatic networking	CRE	24		x	x	x	`+WAUTONET`	x	3.2.9
General node management									
Power management	E	31, 32		x	x	x	`+WPWR`	x	3.3.1
Force to sleep	E		x				`+WSLEEP`		3.3.2
Set node role	CRE	33		x	x	x	`+WROLE`	x	3.3.3
Set period for tracking the end-devices	E	37		x	x	x	`+WSYNCPRD`	x	3.3.4
TX power level	CRE	34		x	x	x	`+WTXPWR`	x	3.3.5
Encryption key	CRE		x				`+WSEC`	x	3.3.6
Request for LQI	CRE		x				`+WLQI`		3.3.7
Request for RSSI	CRE		x				`+WRSSI`		3.3.8
Set network addressing mode	CRE	30		x	x		`S30`		3.3.9
Data transmission									
Send data to specific node	CRE		x				`D`		3.4.1
Send broadcast data	CRE		x				`DU`		3.4.2
Send S-register value to specific node	CRE		x				`DS`		3.4.3
Request for buffered data by sleeping device	E		x				`DR`		3.4.4
Ping the node	CRE		x				`+WPING`		3.4.5

Function	Node type (C/R/E)	S-register	Action syntax	Parameter set syntax	Parameter read syntax	Parameter test syntax	Command	Persistence	Reference
Generic control									
Help	CRE		x				`&H`		3.5.1
Display parameters and S-register values	CRE		x				`%H`		3.5.2
Display product identification information	CRE		x				`I, I0`		3.5.3
Request for Manufacturer Identification	CRE		x				`+GMI` or `I1`		3.5.4
Request for Model Identification	CRE		x				`+GMM` or `I2`		3.5.5
Request for hardware/software revision Identification	CRE		x				`+GMR` or `I3`		3.5.6
Read/Write MAC address	CRE			x	x		`+GSN` or `I4`		3.5.7
Warm reset	CRE		x				`Z`		3.5.8
Set to factory-defined configuration	CRE		x				`&F`		3.5.9
Host interface commands									
termination character	CRE	3		x	x		`S3`	x	3.6.1
response formatting character	CRE	4		x	x		`S4`	x	3.6.2
command editing character	CRE	5		x	x		`S5`	x	3.6.3
Command echo	CRE		x				`E`	x	3.6.4
Result code suppression	CRE		x				`Q`	x	3.6.5
Response format	CRE		x				`V`	x	3.6.6

Function	**Node type (C/R/E)**	**S-register**	**Action syntax**	**Parameter set syntax**	**Parameter read syntax**	**Parameter test syntax**	**Command**	**Persistence**	**Reference**
Result code selection	CRE		x				`X`	x	3.6.7
Serial port communication rate	CRE			x	x	x	`+IPR`	x	3.6.8
Serial port flow control	CRE			x	x	x	`+IFC`	x	3.6.9
DTR behavior	CRE	50	x				`&D`	x	3.6.10
Request for the latest result code	CRE	0			x		`S0`		3.6.11
Parameters									
Data delivery timeout	CRE	51		x	x	x	`+WTIMEOUT`	x	3.7.1
Repetition count	CRE	52		x	x	x	`+WRETRY`	x	3.7.2
Data transmission waiting timeout	CRE	53		x	x	x	`+WWAIT`	x	3.7.3
Read/Write logical address	CRE	55		x	x	x	`+WSRC`	x	3.7.4
Hardware control									
GPIO configuration	CRE	120 … 128		x	x		`S120…S128`	x	3.8.1
GPIO	CRE	130 … 138		x	x		`S130…S138`		3.8.2
A/D configuration	CRE	100		x	x		`S100`	x	3.8.3
A/D	CRE	101 … 108		x			`S101…S108`		3.8.4
Remote management									
Set a password	CRE		x				`+WPASSWORD`	x	3.9.1
Remote execution of AT command	CRE		x				`R`		3.9.2

NOTE:

The second column contains the node role. C means coordinator, R – router, E – end-device.

2.3.1. Future extensions

SerialNet will support ZigBeeNet software that will become available soon. It enables AT-commands access to the new features such as:

- USART/I2C
- IRQ support
- Full functional ZigBee-style data delivery
- More parameters of ZDO
- Over-the-air update.

2.4. Result Codes

Result codes appear either in response to a command or, asynchronously, due to the specific events occurred in the network or a module. See detailed description of result codes in 3.1.11.

Table 2. Result Codes

Verbose Code	Numeric Code	Parameters	Parameter Description
OK	0	None	
ERROR	4	None	
DATA	8	**<addr>,<bcast>,<length>:<data>**	addr is a logical address of a source node having sent this data block bcast is set to 1 if data is sent by broadcast transmission, otherwise it is set to zero length is a length of the <data> field data is a byte sequence NOTE: +WPING command (see 3.4.5) results in the following code on the destination node: **DATA <addr>,0,0:**
EVENT	7	**:<text>**	text is a text specifying an event.

Verbose Code	Numeric Code	Parameters	Parameter Description
		`:JOINED`	This event indicates that the node is joined to the network. Also, this event appears when the node is rejoined automatically after orphaning by its parent.
		`:LOST`	This event indicates that the node lost the network connection. Also, this event appears when the node is orphaned by its parent.
		`:CHILD_JOINED <mac_addr>`	This event indicates that some new child has joined. This event can appear on the router or coordinator. This event is not propagated over the whole network and appeared only on the parent of the new child.
		`:CHILD_LOST <mac_addr>`	This event indicates that some child loose a connection with its parent. This event can appear on the router or coordinator. This event is not propagated over the whole network and appeared only on the parent of the loosing child.

2.5. S-registers

S-registers are associated with the networking parameters that are controlled by the corresponding AT-commands.

Table 3. S-Registers

Parameter	Parameter Type (R/RW)	S-register	Command Reference
The latest result code	R	`S0`	3.6.11
Termination character	RW	`S3`	3.6.1
Response formatting character	RW	`S4`	3.6.2
Command editing character	RW	`S5`	3.6.3
PAN ID	RW	`S21, S20`	3.2.1
Active channel	R	`S22`	3.2.2
Channel mask	RW	`S23`	3.2.3
Automatic networking	RW	`S24`	3.2.9

Parameter	Parameter Type (R/RW)	S-register	Command Reference
Network addressing mode	RW	`S30`	3.3.9
Power management	RW	`S31, S32`	3.3.1
Node role	RW	`S33`	3.3.3
TX power level	RW	`S34`	3.3.5
Period for tracking the end-devices	RW	`S37`	3.3.4
DTR behavior	RW	`S50`	3.6.10
Data delivery timeout	RW	`S51`	3.7.1
Repetition count	RW	`S52`	3.7.2
Data transmission waiting timeout	RW	`S53`	3.7.3
Own logical address	RW	`S55`	3.7.4
A/D configuration	RW	`S100`	3.8.3
A/D	R	`S101…S108`	3.8.4
GPIO configuration	RW	`S120…S128`	3.8.1
GPIO	RW	`S130…S138`	3.8.2

2.6. Examples

The examples given below show usage of AT-commands to control the MeshBean2 boards included into the ZigBee Development Kit.

2.6.1. Connection with board

To begin communication with nodes, you have to follow guidelines from the User's Guide document, see ZEK/ZDK User's Guide. In brief, you have to connect the boards to the PC using USB or RS-232 cables, program the nodes with the SerialNet demo (via JTAG, USB or RS-232), run HyperTerminal software from the standard Windows package, select the corresponding COM-port and set the following parameters:

Bits per second:	38400
Data bits:	8
Parity:	none
Stop bits:	1
Flow control:	None

To check the connection, enter `AT` on the terminal window and press `<Enter>`. If the board responds with `OK`, everything is configured properly.

2.6.2. Control of LED and DIP switches

Mapping of I/O pins of the ZigBit module and their functions on the MeshBean2 boards is summarized in the table below.

Table 4. GPIO Pins Summary

Component	I/O pin	Description
LED1	GPIO0	output, 1 means LED on
LED2	GPIO1	output, 1 means LED on
LED3	GPIO2	output, 1 means LED on
SW4:1	GPIO3	input (no pull-up on the board), ON – logical zero
SW4:2	GPIO4	input (no pull-up on the board), ON – logical zero
SW4:3	GPIO5	input (no pull-up on the board), ON – logical zero
	GPIO6	reserved for MeshBean2 sensor interfaces
	GPIO7	reserved for MeshBean2 sensor interfaces
	GPIO8	reserved for MeshBean2 sensor interfaces

Initially, you need to set physically SW1 to OFF, SW2 and SW3 to ON, and configure I/O pins via command:

Command/Response	Comment
`ATS120=3 S121=3 S122=3`	configure GPIO0, GPIO1, GPIO2 for output
`OK`	
`ATS123=1 S124=1 S125=1`	configure GPIO3, GPIO4, GPIO5 for input and turn on internal pull-up
`OK`	

Afterwards, you can turn on LEDs and read DIP-switches:

Command/Response	Comment
`ATS130=1 S131=0 S132=1`	turn on LED1 and LED3
`OK`	
`ATS133? S134? S135?`	
`1`	SW1 is in the OFF state

Command/Response	Comment
`0`	SW2 is in the ON state
`0`	SW3 is in the ON state
`OK`	

2.6.3. Prepare nodes for networking

The following examples require at least 2 nodes. The first step is configuring the network parameters. To do that, one of the nodes should function as a coordinator and others could be routers or end-devices. It is also important that all nodes should have different MAC and logical addresses. Typically, coordinator should have logical address 0, and all child nodes should have non-zero addresses[1].

Command/Response	Comment
`ATX`	set a node to transmit **`EVENT`** and **`DATA`** to a host
`OK`	
`AT+GSN=1`	set MAC address for the node
`OK`	
`AT+WPANID=1620`	set node's PAN ID
`OK`	
`AT+WROLE=0 +WSRC=0`	switch to coordinator function, set zero address
`OK`	
`AT+WAUTONET=1 Z`	enable automatic networking (1 second timeout if failed) and reboot
`OK`	

If node indicates **`ERROR`**, that means the embedded software does not support coordinator function and cannot be configured in such a way. In this case, try checking the coordinator support on other nodes using `AT+WROLE?` command, as described in 3.3.3.

Then, get another node and force it to be router:

[1] Selection of particular addresses is application dependent. It should be done only the first time during manufacturing process of initial installation.

Command/Response	Comment
`ATX`	set a node to transmit **EVENT** and **DATA** to a host
OK	
`AT+GSN=2`	set MAC address for the node
OK	
`AT+WPANID=1620`	set node's PAN ID
`OK`	
`AT+WROLE=1 +WSRC=55`	switch to router function, set address 55
OK	
`AT+WAUTONET=1   Z`	enable automatic networking (10 seconds timeout if failed) and reboot
OK	

2.6.4. Checking network status and basic data transmission

When both of the nodes were rebooted, after delay time set in `+WAUTONET` command, we can easily check networking status on the coordinator by `AT+WNWK` command and simply transmit the data from one node to another:

Command/Response	Comment
`AT+WNWK`	request networking status
OK	means that the node is in the network

`AT+WWAIT = 3000`
OK
`ATD 55`
`HELLO`
OK
	Set 3 sec timeout to wait for input and send HELLO word to the node with address 55

Simultaneously, `HELLO` word will appear on the terminal connected to the router in form of `DATA` event:

Command/Response	Comment
DATA 0000,0,5:HELLO	data (5 bytes) came from device with address 0 by unicast request

2.6.5. Remote Execution

Switches of the remote device having address 55 should be configured in the same way as described in Section 2.6.2. Then, send the `ATR` commands from the coordinator.

Command/Response	Comment

`ATR 55,0,S120=3 S121=3 S122=3`
`OK`
`ATR 55,0,S123=1 S124=1 S125=1`
`OK`
`ATR 55,0,S130=1 S131=0 S132=1`
`OK`
`ATR 55,0,S133? S134? S135?`
`1`
`0`
`0`
`OK`
	Configure GPIO0, GPIO1, GPIO2 for output
	Configure GPIO3, GPIO4, GPIO5 for input and turn on internal pull-up
	Check the end-device switches

`ATR55,0,+GMI?`
`+GMI:MESHNETICS`
`OK`
	Get model number and RSSI from the router

2.6.6. End-Device Power Control

This example will demonstrate how to configure end-device. An additional board should be connected to PC with Hyper Terminal run. Send the following commands from the Hyper Terminal to this board to set its duty cycle:

Command/Response	Comment

`AT+WCHMASK=50000 +WROLE=2 +WSRC=56`
`OK`
	Set the board as end-device with address 56 and set channel mask (channels `0x10, 0x12`)

`AT+WPWR=60,100`
`OK`
`AT+WPWR?`
`+WPWR:60,100`
`OK`
	Set duty cycle 6 sec sleep / 1 sec active
	Check on the duty cycle if accepted successfully

Command/Response	Comment
AT+WSYNCPRD=? **+WSYNCPRD:(10-30000)** **OK**	Check on the tracking valid range
AT+WSYNCPRD=120 **OK**	Set tracking period to 12 sec
ATS37? 120	Check on the tracking period parameter if stored in S-register
AT+WAUTONET=1 **OK**	Enable automatic networking (1 second)
ATZ **OK**	Reboot the end-device

Now, you can perform power consumption measurement for ZigBit module installed on the board. Simply connect ammeter to the clamps `CM+` and `CM-` and remove jumper `J1`. Make sure that the board is powered by batteries only. See ZEK/ZDK User's Guide for details.

Now, the data intended for the end-device is sent from the coordinator:

Command/Response	Comment
ATD56,0,4 test **OK**	Send test data from coordinator for the end device staying in a sleep mode

Request is sent from the end-device in active mode to its parent in order to check for the data buffered there:

Command/Response	Comment
DATA 0000,0,4:test **OK**	Request from the end-device for the data; the `test` word is transmitted back

3. Command Description

3.1. Protocol General Description

3.1.1. Character Formatting and Data Rates

Data transmitted between the host and the module over UART interface conforms to the requirements for start-stop data transmission specified in the ITU-T Recommendation V.4 [6]. Parity is even, odd or not used. Each character has at least one complete stop bit. The module accepts commands using any combination of parity and stop bits supported. These include, at least, the following combinations, each of which consists of up to ten bits (including the start bit):

- 7 data bits, even parity, 1 stop bit
- 7 data bits, odd parity, 1 stop bit
- 8 data bits, no parity, 1 stop bit.

Both the host and the module are able to accept commands at 1200 bit/s at least. Particular character formatting and the data rate can be changed using appropriate AT-commands – see 3.6.8 `(+IPR)`, 3.6.9 `(+IFC)`, 3.6.6 (`V`). The host has the means to select explicitly data rate and character formatting according to the specifications above.

3.1.2. Alphabet

For any information exchange between the module and the host the T.50 International Alphabet 5 (IA5) is used – see [4]. Only the seven low-order bits of each character are significant, any of eighth or higher-order bit(s), if present, are ignored for the purpose of identifying commands and parameters. Lower-case characters (hex codes `0x61` through 0`x7A`) are considered identical to their upper-case equivalents (hex codes `0x41` through `0x5A`) when received by the module from the host. Result codes from the module, which are particularly defined, are specified in upper case.

3.1.3. Basic Command-Line Operations

Command line editing, echoing and repeating are done in accordance with the Clauses 5.2.2, 5.2.3 and 5.2.4 of the Recommendation V.250. The description below follows the statements introduced in [3].

The module may echo the characters received from the host back to the host, depending on the setting of the **`E`** command (see 3.6.4). If so enabled, the characters received from the host are echoed at the same rate, parity, and format as those received.

The module checks on the characters coming from the host first, to see if they match the termination character `S3` (see 3.6.1). Next, it checks the editing character (`S5,` see 3.6.3), before considering any other character. That insures the characters will be properly recognized even though they were set to values which the module uses for other purposes. If `S3` and `S5` are set to the same value, the character checked will be treated as a character matching `S3` (as `S3` is checked before `S5`).

The character defined by `S5` parameter (by default, it is backspace character – BS [hex code `0x08`], see 3.6.3) is intended to be interpreted as a request from the host to the module to delete the previous character. Any control characters (hex codes `0x00` through `0x1F`, inclusive) that remain in command line after receiving the termination character will be ignored by the module.

Once the module finds the termination character, it starts processing the command line. Command line starts with `AT` (characters `0x41`, `0x54`) and should contain a sequence of commands in the following syntax formats:

Table 5. Command Syntax Formats

Command	Syntax
Action command	<command>[<value>]
Parameter set command	<command>=<value>
Parameter read command	<command>?
Testing a range of valid values	<command>=?

Where `<command>` is one of the following:

- a single character
- `'&'` character (`0x26`) followed by a single character
- `'%'` character (`0x25`) followed by a single character
- `'+'` character followed by a string of characters.

The characters allowed to be used in `<command>` should be taken from the T.50 International Alphabet 5. The first three of the command cases above are referred to as basic commands; they may be of the action command syntax only. Commands beginning with the plus sign are known as the extended syntax commands and can fit all the syntax rules depending on their type. Typically, a command that supports the parameter set syntax also supports the testing syntax.

A command (with associated parameters, if any) may be followed by additional commands in the same command line without using any delimiting character. Some commands may cause the remainder of the command line being ignored (the `D` command, see 3.4.1, for instance).

If command line is started with the '`A/`' or '`a/`' prefix (hex codes `0x41, 0x2F` or `0x61, 0x2F`), the module repeats immediately the execution of the preceding command line. No editing is possible, and no termination character is required. With this mechanism, a command line may be repeated as much as desired.

3.1.4. Parameter Values

Parameters may take either a single value, or multiple (compound) values. A compound value consists of any combination of numeric values (as defined in the description of the action or parameter command). The comma character (hex code `0x2C`) is included as a separator, before the second and all subsequent values in the compound value. If a value is not specified as missed (i.e. defaults assumed), the required comma separator should be specified; however, trailing comma characters may be omitted if all the associated values are also omitted.

Actions may have more than one of associated sub-parameters, and parameters may have more than one value. These are known as "compound values", and their treatment is the same in both the action command syntax and the parameter command syntax.

Each value may be either decimal or hexadecimal number[2]. The choice depends on a particular command and hexadecimal numbers if they are not preceded with `'0x'`. Hexadecimal numbers can represent 16-bit, 32-bit, 64-bit and 128-bit values.

Decimal numeric constants consist of a sequence of one or more of the characters `'0'` (hex code `0x30`) through `'9'` (hex code `0x39`), inclusive, and can be preceded by minus “-“. The most significant digit is specified first. The leading `'0'` characters will be ignored.

Hexadecimal numbers consist of characters “0” through “9” and “A” through “F”, inclusive. Minus sign is not allowed. The leading `'0'` characters will be ignored. To prevent misinterpretation of hexadecimal numbers in cases when the command containing them is not the last in the AT string, it is strongly recommended to add the leading zeroes. So, if a parameter is 32-bit long, it would be 8 characters long, if it is a 64-bit number, it would contain 16 characters and so on.

As a special case, string constant appears in `R` command (see 3.9.2) only. Then, it is just a sequence of displayable IA5 characters, each in the range of `0x20` to `0x7F`, inclusive.

3.1.5. Command Types

A command type may be one of the following:

- An action command
- A parameter command
- An S-registers command.

Parameters may be defined as "Read-only" (R) or "Read/Write" (RW). "Read-only" parameters are used to provide the host with the status or identifying information, but are not set by the host. Attempting to set such a parameter will result in an error. In some cases (depending on the particular parameter), the module may ignore any attempt to set the value for such parameter rather than respond with the **ERROR** result code. “Read-only” parameters may be read and tested.

"Read/Write" parameters may be set by the host in order to store a value or values for later use. “Read/Write” parameters may be set, read, and tested.

If `<command>` is not recognized, the module generates the **ERROR** result code and stops processing of the command line. The **ERROR** result code is also generated if: a sub-parameter is specified for an action that does not imply using sub-parameters; too many sub-parameters are specified; a mandatory sub-parameter is not specified; a value is specified of the wrong type; or if a value is specified that is not within the supported range.

Some commands allow omitting a `value`. If a command does omit one, then it should be immediately followed by another command (or the termination character) in the command line. The `'0'` value is assumed unless otherwise specified in the `<command>` description. If the `<command>` does not expect a `value` but the `value` is present, the **ERROR** code is generated.

[2] R command (see 3.9.2) is just a special case.

3.1.6. Action Command Syntax

The format of the action commands, except for the `D, DU` and `S` commands, is as follows:

Table 6. Action Command Syntax

Command	AT Syntax
Action command with no parameters used	<command>
Action command with one or more sub-parameters used	<command>[<value>]

The `value` may be either a single value parameter or a compound value parameter as described in 3.1.4. Some commands may have no parameters at all. Expecting a `value` is noted in the description of a particular command.

Example:

Command/Response	Comment
AT+WLEAVE	Leave the network
OK	Result code
ATX2	2 - Disables events and data indications
OK	Result code

3.1.7. Parameter Set Command Syntax

The following syntax is used for a parameter set command:

Table 7. Parameter Set Command Syntax

Command	AT Syntax
Parameter set command	<command>=[<value>]

If the named parameter is implemented in the module, all the mandatory values are specified, and all values are valid according to the definition of the parameter, the specified values should be stored. If `<command>` is not recognized, one or more of mandatory values are omitted, or one or more values are of wrong type or beyond the valid range, the module generates the **ERROR** result code and terminates processing of the command line. **ERROR** is also generated if too many values are specified. In case of error, the previous values of the parameter are unaffected.

Example:

Command/Response	Comment
AT+WRETRY=3	Set parameter +WRETRY
OK	Result code

3.1.8. Parameter Read Command Syntax

The host may determine current value or values stored in a parameter by using the following syntax:

The following syntaxes are used.

Table 8. Parameter Read Command Syntax

Command	AT Syntax
`Parameter read command`	`<command>?`

If the named parameter is implemented, its current values are sent to the host in an information text response. The format of this response is described in definition of the parameter. Generally, the response string is beginning with `<command>` followed by `':'` character and the values represented in the same form, in which they would be generated by the host in a parameter set command. If multiple values are supported, they will generally be separated by commas, as in a parameter set command. For example:

Command/Response	Comment
`AT+WRETRY?`	Request for parameter `+WRETRY`
`+WRETRY:3`	Returned value
`OK`	Result code

3.1.9. Parameter Test Command Syntax

The host may test if an action command or parameter set command is implemented in the module, and determine the supported values, by using the following syntax:

Table 9. Parameter Test Command Syntax

Command	AT Syntax
`Parameter test command`	`<command>=?`

If the module does not recognize the indicated `<command>`, it returns the **`ERROR`** result code and terminates processing of the command line. If the module does recognize the parameter name, it returns an information text response to the host, followed by the **`OK`** result code. The information text response will indicate the values supported by the module for each of sub-parameters, and, possibly, additional information. The format of this information text response is defined for each parameter. See 3.1.12 for the general formats for specification of sets and ranges of numeric values. Generally, an information text response is started with a `<command>` followed by `':'`.

When an action/parameter accepts a single numeric sub-parameter, or the parameter accepts only one numeric value, the set of supported values may be presented in an information text as an ordered list of values. The list should be preceded by left parenthesis '`(`', (hex code `0x28`), and closed by right parenthesis '`)`', (hex code `0x29`). If that very single value is supported, it should appear in parentheses. If more than one value is supported, then the values may be listed individually, separated by comma characters (hex code `0x2C`). When a continuous range of values is supported, the values appear in form of the first value in the range, and the last value in the range, both separated by a hyphen character (hex code `0x2D`). The specification of single values and value ranges may be alternated within a single information text. Nevertheless, the supported values should be indicated in an ascending order. For example, the following are some examples of value range indications:

`(0)`	Only the `0` value is supported.
`(1,2,3)`	The values `1, 2`, and `3` are supported.
`(1-3)`	The values `1` through `3` are supported.
`(0,4,5,6,9,11,12)`	The several listed values are supported.
`(0,4-6,9,11-12)`	Alternative expression of the previous list.

Example:

Command/Response	**Comment**
`AT+WPANID=?`	Request for valid range of the parameter `PAN ID`
`+WPANID:(0-FFFF)`	Returned value
`OK`	Result code

When an action/parameter accepts more than one sub-parameter, or the parameter accepts more than one value, the set of supported values may be presented as a list of the parenthetically-enclosed value range strings, separated by commas. For example, the information text in response to testing an action that accepts three sub-parameters, and supports various ranges for each of them, could appear as follows:

```
   (0),(1-3),(0,4-6,9,11-12)
```

This indicates that the first sub-parameter accepts only the `0` value, the second accepts any value from `1` through `3,` inclusively, and the third sub-parameter accepts any of the values `0, 4, 5, 6, 9, 11` or `12`.

3.1.10. S-registers

S-registers represent a group of numerical parameters that can be addressed in a special syntax. Each S-register has its own address and value. Some S-registers are standardized by the V.250 recommendations and are used in the module. Some of the S-registers are non-standard defined specifically by the SerialNet software.

AT-commands that begin with the **`'S'`** character are allowed for S-register access. These differ from other AT-commands in some respects. The number following the **`'S'`** character indicates the referenced "register number". If the number is not recognized as a valid register number (register is omitted), the **`ERROR`** result code is generated.

Immediately following that number, either a `'?'` or `'='` character (hex codes `0x3F` or `0x3D`, respectively) should appear. `'?'` is used to read the current value of the indicated S-parameter. `'='` is used to set the S-parameter to a new value.

Table 10. S-Registers

Command	AT Syntax
`Reading the S-register`	`S<parameter_number>?`
`Setting the S-register`	`S<parameter_number>=[<value>]`

If the `'='` character is used, the new value to be stored in the S-parameter is specified in decimal form following the `'='` character. If no value is given (i.e. the end of the command line occurs or the next command follows immediately), the corresponding S-parameter will be set to `0`. The ranges of acceptable values are given in description of each S-register.

Section 2.5 gives functional representation of S-registers associated to the commands.

3.1.11. Module Responses

There are two types of responses that may be generated by the module:

- information text
- result codes.

Basically, any information text response consists of three parts: header, text, and trailer. The characters generated in header are determined by user's setting (see `V` command, 3.6.6). Trailer consists of two characters, namely the ordinal value of parameter `S3` followed by the ordinal value of parameter `S4`. Information text may contain multiple lines, and the text may include any formatting characters to improve readability.

A result code consists of three parts: header, the result text, and trailer. The characters to be generated in header and trailer are determined by user's setting (see the **`V`** command, 3.6.6). The result text may be generated as a number or a string, depending on the user-selected setting (see the **`V`** command, 3.6.6).

There are two general types of result codes: final and unsolicited.

Final result codes (**`OK/ERROR`**) indicate completion of the module action and readiness to accept new commands from the host. Unsolicited result codes (such as **`DATA`**) may not be directly associated with the issuance of a command from the host. They indicate the occurrence of another **`EVENT`** causing them.

Command `X` (see 3.6.7) controls the generation of result codes, while command `Q` (see 3.6.5) – results in their total suppression.

Section 3.1.11 summarizes representations the result codes are in both verbose and numeric forms with the corresponding parameter(s), if any, and their brief description. Each command description itself refers to the specific result codes that may be generated in relation to the command and the circumstances, under which they may be issued.

3.1.12. Information Text Formats

In general, the particular format of information text returned by extended syntax commands will be specified in the command definition.

Note that the module may insert intermediate <`CR`> characters in very long information text responses, in order to avoid overflow in the host receive buffers. If intermediate `<CR>` characters are included, the module does not include the character sequences "**`0`** `<CR>`" (`0x30`, `0x0D`) or "**`OK`**`<CR>`" (`0x4F`, `0x4B`, `0x0D`), so that the host can avoid false detection of the end of these information text responses.

3.2. Network management functions

3.2.1. "+WPANID" – Set/request for PAN ID

Syntax/Descriptor	Explanation
`+WPANID=<value>`	The command sets PAN ID for the node. `value` is a hexadecimal 16-bit number that will be used for all the network operations. If PAN ID is set to `FFFF`, the module will join the network irrespectively to its PAN ID. NOTE: Setting the PAN ID will affect the next network join and will not require rejoin, if the node is in the network already.
`+WPANID?`	The command returns PAN ID that was previously set by `+WPANID=<value>` command.
`+WPANID=?`	The command requests for PAN ID valid range.
S-register	`S21` (RW). This register is just keeping a copy of the parameter accessible through `+WPANID` command. `S20` (R). This register contains actual PAN ID that is used for networking. If `S21` register is set to `FFFF,` and the node has been joined the network, this register will keep PAN ID of the selected network. If the node has not been connected, this register contains `FFFF`.
Result codes	The set command is executed if the node is not in the network and PAN ID is in the valid range. In such case the module returns `OK` upon completion. Otherwise, PAN ID is ignored and the node responds with `ERROR`.

Example	`AT+WPANID=10`
	`OK`
	`AT+WPANID?`
	`+WPANID:10`
	`OK`
	`AT+WPANID=?`
	`+WPANID:(0000-FFFF)`
	`OK`

Default value	`FFFF` for routers and end-devices and `FFFE` for coordinator
Persistence	`value` is stored in EEPROM
Node types	Coordinator/Router/End-device

3.2.2. "+WCHAN" – Request for active channel

Syntax/Descriptor	Explanation
`+WCHAN?`	The command requests for a channel number (in hexadecimal form) which is currently used for networking. Channel numbering conforms to 802.15.4-2003 allocations; channel 0 corresponds to 868 MHz, channels 01 through 0A – to 915 MHz band, 0B through 1A – to 2450 MHz band. If the node is not connected to the PAN, FF is returned.
S-register	`S22` (R)
Result codes	**OK**
Example	`AT+WCHAN?` **+WCHAN:0B** **OK**
Node types	Coordinator/Router/End-device

3.2.3. "+WCHMASK" – Set/get Channel mask

Syntax	Explanation
`+WCHMASK=<value>`	The command sets channel mask that will be used for networking. Channel mask `value` is a 32-bit unsigned hexadecimal number, where the 27 LSBs (b0, b1 ... b26) represent the status (`1`=available; `0`=unavailable) for each of the 27 valid channels, correspondingly. The b0 bit corresponds to 868 MHz frequency band, bits b1...b10 – to 915 MHz band, and bits b11 through b26 – to 2450 MHz band. Detailed description can be found in 6.1.2 of the 802.15.4-2003 standard [7]. NOTE: Setting the channel mask will affect the subsequent network operations and do not affect actual channel selection, if the node is already in the network.
`+WCHMASK?`	The command returns actual channel mask. The returned channel mask can be different from the channel mask set by `+WCHMASK=<value>` command and depends on the hardware capabilities. The cleared bits mark unsupported channels.
`+WCHMASK=?`	The command returns channel capability mask in form of 32-bit unsigned hexadecimal number. For example, for 2.4 GHz chipset, it returns `07FFF800`.
S-register	`S23` (RW).

Syntax	Explanation
Result codes	The set command is executed if the node is not in the network. Channel mask is set according to hardware capabilities really available. In such case the module returns **OK**. Otherwise, channel mask is ignored and the node responds with **ERROR**.
Example	AT+WCHMASK=FFFF **OK** AT+WCHMASK? **+WCHMASK:0000F800** **OK** AT+WCHMASK=? **+WCHMASK:07FFF800** **OK**
Default value	00000800. This means that the module will attempt using 000B channel first time (see 3.2.2).
Persistence	The value is stored in the EEPROM.
Node types	Coordinator/Router/End-device

3.2.4. “+WLEAVE” – Leave the network

Syntax	Explanation
+WLEAVE	The command forces the module (the node) to leave the network. The node forces all its children to leave and signalize a CHILD_LOST event to its parent node. NOTE: This function disables automatic networking (see 3.2.9) temporarily. To enable automatic networking, the node should either execute +WJOIN command or has to be rebooted by Z command.
Result codes	**OK** is returned on the process completion. If the node was not connected before starting the process, it returns **ERROR** immediately.
Example	AT+WLEAVE **OK**
Node types	Coordinator/Router/End-device

3.2.5. "+WJOIN" – Start/Join to the network

Syntax	Explanation
`+WJOIN`	The command forces the module (the node) to start (for Coordinator node) a network or to join (for Router or End-device node) the existing network. NOTE: The nodes can share the same frequency band, and several networks can work in parallel on the same channel. The node selects required network via setting the PAN ID (3.2.1) and the channel mask (3.2.3).
Result codes	**`OK`** is returned if formation/joining the network completed successfully, or **`ERROR,`** if failed. If the node is in the network already, it returns **`OK`** immediately.
Example	`AT+WJOIN` **`OK`**
Node types	Coordinator/Router/End-device

3.2.6. "+WNWK" – Request for networking status

Syntax	Explanation	
`+WNWK`	The command requests for networking status	
Result codes	**`OK`** is returned if the node has been already joined the network, otherwise it returns **`ERROR`**, if it has being orphaned by its parent or the network is not found during the joining process.	
Example	`AT+WLEAVE`	Leave the network first
	`OK`	
	`AT+WNWK`	
	`ERROR`	We are not in the network now
Node types	Coordinator/Router/End-device	

3.2.7. "+WPARENT" – Request for parent address

Syntax	Explanation
`+WPARENT?`	The command requests for parent address. MAC parent address is returned as a 64-bit hexadecimal number if `S30` register is set to 0. NWK parent address is returned if `S30` register is set to 1. See Section 3.3.9 for details. This command does not cause network operations and just returns a copy of the parent address that was assigned during the joining process.
Result codes	**`OK`** is returned if the module is in the network and has a parent. If the module is not in the connected state or if it is run as Coordinator, **`ERROR`** will be returned.
Example	`AT+WPARENT?` **`+WPARENT:0123456789ABCDEF`** **`OK`**
Node types	Routers and End-devices

3.2.8. "+WCHILDREN" – Request for children addresses

Syntax	Explanation
`+WCHILDREN?`	The command requests for children addresses. MAC children addresses are returned as a 64-bit hexadecimal numbers if `S30` register is set to 0. NWK children addresses are returned if `S30` register is set to 1. See Section 3.3.9 for details. Children addresses are returned delimited by commas.
Result codes	**`OK`** is returned if the module is in the network even though there is no child connected yet. If the module is not in the connected state or if it is run as End-Device, **`ERROR`** will be returned.

Example	`AT+WCHILDREN?`
	`+WCHILDREN:0123456789ABCDEF,123456789ABCDEF0`
	`OK`

Node types	Coordinator and Routers

3.2.9. “+WAUTONET” – Automatic networking

<table>
<tr><th>Syntax</th><th colspan="2">Explanation</th></tr>
<tr><td>+WAUTONET=<value&gt;</td><td colspan="2">The command controls the node activity behaved at power-up, reset or when it detects connection loss. value is a 16-bit value that represents the sleeping interval in seconds between two consecutive attempts to join the network in case of failure. If the value is zero that means that automatic joining to the network is disabled.</td></tr>
<tr><td>+WAUTONET?</td><td colspan="2">The command requests for actual value.</td></tr>
<tr><td>+WAUTONET=?</td><td colspan="2">The command requests for the range of supported values.</td></tr>
<tr><td>S-register</td><td colspan="2">S24 (RW).</td></tr>
<tr><td>Result codes</td><td colspan="2">OK</td></tr>
<tr><td>Example</td><td>AT+WAUTONET=10
OK
AT+WAUTONET?
+WAUTONET:10
OK
AT+WAUTONET=?
+WAUTONET:(0-1000)
OK</td><td>Set 10 sec interval between automatic joining</td></tr>
<tr><td>Default value</td><td colspan="2">0. Disabling an automatic networking.</td></tr>
<tr><td>Persistence</td><td colspan="2">value is stored in the EEPROM.</td></tr>
<tr><td>Node types</td><td colspan="2">Coordinator/Router/End-device</td></tr>
</table>

3.3. General node management

3.3.1. "+WPWR" – Power management

Syntax	Explanation	
`+WPWR=<sleep>,<active>`	The command sets sleep/active duration; `sleep` duration is specified in 100 msec units but `active` duration – in 10 msec units. NOTES: Actual sleep/active periods will be slightly different and their values depend on multiple circumstances such as the network activity, external interfaces to the sensors, and so on. They can not be used for absolute timing. These values are sent to the Router for management of the delayed data to be saved there during the periods of the node's inactivity. Thus, a proper change of these values requires the node to rejoin.	
`+WPWR?`	The command requests for sleep/active durations set before by the `+WPWR=` command	
`+WPWR=?`	The command requests for valid ranges of sleep/active durations.	
S-registers	`S31, S32` (RW).	
Result codes	**`OK`** is returned if parameters are within their valid ranges. **`ERROR`** will be returned if requested for Coordinators and Routers.	

Example	`AT+WPWR=600,10`
	`OK`
	`AT+WPWR?`
	`+WPWR:600,10`
	`OK`
	`ATS31?`
	`600`
	`OK`
	`AT+WPWR=?`
	`+WPWR:(2-30000),(2-30000)`
	`OK`
		Set duty cycle 1 min sleep / 100 msec active

Syntax	Explanation
Default values	10,10 (the node sleeps for 1 second and is active for 100 msec)
Persistence	The `sleep, active` values are stored in the EEPROM.
Node types	End-Devices

3.3.2. “+WSLEEP” – Force to sleep

Syntax	Explanation
`+WSLEEP`	The command forces the module to fall into sleep mode. This command lets power management of End-Devices be more flexible. IMPORTANT: Take in mind that the module in sleep mode can respond to the subsequent commands with a delay, depending on the sleeping interval specified (see 3.3.1), the module version and DTR configuration (see 3.6.10).
Result codes	**`OK`** is returned for End-Devices, otherwise **`ERROR`**. NOTE: The command is executed as follows: the module returns the result code first, and then it disables any of subsequent commands, completes pending operations and finally falls into the sleep mode. Wake-up is scheduled by `+WPWR` command.
Example	`AT+WSLEEP` **`OK`**
Node types	End –Devices

3.3.3. “+WROLE” – Set/request for node role (coordinator/router/end-device)

Syntax	Explanation
`+WROLE=<value>`	The command sets the node role (`0` – Coordinator, `1` – Router, `2` – End-Device). NOTE: It is strongly recommended to avoid changing the role during any working-mode operation and then to execute warm reboot (`ATZ` command) after setting the new role. This setting may be done during commissioning process only and, since the role is a persistent parameter, the node will carry the selected function until set to another role or executing `&F` command (see 3.5.9).
`+WROLE?`	The command requests for actual node role.

<table>
<tr><th>Syntax</th><th colspan="2">Explanation</th></tr>
<tr><td><code>+WROLE=?</code></td><td colspan="2">The command requests for the allowable range. Indicated capabilities depend on the particular firmware version burned in the module.</td></tr>
<tr><td>S-register</td><td colspan="2"><code>S33</code> (RW).</td></tr>
<tr><td>Result codes</td><td colspan="2"><b><code>OK</code></b> is returned if <code>value</code> is in the valid range, otherwise <b><code>ERROR.</code></b></td></tr>
<tr><td rowspan="12">Example</td><td><code>AT+WLEAVE</code></td><td>Leave the network</td></tr>
<tr><td><b><code>OK</code></b></td><td></td></tr>
<tr><td><code>AT+WCHMASK=0</code></td><td>Disable air transmission</td></tr>
<tr><td><b><code>OK</code></b></td><td></td></tr>
<tr><td><code>AT+WROLE=?</code></td><td></td></tr>
<tr><td><b><code>+WROLE:(1,2)</code></b></td><td>Can be either Router or End-Device</td></tr>
<tr><td><b><code>OK</code></b></td><td></td></tr>
<tr><td><code>AT+WROLE=2</code></td><td>Switch to the End-Device role</td></tr>
<tr><td><b><code>OK</code></b></td><td></td></tr>
<tr><td><code>AT+WROLE?</code></td><td></td></tr>
<tr><td><b><code>+WROLE:2</code></b></td><td></td></tr>
<tr><td><b><code>OK</code></b></td><td></td></tr>
<tr><td>Default value</td><td colspan="2">Depends on the firmware version. Typically 1 – Router.</td></tr>
<tr><td>Persistence</td><td colspan="2"><code>value</code> is stored in the EEPROM.</td></tr>
<tr><td>Node types</td><td colspan="2">Coordinator/Router/End-device</td></tr>
</table>

3.3.4. "+WSYNCPRD" – Period for tracking the End-Devices

<table>
<tr><th>Syntax</th><th colspan="2">Explanation</th></tr>
<tr><td>+WSYNCPRD=<period&gt;</td><td colspan="2">The command sets the <period> value measured in 100 msec units for tracking the End-Device by its Router. The End-Device sends the <period> value to the Router during the join process. Router uses this value to control lifetime timer and pending data for this End-Device.
NOTES:
This value will affect the subsequent join operations and does not apply the actual values on the Router. Thus the node has to rejoin to apply this setting.
Right selection for this value is application specific. It depends on various circumstances such as network structure, its size, average air total rate, sending data interval and so on. This number is recommended to be set at least as much as 3 times more than the sleep duration set by +WPWR command. To secure a fair robustness against short-term network overflows, this timeout should not being set too small, for typical cases not being less than 1 minute.</td></tr>
<tr><td>+WSYNCPRD?</td><td colspan="2">The command requests actual tracking period.</td></tr>
<tr><td>+WSYNCPRD=?</td><td colspan="2">The command requests allowable range of tracking period duration.</td></tr>
<tr><td>S-registers</td><td colspan="2">S37 (RW).</td></tr>
<tr><td>Result codes</td><td colspan="2">**OK** is always returned.</td></tr>
<tr><td>Example</td><td>AT+WPWR=600,10
OK
AT+WSYNCPRD=1800
OK
ATS37?
1800
OK
AT+WSYNCPRD=?
+WSYNCPRD:(10-30000)
OK</td><td>Set duty cycle 1 min sleep / 100 msec active
Set tracking period to 3 minutes</td></tr>
<tr><td>Default values</td><td colspan="2">10 (1 second)</td></tr>
<tr><td>Persistence</td><td colspan="2">The period value is stored in the EEPROM.</td></tr>
<tr><td>Node types</td><td colspan="2">End-Devices</td></tr>
</table>

3.3.5. "+WTXPWR" – TX power level

Syntax	Explanation	
`+WTXPWR=<value>`	The command sets transmit power level. The `value` represents TX power level measured in dBm. NOTE: In the eZeeNet ZDK/ZEK versions, this setting will be applied to the radio circuitry during the warm reset procedure only. Thus, the accurate setting of TX power requires warm reboot of the module in using `Z` command, see 3.5.8.	
`+WTXPWR?`	The command requests for actual TX power level. NOTE: Power level resolution is typically 3 dB. This command just returns the number set by the `+WTXPWR=` command, but does not indicate real power level, which can vary due to the temperature, supply voltage and another factors.	
`+WTXPWR=?`	The command requests for the allowable range of TX level.	
S-register	`S34` (RW).	
Result codes	**`OK`** is returned if `value` is in the valid range, otherwise **`ERROR`**.	

Example	`AT+WTXPWR=-5`
	`OK`
	`AT+WTXPWR?`
	`+WTXPWR:-5`
	`OK`
	`AT+WTXPWR=?`
	`+WTXPWR:(-17-3)`
	`OK`
		set -5dBm TX level

Default value	0	
Persistence	`value` is stored in the EEPROM.	
Node types	Coordinator/Router/End-device	

3.3.6. “+WSEC” – Encryption key

Syntax	Explanation
`+WSEC <hi>,<lo>`	The command sets encryption key. `<lo>` and `<hi>` are 64-bit hexadecimal numbers representing high and low parts of 128-bit encryption key. This key will be used for secured transmission within the PAN. NOTES: It is strongly recommended to avoid changing encryption key during networking. Before doing that let the node to leave the network, otherwise it can loose connection. If security key change command comes from the air via `R` command (see 3.9.2), the module should respond with the key used before executed command and switch to the new value after processing the line containing new encryption key.
Result codes	**`OK`** is returned always .

Example	`AT+WSEC 0123456789ABCDEF,0123456789ABCDEF`
	`OK`

Default value	`0,0` – no encryption
Persistence	`value` is stored in the EEPROM.
Node types	Coordinator/Router/End-device

3.3.7. “+WLQI” – Request for LQI value

Syntax	Explanation	
`+WLQI <addr>`	The command requests for LQI for a signal received from the node having the `addr` MAC address. MAC address is specified in 64-bit hexadecimal format. The command returns the actual LQI value ranged by `0`…`255`. If the node is not in the network or LQI information is not available, `0` is returned. NOTE: LQI information is retrieved for links within one-hop radius. LQI is not provided for multi-hop links.	
Result codes	The module returns **`OK`** if LQI value for this particular link exists, otherwise **`ERROR`** will be returned.	
Example	`AT+WLQI` 000100000a3b9cf9 **`+WLQI:254`** **`OK`**	request for LQI for link with node having 000100000a3b9cf9 address
Node types	Coordinator/Router/End-device	

3.3.8. "+WRSSI" – Request for RSSI

Syntax	**Explanation**	
`+WRSSI <addr>`	The command requests for actual RSSI value for a signal received from the node having the `addr` MAC-address. MAC address is specified in 64-bit hexadecimal format. The command returns the actual RSSI value expressed in dBm. If RSSI is not available, then -128 value is returned. NOTE: RSSI information is retrieved for links within one-hop radius. RSSI for multi-hop links is not provided.	
Result codes	The module returns **`OK`** if RSSI value exists for this particular link, otherwise **`ERROR`** will be returned.	

Example	`AT+WRSSI 000100000a3b9cf9`
	`+WRSSI:-80`
	`OK`
		request for RSSI for link with node having 000100000a3b9cf9 address
		-80 dBm

Node types	Coordinator/Router/End-device	

3.3.9. "S30" – Set network addressing mode

Syntax	Explanation
S30=<value>	The command sets the mode for addressing to be used by some commands. `<value>`: Addressing mode 0 the particular command dependent addressing 1 NWK addressing
S30?	The command requests for the addressing mode currently valid.
Result codes	The command returns **OK** if <`value`> is in range, otherwise **ERROR**.
S-register	`S30` (RW)

Example	`ATS30=0`
	OK
	`AT+WSRC?+WPARENT?+WCHILDREN?`
	0055
	000100000A3B98CC
	000100000A3B10AA
	OK
	`ATS30=1`
	OK
	`AT+WSRC?+WPARENT?+WCHILDREN?`
	0004
	0000
	0007
	OK
	`ATR7,0,S30?`
	`0`
	OK

Node types	Coordinator/Router/End-device
Default value	0
Persistence	`value` is NOT stored in EEPROM

NOTE:

Setting the addressing mode, the `S30` command affects the performance of the following commands: `+WPARENT?` (see Section 3.2.7), `WCHILDREN?` (see Section 3.2.8), `WSRC?` (see Section 3.7.4), and `R` (see Section 3.9.2). Those commands use either MAC or logical address if `S30` is set to 0. They will use NWK addressing if `S30` is set to 1.

As advantage, logical address of a node is not fixed. Logical addressing is preferable when the address of each node is known in advance or when the addresses can be preset during the commissioning procedure. As disadvantage, address conflicting is possible and should be resolved manually or by dedicated software running on the coordinator node.

NWK addresses are allocated and changed dynamically. NWK addressing scheme is only recommended for initial network addressing setup when application receives the data from some unknown node or when several nodes in the network have to use the same logical address. This would be the way to resolve address duplication or provide plug-and-play node installation.

NWK addressing scheme can be also used in wireless network where data is collected at single central point (sink) and no data should be transmitted back. There, logical addressing is not required because NWK address is known for coordinator and it equals zero.

3.4. Data transmission

Data can be transmitted in two ways:

- direct addressing a particular node in using the `D, DS, +WPING` commands;
- addressing all the nodes in using the `DU` command.

An End-Device which is sleeping should initiate data requests periodically to get the data from its Router. That can be done in using the `DR` command.

First two of the above cases fit the optimal data delivery mode and routing adjustment for small networks. It is important that MAC addresses are not used for networking directly; instead, they are substituted by short logical addresses which are convenient for node replacement in network installation and maintenance.

3.4.1. "D" – Send data to a specific node

Syntax	Explanation

`D <addr>[,[<arq>][,<length>]]`
`<data>`
	The command sends data to a specific node (using the implicitly defined MeshNetics private ProfileID, clusterID, end-point). `arq` parameter (equal to 1 or 0) controls ARQ/nonARQ data delivery mode, meaning 1 (i.e. ARQ) as default when omitted. Destination address should be a 16-bit hexadecimal logical address.
	The data portion may not exceed the maximum allowable number (80 characters).
	`length` means the length in bytes of the data portion to be sent. Data transmission starts up either from the specified number of data bytes is received or the time interval between two consecutive symbols in data field exceeds the timeout preset (`+WWAIT` command, 3.7.3). If length parameter is omitted, the maximum allowable number is implied by default.
	NOTE:
	Data should be preceded by `<CR>` (`S3` character, see 3.6.1). This symbol is not transmitted over the air and it is not counted in length.

Syntax	Explanation	
Result codes	If acknowledgement is requested (`arq` is set to 1), the module responds with **OK** upon receiving an acknowledgement in several attempts (see parameter `+WRETRY`, 3.7.2), otherwise it returns **ERROR**. If the destination node or the sending node itself is not in the network **ERROR** is returned.	

Example	`ATD 12,1,5`
	HELLO
	OK
	`ATD 12`
	HELLO
	OK
		Send **HELLO** to the node with address 12 using ARQ.
		The same as above, but the module will be awaiting for the timeout expiration before going to the air.

Node types	Coordinator/Router/End-device	

3.4.2. “DU” – Send broadcast data

Syntax	Explanation	

`DU [<length>]`
`<data>`
	The command sends `data` in using broadcast transmission (using the implicitly defined MeshNetics private ProfileID, clusterID, end-point).
	The `data` portion may not exceed the maximum allowable number (80 characters).
	`length` means the length in bytes of the `data` portion to be sent. Data transmission starts upon either the specified number of data bytes is received or the time interval between two consecutive symbols in data field exceeds the timeout preset (`+WWAIT` command, 3.7.3). If `length` parameter is omitted, the maximum allowable number is implied by default.
	NOTES:
	Data should be preceded by `<CR>` (`S3` character, see 3.6.1). This symbol is not transmitted over the air and it is not counted in length.
	Data is broadcasted in one-hop range. Broadcast data retransmission is not made to prevent flooding the network.	

Result codes	The module responds with **OK** immediately after the transmission if the node itself is in the PAN. Otherwise, **ERROR** is returned.	
Example	`ATDU` **HELLO** **OK**	Send **HELLO** to all nodes in one-hop range

Syntax	Explanation
Node types	Coordinator/Router/End-device

3.4.3. "DS" – Send S-register value to a specific node

Syntax	Explanation	
`DS <S-reg>,<addr>[,[<arq>]`	The command sends S-register value to a specific node (using the implicitly defined MeshNetics private ProfileID, clusterID, end-point). `arq` parameter (is set to `1` or `0`) controls ARQ/nonARQ data delivery mode, meaning `1` (ARQ) as default if omitted. Destination address should be a 16-bit hexadecimal logical address. S-register data is sent in the form readable by `ATS` command without the line termination characters. NOTE: S-registers defined by user extensions are also accessible by this command.	
Result codes	If acknowledgement is requested (`arq` is set to 1), the module responds with **`OK`** upon receiving acknowledgement in several attempts (see parameter `+WRETRY`, 3.7.2), otherwise it returns **`ERROR`**. If the destination node or the sending node itself is not in the network **`ERROR`** is returned. Also, if the specified S-register can not be read, the command returns **`ERROR`** and the module does not send anything to the air.	

Example	`ATDS130,2,1`
	`OK`
		Send `GPIO0` value to the node with address `2` using ARQ.

Node types	Coordinator/Router/End-device	

3.4.4. "DR" – Delayed data request

Syntax	Explanation
`DR`	The command requests explicitly for the data buffered on the Router. This function is used by a sleeping device in order to request for the data which are possibly buffered on its Router. This can be required if the sleeping device has to participate in two-way communications with another node. Anyhow, the device executes this request automatically on each wakeup.

Syntax	Explanation	
Result codes	The module responds with **OK** upon receiving acknowledgement from the Router irrespectively of the buffered data size. If Router is not responding then the module returns **ERROR**. NOTES: The buffered data will be delivered in form of **DATA** response, it may come either ordinarily or even before the **OK** result code is returned. If there is no buffered data, the **DATA** response will not be present.	

Example	`ATDR`
	`DATA 0002,0,5,HELLO`
	`OK`
		HELLO word came from the node 2

Node types	End-Devices	

3.4.5. "+WPING" – Ping the node

Syntax	Explanation
`+WPING <addr>`	The command pings the targeted node. `addr` destination address should be 16-bit hexadecimal logical address. In fact, this command is equivalent to `D` command with zero data length: `ATD <addr>,1,0`.
Result codes	The module responds with **OK** upon receiving acknowledgement in several attempts (see parameter `+WRETRY`, see 3.7.2, otherwise it returns **ERROR**. If the destination node or the sending node itself is not in the network **ERROR** is returned.

Example	`AT+WPING 1`
	`OK`

Node types	Coordinator/Router/End-device

3.5. Generic control

3.5.1. “&H” – Command Help

Syntax	Explanation
&H	The command outputs a list of valid AT-commands. The listing order may change. It depends on firmware version.
Result codes	**OK** is always returned
Example	AT&H **E** **V** **Q** **Z** **&F** **+IPR** **+IFC** **&D** **&H** **%H** **I** **+GMI** **+GMM** **+GMR** **+GSN** **…** **S135** **S136** **S137** **OK**
Node types	Coordinator/Router/End-device

3.5.2. “%H” – Display parameters and S-register values

Syntax	Explanation
%H	The command outputs the values of parameters and S-registers. The listing order may change. It depends on firmware version.
Result codes	**OK** is always returned

Syntax	Explanation

Example	`AT%H`
	`E:1`
	`V:1`
	`Q:0`
	`+IPR:9600`
	`+IFC:2,2`
	`&D:1`
	`+GMI:MESHNETICS`
	`+GMM:ZIGBIT`
	`+GMR: ZDM-A1281-U Rev.1.1/eZeeNet software v.1.6.3.0`
	`+GSN:12`
	…
	`S135:0`
	`S136:0`
	`S137:0`
	`OK`

Node types	Coordinator/Router/End-device

3.5.3. “I” – Display the product identification information

Syntax	Explanation
`I[<value>]`	The command instructs the module to transmit an information text intended to identify the module, depending on the `value` as follows: `value` — Information text — Reference 0 — All the identifier below 1 — Manufacturer identifier — 3.5.4 2 — Model identifier — 3.5.5 3 — Hardware/software revision identifier — 3.5.6 4 — Product serial number identifier — 3.5.7 If `value` is omitted, 0 is implied by default.
Result codes	`OK` is always returned

Syntax	Explanation

Example	`ATI0`
	`MESHNETICS`
	`ZIGBIT`
	`ZDM-A1281-U Rev.1.1/eZeeNet software v.1.6.3.0`
	`FEDCBA0987654321`
	`OK`

Node types	Coordinator/Router/End-device

3.5.4. "+GMI" – Request for the manufacturer identifier

Syntax	Explanation	
`+GMI?` `I1`	The command instructs the module to transmit an information text intended to identify the manufacturer.	
Result codes	**`OK`** is always returned	

Example	`AT+GMI?`
	`+GMI:MESHNETICS`
	`OK`
	`ATI1`
	`MESHNETICS`
	`OK`
		Just an alias to `+GMI`

Node types	Coordinator/Router/End-device	

3.5.5. "+GMM" – Request for the model identifier

Syntax	Explanation	
`+GMM?` `I2`	The command instructs the module to transmit an information text intended to identify the particular model of the device.	
Result codes	**`OK`** is always returned	

Example	`AT+GMM?`
	`+GMM:ZIGBIT`
	`OK`
	`ATI2`
	`ZIGBIT`
	`OK`
		Just an alias to `+GMM`

Node types	Coordinator/Router/End-device	

3.5.6. “+GMR” – Request for the hardware/software revision identifier

Syntax	Explanation	
`+GMR?` `I3`	This command instructs the module to transmit an information text intended to identify the actual revision of hardware or software product burned into the device.	
Result codes	**`OK`** is always returned	

Example	`AT+GMR?`
	`+GMR: ZDM-A1281-U Rev.1.1/eZeeNet software v.1.6.3.0`
	`OK`
	`ATI3`
	`ZDM-A1281-U Rev.1.1/eZeeNet software v.1.6.3.0`
	`OK`
		Just an alias to `+GMR`

Node types	Coordinator/Router/End-device	

3.5.7. “+GSN” – Read/Write MAC address

Syntax	Explanation	
`+GSN?` `I4`	The command outputs the module’s MAC address in form of a 64-bit hexadecimal number.	
`+GSN=<address>`	The command sets the module’s MAC address in form of a 64-bit hexadecimal number.	
Result codes	**`OK`** is always returned	

Example	`AT+GSN=FEDCBA0987654321`
	`OK`
	`AT+GSN?`
	`+GSN:FEDCBA0987654321`
	`OK`
	`ATI4`
	`FEDCBA0987654321`
	`OK`
		Just an alias to `I4`

Syntax	Explanation
Default value	0000000000000000 Important Note: If MAC address was not defined by user (so it is equal to zero or to 0xFFFFFFFFFFFFFFFF), the MAC address is attempted to find in the module's ZigBit hardware. The detected address will then be used. The module will not join network until user will set MAC address to any value which differs from zero and from 0xFFFFFFFFFFFFFFFF, or MAC address is detected in hardware.
Persistence	address value is stored in EEPROM
Node types	Coordinator/Router/End-device

3.5.8. "Z" – Warm reset

Syntax	Explanation
Z	The command instructs the module to simulate warm (software) reset. This command resets the hardware, restores all persistent variables from EEPROM and restarts the firmware. IMPORTANT: The command should be used with precautions since it does not send 'leaving the network' signals to other nodes, so the command can affect PAN's integrity. If automatic networking is disabled then the node will not join PAN automatically. If Z command is not the last in command line the command disables processing of the subsequent commands in command line. Result code is sent upon the reset process is completed. During the reset process some transients can be observed on the module pins (including GPIO) because of the nature of the MCU used. It is strongly recommended to wait until **OK** result code is received (or an equivalent numerical code 0 if verbose result codes are disabled by V0 command, see 3.6.6) before sending any new command to the module.
Result codes	**OK** is always returned
Example	ATZ **OK**
Node types	Coordinator/Router/End-device

3.5.9. "&F" – Set to factory-defined configuration

Syntax	Explanation
`&F`	The command instructs the module to set all the parameters (including the persistent variables from EEPROM) to the factory defaults. This command forces hardware reset like the `Z` command does, so all the precautions in 3.5.8 should be considered. Result code will be issued according to result code suppression setting (see 3.6.5), response formatting (see 3.6.6) and the transmission rate (see 3.6.8) set before execution of this command.
Result codes	`OK` is always returned
Example	`AT&F` `OK`
Node types	Coordinator/Router/End-device

3.6. +Host interface commands

3.6.1. "S3" – Termination character

Syntax	Explanation
`S3=<value>`	The command sets ASCII code to be used as termination character in command line, response and result code formatting. `value` may be specified in the range of 0…127. NOTE: It is strongly recommended to avoid changing of this parameter during the network operation.
`S3?`	The command requests for actual ASCII code currently used as the termination character.

Syntax	Explanation
Result codes	The module returns **OK** if `value` is in range, otherwise **ERROR**. IMPORTANT: It is the previous value of `S3` which is used in entering the command line containing the `S3` setting command to specify the next command line termination character. However, the result code when issued will use the value of S3 as that one set during the processing of the command line. For example, if `S3` was previously set to 13 and the '`ATS3=30`' command line is issued, the command line will be terminated with a CR character, but the result code when issued will use the character with the decimal value 30 instead of **<CR>**.

Example	`ATS3=13`
	OK
	`ATS3?`
	13
	OK

Node types	Coordinator/Router/End-device
Default value	13 - **<CR>** (carriage return character)
Persistence	`value` is stored in the EEPROM.

3.6.2. "S4" – Response formatting character

Syntax	Explanation
`S4=<value>`	The command sets ASCII code of character to be used in responses and result code formatting along with the `S3` parameter (see 3.6.1). The description of `V` command shows the parameter usage, see 3.6.6 for details. `value` may be specified in the range of 0…127. NOTE: It is strongly recommended to avoid changing of this parameter during the network operation.
`S4?`	The command requests for actual ASCII code currently used as the response formatting character.
Result codes	The module returns **OK** if `value` is in range, otherwise **ERROR**. NOTE: The changed value of `S4` will be used in formatting of the result code and information responses immediately after processing the '`S4=<value>`' command. If the value of `S4` is changed in a command line, the result code issued in response to that command line will use the new value of `S4`.

Syntax	Explanation
Example	ATS4=10 **OK** ATS4? **10** **OK**
Node types	Coordinator/Router/End-device
Default value	`10 - <LF>` (Line Feed character)
Persistence	`value` is stored in the EEPROM.

3.6.3. "S5" – Command editing character

Syntax	Explanation
`S5=<value>`	The command sets ASCII code to be used as the control character pointing to delete the character just having been input in the command line, see 3.1.3. `value` may be specified in the range of 0…127.
`S5?`	The command requests for actual ASCII code of the command editing character.
Result codes	The module returns **OK** if `value` is in range, otherwise **ERROR**. NOTE: The changed value of `S5` will be used in editing of subsequent command lines and will be applied after processing the line containing `S5` register change.
Example	ATS5=8 **OK** ATS5? **8** **OK**
Node types	Coordinator/Router/End-device
Default value	`8 - <BS>` (Backspace Character)
Persistence	`value` is stored in the EEPROM.

3.6.4. "E" – Command echo

Syntax	Explanation
`E[<value>]`	Setting this parameter instructs if the module should echo the characters received from UART. `value` may be specified as 0 or 1 to disable or enable echoing, correspondingly. If `value` is omitted 0 is implied by default.

<table>
<tr><th>Syntax</th><th colspan="2">Explanation</th></tr>
<tr><td>Result codes</td><td colspan="2">The module returns <code>OK</code> if <code>value</code> is 0 or 1, otherwise <code>ERROR</code>.</td></tr>
<tr><td>Example</td><td>ATE
OK
ATE1
OK</td><td>Disable echo

Enable echo</td></tr>
<tr><td>Node types</td><td colspan="2">Coordinator/Router/End-device</td></tr>
<tr><td>Default value</td><td colspan="2"><code>1</code> - echoing is enabled</td></tr>
<tr><td>Persistence</td><td colspan="2"><code>value</code> is stored in the EEPROM.</td></tr>
</table>

3.6.5. “Q” – Result code suppression

<table>
<tr><th>Syntax</th><th colspan="2">Explanation</th></tr>
<tr><td><code>Q[&lt;value&gt;]</code></td><td colspan="2">Setting this parameter instructs if the module should transmit the result codes to UART. When result codes are being suppressed, no portion of any intermediate, final, or unsolicited result code – header, result text, line terminator, or trailer (see 2.4, and Table 11) – is transmitted. Information text transmitted in response to a command is not affected by setting of this parameter.
There are two possibilities for <code>value</code> :
<code>0</code> The module transmits result codes.
<code>1</code> Result codes are suppressed so not transmitted.
If <code>value</code> is omitted, 0 is implied.</td></tr>
<tr><td>Result codes</td><td colspan="2">Nothing will be received for <code>ATQ1</code> command,
<code>OK</code> if <code>value</code> is 0, otherwise the module returns <code>ERROR</code>.</td></tr>
<tr><td>Example</td><td>ATQ0
OK
ATQ1</td><td>Enable the result codes

Suppress the result codes. No <code>OK</code> will be sent because it is suppressed</td></tr>
<tr><td>Node types</td><td colspan="2">Coordinator/Router/End-device</td></tr>
<tr><td>Default value</td><td colspan="2">0 – enables result codes</td></tr>
<tr><td>Persistence</td><td colspan="2"><code>value</code> is stored in the EEPROM.</td></tr>
</table>

3.6.6. “V” – Response format

Syntax	Explanation	
`V[<value>]`	Setting this parameter defines the contents of header and trailer transmitted with result codes and information responses. It also determines whether result codes are transmitted in numeric, alphabetic, or "verbose", form. The text portion of information responses is not affected by this setting. Table 11 shows the effect of the setting of this parameter on the format of information text and result codes. If `value` is omitted, `0` is implied.	

Result codes	`0`
	`OK`
	`4`
	`ERROR`
		If `value` is `0` (because numeric response text is being used)
		If `value` is `1`.
		For unsupported values (if previous `value` was `0`).
		For unsupported values (if previous `value` was `1`).

Example	`ATV1`
	`OK`
	`ATV0`
	`0`
		`0` will be output on the same line because `<LF>` is not used for formatting of result code!

Node types	Coordinator/Router/End-device	
Default value	`1` – verbose format	
Persistence	`value` is stored in the EEPROM	

Table 11 below summarizes the usage of response formats. All references to `<CR>` mean "the character ASCII coded as specified in parameter `S3` (see 3.6.1)"; all references to `<LF>` likewise mean "the character ASCII coded as specified in parameter `S4` (see 3.6.2)". Numeric and verbose codes are discussed in 2.4.

Table 11. Response Formatting

Value	0	1
Information responses	<text>`<CR><LF>`	`<CR><LF>`<text>`<CR><LF>`
Result codes	<numeric code>`<CR>`	`<CR><LF>`<verbose code>`<CR><LF>`

3.6.7. "X" – Result code selection

Syntax	Explanation
`X[<value>]`	Setting this parameter defines whether the module transmits particular result codes (see 2.4) to the host, or it does not. `value` Description `0` all result codes are sent to the host `1` **EVENT** result codes are not sent `2` **EVENT** and **DATA** result codes are not sent If `value` is omitted, `0` is implied.
Result codes	**OK** if `value` is from valid range. Otherwise, **ERROR** is returned.
Example	`ATX2` **OK** — Disable events and data indications
Node types	Coordinator/Router/End-device
Default value	`1` – all result codes will be sent, excluding **EVENT**.
Persistence	`value` is stored in the EEPROM.

3.6.8. "+IPR" – Serial port communication rate

Syntax	Explanation
`+IPR=<value>`	The command specifies the data rate at which the DCE will accept commands and will respond. At least, 1200 bit/s and 9600 bit/s are supported, but particular hardware version can support extended set of rates. NOTE: The rate specified takes effect following the issuance of any result code associated with the current command line even subsequent commands in a command line will return **ERROR**.
`+IPR?`	The command requests for actual communication rate.
`+IPR=?`	The command requests for the list of supported rates. This depends on the hardware capabilities of the particular model.
Result codes	The module returns **OK** if the requested rate is present in the supported list, otherwise **ERROR**.

Syntax	Explanation
Example	AT+IPR=38400 **OK** AT+IPR? **+IPR:38400** **OK** AT+IPR=? **+IPR:(1200,9600,38400)**
Node types	Coordinator/Router/End-device
Default value	Depends on the hardware version. For MeasBean2 boards it is 38400
Persistence	`value` is stored in the EEPROM

3.6.9. “+IFC” – Serial port flow control

Syntax	Explanation

`+IFC=<rx_flow>, <tx_flow>`	The command is used to specify the methods for local flow control over the UART interface between the host and the module. It accepts two numeric sub-parameters:
	● `rx_flow`, which specifies the method the host to control the flow of data received from the module
	● `tx_flow`, which specifies the method the module to control the flow of data transmitted from the host
	`rx_flow`
	`0` None
	`2` use RTS (Request to Send) line
	`tx_flow`
	`0` None
	`2` use CTS (Clear to Send) line
	NOTE:
	It is strongly recommended to use the CTS method because, if no flow control method is selected, there would be no means to use power-down modes when the module would not accept any data coming to UART.

`+IFC?`	The command requests for actual flow control settings.
`+IFC=?`	The command requests to list the flow control settings supported.

Syntax	Explanation
Result codes	**OK** is returned if specified flow control combinations are supported, otherwise **ERROR**.
Example	AT+IFC=2,2 **OK** AT+IFC? **+IFC:2,2** **OK** AT+IFC=? **(0,2),(0,2)**
Node types	Coordinator/Router/End-device
Default value	Depends on the hardware version. For MeshBean2 boards it is 0,0
Persistence	value is stored in the EEPROM

3.6.10. "&D" – DTR behavior

Syntax	Explanation
&D<value>	The command specifies the method how the module manages DTR line. value Description 0 module ignores DTR line 1 module wakes up if it is sleeping, thus it can process the data coming from UART with a shortest delay
S-register	S50 (RW).
Result codes	**OK** is returned if the requested mode is supported, otherwise **ERROR**.
Example	AT&D1 **OK**
Node types	Coordinator/Router/End-device
Default value	0
Persistence	value is stored in the EEPROM.

3.6.11. S0 – Request for the latest result code

<table>
<tr><th>Syntax</th><th colspan="2">Explanation</th></tr>
<tr><td>S0?</td><td colspan="2">Request for result code from the latest executed command. If the latest executed command was completed with <b>ERROR</b> result code, register S0 will contain nonzero value.
Returned values:
0 no error
1 syntax error
2 improper number of parameters
3 parameter value(s) is out of range (example: AT+IFC=12,34)
4 unspecified error
5 requested value cannot be read (example: +WLQI command for non-existent link)
6 operation is not permitted in current state (example: setting PAN ID in the connected state or +WSLEEP for router)
7 operation cannot be completed due to networking problems, e.g. due to connection loss
8 data transmission error</td></tr>
<tr><td>Result codes</td><td colspan="2">Always OK</td></tr>
<tr><td>Example</td><td>AT+WROLE=0+WPWR=30,30
<b>ERROR</b>
ATS0?
<b>6</b>
<b>OK</b>
AT+ABCD
<b>ERROR</b>
ATS0?
<b>1</b>
<b>OK</b>
AT+IFC=12,34
<b>ERROR</b>
ATS0?
<b>3</b>
<b>OK</b></td><td>6 is returned as setting +WPWR is not permitted for coordinator
syntax error
parameter is out of range</td></tr>
<tr><td>Node types</td><td colspan="2">Coordinator/Router/End-device</td></tr>
</table>

3.7. Parameters

3.7.1. "+WTIMEOUT" – Data delivery timeout

Syntax	Explanation
`+WTIMEOUT=<value>`	The parameter assigns the module's waiting timeout for getting acknowledgement on the data transmission, before starting retransmission. The parameter is specified in milliseconds. NOTES: The parameter corresponds to the `apscAckWaitDuration` variable introduced by ZigBee recommendation [1]. In the eZeeNet configurations, the `value` parameter set by this command will be applied upon warm reset is completed.
`+WTIMEOUT?`	The command returns the actual timeout `value`.
`+WTIMEOUT=?`	The command requests for the range of valid timeouts.
S-register	`S51` (RW).
Result codes	**`OK`** is returned if `value` is in range, otherwise **`ERROR`** is returned.

Example	`AT+WTIMEOUT=200`
	`OK`
	`AT+WTIMEOUT?`
	`+WTIMEOUT:200`
	`OK`
	`AT+WTIMEOUT=?`
	`+WTIMEOUT:(10-3000)`
	`OK`

Default value	`1000`
Persistence	`value` is stored in the EEPROM
Node types	Coordinator/Router/End-device

3.7.2. "+WRETRY" – Repetition count

Syntax	Explanation
`+WRETRY=<value>`	The parameter assigns the module's limiting number of retries allowed in case of transmission failure. NOTES: The parameter corresponds to the `apscMaxFrameRetries` variable introduced by ZigBee recommendation [1]. In the eZeeNet configurations, the `value` parameter set by this command will be applied upon warm reset is completed.
`+WRETRY?`	The command returns actual number of retransmission.
`+WRETRY=?`	The command requests for the range of valid values
S-register	`S52` (RW).
Result codes	**OK** is returned if `value` is in range, otherwise **ERROR** is returned.

Example	`AT+WRETRY=1`
	OK
	`AT+WRETRY?`
	+WRETRY:1
	OK
	`AT+WRETRY=?`
	+WRETRY:(0-5)
	OK

Default value	3
Persistence	`value` is stored in the EEPROM
Node types	Coordinator/Router/End-device

3.7.3. "+WWAIT" – Data transmission waiting timeout

Syntax	Explanation
`+WWAIT=<value>`	The `value` parameter sets the timeout (in milliseconds) for the module to wait for entering the `D` (see 3.4.1) or the `DU` (see 3.4.2) command. Then, if a pause between two consecutive characters coming from UART exceeds the timeout specified, the module will start data transmission even though the data length encountered has not yet reached the number specified by the `length` argument of the `D/DU` commands considered. In such case, the `length` is replaced with its actual value according to the data transmitted.
`+WWAIT?`	The command returns actual timeout `value`.
`+WWAIT=?`	The command requests for the range of valid timeouts.
S-register	`S53` (RW).
Result codes	**`OK`** is returned if the `value` is in range, otherwise **`ERROR`** is returned.

Example	`AT+WWAIT=500`
	`OK`
	`AT+WWAIT?`
	`+WWAIT:500`
	`OK`
	`AT+WWAIT=?`
	`+WWAIT:(100-5000)`
	`OK`

Default value	`5000`
Persistence	`value` is stored in the EEPROM.
Node types	Coordinator/Router/End-device

3.7.4. "+WSRC" – Read/Write logical address

Syntax	Explanation
`+WSRC=<addr>`	The parameter sets the address for a node which will be used for communications between applications, depending on status of `S30` register. If `S30` register is set to 0 logical `addr` is set for the node as a 16 bit hexadecimal number which should be unique within particular PAN. If `S30` register is set to 1 +WSRC returns NWK address of the node. See Section 3.3.9 for details. NOTES: Logical address will be applied during network join procedure, setting a new address thus requires rejoining. Default address `0` is used typically for Coordinator. If the only one data collection point exists in the network, this address can be used as a 'data sink'.
`+WSRC?`	The command returns the actual logical address.
`+WSRC=?`	The command requests for the range of valid addresses.
S-register	`S55` (RW).
Result codes	**`OK`** is returned if `value` is in range, otherwise **`ERROR`** is returned.

Example	`AT+WSRC=2ABC`
	`OK`
	`AT+WSRC?`
	`+WSRC:2ABC`
	`OK`
	`AT+WSRC=?`
	`+WSRC:(0000-FFFF)`
	`OK`

Default value	0
Persistence	`addr` value is stored in the EEPROM.
Node types	Coordinator/Router/End-device

3.8. GPIO

3.8.1. GPIO configuration

<table>
<tr><th>Syntax</th><th colspan="2">Explanation</th></tr>
<tr><td>S&lt;reg&gt;=<value&gt;</td><td colspan="2">Command selects configuration of particular GPIO pins. reg corresponds to GPIO pins, GPIO0…GPIO8, on the module and it is in the range of 120…128.
value Description
0 input pin, no internal pull-up
3 output
2 tri-state
1 input pin, internal pull-up is turned on
NOTES:
Using of internal pull-up improves noise immunity but take in mind that it results in power consumption increased.
On the MeshBean2 board, tri-stated pins are configured as input with no pull-up.</td></tr>
<tr><td>S&lt;reg&gt;?</td><td colspan="2">The command requests for actual GPIO pin configuration.</td></tr>
<tr><td>Result codes</td><td colspan="2">OK is returned if the value is in valid range, otherwise ERROR is returned.</td></tr>
<tr><td>Example</td><td>ATS120=1 S121=3
OK</td><td>Set GPIO0 as input with internal pull-up and GPIO 1 as output</td></tr>
<tr><td>Default value</td><td colspan="2">2, tri-state</td></tr>
<tr><td>Persistence</td><td colspan="2">Values are stored in the EEPROM.</td></tr>
<tr><td>Node types</td><td colspan="2">Coordinator/Router/End-device</td></tr>
</table>

3.8.2. GPIO

<table>
<tr><th>Syntax</th><th colspan="2">Explanation</th></tr>
<tr><td>S&lt;reg&gt;=<value&gt;</td><td colspan="2">The command assigns value to a particular GPIO pin. Each of pins GPIO0…GPIO8 of the module is numbered by reg which is in the range of 130…138, correspondingly.
<table><tr><td>&lt;value&gt;</td><td>Description</td></tr><tr><td>0</td><td>Logical 0</td></tr><tr><td>1</td><td>Logical 1</td></tr></table>
NOTE:
Command does not affect any pin configured as input or tri-state.</td></tr>
<tr><td>S&lt;reg&gt;?</td><td colspan="2">The command reads a particular GPIO pin numbered and coded as above, so it returns 0 or 1. If pin is configured for output or as tri-state, returned value is not defined</td></tr>
<tr><td>Result codes</td><td colspan="2"><b>OK</b> is returned if value is 0 or 1, otherwise <b>ERROR</b> is returned.</td></tr>
<tr><td>Example</td><td>ATS120=1 S121=3
ATS130?
<b>1</b>
<b>OK</b>
ATS131=0
<b>OK</b></td><td>Set GPIO0 as input and GPIO1 as output, both with internal pull-up
GPIO0 is 1
Clear GPIO1</td></tr>
<tr><td>Default value</td><td colspan="2">Not defined</td></tr>
<tr><td>Persistence</td><td colspan="2">Values are not stored in the EEPROM because GPIO pins are configured as tri-state at the startup.</td></tr>
<tr><td>Node types</td><td colspan="2">Coordinator/Router/End-device</td></tr>
</table>

3.8.3. A/D configuration

Syntax	Explanation	
S100=<value>	The command selects configuration of particular A/D pins. value is a hexadecimal number containing a bit-field. 8 least significant bits (b0... b3) enable or disable each A/D channel. If bit is cleared then A/D conversion of a corresponding channel is disabled and A/D pin goes to the high impedance without internal pull-up. NOTES: Take in mind that enabling A/D conversion increases power consumption. Conversion is executed in single conversion mode (see ATmega datasheet [10]) with 125 kHz clock rate and external reference, thus enabling the maximum conversion rate of approximately 5 kbps. Proper conversion results are achieved for ZigBit if the external reference signal of 1.25V is applied to the A_VREF pin. If conversion is disabled on all A/D pins, the A_VREF pin is moved to tri-state. Pins AD4…AD7 can be also used as JTAG port and ADC function for this inputs are disabled. When using the ZigBit module installed on the MeshBean2 board, the following restriction is imposed due to the board schematics. Before configuring or reading of the particular A/D pins, you must configure GPIO 6, GPIO 7 and GPIO 8 for output, then set GPIO to 0 while setting GPIO7 and GPIO8 to 1. For example, you must send the following commands: ATS126=3 S127=3 S128=3 ATS136=0 S137=1 S138=1 before performing ATS100=0F See additionally Section 3.8.4.	
S<reg>?	The command requests for actual A/D configuration.	
Result codes	OK is always returned.	
Example	ATS100=8 OK	Enable conversion on pin AD3
Default value	0 – disable A/D conversion	
Persistence	Value is stored in the EEPROM.	
Node types	Coordinator/Router/End-device	

3.8.4. A/D

<table>
<tr><th>Syntax</th><th colspan="2">Explanation</th></tr>
<tr><td>S<reg>?</td><td colspan="2">The command reads particular A/D pin and returns its value in decimal format. reg corresponds to pins AD0…AD3 on the module and it is in the range of 101…104. If A/D conversion for particular channel is disabled by the S100 register, no value is returned.
NOTE:
When using the ZigBit module installed on the MeshBean2 board, the following restriction is imposed due to the board schematics. Configure GPIO 6, GPIO 7 and GPIO 8 for output. Set GPIO to 0 while setting GPIO7 and GPIO8 to 1. Then you can configure or read the particular A/D pins. For example, you must send the following commands:
ATS126=3 S127=3 S128=3
ATS136=0 S137=1 S138=1
before performing these commands:
ATS100=0F
ATS101? S102? S103? S104?</td></tr>
<tr><td>Result codes</td><td colspan="2">OK is always returned .</td></tr>
<tr><td>Example</td><td>ATS100=8
OK
ATS103?
125
OK</td><td>Enable conversion on pin AD3
Read AD3 pin</td></tr>
<tr><td>Node types</td><td colspan="2">Coordinator/Router/End-device</td></tr>
</table>

3.9. Remote management

3.9.1. "+WPASSWORD" – Set a password

Syntax	Explanation
`+WPASSWORD <psw>`	The command sets a new password for remote management command. Password is in form of 32-bit hexadecimal number.
Result codes	**OK** is always returned .
Example	`AT+WPASSWORD 65432178` **OK**
Default value	`0`
Persistence	`psw` value is stored in the EEPROM. NOTE: The password cannot be reloaded with default value through `&F` command (see 3.5.9) but it can be rewritten over the air using remote AT-command (see 3.9.2).
Node types	Coordinator/Router/End-device

3.9.2. "R" – Remote execution of AT command

Syntax	Explanation
`R<addr>,<psw>,<cmd>`	The command lets the execution of AT-commands on a remote node, with output redirected. Password (`psw`) is a 32-bit hexadecimal number, which is set for this specific node. `addr` should be a 16-bit hexadecimal logical address if the status of `S30` register is set to 0. `addr` should be a NWK address if the status of S30 register is set to 1. See Section 3.3.9 for details. `cmd` is a sequence of AT-commands without `AT` prefix.
Result codes	All the responses and result codes are received from the remote node in text form thus can be normally processed. If a connection loss will be detected, the **ERROR** result code will be returned after timeout since last response packet is received (approx 3 sec). If remote command is send to End-device with sleeping period longer than timeout, **ERROR** will be returned. If the controlled node is not in the PAN, **ERROR** will be returned. Remote execution is not allowed for commands that cause sending data throughout the network: `D`, `DU`, `DS`, `+WPING`, `R`. Attempting will result in **ERROR** code with the command processing aborted.

Syntax	Explanation	

Example	`ATR0,65432178,+GMM?+WRSSI 2`
	`+GMM:ZIGBIT`
	`+WRSSI:-80`
	`OK`
	`ATR0,65432178,+WLEAVE`
	`ERROR`
		Get model number and RSSI
		Remove node from network – **`ERROR`** will be returned but delayed.

Node types	Coordinator/Router/End-device	