**PRG\_22-14 IDTRONIC LEUZE RFID SYSTEMS**

**HF BASIC SERIAL COMMUNICATION PROTOCOL AND DEVICE CONFIGURATION**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Changelist |
| 01 | 08/01/2024 | Raffaele Picco | First draft |
| 02 | 29/01/2024 | Fabrizio Picotto | First release |
| 03 | 16/02/2024 | Fabrizio Picotto | Added “tag type” field description in the response of “set trigger on” message in “read mode of serial number”  Defined the response of the “get firmware version” message |
| 04 | 14/03/2024 | Nicolò Garolla | Add more tags support (MIFARE, …)  Fixed response in “read configuration” command  Add “firmware download” command |

# Table of Contents

[1 Table of Contents 3](#_Toc164152937)

[2 Scope 6](#_Toc164152938)

[3 Field of Application 6](#_Toc164152939)

[4 Definitions and Abbreviations 6](#_Toc164152940)

[5 Transponder (Tag) Types 7](#_Toc164152941)

[6 Configuration of the Device 8](#_Toc164152942)

[6.1 Configuration AFI (Application Family Identifier) Filter (Address 00h) 8](#_Toc164152943)

[6.2 Configuration Functions Register 1 (Address 01h) 8](#_Toc164152944)

[6.3 Configuration Functions Register 2 (Address 02h) 9](#_Toc164152945)

[6.4 Configuration Transponder Type (Addresses 03h-04h) 10](#_Toc164152946)

[6.5 Configuration Trigger / Output Switch (Addresses 05h-09h) 11](#_Toc164152947)

[6.5.1 Configuration Trigger Mode (Address 05h) 11](#_Toc164152948)

[6.5.2 Configuration Trigger Pulse Time (Addresses 06h-07h) 12](#_Toc164152949)

[6.5.3 Configuration Output Pulse Time (Addresses 08h-09h) 13](#_Toc164152950)

[6.6 Configuration Start Address Read (Addresses 0Ah-0Bh) 13](#_Toc164152951)

[6.7 Configuration Read Number of Blocks (Address 0Ch) 13](#_Toc164152952)

[6.8 Configuration Start Address Write (Addresses 0D-0Eh) 13](#_Toc164152953)

[6.9 Configuration Write Number of Blocks (Address 0Fh) 13](#_Toc164152954)

[6.10 Configuration Write data (Addresses 10h-57h) 14](#_Toc164152955)

[7 Message Structure of the Device 15](#_Toc164152956)

[7.1 Command Codes 15](#_Toc164152957)

[8 Response Structure of the Device 16](#_Toc164152958)

[8.1 Acknowledgement Codes 16](#_Toc164152959)

[8.2 Error Codes 16](#_Toc164152960)

[9 Messages Definitions of the Device 17](#_Toc164152961)

[9.1 Get Firmware Version 17](#_Toc164152962)

[9.2 Reset to Default 17](#_Toc164152963)

[9.3 Reset Software 17](#_Toc164152964)

[9.4 Set Trigger On 18](#_Toc164152965)

[9.5 Set Trigger Off 19](#_Toc164152966)

[9.6 Inventory 20](#_Toc164152967)

[9.7 Set Output 20](#_Toc164152968)

[9.8 Switch Field 21](#_Toc164152969)

[9.9 Read Configuration 21](#_Toc164152970)

[9.10 Write Configuration 22](#_Toc164152971)

[9.11 Read Block 22](#_Toc164152972)

[9.12 Read Transponder 23](#_Toc164152973)

[9.13 Write Block 24](#_Toc164152974)

[9.14 Firmware Download 25](#_Toc164152975)

[10 Transponder (Tag) Specific Information 26](#_Toc164152976)

[10.1 Memory Organization NXP I-CODE 1 26](#_Toc164152977)

[10.2 Memory Organization NXP I-CODE SLI 26](#_Toc164152978)

[10.2.1 Unique Identifier (UID) NXP I-CODE SLI 27](#_Toc164152979)

[10.3 Memory Organization NXP I-CODE SLI-S 27](#_Toc164152980)

[10.3.1 Unique Identifier (UID) NXP I-CODE SLI-S 27](#_Toc164152981)

[10.4 Memory Organization NXP I-CODE SLI-L 27](#_Toc164152982)

[10.4.1 Unique Identifier (UID) NXP I-CODE SLI-L 28](#_Toc164152983)

[10.5 Memory Organization NXP I-CODE SLIX 28](#_Toc164152984)

[10.5.1 Unique Identifier (UID) NXP I-CODE SLI 28](#_Toc164152985)

[10.6 Memory Organization NXP I-CODE SLIX-S 29](#_Toc164152986)

[10.6.1 Unique Identifier (UID) NXP I-CODE SLIX-S 29](#_Toc164152987)

[10.7 Memory Organization NXP I-CODE SLIX-L 29](#_Toc164152988)

[10.7.1 Unique Identifier (UID) NXP I-CODE SLIX-L 30](#_Toc164152989)

[10.8 Memory Organization NXP I-CODE SLIX2 30](#_Toc164152990)

[10.8.1 Unique Identifier (UID) NXP I-CODE SLIX-L 30](#_Toc164152991)

[10.9 Memory Organization TI Tag-it HF-I Standard 31](#_Toc164152992)

[10.9.1 Unique Identifier (UID) TI Tag-it HF-I Standard 31](#_Toc164152993)

[10.10 Memory Organization TI Tag-it HF-I Plus 31](#_Toc164152994)

[10.10.1 Unique Identifier (UID) TI Tag-it HF-I Plus 32](#_Toc164152995)

[10.11 Memory Organization TI Tag-it HF-I Pro 32](#_Toc164152996)

[10.11.1 Unique Identifier (UID) TI Tag-it HF-I Pro 32](#_Toc164152997)

[10.12 Memory Organization STM LRI 512 33](#_Toc164152998)

[10.12.1 Unique Identifier (UID) STM LRI 512 33](#_Toc164152999)

[10.13 Memory Organization Infineon my-d (02P) 33](#_Toc164153000)

[10.13.1 Unique Identifier (UID) Infineon my-d (02P) 33](#_Toc164153001)

[10.14 Memory Organization Infineon my-d (10P) 34](#_Toc164153002)

[10.14.1 Unique Identifier (UID) Infineon my-d (10P) 34](#_Toc164153003)

[10.15 Memory Organization EM EM4135 34](#_Toc164153004)

[10.15.1 Unique Identifier (UID) EM EM4135 35](#_Toc164153005)

[10.16 Memory Organization Fujitsu MB89R118C 35](#_Toc164153006)

[10.16.1 Unique Identifier (UID) Fujitsu MB89R118C 35](#_Toc164153007)

# Scope

This document refers to the Leuze HF Basic serial device and describes in detail the communication protocol and the configuration parameters.

# Field of Application

This document applies to the device HF Basic serial device with firmware version v1.00.

# Definitions and Abbreviations

| Term / Abbreviation | Definition |
| --- | --- |
| **TBD** | To Be Determined |
| **UID** | Unique Identifier |

# Transponder (Tag) Types

In many commands and acknowledgements, the transponder type (tag type) is also transmitted. Listed in the following table are the supported transponder types.

| Tag Type | Transponder Type | Number of Bytes | Start Block / Pages (when Writing) | Number of Blocks / Pages | Block / Page Size |
| --- | --- | --- | --- | --- | --- |
| 01h | NXP I-CODE 1 | 44 | 5 | 11 | 4 |
| 02h | STM LRI 512 | 60 | 0 | 16 | 4 |
| 03h | Reserved | - | - | - | - |
| 04h | NXP I-CODE SLI | 112 | 0 | 28 | 4 |
| NXP I-CODE SLI-S | 160 | 0 | 40 | 4 |
| NXP I-CODE SLI-L | 32 | 0 | 8 | 4 |
| 05h | Infineon my-d (02P) | 224 | 0 | 56 | 4 |
| Infineon my-d (10P) | 992 | 0 | 248 | 4 |
| 06h | EM EM4135 | 288 | 13 | 36 | 8 |
| 07h | TI Tag-it HF-I Standard | 32 | 0 | 8 | 4 |
| TI Tag-it HF-I Plus | 256 | 0 | 64 | 4 |
| TI Tag-it HF-I Pro | 32 | 0 | 8 | 4 |
| 08h | NXP I-CODE SLIX | 112 | 0 | 28 | 4 |
| NXP I-CODE SLIX-S | 160 | 0 | 40 | 4 |
| NXP I-CODE SLIX-L | 32 | 0 | 8 | 4 |
| 09h | NXP I-CODE SLIX2 | 320 | 0 | 80 | 4 |
| 0Ah | Fujitsu MB89R118C | 2000 | 0 | 250 | 8 |
| 0Bh | NXP MIFARE Classic 1k | 1024 | 0 | 64 | 16 |
| NXP MIFARE Classic 4k | 4096 | 0 | 256 | 16 |
| NXP MIFARE Ultralight C | 144 | 4 | 36 | 4 |
| NXP NTAG 210 | 48 | 4 | 12 | 4 |
| NXP NTAG 212 | 128 | 4 | 32 | 4 |
| NXP NTAG 213 | 144 | 4 | 36 | 4 |
| NXP NTAG 215 | 504 | 4 | 126 | 4 |
| NXP NTAG 216 | 888 | 4 | 222 | 4 |
| … | … | … | … | … | … |
| FEh | Reserved | - | - | - | - |
| FFh | Reserved | - | - | - | - |

# Configuration of the Device

The setting parameters of the device are stored in 16 different registers that can be accessed in read and write. The following table shows a list of the configuration registers:

| Address | Parameters / Function |
| --- | --- |
| 00h | AFI (Application Family Identifier) filter |
| 01h | Functions register 1 |
| 02h | Functions register 2 |
| 03h | Transponder type High byte |
| 04h | Transponder type Low byte |
| 05h | Trigger mode |
| 06h | Trigger pulse time (ms) High byte |
| 07h | Trigger pulse time (ms) Low byte |
| 08h | Output pulse time (ms) High byte |
| 09 h | Output pulse time (ms) Low byte |
| 0Ah | Start address Read High byte |
| 0B h | Start address Read Low byte |
| 0Ch | Read operation Number of blocks |
| 0D h | Start address Write High byte |
| 0Eh | Start address Write Low byte |
| 0Fh | Write operation Number of blocks |
| 10h-57h | Write data (max. 9 x 8 bytes) |
| 58h-FFh | Reserved |

## Configuration AFI (Application Family Identifier) Filter (Address 00h)

The AFI filter is a legitimation for the ISO15693 transponder in this application: only if the AFI on the transponder and the data stored in this register are the same, the transponder can be read or written.

*Default setting: 00h*

## Configuration Functions Register 1 (Address 01h)

| Bit | Function | Level | Description |
| --- | --- | --- | --- |
| 0…1 | Operation mode | 00 | Write Mode |
| 01 | Read Mode |
| 10 | Multiple Read |
| 2 | Reserved | 0 |  |
| 3 | Reserved | 0 |  |
| 4 | Trigger | 0 | Permanent ready for read |
| 1 | Read on trigger pulse |
| 5 | Read mode | 0 | Permanent read and data output |
| 1 | Single shot. Read once while in field |
| 6 | Write forward | 0 | Not active, a write command must be sent with the tag in the reading field |
| 1 | Active, a write command can be sent before the tag enters the field |
| 7 | Reserved | 0 |  |

The parameter to set is combined via Bit column. The MSB (most significant Bit) is Bit 7 on first position.

*Default setting: 71h*

The operation mode defines, what function a trigger pulse (or '+') causes. The factory setting is "Read", that means after a trigger the serial no. or data blocks is read (addresses 0A-0Ch). The response is the same as after an "N" command: state, block no. (or @0), tag type, data. With operation mode "Write" the stored data (address C10h following) is written into every tag after trigger, answer is "Q5". The operation mode "Multiple read" delivers the whole tag data on trigger pulse. Note that this operation takes more time (about double the time) than a read operation for one block.

## Configuration Functions Register 2 (Address 02h)

| Bit | Function | Level | Description |
| --- | --- | --- | --- |
| 0 | Serial number  (W and N command) | 0 | Not active, no transmission |
| 1 | Active, serial number must be transmitted |
| 1 | Anticollision | 0 | Not active, only one tag in field |
| 1 | Active, several tags in field |
| 2 | Reserved | 0 |  |
| 3 | Filter (AFI) | 0 | Not active |
| 1 | Active, AFI code in address 00h |
| 4 | Switching output | 0 | Not active |
| 1 | Automatically activated, address 05h |
| 5 | Data block size | 0 | 4 Byte |
| 1 | 8 Byte |
| 6 | Large data | 1 | Further data is sent automatically (> 256 bytes) |
| 7 | Reserved | 0 |  |

The parameter to set is combined via Bit column. The MSB (most significant Bit) is Bit 7 on first position.

*Default setting: 50h*

## Configuration Transponder Type (Addresses 03h-04h)

Address 03h:

| Bit | Description |
| --- | --- |
| 0 | Reserved |
| 1 | NXP I-CODE 1 |
| 2 | STM LRI 512 |
| 3 | Reserved |
| 4 | NXP I-CODE SLI  NXP I-CODE SLI-S  NXP I-CODE SLI-L |
| 5 | Infineon my-d (02P)  Infineon my-d (10P) |
| 6 | EM EM4135 |
| 7 | Tag-It HF-I Standard  Tag-It HF-I Plus |

The parameter to set is combined via Bit column. The MSB (most significant Bit) is Bit 7 on first position.

If the bit level is 1, the device operations are enabled for the corresponding transponder type.

*Default setting: 12h*

Address 04h:

| Bit | Description |
| --- | --- |
| 0 | NXP I-CODE SLIX  NXP I-CODE SLIX-S  NXP I-CODE SLIX-S |
| 1 | NXP I-CODE SLIX2 |
| 2 | Fujitsu MB89R118C |
| 3 | NXP MIFARE Classic 1k  NXP MIFARE Classic 4k |
| 4 | NXP MIFARE Ultralight C  NXP NTAG 210  NXP NTAG 212  NXP NTAG 213  NXP NTAG 215  NXP NTAG 216 |
| 5 | Reserved |
| 6 | Reserved |
| 7 | Reserved |

The parameter to set is combined via Bit column. The MSB (most significant Bit) is Bit 7 on first position.

If the bit level is 1, the device operations are enabled for the corresponding transponder type.

*Default setting: 00h*

## Configuration Trigger / Output Switch (Addresses 05h-09h)

The trigger is a combination of the trigger functionality and the trigger pulse time. For the output it is similar: the functionality and the output pulse time. In the address 05h the functionality for both output and trigger is combined. The trigger pulse time is in address 06/07h and the output pulse time in 08/09h.

### Configuration Trigger Mode (Address 05h)

Only the bits 0/1 of this byte are used for trigger and the Bits 3-5 for the output functionality. Other Bits are set to "0". Therefore the possible combinations are as follows:

| Value | Description |
| --- | --- |
| 00 | Trigger: read as long high level at input  Output: good read signal on low level |
| 01 | Trigger: read for time after positive slope  Output: good read signal on low level |
| 02 | Trigger: read after positive slope, time counts after negative slope  Output: good read signal on low level |
| 08 | Trigger: read as long high level on input  Output: no read signal on low level |
| 09 | Trigger: read for time after positive slope  Output: no read signal on low level |
| 0A | Trigger: read after positive slope, time counts after negative slope  Output: no read signal on low level |
| 20 | Trigger: read as long high level at input  Output: good read signal on high level |
| 21 | Trigger: read for time after positive slope  Output: good read signal on high level |
| 22 | Trigger: read after positive slope, time counts after negative slope  Output: good read signal on high level |
| 28 | Trigger: read as long high level on input  Output: no read signal on high level |
| 29 | Trigger: read for time after positive slope  Output: no read signal on high level |
| 2A | Trigger: read after positive slope, time counts after negative slope  Output: no read signal on high level |
| 03 | Trigger: read in multi tag mode  Output: good read signal on low level |
| 0B | Trigger: read in multi tag mode  Output: no read signal on low level |
| 23 | Trigger: read in multi tag mode  Output: good read signal on high level |
| 2B | Trigger: read in multi tag mode  Output: no read signal on high level |

*Default setting: 20h*

### Configuration Trigger Pulse Time (Addresses 06h-07h)

These registers store the value of the time after the trigger pulse, represented in hexadecimal numeric system. The time can be set in a range from 0 to 9000 ms.

*Default setting: 0000h*

**Examples:**

* 500 ms 01F4h
* 1000 ms 03E8h

### Configuration Output Pulse Time (Addresses 08h-09h)

These registers store the value of activation time for ‘good read’ or ‘no read’, represented in hexadecimal numeric system. The time can be set in a range from 30 to 9000 ms.

*Default setting: 33 (300 ms)*

**Examples:**

* 500 ms 01F4h
* 1000 ms 03E8h

## Configuration Start Address Read (Addresses 0Ah-0Bh)

These registers store the address of the first block read from the transponder after trigger in the operation mode ‘Read’.

*Default setting: 0000h*

**Example:**

* Block 05 0005h

## Configuration Read Number of Blocks (Address 0Ch)

This register stores the number of data blocks read from the transponder after trigger in the operation mode ‘Read’. The number of blocks can be set from 1 to 9.

*Default setting: 01h (1 block)*

**Examples:**

* 5 blocks 05h
* 9 blocks 09h

## Configuration Start Address Write (Addresses 0D-0Eh)

These registers store the address of the first block written into the transponder after trigger in the operation ‘Write’.

*Default settings: 0005h*

**Example:**

* Block 10 00A0h

## Configuration Write Number of Blocks (Address 0Fh)

This register stores the number of data blocks written into the transponder after trigger in the operation ‘Write’. The number of blocks can be set from 1 to 9.

*Default setting: 01h*

**Examples:**

* 5 blocks 05h
* 9 blocks 09h

## Configuration Write data (Addresses 10h-57h)

These registers store the data that is written into the transponder data blocks after trigger in the operation ‘Write’.

# Message Structure of the Device

For the data interface the Leuze protocol provides a baud rate of 9600, 1 start bit, 8 data bits, 1 stop bit and no parity bits. The structure of the messages is the following:

|  |  |  |
| --- | --- | --- |
| **STX** | **Payload** | **CR LF** |

Where:

|  |  |
| --- | --- |
| **STX** | =0x02, start of the message |
| **Payload** | Payload of the message |
| **CR LF** | =0x0D 0x0A, end of the message |

The data from and to the device is always coded in ASCII-Hex coding and always read out or written in complete data blocks. Usable as payload are all characters of the ASCII table.

Messages are recognized in capital letters as well as in small letters. To address the device several commands codes (in the standard message structure specified above) are defined.

## Command Codes

|  |  |
| --- | --- |
| Code | Command |
| **‘V/v’** | Get Firmware Version |
| **‘R/r’** | Reset to Default |
| **‘H/h’** | Reset Software |
| **‘+’** | Set Trigger On |
| **‘-‘** | Set Trigger Off |
| **‘I/i’** | Inventory |
| **‘A/a’** | Set Output |
| **‘F/f’** | Switch Field |
| **‘G/g’** | Read Configuration |
| **‘C/c’** | Write Configuration |
| **‘N/n’** | Read Blocks Data |
| **‘M/m’** | Read Transponder |
| **‘W/w’** | Write Blocks Data |
| **‘D/d’** | Firmware Upgrade |

# Response Structure of the Device

After receiving a command, the device sends a message back containing information about the result of the operation. The structure of a response message is the following:

|  |  |  |
| --- | --- | --- |
| **STX** | **Payload** | **CR LF** |

Where:

|  |  |
| --- | --- |
| **STX** | =0x02, start of the message |
| **Payload** | Payload of the message |
| **CR LF** | =0x0D 0x0A, end of the message |

To receive acknowledgment to specific commands and to recognize transmission errors, several acknowledgment and error codes (in the standard response structure specified above) are defined.

## Acknowledgement Codes

| Code | Description/meaning |
| --- | --- |
| **‘Q0’** | Command could not be carried out |
| **‘Q1’** | Configuration change carried out |
| **‘Q2’** | Action carried out |
| **‘Q4’** | Write command understood |
| **‘Q5’** | Data successfully written |

## Error Codes

| Code | Description/meaning |
| --- | --- |
| **‘E01’** | Invalid command |
| **‘E02’** | Invalid parameter |
| **‘E04’** | Data frame error |
| **‘E08’** | CRC checksum error |
| **‘E10’** | Controvert configuration settings |
| **‘E20’** | Firmware not valid |

# Messages Definitions of the Device

## Get Firmware Version

This command is used to get the current version of the firmware installed in the device.

**Command:**

**V**

Where:

|  |  |
| --- | --- |
| **V** | Command code |

**Response:**

**RDH 202 00 V x.y.z yyyy-mm-dd**

Where:

|  |  |
| --- | --- |
| **RDH 202 00** | Device name, it is a fixed field |
| **V x.y.z** | Version of release in the format major.minor.release, for example V 1.0.0 |
| **yyyy-mm-dd** | Date of release, for example 2024-02-16 |

## Reset to Default

This command is used perform a restart and to set the device to factory configuration.

**Command:**

**R**

Where:

|  |  |
| --- | --- |
| **R** | Command code |

**Response:**

**Q2** and **S**

Where:

|  |  |
| --- | --- |
| **Q2** | Action carried out |
| **S** | Ready for operation |

## Reset Software

This command is used to perform a software restart maintaining all the current settings.

**Command:**

**H**

Where:

|  |  |
| --- | --- |
| **H** | Command code |

**Response:**

**Q2**

Where:

|  |  |
| --- | --- |
| **Q2** | Action carried out |

## Set Trigger On

This command is used to turn the trigger on, activating a read or write operation depending on the configuration. Using only the command itself does not get a response from the device. The device sends a response when a transponder enters in the device reading/writing field and the operation is completed. Once the transponder enters the reading/writing field and the operation is completed the tigger turns off.

**Command:**

**+**

Where:

|  |  |
| --- | --- |
| **+** | Command code |

**Response, Read Mode, Serial Number:**

**F@0TagtypeSNR**

Where:

|  |  |
| --- | --- |
| **F** | Telegram flag:  =0: only 1 telegram is output  =1: multiple telegrams are output  (for more than 256 bytes out) |
| **@0** | The designator for following serial number |
| **Tagtype** | The transponder type |
| **SNR** | The serial number of the transponder |

**Response, Read Mode, Block Data**

**FB#TagtypeData**

Where:

|  |  |
| --- | --- |
| **F** | Telegram flag:  =0: only 1 telegram is output  =1: multiple telegrams are output  (for more than 256 bytes of data) |
| **B#** | Number of the first block read |
| **Tagtype** | The transponder type |
| **Data** | 1 to 9 blocks of the transponder starting from the first block read |

**Response, Read Mode, Multiple Read**

**FB#TagtypeData**

Where:

|  |  |
| --- | --- |
| **F** | Telegram flag:  =0: only 1 telegram is output  =1: multiple telegrams are output  (for more than 256 bytes of data) |
| **B#** | Number of the first block read |
| **Tagtype** | The transponder type |
| **Data** | All blocks of the transponder starting from the first block read |

**Response, Write Mode with Write Forward**

**Q5**

Where:

|  |  |
| --- | --- |
| **Q5** | Data successfully written |

## Set Trigger Off

This command is used to terminate the read process.

**Command:**

**-**

Where:

|  |  |
| --- | --- |
| **-** | Command code |

**Response:**

No answer. If no transponder was read, a NO READ (0x18) is output.

## Inventory

This command is used to get the serial number of the tags in the reading field of the device. It normally detects just one tag at every use of the command. If it’s needed to detect multiple tags in the reading field is necessary to activate the anticollision mode.

**Command:**

**I**

Where:

|  |  |
| --- | --- |
| **I** | Command code |

**Response:**

**F@0TagtypeSNR**

Where:

|  |  |
| --- | --- |
| **F** | Telegram flag:  =0: only 1 telegram is output  =1: multiple telegrams are output  (for more than 256 bytes out) |
| **@0** | The designator for following serial number |
| **SNR** | The serial number of the transponder |

Or if no transponder was read, a NO READ (0x18) is output**.**

## Set Output

This command is used to permanently set the output.

**Command:**

**Anxx**

Where:

|  |  |
| --- | --- |
| **A** | Command code |
| **n** | =0, output 1  =1, output 2 |
| **xx** | =FF, output on  =00, output off |

**Response:**

None.

## Switch Field

This command is used to switch on/off the RF field. The RF field is generally off. It is automatically switched on after a new trigger.

**Command:**

**Fx**

Where:

|  |  |
| --- | --- |
| **F** | Command code |
| **x** | =1, field on  =2, field off  =3, reset field |

**Response:**

**Q2**

Where:

|  |  |
| --- | --- |
| **Q2** | Action carried out |

## Read Configuration

This command is used to read the content of the configuration registers.

**Command:**

**Gxxxx**

Where:

|  |  |
| --- | --- |
| **G** | Command code |
| **xxxx** | =FF00: completely read out the configuration  =1000: only addresses 00 … 0Fh  =01xx: only one address |

**Response:**

**0Gxxyy**

Where:

|  |  |
| --- | --- |
| **xx** | The register (if only one address requested) |
| **yy** | The configuration read from the device |

## Write Configuration

This command is used to write the configuration data of the device.

**Command:**

**Cyyzz**

Where:

|  |  |
| --- | --- |
| **C** | Command code |
| **yy** | Address of the configuration register to write |
| **zz** | Configuration data to write |

**Response:**

**Q1**

Where:

|  |  |
| --- | --- |
| **Q1** | Configuration change carried out |

## Read Block

This command is used to read one or several blocks of data of a transponder.

**Command:**

**NB#TagtypeNOBSNR**

Where:

|  |  |
| --- | --- |
| **N** | Command code |
| **B#** | Number of the first block to read |
| **Tagtype** | The transponder type |
| **NOB** | Number of blocks to read, from 1 to 9 |
| **SNR** | Serial number of the transponder to be read. Is necessary if multiple transponders are in the field |

**Response:**

**FB#TagtypeData**

Where:

|  |  |
| --- | --- |
| **F** | Telegram flag:  =0: only 1 telegram is output  =1: multiple telegrams are output  (for more than 256 bytes of data) |
| **B#** | Number of the first block to read |
| **Tagtype** | The transponder type |
| **Data** | Content of the blocks of data specified by the command |

|  |  |
| --- | --- |
|  | It is absolutely necessary that a read process first be performed by means of trigger and that the transponder remains in the field |

|  |  |
| --- | --- |
|  | If anticollision is active, serial number transmission must be activated and the serial number of the desired transponder must be specified in the command. An average answer time of 50 ms can be assumed per data block |

## Read Transponder

This command is used to read the entire blocks of data of a transponder.

**Comand:**

**MTagtype**

Where:

|  |  |
| --- | --- |
| **M** | Command code |
| **Tagtype** | The transponder type |

**Response:**

**FTagtypeData**

Where:

|  |  |
| --- | --- |
| **F** | Telegram flag:  =0: only 1 telegram is output  =1: multiple telegrams are output  (for more than 256 bytes of data) |
| **Tagtype** | The transponder type |
| **Data** | All data beginning with block 0 |

|  |  |
| --- | --- |
|  | It is absolutely necessary that a read process first be performed by means of trigger and that the transponder remains in the field |

|  |  |
| --- | --- |
|  | This command works only with one transponder simultaneously in the reading field. If the transponder has more than 256 byte of data then the answer will be splitted. This command is not provided with the EM4135 |

## Write Block

This command is used to write one or several blocks of data of the transponder.

**Command:**

**WB#TagtypeNOBSNRData**

Where:

|  |  |
| --- | --- |
| **W** | Command code |
| **B#** | Number of the first block to write |
| **Tagtype** | The transponder type |
| **NOB** | Number of blocks to write, from 1 to 9 |
| **SNR** | Serial number of the transponder to be written. It is necessary if multiple transponders are in the field |
| **Data** | Data to be written (hexadecimal) for 1 block |

**Response:**

**yy**

|  |  |
| --- | --- |
| **yy** | =Q4: command understood (if write forward activated)  =Q5: write operation successful (after trigger)  =Q0: write operation failed |

|  |  |
| --- | --- |
|  | If write forward is disabled in the configuration registers, it is necessary to have a trigger operation before and the transponder must stay in the writing field. If write forward is enabled, the command is received even if the transponder is not in the writing field and the data is written after a trigger |

## Firmware Download

This command is used to download the firmware to the device.

**Command:**

**DBlockData**

Where:

|  |  |
| --- | --- |
| **D** | Command code |
| **Block** | The block number of firmware image (0000h for the first block, FFFFh for the last). |
| **Data** | The data of the block (64 bytes). Leave blank for the last block FFFFh |

**Response:**

**yy**

Where:

|  |  |
| --- | --- |
| **Q2** | Action carried out |
| **Q0** | Command could not be carried out |
| **E02** | Invalid parameter |
| **E20** | Firmware not valid |

# Transponder (Tag) Specific Information

## Memory Organization NXP I-CODE 1

| Block | Byte 0 | Byte 1 | Byte 2 | Byte 3 | Description |
| --- | --- | --- | --- | --- | --- |
| 0 | SNR0 | SNR1 | SNR2 | SNR3 | Serial number (low) |
| 1 | SNR4 | SNR5 | SNR6 | SNR7 | Serial number (high) |
| 2 | F0 | FF | FF | FF | Write access |
| 3 | x | x | x | x | Special functions |
| 4 | x | x | x | x | Filter code / App Id / User data |
| 5 | x | x | x | x | User data |
| 6 | x | x | x | x | User data |
| … | … | … | … | … | … |
| 14 | x | x | x | x | User data |
| 15 | x | x | x | x | User data |

## Memory Organization NXP I-CODE SLI

| Block | Bits | Description |
| --- | --- | --- |
| UID | 64 | Fixed serial number |
| 0 | 32 | User data |
| 1 | 32 | User data |
| … | … | … |
| 26 | 32 | User data |
| 27 | 32 | User data |

### Unique Identifier (UID) NXP I-CODE SLI

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 64 | 57 | 56 | 49 | 48 | 41 | 40 |  |  |  |  |  |  |  |  | 1 |
| E0 | | 04 | | 01 | | IC manufacturer serial number | | | | | | | | | |
| UID 7 | | UID 6 | | UID 5 | | UID 4 | | UID 3 | | UID 2 | | UID 1 | | UID 0 | |

To differentiate it from the other I-CODE types of transponder bits 37 and 36 are programmed to ‘00’.

## Memory Organization NXP I-CODE SLI-S

| Block | Bits | Description |
| --- | --- | --- |
| UID | 64 | Fixed serial number |
| 0 | 32 | User data |
| 1 | 32 | User data |
| … | … | … |
| 38 | 32 | User data |
| 39 | 32 | User data |

### Unique Identifier (UID) NXP I-CODE SLI-S

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 64 | 57 | 56 | 49 | 48 | 41 | 40 |  |  |  |  |  |  |  |  | 1 |
| E0 | | 04 | | 02 | | IC manufacturer serial number | | | | | | | | | |
| UID 7 | | UID 6 | | UID 5 | | UID 4 | | UID 3 | | UID 2 | | UID 1 | | UID 0 | |

To differentiate it from the other I-CODE types of transponder bits 37 and 36 are programmed to ‘00’.

## Memory Organization NXP I-CODE SLI-L

| Block | Bits | Description |
| --- | --- | --- |
| UID | 64 | Fixed serial number |
| 0 | 32 | User data |
| 1 | 32 | User data |
| … | … | … |
| 7 | 32 | User data |
| 8 | 32 | User data |

### Unique Identifier (UID) NXP I-CODE SLI-L

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 64 | 57 | 56 | 49 | 48 | 41 | 40 |  |  |  |  |  |  |  |  | 1 |
| E0 | | 04 | | 03 | | IC manufacturer serial number | | | | | | | | | |
| UID 7 | | UID 6 | | UID 5 | | UID 4 | | UID 3 | | UID 2 | | UID 1 | | UID 0 | |

To differentiate it from the other I-CODE types of transponder bits 37 and 36 are programmed to ‘00’.

## Memory Organization NXP I-CODE SLIX

| Block | Bits | Description |
| --- | --- | --- |
| UID | 64 | Fixed serial number |
| 0 | 32 | User data |
| 1 | 32 | User data |
| … | … | … |
| 26 | 32 | User data |
| 27 | 32 | User data |

### Unique Identifier (UID) NXP I-CODE SLI

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 64 | 57 | 56 | 49 | 48 | 41 | 40 |  |  |  |  |  |  |  |  | 1 |
| E0 | | 04 | | 01 | | IC manufacturer serial number | | | | | | | | | |
| UID 7 | | UID 6 | | UID 5 | | UID 4 | | UID 3 | | UID 2 | | UID 1 | | UID 0 | |

To differentiate it from the other I-CODE types of transponder bits 37 and 36 are programmed to ‘10’.

## Memory Organization NXP I-CODE SLIX-S

| Block | Bits | Description |
| --- | --- | --- |
| UID | 64 | Fixed serial number |
| 0 | 32 | User data |
| 1 | 32 | User data |
| … | … | … |
| 38 | 32 | User data |
| 39 | 32 | User data |

### Unique Identifier (UID) NXP I-CODE SLIX-S

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 64 | 57 | 56 | 49 | 48 | 41 | 40 |  |  |  |  |  |  |  |  | 1 |
| E0 | | 04 | | 02 | | IC manufacturer serial number | | | | | | | | | |
| UID 7 | | UID 6 | | UID 5 | | UID 4 | | UID 3 | | UID 2 | | UID 1 | | UID 0 | |

To differentiate it from the other I-CODE types of transponder bits 37 and 36 are programmed to ‘10’.

## Memory Organization NXP I-CODE SLIX-L

| Block | Bits | Description |
| --- | --- | --- |
| UID | 64 | Fixed serial number |
| 0 | 32 | User data |
| 1 | 32 | User data |
| … | … | … |
| 6 | 32 | User data |
| 7 | 32 | User data |

### Unique Identifier (UID) NXP I-CODE SLIX-L

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 64 | 57 | 56 | 49 | 48 | 41 | 40 |  |  |  |  |  |  |  |  | 1 |
| E0 | | 04 | | 03 | | IC manufacturer serial number | | | | | | | | | |
| UID 7 | | UID 6 | | UID 5 | | UID 4 | | UID 3 | | UID 2 | | UID 1 | | UID 0 | |

To differentiate it from the other I-CODE types of transponder bits 37 and 36 are programmed to ‘10’.

## Memory Organization NXP I-CODE SLIX2

| Block | Bits | Description |
| --- | --- | --- |
| UID | 64 | Fixed serial number |
| 0 | 32 | User data |
| 1 | 32 | User data |
| … | … | … |
| 77 | 32 | User Data |
| 78 | 32 | User data |
| 79 | 32 | Counter |

### Unique Identifier (UID) NXP I-CODE SLIX-L

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 64 | 57 | 56 | 49 | 48 | 41 | 40 |  |  |  |  |  |  |  |  | 1 |
| E0 | | 04 | | 03 | | IC manufacturer serial number | | | | | | | | | |
| UID 7 | | UID 6 | | UID 5 | | UID 4 | | UID 3 | | UID 2 | | UID 1 | | UID 0 | |

To differentiate it from the other I-CODE types of transponder bits 37 and 36 are programmed to ‘01’.

## Memory Organization TI Tag-it HF-I Standard

| Block | Bits | Description |
| --- | --- | --- |
| UID | 64 | Fixed serial number |
| 0 | 32 | User data |
| 1 | 32 | User data |
| … | … | … |
| 6 | 32 | User Data |
| 7 | 32 | User data |

### Unique Identifier (UID) TI Tag-it HF-I Standard

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 64 | 57 | 56 | 49 | 48 | 41 | 40 |  |  |  |  |  |  |  |  | 1 |
| E0 | | 07 | | C1 | | IC manufacturer serial number | | | | | | | | | |
| UID 7 | | UID 6 | | UID 5 | | UID 4 | | UID 3 | | UID 2 | | UID 1 | | UID 0 | |

## Memory Organization TI Tag-it HF-I Plus

| Block | Bits | Description |
| --- | --- | --- |
| UID | 64 | Fixed serial number |
| 0 | 32 | User data |
| 1 | 32 | User data |
| … | … | … |
| 62 | 32 | User Data |
| 63 | 32 | User data |

### Unique Identifier (UID) TI Tag-it HF-I Plus

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 64 | 57 | 56 | 49 | 48 | 41 | 40 |  |  |  |  |  |  |  |  | 1 |
| E0 | | 07 | | 01 | | IC manufacturer serial number | | | | | | | | | |
| UID 7 | | UID 6 | | UID 5 | | UID 4 | | UID 3 | | UID 2 | | UID 1 | | UID 0 | |

## Memory Organization TI Tag-it HF-I Pro

| Block | Bits | Description |
| --- | --- | --- |
| UID | 64 | Fixed serial number |
| 0 | 32 | User data |
| 1 | 32 | User data |
| … | … | … |
| 6 | 32 | User Data |
| 7 | 32 | User data |

### Unique Identifier (UID) TI Tag-it HF-I Pro

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 64 | 57 | 56 | 49 | 48 | 41 | 40 |  |  |  |  |  |  |  |  | 1 |
| E0 | | 07 | | C5 | | IC manufacturer serial number | | | | | | | | | |
| UID 7 | | UID 6 | | UID 5 | | UID 4 | | UID 3 | | UID 2 | | UID 1 | | UID 0 | |

## Memory Organization STM LRI 512

| Block | Bits | Description |
| --- | --- | --- |
| UID | 64 | Fixed serial number |
| 0 | 32 | User data |
| 1 | 32 | User data |
| … | … | … |
| 14 | 32 | User Data |
| 15 | 32 | User data |

### Unique Identifier (UID) STM LRI 512

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 64 | 57 | 56 | 49 | 48 |  |  |  |  |  |  |  |  |  |  | 1 |
| E0 | | 02 | | IC manufacturer serial number | | | | | | | | | | | |
| UID 7 | | UID 6 | | UID 5 | | UID 4 | | UID 3 | | UID 2 | | UID 1 | | UID 0 | |

## Memory Organization Infineon my-d (02P)

| Block | Bits | Description |
| --- | --- | --- |
| UID | 64 | Fixed serial number |
| 0 | 32 | User data |
| 1 | 32 | User data |
| … | … | … |
| 54 | 32 | User Data |
| 55 | 32 | User data |

### Unique Identifier (UID) Infineon my-d (02P)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 64 | 57 | 56 | 49 | 48 | 41 | 40 |  |  |  |  |  |  |  |  | 1 |
| E0 | | 05 | | 40 | | IC manufacturer serial number | | | | | | | | | |
| UID 7 | | UID 6 | | UID 5 | | UID 4 | | UID 3 | | UID 2 | | UID 1 | | UID 0 | |

## Memory Organization Infineon my-d (10P)

| Block | Bits | Description |
| --- | --- | --- |
| UID | 64 | Fixed serial number |
| 0 | 32 | User data |
| 1 | 32 | User data |
| … | … | … |
| 246 | 32 | User Data |
| 247 | 32 | User data |

### Unique Identifier (UID) Infineon my-d (10P)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 64 | 57 | 56 | 49 | 48 | 41 | 40 |  |  |  |  |  |  |  |  | 1 |
| E0 | | 05 | | 00 | | IC manufacturer serial number | | | | | | | | | |
| UID 7 | | UID 6 | | UID 5 | | UID 4 | | UID 3 | | UID 2 | | UID 1 | | UID 0 | |

## Memory Organization EM EM4135

| Block | Bits | Description |
| --- | --- | --- |
| UID | 64 | Fixed serial number |
| 13 | 64 | User data |
| 14 | 64 | User data |
| … | … | … |
| 47 | 64 | User Data |
| 48 | 64 | User data |

### Unique Identifier (UID) EM EM4135

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 64 | 57 | 56 | 49 | 48 |  |  |  |  |  |  |  |  |  |  | 1 |
| E0 | | 16 | | IC manufacturer serial number | | | | | | | | | | | |
| UID 7 | | UID 6 | | UID 5 | | UID 4 | | UID 3 | | UID 2 | | UID 1 | | UID 0 | |

## Memory Organization Fujitsu MB89R118C

| Block | Bits | Description |
| --- | --- | --- |
| UID | 64 | Fixed serial number |
| 0 | 64 | User data |
| 1 | 64 | User data |
| … | … | … |
| 248 | 64 | User Data |
| 249 | 64 | User data |

### Unique Identifier (UID) Fujitsu MB89R118C

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 64 | 57 | 56 | 49 | 48 | 41 | 40 |  |  |  |  |  |  |  |  | 1 |
| E0 | | 08 | | 01 | | IC manufacturer serial number | | | | | | | | | |
| UID 7 | | UID 6 | | UID 5 | | UID 4 | | UID 3 | | UID 2 | | UID 1 | | UID 0 | |