Inveo Sp. z o.o.

# RFID USB Desk

Instruction Manual

# Overview

**Warning**

**This manual applies only to the firmware version v1.37 onwards. Inveo does not guarantee that the information contained in this document applies to previous firmware revisions.**


## Purpose of the device

The USB Desk is a desktop RFID card and tag reader that can be connected directly to the PC's USB port.

It's designed for reading various RFID tags and transponders, offering a wide array of configuration options. Thanks to the configuration software, the user can modify the method of code reading, setting prefixes, suffixes and code formatting, making integration with various systems easy.

The reader can operate in keyboard emulation or virtual serial port mode, allowing for flexible integration with computer software. The user can define various readout parameters such as bit negation, bit and byte swap, tag length, readout speed and many more. The device can be integrated with proprietary software thanks to the virtual serial port mode, making full device control and reaction for read transponder codes possible.

It supports multiple popular RFID transponder standards, making it a comprehensive solution for various applications, from access control to facility monitoring.

## Changelog

**1.2** 4th of December 2025

- Firmware revision v1.38
  - Compatibility patch for AMD CPU's

**1.1** 18th of September 2025

- Firmware revision v1.37
  - Windows 11 compatibility patch

**1.0** 14th of March 2025

- Firmware revision v1.32

Table of contents

# Device construction

## Technical characteristics

**Information**

**Power supply: USB 5V DC (The device is powered from the PC's USB port)**

**Transponders depending on the device's version:**

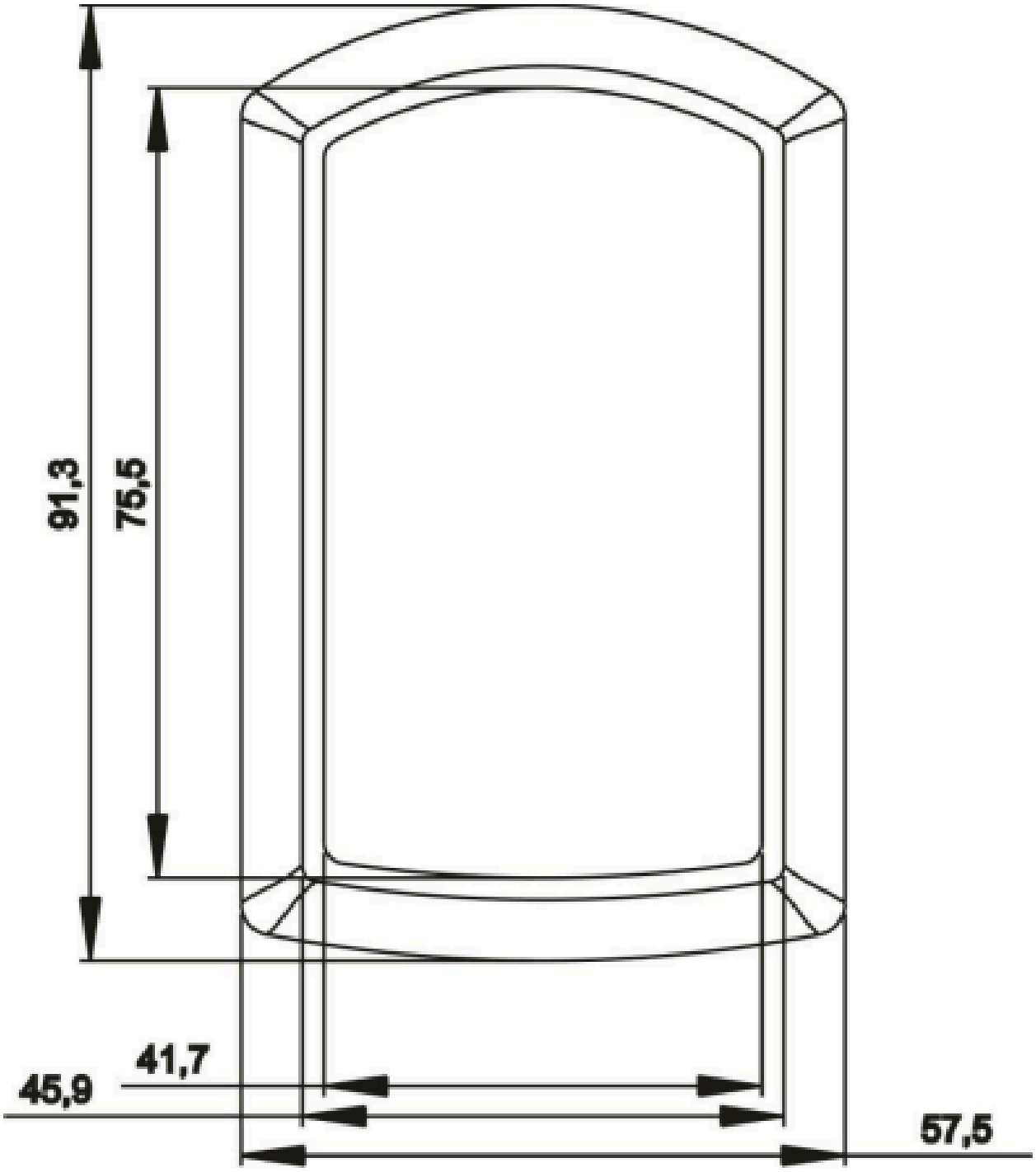
| Product name | Supported transponders |
|---|---|
| RFID Desk Mif | Mifare Classic® (ISO/IEC 14443-A), Mifare Plus® (UID), Mifare DESFire® (UID) |
| RFID Desk Uni | Unique EM4100 EM4102 |
| RFID Desk Ico | ICODE® (ISO 15693) |
| RFID Desk HT1 | HITAG 1/S |
| RFID Desk HT2 | HITAG 2 |

**General view**

## General characteristics

| General characteristics | Description |
|---|---|
| **Modes of operation** | The device can operate in three modes:<br><br>- **HID** (standard keyboard emulation)<br>- **CDC** (Virtual Com Port)<br>- **WebUSB API** (communication via a web page) |
| **LED indicator** | The device is equipped with three LED indicators that can be turned on or react to:<br><br>- **TAG readout**,<br>- **TAG application**,<br>- **power supply connection** |
| **Sound signal** | Can be disabled or enabled to generate a signal after a successful RFID tag read |
| **Weight** | ca. 52g (1,8oz) |

### Dimensions

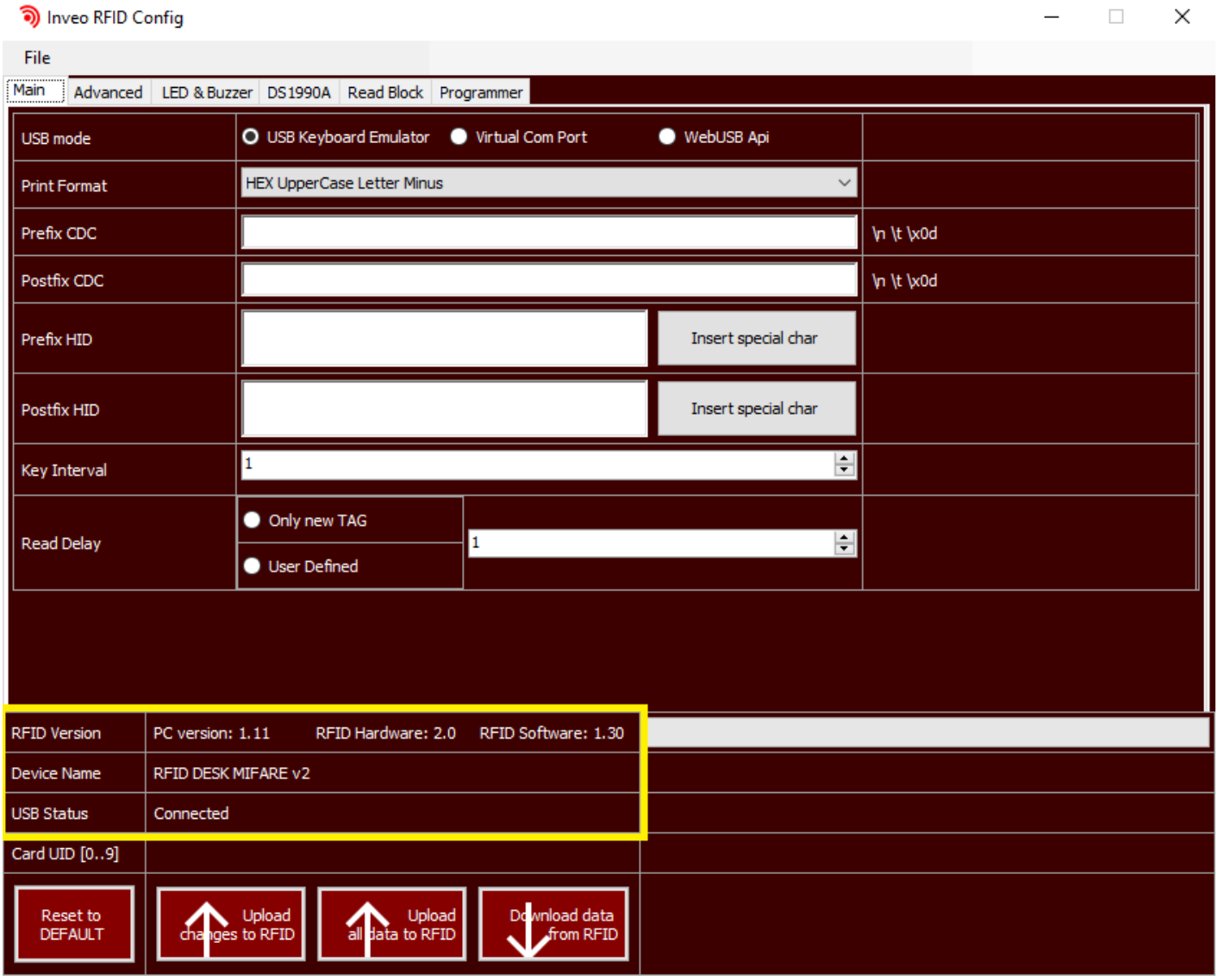



## Device configuration

### Basic settings

To configure the device, download the free **Inveo RFID Config** programme from the **INVEO** website.

After installing and running the programme, connect the reader to the PC. In the programme's bottom secion, the current USB port (to which the device is connected) status and basic reader information should be displayed.:




- **PC version** – PC application version,
- **RFID Hardware** – RFID reader hardware revision,
- **RFID Software** – RFID reader software revision.

If the reader was detected by the programme, configuration can begin.

The **USB mode** of operation should be selected first. Available options:

- **USB Keyboard emulator** – Standard keyboard emulation (HID),
- **Virtual Com Port** – Virtual COM port communication (CDC),
- **WebUSB Api** – Communication via a web page.

## Keybord emulation mode

In keyboard emulation mode, the reader behaves like a standard USB keyboard. After an RFID tag is applied, its number is stored in the space selected by the cursor in an appropriate format.

## Virtual CDC serial port mode

If the **Virtual Com Port** mode is selected, the reader sends the applied tag's UID number to a virtual serial port. In this mode both the sound and LED indicators can be controlled.

Sending the byte to the virtual com port causes a reaction:

| Bit | Function | dec | hex | Description |
|---|---|---|---|---|
| **0** |  | 1 | 1 | LED 1 |
| **1** |  | 2 | 2 | LED 2 |
| **2** |  | 4 | 4 | LED 3 |
| **3** |  | 8 | 8 | Continuous tone |
| **4** |  | 16 | 10 | Ascending tone |
| **5** |  | 32 | 20 | Descending tone |
| **6** | * | 64 | 40 | **0** – Standard operating mode, the code is sent after a card is applied<br>**1** – The code is sent by request only |
| **7** | ** | 128 | 80 | **1** – Card code send request |

**Tip**

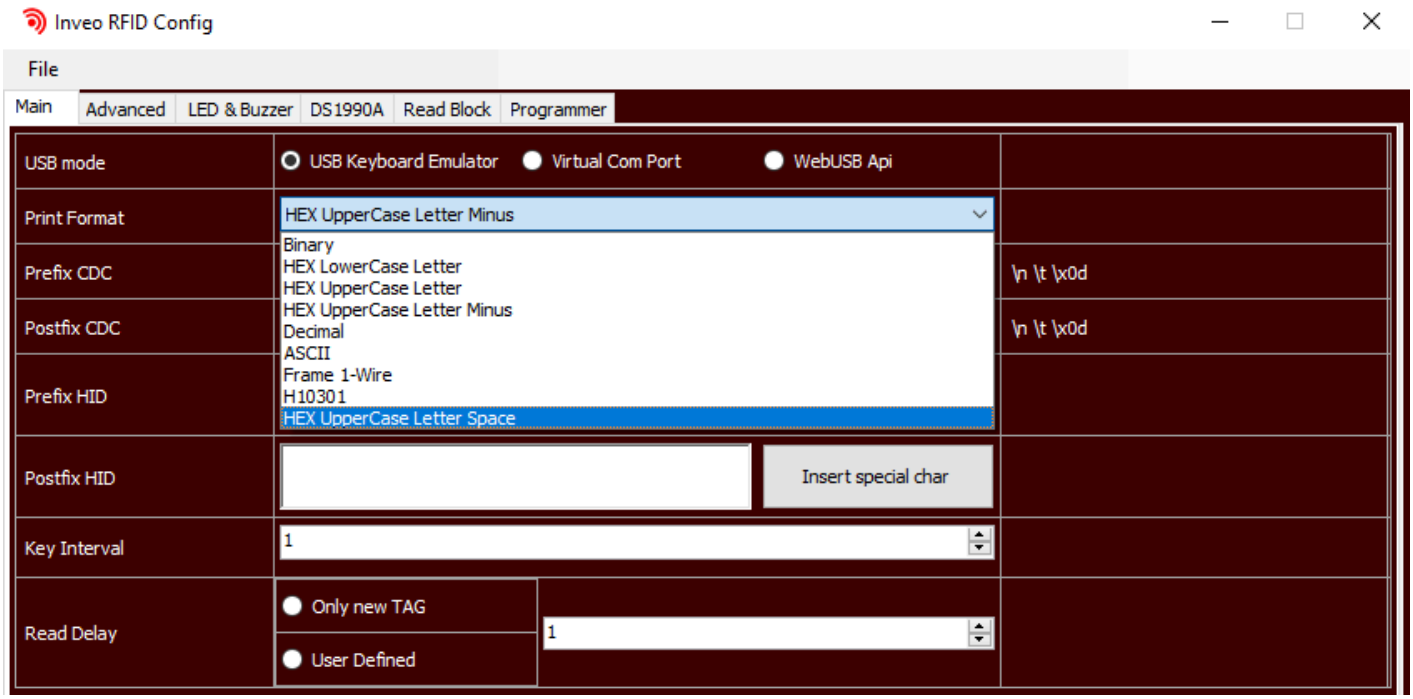
**If a request (7th bit) is sent but there is no card in the read field, the device will not return a value.**

**Examples**

- To activate **LED1** and **LED 3**, value of 5 (0x05) should be sent,
- To activate the sound signal (continuous tone, bit 3), value of 8 (0x08) should be sent,
- To activate the sound signal (momentary tone, bit 5), **LED1**, and **LED3**, value of 37 (0x25) should be sent,
- To send a card read request, value of 192 (0xC0 - sum of bit 6 and 7) should be sent.

## Number sending format

The next step is to set the format for sending the read card number.




The following output data formats are available:

- **Binary** – The card code bytes are sent in the binary format (applies to the **Virtual Com Port** mode),
- **HEX LowerCase Letter** – The tag code is sent in the hexadecimal format with lower case letters,
- **HEX UpperCase Letter** – The tag code is sent in the hexadecimal format with upper case letters,
- **HEX UpperCase Letter Minus** – The tag code is sent in the hexadecimal format with upper case letters, dashes separating the code bytes:

| Data format | Example |
|---|---|
| **HEX LowerCase Letter** | `aabbccdd` |
| **HEX UpperCase Letter** | `AABBCCDD` |
| **HEX UpperCase Letter Minus** | `AA-BB-CC-DD` |

- **Decimal** – The tag is converted to decimal format:

**Example**

**Tag read:** `A6-0A-9D-95` **Value displayed:** `2785713557`

- **ASCII** – Decoding of card data to the ASCII format (applies to Mifare tags). The string must end with a "0".

- **Frame 1-Wire** – DS2401 emulation.
- **H10301**

**Prefix CDC** – Is a string of numbers and letters sent before a tag is read in the virtual serial port mode.

**Postfix CDC** – Is a string of numbers and letters sent after a tag is read in the virtual serial port mode.

**Prefix HID** – Is a string of numbers and letters or special characters (Alt, Enter, key combination) sent before a tag is read in the keyboard emulation mode.

**Postfix HID** – Is a string of numbers and letters or special characters (Alt, Enter, key combination) sent after a tag is read in the keyboard emulation mode.

> **Tip**
>
> **To introduce a delay equal to the time specified in Key Interval, enter the character "ý". No characters will be sent, but the transmission of subsequent characters will be delayed.**
> **To delay character transmission by e.g. 100ms, for Key Interval=10, enter 10 "ý" characters – 10x10ms=100ms**

**Key Interval** – the delay between sent characters. This setting is used on slower devices such as tablets. If the device to which the reader is connected isn't fast enough to input the UID or characters defined in the Prefix/Postfix fields correctly (eg. doesn't input all characters), define a longer time delay between characters (Key Interval). The default value is 10.

**Read Delay - Only new TAG** – readout of the same tag is possible only after the defined time has passed, other tags are read immediately. This setting prevents the card from being read multiple times. The delay time is defined in x * 0,1s.

**Read Delay - User Defined** – User defined tag readout delay time. The delay time is defined in x * 0,1s.

To define a special character to be sent as a HID prefix or postfix, click **Insert special char** next to the appropriate position.

Using the virtual keyboard window, any key and character combination can be created.

**Example**

The reader should display the following text string when a card is applied:
**User logged in: (UID NUMBER HERE). and input enter key.**

First, the **Prefix HID** field should be filled with: "User logged in:"

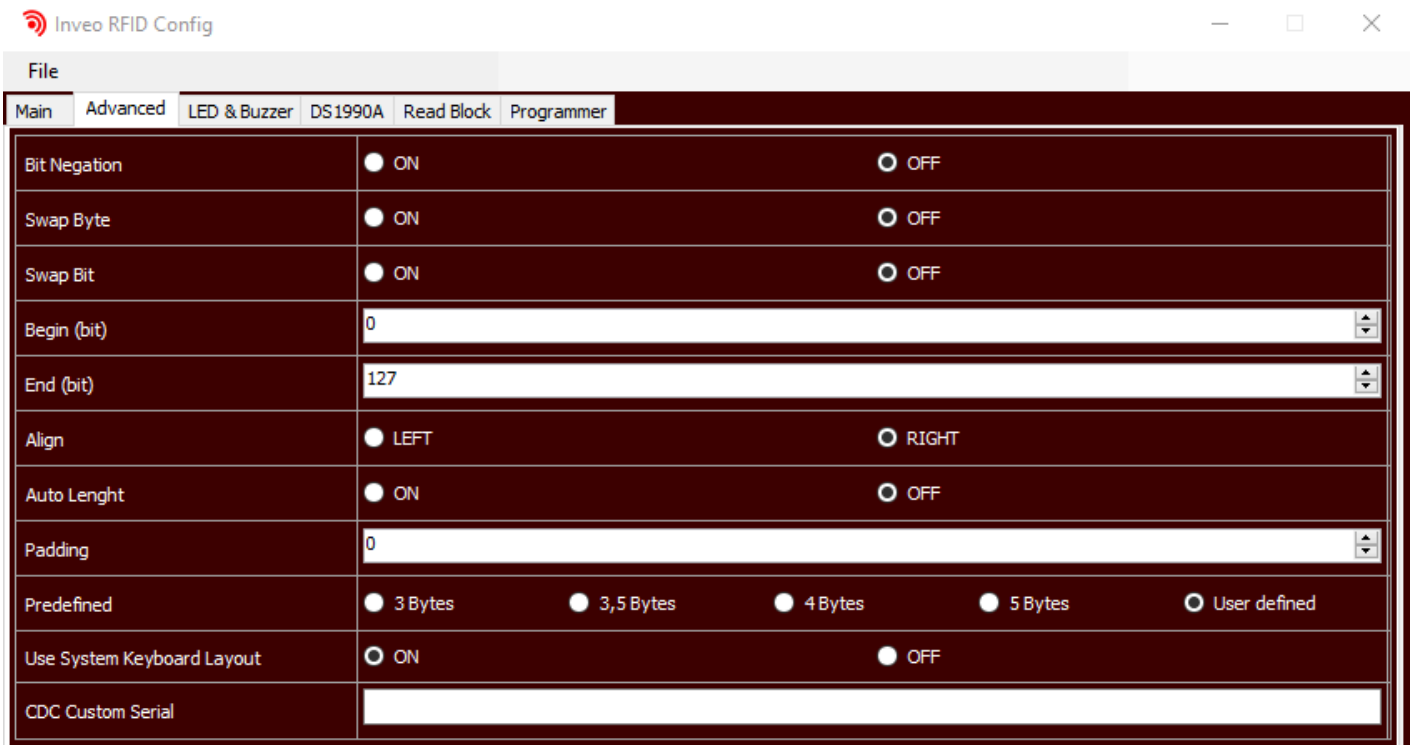
In the **Postfix HID** field, "." should be inserted and an Enter key pressed on the virtual keyboard.

To confirm the settings, it's required to press the **Upload data to RFID** button.

Once this is done, every tag application will cause this string to display:
"User logged in: `UIDnumber`."

## Advanced conversion functions

Under the **Advanced** tab, advanced code readout options are located.




The application allows modification of the read code in a wide range.
The following options are available:

**Bit Negation** – If selected, the read values are negated:

| | HEX | BIN |
|---|---|---|
| **Read tag** | 04-00-1B-B7-BC | 00000100-00000000-00011011-10110111-10111100 |
| **Modified tag** | FB-FF-E4-48-43 | 11111011-11111111-11100100-01001000-01000011 |

**Swap Byte** – Function enabling byte swap:

| | HEX | BIN |
|---|---|---|
| **Read tag** | 04-00-1B-B7-BC | 00000100-00000000-00011011-10110111-10111100 |
| **Modified tag** | BC-B7-1B-00-04 | 10111100-10110111-00011011-00000000-00000100 |

**Swap Bit** – Function causing the bit string to change:
Replacement of the first bit with the last one, the second bit with the second last one etc.

| Before modification HEX | Before modification BIN | After modification BIN | After modification HEX |
|---|---|---|---|
| 04BC | 0000 0100 1011 1100 | 0011 1101 0010 0000 | 3D20 |

| | HEX | BIN |
|---|---|---|
| **Read tag** | 04-00-1B-B7-BC | 00000100-00000000-00011011-10110111-10111100 |
| **Modified tag** | 3D-ED-D8-00-20 | 00111101-11101101-11011000-00000000-00100000 |

**Begin (bit)** – Number of the bit the reader begins the UID readout with.

**End (bit)** – Number of the bit the reader ends the readout operation with.

Input of a wrong value to the **Begin / End (bit)** field will result in a change to the entire tag value, because the bits of each byte will be shifted:

The UID code is in a 0-39 range. The user filled the fields as follows:
**Begin (bit)** = 1
**End (bit)** = 40
correct TAG: 04-00-1B-B7-BC → TAG read: 08-00-37-6F-78,

| Begin | End | TAG (in bytes) | TAG (in bits) |
|---|---|---|---|
| 0 | 39 | 04-00-1B-B7-BC | **0100000000000001101110110111101111100** |
| 1 | 40 | 08-00-37-6F-78 | **1000000000000011011101101111011110000** |

The entire tag (in bits) was shifted, that's why the value in bytes is completely different.

**Align** – Alignment of the read tag,

**Example 1**

Bytes 3, 5 are to be read off the tag: 01-0F-A0-D2-61
**Begin (bit):** 12
**End (bit):** 39
**Auto Length:** ON

| Operation | Value |
|---|---|
| Complete tag code (bytes) | 01-0F-A0-D2-61 |
| Complete tag code (bits) | 00000001 00001111 10100000 11010010 01100001 |
| Byte to be aligned | XF-A0-D2-61 |
| Align RIGHT (B) | FA-0D-26-10 |
| Align RIGHT (b) | 11111010 00001101 00100110 00010000 |
| Align LEFT (B) | 0F-A0-D2-61 |
| Align LEFT (b) | 00001111 10100000 11010010 01100001 |

**Example 2**

Bytes 1, 5 are to be read off the tag: 05-00-EE-9C-86
**Begin (bit):** 28
**End (bit):** 39
**Auto Length:** ON

| Operation | Value |
|---|---|
| Complete tag code (bytes) | 05-00-EE-9C-86 |
| Complete tag code (bits) | 00000101 00000000 11101110 10011100 10000110 |
| Byte to be aligned | XC-86 |
| Align RIGHT (B) | C8-60 |
| Align RIGHT (b) | 11001000 01100000 |
| Align LEFT (B) | 0C-86 |
| Align LEFT (b) | 00001100 10000110 |

**Auto Length** – Automatic adjustment of the read code length.

**Padding** – Manual setting of the code value length. The value of 1 equals to one character.

**Example**

UNIQUE standard, each UID code has 5 bytes (for HEX format).
Input 0 / 39 to the Begin / End (bit) fields respectively.
In the Padding field input the value of 10:
From Begin (bit)=0 to End (bit) = 39 → 40 bits = 5 bytes (for the HEX format it will be 10 characters → Padding = 10).

**Predefined** – Predefined values for **Begin (bit)** and **End (bit)**.

**Use System Keyboard Layout** – The system keyboard layout is taken into consideration when letters and numbers are inputted.

**Example**

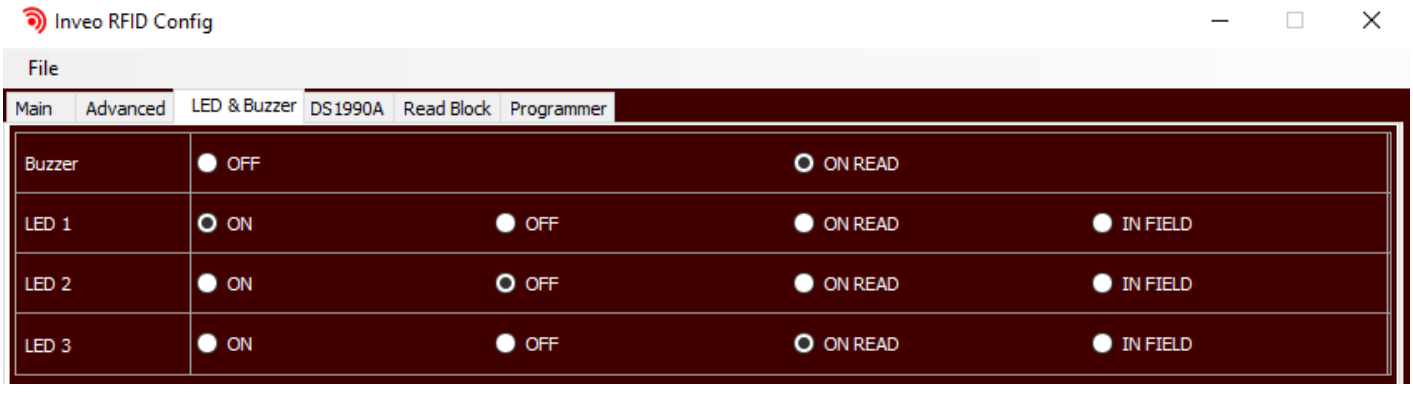
Polish keyboard layout:

Hungarian keyboard layout:


**CDC Custom Serial** – Optional number that can be read in the virtual COM port mode.

## LED's and sound

The **Inveo RFID Config** programme enables simple configuration of visual and sound signalling. All you have to do is to go to the **LED & Buzzer** tab and select the appropriate settings.




For the buzzer, two options can be selected:

- **OFF** – Buzzer off,
- **ON READ** – Sound a signal when a tag is read.

| Icon | Name | Description |
| --- | --- | --- |

| | | |
|---|---|---|
| | LED 1 | By default, the diode informs about the fact that power is connected to the module. LED colour - green. |
| | LED 2 | By default, the diode is not used. Led colour - red. |
| | LED 3 | By default, the diode informs about a successful card read. LED colour - green. |

The user can configure the LED's according to his needs. Each LED can be configured in 4 ways:

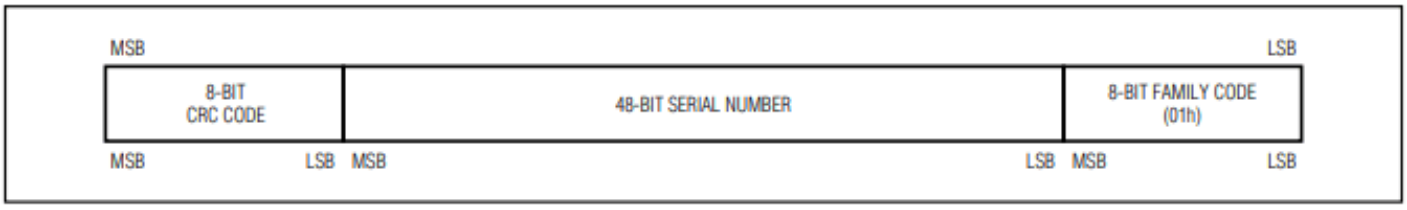
- **ON** – The LED glows constantly when power is connected,
- **OFF** – LED disabled,
- **ON READ** – The LED signals a successful tag read,
- **IN FIELD** – The LED signals that a tag has been applied and glows when it is in range of the module.

## DS1990A

iButton DS1990A emulation.

The iButton by default contains the following data in its memory:

- 8-bit CRC code
- 48 bit serial number
- 8 bit family code (0x01)



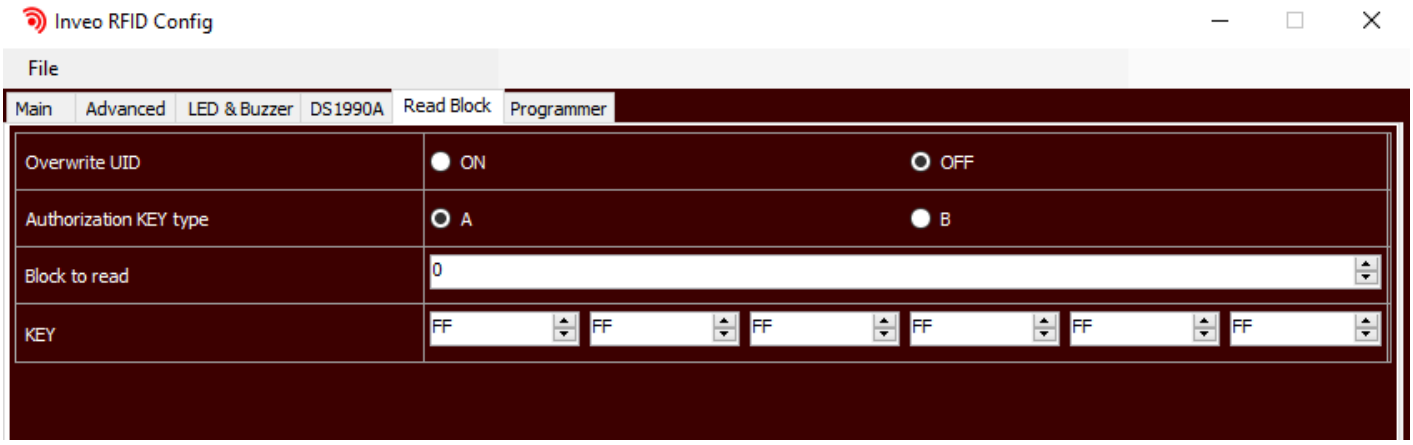

If the **Frame 1-Wire** print format is selected, the reader will format the read card code to the DS1990A format.

**Example**

| Description | Tag (HEX) |
|---|---|
| **Read tag** | A6-0A-9D-95 |
| **Format** | Family codeRead card numberAddressCRC |
| **Modified tag** | 01a60a9d9500000f |
| **UpperCase** | 01A60A9D9500000F |
| **Swap** | 0f0000959d0aa601 |

## Mifare block reading

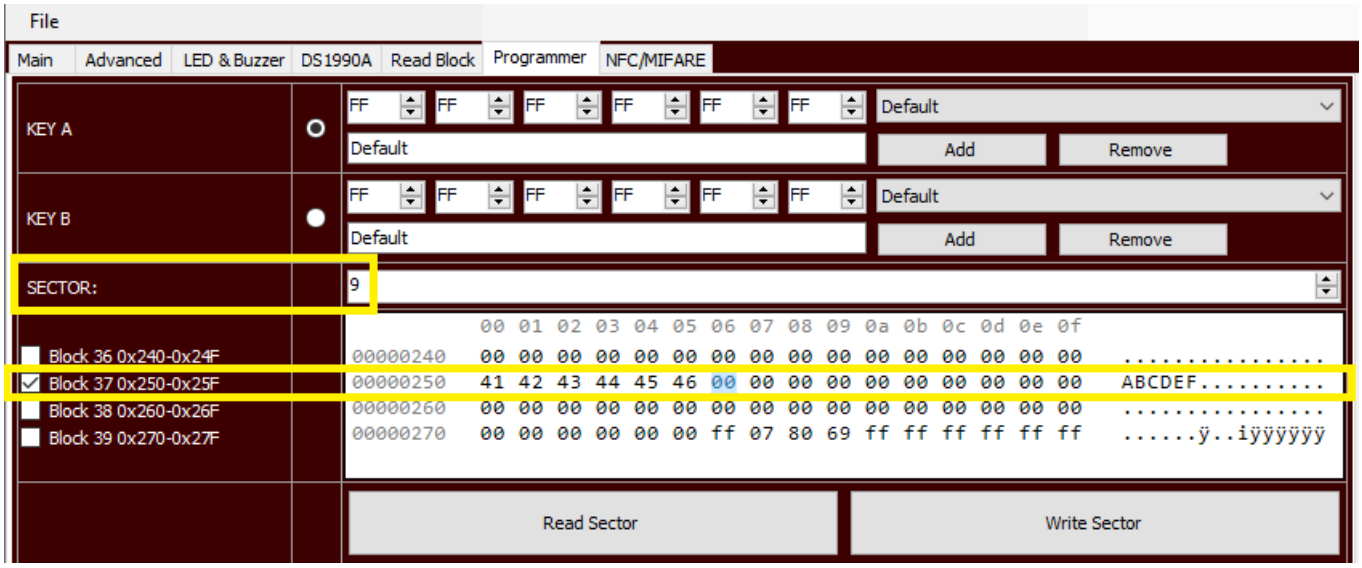
The reader has the ability to read Mifare Classic 1k/4k card memory blocks. Any single block can be read from the tag's memory.




**Overwrite UID** – The function enables block readout. Enabling this function will overwrite the UID number with the read block value, **Authorization KEY type** – Selection of the authorization key (Key A or Key B), **Block to read** – Number of the block to be read, **KEY** – Authentication key, **FF-FF-FF-FF-FF-FF** by default.

**Example**

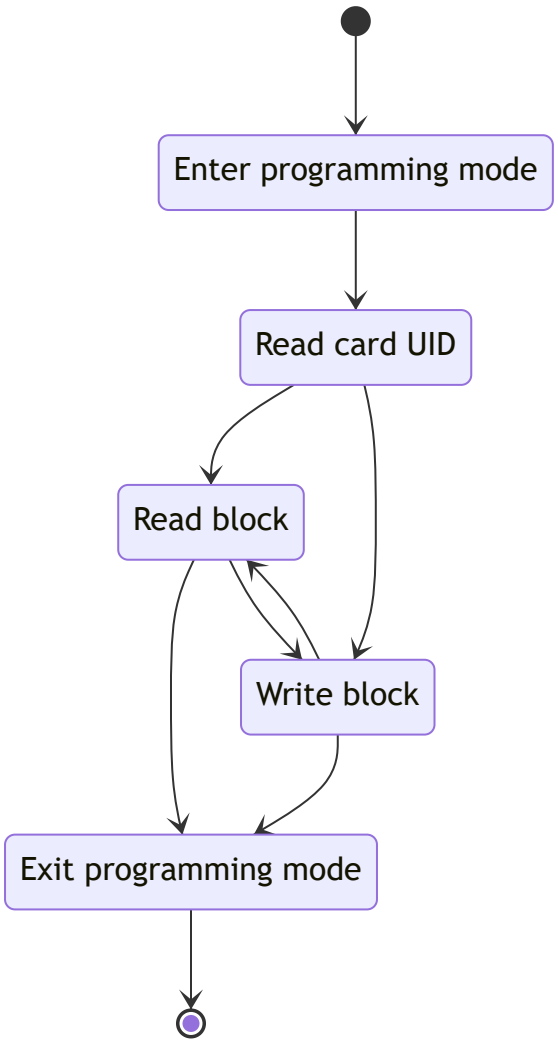
We have a Mifare tag where in the block 37 (SECTOR 9, block 1), a string of ABCDEF ending with a zero is located.




We want to display this character string after a card is applied to the reader.
The **Overwrite UID** field should be set to **ON**, the **Block to read** field should be set to **37**, in the **Main -> Print Format** tab, set **ASCII**. After saving the data in the reader's memory and applying an RFID tag, the **ABCDEF** string will be displayed.

## Reading and writing blocks in CDC (Virtual COM port) mode

It is possible to safely read and write data from/to Mifare data blocks during Virtual COM port operation. Any operation on the block data requires authentication.
Multiple read and write operations can be conducted during a single programming mode "session".




**1. Enter programming mode**

Frame: `0x00 0x01 0x02 0x03`

**2. Read card UID**

> **Note**
>
> **The card UID should be read before conducting any operations on block data (reading/writing)**

Frame: `0x0F`

The reader returns the UID length and its value.

- `0x00` – No card present in the read field.

**Example**

`0x04 0x23 0x34 0x23 0x12`

### 3. Read block

Frame: `0x0C A/B SecAddr Res1 Res2 KEY[6]`

- `0x0C` – Enter block read mode,
- **A/B** – Authorization key A or B selection:
  - `0x00` – Key A,
  - `0x01` – Key B.
- **SecAddr** – Block address,
- **Res1, Res2** – Unused,
- **KEY** – 6 key bytes (default: `0xFF 0xFF 0xFF 0xFF 0xFF 0xFF`).

The reader will return `0x00` and 16 data bytes if the read operation was successful, or an error code and 16 zeroes if it failed.

**Example**

**Read block 16:** `0x0C 0x00 0x10 0x00 0x00 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF`

### 3. Write sector

Frame: `0x0D A/B SecAddr Res1 Res2 KEY[6] DATA[16]`

- `0x0D` – Enter block write mode,
- **A/B** – Authorization key A or B selection:
  - `0x00` – Key A,
  - `0x01` – Key B.
- **SecAddr** – Block address,
- **Res1, Res2** – Unused,
- **KEY** – 6 key bytes (default: `0xFF 0xFF 0xFF 0xFF 0xFF 0xFF`),
- **DATA** – 16 data bytes.

**Information**

**It is mandatory to enter all 16 data bytes while writing the block to avoid corruption.**

The reader will return `0x00` if the operation was successful.

**4. Exit programming mode**

Frame: `0x0B`

**Example: Reading the UID, blocks 4 and 6, writing and reading block 6 using the key A FF FF FF FF FF FF**

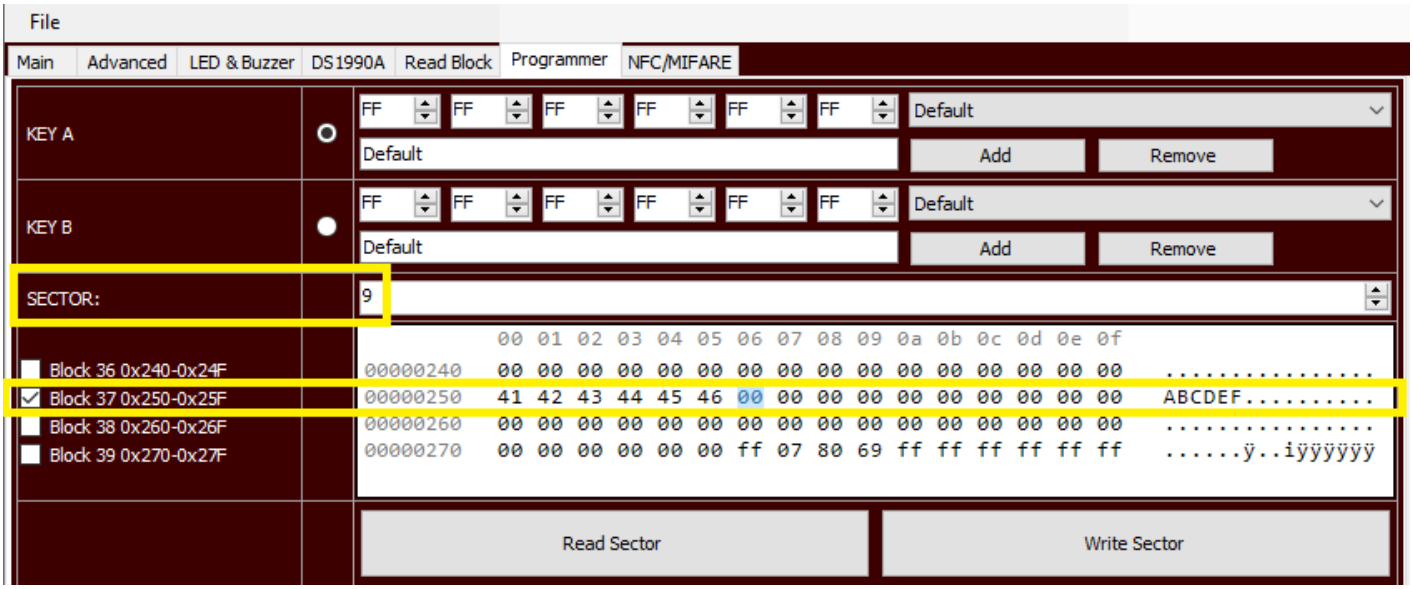
**S - data sent to the rader**

**R - data received from the reader**

```
S: 0x00 0x01 0x02 0x03 //Enter programming mode
S: 0x0F //Card UID read request
R: 0x04 0x71 0x7A 0xA1 0x1B //UID length: 4 bytes, UID: 71 7A A1 1B
S: 0x0C 0x00 0x04 0x00 0x00 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF //Block 4 read request
R: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 //Response with
block 4 contents
S: 0x0C 0x00 0x06 0x00 0x00 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF //Block 6 read request
R: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 //Response with
block 6 contents
S: 0x0D 0x00 0x06 0x00 0x00 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08
0x09 0x0A 0x0B 0x0C 0x0D 0x0E 0x0F 0x10 //Block 6 write request
R: 0x00 //Write operation successful
S: 0x0C 0x00 0x06 0x00 0x00 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF //Block 6 read request
R: 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0A 0x0B 0x0C 0x0D 0x0F 0x10 //Response with
block 6 contents
S: 0x0B //Exit programming mode
```

# Programmer

The programmer enables reading and writing data to a Mifare card




Editing of the data is done by input of the corresponding values:

- Selection of the appropriate key (A or B),
- Input of the key,
- Selection of a sector to be read or written,

- Editing the HEX or ASCII values,
- Selection of the block we want to save.

As shown above we are writing sector 9 (block 37 0x250 - 0x25f) with the **ABCDEF** string

The A and B keys are set to default values - **0xFF,0xFF,0xFF,0xFF,0xFF,0xFF**

After clicking the **Write Sector** button, the tag will be saved with corresponding values.

## Factory settings

There are two methods of restoring the factory settings:

1. Using the **RFID Config** programme
    - Run the **RFID Config** programme.
    - Connect the device to the PC.
    - In the **RFID Config** programme, press the **Reset to DEFAULT** button.
2. Using the button
    - Power up the device
    - Press and hold the RESET button for 10 to 15 seconds,
    - Release the button when a continuous sound alert starts to play.

# Warranty and manufacturer's liability

> **Warning**
>
> **The manufacturer provides a two-year warranty for the device and post-warranty service for a period of 10 years from the date of introduction of the device to the market. The warranty covers all material and production defects.**

The manufacturer undertakes to respect the warranty agreement, if the following conditions are met:

- All repairs, changes, expansions and device calibrations are carried out by the manufacturer or an authorized service center,
- The power supply system meets the applicable standards,
- The device is operated in accordance with the suggestions presented in this manual,
- The device is operated in accordance with its intended purpose.

The manufacturer assumes no responsibility for consequences resulting from improper installation, improper use of the device, failure to comply with the instruction manual, and repairs made by unauthorized personnel.

> **Warning**
>
> **The device contains no user serviceable parts inside.**

## Storage, operation and transport conditions

**The device should be stored in enclosed rooms, where the atmosphere is free from vapours and corrosive substances:**

- Environment temperature from -30°C to +60°C (-22°F - 140°F),
- Humidity from 25% to 90% (condensation unacceptable),
- Atmospheric pressure from 700 to 1060 hPa.

**The device is intended to operate in the following conditions:**

- Environment temperature from -10°C do +55°C (14°F - 131°F),
- Humidity from 30% to 75%,
- Atmospheric pressure from 700 to 1060 hPa.

**Recommended transport conditions:**

- Environment temperature from -40°C do +85°C (-40°F - 185°F),
- Humidity from 5% to 95%,
- Atmospheric pressure from 700 to 1060 hPa.

**Installation and device operation:**

- The module should be operated in accordance with recommendations provided later in this manual.

## Disposal and decommissioning

In an event the device needs to be decommissioned (eg. after its intended life period is surpassed), it is recommended to contact the manufacturer or his representative, who are responsible to respond appropriately, i.e., to collect the device from the user. The user can alternatively contact companies specializing in electronic device or computer equipment disposal and/or decommissioning. Under no condition should the device be placed with other waste.