

UHF Reader

IQBoxx

Communication Protocol

Summary

1	Introduction	4
1.1	Configuration Parameters	4
1.1.1	Section 0x00 – General parameters.....	4
1.1.2	Section 0x01 – Working Area Configuration	5
1.1.3	Section 0x02 – Antenna Configuration	5
1.1.4	Section 0x03 – Channel Frequency Configuration	5
1.1.5	Section 0x04 – Configuration Reader Parameters.....	6
1.1.6	Section 0x05 – Tag Protocol Parameter Configuration.....	6
1.2	Miscellaneous notes and concepts	8
1.2.1	RSSI Power	8
1.2.2	ISO 18000-63	8
1.2.3	TAG memory.....	10
1.2.4	Reflected power	12
1.3	Command specifications	14
1.3.1	0x30 – Device Reset.....	14
1.3.2	0x34 – Read Firmware Version	15
1.3.3	0x33 – Firmware update	16
1.3.4	0x3D – Write ROM – Configuration Parameters	19
1.3.5	0x3E – Read ROM – Configuration Parameters.....	20
1.3.6	0x31 – ROM Reset – Configuration Parameters to Default	21
1.3.7	0x39 – Activation Status Set.....	22
1.3.8	0xFE – Reading Reflected Power	23
1.3.9	0x18 – Inventory request	25
1.3.10	0x1E – EPC Writing.....	27
1.3.11	0x19 – Reading data from TAG.....	29
1.3.12	0x1A – Writing data to the TAG.....	31
1.3.13	0x1C – Kill the TAG.....	34
1.3.14	0xE9 – Get Product Code.....	36
1.3.15	0x06 – Read Database Data.....	37
1.3.16	0x07 – Number of records in the Database	39
1.3.17	0x08 – Reset Database Data	40
2	Command transmission in TCP mode	41

2.1	ASCII-Encoded Format.....	41
2.2	CRC Calculation.....	41
2.3	Examples	42

1 Introduction

1.1 Configuration Parameters

The device can be configured by means of a series of parameters that define its behavior and characteristics.

At the moment we do not distinguish into volatile and permanent parameters but consider all parameters as permanent in ROM of the main board.

We describe below the division of the parameters into *sections*, indicating for each the ordered list of bytes that represent the section itself and the positioning of the parameter values within the list; at the time of writing (reading) the parameters on (from) the device each section will be written (read) atomically sending (or receiving) the bytes of the entire section.

Parameters can have a value of more than one byte, just as the same byte can represent several parameters (e.g. LSB one parameter, MSB another parameter). So, the number of bytes may not usually coincide with the number of parameters.

1.1.1 Section 0x00 – General parameters

General device configurations:

Byte Index	Value/Meaning
0x00	Device address: 0x00 to 0xFF
0x01	Enable "Continuous mode" (0x01 enabled, 0x00 not enabled) - Not used
0x02	Enabling "Spontaneous mode" (0x01 enabled, 0x00 not enabled)
0x04	Duration of the inventory-by-request, in 0.1s units
0x04 ... 0x0F	0x00 (unused)
0x10 ... 0x13	IP address
0x14 ... 0x17	Subnet mask
0x18 ... 0x19	Port Es: 0x0BB8: 3000
0x1A ... 0x2F	0x00 (unused)
0x30 ... 0x33	Baud rate: 0x000004B0: 1200 0x00000960: 2400 0x000012C0: 4800 0x00002580: 9600 0x00004B00: 19200 0x00009600: 38400 0x0000E100: 57600 0x0001C200: 115200
0x34	Data bits Values: 0x07, 0x08
0x35	Stop bits Values: 0x01, 0x02

0x36	Parity: 0x00: None 0x01: Odd 0x02: Even
0x37 . . . 0x63	0x00 (unused)

Configuration bytes length: 100 bytes

1.1.2 Section 0x01 – Working Area Configuration

For now, the Europe area (ETSI) is considered fixed. NOT CONFIGURABLE.

1.1.3 Section 0x02 – Antenna Configuration

Let's assume N managed antennas; For each we set:

- Number: 1 byte, index from 0x00 to 0x20
- Tx power: 2 bytes - (0x00, 0x00) to (0x0C, 0xE4) (up to 33.00 dBm)
- Rx power: 2byte - (0x00, 0x00) to (0x0C, 0xE4) (up to 33.00 dBm)
- If Active: 1 byte {0x00 = inactive, 0x01 = active}

Each antenna therefore contains 6 bytes.
Total Byte Configuration: 6 * NR Antennas

1.1.4 Section 0x03 – Channel Frequency Configuration

We specify the N rotating frequencies, in the ranges provided for the selected area (ETSI time), in KHz (integers) at 4 bytes, from 865,700 KHz to 867,500 KHz (0x000D35A4 ... 0x000D3CAC)

Maximum number of frequencies that can be specified: 50
Unspecified frequencies will be bytes 0x00000000

Two dwell values can also be specified:

- Dwell per antenna (2 bytes): 0x0000 ...0x0190
- Dwell for frequency (2 bytes): 0x0000 ...0x0190

Total bytes: 50*4+2+2

NOTE – For ETSI working area, only the following frequencies are supported:

- 865,700 KHz
 - 866,300 KHz
 - 866,900 KHz
 - 867,500 KHz
-

1.1.5 Section 0x04 – Configuration Reader Parameters

Set (some) values (flags for now) related to the reader.

We plan to send 16 bytes, for 16 possible values.

Byte Index	Value/Meaning
0x00	Antenna port number as the identifier of the item in the buffer: 0x00: YES 0x01: NO
0x01 ... 0x05	0x00 (unused)
0x06	Indicates how the RSSI value seen during the read is used 0x00: the max is not recorded, but the last bed is used 0x01: The maximum value seen is recorded
0x07	0x00 (unused)
0x08	Indicates whether tag bank data is used as an identifier 0x00: YES 0x01: NO
0x09 ... 0x0F	0x00 (unused)

Configuration bytes length: 16 bytes

1.1.6 Section 0x05 – Tag Protocol Parameter Configuration

Set values for how tags are read.

Byte Index	Value/Meaning																																																																																																						
0x00	Session: 0x00: Session0 0x01: Session1 0x02: Session2 0x03: Session3																																																																																																						
0x01	Target: 0x00: A 0x01: B																																																																																																						
0x02	RF mode Allowable values: 0x01, 0x03, 0x05, 0x07, 0x11, 0x12, 0x13, 0x15 <div>Table 10: Impinj Reader Chip RF Mode Parameters and Performance</div> <table><tr><th rowspan="2">Mode ID</th><th rowspan="2">Mode Optimization</th><th rowspan="2">Forward Link Modulation</th><th rowspan="2">Tari (µs)</th><th rowspan="2">PIE</th><th rowspan="2">BLF (kHz)</th><th rowspan="2">Reverse Link Modulation</th><th colspan="3">Chip Receive Sensitivity Minimum* (dBm)</th><th rowspan="2">Maximum Read Rate** (tags/s)</th></tr><tr><th>E710</th><th>E510</th><th>E310</th></tr><tr><td>11</td><td>Read Rate</td><td>PR-ASK</td><td>7.5</td><td>2</td><td>640</td><td>FM0</td><td>-78</td><td>N/A</td><td>N/A</td><td>700+</td></tr><tr><td>1</td><td>Read Rate</td><td>PR-ASK</td><td>7.5</td><td>2</td><td>640</td><td>Miller M=2</td><td>-81</td><td>-75</td><td>N/A</td><td>550+</td></tr><tr><td>15</td><td>Read Rate</td><td>PR-ASK</td><td>7.5</td><td>2</td><td>640</td><td>Miller M=4</td><td>-84</td><td>-78</td><td>N/A</td><td>400+</td></tr><tr><td>12</td><td>ETSI</td><td>PR-ASK</td><td>15</td><td>2</td><td>320</td><td>Miller M=2</td><td>-84</td><td>-78</td><td>-71</td><td>300+</td></tr><tr><td>3</td><td>ETSI</td><td>PR-ASK</td><td>20</td><td>2</td><td>320</td><td>Miller M=2</td><td>-84</td><td>-78</td><td>-71</td><td>250+</td></tr><tr><td>5</td><td>ETSI DRM</td><td>PR-ASK</td><td>20</td><td>2</td><td>320</td><td>Miller M=4</td><td>-87</td><td>-81</td><td>-74</td><td>200+</td></tr><tr><td>7</td><td>FCC DRM</td><td>PR-ASK</td><td>20</td><td>2</td><td>250</td><td>Miller M=4</td><td>-88</td><td>-82</td><td>-75</td><td>150+</td></tr><tr><td>13</td><td>Sensitivity</td><td>PR-ASK</td><td>20</td><td>2</td><td>160</td><td>Miller M=8</td><td>-93</td><td>-87</td><td>-80</td><td>50+</td></tr></table>	Mode ID	Mode Optimization	Forward Link Modulation	Tari (µs)	PIE	BLF (kHz)	Reverse Link Modulation	Chip Receive Sensitivity Minimum* (dBm)			Maximum Read Rate** (tags/s)	E710	E510	E310	11	Read Rate	PR-ASK	7.5	2	640	FM0	-78	N/A	N/A	700+	1	Read Rate	PR-ASK	7.5	2	640	Miller M=2	-81	-75	N/A	550+	15	Read Rate	PR-ASK	7.5	2	640	Miller M=4	-84	-78	N/A	400+	12	ETSI	PR-ASK	15	2	320	Miller M=2	-84	-78	-71	300+	3	ETSI	PR-ASK	20	2	320	Miller M=2	-84	-78	-71	250+	5	ETSI DRM	PR-ASK	20	2	320	Miller M=4	-87	-81	-74	200+	7	FCC DRM	PR-ASK	20	2	250	Miller M=4	-88	-82	-75	150+	13	Sensitivity	PR-ASK	20	2	160	Miller M=8	-93	-87	-80	50+
Mode ID	Mode Optimization								Forward Link Modulation	Tari (µs)	PIE		BLF (kHz)	Reverse Link Modulation	Chip Receive Sensitivity Minimum* (dBm)			Maximum Read Rate** (tags/s)																																																																																					
		E710	E510	E310																																																																																																			
11	Read Rate	PR-ASK	7.5	2	640	FM0	-78	N/A	N/A	700+																																																																																													
1	Read Rate	PR-ASK	7.5	2	640	Miller M=2	-81	-75	N/A	550+																																																																																													
15	Read Rate	PR-ASK	7.5	2	640	Miller M=4	-84	-78	N/A	400+																																																																																													
12	ETSI	PR-ASK	15	2	320	Miller M=2	-84	-78	-71	300+																																																																																													
3	ETSI	PR-ASK	20	2	320	Miller M=2	-84	-78	-71	250+																																																																																													
5	ETSI DRM	PR-ASK	20	2	320	Miller M=4	-87	-81	-74	200+																																																																																													
7	FCC DRM	PR-ASK	20	2	250	Miller M=4	-88	-82	-75	150+																																																																																													
13	Sensitivity	PR-ASK	20	2	160	Miller M=8	-93	-87	-80	50+																																																																																													

0x03 ... 0x0B	0x00 (unused)
0x0C	Q value and algorithm: Static: 0x01 to 0x0F Dynamic: 0xFF
0x0D ... 0x1F	0x00 (unused)

Configuration bytes length: 32 bytes

1.2 Miscellaneous notes and concepts

Below is a series of theoretical information retrieved during the drafting of the document.
[subject to validation]

1.2.1 RSSI Power

RSSI stands for *Received Signal Strength Indicator*. It is a measure of the strength with which an RFID signal is received by the tag by the reader.

It is useful for:

- Understand how close or far a tag is.
- Evaluate the quality of the reception.
- Diagnose issues such as reflection, interference, misplaced or damaged tags.

The possibility of reading the RSSI value is relative to the presence of a TAG, as it indicates the goodness of the signal received in the search for an external RF source; it is therefore a value that is, if provided for by configuration, returned with read commands.

We agree that the return value, where applicable, is in dBm in signed format with two's complement; the returned value (byte) is subtracted from 256 to obtain the actual value.

- ES. 0xDB = -37 dBm
- ES. 0xEC = -20 dBm

1.2.2 ISO 18000-63

ISO/IEC 18000-63 is an international standard for passive UHF RFID communication.

It is an ISO standard that:

- It defines the physical layer (PHY) and communication protocol for passive UHF RFID systems.
- It is based on the EPCglobal Class-1 Generation-2 (Gen2) protocol.
- It operates in the UHF band (860 MHz – 960 MHz).
- It replaces the previous ISO 18000-6C version.

It is a globally harmonized technical standard for long-distance radio frequency object identification.

Aspect	Notes
Full name	ISO/IEC 18000-63:2013
Based on	EPCglobal Gen2 v2.0
Tag type	Passive (without battery)
Operating frequencies	860–960 MHz
Modulation Type	ASK, PSK (depending on configuration)
Reading distance	Typically from 1 to 10 meters (depends on power and antenna)
Transmission Rate	Up to ~640 kbps
Communication	Half-duplex reader-talks-first

Compatibility	It is the standard used by most modern UHF RFID systems
---------------	---

Notes:

- ISO 18000-63 = ISO version of the EPC Gen2 standard.
- Tags and readers that are compatible with one are compatible with the other.

The protocol defines a series of commands (mandatory and non-mandatory) that govern the operation of the protocol, e.g. to start an inventory *round* or to read or write one or more blocks on the tag memory, e.g. to read data from a tag memory bank (EPC, TID, USER, etc.).

The commands described in this document abstract from the ISO 18000-63 protocol, proposing commands (also in relation to the EX10 module used by the device) that encapsulate the "underlying" commands of the ISO 18000-63 protocol.

For example, the *Inventory* command described in this document (see command details) "replaces" the following underlying command flow of the ISO 18000-63 protocol:

1. **Select** → filter the tags to query.
2. **Query** → starts the inventory cycle.
3. **QueryRep** → repeats the loop.
4. **ACK** → select tags.
5. (optional) **Read** → EPC, TID, etc reads

The EX10 module has a level of abstraction with respect to ISO 18000-63, so it exposes, for example, the `0x22` command which, in turn, is invoked (with appropriate parameters) when the *Inventory command* described here is executed.

1.2.3 TAG memory

In UHF RFID tags according to the ISO/IEC 18000-63 / EPC Gen2 standard, there are four standardized memory banks, each with a specific purpose.

Bank	Name	Code	Contents	Access
0	Reserved	0x00	Access and Kill passwords	Protected
1	EPC	0x01	PC Word + EPC + CRC16	Writable
2	TID	0x02	Fixed identifier, factory programmed	Read
3	User	0x03	Customizable user data space	Writable

Bank 0 – Reserved

- Contains:
 - *Kill Password* (word 0–1)
 - *Access Password* (word 2–3)
- It is used for:
 - Secure access (write/read)
 - Disable the tag (*Kill*)
- Protection: Requires password to read/write

Bank 1 – EPC

- Contains:
 - 0x00: PC Word
 - From 0x01 onwards: EPC
 - End: CRC-16 automatically calculated
- Writable (except CRC)

PC Word is a 16-bit word (2 bytes, *little endian*) that is written and maintained by the reader or system that programs the EPC. It serves to communicate to the reader:

- the length of the EPC (in bits),
- if the tag uses ISO or EPC *numbering scheme*,
- and if there are active flags (e.g. custom, user-defined, etc.).

Bit	Meaning
[7:0]	Length of the EPC in 16-bit word
[8]	NSI: 0 = EPCglobal, 1 = ISO
[9]	X: Extended Protocol Control (custom use)
[10]	U: 1 if user memory is present
[15:11]	Reserved

Bank 2 – TID

- Programmed by the manufacturer
-

-
- Contains unique and immutable chip identifier
 - Some tags also provide an extended serial (Extended TID)
 - Read Only
 - Used for anti-cloning or certification

Bank 3 – USER

- Completely free and customizable space
 - Some tags don't have it, or have it small in size
 - It can be used for:
 - Technical data
 - Logistics info
 - Timestamp
 - Operator codes
-

1.2.4 Reflected power

The term "reflected power" refers to the portion of radio signal (RF) signal power that is not absorbed or transmitted by the antenna and instead is reflected back toward the transmitter. This phenomenon happens when the impedance of the antenna is not properly matched to the impedance of the transmitter or coaxial cable.

Whenever an RF signal is transmitted along a line (e.g. a coaxial cable) to a load (the antenna), if the load does not absorb all the power, a part of it comes back.

This can be due to:

- Antenna disconnected or damaged.
- Antenna impedance different from cable impedance (typically 50 ohms).
- Physical problems in the cable or connector.

Associated parameters:

- VSWR (Voltage Standing Wave Ratio): Measures how well the antenna is matched. The closer it is to 1:1, the better. A high VSWR indicates a lot of reflected power.
- Return Loss: This is a logarithmic measure of reflected power, the higher the better (the less reflected power).

Reflected power can damage the transmitter if not handled properly.

It also reduces the efficiency of the system: less power is transmitted over the air, so less range or read capacity in the case of an RFID reader.

Measuring the reflected power allows you to understand if the antenna is properly connected and working.

Examples and industry references:

Return Loss (dB)	Reflected Power (%)	Comment
40 dB	0.01%	Negligible reflex (excellent)
30 dB	0.1%	Very good
20 dB	1%	Good
10 dB	10%	Acceptable but needs improvement
6 dB	25%	Critical, lots of reflected energy
< 3 dB	>50%	Very inefficient

Calculations and Formulas

Since it is a ratio between powers (Reflected Power over Incident Power), the Return Loss in dB is given by:

$$RL = 10 \log \left(\frac{P_{Inc}}{P_{Rif}} \right) \text{ oppure } RL = -10 \log \left(\frac{P_{Rif}}{P_{Inc}} \right)$$

The *Voltage Standing Wave Ratio*:

$$VSWR = \frac{1 + 10^{\frac{RL}{10}}}{1 - 10^{\frac{RL}{10}}} \quad (\text{formula RP1})$$

The *Reflection Coefficient*:

$$\Gamma = \frac{VSWR - 1}{VSWR + 1}$$

The *Return Loss* can be calculated from the *Reflection Coefficient*:

$$RL = -10 \log (\Gamma^2)$$

The *Reflected Power* is calculated from the incident power using:

$$P_{Ref} = P_{Inc} * \Gamma^2$$

Average Return Loss Calculation

If we have N frequencies, each with its own RL_i con $i = 1 \dots N$, we cannot average the values in dB directly; we must first convert each to its linear form, compute the average (thus obtaining the linear mean value):

$$RL_{AVG}^* = \frac{1}{N} \sum_{i=1}^N 10^{\frac{RL_i}{10}} \quad (\text{Average of the linear values})$$

Then:

$$RL_{AVG} = -10 \log (RL_{AVG}^*)$$

This represents the Return Loss associated with the average reflection from an energy perspective—that is, the average of the reflections in terms of power. It differs from the average of the RL values (in dB, due to the logarithmic scale) and from the RL that could be derived from the average reflection coefficient Γ_{AVG} , which is the arithmetic mean of the individual Γ_i (representing a kind of *Return Loss* associated with the average reflection in amplitude).

Average VSWR Calculation

The average $VSWR_{AVG}$ can be calculated from the average Return Loss RL_{AVG} using formula RP1.

Alternatively, from the average reflection coefficient Γ_{AVG} using the following:

$$VSWR = \frac{1 + \Gamma}{1 - \Gamma} \quad (\text{formula RP2})$$

1.3 Command specifications

1.3.1 0x30 – Device Reset

The command is used to send the reset to the device, which, once it responds to the command, will perform a reboot.

Command:

#	Value	
1	0x01	Length (LSB)
2	0x00	Length (MSB)
3	0x30	Command code of the original request

Fail:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0x30	Command code of the original request
4	0x15	NAK

Successful:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0x30	Command code of the original request
4	0x00	OK

1.3.2 0x34 – Read Firmware Version

Requires the current Firmware version running on the device.

The maximum string length representative of the FW version is 16 characters.

Command:

#	Value	
1	0x01	Length (LSB)
2	0x00	Length (MSB)
3	0x34	Command code of the original request

Fail:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0x34	Command code of the original request
4	0x15	NAK

Successful:

#	Value	
1	0x12	
2	0x00	
3	0x34	Command code of the original request
4	0x00	OK
5..20	...	Bytes with data (16 ASCII characters)

1.3.3 0x33 – Firmware update

Sends the firmware update binary to the device.

Once the writing on flash has been carried out, a validation occurs. In case of errors, in addition to the usual NAK, we also expect specific errors such as "unable to write to flash" or "Invalid image".

The binary must be split into numbered packets (*chunks*) of up to 2500 bytes each, to be sent sequentially. A special final chunk must be included to indicate the end of the transmission.

A send command is used for each chunk.

Note, at present the FW has dimensions > 1Mb.

Details

Once the binary of the new version has been produced, the bytes that compose it must be sent to the device at *chunks*. The entire byte array must then be divided into N *chunks* of which the first N-1 will have a size of 2500 bytes, the N-th will have a size of <= 2500 bytes.

Sending is done by calling this command in sequence, *chunk* after *chunk*. Since the total length is not declared in advance, a special *chunk* is sent to indicate the end of the operations (i.e. the device is given the OK to proceed).

E.g. Binary size: 1,050,200 bytes => N = 421:

- N-1 = 420 2500-byte chunks (for a total of 1,050,000 bytes)
- N-th *chunk*: 200bytes
- (N+1)-th *chunk*: ok to proceed

NOTE – The (N+1)-th chunk is mandatory, since the N-th may also be 2500 bytes, and therefore indistinguishable from a non-final chunk.

The final chunk (N+1) will have no data and will have index 0xFFFF.

NOTE: the maximum size of the binary is 19,398,064 bytes:

- Maximum number of *data cunks*: 7,760
- Last *chunk* full data

Chunk structure

Each *chunk* can contain up to 2502 bytes:

- 2 bytes for *chunk* index, base 0: MSB, LSB
- 2500 bytes of FW data

Data chunk (e.g. index 120)

Chunk index	FW bytes from 0 to 2499
-------------	-------------------------

0x00	0x78	bytes 0x00	...	(2500-th)
------	------	------------	-----	-----------

Last partial data *chunk* (e.g. index 180)

Chunk index		FW bytes from 0 to 31		
0x00	0xB4	bytes 0x00	...	(32-th)

Final Chunk (No Data)

Chunk index	
0xFF	0xFF

To be considered that each *chunk* trasmission requires an execution of the current command, and that to each chunk *the initial protocol bytes are added*:

- 2 bytes for length,
- 1 byte for the command code
- Up to 2502 bytes for chunk:
 - 2 bytes for index chunk number
 - Up to 2500 bytes for FW bytes

NOTE: the *chunk* index is represented in *big endian* and is zero- based.

Command:

#	Value	
1	0x83	Length (LSB) – i.e. 131
2	0x00	Length (MSB)
3	0x33	Command code
4	0x00	Chunk index (MSB)
5	0x78	Chunk index (LSB) – i.e. 120
6...n	...	Chunk data – Not present for final chunk

Fail:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0x33	Command code of the original request
4	0x15	NAK

Specific failure for "bad image", where applicable:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0x33	Command code of the original request
4	0x02	NAK

NOTE - To be defined how to determine it, e.g. checksum.

Specific failure for "flash write failed":

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0x33	Command code of the original request
4	0x03	NAK

Successful:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0x33	Command code of the original request
4	0x00	OK

1.3.4 0x3D – Write ROM – Configuration Parameters

We use a single command to send the device configuration values. With reference to the descriptive understanding of the parameters and their division into sections, we plan to send to the device the parameters relating to one section (one at a time), including the unused values (for simplicity).

The writing involves prefixing the bytes of the section with an additional byte indicating the section number. So if the section is composed of N bytes, the following will be sent, according to the usual standard:

- The two bytes of data length
- The command byte (0x3D)
- Configuration data
 - The section identifier byte
 - The N bytes representing the values of the section parameters

Command:

#	Value	
1	0x12	Length (LSB) – i.e. 131
2	0x00	Length (MSB)
3	0x3D	Command code
4	0x04	Section number – i.e. #4
5... N + 5	...	Section data – i.e. N = 16 for section #4

Fail:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0x3D	Command code of the original request
4	0x15	NAK

Successful:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0x3D	Command code of the original request
4	0x00	OK

1.3.5 0x3E – Read ROM – Configuration Parameters

Reads the data of a specific section of the ROM. For the meaning of the data (reference to the parameter represented and its value), see the map of the parameters of the appropriate section.

Note – The number of bytes relative to the section and therefore the value of X depends on the structure of the section itself; see map of the parameters relating to the section.

The section of interest is passed with a byte; e.g.: 0x04.

The response does not indicate the section, but only the data (may change in future releases).

Command:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0x3E	Command code
4	0x04	Section number – i.e. #4

Fail:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0x3E	Command code of the original request
4	0x15	NAK

Successful:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0x3E	Command code of the original request
4	0x00	OK
5... N+4	...	Memory Section Data (N Bytes)

1.3.6 0x31 – ROM Reset – Configuration Parameters to Default

Resets the values of the specific configuration section to the default values.

The section of interest is passed with a byte; e.g.: 0x04.

Command:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0x31	Command code
4	0x04	Section number – i.e. #4

Fail:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0x31	Command code of the original request
4	0x15	NAK

Successful:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0x31	Command code of the original request
4	0x00	OK

1.3.7 0x39 – Activation Status Set

Sets the RF status, active/inactive.

1 byte is passed indicating the status:

- 0x01: Active RF
- 0x00: RF Not Active

Command:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0x39	Command code
4	0x04	Status parameter

Fail:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0x39	Command code of the original request
4	0x15	NAK

Successful:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0x39	Command code of the original request
4	0x00	OK

1.3.8 0xFE – Reading Reflected Power

This command allows the device to report the ~~Return Loss~~ VSWR of a specific antenna, ~~expressed in decibels (dB)~~, which can be used to calculate the corresponding *reflected power* percentage.

The antenna is specified as a parameter of the command.
The current antenna TX power value is used as test power.

The response contains the VSWR (2 bytes) for each frequency (4) bytes and the “average” VSWR over all frequencies (VSWR*).

The VSWR values are expressed in thousandths.
For example, a VSWR value of 1.321 becomes 1321 thousandths, and consequently, the returned value will be 0x0529.

The maximum returnable value is 0xFFFF;
values greater than this will always be represented as 0xFFFF, since there is no practical benefit in further detailing such poor impedance matching.

Command:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0xFE	Command code
4	0x01	Antenna of interest – i.e. #1

Fail:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0xFE	Command code of the original request
4	0x15	NAK

Successful – Let N be the Number of frequencies

#	Value	
1	...	Length (LSB)
2	...	Length (MSB)
3	0xFE	Command code of the original request
4	0x00	OK
5, 6	...	Average VSWR
7,8,9,10		Frequency #1 – Value in KHz as hex integer (4 bytes)
11, 12		Frequency #1 – VSWR ₁

...		
6+6*(N-1)+1, ... 6+6*(N-1)+4		Frequency #N – Value in KHz as hex integer (4 bytes)
6+6*(N-1)+5, ... 6+6*(N-1)+6		Frequency #N – VSWR _N

Please note: as discussed before, VSWR* is not the simple average value of all VSWR_i.

1.3.9 0x18 – Inventory request

Requests the device to send the inventory of ISO 18000-63 TAGs detected within the antennas' range, optionally including the RSSI value.

The command supports the following parameters:

- Flag for returning the antenna, e.g. 0x01= YES.
- Flag for returning the RSSI, e.g. 0x01= YES.

If any TAGs are found, the response length may vary depending on several factors:

- Based on the number of TAGs found, N
- Based on the information returned for each TAG: C bytes for the identification code
- Depending on whether the antenna number is included: 1 extra byte for the antenna index, referred to as A; A may be 1 byte (antenna included) or 0 bytes (antenna not included)
- Depending on whether the RSSI is included: 1 extra byte for the RSSI value, referred to as R; R may be 1 byte (RSSI included) or 0 bytes (RSSI not included)

Command:

#	Value	
1	0x03	Length (LSB)
2	0x00	Length (MSB)
3	0x18	Command code
4	0x01	Flag for returning the antenna – i.e. true
5	0x01	Flag for returning the RSSI – i.e. true

Fail:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0x18	Command code of the original request
4	0x15	NAK

Success with no Tag found:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0x18	Command code of the original request
4	0x00	OK

Let's define:

- T the number of tags found, $T > 0$
 - n_j : the length in word of the j-th tag, for $j \in \{1 \dots T\}$
 - $N_j = n_j * 2$ the length in bytes b, of the j-th tag, for $j \in \{1 \dots T\}$
-

Successful Response format, for $T > 0$, has the following response:

#	Value	
1	...	Length (LSB)
2	...	Length (MSB)
3	0x18	Command code of the original request
4	0x00	OK
4 + 1	(n1)	TAG #1 - Length of EPC Id in word ($n1 \Rightarrow N1$)
4 + 2		TAG #1 - EPC Id: first byte of the first word
4 + 3		TAG #1 - EPC Id: second byte of the first word
...		
4 + 1 + N1 - 1		TAG #1 - EPC Id: first byte of the word $n1$
4 + 1 + N1		TAG #1 - EPC Id: second byte of word $n1$
4 + 1 + N1 + A1		TAG #1 - Antenna No. (if required) associated with the TAG
4+1+N1+A1+R1		TAG #1 - RSSI value (if required) associated with the TAG
4 + 1 + N ₁ + A1 + R1 + 1	(n2)	TAG #2 - Length of EPC Id in word ($n2 \Rightarrow n2$)
4 + 1 + N ₁ + A1 + R1 + 2		TAG #2 - EPC Id: first byte of the first word
4 + 1 + N ₁ + A1 + R1 + 3		TAG #2 - EPC Id: second byte of the first word
...		
4+2+N1+A1+R1+N2 - 1		TAG #2 - EPC Id: first byte of the word $n2$
4+2+N1+A1+R1+N2		TAG #2 - EPC Id: second byte of the word $n2$
4+2+N1+A1+R1+N2+A2		TAG #2 - Antenna No. (if required) associated with the TAG
4+2+N1+A1+R1+N2+A2+R2		TAG #2 - RSSI value (if required) associated with the TAG
...		(Other TAGS)
4+T-1+1 + $\sum_{j=1}^{T-1} (N_j + A_j + R_j)$	(nT)	TAG #T - Length of EPC Id in word ($nT = > NT$)
4+T-1+2 + $\sum_{j=1}^{T-1} (N_j + A_j + R_j)$		TAG #T - EPC Id: first byte of the first word
4+T-1+3 + $\sum_{j=1}^{T-1} (N_j + A_j + R_j)$		TAG #T - EPC Id: second byte of the first word
...		
4+T+ $\sum_{j=1}^T N_j$ + $\sum_{j=1}^{T-1} (A_j + R_j - 1)$		TAG #T - EPC Id: first byte of the nT word
4+T+ $\sum_{j=1}^T N_j$ + $\sum_{j=1}^{T-1} (A_j + R_j)$		TAG #T - EPC Id: second byte of the nT word
4+T+ $\sum_{j=1}^T (N_j + R_j)$ + $\sum_{j=1}^{T-1} A_j$		TAG #T - Antenna No. (if required) associated with the TAG
4+T+ $\sum_{j=1}^T (N_j + A_j + R_j)$		TAG #T - RSSI value (if required) associated with the TAG

1.3.10 0x1E – EPC Writing

It is used for programming the EPC of an identified ISO 18000-63 TAG.

The command includes the EPC code identifying the TAG (n words), the password (4 bytes), and the data to be programmed - the new 'net' EPC (i.e., without the PC word and CRC) - consisting of m words.

NOTE - For historical and technical reasons, *word* (2 bytes) is used:

- n word \rightarrow N bytes, with $N = 2 \cdot n$
- m word \rightarrow M bytes, with $M = 2 \cdot m$

When the password not set: 4 bytes per 0x00.

PC word (zero-address word) and CR word do not need to be passed in the command as they are automatically calculated

So be careful:

- EPC identifier \rightarrow contains PC word and CRC
- new EPC Id to be written \rightarrow does not contain PC word and CRC

Data to be sent (to which the usual two bytes of length should be added at the beginning):

#	Value	
1	...	Length (LSB)
2	...	Length (MSB)
3	0x1E	Command
4		Length of EPC Id in word (n), e.g. 8 for Id from 6 word
4 + 1		EPC Id: First byte of the first word
4 + 2		EPC Id: Second byte of the first word
...		
4 + N - 1		EPC Id: First byte (N-1) of the last word (n)
4 + N		EPC Id: Second byte (N) of the last word (n)
4 + N + 1		Password: First Byte
4 + N + 2		Password: Second Byte
4 + N + 3		Password: Third Byte
4 + N + 4		Password: Fourth Byte
4 + N + 4 + 1		New EPC: First Byte of First Word
4 + N + 4 + 2		New EPC: Second byte of the first word
...		
4 + N + 4 + M - 1		New EPC: First byte (M-1) of the last word (m)
4 + N + 4 + M		New EPC: Second byte (M) of the last word (m)

NOTE - The length byte (byte nr. 2) is used to indicate the length of the EPC Id "field". It is not necessary to indicate the length of the new EPC instead, as it is implicit.

Fail:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0x1E	Command code of the original request
4	0x15	NAK

Fails, tag not present:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0x1E	Command code of the original request
4	0x01	No tag

Note – The TAG not present here is considered as an error.

Fail, write failed:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0x1E	Command code of the original request
4	0x02	Write failed or tag does not support writing to the specified block

Successful:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0x1E	Command code of the original request
4	0x00	OK

1.3.11 0x19 – Reading data from TAG

It is used to read the data part of an identified ISO 18000-63 TAG.

The command includes the EPC code identifying the TAG (n words, including the PC word and CRC), the password (4 bytes), the memory bank to read from (1 byte), the initial memory address (relative to the specified bank) from which to start reading (4 bytes), and the number w of blocks (words) to be read (1 byte, maximum 64 blocks) from the specified address.

NOTE - For historical and technical reasons, *word* (2 bytes) is used:

- n word \rightarrow N bytes, with $N = 2 * n$
- w blocks \rightarrow W bytes, with $W = 2 * w$

When the password not set: 4 bytes per 0x00.

Values used to identify memory banks:

Bank Index	Description
0x00	<i>Reserved</i>
0x01	<i>EPC</i>
0x02	<i>TID</i>
0x03	<i>User</i>

Data to be sent (to which the usual two bytes of length are added at the beginning):

#	Value	
1	...	Length (LSB)
2	...	Length (MSB)
3	0x19	Command
4		Length of EPC Id in word (n), e.g. 8 for Id from 6 word
4 + 1		EPC Id: First byte of the first word
4 + 2		EPC Id: Second byte of the first word
...		
4 + N - 1		EPC Id: First byte (N-1) of the last word (n)
4 + N		EPC Id: Second byte (N) of the last word (n)
4 + N + 1		Password: First Byte
4 + N + 2		Password: Second Byte
4 + N + 3		Password: Third Byte
4 + N + 4		Password: Fourth Byte
4 + N + 4 + 1		Bench
4 + N + 4 + 1 + 1		Start Address: First Byte
...		
2 + N + 4 + 1 + 4		Start Address: Fourth Byte
2 + N + 4 + 1 + 4 + 1		Number w of word to read (max 64)

Byte 2 (length byte) is used to indicate the length of the EPC Id field without relying on the corresponding PC word. This provides a level of resilience in case of an incorrect or corrupted PC word.

Fail:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0x19	Command code of the original request
4	0x15	NAK

Fails, tag not present:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0x19	Command code of the original request
4	0x01	No tag

Note – The TAG not present here is considered as an error.

Fails, read failed:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0x19	Command code of the original request
4	0x02	Write failed or tag does not support reading from the specified block

Successful:

#	Value	
1	...	Length (LSB)
2	...	Length (MSB)
3	0x19	Command code of the original request
4	0x00	OK
4 + 1		First byte of data read
4 + 2		Second byte of data read
...		
4 + W - 1		First byte (W - 1) of the last block (w) of the data read
4 + W		Second byte(W) of the last block (w) of data read

1.3.12 0x1A – Writing data to the TAG

It is used to write data to the memory of an identified ISO 18000-63 TAG.

The command includes the EPC code identifying the TAG (n words, including the PC word and CRC), the password (4 bytes), the memory bank (1 byte), the initial memory address (relative to the specified bank) from which to start writing (4 bytes), the number w of blocks (words) to be written (1 byte, maximum 64 blocks), and the W bytes corresponding to the w blocks to be written.

NOTE - For historical and technical reasons, *word* (2 bytes) is used:

- n word \rightarrow N bytes, with $N = 2 * n$
- w blocks \rightarrow W bytes, with $W = 2 * w$

When the password not set: 4 bytes per 0x00.

Values used to identify memory banks:

Bank Index	Description
0x00	<i>Reserved</i>
0x01	<i>EPC</i>
0x02	<i>TID</i>
0x03	<i>User</i>

Note: since the 0x1A a "generic" writing command, it is limited to writing the blocks indicated in the indicated bank; if the EPC bank is specified, unlike the specific programming command, the CRC is not automatically calculated and the PC word is not recalculated or the consistency with the current PC word is checked.

ATTENTION! Altering an EPC without properly updating the PC word and CRC can lead to its corruption.

The response will be multi-packet containing the requested K bytes.

Data to be sent (to which the usual two bytes of length are added at the beginning):

#	Value	
1	0x1A	Command
2		
3		
4		Length of EPC Id in word (n), e.g. 8 for Id from 6 word
4 + 1		EPC Id: First byte of the first word
4 + 2		EPC Id: Second byte of the first word
...		
4 + $N - 1$		EPC Id: First byte ($N-1$) of the last word (n)
4 + N		EPC Id: Second byte (N) of the last word (n)
4 + $N + 1$		Password: First Byte
4 + $N + 2$		Password: Second Byte

$4 + N + 3$		Password: Third Byte
$4 + N + 4$		Password: Fourth Byte
$4 + N + 4 + 1$		Bench
$4 + N + 4 + 1 + 1$		Start Address: First Byte
...		
$4 + N + 4 + 1 + 4$		Start Address: Fourth Byte
$4 + N + 4 + 1 + 4 + 1$		Number w of word to write (max 64)
$4 + N + 4 + 1 + 4 + 1 + 1$		Data to be written: first byte of the first block
$4 + N + 4 + 1 + 4 + 1 + 2$		Data to be written: second byte of the first block
...		
$4 + N + 4 + 1 + 4 + 1 + W - 1$		Data to be written: first byte ($W - 1$) of block w
$4 + N + 4 + 1 + 4 + 1 + W$		Data to be written: second byte (W) of block w

NOTE - Byte 2 (length byte) is used to indicate the length of the EPC Id field without relying on the corresponding PC word. This provides a level of resilience in case of an incorrect or corrupted PC word.

Fail:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0x1A	Command code of the original request
4	0x15	NAK

Fails, tag not present:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0x1A	Command code of the original request
4	0x01	No tag

Note – The TAG not present here is considered as an error.

Failed:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0x1A	Command code of the original request
4	0x02	Write failed or tag does not support reading from the specified block

Successful:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0x1A	Command code of the original request
4	0x00	OK

1.3.13 0x1C – Kill the TAG

Burn (*kill*) an identified ISO 18000-63 TAG.

The command includes the EPC code identifying the TAG (n words, including the PC word and CRC) and the kill password (4 bytes).

NOTE - For historical and technical reasons, *word* (2 bytes) is used:

- n word \rightarrow N bytes, with $N = 2 * n$

The kill-password is present in the Reserved (0x00) bank and must be set to make the command effective.

Data to be sent (to which the usual two bytes of length are added at the beginning):

#	Value	
1	...	Length (LSB)
2	...	Length (MSB)
3	0x1C	Command
4		Length of EPC Id in word (n), e.g. 8 for Id from 6 word
4 + 1		EPC Id: First byte of the first word
4 + 2		EPC Id: Second byte of the first word
...		
4 + N - 1		EPC Id: First byte (N -1) of the last word (n)
4 + N		EPC Id: Second byte (N) of the last word (n)
4 + N + 1		Kill-password: first byte
4 + N + 2		Kill-password: second byte
4 + N + 3		Kill-password: third byte
4 + N + 4		Kill-password: fourth byte

Byte 2 (length byte) is used to indicate the length of the EPC Id field without relying on the corresponding PC word. This provides a level of resilience in case of an incorrect or corrupted PC word.

Fail:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0x1C	Command code of the original request
4	0x15	NAK

Fails, tag not present:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0x1C	Command code of the original request

4	0x01	No tag
---	------	--------

Note – The TAG not present here is considered as an error.

Failed:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0x1C	Command code of the original request
4	0x02	Write failed or tag does not support reading from the specified block

Successful:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0x1C	Command code of the original request
4	0x00	OK

1.3.14 **0xE9** – Get Product Code

The command is used to receive the unique product key stored on the device.

Command:

#	Value	
1	0x01	Length (LSB)
2	0x00	Length (MSB)
3	0xE9	Command code of the original request

Fail:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0xE9	Command code of the original request
4	0x15	NAK

Successful:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0xE9	Command code of the original request
4	0x00	OK
5..20	...	Bytes with data (16 ASCII characters)

1.3.15 0x06 – Read Database Data

The command is used to read from the circular buffer that represents the device's internal database.

The reader inserts each tag found into the internal buffer, forming a database managed in FIFO mode.

The command returns the most recent records (tag info), possibly removing them from the queue, based on the parameters specified:

- TMax: Maximum number of records returned
- Bytes indicating whether or not to remove records after return

For each record returned, in addition to the EPC itself, the length of the EPC and, if available, the RSSI and antenna values are indicated. If not available, these values are set to 0xFF.

Data to be sent (to which the usual two bytes of length should be added at the beginning):

#	Value	
1	0x03	Length (LSB)
2	0x00	Length (MSB)
3	0x06	Command
4		Maximum number of tags to return, from 0x01 to 0xFF
5		Indicates whether to remove tags from the queue after return: 0x01: Tags are removed 0x00: Tags are not removed

Fail:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0x06	Command code of the original request
4	0x15	NAK

Attempt to run with continuous mode not active:

#	Value	
1	0x03	Length (LSB)
2	0x00	Length (MSB)
3	0x06	Command code of the original request
4	0x15	NAK
5	0x01	Continuous mode not active

Success with no records present:

#	Value	
1	0x02	Length (LSB)

2	0x00	Length (MSB)
3	0x06	Command code of the original request
4	0x00	OK

Single-packet response

Define:

- T the number of tags returned, $T > 0$ and $T < T_{MAX}$
- n_j : the length in word of the j -th tag, for $j \in \{1 \dots T\}$
- $N_j = n_j * 2$ the length in bytes b , of the j -th tag, for $j \in \{1 \dots T\}$

Successful Response format, for $T > 0$, has the following response:

#	Value	
1	...	Length (LSB)
2	...	Length (MSB)
3	0x06	Command code of the original request
4	0x00	OK
4 + 1	(n_1)	TAG #1 - Length of EPC Id in word ($n_1 \Rightarrow N_1$)
4 + 2		TAG #1 - EPC Id: first byte of the first word
4 + 3		TAG #1 - EPC Id: second byte of the first word
...		
4 + 1 + $N_1 - 1$		TAG #1 - EPC Id: first byte of the word n_1
4 + 1 + N_1		TAG #1 - EPC Id: second byte of word n_1
4 + 1 + $N_1 + 1$		TAG #1 - Nr. Antenna associated with the TAG
4 + 1 + $N_1 + 2$		TAG #1 - RSSI value associated with the TAG
4 + 1 + $N_1 + 2 + 1$	(n_2)	TAG #2 - Length of EPC Id in word ($n_2 \Rightarrow n_2$)
4 + 1 + $N_1 + 2 + 2$		TAG #2 - EPC Id: first byte of the first word
4 + 1 + $N_1 + 2 + 3$		TAG #2 - EPC Id: second byte of the first word
...		
4+2+ N_1 +2+ $N_2 - 1$		TAG #2 - EPC Id: first byte of the word n_2
4+2+ N_1 +2+ N_2		TAG #2 - EPC Id: second byte of the word n_2
4+2+ N_1 +2+ N_2 +1		TAG #2 - Nr. Antenna associated with the TAG
4+2+ N_1 +2+ N_2 +2		TAG #2 - RSSI value associated with the TAG
...		(Other TAGS)
4+T-1+) + $1 \sum_{j=1}^{T-1} (N_j + 2$	(n_T)	TAG #T - Length of EPC Id in word ($n_T = > N_T$)
4+T-1+) + $2 \sum_{j=1}^{T-1} (N_j + 2$		TAG #T - EPC Id: first byte of the first word
4+T-1+) + $3 \sum_{j=1}^{T-1} (N_j + 2$		TAG #T - EPC Id: second byte of the first word
...		
4+T-1+) $\sum_{j=1}^{T-1} (N_j + 2 + N_T - 1$		TAG #T - EPC Id: first byte of the n_T word
2+T-1+) $\sum_{j=1}^{T-1} (N_j + 2 + N_T$		TAG #T - EPC Id: second byte of the n_T word
2+T-1+) $\sum_{j=1}^T (N_j + 2 + 1$		TAG #T - Nr. Antenna associated with the TAG
2+T-1+) $\sum_{j=1}^T (N_j + 2 + 2$		TAG #T - RSSI value associated with the TAG

1.3.16 0x07 – Number of records in the Database

The command is used to get the number of records in the device queue.

Comando:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0x07	Command

Fail:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0x07	Command code of the original request
4	0x15	NAK

Attempt to run with continuous mode not active:

#	Value	
1	0x03	Length (LSB)
2	0x00	Length (MSB)
3	0x07	Command code of the original request
4	0x15	NAK
5	0x01	Continuous mode not active

Success with no records present:

#	Value	
1	0x06	Length (LSB)
2	0x00	Length (MSB)
3	0x07	Command code of the original request
4	0x00	OK
5,6,7,8	...	Number of records present. Four bytes little-endian.

1.3.17 0x08 – Reset Database Data

The command is used to command the cleanup of the database which, after the command has been executed, will be "empty" (the execution of the read commands will give 0 records).

Comando:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0x08	Command

Fail:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0x08	Command code of the original request
4	0x15	NAK

Success with no records present:

#	Value	
1	0x02	Length (LSB)
2	0x00	Length (MSB)
3	0x08	Command code of the original request
4	0x00	OK

2 Command transmission in TCP mode

The commands described in the previous chapter can be sent to the device using TCP mode.

The device ID, IP address, and PORT to connect to must be known.

Since the TCP protocol does not define a message structure, we apply a transformation to the original command bytes according to the following rules.

The general structure of the message is as follows:

#	Value	
0x01	SOH	Start of message
		Device address (ASCII-encoded format)
0x02	STX	Start of command
		Command content
0x03	ETX	End of command (ASCII-encoded format)
	CRC	Checksum character
0x0D	CR	End of message

2.1 ASCII-Encoded Format

The transformation of a byte into ASCII-encoded format is performed by representing it with two ASCII characters: the first character corresponds to the ASCII encoding of the high nibble of the byte, and the second character corresponds to the ASCII encoding of the low nibble.

For example, if the value of a byte is 0xFE, the ASCII-encoded version is (0x46, 0x45). That's because 0xFE → ('F', 'E') → (0x46, 0x45).

The ASCII-encoded version of a byte sequence is the ordered concatenation of the ASCII-encoded representation of each individual byte.

2.2 CRC Calculation

The checksum (CRC) is the XOR of all bytes in the message, starting from SOH and ending at ETX, after the message has been wrapped and ASCII-encoded according to the previously explained rules.

Moreover, the following special rule is applied: resulting CRC is incremented by 1 if it is equal to 0x01, 0x04 or 0x0D.

Below is an example of a CRC calculation algorithm written in C#.

```
private byte CalculateCrc8(byte[] data)
{
```

```

    if (data == null || data.Length < 2)
        throw new ArgumentException("Message must have at least SOH and CR.");

    byte crc = 0x00;
    for (int i = 0; i < data.Length; i++)
    {
        crc ^= data[i];
    }

    // Special rule, if needed
    if (crc == 0x01 || crc == 0x04 || crc == 0x0D)
    {
        crc ++;
    }

    return crc;
}

```

Request for configuration of section 0:

Inventory request with result “No TAGs found”: