ALCIOM
3 rue des Vignes
78220 VIROFLAY - FRANCE

Tel : 01 4709 3051 / Fax : 01 3024 1796
✉ contact@alciom.com

www.alciom.com

PROJECT :

WIZE'UP

DOCUMENT :

AT command interpreter (ATCI) Specification

REFERENCE :

AL/RL/2031/008

DATE :	VERSION :	AUTHOR :
April, 13th, 2021	**1E**	**R.Lacoste & P. Rousseau / ALCIOM**

SUMMARY :

This documents is the specification of the AT command interpreter (ATCI) compativle with the Wize'Up 169MHz Wize module developped by ALCIOM under a grant contrat by the WIZE ALLIANCE, and based on the open-source Wize protocol stack developped by GRDF.

DOCUMENT HISTORY

DATE	VERSION	AUTHOR	COMMENT
31/07/2020	1A	R.Lacoste / ALCIOM	Draft
6/8/2020	1B	R.Lacoste / ALCIOM	After initial review
12/01/2021	1C	P.Rousseau / ALCIOM	Update
01/03/2021	1D	P.Rousseau / ALCIOM	Add ATFC command
13/04/2021	1E	P.Rousseau / ALCIOM	Update ATFC and add ATTEST command

TABLE OF CONTENTS

1 Introduction

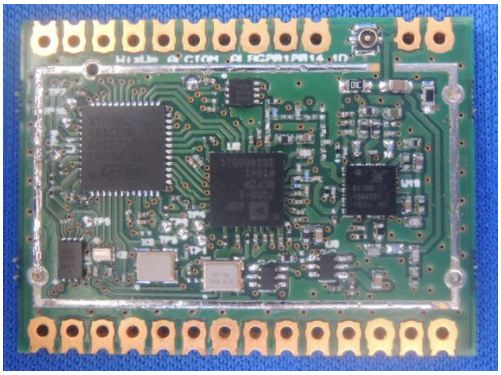
1.1 Document content

This document is the specification of the AT command interpreter (ATCI), which could be used to drive a Wize'Up module and its accompanying Wize open-source stack by an externa host through a simple ASCII-coded UART interface.

1.2 Wize'Up presentation

Wize is a long range low power wide aera networks (LPWAN) protocol, promoted and standardized by the Wize Alliance association (www.wize-alliance.com). This protocol, based on European standard EN13757-4/N2, uses a VHF low frequency band (169MHz in Europe) and provides deep indoor penetration, particularly useful for smart metering applications.

Wize'Up is a high-performance Wize-compatible module, developped by ALCIOM (www.alciom.com) under a grant contrat by the Wize Alliance. Wize'Up is provided by ALCIOM as **open source hardware** (see licence here under). The full design, including schematics, gerbers and test reports are provided free of charge by ALCIOM and could be used for any Wize compatible product (*).



Wize'Up module

Wize'Up evaluation board

Wize'Up can either :

- Be directly purchased as a plug-in module from a Wize'Up module manufacturer, as ALCIOM, and integrated in any end products ;
- Be duplicated by any manufacturer, and integrated as a Wize'Up module in any end product (*) ;
- Be modified by any design house in order to make a customized Wize-compatible device based on the Wize'Up architecture (*). ALCIOM, as a design house, can of course help you in that job through design services.

(*) Assuming the conditions of the licence are respected, see here under (1.3).

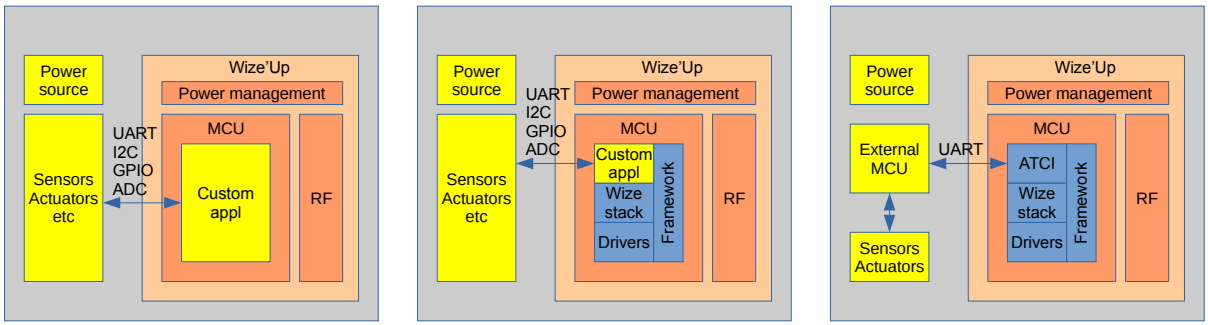
Wize'Up is a hardware project and is software-agnostic, but it is designed to be compatible with the **open-source Wize protocol stack and framework** developped and provided independantly by GRDF.

1.3 Wize'Up software support

In terms of software, Wize'Up can be either :

- Programmed with any custom software developped from scratch by a third party, based on Wize'Up publicaly available schematic and documentation. This software then run directly on the Wize'Up on-board processor ;
- Programmed with a custom application developped by a third party but using the open-source Wize stack and framework provided by GRDF, both interfaced through the API specified in the accompanying documentation. In that case, both the application and the Wize stack are running on the Wize'Up on-board processor and could be interfaced to any external sensor or co-processor ;
- Programmed with a standard application provided as part of the open source Wize stack and framework distribution, and interfaced with any external host processor (microcontroler, PC, etc), which then run the application itself. This application includes the Wize stack and framework and an AT command interpreter (ATCI), co-developped by ALCIOM and GRDF, which allows to access to the features of the Wize stack from the external processor through a simple UART link.

These three modes are illustrated here below :



1.4 Reference documents

Specification, pinout and usage of the Wize'Up module and its standard evaluation kit, is provided in document [R1] :

[R1] Title : Wize'Up – Technical Manual
Reference : AL/RL/2031/007
Version : 1B

1.5 Licence and warranty

This document and all Wize'Up related sources and documentations describes Open Hardware and are licensed under the CERN-OHL-P v2. You may redistribute and modify this documentation and make products using it under the terms of this licence (https://cern.ch/cern-ohl).

As per CERN-OHL-P v2 section 4, should you produce hardware or documentation based on these sources, you must maintain the Source Location clearly visible on the printed-circuit board of the product as well as on any documents you make using this documentation and sources. Source location: www.alciom.com/wizeup.

This documentation and all Wize'Up related sources and documentations are distributed WITHOUT ANY EXPRESS OR IMPLIED WARRANTY, INCLUDING OF MERCHANTABILITY, SATISFACTORY QUALITY AND FITNESS FOR A PARTICULAR PURPOSE, in particular from ALCIOM, GRDF or WIZE ALLIANCE. Please see the CERN-OHL-P v2 for applicable conditions.

2 AT command principles

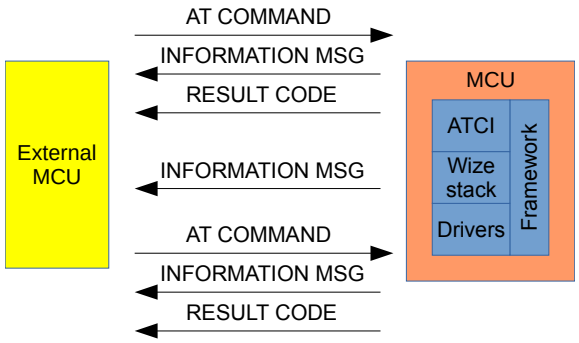
2.1 UART configuration

The standard UART configuration for communication with Wize'Up using ATCI is the following :

- UART type : RX/TX only
- Electrical levels : 2,5V LVTTL, not inverted
- Baud speed : 115 200 bps
- Frame format : 8 bits, no parity, one stop bit
- Flow control : None

2.2 Basic organization format

When using ATCI, the Wize'up module is a slave of the external host processor running the application. It receives **AT commands** through the UART, and respond to each AT command with a **result code.** Additionnally the Wize'Up module can send **information messages**, linked or not with a given AT command. The host processor must wait for the result code of a given AT command before sending another command.



2.3 AT commands format

The generic format of an AT command is one of the following :

Execution command :
ATCMD<CR>
Prefix | Command name

Execution command with parameters :
ATCMD=11,252,$E2,11<CR>
Prefix | Command name | Parameters

Read command :
ATCMD?<CR>
Prefix | Command name | Read flag

Read command with parameter :
ATCMD=11?<CR>
Prefix | Command name | Param | Read flag

All received characters others than digits, letters, '=', '$', ' ?','&' and <CR> are ignored.

2.4 Result codes format

Each AT command is answered by ATCI with one of the following result codes :

Correct execution
<CR><LF>OK<CR><LF>

Error
<CR><LF>ERROR<CR><LF>

Error with supplied error code
<CR><LF>ERROR:22<CR><LF>

2.5 Information message format

In addition to result codes, ATCI may send to the host additionnal information messages, related or not to an AT command execution. The format of these information messages is one of the following :

Information message related to an AT command execution :
<CR><LF>+CMD:252,33,"HELLO"<CR><LF>
Prefix | Command name | Information data

Information message not related to an AT command execution :
<CR><LF>+INF:252,33,"HELLO"<CR><LF>
Prefix | Information code | Information data

Debug messages :
<CR><LF>+DBG:252,33,"HELLO"<CR><LF>
Prefix | Debug code | Debug data

2.6 Numeric formats

All parameters to AT commands are either hexadecimal (dollar sign followed by any number of pairs of hexadecimal digits), or decimal for 1-byte parameters only (1 to 3 numeric digits).

2.7 Power management

In order to reduce its power consumption, Wize'Up sends an information message (+SLEEP) and switch to sleep mode after 5 seconds of inactivity or when receiving a sleep command (ATQ). To wake up the module, it is required to send any character through the UART. This character may be lost or processed, so it is recommanded to send a <CR> which will in any case not be processed by ATCI. An information message (+WAKEUP) is send upon wakeup.

Example :

→ <CR><LF>+SLEEP<CR><LF>
← <CR>
→ <CR><LF>+WAKEUP<CR><LF>
(5s)
→ <CR><LF>+SLEEP<CR><LF>

3 AT commands

3.1 Generic commands

3.1.1 AT-Connexion check

Command	AT
Semantics	Check if ATCI is up and running
Parameters	None
Information responses	None
Error codes	None

Example :

← AT<CR>
→ <CR><LF>OK<CR><LF>

3.1.2 ATI-Request identification

Command	ATI
Semantics	Queries the identification of the module
Parameters	None
Information responses	+ATI :"WIZEUP",<manufacturer>,<model>,<hw version>,<major sw version>,<minor sw version>
Error codes	None

Example :

← ATI<CR>
→ <CR><LF>+ATI :"WIZEUP","ALCIOM","WZ1000",001,001,000<CR><LF>
→ <CR><LF>OK<CR><LF>

3.1.3 ATZ - Reset module

Command	ATZ
Semantics	Reset the Wize'Up module (cold boot), and restore all registers to their last stored values (AT&W)
Parameters	None
Information responses	None
Error codes	None

Example :

← ATZ<CR>
→ <CR><LF>OK<CR><LF>
→ <CR><LF>+WAKEUP<CR><LF>

3.1.4 ATQ – Go to sleeep mode immediately

Command	ATQ
Semantics	Switch immediately the Wize'Up module to low power mode. This allows to avoid waiting for 5s and thus reduce the overall power consumption.
Parameters	None
Information responses	None
Error codes	None

Example :

← ATQCR>
→ <CR><LF>OK<CR><LF>
→ <CR><LF>+SLEEP<CR><LF>

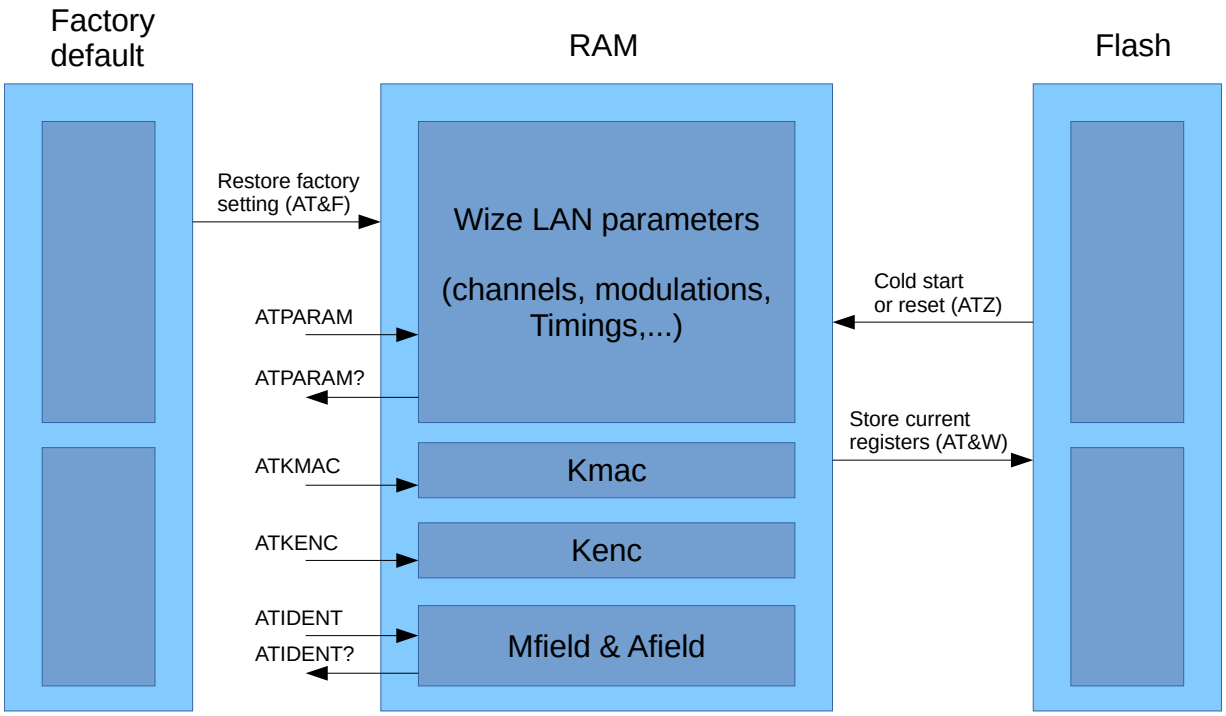
3.2 Register access commands

3.2.1 Registers principles

ATCI allows to configure the Wize'Up module using a register model : All required Wize stack settings are stored in a register file, which can be read and written through AT commands. More specifically, these registers are split into two groups :

- Registers which directly map to the LAN parameter dictionnary specified by the Wize standard (RF channels and modulations, TX power, timing paramters, etc) ;
- Additionnal registers specific to the Wize'Up module :
  - Module identity (Mfield and Afield)
  - Ciphering keys (Kmac and Kenc)

These registers have factory-preset values, and could be modified by AT commands. The modified values are stored in RAM, but could be stored in non-volatile flash thanks to a specific AT command. The flash-stored values are restored in RAM at power-up. The following diagram illustrates these register storage areas and the corresponding AT commands :



3.2.2 AT&F – Restore registers to their factory settings

Command	AT&F
Semantics	Restore all registers (Wize LAN parameters and ciphering keys) to their factory settings
Parameters	None
Information responses	None
Error codes	01 : Wrong number of parameters for this command

Example :

← AT&F<CR>
→ <CR><LF>OK<CR><LF>

3.2.3 AT&W – Store current registers values in flash

Command	AT&W
Semantics	Store the current content of all registers (Wize LAN parameters and ciphering keys) in non volatile memory. The registers will be restored to their current values at the next power cycle or ATZ command
Parameters	None
Information responses	None
Error codes	01 : Wrong number of parameters for this command

Example :

← AT&W<CR>
→ <CR><LF>OK<CR><LF>

3.2.4 ATPARAM – Modify the value of a Wize LAN parameter

Command	ATPARAM
Semantics	Modify the value of a Wize LAN parameter in RAM. This value will be immediately used for all future commands, and could be stored in non volatile memory using AT&W command
Parameters	Wize parameter adress Wize parameter value
Information responses	None
Error codes	01 : Wrong number of parameters for this command 02 : Wrong length of a parameter for this command 03 : Illegal parameter value for this command

Example :

← ATPARAM=$0A, 2<CR> (uplink modulation = WM_HSPEED)
→ <CR><LF>OK<CR><LF>
← ATPARAM=$0A,$1234<CR> (uplink modulation = two bytes rather than one)
→ <CR><LF>ERROR:02<CR><LF>

3.2.5 ATPARAM ? - Read the value of Wize LAN parameters

Command	ATPARAM ?
Semantics	Read the value of one or all Wize LAN parameter in RAM.
Parameters	Wize parameter adress Nota : All parameters are read if no adress is supplied
Information responses	+ATPARAM : <adress>, <value>
Error codes	01 : Wrong number of parameters for this command 02 : Wrong length of a parameter for this command 03 : Illegal parameter value for this command

Example :

← ATPARAM=$FA?<CR> (request parameter $FA)
→ <CR><LF>ERROR:03<CR><LF> (non existant)
← ATPARAM=$0A?<CR> (uplink modulation ?)
→ <CR><LF>+ATPARAM:$0A,$04<CR><LF> (uplink modulation = WM_HSPEED)
→ <CR><LF>OK<CR><LF>
← ATPARAM ? <CR> (request all parameters)
→ <CR><LF>+ATPARAM:$01,$1710<CR><LF>
→ <CR><LF>+ATPARAM:$02,$04<CR><LF>
→ ...
→ <CR><LF>OK<CR><LF>

3.2.6 ATKMAC – Modify the value of the Kmac key

Command	ATKMAC
Semantics	Modify the value of the KMAC key in RAM. This value will be immediately used for all future commands, and could be stored in non volatile memory using AT&W command
Parameters	New key (16 bytes)
Information responses	None
Error codes	01 : Wrong number of parameters for this command 02 : Wrong length of a parameter for this command

Example :

← ATKMAC=$10111213141516171819 1A1B1C1D1E1F<CR> (new key, 128 bits)
→ <CR><LF>OK<CR><LF>

3.2.7 ATKENC – Modify the value of the Kenc key

Command	ATKENC
Semantics	Modify the value of the KENC key in RAM. This value will be immediately used for all future commands, and could be stored in non volatile memory using AT&W command
Parameters	Key number (1 byte) New key (16 bytes)
Information responses	None
Error codes	01 : Wrong number of parameters for this command 02 : Wrong length of a parameter for this command

Example :

← ATKENC=$01,$10111213141516171819 1A1B1C1D1E1F<CR> (new key, 128 bits)
→ <CR><LF>OK<CR><LF>

Nota : Keys can't be read back using ATCI.

3.2.8 ATIDENT – Modify the value of Mfield and Afield

Command	ATIDENT
Semantics	Modify the value of the Mfield and Afield in RAM. These values will be immediately used for all future commands, and could be stored in non volatile memory using AT&W command
Parameters	M-field (2 bytes) A-field (6 bytes)
Information responses	None
Error codes	01 : Wrong number of parameters for this command 02 : Wrong length of a parameter for this command

Example :

← ATIDENT=$AABB,$10111213141516<CR>
→ <CR><LF>OK<CR><LF>

3.2.9 ATIDENT ? – Read the value of Mfield and Afield

Command	ATIDENT ?
Semantics	Read the value of the Mfield and Afield from RAM
Parameters	None
Information responses	+ATIDENT:<mfield>,<afield>
Error codes	01 : Wrong number of parameters for this command

Example :

← ATIDENT ?<CR>
→ <CR><LF>+ATIDENT :$AABB,$10111213141516<CR><LF>
→ <CR><LF>OK<CR><LF>

3.3 Wize messaging commands

3.3.1 ATSEND – Send a Wize message

Command	ATSEND
Semantics	Send a Wize message using the current configuration. The stack then wait for a potential downstream message, manage autonomously any parameter change sent by the network (APP-ADMIN read/write commands), and report any
Parameters	L6App Raw L7 message bytes (unciphered)
Information responses	If an APP-ADMIN write command was received and processed by the on-board Wize stack : +ATADMWRITE :<paramid>,<paramvalue>,<rssi> If a response was received in response of the Wize message, which can't be managed by the on-board Wize stack (other application layer than APP-ADMIN) : +ATRCV:<L6App>,<raw L7 response message, unciphered>,<rssi>
Error codes	01 : Wrong number of parameters for this command 02 : Wrong length of a parameter for this command 03 : Illegal parameter value for this command

Example :

← ATSEND=$F0,$111213141516<CR> (send 6-byte message using L6App=$F0)
→ <CR><LF>OK<CR><LF> (no response received)
← ATSEND=$F0,$111213141516<CR> (send 6-byte message using L6App=$F0)
→ <CR><LF>+ATADMWRITE:$0A,$04,$0322<CR><LF> (APP-ADMIN write processed)
→ <CR><LF>OK<CR><LF> (no response received)

3.3.2 PING – Send an INSTPING request

Command	ATPING
Semantics	Send a Wize INSTPING message using the current configuration. The stack then wait for a potential INSPONG messages, and report any using +INSTPONG information responses
Parameters	None
Information responses	None
Error codes	01 : Wrong number of parameters for this command

Example :

← ATPING<CR>
→ <CR><LF>OK<CR><LF>

Nota : The received PONG messages are not reported to the host processor through notification messages, but are stored in the Wize LAN parameter registers as required by the Wize standard. The host processor must then send ATPARAM ? commands to read the results of the ATPING command (see registers $35 to $3D, chapter 5.1).

3.4 Factory configuration and tests commands

3.4.1 ATFC – Set factory configuration

Command	ATFC
Semantics	Update a system settings or send a system command.
Parameters	ID: setting ID (1 byte, see table below) VAL1 to VALn: setting values (optional, n values of any size depending on ID)
Information responses	None
Error codes	01 : Wrong number of parameters for this command 02 : Wrong length of a parameter for this command 03 : Illegal parameter value for this command

Factory configuration settings		
ID	Description	Values
0x00	Set ADF7030 output power (PA configuration) for maximum Wizeup board output power (when TX_POWER parametter is at 0)	VAL1 : coarse PA settings, from 1 to 6 (8 bits integer, see ADF7030 Datasheet) VAL2 : fine PA settings, from 3 to 127 (8 bits integer, see ADF7030 Datasheet) VAL3 : micro PA settings, from 1 to 31 (8 bits integer, see ADF7030 Datasheet)
0x01	Idem for output power 6dB under maximal power (TX_POWER to 1)	
0x02	Idem for output power 12dB under maximal power (TX_POWER to 2)	
0x10	Set power amplifier (SKY66100-11) enable or bypass mode (ADF7030 output power must be at 0dBm maximum if PA is enabled)	VAL1 : 0 to bypass power amplifier, 1 to enable it (8 bits integer)
0x20	RSSI calibration (apply a carrier at mid band frequency with -77dbm level then execute this command)	*No parameter (a read of this command ID will fail)*
0xFC	Start ADF7030 Auto-Calibration.	*No parameter (a read of this command ID will fail)*
Others	*Reserved*	

Example:

← ATFC=0,6,22,0<CR>
→ <CR><LF>OK<CR><LF>

← ATFC=$01,$06,$0A,$00<CR>
→ <CR><LF>OK<CR><LF>

3.4.2 ATFC? – Get factory configuration

Command	ATFC=<ID> ?
Semantics	Get a system settings.
Parameters	ID: setting ID (1 byte, mandatory, see table in ATFC command)
Information responses	VAL1 to VALn: setting values (n values of any size depending on ID)
Error codes	01 : Wrong number of parameters for this command 03 : Illegal parameter value for this command

Multiread command (ATFC?) is not available, only single ID read command is available. If there is no parameter to read (system command), this command return an error.

Example:

← ATFC=0?<CR>
→ <CR><LF>+ATFC:$00,$06,$16,$00<CR><LF>
→ <CR><LF>OK<CR><LF>

3.4.3 ATTEST – Set test mode

Command	ATTEST	
Semantics	Enable TX or RX test mode or disable it.	
Parameters	Test mode (8 bits integer):	
	0x00	disable test mode (RX or TX test)
	0x01	enable TX test mode, transmit a carrier
	0x02	enable TX test mode, transmit frequency deviation tone, −fDEV, in 2FSK
	0x03	enable TX test mode, transmit −fDEV_MAX in 4FSK only
	0x04	enable TX test mode, transmit +fDEV in 2FSK
	0x05	enable TX test mode, transmit +fDEV_MAX in 4FSK only
	0x06	enable TX test mode, transmit transmit preamble pattern
	0x07	enable TX test mode, transmit pseudorandom (PN9) sequence
	0x10	enable RX test mode, get copy of SPORT_CLK to EXT_I2C_SCL and SPORT_DATA to EXT_I2C_SDA
	0x11	enable RX test mode, get PREAMBLE detect on EXT_I2C_SCL and SYNCH detect on EXT_I2C_SDA
Information responses	None	
Error codes	01 : Wrong number of parameters for this command 02 : Wrong length of a parameter for this command 03 : Illegal parameter value for this command	

Transmit a CW signal:
← ATTEST=$01<CR>
→ <CR><LF>OK<CR><LF>

Stop test mode:
← ATTEST=$00<CR>
→ <CR><LF>OK<CR><LF>

Set RX test mode (with ADF7030 preamble and synchronisation words detection signals on EXT_I2C port):
← ATTEST=$11<CR>
→ <CR><LF>OK<CR><LF>

4 Other information messages

4.1 +WAKEUP – Module just out from sleep

Message	+WAKEUP
Semantics	Send when the module exit from sleep mode. Sent also at initial power on and after a forced reset (ATZ)
Parameters	None

Example :

→ <CR><LF>+WAKEUP<CR><LF>

4.2 +SLEEP – Module going to sleep

Message	+SLEEP
Semantics	Send when the module enters sleep mode.
Parameters	None

Example :

→ <CR><LF>+SLEEP<CR><LF>

5 Register map

5.1 Wize LAN parameters

Cf LAN propotol specification version 1.1, « common application layers », annex 1, reproduced here for information.

Id	Parameter name	Description	Size (bytes)	Mode	L/R	Coding	Supported by Wize'Up / ATCI ?
01	**VERS_HW_TRX**	Hardware version number of the device (or transceiver for a remote module)	2	R	L/R	Byte 1 : Version, Byte 2 : Revision	Yes, read only
02	**VERS_FW_TRX**	Software version number run by the device (or transceiver for a remote module)	2	R	L/R	Byte 1 : Version, Byte 2 : Revision	Yes, read only
03	**DATEHOUR_LAST_UPDATE**	Date/time of the last successful firmware download	4	R	L/R	EPOCH encoded on 32 bits and corresponding to the number of seconds since 1st January 2013 at 00:00: MSBs first (big endian)	No
08	**RF_UPLINK_CHANNEL**	Frequency channel to be used for all uplink message transmissions	1	R/W	L/R	100, 110, 120, 130, 140, 150 (see Regional Parameters document, UE169MHz)	Yes, R/W
09	**RF_DOWNLINK_CHANNEL**	Frequency channel to be used for all message receptions (except firmware download)	1	R/W	L/R	100, 110, 120, 130, 140, 150 (see Regional Parameters document, UE169MHz)	Yes, R/W
0A	**RF_UPLINK_MOD**	Modulation to be used for all uplink message transmissions	1	R/W	L/R	00 : WM-2400, 01 : WM-4800, 02 : WM-HSPEED (see Regional Parameters document, UE169MHz)	Yes, R/W
0B	**RF_DOWNLINK_MOD**	Modulation to be used for all message receptions (except firmware download)	1	R/W	L/R	00 : WM-2400, 01 : WM-4800, 02 : WM-HSPEED (see Regional Parameters document, UE169MHz)	Yes, R/W
10	**TX_POWER**	Transceiver nominal transmission power	1	R/W	L/R	00 : Pmax, 01 : PMax – 6dB, 02 : PMax-12dB (see Regional Parameters document, UE169MHz)	Yes, R/W
11	**TX_DELAY_FULLPOWER**	Maximum time between two COMMAND messages before the device automatically returns to maximum transmission power	2	R/W	L/R	Number of days, from 1 to 65535 Value 0000:function disabled Byte 1: MSBs Byte 2: LSBs	Yes, R/W

12	**TX_FREQ_OFFSET**	Absolute correction of transmission frequency	2	R/W	L/R	In Hertz, from -32768 (-32.768KHz) to +32767 (+32.767KHz). Signed number on 16 bits encoded in 2- complement : Byte 1: MSBs Byte 2: LSBs Note: the device can round off this value, provided that the accuracy requirements specified in chapter 5 are complied with).	Yes, R/W
18	**EXCH_RX_DELAY**	Fixed wait time after transmission of a DATA message by the device and before opening the COMMAND message listening window	1	R/W	L/R	In seconds, from 1 (1s) to 255 (255s)	Yes, R/W
19	**EXCH_RX_LENGTH**	Duration of the COMMAND message listening window by the device	1	R/W	L/R	In multiples of 5 milliseconds, from 0 (reception disabled) to 255 (1.27s)	Yes, R/W
1A	**EXCH_RESPONSE_DELAY**	Time between reception of a COMMAND message by the device and transmission of the corresponding RESPONSE message	1	R/W	L/R	In seconds, from 0 (0s) to 255	Yes, R/W
1B	**EXCH_RESPONSE_DELAY_MIN**	Minimum value accepted for the EXCH_RESPONSE_DELAY parameter (defined by the device MANUFACTURER)	1	R	L/R	In seconds, from 0 (0s) to 255 (255s)	Yes, R/W
1C	**L7TRANSMIT_LENGTH_MAX**	Maximum length of application messages that can be sent by the device (fixed value defined by SUEZ)	1	R	L/R	In bytes, from 40 to 100	Yes, read only
1D	**L7RECEIVE_LENGTH_MAX**	Maximum length of application messages that can be received by the device (Fixed value defined by SUEZ)	1	R	L/R	En bytes, de 50 à 100	Yes, read only
20	**CLOCK_CURRENT_EPOC**	Current time of device	4	R/W	L/R	EPOCH encoded on 32 bits and corresponding to the number of seconds since 1st January 2013 at 00:00: MSB first (big endian) Programmed in factory	No
21	**CLOCK_OFFSET_CORRECTION**	Relative correction (time delta) to be applied to the device clock once only to correct its absolute drift	2	W	L/R	All values from -32768s to +32767s. Signed number on 16 bits encoded in complement on 2: Byte 1: MSBs Byte 2: LSBs This parameter is a virtual parameter: each writing corrects the current time in relative manner	No
22	**CLOCK_DRIFT_CORRECTION**	Correction of device clock frequency	2	R/W	L/R	Number S of seconds to add to or subtract from the current time every D days: Byte 1: Number of seconds S (signed integer from -128 to +127, complement on two) Byte 2: Number of days D (from 1 to 255)	No

28	**CIPH_CURRENT_KEY**	Current key number	1	R/W	L/R	00: No encryption (not accepted through APP-ADMIN request) 01 to CIPH_KEY_COUNT: Kenc key number enabled Other values: reserved	Yes, read only (01)
29	**CIPH_KEY_COUNT**	Number of encryption keys available in the device	1	R	L/R	unsigned int	Yes, read only (01)
2A	**L6NetwIdSelected**	Value used by the device to initialize the L6NetwId field of any upstream messages	1	R	L/R	00 to 255	Yes, R/W
30	**PING_RX_DELAY**	Fixed waiting time after transmission of an INSTPING message by the device and before opening the INSTPONG message listening window	1	R/W	L/R	In seconds, from 1 (1s) to PING_RX_DELAY_MIN	Yes, R/W
31	**PING_RX_LENGTH**	Duration of the INSTPONG message listening window by the device	1	R/W	L/R	In seconds, from 1 (1s) to PING_RX_LENGTH_MAX	Yes, R/W
32	**PING_RX_DELAY_MIN**	Minimum value of the PING_RX_DELAY parameter	1	R	L/R	In seconds, from 1 (1s) to 255 (255s)	Yes, read only
33	**PING_RX_LENGTH_MAX**	Maximum value of the PING_RX_LENGTH parameter	1	R	L/R	In seconds, from 1 (1s) to 255 (255s)	Yes, read only
34	**PING_LAST_EPOCH**	Execution time of the last connectivity test (INSTPING/INSTPONG)	4	R	L/R	EPOCH	No
35	**PING_NBFOUND**	Number of different INSTPONG messages received in response to the last connectivity test	1	R	L/R	From 00 to FF	Yes, read only
36	**PING_REPLY1**	Response 1 received for the last connectivity test (Bigest L7RssiDown)	9	R	L/R	Concatenation of the following fields: L7ConcentId (6 bytes) L7ModemId (1 byte) L7RssiUpstream (1 byte) L7RssiDownstream (1 byte)	Yes, read only
37	**PING_REPLY2**	…	9	R	L/R	Idem Ping_reply1	Yes, read only
38	**PING_REPLY3**	…	9	R	L/R	Idem Ping_reply1	Yes, read only
39	**PING_REPLY4**	…	9	R	L/R	Idem Ping_reply1	Yes, read only
3A	**PING_REPLY5**	…	9	R	L/R	Idem Ping_reply1	Yes, read only
3B	**PING_REPLY6**	…	9	R	L/R	Idem Ping_reply1	Yes, read only
3C	**PING_REPLY7**	…	9	R	L/R	Idem Ping_reply1	Yes, read only
3D	**PING_REPLY8**	Response 8 received for the last connectivity test, (Weakest L7RssiDown)	9	R	L/R	Idem Ping_reply1	Yes, read only
3E	**EXECPING_PERIODE**	Periodic time of execping sending by the device, in months	1	R/W	L/R	Unsigned int, 0 = deactivated	Yes, read only (0)