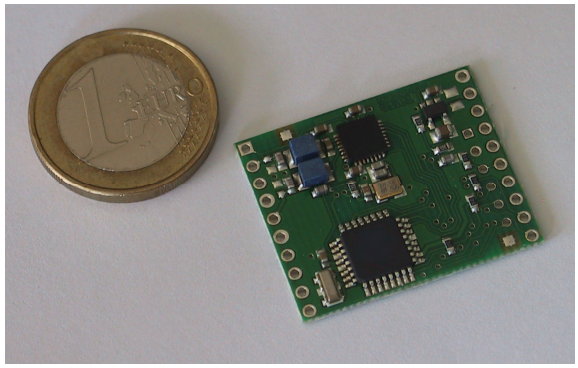
ELATEC

RFID Systems



Transponder Reader Mifare Technical Manual

Doc.-Rev. 1.04

Content

1. Introduction

This document is the reference guide for Elatec's Mifare® transponder reader family TWN3 Mifare® and NanoMF. The readers are using the same reading technology, so this documentation is applicable for both devices.

Elatec's proximity reader TWN3 Mifare® is an integrated compact reader for reading and writing 13.56MHz ISO14443A Mifare® transponders. It combines state of the art technology in a shapely and tiny housing. Reading distances of up to 8cm (depending on transponder and environment) and various cabling options are only two of the outstanding technical benefits offered by TWN3 Mifare®.

This makes TWN3 Mifare® as your ideal Mifare® reading/writing device e.g. for desktop applications.

Elatec's reader module NanoMF is designed for integration into machines, handheld computers or any other device. The focus has especially been set on size, price and flexibility.

Thanks to its compact dimensions, integration directly on a PC board is possible.

Unique features are: 4 user configurable ports (to be configured as input or output), beeper support and different sleep modes for lowest power consumption. The simple proprietary ASCII protocol enables quick software development cycles. All host communication is done via RS-232 interface.

Note:

In order to use all the functionality, which is described in this document, your Mifare® reader needs a firmware version V1.11 or above. In order to update the firmware from an elder version, please refer to chapter 9 "Firmware update".

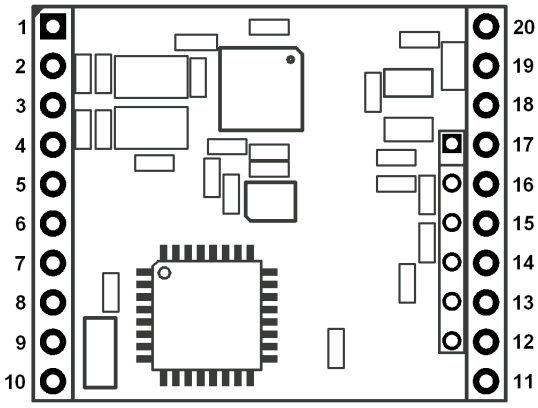
2. Installation of NanoMF

This chapter covers the installation of NanoMF in an embedded environment.

2.1 Dimensions

Dimensions: 30.48mm x 25.40mm x 3.5mm (1.2” x 1” x 0.14”)

Connectors are in 2.54mm (0.1”) grid.



Schematic representation (component side)

2.2 Pinning

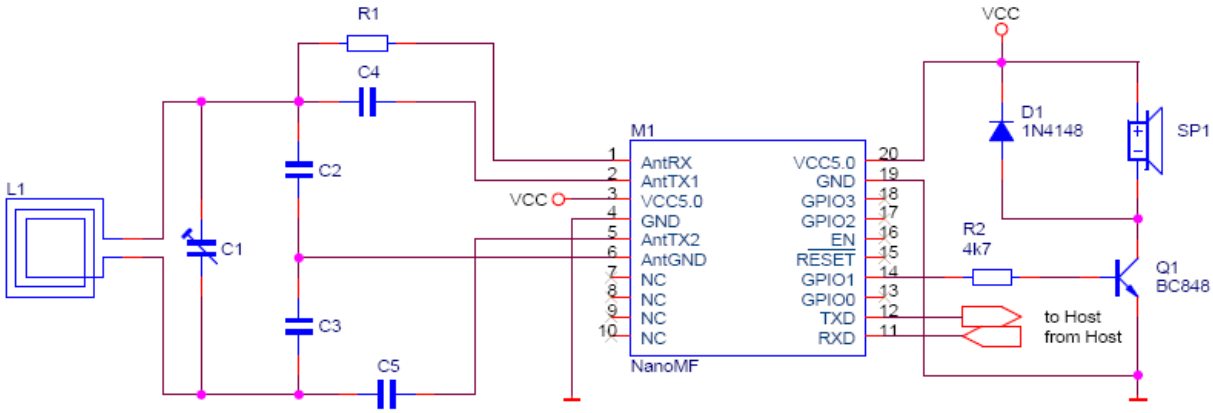
Pin	Name	Description
1	AntRX	Antenna receiver input
2	AntTX1	Antenna transmitter output 1
3, 20	VCC	3.3 – 5V
4, 6, 19	GND	Ground
5	AntTX2	Antenna transmitter output 2
7, 8, 9, 10	NC	Not connected
11	RXD	RS-232 receiver input
12	TXD	RS-232 transmitter output
13	GPIO0	General purpose input / output 0
14	GPIO1 / Beeper	General purpose input / output 1, connection port for beeper
15	Reset	Asynchronous reset, active low
16	RFU	Reserved for future use, leave open
17	GPIO2	General purpose input / output 2
18	GPIO3	General purpose input / output 3

2.3 Electrical Characteristics

Frequency	13.56 MHz	
Power supply	3.3 – 5.0V DC	
Current consumption	RF field off:	2mA
	RF field on:	Typically 80mA, depending on antenna

2.4 External Connections

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



2.4.1 Antenna

NanoMF 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.

Little PCB Antenna Design Guide:

- Antenna inductance value is usually in the range of 1 µH.
- For antenna loop up to 40 mm diameter use 5 turns.
- For antenna loop larger than 40 mm 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: 10 pF … 470 pF. 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 Serial connection

Because NanoMF is transmitting and receiving TTL levels, it can be directly connected to a microcontroller. If you plan to run NanoMF at a PC, an appropriate interface converter circuit must be connected that can handle 3.0V TTL levels.

2.4.3 Usage of GPIOs

NanoMF 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 and the overall current consumption of the complete module must not exceed 150mA. 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. NanoMF continues operation as soon as the reset pin is released.

2.4.5 Power supply

A power supply of 3.3V – 5V must be applied between the VCC and GND pins. Please make sure that the power supply provides a current capability of at least 100mA.

3. 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

4. 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.

Furthermore the reader is able to store up to 32 authentication keys to log in standard Mifare® cards internally. All keys are write-only and cannot be accessed via the interface lines.

4.1 EEPROM memory organization

Register	Access	Default value	Description
00h … 03h	RO	00h	RFU
04h	R/W	01h	Station ID
05h	R/W	01h	Protocol Configuration
06h	R/W	00h	Baudrate control register
07h	R/W	10h	Reset Off Time
08h	R/W	10h	Reset Recovery Time
09h	R/W	00h	RFU, never change this register
0Ah … 0Fh	R/W	00h	RFU, never change this register
10h … 13h	R/W	00h	User data
14h	R/W	22h	Beep tone and length (NanoMF only)

4.2 Station ID (04h)

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

4.3 Protocol Configuration register (05h)

The Protocol Configuration register 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
REQA2X	RFU	Disable Startup Message	MultiTag	Binary Timeout	ExtendID	Binary	Auto Start

4.3.1 AutoStart (default 1)

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

4.3.2 Binary (default 0)

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

4.3.3 ExtendID (default 0)

If set, the unique serial number (UID) of the transponder is extended by a single prefix byte. This setting affects the commands continuous reading (‘c’), single tag select (‘s’) and multi tag select (‘m’).

Possible values for the prefix byte are:

Prefix	Transponder type
01h	Mifare® Light Transponder
02h	Mifare® Classic 1K Transponder
03h	Mifare® Classic 4K Transponder
04h	Mifare® ProX Transponder
05h	Mifare® Ultralight Transponder
06h	Mifare® DESFire Transponder
FFh	Unknown Transponder

4.3.4 Binary Timeout (default 0)

This flag is only interpreted if the reader operates in binary mode.

If set and the serial line stays idle for more than 96ms (no data is transmitted from the host to the reader), the reader invalidates all received data and waits for valid data frames beginning with the STX code.

4.3.5 Multitag (default 0)

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.

4.3.6 DisableStartupMessage (default 0)

If set, the reader will not print the start-up message (`Mifare 1.11`) when powered up or if a reset occurs.

4.3.7 REQA2X (default 0)

If set the reader performs a second REQA if the tag does not answer to the first one.

4.4 Baudrate control register (06h)

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	RFU	BS1	BS0

BS1	BS0	Resulting Baudrate
0	0	9600 bps (default)
0	1	19200 bps
1	0	38400 bps
1	1	57600 bps

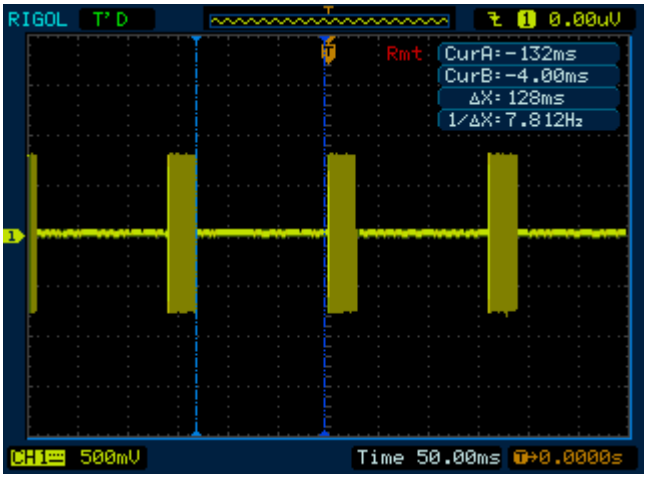
4.4.1 Resetting the baudrate to default

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, ASCII protocol. 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.

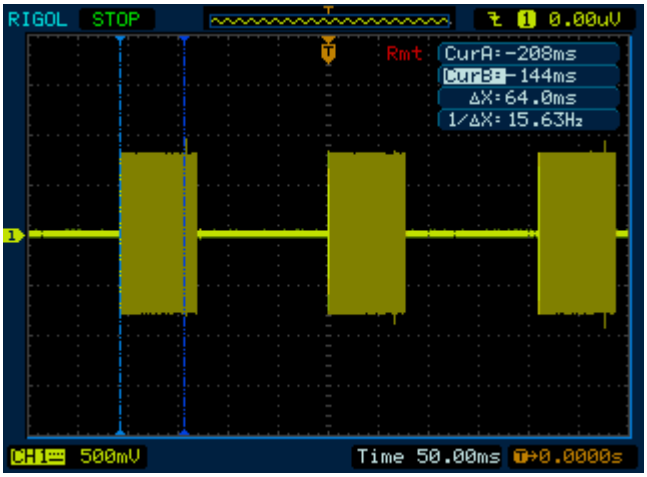
4.5 Reset Off Time (07h)

The Reset Off Time register represents the time in milliseconds, the HF-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.
The image below shows the activity of the HF-field. The Reset Off Time register has been set to 80h (128 ms):



4.6 Reset Recovery Time (08h)

The Reset Recovery Time register represents the recovery time in milliseconds after the HF-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.
The image below shows the activity of the HF-field. The Reset Recovery Time register has been set to 40h (64 ms):



4.7 User data (10h – 13h)

These registers are for free use.

4.8 Beep tone and length (14h, NanoMF only)

This register defines the beep tone (frequency) and beep length. It is only implemented on NanoMF.

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

5. Communication Protocol

5.1 ASCII Protocol

The ASCII protocol has been designed for easy handling. Data is 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:

Mifare 1.11<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. **wp071F**

5.2 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.

5.2.1 STX

Start of Transmission (02h)

5.2.2 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 its response.

5.2.3 Length

Denotes the length of the data block in bytes.

5.2.4 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.

5.2.5 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.

5.2.6 ETX

End of Transmission (03h)

5.2.7 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

5.2.8 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.

6. Instruction set

6.1 Command overview

Command	Implemented on		Description
	TWN3 Mifare	NanoMF	
`'.'`			Cancel command
`'+'`			Increment value block
`'-'`			Decrement value block
`'='`			Copy value block
`'b'`			Beep
`'c'`			Continuous read mode
`'g'`			Get station ID
`'ir'/'iw'`			Read / write GPIO0…3
`'m'`			MultiTag select / tag list
`'poff'`			Antenna power off
`'pon'`			Antenna power on
`'pr'/'pw'`			Read / write GPIO1
`'r'/'rb'`			Read data block
`'re'`			Read EEPROM register
`'rv'`			Read value block
`'s'`			Select single tag
`'v'`			Get version
`'w'/'wb'`			Write data block
`'we'`			Write EEPROM register
`'wm'`			Write Mifare® EEPROM key into reader EEPROM
`'wv'`			Write value block
`'x'`			Reset

6.2 Error Codes

Following table shows an overview of all error messages of the reader device.

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
'O<CR><LF>'	Operation mode failure
'X<CR><LF>'	Error during value operation

6.3 Transponder serial number related commands

6.3.1 Continuous read mode ‘c’

The reader device 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.

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)

6.3.2 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

6.3.3 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`
	`56AB3798`
	`02`
		First card
		Second card
		Third card
		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.

6.4 Data-transaction related commands

6.4.1 Login (authenticate tag) '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 FFh: authenticate with key type A, transport key FFFFFFFFFFFFh BBh: authenticate with key type B 10h … 2Fh: authenticate with stored key type A (00h … 1Fh) 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
`'101AA<CR>'`	Authenticate into sector 01 using transport key type A A0A1A2A3A4A5h
`'102BB<CR>'`	Authenticate into sector 02 using transport key type B B0B1B2B3B4B5h
`'103FF<CR>'`	Authenticate into sector 03 using transport key type A FFFFFFFFFFFFh
`'104AA1234567890AB'`	Authenticate into sector 04 using specified key type A 1234567890ABh
`'10510'`	Authenticate into sector 05 using EEPROM key 0, key type A
`'10637'`	Authenticate into sector 06 using EEPROM key 7, key type B

6.4.1.1 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.

6.4.2 Read data block ‘r’ / ‘rb’

This command reads an entire data block of a transponder. The range of valid block addresses depends on the used transponder. Keep in mind that any read- or write access requires a successful login into the respective sector.

Command: `'r[BlockAddr]' / 'rb[BlockAddr]'`

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
`'O<CR><LF>'`	Error: Block address out range; the block address of the ‘r’ command is greater or equal than 40h, use the ‘rb’ command instead.

Example

Command	Description
`r08`	Reads block 08 (sector 02, block 00)

6.4.2.1 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.

6.4.3 Write data block ‘w’ / ‘wb’

This command writes an entire data block to a 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.

Command: `'w[BlockAddr]' / 'wb[BlockAddr]'`

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
`'O<CR><LF>'`	Error: Operation mode failure; the block address of the ‘w’ command is greater or equal than 40h, use the ‘wb’ command instead.

Example

Command	Description
`w0800010203040506070809 0A0B0C0D0E0F`	Writes data `00010203040506070809 0A0B0C0D0E0F` to block 08 (sector 02, block 00)

6.4.3.1 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: `00000000040506070809 0A0B0C0D0E0F`
`w0400010203040506070809 0A0B0C0D0E0F`	Writes data `00010203` to page 04

6.4.4 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.

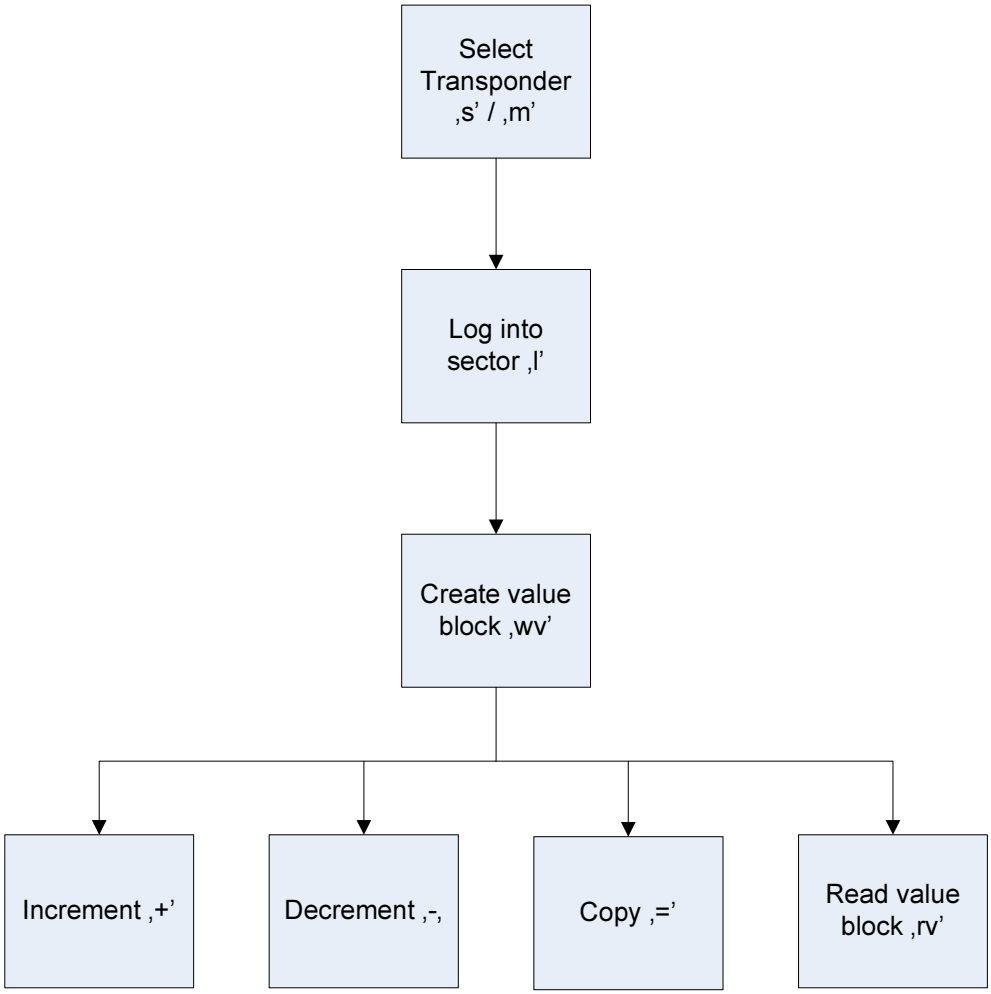
A value block contains 4 bytes user data, the remaining 12 bytes are processed internally by the Mifare® transponder for increased data integrity.

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	$Value$				$\overline{Value}$				$Value$				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:



6.4.4.1 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). 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)

6.4.4.2 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

6.4.4.3 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

6.4.4.4 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

6.4.4.5 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

6.5 Setup-related commands of the reader

6.5.1 Read EEPROM register ‘re’

Use this command to read any internal reader EEPROM register. The register address must be range 00h … 14h.

Command: `'re[EERegAddr]'`

Answer

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

Example

Command	Description
`re04`	Reads register 04h (Station ID)

6.5.2 Write EEPROM register ‘we’

Use this command to write any internal reader EEPROM register. The register address must be range 04h … 14h. A read after write check is performed automatically.

Command: `'we[EERegAddr][Data]'`

Answer

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

Example

Command	Description
`wp0415`	Writes 15h to register 04 (Station ID)

6.5.3 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

6.6 Miscellaneous commands

6.6.1 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)

6.6.2 Antenna power off ‘poff’

Use this command to turn off the HF-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

6.6.3 Antenna power on ‘pon’

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

Command: `'pon'`

Answer	Description
`'P<CR><LF>'`	Power on command performed

6.6.4 Get version ‘v’

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

Command: `'v'`

Answer	Description
`'Mifare 1.11<CR><LF>'`	Version string

6.6.5 Reset ‘x’

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

Command: `'x'`

Answer	Description
`'Mifare 1.11<CR><LF>'`	Version string

6.7 NanoMF related commands

6.7.1 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 Beep tone and length register.

Command: `'b'`

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

6.7.2 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.

6.7.3 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

6.7.4 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 maintain backward compatibility.

Command: **`‘pr’`**

Answer

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

6.7.5 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]’`**

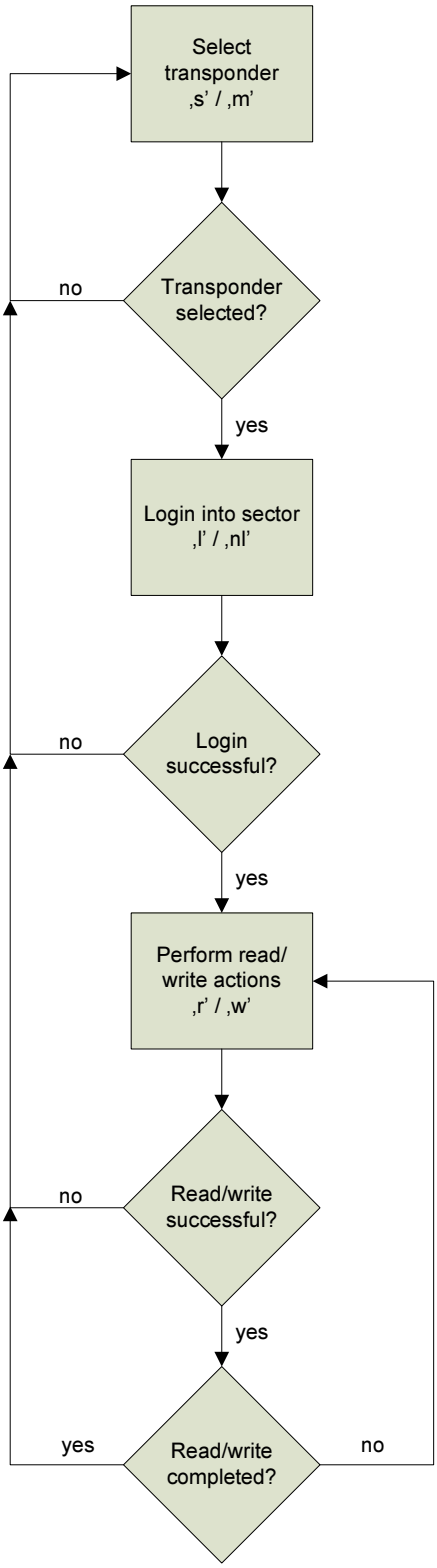
Answer

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

7. Typical data transaction procedure

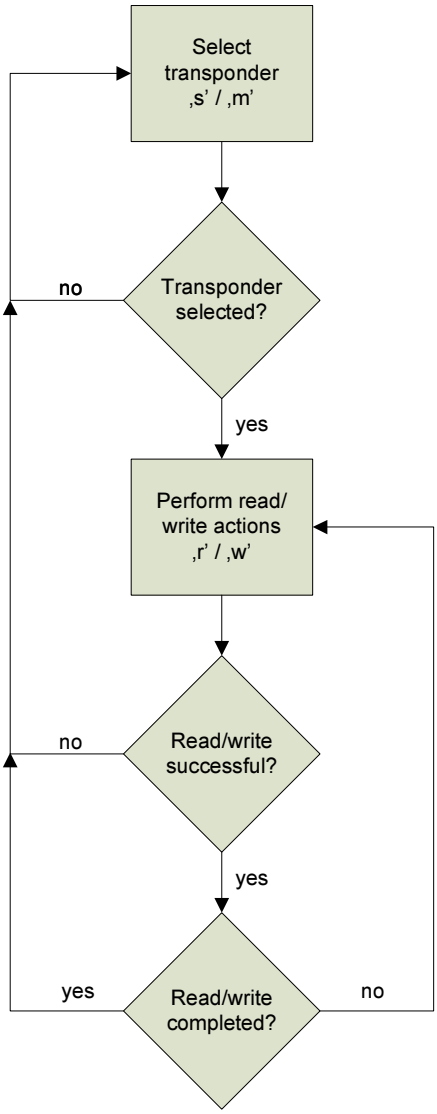
7.1 Mifare® Classic

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



7.2 Mifare® Ultralight

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



8. Memory Organization of Mifare® transponders

8.1 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

8.2 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:

Bytes	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

8.3 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

8.4 Mifare® Ultralight C

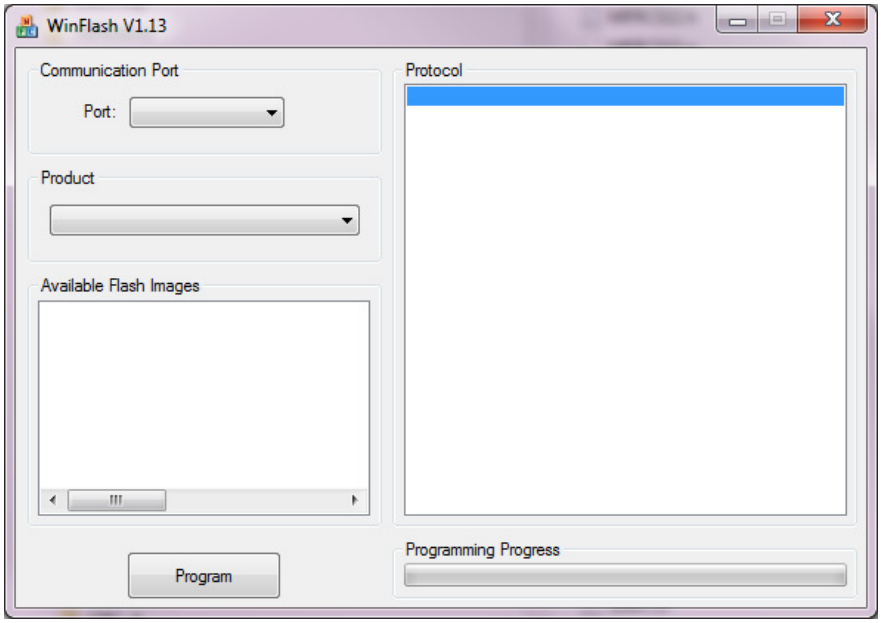
A Mifare® Ultralight C transponder consists of 48 pages with 4 bytes each. The UID is read-only and is also mapped into the memory area. The 16 bytes Triple-DES key is stored in the last 4 pages of the transponder.

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

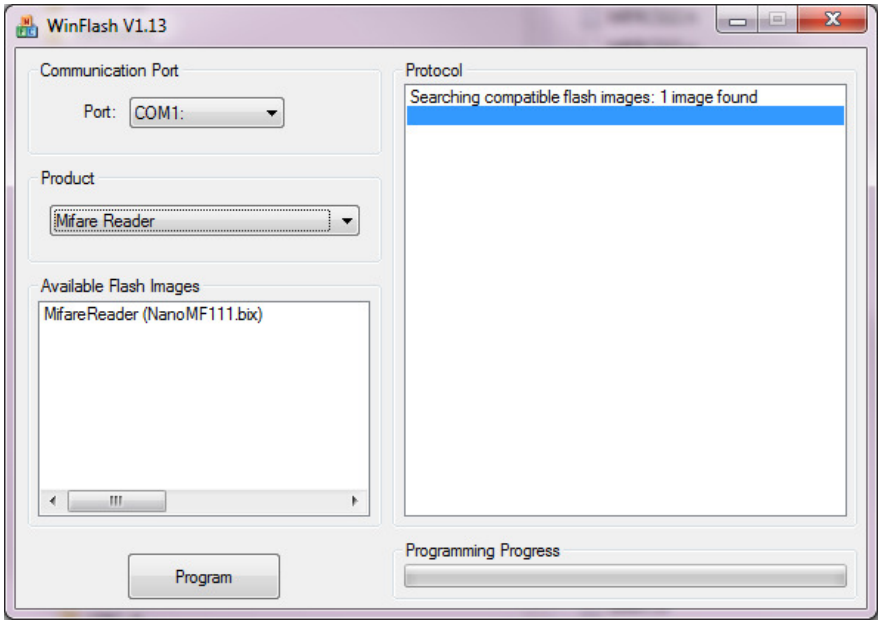
9. Firmware Update

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

- Power up the reader
- On TWN3, the reader must be configured to run in Transparent Mode
- 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 “MifareReader” from *Product*
- Select the Flash Image



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

10. Firmware History

Version	Changes
V1.00	• Initial release
V1.01	• Bootloader included
V1.03	• New commands ‘rv’, ‘wv’, ‘+’, ‘-‘, ‘=’
V1.06	• Bugfix login command
V1.07	• Bugfix write command (‘wb’)
V1.08	• Binary protocol implemented • Support for baudrates 19200, 38400 and 57600 • Baudrate recovery implemented
V1.09	• Improved login: No select tag before sector change required any more • Login: Support for transponders with 7 bytes UID
V1.10	• EE-Register PCON: Bit 'REQA2X' implemented
V1.11	• Bugfix Binary Protocol: the value 0x3F ('?') caused problems

11. Trademarks

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