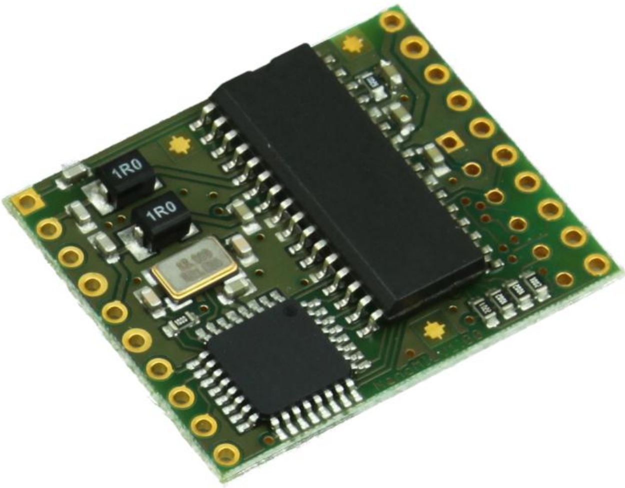
ELATEC
RFID Systems



# Transponder Reader
# Multi ISO
# Technical Manual

Doc.-Rev. 1.07

# Content

# 1. Introduction

This document is the reference guide for the transponder reader family TWN3 Multi ISO and Nano Module Multi ISO. The readers are using the same reading technology, so this document is applicable for both devices.

**Note:**

**In order to use the functionality, which is described in this document, your Multi ISO reader needs a firmware version V1.17 or above. In order to update the firmware from an elder version, please refer to chapter 8 “Firmware update”.**

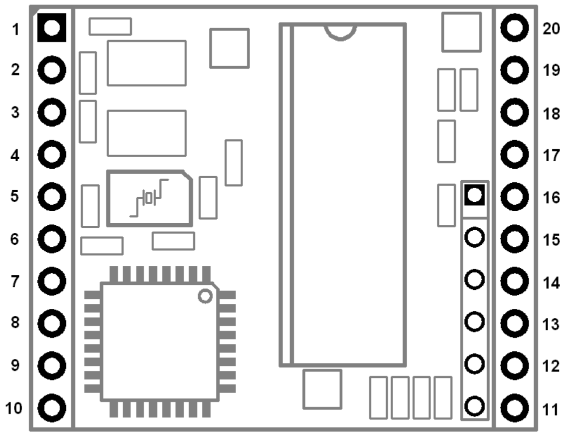
# 2. Installation of Nano Module Multi ISO

This chapter covers the installation of Nano Module Multi ISO in an embedded environment.

## 2.1 Mechanical Outline

Dimensions: 30.48mm x 25.40mm x 3.7mm (1.2” x 1” x 0.15”)

Connectors are in 2.54mm (0.1”) grid.



Component side shown

## 2.2 Pinning

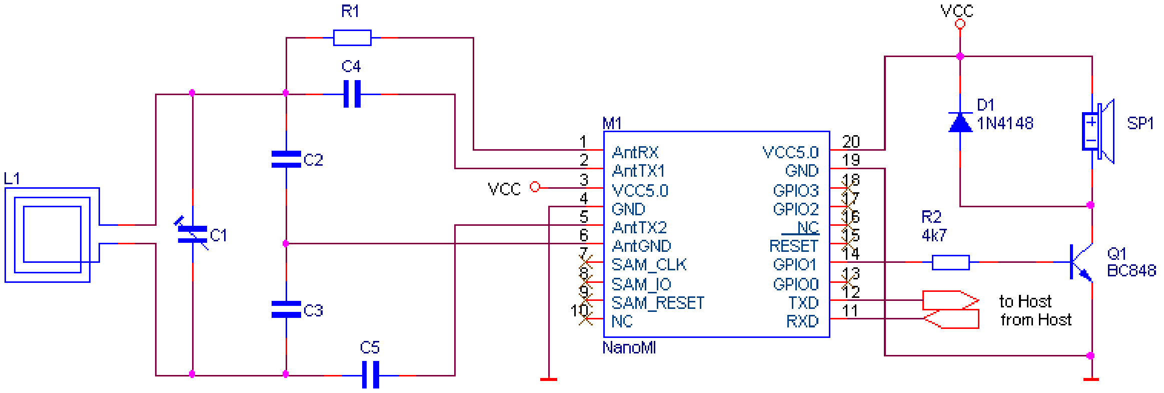
| Pin | Name | Description |
|---|---|---|
| 1 | AntRX | Antenna receiver input |
| 2 | AntTX1 | Antenna transmitter output 1 |
| 3, 20 | VCC | 5V |
| 4, 6, 19 | GND | Ground |
| 5 | AntTX2 | Antenna transmitter output 2 |
| 7 | SAM_CLK | SAM clock |
| 8 | SAM_IO | Bidirectional SAM I/O line |
| 9 | SAM_RESET | SAM reset |
| 10 | NC | Not connected |
| 11 | RXD | RS-232 receiver input |
| 12 | TXD | RS-232 transmitter output |
| 13 | GPIO0 | General purpose input / output 0 |
| 14 | GPIO1 / Beeper | General purpose input / output 1, connection port for beeper |
| 15 | $\overline{\text{RESET}}$ | Asynchronous reset |
| 16 | NC | Not connected |
| 17 | GPIO2 | General purpose input / output 2 |
| 18 | GPIO3 | General purpose input / output 3 |

## *2.3 Electrical Characteristics*

| | | |
|---|---|---|
| Frequency | 13.56MHz | |
| Power supply | 4.5V - 5.5V DC | |
| Current consumption | RF field off: | 20mA |
| | RF field on: | Typically 120mA, depending on antenna |

## *2.4 External Connections*

The schematic below shows a typical application circuit for operating the reader module.



### 2.4.1 Antenna

Nano Module Multi ISO requires an external antenna circuit for reading transponders. The part values of C1 – C5 and R1 are strongly depending on the used antenna inductor. Take the guide below as a reference for creating your own PCB antenna.

<u>Little PCB Antenna Design Guide:</u>

- Antenna inductance value is usually in the range of 1µH.
- For antenna loop up to 40mm diameter use 5 turns.
- For antenna loop larger than 40mm diameter use 3 turns.
- Use between 1mm to 2mm track width.
- Middle tap improves EMC compliance, but can be omitted for simple designs.
- All capacitors have to be chosen to create maximum field strength (measure antenna field with an oscilloscope and the voltage induced in some auxiliary loop; alternatively measure the module power consumption which is maximum at optimum tuning).
- C1 is used for fine tuning. Some extra capacitors may be connected in parallel to achieve best performance.
- Typical capacitor values: 10pF … 470pF. Prefer NP0/C0G ceramic capacitors.
- Value of R1 is usually 4.7kΩ, but this value may differ from design to design.

For further information regarding antenna design please contact NXP in order to receive the appropriate Application notes (Application Note Micore Reader IC Family; Directly Matched Antenna Design)

### 2.4.2 Asynchronous Serial Connection (UART)

Because Nano Module Multi ISO is transmitting and receiving TTL levels, it can be directly connected to a microcontroller. If you plan to run Nano Module Multi ISO at a PC, an appropriate interface converter circuit must be connected.

By default, the reader starts communication at 9600 baud, 8 databits, 1 stopbit, no parity. The baudrate can be increased up to 115200 baud in order to achieve a higher communication speed.

### 2.4.3 Usage of GPIOs

Nano Module Multi ISO provides four general purpose I/Os that can be configured individually. These I/Os can be read and written via commands.

Please consider, that the GPIOs have limited current source and sink capability of max. 25mA. An overcharge of GPIO-pins can damage the module!

A non self-oscillating beeper can also be connected to GPIO1. The tone length and frequency can be controlled by commands.

### 2.4.4 Asynchronous Reset

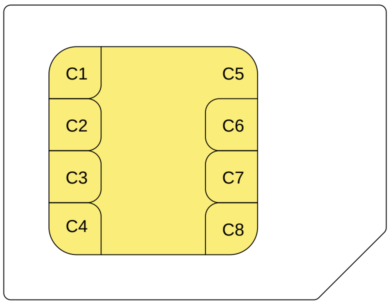
In usual operating environments, the reset pin can be left floating. For an asynchronous hardware reset pull the reset pin to a logic low level. Nano Module Multi ISO continues operation as soon as the reset pin is released.

### 2.4.5 Power supply

A power supply of 4.5V – 5.5V must be applied between the VCC and GND pins. Please make sure that the power supply provides a current capability of at least 120mA.

### 2.4.6 SAM

The reader supports connection of ISO7816 compatible SAM cards. The lines SAM_IO, SAM_CLK, SAM_RST, VCC and GND have to be connected as follows:

| SAM card | Pad | Line |
|---|---|---|
|  | C1 | VCC |
| | C2 | SAM_RST |
| | C3 | SAM_CLK |
| | C4 | Leave open |
| | C5 | GND |
| | C6 | Leave open |
| | C7 | SAM_IO |
| | C8 | Leave open |

# 3. Supported transponders

| ISO14443A | ISO14443B | ISO15693 |
|---|---|---|
| Mifare<br>- Classic 1k, 4k<br>- DESFire EV1<br>- Light<br>- Mini<br>- Plus S, X<br>- Pro X<br>- SmartMX<br>- Ultralight<br>- Ultralight C<br>Legic Advant[1]<br>SLE44R35<br>SLE66Rxx<br>my-d move | Calypso<br>CEPAS<br>Moneo<br>SRI512, SRT512, SRI4K, SRIX4K<br>Inside Contactless PicoPass[1]<br>HID ICLASS[1] | ICODE SLI<br>Tag-it<br>Legic Advant[1]<br>Inside Contactless PicoPass[1]<br>HID ICLASS[1]<br>my-d<br>EM4033<br>EM4233<br>EM4135<br>ST 24LR16<br>ST 24LR64<br>Fujitsu MB89R118/119 |

[1] UID only

# 4. Setting up a Terminal Program

In order to establish a connection between the reader and a terminal program, the following steps have to be done:

- Connect your reader to a PC
- Start your preferred terminal program (for Windows e.g. HyperTerminal)
- Select the serial port (e.g. COM1), where you connected the reader.
- Set up the connection speed and format: 9600 baud (default), 8 data bits, no parity, 1 stop bit
- Select no software handshake and no hardware handshake

# 5. Register Set

The reader has several system flags customizing its behavior. The flags are stored non-volatile in its EEPROM. It is recommended to keep all bits labeled RFU at their default state to guarantee further compatibility.

Please consider that if any register is written, the reader needs a reset so that the changes may take effect.

## *5.1 EEPROM Memory Organization*

| Register | Access | Default value | Description |
|---|---|---|---|
| 00h … 04h | RO | XXh | Unique device ID |
| 05h … 09h | RO | 00h | Internal use |
| 0Ah | R/W | 01h | Station ID |
| 0Bh | R/W | 41h | Protocol Configuration Register 1 |
| 0Ch | R/W | 00h | Baudrate Control Register |
| 0Dh | R/W | 20h | RFU, never change this register |
| 0Eh | R/W | 7Fh | Operation mode Register 1 |
| 0Fh | R/W | 0Ah | RFU, never change this register |
| 10h | R/W | 00h | Internal use |
| 11h | R/W | 00h | Internal use |
| 12h | R/W | 00h | Internal use |
| 13h | R/W | 00h | Protocol Configuration Register 2 |
| 14h | R/W | 0Ah | Reset Off Time |
| 15h | R/W | 25h | Reset Recovery Time |
| 16h | R/W | 00h | Application Family Identifier AFI |
| 17h | R/W | 10h | Internal use |
| 18h | R/W | 50h | Internal use |
| 19h | R/W | 10h | Internal use |
| 1Ah | R/W | 20h | Internal use |
| 1Bh | R/W | 00h | Protocol Configuration Register 3 |
| 1Ch … 1Fh | R/W | 00h | Internal use |
| 20h … DFh | R/W | FFh | RFU, never change this register |
| E0h … EFh | R/W | 00h | Installation Identifier |
| F0h … FDh | R/W | FFh | RFU, never change this register |
| FEh | R/W | 01h | Operation Mode Register 2 |
| FFh | R/W | 22h | Beep tone and length (Nano Module Multi ISO only) |

## *5.2 Unique Device ID (00h … 04h)*

The Unique Device ID is factory programmed and cannot be changed. It forms a unique 40 bit ID of the reader.

## *5.3 Station ID (0Ah)*

In ASCII mode, the Station ID has principally no influence on the functional behavior of the reader. Nevertheless it can be used to identify a reader in a multi-reader environment.

In Binary mode, the Station ID is used to address the reader in protocol header. The Station ID has the range 01h to FEh and can be set freely. The value 00h is reserved for the bus-master, all readers send their response to this device. The broadcast message (FFh) forces all readers to response to the command.

Default value: 01h

## *5.4 Protocol Configuration Register 1 (0Bh)*

The Protocol Configuration Register 1 specifies the general behavior of the reader.

| **Protocol Configuration Register 1** | | | | | | | |
|---|---|---|---|---|---|---|---|
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| ExtendID | RFU | RFU | RFU | NewSerial Mode | MultiTag | Binary | AutoStart |

### 5.4.1 AutoStart

If set the reader will start up in continuous read mode.

Default value: 1

### 5.4.2 Binary

If set the reader uses binary protocol. Refer to binary protocol description for further information.

Default value: 0

### 5.4.3 MultiTag

The MultiTag flag will enable multi tag recognition in continuous read mode. All tags are detected and displayed. Due to the more complex search algorithm detection speed is decreased in continuous read mode.

Default value: 0

### 5.4.4 NewSerialMode

The NewSerialMode flag controls the addition of a leading character to the serial number of transponders. This eases the recognition of transponder types.

This setting affects the commands continuous reading (‘c’), single tag select (‘s’) and multi tag select (‘m’)

Default value: 0

| Character | Transponder type |
|---|---|
| M | ISO14443A, e.g. Mifare® |
| P | PicoPass / ICLASS |
| S | ISO14443B, e.g. SRIX4K |
| V | ISO15693, e.g. I•Code SLI |

### 5.4.5 ExtendID

If set, the unique serial number (UID) of the transponder is extended by a single prefix byte. This bit has only effect for ISO14443A transponders.

This setting affects the commands continuous reading (‘c’), single tag select (‘s’) and multi tag select (‘m’)

Default value: 0

Possible values for the prefix byte are:

| Prefix | Description |
|---|---|
| 01h | Cascade Level 1 transponder, e.g. Mifare® Classic |
| 02h | Cascade Level 2 transponder, e.g. Mifare® Ultralight |
| 03h | Cascade Level 3 transponder |

## *5.5 Baudrate Control Register (0Ch)*

The Baudrate control register defines the start-up baudrate of the reader. For baudrate values, refer to the tables below:

| Baudrate Control Register | | | | | | | |
|---|---|---|---|---|---|---|---|
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| RFU | RFU | RFU | RFU | RFU | BS2 | BS1 | BS0 |

| BS2 | BS1 | BS0 | Resulting Baudrate |
|---|---|---|---|
| 0 | 0 | 0 | 9600 bps (default) |
| 0 | 0 | 1 | 19200 bps |
| 0 | 1 | 0 | 38400 bps |
| 0 | 1 | 1 | 57600 bps |
| 1 | 0 | 0 | 115200 bps |
| 1 | 0 | 1 | reserved |
| 1 | 1 | 0 | reserved |
| 1 | 1 | 1 | reserved |

### 5.5.1 Resetting the Baudrate to Default

The reader has a built-in mechanism to reset the baudrate to the default value of 9600 bps. This is useful, if no communication is possible any more due to an errant setting of the Baudrate control register. The reset can be done by sending a byte sequence at 9600 bps to the reader. If the sequence matches, the reader turns its baudrate setting to 9600 bps. This configuration remains valid until a subsequent reset or power-down event occurs. The user may now write the desired baudrate setting to the Baudrate Control Register. After that, a reset must be executed, so that the new settings may take effect.

In order to reset the baudrate, the host has to send the value FFh at 9600 bps three times to the reader. The bytes must have a temporal distance of at least 100ms. If a valid sequence is recognized, the reader switches to 9600 bps and sends an acknowledge to the host (‘B<CR><LF>’) using the ASCII protocol.

## *5.6 Operation Mode Register 1(0Eh)*

The Operation Mode Register 1 defines which tag types the reader shall support. Including or excluding of transponder types has direct influence in the transponder detection speed of the reader.

| **Operation Mode Register 1** | | | | | | | |
|---|---|---|---|---|---|---|---|
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| RFU | RFU | RFU | ISO15693 | RFU | SRX | RFU | ISO14443A |

## *5.7 Protocol Configuration Register 2 (13h)*

The Protocol Configuration Register 2 specifies the further general behavior of the reader.

| **Protocol Configuration Register 2** | | | | | | | |
|---|---|---|---|---|---|---|---|
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| RFU | RFU | Reset Recovery Time Multiplier | | Noisy Environment | RFU | Disable Startup Message | RFU |

### 5.7.1 Disable Startup Message

If set, the reader will not print the start-up message (`MultiISO 1.17`) when powered or if a reset occurs.

Default value: 0

### 5.7.2 Noisy Environment

If set, the reader will only quit continuous read mode if the ‘`.`’ character is received.

Default value: 0

### 5.7.3 Reset Recovery Time Multiplier

Multiplies the value of the Reset Recovery Time:

Default value: 00

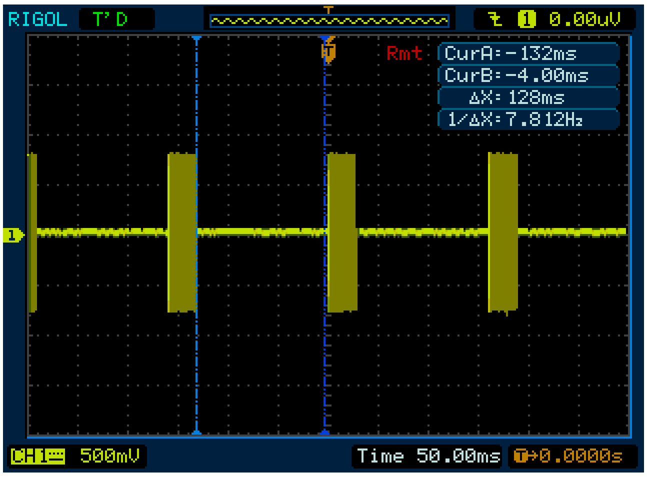
| Reset Recovery Time Multiplier[1:0] | Reset Recovery Time |
|---|---|
| 00 | 1x |
| 01 | 2x |
| 10 | 3x |
| 11 | 4x |

## 5.8 Reset Off Time (14h)

The Reset Off Time register represents the time in milliseconds, the RF-field is switched off, after a reading attempt has finished. This register is used for the continuous read mode (‘c’).
The higher the value of the register, the more energy can be saved. As soon as there is a transponder in range, the field turned on permanently. Keep in mind that increased saving of energy results in decreased detection speeds.

Default value: 0Ah

The image below shows the activity of the RF-field. The Reset Off Time register has been set to 80h, this results in a pause of 128ms:

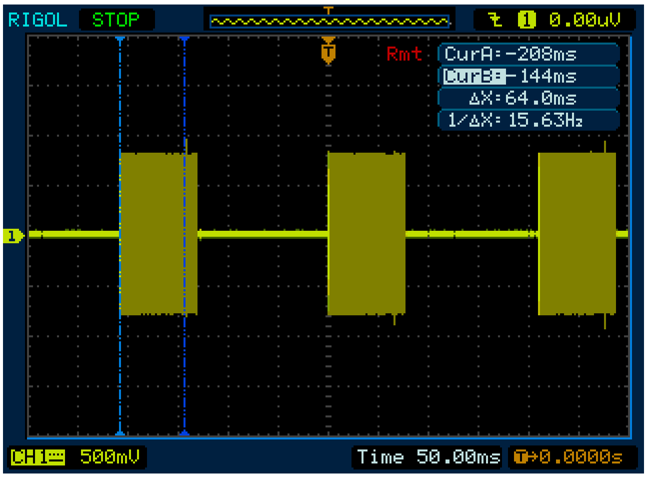


## 5.9 Reset Recovery Time (15h)

The Reset Recovery Time register represents the recovery time in milliseconds after the RF-field is turned on. This register is used for the continuous read mode (‘c’), single tag select (‘s’) and multi tag select (‘m’) commands.
The value of the register determines the time the reader waits before any reading attempt. Keep in mind that a higher value results in increased energy consumption.

Default value: 25h

The image below shows the activity of the RF-field. The Reset Recovery Time register has been set to 40h (64 ms):



## *5.10 Application Family Identifier AFI (16h)*

The Application Family Identifier (AFI) is only supported by ISO15693 transponders. If the value of this register is different from 00h, the AFI is used. This means, that only transponders containing an identical AFI will answer to the reader.

Default value: 00h

## *5.11 Protocol Configuration Register 3 (1Bh)*

The Protocol Configuration register 3 specifies the further general behavior of the reader.

| **Protocol Configuration register 3** | | | | | | | |
|---|---|---|---|---|---|---|---|
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| TagInfo | ReqA Extended ID | Internal use / do not change | | | RFU | RFU | RFU |

### 5.11.1 ReqA Extended ID

If set, the reader will replace the cascade level information (1 byte) by the ReqA response (2 bytes). This bit has only effect in combination with ISO14443A transponders.

Default value: 0

### 5.11.2 TagInfo

If set, the reader extends the UID by a single byte. The value of the byte gives information about the tag type.

Default value: 0

Refer to the table below for tag types:

| Byte value | Corresponding transponder type |
|---|---|
| 00 | Mifare® Mini |
| 01 | Mifare® Classic 1K |
| 02 | Mifare® Classic 4K |
| 03 | Mifare® Plus S |
| 04 | Mifare® Plus X |
| 05 | Mifare® Ultralight / Ultralight C |
| 06 | Mifare® DESFire |
| 0F | Unknown ISO14443A transponder |
| 10 | I•Code SL2 |
| 11 | I•Code SL2-S |
| 12 | Tag-it™ HF-I Plus Inlay |
| 13 | Tag-it™ HF-I Plus Chip |
| 14 | Tag-it™ HF-I Standard |
| 15 | Tag-it™ HF-I Pro |
| 16 | ST 24LR16 Dual Interface EEPROM |
| 17 | ST 24LR64 Dual Interface EEPROM |
| 18 | Fujitsu MB89R118 |
| 19 | Fujitsu MB89R119 |
| 1F | Unknown ISO15693 transponder |
| 30 | Picopass / ICLASS transponder |
| 40 | SRX transponder |
| 80 | ISO14443B transponder |
| FF | Unknown transponder |

## 5.12 Installation Identifier (E0h … EFh)

The Installation Identifier is used for Mifare® PLUS in virtual card environment. The Installation Identifier must match with the one stored in the Mifare® PLUS transponder that shall be identified using Virtual Card Select command.

## 5.13 Operation Mode Register 2 (FEh)

The Operation Mode register 2 defines which tag types the reader shall support. Including or excluding transponder types has direct influence in the transponder detection speed of the reader.

| Operation Mode Register 2 | | | | | | | |
|---|---|---|---|---|---|---|---|
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| RFU | RFU | RFU | RFU | RFU | RFU | RFU | PicoPass / ICLASS |

## *5.14 Beep Tone and Length (FFh)*

This register defines the beep tone (frequency) and beep length. It is only implemented on Nano Module Multi ISO.

| **Beep Tone and Length Register** | | | | | | | |
|---|---|---|---|---|---|---|---|
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Length | | | | Tone | | | |

The beep tone code selects a predefined frequency from the table below:

| Tone[3:0] | Frequency | Standard pitch (Vienna scale) |
|---|---|---|
| `0000` | 523Hz | C |
| `0001` | 554Hz | Cis |
| `0010` | 587Hz | D |
| `0011` | 622Hz | Dis |
| `0100` | 659Hz | E |
| `0101` | 699Hz | F |
| `0110` | 739Hz | Fis |
| `0111` | 784Hz | G |
| `1000` | 831Hz | Gis |
| `1001` | 880Hz | A |
| `1010` | 932Hz | Ais |
| `1011` | 988Hz | H |
| `1100` | Reserved | |
| `1101` | Reserved | |
| `1110` | Reserved | |
| `1111` | Reserved | |

The following formula defines the beep length:

BeepLength = (Length[3:0] + 1) x 62.5ms

Examples:

| Length[3:0] | BeepLength |
|---|---|
| `0000` | 62.5ms |
| `0001` | 125ms |
| `0111` | 500ms |
| `1111` | 1000ms |

# 6. Communication Protocol

## 6.1 ASCII Protocol

The ASCII protocol has been designed for easy handling. Data is always transmitted in hexadecimal notation, i.e. `5E`.

Every time the reader is powered up, a startup message is displayed. On the terminal screen this should look like this:

```
MultiISO 1.17<CR><LF>
```

The reader is now ready for reception of commands. By default, the reader starts in continuous reading mode, this means the reader is scanning for transponders and prints the present UIDs.

Please note that pseudo-tetrade values always must be submitted in capital letters, e.g. `1234ABCD`

Commands must be submitted in lower-case letters, e.g. `wp0C03`

## 6.2 Binary Protocol

The binary protocol has been designed for industrial applications with synchronization and frame checking. The reader uses a watchdog timer internally to ensure correct framing. A binary frame is built up as follows:

| STX | Station ID | Length | Data | BCC | ETX |
|---|---|---|---|---|---|
| 1 byte | 1 byte | 1 byte | Various length | 1 byte | 1 byte |

In binary mode, the reader only gives a response if a command is issued. This means, the reader does not show the start-up string and also continuous read does not work.

### 6.2.1 STX

Start of transmission (02h)

### 6.2.2 Station ID

| | |
|---|---|
| 00h: | Reserved for the bus master. Responses are sent with Station ID set to 00h. |
| FFh: | Broadcast message. All devices will execute the command and send their response. |

### 6.2.3 Length

Denotes the length of the data block in bytes.

### 6.2.4 Data

The data block contains the command including its arguments. The command values are the same as in ASCII protocol mode whereas the arguments are transmitted in binary. If a command requires a subsequent carriage return (0Dh) it also must be submitted.

### 6.2.5 Block Check Character (BCC)

The BCC is used to detect transmission errors. The BCC is calculated by XOR-ing each byte of the transmission frame except the STX/ETX characters.

### 6.2.6 ETX

End of Transmission (03h)

## 6.2.7 Example

This example shows how to log into sector 0 of a Mifare® card, using transport key FFh:

| STX | Station ID | Length | Data | BCC | ETX |
|---|---|---|---|---|---|
| 02 | 01 | 04 | 6C 00 FF 0D | 9B | 03 |

## 6.2.8 Remarks

If an invalid instruction frame is received (e.g. BCC is wrong) or if the requested Station ID does not match the internal ID of the reader, the command is not executed.

# 7. Instruction Set

## *7.1 Command Overview*

### 7.1.1 Common Commands

| Command | Description | For details view chapter |
|---|---|---|
| **‘ ’** | Cancel command | 7.6.6 |
| **‘as’** | AES data encryption/decryption | 7.6.15 |
| **‘b’** | Beep[1] | 7.6.11 |
| **‘c’** | Continuous read mode | 7.3.1 |
| **‘.’** | Stop continuous read mode (in noisy environment) | 7.6.6 |
| **‘ds’** | DES / Triple-DES data encryption/decryption | 7.6.16 |
| **‘ei’** | Init SAM card | 7.5.9.1 |
| **‘et’** | SAM card transmit data | 7.5.9.2 |
| **‘g’** | Get station ID | 7.6.1 |
| **‘ir’/’iw’** | Read / write GPIO0…3[1] | 7.6.7 and 7.6.8 |
| **‘m’** | MultiTag select / tag list | 7.3.3 |
| **‘of’** | Set configuration flags | 7.5.3 |
| **‘og’** | Set configuration registers | 7.5.4 |
| **‘o+a’ / ‘o-a’** | Include / Exclude tag type ISO14443A | 7.5.6 and 7.5.7 |
| **‘o+b’ / ‘o-b’** | Include / Exclude tag type ISO14443B | 7.5.6 and 7.5.7 |
| **‘o+p’ / ‘o-p’** | Include / Exclude tag type PicoPass / ICLASS | 7.5.6 and 7.5.7 |
| **‘o+s’ / ‘o-s’** | Include / Exclude tag type SRX | 7.5.6 and 7.5.7 |
| **‘o+v’ / ‘o-v’** | Include / Exclude tag type ISO15693 | 7.5.6 and 7.5.7 |
| **‘oa’** | Set tag type ISO14443A | 7.5.5 |
| **‘ob’** | Set tag type ISO14443B | 7.5.5 |
| **‘op’** | Set tag type PicoPass / ICLASS | 7.5.5 |
| **‘os’** | Set tag type SRX | 7.5.5 |
| **‘ot’** | Search for all supported tag types | 7.5.5 |
| **‘ov’** | Set tag type ISO15693 | 7.5.5 |
| **‘poff’** | Antenna power off | 7.6.2 |
| **‘pon’** | Antenna power on | 7.6.3 |
| **‘pr’/’pw’** | Read / write GPIO1[1] | 7.6.9 and 7.6.10 |
| **‘r’ / ‘rb’ /’rc’** | Read data block | 7.4.2 |
| **‘ra’** | Resend last answer | 7.6.12 |
| **‘rp’** | Read EEPROM register | 7.5.1 |
| **‘s’** | Select single tag | 7.3.2 |
| **‘t’** | Transparent ISO14443-4 transponder access | 7.4.6 |
| **‘v’** | Get version | 7.6.4 |
| **‘w’ / ‘wb’ /’wc’** | Write data block | 7.4.3 |
| **‘wd’** | Write AES / Triple DES key | 7.5.8.2 |
| **‘wp’** | Write EEPROM register | 7.5.2 |
| **‘x’** | Reset | 7.6.5 |
| **‘y’** | Antenna field reset | 7.6.13 |

---

[1] No function on TWN3

## 7.1.2 Mifare® Classic Specific Commands

| Command | Description | For details view chapter |
|---|---|---|
| **'+'** | Increment value block | 7.4.4.3 |
| **'-'** | Decrement value block | 7.4.4.4 |
| **'='** | Copy value block | 7.4.4.5 |
| **'l'** | Login (authenticate tag) | 7.4.1.1 and 7.4.1.2 |
| **'rv'** | Read value block | 7.4.4.2 |
| **'wm'** | Write Mifare® Classic key | 7.5.8.1 |
| **'wv'** | Write value block | 7.4.4.1 |

### *7.1.2.1 Mifare® PLUS Specific Commands*

| Command | Description | For details view chapter |
|---|---|---|
| **'ncp'** | Commit Personalization | 7.4.5.2 |
| **'nl'** | Perform Mifare® PLUS authentication | 7.4.1.3 |
| **'nvcs'** | Virtual Card Select | 7.3.4 |
| **'nw'** | Write configuration data | 7.4.5.3 |
| **'nwp'** | Write Personalization Data | 7.4.5.1 |
| **'npc'** | Proximity Check | 7.4.5.4 |

## 7.1.3 ISO15693 specific commands

| Command | Description | For details view chapter |
|---|---|---|
| **'k'** | Lock block | 7.6.14 |

## *7.2 Error Codes*

| Error Code | Description |
|---|---|
| **'?<CR><LF>'** | Unknown command |
| **'E<CR><LF>'** | Invalid key format |
| **'F<CR><LF>'** | General failure |
| **'I<CR><LF>'** | Invalid value block (block does not match the value format) |
| **'N<CR><LF>'** | No tag in the field / SAM card not found |
| **'O<CR><LF>'** | Operation mode failure |
| **'R<CR><LF>'** | Parameter out of range |
| **'X<CR><LF>'** | Block already locked / Error during value operation |

# *7.3 Transponder Serial Number Related Commands*

## 7.3.1 Continuous read mode ‘c’

The reader reads and displays serial numbers continuously while one or more tags remain in the field. This command stops as soon as any character is sent to the reader if the Noisy Environment bit is cleared.

The reader supports different tag types at the same time. In order to increase the reading performance, switch to single tag mode. If more than one tag shall be detected at the same time, the MultiTag flag must be activated. The response data length mainly depends on the tag type.

**Command: `‘c’`**

**Answer**

| Answer | Description |
|---|---|
| `Data<CR><LF>` | Serial number (UID, n bytes) |

## 7.3.2 Select Single Tag ‘s’

This command selects a single tag in the antenna field. It shall only be used in single tag environments (use the ‘m’ command in multiple tag environments). In case of success the command returns the UID of the selected card. The length of the UID is detected automatically.

As soon as a transponder has been selected, it is ready for further data transactions, e.g. authentication, read block or write block.

**Command: `‘s’`**

**Answer**

| Answer | Description |
|---|---|
| `Data<CR><LF>` | Serial number (UID, n bytes) |
| `‘N<CR><LF>’` | Error: No tag in the field |

## 7.3.3 MultiTag Selection / Tag List ‘m’

This command detects several tags at the same time. It replaces the fast select command (‘s’) in multiple tag surroundings. The MultiTag list command lists all present tags with their serial numbers. Use the MultiTag select command to select a single tag. Each tag has to be selected separately.

After selection, the transponder is ready for further data transactions, e.g. authentication, read block or write block.

Keep in mind that each transponder consumes its individual amount of energy, provided by the reader. Due to the limited availability of emitted energy, the operating distance is decreased, the more transponders are present. Principally the operating distance depends on the used transponders, the total amount of transponders and the ambient conditions, e.g. if metal is surrounding the reader.

**Command:** **`‘m<CR>’ / ‘m[UID]<CR>’`**

| Command | Description |
|---|---|
| `‘m<CR>’` | List all present tags |
| `‘m[UID]<CR>’` | Select tag by its UID |

**Answer**

| Answer | Description |
|---|---|
| `Data<CR><LF>` | Serial number (UID, n bytes) |
| `‘N<CR><LF>’` | Error: No tag in the field |

**Examples**

| Command | Answer | Description |
|---|---|---|
| `‘m<CR>’` | `044A3A11FC1E80`<br>`56AB3798`<br>`02` | First tag<br>Second tag<br>Number of detected tags |
| `‘m56AB3798000000<CR>’` | `56AB3798` | Second tag selected |

**Note:**

The MultiTag selection command always requires a 7 byte UID in case of a Mifare® card. If the UID of the desired transponder possesses only 4 bytes (e.g. Mifare® Classic), the passed UID must be filled up with zeros – see the example above.

### 7.3.4 Mifare® PLUS Virtual Card Select ‘nvcs’

Use this command to retrieve the ‘real’ UID of the transponder in a virtual card environment. If a card is configured to show random IDs, the virtual card select command offers the possibility to retrieve the UID in a fast and secure way. In order to get the UID, the 16 bytes Installation Identifiers of the card and the reader must match. Furthermore, two AES keys are required. These are the Virtual Card Polling Encryption key and the Virtual Card Polling MAC key. Please note, that only EEPROM keys can be used.

**Command:** `‘nvcs[VCPEKey][VCPMACKey]’`

| Parameters | Description |
|---|---|
| `[VCPEKey]` | Virtual Card Polling Encryption EEPROM AES key (1 byte) |
| `[VCPMACKey]` | Virtual Card Polling MAC EEPROM AES key (1 byte) |

**Answer**

| Answer | Description |
|---|---|
| `UID<CR><LF>` | Real UID of the transponder |
| `‘F<CR><LF>’` | Error: Proximity check failed |
| `‘R<CR><LF>’` | Error: Parameter out of range |

**Example**

| Command | Description |
|---|---|
| `‘nvcs5051<CR>’` | Perform virtual card select, using EEPROM AES keys 00 and 01 |

# 7.4 Data Transaction Related Commands

## 7.4.1 Authenticate Tag

### 7.4.1.1 Mifare® Classic Login 'l'

This command performs an authentication into a specific sector of a Mifare® Classic transponder. Only one transponder and only one sector can be accessed at the same time. Prior access, the transponder must be selected by either single tag or MultiTag selection commands.

For authentication into a sector, the matching login key is needed. The key may either be entered by command, or can be stored in the readers' EEPROM. The reader is able to store up to 32 Mifare® Classic keys. Principally every stored key may act either as key A or key B, the selection is done via parameter list.

**Command:** `'l[Sector][KeyType][Key / <CR>]'`

| Parameters | Description |
|---|---|
| `[Sector]` | Sector number, valid range 00h – 3Fh |
| `[KeyType]` | AAh: authenticate with key type A<br>FFh: authenticate with key type A, transport key FFFFFFFFFFFFh<br>BBh: authenticate with key type B<br>10h … 2Fh: authenticate with stored key type A (00h … 1Fh)<br>30h … 4Fh: authenticate with stored key type B (00h … 1Fh) |
| `[Key / <CR>]` | Enter key manually (6 bytes) or tell the reader to login with a transport key by submitting a carriage return `<CR>` (1 byte) |

**Answer**

| Answer | Description |
|---|---|
| `'L<CR><LF>'` | Login success |
| `'F<CR><LF>'` | Error: general failure |
| `'N<CR><LF>'` | Error: no tag in the field or wrong key |

**Examples**

| Command | Description |
|---|---|
| `'l01AA<CR>'` | Authenticate into sector 01 using transport key type A A0A1A2A3A4A5h |
| `'l02BB<CR>'` | Authenticate into sector 02 using transport key type B B0B1B2B3B4B5h |
| `'l03FF<CR>'` | Authenticate into sector 03 using transport key type A FFFFFFFFFFFFh |
| `'l04AA1234567890AB'` | Authenticate into sector 04 using specified key type A 1234567890ABh |
| `'l0510'` | Authenticate into sector 05 using EEPROM key 0, key type A |
| `'l0637'` | Authenticate into sector 06 using EEPROM key 7, key type B |

### 7.4.1.2 Mifare® Ultralight C Login 'l'

This command performs an authentication into a Mifare® Ultralight C transponder. The authentication scheme into this transponder type is different to Mifare® Classic due to the fact, that Ultralight C uses Triple-DES cryptography.

Unlike Mifare® Classic the authentication concerns the entire transponder. This means, once an authentication succeeds, the complete transponder is accessible.

Prior login, the transponder must be selected by either single tag or MultiTag selection commands.

For authentication, the matching login key is needed. The key may either be entered by command, or can be stored in the readers' EEPROM. The reader is able to store up to 16 Triple-DES keys.

**Command:** `'l[00][KeyType][Key / <CR>]'`

| Parameters | Description |
|---|---|
| `[00]` | To maintain backward compatibility the first parameter shall be set to 00h |
| `[KeyType]` | CCh: denotes Triple-DES (Ultralight C) key<br>50h … 5Fh: authenticate with stored Triple-DES key (00h … 0Fh) |
| `[Key / <CR>]` | Enter key manually (16 bytes) or tell the reader to login with a transport key by submitting a carriage return **`<CR>`** (1 byte) |

**Answer**

| Answer | Description |
|---|---|
| `'L<CR><LF>'` | Login success |
| `'F<CR><LF>'` | Error: general failure |
| `'N<CR><LF>'` | Error: no tag in the field or the tag does not respond |

**Examples**

| Command | Description |
|---|---|
| `'l00CC<CR>'` | Authenticate into transponder using Triple-DES transport key:<br>K1 = 49454D4B41455242h<br>K2 = 214E4143554F5946h |
| `'l00CC000102030405060708090A0B0C0D0E0F'` | Authenticate into transponder using Triple-DES key:<br>K1 = 0001020304050607h<br>K2 = 08090A0B0C0D0E0Fh |
| `'l0055'` | Authenticate into transponder using Triple-DES EEPROM key 05h |

### *7.4.1.3 Mifare® PLUS Login 'nl'*

Mifare® PLUS offers various authentication features to the user. Depending on the desired transaction or configured security level of the transponder, the appropriate authentication procedure must be chosen by the user. The reader always demands an AES key for authentication; if the transponder is configured to security level 2, optionally a Crypto1 key can be passed to the reader. This enables the reader to compute the appropriate session key for further communication ciphered by the Mifare® Crypto1 algorithm. The Crypto1 login is performed automatically after AES authentication.

The login procedure is also used to switch a transponder to a higher security level. Furthermore, an optional AES-authentication can be done in SL1 using the SL1 card authentication key.

**Command:** `'nl[Sector / KeyNumber][KeyType][KeyAES][KeyCrypto1]'`

| Parameters | Description |
|---|---|
| `[Sector / KeyNumber]` | Sector- or Keynumber, valid range 00h – 2Ch.<br>00h … 27h: Parameter addresses a data sector<br>28h: Card master key<br>29h: Card configuration key<br>2Ah: Level 2 switch key<br>2Bh: Level 3 switch key<br>2Ch: SL1 card authentication key |
| `[KeyType]` | AAh: authenticate with key type A<br>BBh: authenticate with key type B |
| `[KeyAES]` | Enter key manually (16 bytes) or tell the reader to login with a stored EEPROM key (1 byte). EEPROM keys are range 50h … 5Fh |
| `[KeyCrypto1]` | Enter key manually (6 bytes), or tell the reader to generate the Crypto1 session key using either a stored EEPROM key or a transport key (1 byte). EEPROM keys are range 10h … 2Fh, Transport keys are indicated by AA/BB/FF |

**Answer**

| Answer | Description |
|---|---|
| `'L<CR><LF>'` | Login / Authentication / Security level switch success |
| `'F<CR><LF>'` | Error: general failure |
| `'N<CR><LF>'` | Error: no tag in the field or the tag does not respond |

## Examples

| Command | Description |
|---|---|
| **‘nl00AA50<CR>’** | Authenticate to sector 00 using EEPROM AES key 00, type A |
| **‘nl02BB51<CR>’** | Authenticate to sector 02 using EEPROM AES key 01, type B |
| **‘nl28AA52<CR>’** | Authenticate with card master key using EEPROM AES key 02, KeyType is ignored but has to be transmitted to the reader. |
| **‘nl2BAA53<CR>’** | Switch card to SL3 using EEPROM AES key 03, KeyType is ignored but has to be transmitted to the reader. |
| **‘nl03AA00112233445566778899AABBCCDDEEFF<CR>’** | Authenticate into sector 03 using specified AES key 00112233445566778899AABBCCDDEEFFh, key type A |
| **‘nl04AA00112233445566778899AABBCCDDEEFFAA<CR>’** | Authenticate into sector 04 using specified AES key 00112233445566778899AABBCCDDEEFFh, key type A, compute Crypto1 session key, key base is transport key A0A1A2A3A4A5h |
| **‘nl05BB00112233445566778899AABBCCDDEEFFBB<CR>’** | Authenticate into sector 05 using specified AES key 00112233445566778899AABBCCDDEEFFh, key type B, compute Crypto1 session key, key base is transport key B0B1B2B3B4B5h |
| **‘nl06AA00112233445566778899AABBCCDDEEFF16<CR>’** | Authenticate into sector 06 using specified AES key 00112233445566778899AABBCCDDEEFFh, key type A, compute Crypto1 session key, key base is EEPROM Crypto1 key 06 |
| **‘nl07AA54A0A1A2A3A4A5<CR>’** | Authenticate into sector 07 using EEPROM AES key 04, compute Crypto1 session key, key base is specified key A0A1A2A3A4A5h |
| **‘nl08AA55FF<CR>’** | Authenticate into sector 08 using EEPROM AES key 05, compute Crypto1 session key, key base is transport key FFFFFFFFFFFFh |
| **‘nl09AA5617<CR>’** | Authenticate into sector 09 using EEPROM AES key 06, compute Crypto1 session key, key base is EEPROM Crypto1 key 07 |
| **‘nl0AAA00112233445566778899AABBCCDDEEFFA0A1A2A3A4A5’** | Authenticate into sector 0A using specified AES key 00112233445566778899AABBCCDDEEFFh, compute Crypto1 session key, key base is specified key A0A1A2A3A4A5h. Note that this command is not followed by a <CR> |

### *7.4.1.4 Login in multiple Tag Surroundings*

In order to log into different tags, list all present tags first and then select the desired tag. After that, perform the login procedure.

## 7.4.2 Read Data Block ‘r’ / ‘rb’ /’rc’

This command reads an entire data block from a transponder. The range of valid block addresses depends on the used transponder. If you are working with Mifare® cards, keep in mind that any read- or write access requires a successful login into the respective sector. The length of returned data depends on the tag-type, e.g. a Mifare® Classic card returns 16 bytes of data, a SRX transponder returns 4 bytes per block. The ‘rc’-command shall be used to specify a 16 bit block address.

**Command:** `‘r[BlockAddr8]’ / ‘rb[BlockAddr8]’ / ‘rc’[BlockAddr16]`

**Answer**

| Answer | Description |
|---|---|
| `Data<CR><LF>` | Block data |
| `‘F<CR><LF>’` | Error: read failure |
| `‘N<CR><LF>’` | Error: no tag in the field, or the tag does not respond |
| `‘R<CR><LF>’` | Error: Block address out range; the block address of the ‘r’ command is higher than 40h, use the ‘rb’ command instead. |

**Example**

| Command | Description |
|---|---|
| `r08` | Reads block 08 (sector 02, block 00) |
| `rc07FF` | Reads block 7FF of a ST 24LR64 transponder |

### *7.4.2.1 Mifare® Ultralight / Ultralight C*

Though the page size of this transponder family is 4 bytes, the read command always returns 16 bytes. This is achieved by reading the three subsequent data pages, e.g. if page 04 is to be read, the reader also returns the content of page 05, 06 and 07.

### 7.4.3 Write Data Block ‘w’ / ‘wb’ / ‘wc’

This command writes an entire data block to the transponder. The range of valid block addresses depends on the used transponder. A read after write is done automatically to ensure data integrity. If you are working with Mifare® cards, keep in mind that any read- or write access requires a successful login into the respective sector. The length of required data to be written to the transponder depends on the tag-type, e.g. a Mifare® Standard block has 16 bytes, a SRX transponder has 4 bytes per block. The ‘wc’-command shall be used to specify a 16 bit block address.

**Command:** **`‘w[BlockAddr8]’ / ‘wb[BlockAddr8]’ / ‘wc’[BlockAddr16]`**

**Answer**

| Answer | Description |
|---|---|
| **`Data<CR><LF>`** | Block data |
| **`‘F<CR><LF>’`** | Error: write failure |
| **`‘N<CR><LF>’`** | Error: no tag in the field, or the tag does not respond |
| **`‘R<CR><LF>’`** | Error: Operation mode failure; the block address of the ‘w’ command is higher than 40h, use the ‘wb’ command instead. |

**Example**

| Command | Description |
|---|---|
| **`w08000102030405060708090A0B0C0D0E0F`** | Writes data **`000102030405060708090A0B0C0D0E0F`** to block 08 (sector 02, block 00) |
| **`w1A11223344`** | Writes data **`11223344`** to block 1Ah |
| **`wc07FFAABBCCDD`** | Writes data **`AABBCCDD`** to block 7FFh |

#### *7.4.3.1 Mifare® Ultralight / Ultralight C*

If data is written to a page of this transponder family, the reader performs the so-called “Compatibility-write” procedure. This means, the reader expects 16 bytes of data, but only the first 4 bytes are written to the specified data page; the remaining 12 bytes are ignored by the transponder.

The command executes an automatic read-after-write check, where all 16 bytes are included into the comparison. So the risk of getting a write failure is very high, if the content of subsequent memory-pages is unknown and the remaining 12 bytes are filled with dummy bytes! So it could be useful, to make a manual read-before-write, to construct the parameter list correctly in order not to get such errors.

**Example**

| Command | Description |
|---|---|
| **`r04`** | Reads pages 04 … 07, content:<br>**`000000000405060708090A0B0C0D0E0F`** |
| **`w04000102030405060708090A0B0C0D0E0F`** | Writes data **`00010203`** to page 04 |

## 7.4.4 Mifare® Value Block Related Commands

A sector of a Mifare® transponder may contain so-called ‘value blocks’. A value block is a usual data block, where the information is stored in a certain format. Special Mifare® commands like increment or decrement may be applied to such a block, e.g. for electronic purse functionality. Value block commands only work in combination with Mifare® Classic and Mifare® PLUS transponders. Please note, that only Mifare® PLUS X transponders support value block commands in SL3.

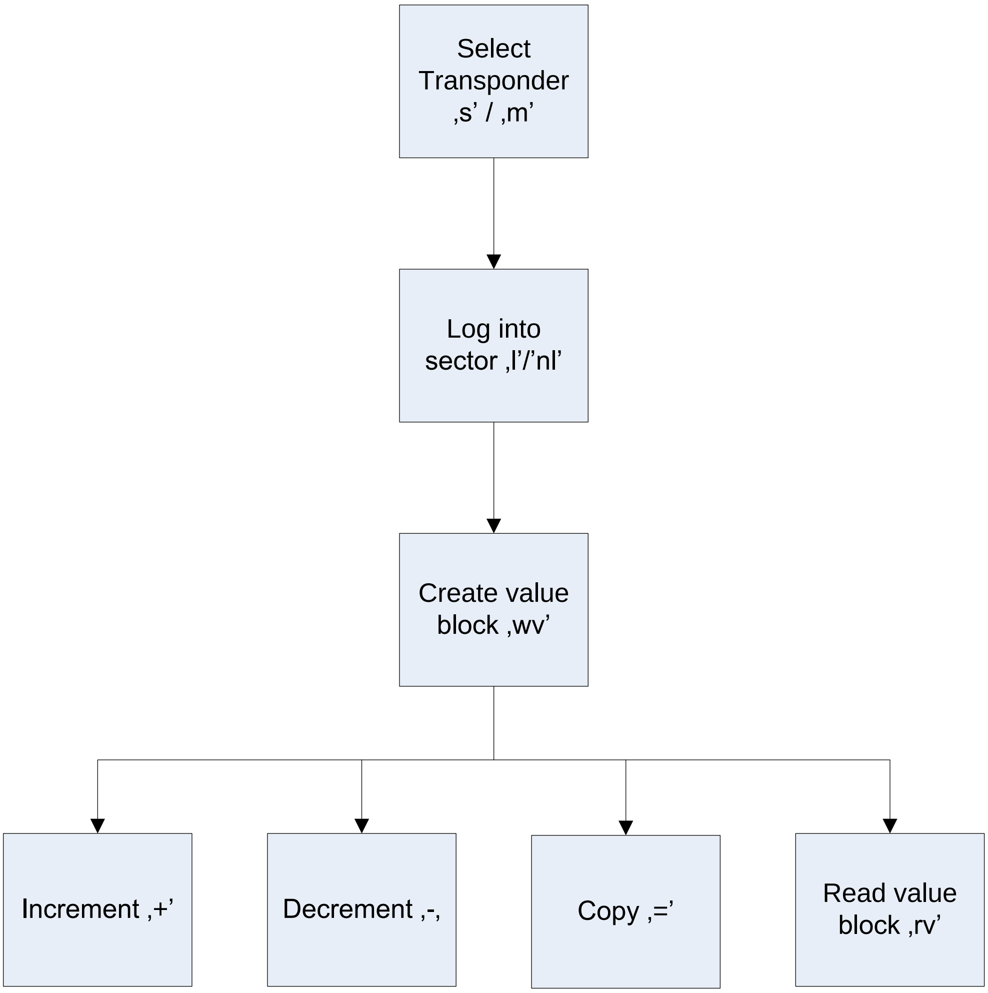
A value block contains 4 bytes user data, the remaining 12 bytes are processed internally by the Mifare® transponder for increased data integrity.

A value block is formatted as follows:

| Byte | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Data | Value | | | | $\overline{\text{Value}}$ | | | | Value | | | | A | $\overline{A}$ | A | $\overline{A}$ |

The value data is stored three times: twice non-inverted and once inverted. The lowest significant byte is stored in the lowest address byte. The last four bytes are for internal use and shall not be altered.

The following diagram shows the typical command-flow in order to work with Mifare® value blocks:



### *7.4.4.1 Create Value Block ‘wv’*

Use this command to format a common data block as a value block. The range of valid block addresses depends on the used transponder type. You must log into the respective sector before this command can be executed.

**Command:** **`'wv[BlockAddr][Value]'`**

| Parameters | Description |
|---|---|
| `[BlockAddr]` | Block address |
| `[Value]` | Initial value stored on the value block (4 bytes).<br>The value is stored signed (most significant bit). Negative values are stored in 2’s complement format. |

**Answer**

| Answer | Description |
|---|---|
| `Data<CR><LF>` | Written value |
| `'F<CR><LF>'` | Error: write failure |
| `'I<CR><LF>'` | Error: invalid block format |

**Examples**

| Command | Description |
|---|---|
| `wv0800000000` | Formats block 08 as a value block. Initial value: 00000000h |
| `wv0912345678` | Formats block 09 as a value block. Initial value: 12345678h |
| `wv0AFFFFFFFE` | Formats block 0A as a value block. Initial value: FFFFFFFEh (-2) |

### *7.4.4.2 Read Value Block ‘rv’*

Use this command to read a value block. This command checks if data of the specified block is stored in value format. You must log into the respective sector before this command can be executed.

**Command:** **`'rv[BlockAddr]'`**

| Parameters | Description |
|---|---|
| `[BlockAddr]` | Block address |

**Answer**

| Answer | Description |
|---|---|
| `Data<CR><LF>` | Value of block |
| `'F<CR><LF>'` | Error: read failure |
| `'I<CR><LF>'` | Error: invalid block format |

**Example**

| Command | Description |
|---|---|
| `rv08` | Reads value block 08 |

### *7.4.4.3 Increment Value Block '+'*

Use this command to increment a value block by a defined value. A read after increment is performed automatically. The command fails if the specified block is not formatted as value block. You must log into the respective sector before this command can be executed.

**Command:** `'+[BlockAddr][Value]'`

| Parameters | Description |
|---|---|
| `[BlockAddr]` | Block address (1 byte) |
| `[Value]` | Value to be added (4 bytes) |

**Answer**

| Answer | Description |
|---|---|
| `Data<CR><LF>` | Value of block |
| `'F<CR><LF>'` | Error: read failure |
| `'I<CR><LF>'` | Error: invalid block format |
| `'X<CR><LF>'` | Error: Bad value operation, e.g. sign overflow |

**Examples**

| Command | Description |
|---|---|
| `+0800000001` | Increments block 08 by 00000001h |
| `+0812345678` | Increments block 08 by 12345678h |

### *7.4.4.4 Decrement Value Block '-'*

Use this command to decrement a value block by a defined value. A read after decrement is performed automatically. The command fails if the specified block is not formatted as value block. You must log into the respective sector before this command can be executed.

**Command:** `'-[BlockAddr][Value]'`

| Parameters | Description |
|---|---|
| `[BlockAddr]` | Block address (1 byte) |
| `[Value]` | Value to be subtracted (4 bytes) |

**Answer**

| Answer | Description |
|---|---|
| `Data<CR><LF>` | Value of block |
| `'F<CR><LF>'` | Error: read failure |
| `'I<CR><LF>'` | Error: invalid block format |
| `'X<CR><LF>'` | Error: Bad value operation, e.g. sign overflow |

**Examples**

| Command | Description |
|---|---|
| `-0800000001` | Decrements block 08 by 00000001h |
| `-0812345678` | Decrements block 08 by 12345678h |

### 7.4.4.5 Copy Value Block ‘=’

Use this command to copy a value block to another value block of the same sector. A read after copy is performed automatically. The command fails if one of the specified blocks is not in value format. You must log into the respective sector before this command can be executed.

**Command:** `‘=[SourceBlock][TargetBlock]’`

| Parameters | Description |
|---|---|
| `[SourceBlock]` | Source block address (1 byte) |
| `[TargetBlock]` | Target block address (1 byte) |

**Answer**

| Answer | Description |
|---|---|
| `Data<CR><LF>` | New value of target block |
| `‘F<CR><LF>’` | Error: general failure |
| `‘I<CR><LF>’` | Error: invalid block format |

**Examples**

| Command | Description |
|---|---|
| `=0809` | Copy value block 08 to block 09 |
| `=080A` | Copy value block 08 to block 0A |

## 7.4.5 Mifare® PLUS Related Commands

### 7.4.5.1 *Write Personalization Data ‘nwp’*

Use this command to transmit personalization data to the transponder in SL0. At least Card Master key, Card configuration key, Level 2 switch key (Mifare® PLUS X only) and Level 3 switch key must be written in order to personalize the transponder. The personalization is finished by sending the command ‘Commit Personalization’. Personalization shall happen at a secure place because all data is transmitted in plain.

**Command:** **`‘nwp[DataBlock / SpecialBlock][KeyType][Data]’`**

| Parameters | Description |
|---|---|
| **`[DataBlock / SpecialBlock]`** | If KeyType is present and set to AA or BB, the first parameter is interpreted as SpecialBlock. This makes it possible to address e.g. AES keys and Configuration blocks. Every sector can hold two AES keys (type A and type B). Otherwise the standard data blocks are accessed (00h … FFh)<br>00h … 27h: Parameter addresses an AES key of data sector<br>28h: Card master key<br>29h: Card configuration key<br>2Ah: Level 2 switch key<br>2Bh: Level 3 switch key<br>2Ch: SL1 card authentication key<br>2Dh: Select Virtual Card key<br>2Eh: Proximity Check key<br>2Fh: Virtual Card Polling Encryption key<br>30h: Virtual Card Polling MAC key<br>40h: MFP Configuration Block<br>41h: Installation Identifier<br>42h: ATS Information<br>43h: Field Configuration Block |
| **`[KeyType]`** | AAh: indicate key type A<br>BBh: indicate key type B |
| **`[Data]`** | Data to be written to the transponder (16 bytes) |

**Answer**

| Answer | Description |
|---|---|
| **`Data<CR><LF>`** | Written data |
| **`‘F<CR><LF>’`** | Error: General failure |
| **`‘R<CR><LF>’`** | Error: Parameter out of range |

**Examples**

| Command | Description |
|---|---|
| **`nwp0100112233445566778899AABBCCDDEEFF<CR>`** | Writes 00112233445566778899AABBCCDDEEFFh to data block 01 |
| **`nwp00AA00112233445566778899AABBCCDDEEFF`** | Set AES key of sector 00, key type A, key is: 00112233445566778899AABBCCDDEEFFh |
| **`nwp01BB00112233445566778899AABBCCDDEEFF`** | Set AES key of sector 01, key type B, key is: 00112233445566778899AABBCCDDEEFFh |
| **`nwp28AA000102030405060708090A0B0C0D0E0F`** | Set Card master key to 000102030405060708090A0B0C0D0E0Fh |

### 7.4.5.2 Commit Personalization ‘ncp’

Use this command to finish personalization and to switch the transponder from SL0 to SL1. If the personalized transponder is a so-called ‘L3’-card, the transponder is switched to SL3. This command is only supported if the transponder is in SL0.

**Command:** `‘ncp’`

**Answer**

| Answer | Description |
|---|---|
| `‘L<CR><LF>’` | Personalization successfully completed |
| `‘F<CR><LF>’` | Error: General failure |

### 7.4.5.3 Write AES Sector Key / Special Block ‘nw’

Use this command to write an AES sector key or a special block. If an AES sector key shall be written, you have to be authenticated to the respective sector and the respective key type. If the transponder has been switched to SL2, an authentication with the Card master key is mandatory prior authentication to the respective sector. If a Special block shall be written, you must be authenticated either with the Card master key or the Card configuration key:

| Key | Change |
|---|---|
| Card master key | - Level switch keys<br>- Card configuration key<br>- MFP Configuration block<br>- Installation Identifier<br>- ATS<br>- Card master key |
| Card configuration key | - Field configuration block<br>- Virtual card keys<br>- Proximity check key<br>- Card configuration key |

**Command:** `‘nw[AESSectorKey / SpecialBlock][KeyType][Data]’`

| Parameters | Description |
|---|---|
| `[AESSectorKey / SpecialBlock]` | If KeyType is present and set to AA or BB, the first parameter is interpreted as AESKey. This makes it possible to address the AES keys and their respective key type. Every sector can hold two AES keys (type A and type B).<br>00h … 27h: Parameter addresses an AES key of a data sector<br>28h: Card master key<br>29h: Card configuration key<br>2Ah: Level 2 switch key<br>2Bh: Level 3 switch key<br>2Ch: SL1 card authentication key<br>2Dh: Select Virtual Card key<br>2Eh: Proximity Check key<br>2Fh: Virtual Card Polling Encryption key<br>30h: Virtual Card Polling MAC key<br>40h: MFP Configuration Block<br>41h: Installation Identifier<br>42h: ATS Information<br>43h: Field Configuration Block |
| `[KeyType]` | AAh: indicate key type A<br>BBh: indicate key type B |
| `[Data]` | Data to be written to the transponder (16 bytes) |

### Answer

| Answer | Description |
|---|---|
| `Data<CR><LF>` | Written data |
| `'F<CR><LF>'` | Error: General failure |
| `'R<CR><LF>'` | Error: Parameter out of range |

### Examples

| Command | Description |
|---|---|
| `nw00AA00112233445566778899AABBCCDDEEFF` | Change AES sector key, sector 00, key type A to 00112233445566778899AABBCCDDEEFFh |
| `nw01BB00000000000000000000000000000000` | Change AES sector key, sector 01, key type B to 00000000000000000000000000000000h |
| `nw2800102030405060708090A0B0C0D0E0F0<CR>` | Change Card master key to 000102030405060708090A0B0C0D0E0Fh |

## 7.4.5.4 Proximity Check 'npc'

Use this command to perform a proximity check in order to verify if the transponder is within the proximity of the reader. This helps you to secure the reader from relay attacks. Depending on the configuration of the transponder, proximity check might even be mandatory before any login can take place. Note that only Mifare® PLUS X transponders support proximity check.

**Command:** `'npc[ProxCheckKey]'`

### Answer

| Answer | Description |
|---|---|
| `'F<CR><LF>'` | Error: Proximity check failed |
| `'L<CR><LF>'` | Proximity check succeeded |

### Examples

| Command | Description |
|---|---|
| `'npc50<CR>'` | Perform proximity check using EEPROM AES key 00 |
| `'npc000102030405060708090A0B0C0D0E0F'` | Perform proximity check using specified key 00102030405060708090A0B0C0D0E0F0h |

## 7.4.6 Transparent Transponder Access ‘t’

This command is used to communicate with ISO14443-4 Type A/B or ISO15693 compatible transponders. Data passed as argument is forwarded to the transponder, any response received is transmitted to the host as answer.

The transponder must be selected prior further data access. Basic communication parameters like communication timeout and maximum frame size are negotiated by the reader with the card automatically; this means the commands RATS (ISO14443A) and ATTRIB (ISO14443B) are part of the selection process. The maximum allowed frame size is 40 bytes.

The passed data must comply with ISO14443-4 data framing. For details regarding the Mifare® DESFire command set, please contact NXP Semiconductors in order to receive the appropriate documentation.

**Command:** **`‘t[DataLength][0F][ISO14443-4 Frame]’`**

**Answer**

| Answer | Description |
|---|---|
| `‘Response<CR><LF>’` | ISO 14443-4 compliant response from transponder |
| `‘N<CR><LF>’` | Error: No response from transponder |

**Example**

| Command | Description |
|---|---|
| `t020FABCD` | Sends two bytes `ABCD` to the transponder, received data will be forwarded to the host. |

## *7.5 Setup Related Commands of the Reader*

### 7.5.1 Read Byte from EEPROM ‘rp’

Use this command to read a byte from the internal reader EEPROM. Only the configuration bank is accessible by the user, it is not possible to read any stored keys. The address byte ranges from 00h to FFh.

**Command:** **`‘rp[EEPROMAddr]’`**

**Answer**

| Answer | Description |
|---|---|
| `Data<CR><LF>` | EEPROM data (1 byte) |

**Example**

| Command | Description |
|---|---|
| `rp0A` | Reads register 0Ah (Station ID) |

### 7.5.2 Write Byte to EEPROM ‘wp’

Use this command to write a byte to the internal reader EEPROM. Only the configuration bank is accessible by the user, if any keys shall be written, the respective commands have to be used. The address byte ranges from 00h to FFh. A read after write check is performed automatically.

**Command:** **`‘wp[EEPROMAddr][Data]’`**

**Answer**

| Answer | Description |
|---|---|
| `Data<CR><LF>` | EEPROM data (1 byte) |
| `‘F<CR><LF>’` | Error: read after write failure |

**Example**

| Command | Description |
|---|---|
| `wp0A15` | Writes 15h to register 0A (Station ID) |

## 7.5.3 Set Configuration Flags ‘of’

Use this command to set or clear the Configuration Flags of the reader. In contrast to the EEPROM registers, the settings of the ‘of’ command are volatile, this means, they are taking effect on the behavior of the reader at once and they are lost, when a reset occurs or the reader is powered down.

**Command:** `‘of[FlagAddr][State]’`

### Answer

| Answer | Description |
|---|---|
| `Data<CR><LF>` | State of the modified flag |
| `‘R<CR><LF>’` | Error: Flag out of range |

| Flag address | Description |
|---|---|
| 00h | MultiTag |
| 01h | NewSerialMode |
| 05h | ExtendID |
| 07h | NoisyEnvironment |
| 08h | Reset Recovery Time Multiplier |
| 11h | ReqAExtendID |
| 12h | TagInfo |

### Examples

| Command | Description |
|---|---|
| `of0001` | Set MultiTag flag |
| `of0501` | Set ExtendID flag |
| `of0700` | Clear NoisyEnvironment flag |

## 7.5.4 Set Configuration Register ‘og’

Use this command to set the Configuration registers of the reader. In contrast to the EEPROM registers, the settings of the ‘og’ command are volatile, this means they are taking effect on the behavior of the reader at once and they are lost, when a reset occurs or the reader is powered down.

**Command:** `‘og[RegAddr][Value]’`

**Answer**

| Answer | Description |
|---|---|
| `Data<CR><LF>` | Value of the modified register |
| `‘R<CR><LF>’` | Error: Register out of range |

| Register address | Description |
|---|---|
| 03h | Reset Off Time |
| 04h | Reset Recovery Time |
| 08h | AFI |

**Example**

| Command | Description |
|---|---|
| `og0310` | Set Reset Off Time to 10h (16ms) |
| `og0420` | Set Reset Recovery Time to 20h (32ms) |

## 7.5.5 Set Tag Type ‘oa’ / ‘ob’ / ‘op’ / ‘os’ / ‘ot’ / ‘ov’

Use this command to search for only one specific transponder type (e.g. only ISO14443A). In contrast to the EEPROM registers, the settings of the ‘o’ command are volatile, this means they are taking effect on the behavior of the reader at once and they are lost when a reset occurs or the reader is powered down.

**Command:** `‘oa’ / ‘ob’ / ‘op’ / ‘os’/ ‘ot’ / ‘ov’`

**Answer**

| Answer | Description |
|---|---|
| `‘OA<CR><LF>’` | Search for ISO14443A transponders |
| `‘OB<CR><LF>’` | Search for ISO14443B transponders |
| `‘OP<CR><LF>’` | Search for PicoPass / ICLASS transponders |
| `‘OS<CR><LF>’` | Search for SRX transponders |
| `‘OT<CR><LF>’` | Search for all supported transponders |
| `‘OV<CR><LF>’` | Search for ISO15693 transponders |

## 7.5.6 Include Tag Type ‘o+a’ / ‘o+b’ / ‘o+p’ / ‘o+s’ / ‘o+v’

Use this command to include specific transponder types to the transponder search. In contrast to the EEPROM register, the settings of the ‘o+’ command are volatile, this means they are taking effect on the behavior of the reader at once and they are lost when a reset occurs or the reader is powered down.

**Command:** `'o+a' / 'o+b' / 'o+p' / 'o+s' / 'o+v'`

**Answer**

| Answer | Description |
|---|---|
| `'O+A<CR><LF>'` | Search for ISO14443A transponders |
| `'O+B<CR><LF>'` | Search for ISO14443B transponders |
| `'O+P<CR><LF>'` | Search for PicoPass / ICLASS transponders |
| `'O+S<CR><LF>'` | Search for SRX transponders |
| `'O+V<CR><LF>'` | Search for ISO15693 transponders |

## 7.5.7 Exclude Tag Type ‘o-a’ / ‘o-b’ / ‘o-p’ / ‘o-s’ / ‘o-v’

Use this command to exclude specific transponder types from the transponder search. In contrast to the EEPROM register, the settings of the ‘o-’ command are volatile, this means they are taking effect on the behavior of the reader at once and they are lost when a reset occurs or the reader is powered down.

**Command:** `'o-a' / 'o-b' / 'o-p' / 'o-s' / 'o-v'`

**Example**

| Answer | Description |
|---|---|
| `'O-A<CR><LF>'` | Don’t search for ISO14443A transponders |
| `'O-B<CR><LF>'` | Don’t search for ISO14443B transponders |
| `'O-P<CR><LF>'` | Don’t search for PicoPass / ICLASS transponders |
| `'O-S<CR><LF>'` | Don’t search for SRX transponders |
| `'O-V<CR><LF>'` | Don’t search for ISO15693 transponders |

# 7.5.8 Key Management

## 7.5.8.1 Write Mifare® Classic Key ‘wm’

Use this command to store a Mifare® Classic authentication key into the EEPROM of the reader. The reader is able to store up 32 keys. The key locations are write-only, so the keys can’t be read back. Each key is 6 bytes long and is stored redundantly to increase data integrity.

**Command:** `'wm[KeyNumber][Key]'`

**Answer**

| Answer | Description |
|---|---|
| `Key<CR><LF>` | Written key |

**Example**

| Command | Description |
|---|---|
| `wm1A1234567890AB` | Writes key `1234567890AB` to location 1Ah |

## 7.5.8.2 Write AES / Triple DES Key ‘wd’

Use this command to store a 16-Byte Triple-DES authentication key into the EEPROM of the reader. The reader is able to store up to 16 keys. The key locations are write-only, so the keys can’t be read back.

**Command:** `'wd[KeyNumber][Key]'`

**Answer**

| Answer | Description |
|---|---|
| `Key<CR><LF>` | Written key |
| `'F<CR><LF>'` | Error: Writing of key failed |
| `'R<CR><LF>'` | Error: Key number out of range |

**Example**

| Command | Description |
|---|---|
| `wd0B000102030405060708090A0B0C0D0E0F` | Writes Triple-DES key `000102030405060708090A0B0C0D0E0F` to location 0Bh |

## 7.5.9 SAM related Commands

The reader supports the connection of an ISO 7816 compatible SAM card (Secure Access Module). The SAM card is driven by the reader via an ISO 7816 compatible UART, using the following default parameters: $f_{CLK}$ = 1.043 MHz (13.56MHz/13); $F_i$ = 372; $D_i$ = 1 (resulting baudrate $f_{BAUD}$ = 2804 Baud).

If the SAM supports PPS exchange (Protocol and parameter selection), the card can be switched to a higher baudrate using the following parameters: $f_{CLK}$ = 1.043 MHz; $F_i$ = 512; $D_i$ = 16 (resulting baudrate $f_{BAUD}$ = 32594 Baud).

### *7.5.9.1 Init SAM 'ei'*

Use this command to perform a warm reset procedure of the SAM card according to ISO 7816-3. If desired, the command also performs a subsequent PPS (Protocol and parameter selection) exchange. Note that not all ISO 7816 compatible cards support PPS.

**Command: `'ei[PPS]'`**

**Answer**

| Answer | Description |
|---|---|
| `'ATR<CR><LF>'` | ISO 7816 compliant answer to reset from the SAM card |
| `'N<CR><LF>'` | Error: No response from SAM card |

**Examples**

| Command | Description |
|---|---|
| `ei00` | Init SAM card, perform no PPS exchange |
| `ei01` | Init SAM card, perform PPS exchange |

### *7.5.9.2 SAM Transmit Data 'et'*

Use this command to transmit data to the SAM card. The card's response is returned as answer. The UART supports T=0 and T=1 protocols according to ISO 7816-3.

**Command: `'et[DataLength][00][Data]'`**

**Answer**

| Answer | Description |
|---|---|
| `'Response<CR><LF>'` | Response from the card |
| `'N<CR><LF>'` | SAM card not initialized, perform 'Init SAM' command before |
| `'F<CR><LF>'` | No response from SAM card |

**Examples**

| Command | Description |
|---|---|
| `et0200ABCD` | Sends two bytes **`ABCD`** to the card, received data will be returned to the host. |

# *7.6 Miscellaneous Commands*

## 7.6.1 Get Station ID ‘g’

Use this command to retrieve the station ID of the reader.

**Command:** `‘g’`

| Answer | Description |
|---|---|
| `Data<CR><LF>` | Station ID of the reader (1 byte) |

## 7.6.2 Antenna Power Off ‘poff’

Use this command to turn off the RF-field and save energy. All present tags in the antenna field are powered down and reset.

**Command:** `‘poff’`

| Answer | Description |
|---|---|
| `‘P<CR><LF>’` | Power off command performed |

## 7.6.3 Antenna Power On ‘pon’

Use this command to turn on the RF-field manually. If a tag-related command is submitted, the RF-field is also turned on.

**Command:** `‘pon’`

| Answer | Description |
|---|---|
| `‘P<CR><LF>’` | Power on command performed |

## 7.6.4 Get Version ‘v’

Use this command to receive the current firmware version of the reader.

**Command:** `‘v’`

| Answer | Description |
|---|---|
| `‘MultiISO 1.17<CR><LF>’` | Version string |

## 7.6.5 Reset ‘x’

Use this command to perform a software reset of the reader.

**Command:** `‘x’`

| Answer | Description |
|---|---|
| `‘MultiISO 1.17<CR><LF>’` | Version string |

## 7.6.6 Break

Send a SPACE (DOT if NoisyEnvironment flag is set) to cancel e.g. Continuous Read Mode.

## 7.6.7 Read GPIO ‘ir’

Use this command to read a specified GPIO. The GPIO is switched to input prior reading.

**Command:** **`‘ir[GPIONumber]’`**

**Answer**

| Answer | Description |
|---|---|
| `Data<CR><LF>` | Status of specified GPIO: 00 means Low; 01 means High. |

**Example**

| Command | Description |
|---|---|
| `‘ir02’` | Returns status of GPIO2. |

## 7.6.8 Write GPIO ‘iw’

Use this command to write a specified GPIO. The GPIO is switched to output prior writing. Please consider the reader’s maximum current source and sink capability!

**Command:** **`‘iw[GPIONumber][Status]’`**

**Answer**

| Answer | Description |
|---|---|
| `Data<CR><LF>` | Status of specified GPIO: 00 means Low; 01 means High. |

**Example**

| Command | Description |
|---|---|
| `‘iw0201’` | Sets GPIO2 to logical state High |
| `‘iw0300’` | Sets GPIO3 to logical state Low |

## 7.6.9 Read GPIO1 ‘pr’

Use this command to read GPIO1. GPIO1 is switched to input prior reading. The command has the same effect as command `‘ir01’`. It has been implemented in order to adhere backward compatibility.

**Command:** **`‘pr’`**

**Answer**

| Answer | Description |
|---|---|
| `Data<CR><LF>` | Status of GPIO1: 00 means Low; 01 means High. |

## 7.6.10 Write GPIO1 ‘pw’

Use this command to write GPIO1. GPIO1 is switched to output prior writing. Please consider the reader’s maximum current source and sink capability! The command has the same effect as command `‘iw010X’`. It has been implemented in order to maintain backward compatibility.

**Command:** **`‘pw[Status]’`**

**Answer**

| Answer | Description |
|---|---|
| `Data<CR><LF>` | Status of GPIO1: 00 means Low; 01 means High. |

### 7.6.11 Beep ‘b’

Use this command to perform a beep. Tone length and frequency are depending on the settings in the Beep tone and length register. The command answers with the value written in the register.

**Command:** `‘b’`

| Answer | Description |
|---|---|
| `‘22<CR><LF>’` | Sounds a beep with tone length 125ms and frequency 587Hz |

### 7.6.12 Resend Last Answer ‘ra’

Use this command to resend the last answer, the reader has sent to the host.

**Command:** `‘ra’`

| Answer | Description |
|---|---|
| `‘E0040100025CCD44<CR><LF>’` | Last read UID |

### 7.6.13 Field Reset ‘y’

Use this command to perform a manual field reset. All present tags in the antenna field are brought into power-up state. The timing of the reset can be controlled by parameters.

**Command:** `‘y[ResetOffTime][ResetRecoveryTime]’`

**Answer**

| Answer | Description |
|---|---|
| `‘Y<CR><LF>’` | Field reset performed |

**Example**

| Command | Description |
|---|---|
| `‘y1020’` | Field reset: Reset Off Time = 16ms<br>Reset Recovery Time = 32ms |

### 7.6.14 Lock Block (ISO15693 only) ‘k’

Some ISO15693 transponders provide the Lock block feature, so that a data block can be turned into a read-only block. Once a block has been locked, the protection cannot be removed any more.

**Command:** `‘k[BlockAddr]’`

**Answer**

| Answer | Description |
|---|---|
| `Data<CR><LF>` | Block address |
| `‘F<CR><LF>’` | Error: write failure |
| `‘N<CR><LF>’` | Error: no tag in the field, or the tag does not respond |
| `‘O<CR><LF>’` | Error: selected tag does not support block locking |
| `‘X<CR><LF>’` | Error: the block has already been locked |

**Example**

| Command | Description |
|---|---|
| `k04` | Lock block 04 |

## 7.6.15 AES Data Encryption/Decryption 'as'

Use this command to perform AES encryption/decryption on 16 byte data blocks. Keys can be passed via serial connection, optionally the internally stored keys can be used.

**Command:** `'as' [Options] [Key/KeyNumber] [Data]`

| Parameters | Description |
|---|---|
| `[Options]` | Option byte (1 byte) |
| `[Key/KeyNumber]` | Key (16 bytes) / Key number (1 byte), 00h…0Fh |
| `[Data]` | Data to be encrypted/decrypted (16 bytes) |

### Option byte:

| Options | | | | | | | |
|---|---|---|---|---|---|---|---|
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| RFU | RFU | RFU | RFU | RFU | Encode | RFU | KeyIndex |

Encode:

| | |
|---|---|
| 1 | Encode the data |
| 0 | Decode the data |

KeyIndex:

| | |
|---|---|
| 1 | The command uses the EEPROM key, specified by KeyNumber |
| 0 | The key is passed via serial connection |

### Answer

| Answer | Description |
|---|---|
| `'Data<CR><LF>'` | Encrypted/Decrypted data |
| `'R<CR><LF>'` | Error: Key index out of range |

### Examples

| Command | Description |
|---|---|
| `as000011223344556677889 9AABBCCDDEEFF00102030 405060708090A0B0C0D0E0F0` | Decode ciphertext 00102030405060708090A0B0C0D0E0F0 using key 00112233445566778899AABBCCDDEEFF<br>Plaintext is 781536D4D011A7C5389598AD60AAFFA0h |
| `as0509001020304050607080 90A0B0C0D0E0F0` | Encode plaintext 00102030405060708090A0B0C0D0E0F0 using EEPROM-key 09h. |

## 7.6.16 DES / Triple-DES Data Encryption/Decryption 'ds'

Use this command to perform encryption/decryption operations on 8 byte data blocks. The command offers both Single-DES and Triple-DES crypto functions. Triple-DES cryptology is based on two keys data encryption/decryption. Keys can be passed via serial connection, optionally the internally stored keys can be used.

**Command:** `'ds' [Options] [Key/KeyNumber] [Data]`

| Parameters | Description |
|---|---|
| `[Options]` | Option byte (1 byte) |
| `[Key/KeyNumber]` | Key (8 or 16 bytes) / Key number (1 byte), 00h…0Fh |
| `[Data]` | Data to be encrypted/decrypted (8 bytes) |

**Option byte:**

| Options | | | | | | | |
|---|---|---|---|---|---|---|---|
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| RFU | RFU | RFU | RFU | RFU | Encode | KeyLength | KeyIndex |

Encode:

| | |
|---|---|
| 1 | Encode the data |
| 0 | Decode the data |

KeyLength:

| | |
|---|---|
| 1 | Use Triple-DES algorithm with 16 bytes key |
| 0 | Use Single-DES algorithm with 8 bytes key |

KeyIndex:

| | |
|---|---|
| 1 | The command uses the EEPROM key, specified by KeyNumber |
| 0 | The key is passed via serial connection |

**Answer**

| Answer | Description |
|---|---|
| `'Data<CR><LF>'` | Encrypted/Decrypted data |
| `'R<CR><LF>'` | Error: Key index out of range |

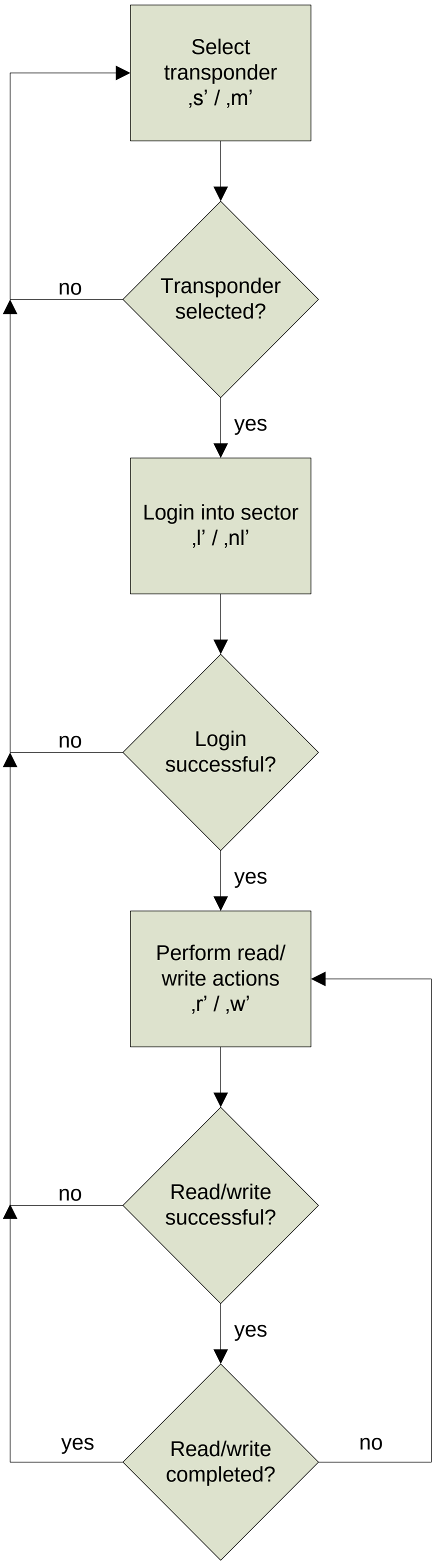
**Examples**

| Command | Description |
|---|---|
| `ds0011223344556677881020304050607080` | Decode ciphertext 1020304050607080 with Single-DES algorithm using key 1122334455667788<br>Plaintext is 1B3446C2C2AA73CDh |
| `ds070A1020304050607080` | Encode plaintext 1020304050607080 with Triple-DES algorithm using EEPROM-key 0Ah. |

# 8. Typical Data Transaction Procedures

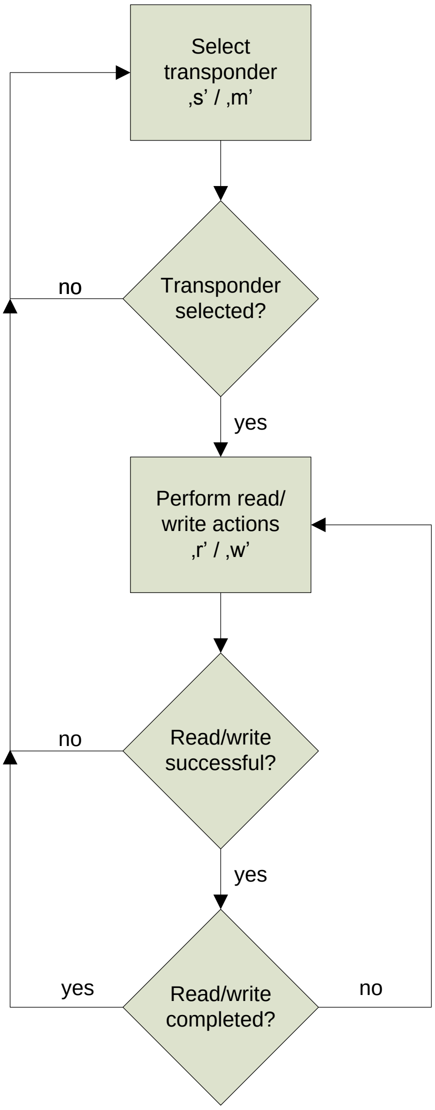
## 8.1 Mifare® Classic / Mifare® PLUS

The following diagram shows the typical command-flow in order to access the data area of a Mifare® Classic / PLUS transponder:



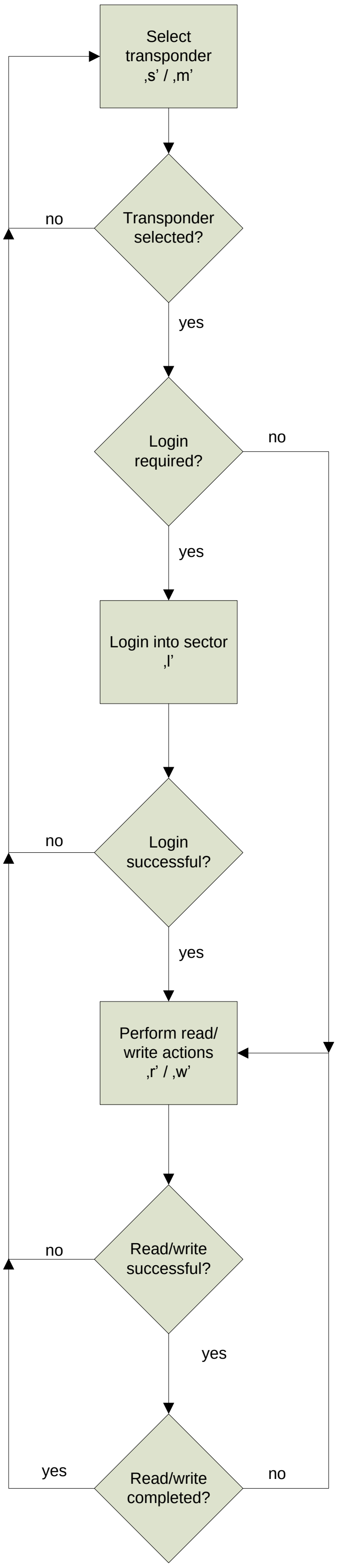
## *8.2 Mifare® Ultralight*

In contrast to Mifare® Classic transponders, no login is required to access the EEPROM memory:



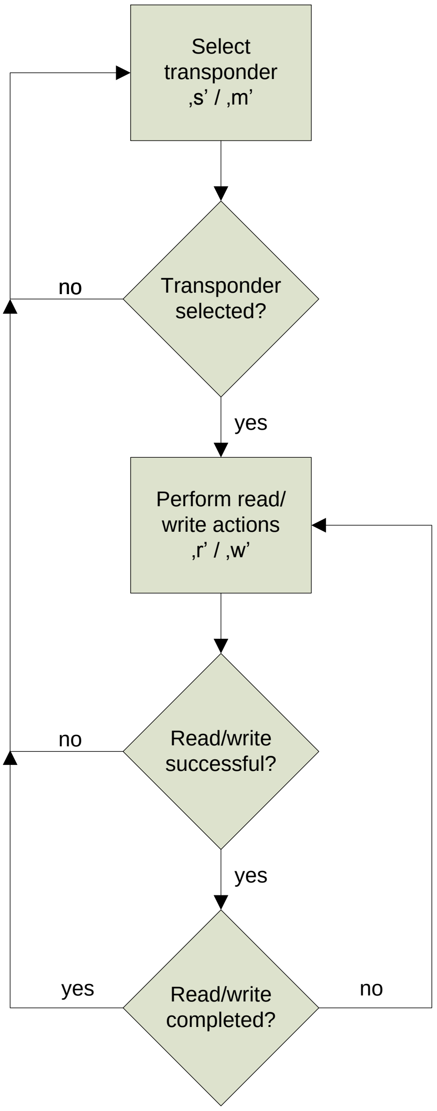
## 8.3 Mifare® Ultralight C

Accessing the EEPROM depends on the settings of the transponder, e.g. if an authentication is required. The following diagram shows the typical command-flow in order to access the data area of a Mifare® Ultralight C transponder:



## 8.4 I•Code SLI / SRX

The following diagram shows the typical command-flow in order to access the data area of an I•Code SLI or SRX transponder. No authentication is required prior accessing the data area:



# 9. Memory Organization of Mifare® Transponders

## 9.1 Mifare® Classic 1k

A Mifare® Classic 1k transponder consists of 16 sectors. Each sector is organized in four data blocks. A data block stores 16 bytes of data. The following table shows the memory organization of a Mifare® Classic 1k card:

| Sector address | Block addresses | | | |
|---|---|---|---|---|
| 00h | 03h | 02h | 01h | 00h |
| 01h | 07h | 06h | 05h | 04h |
| 02h | 0Bh | 0Ah | 09h | 08h |
| 03h | 0Fh | 0Eh | 0Dh | 0Ch |
| 04h | 13h | 12h | 11h | 10h |
| 05h | 17h | 16h | 15h | 14h |
| 06h | 1Bh | 1Ah | 19h | 18h |
| 07h | 1Fh | 1Eh | 1Dh | 1Ch |
| 08h | 23h | 22h | 21h | 20h |
| 09h | 27h | 26h | 25h | 24h |
| 0Ah | 2Bh | 2Ah | 29h | 28h |
| 0Bh | 2Fh | 2Eh | 2Dh | 2Ch |
| 0Ch | 33h | 32h | 31h | 30h |
| 0Dh | 37h | 36h | 35h | 34h |
| 0Eh | 3Bh | 3Ah | 39h | 38h |
| 0Fh | 3Fh | 3Eh | 3Dh | 3Ch |

## 9.2 Mifare® Classic 4k

A Mifare® Classic 4k transponder consists of 40 sectors. In contrast to the Mifare® standard 1k transponder, the 4k version has a different memory organization, shown in the following table:

| | Sector | Block addresses | Login sector | Sector | Block addresses | Login sector |
|---|---|---|---|---|---|---|
| 000h - 7FFh | 00h | 00h...03h | 00h | 10h | 40h...43h | 10h |
| | 01h | 04h...07h | 01h | 11h | 44h...47h | 11h |
| | 02h | 08h...0Bh | 02h | 12h | 48h...4Bh | 12h |
| | 03h | 0Ch...0Fh | 03h | 13h | 4Ch...4Fh | 13h |
| | 04h | 10h...13h | 04h | 14h | 50h...53h | 14h |
| | 05h | 14h...17h | 05h | 15h | 54h...57h | 15h |
| | 06h | 18h...1Bh | 06h | 16h | 58h...5Bh | 16h |
| | 07h | 1Ch...1Fh | 07h | 17h | 5Ch...5Fh | 17h |
| | 08h | 20h...23h | 08h | 18h | 60h...63h | 18h |
| | 09h | 24h...27h | 09h | 19h | 64h...67h | 19h |
| | 0Ah | 28h...2Bh | 0Ah | 1Ah | 68h...6Bh | 1Ah |
| | 0Bh | 2Ch...2Fh | 0Bh | 1Bh | 6Ch...6Fh | 1Bh |
| | 0Ch | 30h...33h | 0Ch | 1Ch | 70h...73h | 1Ch |
| | 0Dh | 34h...37h | 0Dh | 1Dh | 74h...77h | 1Dh |
| | 0Eh | 38h...3Bh | 0Eh | 1Eh | 78h...7Bh | 1Eh |
| | 0Fh | 3Ch...3Fh | 0Fh | 1Fh | 7Ch...7Fh | 1Fh |
| 800h - FFFh | 20h | 80h...8Fh | 20h | 24h | C0h...CFh | 30h |
| | 21h | 90h...9Fh | 24h | 25h | D0h...DFh | 34h |
| | 22h | A0h...AFh | 28h | 26h | E0h...EFh | 38h |
| | 23h | B0h...BFh | 2Ch | 27h | F0h...FFh | 3Ch |

## 9.3 Mifare® PLUS

Mifare® PLUS 2K/4K transponders have a similar memory organization like Mifare® Classic 4K. However, a 2Kbyte card only has the lower 32 Sectors built-in:

| Bytes | Sector | Block addresses | Sector | Block addresses |
|---|---|---|---|---|
| 000h – 7FFh | 00h | 00h...03h | 10h | 40h...43h |
| | 01h | 04h...07h | 11h | 44h...47h |
| | 02h | 08h...0Bh | 12h | 48h...4Bh |
| | 03h | 0Ch...0Fh | 13h | 4Ch...4Fh |
| | 04h | 10h...13h | 14h | 50h...53h |
| | 05h | 14h...17h | 15h | 54h...57h |
| | 06h | 18h...1Bh | 16h | 58h...5Bh |
| | 07h | 1Ch...1Fh | 17h | 5Ch...5Fh |
| | 08h | 20h...23h | 18h | 60h...63h |
| | 09h | 24h...27h | 19h | 64h...67h |
| | 0Ah | 28h...2Bh | 1Ah | 68h...6Bh |
| | 0Bh | 2Ch...2Fh | 1Bh | 6Ch...6Fh |
| | 0Ch | 30h...33h | 1Ch | 70h...73h |
| | 0Dh | 34h...37h | 1Dh | 74h...77h |
| | 0Eh | 38h...3Bh | 1Eh | 78h...7Bh |
| | 0Fh | 3Ch...3Fh | 1Fh | 7Ch...7Fh |
| 800h – FFFh | 20h | 80h...8Fh | 24h | C0h...CFh |
| | 21h | 90h...9Fh | 25h | D0h...DFh |
| | 22h | A0h...AFh | 26h | E0h...EFh |
| | 23h | B0h...BFh | 27h | F0h...FFh |

## *9.4 Mifare® Ultralight*

A Mifare® Ultralight transponder consists of 16 pages with 4 bytes each. The UID is read-only and is also mapped into the memory area.

| Page | Byte 0 | Byte 1 | Byte 2 | Byte 3 | Access rights |
|---|---|---|---|---|---|
| `00h` | SN0 | SN1 | SN2 | BCC0 | Read only |
| `01h` | SN3 | SN4 | SN5 | SN6 | Read only |
| `02h` | BCC1 | Internal | Lock0 | Lock1 | Read only / Lock |
| `03h` | OTP0 | OTP1 | OTP2 | OTP3 | One time programmable |
| `04h` | Data0 | Data1 | Data2 | Data3 | Read / Write |
| `05h` | Data4 | Data5 | Data6 | Data7 | Read / Write |
| `06h` | Data8 | Data9 | Data10 | Data11 | Read / Write |
| `07h` | Data12 | Data13 | Data14 | Data15 | Read / Write |
| `08h` | Data16 | Data17 | Data18 | Data19 | Read / Write |
| `09h` | Data20 | Data21 | Data22 | Data23 | Read / Write |
| `0Ah` | Data24 | Data25 | Data26 | Data27 | Read / Write |
| `0Bh` | Data28 | Data29 | Data30 | Data31 | Read / Write |
| `0Ch` | Data32 | Data33 | Data34 | Data35 | Read / Write |
| `0Dh` | Data36 | Data37 | Data38 | Data39 | Read / Write |
| `0Eh` | Data40 | Data41 | Data42 | Data43 | Read / Write |
| `0Fh` | Data44 | Data45 | Data46 | Data47 | Read / Write |

## *9.5 Mifare® Ultralight C*

A Mifare® Ultralight C transponder consists of 48 pages with 4 bytes each. The UID is read-only and is also mapped into the memory area. The 16 bytes Triple-DES key is stored in the last 4 pages of the transponder.

| Page | Byte 0 | Byte 1 | Byte 2 | Byte 3 | Access rights |
|---|---|---|---|---|---|
| 00h | SN0 | SN1 | SN2 | BCC0 | Read only |
| 01h | SN3 | SN4 | SN5 | SN6 | Read only |
| 02h | BCC1 | Internal | Lock0 | Lock1 | Read only / Lock |
| 03h | OTP0 | OTP1 | OTP2 | OTP3 | One time programmable |
| 04h | Data0 | Data1 | Data2 | Data3 | Read / Write |
| 05h | Data4 | Data5 | Data6 | Data7 | Read / Write |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | Read / Write |
| 27h | Data140 | Data141 | Data142 | Data143 | Read / Write |
| 28h | Lock2 | Lock3 | RFU | RFU | Read / Write |
| 29h | CNT | CNT | RFU | RFU | Read / Write |
| 2Ah | AUTH0 | RFU | RFU | RFU | Read / Write |
| 2Bh | AUTH1 | RFU | RFU | RFU | Read / Write |
| 2Ch | Key1/K0 | Key1/K1 | Key1/K2 | Key1/K3 | Write only |
| 2Dh | Key1/K4 | Key1/K5 | Key1/K6 | Key1/K7 | Write only |
| 2Eh | Key2/K0 | Key2/K1 | Key2/K2 | Key2/K3 | Write only |
| 2Fh | Key2/K4 | Key2/K5 | Key2/K6 | Key2/K7 | Write only |

## 9.5.1 Example: Writing the Triple-DES Key

This chapter shows how to write the Triple-DES key into the transponder.

On example of Key1 = 0001020304050607h and Key2 = 08090A0B0C0D0E0Fh the command sequence shown in the table below has to be executed for writing the key. The following preconditions shall be assumed:

| | |
|---|---|
| UID: | 04169BE1ED2580h |
| K1: | 49454D4B41455242h |
| K2: | 214E4143554F5946h |

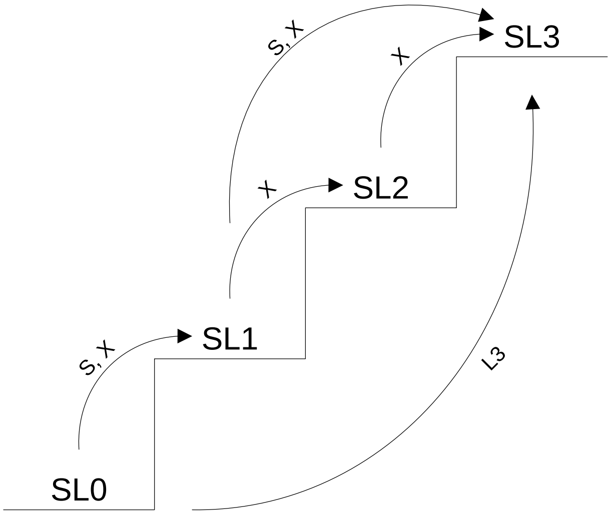
**Please note:**
Pages 2Ch … 2Fh are write-only, so the ‘w’ command will always report an error, because the automatic read-after-write check fails! Furthermore, the transponder looses its authenticated state. So it is necessary, that each write-cycle is followed by a subsequent login with the updated part of the new key:

| Command | Response |
|---|---|
| `s` | `04169BE1ED2580<CR><LF>` |
| `l00CC49454D4B41455242214E4143554F5946` | `L<CR><LF>` |
| `w2C0706050400000000000000000000000000` | `F<CR><LF>` |
| `s` | `04169BE1ED2580<CR><LF>` |
| `l00CC49454D4B04050607214E4143554F5946` | `L<CR><LF>` |
| `w2D0302010000000000000000000000000000` | `F<CR><LF>` |
| `s` | `04169BE1ED2580<CR><LF>` |
| `l00CC00010203040506070214E4143554F5946` | `L<CR><LF>` |
| `w2E0F0E0D0C000000000000000000000000` | `F<CR><LF>` |
| `s` | `04169BE1ED2580<CR><LF>` |
| `l00CC0001020304050607214E41430C0D0E0F` | `L<CR><LF>` |
| `w2F0B0A0908000000000000000000000000` | `F<CR><LF>` |
| `s` | `04169BE1ED2580<CR><LF>` |
| `l00CC000102030405060708090A0B0C0D0E0F` | `L<CR><LF>` |

# 10. Mifare® PLUS

## 10.1 Security Levels

Mifare® PLUS supports four security levels. The initial delivery configuration is SL0. Within this level, cards must be personalized. After that, the card is switched to a higher security level. Once a card has been switched to a higher level, it cannot be switched back to a lower level. Depending on card type, different ways of switching are possible:



SL0: Initial delivery configuration. Only commands ‘Select’, ‘Write Personalization Data’ and ‘Commit Personalization’ work. Depending on transponder type, the ‘Commit Personalization’ command switches the card either to SL1 or SL3.

SL1: Mifare® Classic compatibility mode. The transponder behaves like a Mifare® Classic 2K/4K card. If the transponder is a Mifare® PLUS X card, it can be switched to SL2 or directly to SL3, otherwise only SL3 is possible.

SL2: AES authentication required, Crypto1 session key is generated, communication is furthermore secured by Crypto1.

SL3: Authentication and MACing based on AES. If the transponder is a Mifare® PLUS X card, the entire communication between reader and transponder is secured by AES cryptography, Mifare® PLUS S uses plain communication.

## *10.2 Application Hints*

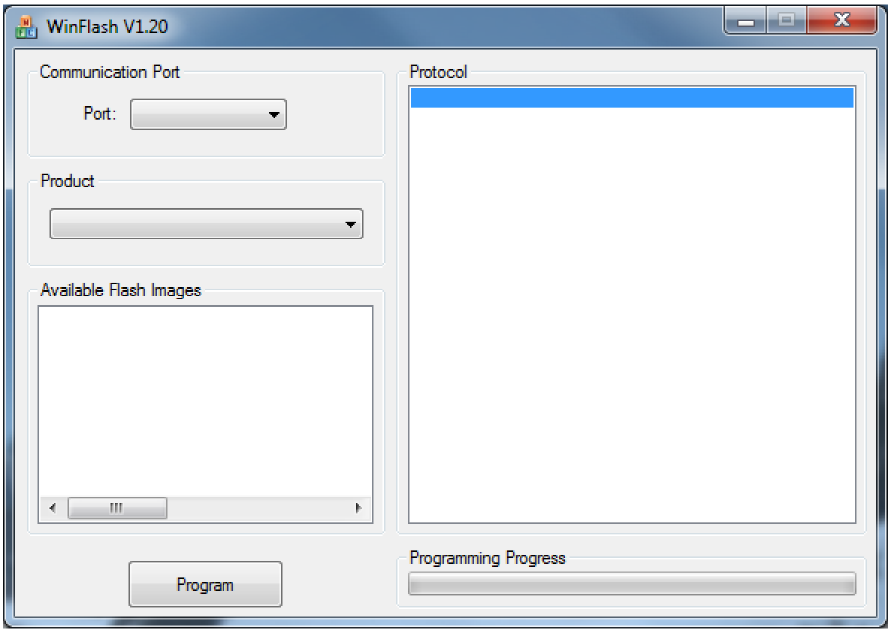
Have a look at the following hints when developing your Mifare® PLUS application. They will help you choosing the correct transponder and reader in order to satisfy your needs:

- In SL3, value formats are only supported by Mifare® PLUS X cards
- In SL3, Mifare® PLUS S transponders use plain communication on the RF-field secured by MACing. Mifare® PLUS X uses fully AES enciphered communication on the RF-field including MACing.
- Proximity Check is only supported by Mifare® PLUS X in SL3
- Mifare® PLUS is supported by firmware version 1.05 or higher

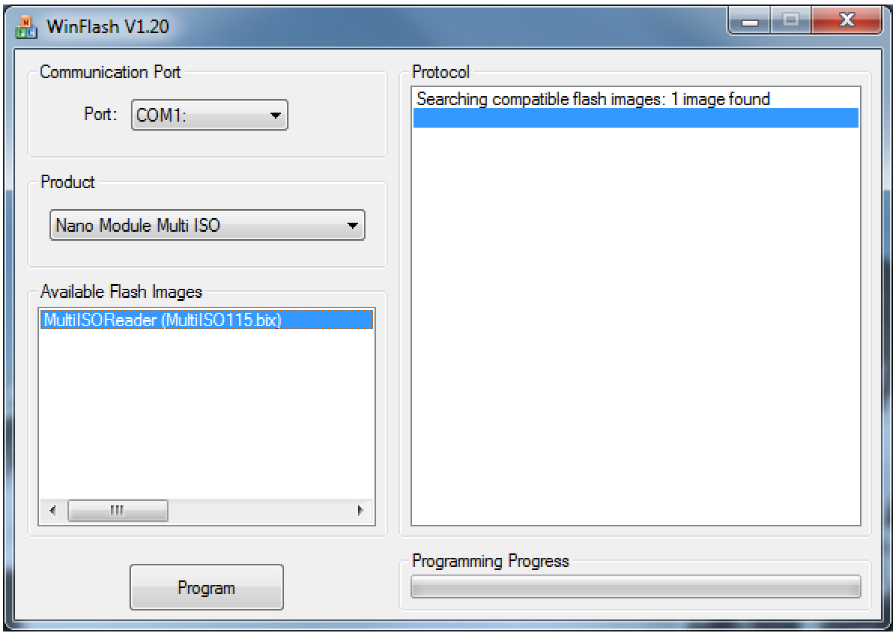
# 11. Firmware Update

The reader contains a bootloader program which makes firmware updates possible. Complete the following steps to update the firmware:

- Power up the reader
- Ensure that the reader runs at 9600 bps
- Start the application WinFlash.exe, you should see the following window



- Select the appropriate Communication Port (e.g. COM1)
- Select “Nano Module Multi ISO” from *Product*
- Select the Flash Image



Port: COM1:

Nano Module Multi ISO

MultiISOReader (MultiISO115.bix)

Searching compatible flash images: 1 image found

- Click button “Program”
- Wait while the programming process is in progress

# 12. Firmware History

| Version | Changes |
|---|---|
| **V1.00** | • Initial release |
| **V1.01** | • Baudrate control register implemented<br>• Baudrate recovery mechanism implemented<br>• Mifare® Ultralight C supported<br>• DES/3DES crypto command implemented |
| **V1.02** | • PicoPass support (UID only) |
| **V1.03** | • Improved communication parameters for PicoPass / ICLASS support |
| **V1.04** | • SAM support |
| **V1.05** | • Mifare® PLUS support<br>• AES crypto command implemented<br>• Improved communication parameters for 115200 baud |
| **V1.06** | • Display additional tag info |
| **V1.15** | • Tag type ISO14443B implemented<br>• Improved Mifare® Login: No select tag before sector change required any more<br>• Binary protocol implemented<br>• EM4034 support<br>• “Late RATS”: RATS is performed only when needed, due to problems at SmartMX MF Classic access |
| **V1.16** | • Support of ST 24LR16/64 tags<br>• New commands ‘rc’ / ‘wc’<br>• Transparent Transponder Access also supports ISO15693 |
| **V1.17** | • Support of Fujitsu MB89R118/119 tags |

# 13. Trademarks

All referenced brands, product names, service names and trademarks mentioned in this document are the property of their respective owners.