# TECHNICAL DATA SHEET
# MW-R7B / MW-R7G

MW-R7G and MW-R7B

Documentation version: MW-R7x-MAN-V3
Date: 03.10.2018
Valid from the firmware version: MW-R7x-v1.0.A1.0
Author: Patryk Burczyński, Paweł Winek

SPIS TREŚCI:

**1 Introduction ..... 4**
**2 Technical data ..... 5**
**3 Wires ..... 6**
**4 Input / Output ..... 7**
*4.1 Physical inputs ..... 7*
*4.2 physical outputs ..... 7*
*4.3 RGB LED ..... 7*
*4.4 Sources of SIGNALS controlling outputs ..... 7*
4.4.1 Source „0” and source „1” ..... 8
4.4.2 Source button ..... 8
4.4.3 Source any Card ..... 8
4.4.4 Source RSx ..... 8
4.4.5 Source PinINx ..... 8
4.4.6 Source SIG_Ax ..... 8
4.4.7 Source SIG_Bx ..... 9
4.4.8 Source SIG_Cx ..... 9
**5 Dimensions ..... 10**
**6 Interface ..... 11**
*6.1 Interface 1-WIRE ..... 11*
*6.2 Interface WIEGAND ..... 11*
*6.3 Interface RS232 / RS485 / CAN ..... 12*
6.3.1 Communication protocol for RS232 / 485 ..... 12
6.3.2 Communication protocol for CAN interface ..... 12
**7 Available commands for RS232/RS485/CAN interface ..... 13**
*7.1 Serial interface configuration ..... 13*
7.1.1 Writing serial interface configuration ..... 13
7.1.2 Reading-out serial interface configuration ..... 13
*7.2 Communication orders with transponders ..... 14*
7.2.1 Key management ..... 14
7.2.1.1 writing mifare classic key to the dynamic key memory ..... 15
7.2.1.2 writing mifare classic key to the static key memory ..... 15
7.2.1.3 writing AES / 3DES key to the static key memory ..... 15
7.2.2 Common commands for communication with transponders ..... 16
7.2.2.1 Enabling and disabling reader field ..... 16
7.2.2.2 Selection of one transponder from many ..... 16
7.2.2.3 Get transponder to sleep mode in the field ..... 16
7.2.3 Commands for communication with mifare classic transponders ..... 17
7.2.3.1 Logging into transponder sector using dynamic key ..... 17
7.2.3.2 Logging into transponder sector using static key buffer ..... 17
7.2.3.3 writing content of transponder block ..... 18
7.2.3.4 Copying content of transponder block to another block ..... 18
7.2.3.5 Writing values to transponder block ..... 18
7.2.3.6 Reading-out valueS from transponder block ..... 19
7.2.3.7 Increasing value contained in transponder block ..... 19
7.2.3.8 Decreasing value contained in transponder block ..... 20
7.2.4 Commands for communication for Mifare Ultralight transponders ..... 21
7.2.4.1 Writing page content in Mifare UL ..... 21
7.2.4.2 Reading-out page content in Mifare UL ..... 21
7.2.4.3 Authentication for Ultralight C transponder ..... 21
7.2.5 Commands for communication for Mifare PLUS transponders ..... 22
7.2.5.1 SL0 level commands ..... 22
7.2.5.2 SL1 level commands ..... 22
7.2.5.3 SL3 level commands ..... 23
7.2.5.4 Durations of operations for mifare plus ..... 24
7.2.6 Support for DESFire, DESFire EV1 transponders ..... 25
7.2.6.1 Authorization, logging into the currently selected application ..... 25
7.2.6.2 Change of master key settings of currently selected application ..... 25
7.2.6.3 key change ..... 25

7.2.6.4 Creating application ... 26
7.2.6.5 deleting application ... 26
7.2.6.6 Downloading list of applications ... 27
7.2.6.7 application selection ... 27
7.2.6.8 Transponder formatting ... 27
7.2.6.9 Initialization of transmission protocol with desfire transponder ... 27
7.2.6.10 Downloading list of files of currently selected application ... 28
7.2.6.11 downloading file properties ... 28
7.2.6.12 creating *Standard Data Files* type ... 29
7.2.6.13 creating *Backup Data Files* type ... 29
7.2.6.14 creating Linear/Cyclic Record Files type ... 29
7.2.6.15 deleting file ... 30
7.2.6.16 Change file settings ... 30
7.2.6.17 Reading-out data from *Std/Back Data File* type ... 30
7.2.6.18 Writing data to *Std/Back Data File* type ... 31
7.2.6.19 Writing record to *Record Data File* type ... 31
7.2.6.20 Reading-out record from *Record Data File* type ... 31
7.2.6.21 clearing out *Record Data File* types ... 32
7.2.6.22 Confirmation command - *DesCommit* ... 32
7.2.6.23 Transponder deselection ... 32
7.2.7 I-block data transmission of T=CL ISO14443-4 protocol ... 33
7.2.8 Support for i-code sli family transponders ... 33
7.2.8.1 Reading-out id number of i-code sli transponder ... 33
7.2.8.2 Reading-out sli transponder page ... 33
7.2.8.3 Writing page content in SLI ... 34
7.2.9 Mifare Application Directory - MAD ... 34
7.2.9.1 Formatting MAD cards ... 34
7.2.9.2 adding application to MAD directory ... 34
7.2.9.3 searching for sector for given application ... 35
7.2.9.4 searching for next application sector ... 35
*7.3 electrical sources, inputs and outputs ... 36*
7.3.1 writing Rsx source status ... 36
7.3.2 Reading-out source status ... 36
7.3.3 Writing port configuration ... 36
7.3.4 Reading-out port configuration ... 37
7.3.5 SIG_A block configuration ... 37
7.3.6 SIG_B block configuration ... 38
7.3.7 SIG_C block configuration ... 39
7.3.8 Colour configuration ... 39
*7.4 Access password ... 39*
7.4.1 Logging into the reader ... 39
7.4.2 PASSWORD change ... 40
7.4.3 Logging out from reader ... 40
*1.1 AutoReader mechanism ... 40*
7.4.4 Writing configuration of machine ... 40
7.4.5 Reading-out configuration of machine ... 42
*7.5 Other commands ... 42*
7.5.1 Remote reader reset ... 42
7.5.2 Reading-out software version from reader ... 42
*7.6 Meaning of operation codes in response frames ... 42*
**8 Return to factory settings ... 44**
**9 Bootloader – changing device's firmware ... 46**

# 1 INTRODUCTION

MW-R7x is a wall-mounted reader of RFID cards which works on 125 kHz rated frequency.
Main features:

- Support for transponders: MIFARE® Classic, Plus, Ultralight C, DESFire, I-CODE SLI, iClass (tylko CSN)
- Interfaces:
    - RS232
    - RS485
    - 1-WIRE
    - WIEGAND
    - CAN
- Built-in buzzer,
- Built-in LED RGB of common purpose
- Button
- Available in colors:
    - black (MW-R7B)
    - beige (MW-R7G)

# 2 TECHNICAL DATA

| Transponder type | Reading-out ID number | Full writing and reading-out of memory blocks |
|---|---|---|
| MIFARE® Classic S50 | YES | YES |
| MIFARE® Classic S70 | YES | YES |
| MIFARE® Plus | YES | SL0, SL1, SL3 |
| Ultra Light | YES | YES |
| Ultra Light C | YES | YES |
| DESFire | YES | YES |
| I-CODE SLI | YES | YES |
| iClass | YES (CSN number) | NO |

| MW-R7x reader parameters | |
|---|---|
| Supply voltage | 7-15V |
| Maximal supply currenht | 120 mA |
| Rated operation RF frequency of module | 13,56MHz |
| Reading-out distance of transponders | Up to 8 cm |
| Dimentions (wid.* len. * height) | 44x83x14 mm |
| Button | Yes - capacitive |
| Interfaces | RS-232<br>RS-485<br>Wiegand<br>1-WIRE<br>CAN |
| Input / output | Anticollision (In/Out)<br>PinOUT (Out)<br>PinIN0 (In)<br>PinIN1 (In) |

# 3 WIRES

| Wire | Name | Function |
|---|---|---|
| **Red** | VCC | VCC (+) |
| **Blue** | GND | GND (-) |
| **White** | Anticollision | Output for connecting readers with each other,that are operating closely together |
| **Brown** | PinOUT | Output for any purpose |
| **Green** | PinINTERFACE1 | Serial interface line<br>(RS232_TX, RS485_B, CAN_H, WIEGAND0, 1WIRE) |
| **Yellow** | PinINTERFACE2 | Serial interface line<br>(RS232_RX, RS485_A, CAN_L, WIEGAND1) |
| **Grey** | PinIN0 | Input for any purpose |
| **Pink** | PinIN1 | Input for any purpose |

# 4 INPUT / OUTPUT

## 4.1 PHYSICAL INPUTS

MW-R7x reader has got three physical inputs:

1. PinIN0
2. PinIN1
3. Button

## 4.2 PHYSICAL OUTPUTS

MW-R7x reader has got six  physical outputs:

1. Kolor0 (RGB LED)
2. Kolor1 (RGB LED)
3. Kolor2 (RGB LED)
4. Kolor3 (RGB LED)
5. Buzzer
6. PinOUT

**NOTE:**
*The active state of the output buzzer locks reading-out transponders.*

## 4.3 RGB LED

MW-R7x reader, using LEDs, can display 4 colours: white, green, red and blue. Colour codes are shown in the table below:

Table 4.1 Colour codes table

| Colour code | Colour |
|---|---|
| **0** | Red |
| **1** | Green |
| **2** | Blue |
| **3** | White |

Assigning a specific colour to the KolorX output can be done with *Colour configuration* command. When determinig which colour is to be displayed, Kolor0 input has the highest priority, Kolor3 input has the lowest priority.

## 4.4 SOURCES OF SIGNALS CONTROLLING OUTPUTS

MW-D7x reader has 18 sources of logic signals. These signals can be used to control outputs. Table below contains a list of all sources and values of signals generated by them.

Table 4.2 Signal sources

| ID | Name | Description |
|---|---|---|
| 0 | „0” | Signal source with value of  0 |
| 1 | „1” | Signal source with value of  1 |
| 2 | Przycisk | Source reflecting the status of button. It has value of 1  when button is pressed and value of 0 otherwise |
| 3 | Każda karta | Source reflecting information about presence of the card in the field. It has value of 1 when the card is the field and value of 0 otherwise |
| 4<br>5<br>6<br>7 | RS_0<br>RS_1<br>RS_2<br>RS_3 | Sources controlled via RS232 serial interface. See *C_WriteOutput* command |
| 8<br>9 | PinIN0<br>PinIN1 | Sources controlled by physical input pin using the INPUT block |

| | | |
|---|---|---|
| 10 | SigA0 | Sources controlled by SIG_Ax block outputs |
| 11 | SigA1 | |
| 12 | SigA2 | |
| 13 | SigA3 | |
| 14 | SigB0 | Sources controlled by SIG_Bx block outputs |
| 15 | SigB1 | |
| 16 | SigB2 | |
| 17 | SigB3 | |
| 18 | SigC0 | Sources controlled by SIG_Cx block outputs |
| 19 | SigC1 | |
| 20 | SigC2 | |
| 21 | SigC3 | |

### 4.4.1 SOURCE „0” AND SOURCE „1”

Signal source „0” has always value of 0, while signal source „1” has the value of 1.

### 4.4.2 SOURCE BUTTON

Source reflecting status of button. It has got value 1 when the button is pressed and value 0 otherwise.
NOTE:
If the button is pressed for more than 3 minutes, the button will be recalibrated and the source value reset to zero.

### 4.4.3 SOURCE ANY CARD

Source reflecting information about the presence of card in the field of the reader. It has value 1 when the card is in the field and value 0 in the opposite case.

### 4.4.4 SOURCE RSX

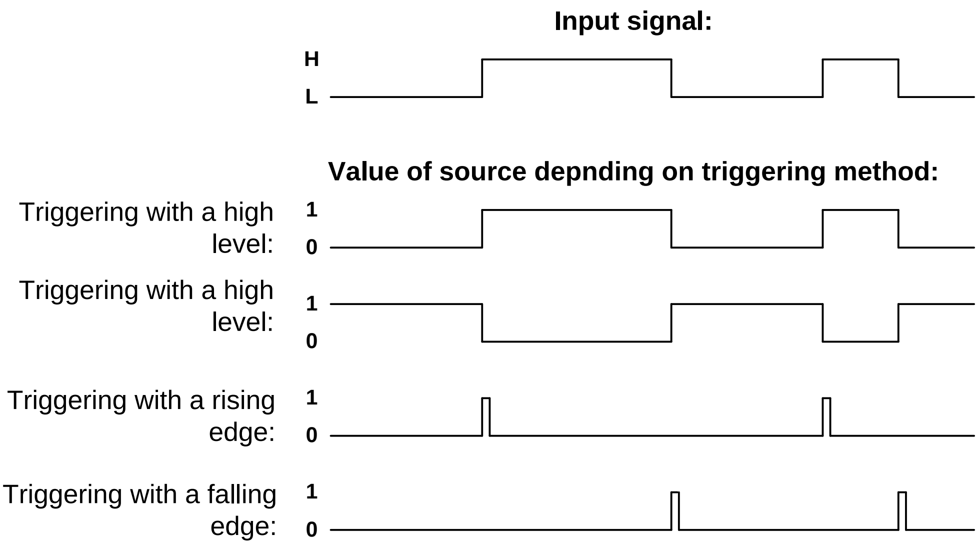
Sources controlled via RS232 serial interface. Source enables:

- Setting value 0
- Setting value 1
- Setting value 1 to a specified time, after which source will automatically change state to 0

See *C_WriteOutput* command

### 4.4.5 SOURCE PININX

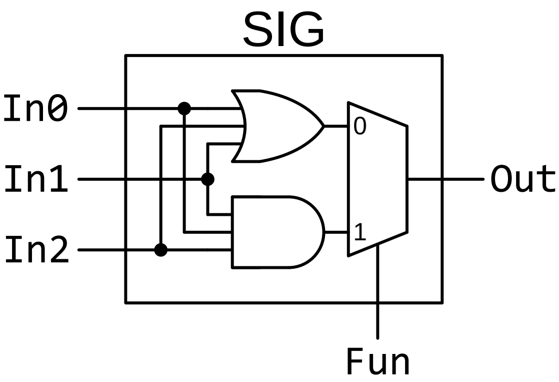
PinINx sources are controlled through physical inputs. Depending on configuration, source has value:


Input signal:
H
L
Value of source depnding on triggering method:
Triggering with a high level:
1
0
Triggering with a high level:
1
0
Triggering with a rising edge:
1
0
Triggering with a falling edge:
1
0

Configuration of trigger method is done using *C_WriteIOConfig* command.

### 4.4.6 SOURCE SIG_AX

MW-R7X reader has 4 virtual SIG_A blocks that allow you to perform logical operations on signals. Each block has 3 signal outputs, one function selection input and one output. Any signal source can be connected to the signal inputs of blocks. At the block output, depending on the *Fun* function selected, there will be a logical sum or logical product of input signals. The SIG blocks are configured using the *SIG_A block configuration* command.

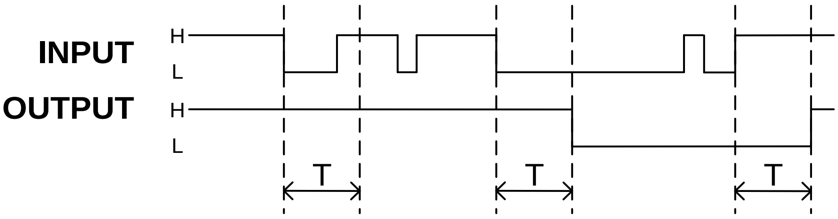


### 4.4.7 SOURCE SIG_BX

MW-D7X reader has 4 virtual SIG_B blocks that allow you to perform logical operations on signals. SIG_B block configuration is done using *SIG_B block configuration* command.

### 4.4.8 SOURCE SIG_CX

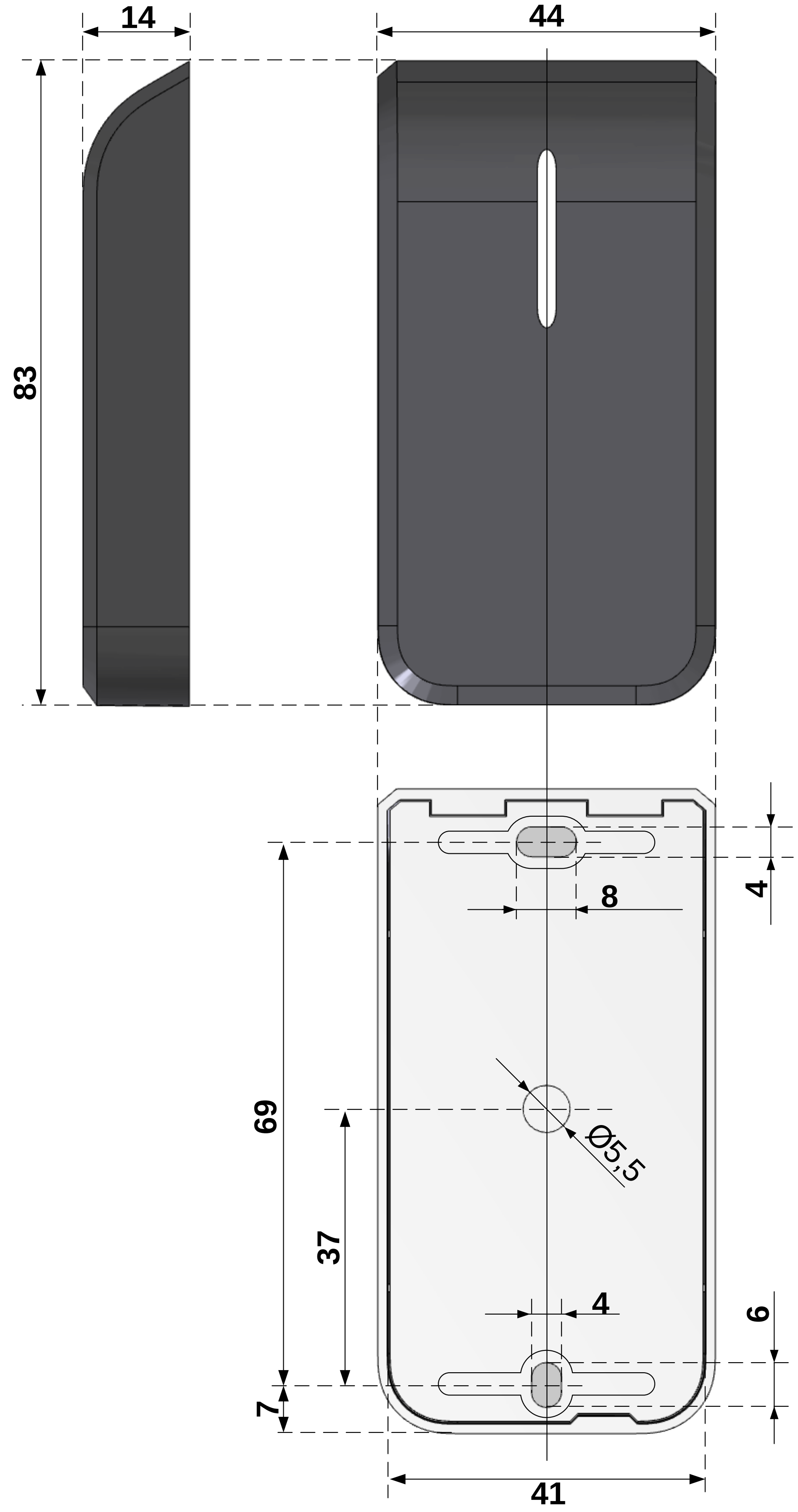
MW-R7x reader has 4 virtual SIG_C blocks that allow you to filter logic signals. The state at the SIG_C output will change to the same as at the input if the input state remains constant for the time defined by the Time parameter. The SIG_C blocks are configured using the *SIG_C Configuration block* command.



**4.1 Sample input and output waveform for the SIG_C block.**

## 5 DIMENSIONS

Dimensions of the reader are shown in the figure below:



Cable lenght: 30cm

# 6 INTERFACE

MW-R7x reader supports 5 serial interfaces:

- RS232
- RS485
- 1-WIRE
- Wiegand
- CAN

RS-232 and RS-485 interfaces are constantly listening in anticipation for the command.
AutoReader sends the read-out ID via the interface selected in the AutoReader configuration.

## 6.1 INTERFACE 1-WIRE

After configuring the device to work in 1-WIRE mode, the reader emulates Dallas DS1990 series of “pills”. As long as the card is in the field, reader will issue a unique number on the 1-WIRE bus. Reader supports READ_ROM oraz SEARCH_ROM commands. Format of sent ID has the form:

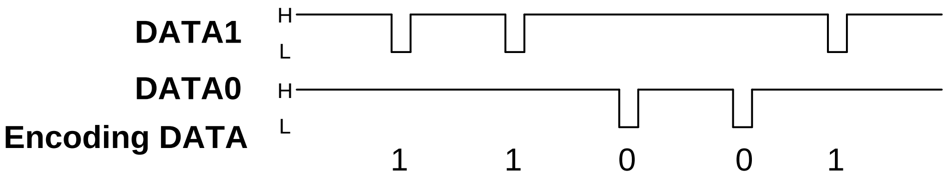
| Family code | Transponder ID | | | | | Address | CRC |
|---|---|---|---|---|---|---|---|
| **ConfFC** | ID0 | ID1 | ID2 | ID3 | ID4 | ConfAdr | xx |

To change a parameter *Address* or *Family code*, please send a *C_SetInterfaceConfig* command to the reader.

## 6.2 INTERFACE WIEGAND

Reader, after being configured to operate in WIEGAND mode, sends a unique ID number of the read card in accordance with the Wiegand protocol with the following parameters:

| | |
|---|---|
| Pulse duration (L level) | 100us |
| Interval between impulses (H level) | 1ms |



MW-R7x reader allows you to change the length of the WIEGAND frame and to select the part of the ID of the card to be sent on the bus.
Exmaples:
ID cards = 0x123456789A = 0b0001001000110100010101100111100010011010

| WIEGAND para-meters | Card ID / responding WIEGAND frame | |
|---|---|---|
| **P1=26, P2=0** | 0b**000100100011010001010110**0111100010011010<br>P**000100100011010001010110**N | Card ID<br>WIEGAND frame |
| **P1=37, P2=0** | 0b**00010010001101000101011001111000100**11010<br>P**00010010001101000101011001111000100**N | Card ID<br>WIEGAND frame |
| **P1=26, P2=1** | 0b0001001000110100**010101100111100010011010**<br>P**010101100111100010011010**N | Card ID<br>WIEGAND frame |

*P,N* – bity parzystości

Another format e.g. WIEGAND, can be obtained by changing the configuration using *C_SetInterfaceConfig* command.

## 6.3 INTERFACE RS232 / RS485 / CAN

MW-R7x reader monitors commands sent via the interface:

- RS232
- RS485/CAN (depends on configuration)

### 6.3.1 COMMUNICATION PROTOCOL FOR RS232 / 485

The NETRONIX protocol is used for communication via the RS232 / RS485 interface.
In this documentation, the description of the protocol has been limited to the description of orders, responses and their parameters. The headline and CRC checksum are always present and are consistent with the full “Netronix Protocol” documentation.

Command frame:

| | | | |
|---|---|---|---|
| header | C_CommandName | Command_parameters1…n | CRC |

Command frame:

| | | | | |
|---|---|---|---|---|
| header | C_CommandName +1 | Command_parameters1…m | OperationCode | CRC |

**NOTES:**
*1. NETRONIX protocol operation can be tested using the tool, free software "FRAMER."*
*2. To configure the device, you can use free software NEFIG.*

### 6.3.2 COMMUNICATION PROTOCOL FOR CAN INTERFACE

When communicating via the CAN serial interface, an intermediary layer is used, enabling the transmission of frames compatible with the NETRONIX protocol using data frames in CAN 2.0B formations.

Full specification of the protocol used for communication via the CAN interface can be found in the documentation "CAN NX 0 protocol".

***For communication using the CAN interface, the manufacturer recommends use of COTER-ED converters. These converters have implemented an intermediate layer, which makes it possible to communicate with MW-R7x devices in the same way as using the RS485 interface.***

# 7 AVAILABLE COMMANDS FOR RS232/RS485/CAN INTERFACE

## 7.1 SERIAL INTERFACE CONFIGURATION

### 7.1.1 WRITING SERIAL INTERFACE CONFIGURATION

Command frame:

| | |
|---|---|
| C_SetInterfaceConfig | P0, P1, P2, [P3] |

Where:

| Parameter name | Opis parametru | Value range |
|---|---|---|
| C_SetInterfaceConfig | Command for changing the serial interface settings | 0x54 |
| P0 | Interface type | 0 – RS232<br>1 – RS485<br>2 – 1-WIRE<br>3 – WIEGAND<br>4 – CAN |
| P1, P2, [P3] | Parameters depending on P0 field value:<br>**For Typ=0**<br>P1 – Logical address (RS232)<br>P2 – Transmission speed (RS232)<br><br>**For Typ=1 or Typ=4**<br>P1 – Logical address (RS485 / CAN)<br>P2 – Transmission speed (RS485)<br>[P3] – Optional parameter. RS485/CAN interface switch<br><br>**For Typ=2**<br>P1 – ConfAdr (7th byte of Dallas frame)<br>P2 – ConfFC (1st byte of Dallas frame)<br><br>**Dla Typ=3**<br>P1 – Number of bits<br>P2 – L/M. This switch determines which part of the card ID will be sent in the WIEGAND frame | <br><br>P1: 0x01 - 0xFE<br>P2: – See Tabela 7.3<br><br><br>P1: 0x01 - 0xFE<br>P2: – See Tabela 7.3<br>P3: 0 – RS485 (default value), 1 - CAN<br><br><br>P1: 0x00-0xFF<br>P2: 0x00-0xFF<br><br><br>P1: 26 - 37<br>P2: 0-1 |

**Tabela 7.3 RS232 interface speed**

| ID | Prędkość |
|---|---|
| **0** | 1200 bps |
| **1** | 2400 bps |
| **2** | 4800 bps |
| **3** | 9600 bps |
| **4** | 19200 bps |
| **5** | 38400 bps |
| **6** | 57600 bps |
| **7** | 115200 bps |

Command frame:

| | | |
|---|---|---|
| C_SetInterfaceConfig +1 | | OperationCode |

### 7.1.2 READING-OUT SERIAL INTERFACE CONFIGURATION

Command frame:

| | |
|---|---|
| C_GetInterfaceConfig | P0 |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_GetInterfaceConfig | Command for reading-out serial interface settings | 0x56 |
| P0 | Type of interface whose configurations we want to read | 0 – RS232<br>1 – RS485<br>2 – 1-WIRE<br>3 – WIEGAND<br>4 – CAN |

Response frame:

| | |
|---|---|
| C_GetInterfaceConfig+1 | P0, P1, P2 |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_GetInterfaceConfig+1 | Command for reading-out serial interface settings | 0x57 |
| P0 | Interface type | 0 – RS232<br>1 – RS485<br>2 – 1-WIRE<br>3 – WIEGAND<br>4 – CAN |
| P1, P2, [P3] | Parameters depending on P0 field value:<br>**For Typ=0**<br>P1 – Logical address (RS232)<br>P2 – Transmission speed (RS232)<br><br>**For Typ=1 or Typ=4**<br>P1 – Logical address (RS485 / CAN)<br>P2 – Transmission speed (RS485)<br>[P3] – Optional parameter. RS485/CAN interface switch<br><br>**For Typ=2**<br>P1 – ConfAdr (7th byte of Dallas frame)<br>P2 – ConfFC (1st byte of Dallas frame)<br><br>**Dla Typ=3**<br>P1 – Number of bits<br>P2 – L/M. This switch determines which part of the card ID will be sent in the WIEGAND frame | <br><br>P1: 0x01 - 0xFE<br>P2: – See Tabela 7.3<br><br><br>P1: 0x01 - 0xFE<br>P2: – See Tabela 7.3<br>P3: 0 – RS485 (default value), 1 - CAN<br><br><br>P1: 0x00-0xFF<br>P2: 0x00-0xFF<br><br><br>P1: 26 - 37<br>P2: 0-1 |

## 7.2 COMMUNICATION ORDERS WITH TRANSPONDERS

### 7.2.1 KEY MANAGEMENT

Key management comes down to saving keys to the internal key memory.
These keys can not be read-out for security purposes. There are two memory areas, separately for Mifare Classic card keys, separately for AES128bits and 3DES keys.

In order to maintain the highest data security, there is a correct philosophy of working with keys.
It consists in writing keys by individuals or persons having the highest degree of trust. Such a record is made only once or very rarely. The operation of reader in a specific application consists not in using the key directly but in calling the appropriate key number in order to log in to the sector. In this way, the key does not actually appear on the data bus in a particular application.

In addition, user should ensure that the key has appropriate access rights to the sectors. This is done through the card initialization process, where new secret keys are written to the cards along with the appropriate access rights assigned to these keys.

Each transponder sector is assigned to key A and key B.
The C_LoadKeyToSKB and C_LoadKeyToDKB commands write Mifare Classic keys to the reader's memory without information what kind of key is (A or B). The C_DesSaveKey command is used to write 3DES / AES key (details in the Mifare Plus chapter)
When logging in to the sector, the user must provide as parameter 0xAA or 0xBB if he wants the called key to be treated as A or as B.

### 7.2.1.1 WRITING MIFARE CLASSIC KEY TO THE DYNAMIC KEY MEMORY

Pamięć dynamiczna charakteryzuje się samoczynnym kasowaniem jej zawartości w przypadku zaniku zasilania. Jej zawartość można wielokrotnie nadpisywać.

Command frame:

| | | | |
|---|---|---|---|
| header | C_LoadKeyToDKB | Key1...6 | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_LoadKeyToDKB | Zapis klucza do dynamicznej pamięci kluczy | 0x14 |
| Key1…6 | 6 bajtowy klucz | dowolne |

Ramka odpowiedzi:

| | | | | |
|---|---|---|---|---|
| nagłówek | C_LoadKeyToDKB +1 | | KodOperacji | CRC |

### 7.2.1.2 WRITING MIFARE CLASSIC KEY TO THE STATIC KEY MEMORY

Static memory is characterized by not deleting its contents in case of a power failure. Its content can be overwritten many times.

Ramka rozkazu:

| | | | |
|---|---|---|---|
| nagłówek | C_LoadKeyToSKB | Key1...6, KeyNo | CRC |

Gdzie:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_LoadKeyToSKB | Writing key to static key memory | 0x16 |
| Key1…6 | 6-byte key | any |
| KeyNo | Key number.<br>Reader can store up to 32 different keys. | 0x00…0x1f |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_LoadKeyToSKB +1 | | OperationCode | CRC |

### 7.2.1.3 WRITING AES / 3DES KEY TO THE STATIC KEY MEMORY

Static memory is characterized by not deleting its contents in case of a power failure. Its content can be overwritten many times.

Command frame:

| | | | |
|---|---|---|---|
| header | C_DesSaveKey | KeyNo, Key0..Key15 | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_ DesSaveKey | Writing key to static key memory | 0x38 |
| KeyNo | Key number.<br>Reader can store up to 32 different keys. | 0x00…0x1f |

| | |
|---|---|
| Key0..Key15 | 16-byte key |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_ DesSaveKey +1 | | OperationCode | CRC |

## 7.2.2 COMMON COMMANDS FOR COMMUNICATION WITH TRANSPONDERS

### 7.2.2.1 ENABLING AND DISABLING READER FIELD

Command frame:

| | |
|---|---|
| C_TurnOnAntennaPower | State |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_TurnOnAntennaPower | Enabling and disabling reader field | 0x10 |
| State | State | 0x00 – disabling field<br>0x01 – enabling field |

Response frame:

| | | |
|---|---|---|
| C_TurnOnAntennaPower +1 | | OperationCode |

### 7.2.2.2 SELECTION OF ONE TRANSPONDER FROM MANY

Command frame:

| | |
|---|---|
| C_Select | |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_Select | Reading-out ID | 0x12 |

Response frame:

| | | |
|---|---|---|
| C_Select +1 | Coll, TType, ID1…….IDn | OperationCode |

Where:

| Parameter name | Parameter description | meaning |
|---|---|---|
| Coll | Collision information (only HITAG transponders) | 0 – no collision<br>1 – collision of two or more transponders |
| TType | Information about the type of transponder from which the read ID number comes from | 1 - Unique,Q5<br>3 - HITAG<br>4 - HID |
| ID1…IDn | Unique transponder number | ID1 – LSB,<br>IDn – MSB |

### 7.2.2.3 GET TRANSPONDER TO SLEEP MODE IN THE FIELD

To get transponder to sleep mode, it must be previously selected.

Command frame:

| | | | |
|---|---|---|---|
| header | C_Halt | | CRC |

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_Halt | Put transponder to sleep mode  in the field | 0x40 |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_Halt+1 | | OperationCode | CRC |

### 7.2.3 COMMANDS FOR COMMUNICATION WITH MIFARE CLASSIC TRANSPONDERS

#### 7.2.3.1 LOGGING INTO TRANSPONDER SECTOR USING DYNAMIC KEY

In order for the login to be successful, it is necessary after each activation of the reader, to reload the Dynamic Key Buffer.

Command frame:

| | | | |
|---|---|---|---|
| header | C_LoginWithDKB | SectorNo, KeyType, DKNo | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_LoginWithDKB | Logging into sector | 0x18 |
| SectorNo | Transponder sector number to which the user wants to log in | **Numberingofblocksandsectors |
| KeyType | Key type that is contained in the internal Dynamic Key Buffer | 0xAA – A type key<br>0xBB – B type key |
| DKNo | Dynamic Key Number | 0x00 |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_LoginWithDKB +1 | | OperationCode | CRC |

#### 7.2.3.2 LOGGING INTO TRANSPONDER SECTOR USING STATIC KEY BUFFER

In order for the login to be successful, it is necessary to load the Static Key Buffer in advance.

Command frame:

| | | | |
|---|---|---|---|
| header | C_LoginWithSKB | SectorNo, KeyType, SKNo | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_LoginWithSKB | Logging into sector | 0x1a |
| SectorNo | Transponder sector number to which the user wants to log in | **Numberingofblocksandsectors |
| KeyType | Key type that is contained in the internal Dynamic Key Buffer | 0xAA –A type key<br>0xBB – B type key |
| SKNo | Static Key Number | 0x00…0x1F |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_LoginWithSKB +1 | | OperationCode | CRC |

Reading-out contents of transponder block

Command frame:

| | | | |
|---|---|---|---|
| header | C_ReadBlock | BlockNo | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_ReadBlock | Reading-out content of transponder block | 0x1e |
| BlockNo | Block number within a given sector | **Numberingofblocksandsectors |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_ReadBlock +1 | Data1..... Data16 | OperationCode | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| Data1…. Data16 | Data read-out from transponder block | |

### 7.2.3.3 WRITING CONTENT OF TRANSPONDER BLOCK

Command frame:

| | | | |
|---|---|---|---|
| header | C_WriteBlock | BlockNo, Data1..... Data116 | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_WriteBlock | Writing content of transponder block | 0x1c |
| BlockNo | Block number within a given sector | **Numberingofblocksandsectors |
| Data1…. Data16 | Dane jakie mają być zapisane w bloku transpondera | any |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_WriteBlock +1 | | OperationCode | CRC |

### 7.2.3.4 COPYING CONTENT OF TRANSPONDER BLOCK TO ANOTHER BLOCK

Command frame:

| | | | |
|---|---|---|---|
| header | C_CopyBlock | SourceBlockNo, TargetBlockNo | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_CopyBlock | Copying content of transponder block to another block | 0x60 |
| SourceBlockNo | Source block | **Numberingofblocksandsectors |
| TargetBlockNo | Target block for data | |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_CopyBlock +1 | | | CRC |

### 7.2.3.5 WRITING VALUES TO TRANSPONDER BLOCK

Command frame:

| | | | |
|---|---|---|---|
| header | C_WriteValue | BlockNo, BackupBlockNo,Value1...4, | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_WriteValue | Writing values to transponder block | 0x34 |
| BlockNo | Block number within a given sector in which the Value will be written | **Numberingofblocksandsectors |
| BackupBlockNo | Declared block number containing a copy of Value<br>BackupBlockNo does not have a significant impact on the operation of system and user can/should make a copy of Value. | **Numberingofblocksandsectors |
| Value1...4 | Value is written to transponder block | any |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_WriteValue +1 | | OperationCode | CRC |

### 7.2.3.6 READING-OUT VALUES FROM TRANSPONDER BLOCK

Command frame:

| | | | |
|---|---|---|---|
| header | C_ReadValue | BlockNo | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_ReadValue | Reading-out values from transponder block | 0x36 |
| BlockNo | Block number within a given sector from which Value will be read-out | **Numberingofblocksandsectors |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_ReadValue+1 | Value1...4, BackupBlockNo | OperationCode | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| Value1...4 | Read-out value from transponder block | |
| BackupBlockNo | Block number that may contain a copy of Value | **Numberingofblocksandsectors |

### 7.2.3.7 INCREASING VALUE CONTAINED IN TRANSPONDER BLOCK

In order to execute command, data must be in the "Value" format.

Command frame:

| | | | |
|---|---|---|---|
| header | C_IncrementValue | BlockNo, Value1...4 | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_IncrementValue | Increasing value contained in transponder block | 0x30 |
| BlockNo | Block number within a given sector in which the Value will be modified | **Numberingofblocksandsectors |
| Value1...4 | Value added to existing real value of transponder block | |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_IncrementValue +1 | | OperationCode | CRC |

### 7.2.3.8 DECREASING VALUE CONTAINED IN TRANSPONDER BLOCK

In order to execute command, data must be in the "Value" format.

Command frame:

| | | | |
|---|---|---|---|
| header | C_DecrementValue | BlockNo, Value1...4 | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_DecrementValue | Decreasing value contained in transponder block | 0x32 |
| BlockNo | Block number within a given sector in which the Value will be modified | **NumeracjaBlokówISektorów |
| Value1...4 | Value subtracted from existing real value of transponder block | dowolna |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_DecrementValue+1 | | OperationCode | CRC |

## 7.2.4 COMMANDS FOR COMMUNICATION FOR MIFARE ULTRALIGHT TRANSPONDERS

### 7.2.4.1 WRITING PAGE CONTENT IN MIFARE UL

Command frame:

| | | | |
|---|---|---|---|
| header | C_WritePage4B | PageAdr, Data1...4 | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_WritePage4B | Writing page content in Mifare UL | 0x26 |
| PageAdr | Page number in transponder | 0x00…0x0f |
| Data1...4 | Data meant to be written | any |
| | | |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_WritePage4B +1 | | OperationCode | CRC |

### 7.2.4.2 READING-OUT PAGE CONTENT IN MIFARE UL

Command frame:

| | | | |
|---|---|---|---|
| header | C_ReadPage16B | PageAdr | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_ReadPage16B | Reading-out page content in Mifare UL | 0x28 |
| PageAdr | Page address from which the next 4 pages should start reading-out. If PageAdr> 0x ???? this will read-out pages at the beginning of memory | 0x00…0x0f |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_ReadPage16B +1 | Data1...16 | OperationCode | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| Data1…16 | Read-out data from 4 consecutive pages. | any |

### 7.2.4.3 AUTHENTICATION FOR ULTRALIGHT C TRANSPONDER

NOTE! Authentication is possible only after the keys have been written in the transponder's memory.

Command frame:

| | | | |
|---|---|---|---|
| header | C_ULC_Auth | KeyIdx | CRC |

Gdzie:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_ULC_Auth | | 0x3C |
| KeyIdx | Key index written in reader | 0x00…0x1f |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_ ULC_Auth +1 | | OperationCode | CRC |

## 7.2.5 COMMANDS FOR COMMUNICATION FOR MIFARE PLUS TRANSPONDERS

### 7.2.5.1 SL0 LEVEL COMMANDS

#### 7.2.5.1.1 WRITE PERSO –CARD INITIALIZATION

Command frame:

| | | | |
|---|---|---|---|
| header | C_MfPlusCMD | 0xA8, AdrH, AdrL, Data{0..15} | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_MfPlusCMD | MFPlus support command | 0x3A |
| 0xA8 | Subcommand ‘Write Perso’ | |
| AdrH, AdrL | Two-byte block number or key to be written | According to MFPLUS Transponder documentation |
| Data{0..15} | Key or data to be written | any |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_ MfPlusCMD +1 | | OperationCode | CRC |

#### 7.2.5.1.2 COMMIT PERSO – MOVE TO NEXT LEVEL OF SL

Command frame:

| | | | |
|---|---|---|---|
| header | C_MfPlusCMD | 0xAA | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_MfPlusCMD | MFPlus support command | 0x3A |
| 0xAA | Subcommand ‘Commit Perso’ | |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_ MfPlusCMD +1 | | OperationCode | CRC |

### 7.2.5.2 SL1 LEVEL COMMANDS

At this level, the Mifare Plus transponder is compatible with the Mifare Classic transponder. All commands related to Mifare Classic support are available, additionally the AES authentication functionality has been implemented.

#### 7.2.5.2.1 SL1 AUTHENTICATION

Command frame:

| | | | |
|---|---|---|---|
| header | C_MfPlusCMD | 0x10, KeyIdx | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_MfPlusCMD | MFPlus support command | 0x3A |
| 0x10 | Subcommand ‘Authentication SL1’ | |
| KeyIdx | Index of AES key written in reader | 0x00-0x1F |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_ MfPlusCMD +1 | | OperationCode | CRC |

#### 7.2.5.2.2 MOVE TO A HIGHER LEVEL OF SL/CHECK AUTHENTICITY OF TRANSPONDER

Moving to a higher SL level or checking the authenticity follows the correct AES authorization with the appropriate key identifier.

Command frame:

| | | | |
|---|---|---|---|
| header | C_MfPlusCMD | 0x70, AdrH, AdrL, KeyIdx | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_MfPlusCMD | MFPlus control command | 0x3A |
| 0x70 | Subcommand ‘First Auth’ | |
| AdrH, AdrL | Two-byte block number or key to write | 0x9002 – transition to level SL2<br>0x9003 – transition to level SL3<br>0x8000 – checking authenticity of transponder |
| KeyIdx | Index of AES  key written in reader | 0x00-0x1F |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_ MfPlusCMD +1 | | OperationCode | CRC |

## 7.2.5.3 SL3 LEVEL COMMANDS

### 7.2.5.3.1 IMPLEMENTING TRANSPONDER INTO ISO14443-4 MODE

Each command associated with SL3 must be preceded by a one-time entry of the transponder into the ISO14443-4 compliance mode

Command frame:

| | | | |
|---|---|---|---|
| header | C_Init_ISO14443-4 | CID | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_Init_ISO14443-4 | | 0x3E |
| CID | CID Identificator | 0x00 |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_Init_ISO14443-4+1 | | OperationCode | CRC |

### 7.2.5.3.2 LOGGING INTO THE SECTOR

Command frame:

| | | | |
|---|---|---|---|
| header | C_MfPlusCMD | 0x1A, Sector, KeyType, KeyIdx | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_MfPlusCMD | MFPlus support command | 0x3A |
| 0x1A | Subcommand ‘sector login’ | |
| Sector | Sector number | 0x00-0x1f – card Plus 2K<br>0x00-0x27 – card Plus 4k |
| KeyType | Key type | 0xAA – klucz A<br>0xBB – klucz B |
| KeyIdx | Index of AES key written in reader | 0x00-0x1F |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_ MfPlusCMD +1 | | OperationCode | CRC |

### 7.2.5.3.3 READING-OUT CONTENT OF TRANSPONDER

Command frame:

| | | | |
|---|---|---|---|
| header | C_ MfPlusCMD | read_cmd, block | CRC |

Where:

<table>
<tr><th>Parameter name</th><th>Parameter description</th><th>Value range</th></tr>
<tr><td>C_MfPlusCMD</td><td>MFPlus support command</td><td>0x3A</td></tr>
<tr><td>read_cmd</td><td>Read-out procedure type:
<table>
<tr><th>cmd.</th><th>MAC on command</th><th>MAC on resonse</th><th>Plain /en-crypted</th></tr>
<tr><td>0x30</td><td>Yes</td><td>No</td><td>Encrypted*</td></tr>
<tr><td>0x31</td><td>Yes</td><td>Yes</td><td>Encrypted*</td></tr>
<tr><td>0x32</td><td>Yes</td><td>No</td><td>Plan</td></tr>
<tr><td>0x33</td><td>Yes</td><td>Yes</td><td>Plan</td></tr>
</table>
</td><td>0x30-0x33</td></tr>
<tr><td>block</td><td>Block number for reading-out</td><td>0-3 for sectors<32<br>0-15 for sectors>32</td></tr>
</table>

*only Plus X transponders

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_ MfPlusCMD +1 | Data1….. Data16 | OperationCode | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| Data1…. Data16 | Data read-out from transponder block | |

### 7.2.5.3.4 WRITING CONTENT OF TRANSPONDER BLOCK

Command frame:

| | | | |
|---|---|---|---|
| header | C_ MfPlusCMD | write_cmd, block, data0..data15 | CRC |

Gdzie:

<table>
<tr><th>Parameter name</th><th>Parameter description</th><th>Value range</th></tr>
<tr><td>C_MfPlusCMD</td><td>MFPlus support command</td><td>0x3A</td></tr>
<tr><td>write _cmd</td><td>Writing procedure type:
<table>
<tr><th>cmd.</th><th>MAC on command</th><th>MAC on resonse</th><th>Plain /en-crypted</th></tr>
<tr><td>0xA0</td><td>Yes</td><td>No</td><td>Encrypted*</td></tr>
<tr><td>0xA1</td><td>Yes</td><td>Yes</td><td>Encrypted*</td></tr>
<tr><td>0xA2</td><td>Yes</td><td>No</td><td>Plain</td></tr>
<tr><td>0xA3</td><td>Yes</td><td>Yes</td><td>Plain</td></tr>
</table>
</td><td>0xA0-0xA3</td></tr>
<tr><td>block</td><td>Block number to read-out</td><td>0-3 for sectors<32<br>0-15 for sectors>32</td></tr>
<tr><td>data0..data15</td><td>Data for writing transponder block</td><td></td></tr>
</table>

*only Plus X transponders

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_ MfPlusCMD +1 | | OperationCode | CRC |

## 7.2.5.4 DURATIONS OF OPERATIONS FOR MIFARE PLUS

Following specification defines the duration of individual operations, counted from the moment of sending command frame (RS) to the moment of sending response frame (RS)

| Operation | Correct result [ms] | Incorrect result [ms] |
|---|---|---|
| SELECT | 14 | 12 |
| LOGIN SL3 | 25 | 100 |
| READ BLOCK | 10 | 100 |
| WRITE BLOCK | 13 | 100 |

### 7.2.6 SUPPORT FOR DESFIRE, DESFIRE EV1 TRANSPONDERS

#### 7.2.6.1 AUTHORIZATION, LOGGING INTO THE CURRENTLY SELECTED APPLICATION

Command frame:

| | | | |
|---|---|---|---|
| header | C_DesAuth (0x42) | KeyNo{0..0x10}, KeyIdx, AuthType | CRC |

Gdzie:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_DesAuth | Authorization command | 0x42 |
| KeyNo | Key number in relation to transponder | 0x00..0x10 |
| KeyIdx | Index of AES key written in reader | 0x00-0x1F |
| AuthType | Authorization type :<br>0x0A – DES<br>0xAA - AES | 0x0A, 0xAA |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_DesAuth +1 | | OperationCode | CRC |

#### 7.2.6.2 CHANGE OF MASTER KEY SETTINGS OF CURRENTLY SELECTED APPLICATION

Command frame:

| | | | |
|---|---|---|---|
| header | C_DesChangeKeySett (0x44) | KeySettings | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_DesChangeKeySett | Command for changing key settings | 0x44 |
| KeySettings | Configurational byte | 0x00..0x0f |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_DesChangeKeySett+1 | | OperationCode | CRC |

Struktura bajtu konfiguracyjnego *KeySettings*:

| Bit | Meaning |
|---|---|
| 0 | 0 – PICC Master key is not modifiable |
| | 1* – PICC Master key is modifiable |
| 1 | 0 – calling C_DesGetAppIDs function requires authorization using PICC Master key |
| | 1* – calling C_DesGetAppIDs does not require authorization |
| 2 | 0 – creating / deleting an application requires authorization using PICC Master key |
| | 1* -creating a new application does not require authorization,<br>removing application requires authorization with key of the given application or PICC Master key |
| 3 | 0 – it is not possible to change the PICC Master Key configuration |
| | 1* - change of PICC Master Key configuration allowed in the case of authorization using this key |
| 4 | RFU – 0 |
| 5 | RFU – 0 |
| 6 | RFU – 0 |
| 7 | RFU – 0 |

* - default setting

#### 7.2.6.3 KEY CHANGE

Command frame:

| | | | |
|---|---|---|---|
| header | C_DesChangeKey (0x46) | KeyNo, NewEESavedKey,[PrevEESavedKey] | CRC |

Gdzie:

| Parameter name | Parameter description | Value range |
|---|---|---|

| | | |
|---|---|---|
| C_DesChangeKey | Key change command | 0x46 |
| KeyNo | Key number to be changed | 0x00..0x0D |
| NewEESavedKey | Index of new key written  in reader's memory | 0x00..0x13 |
| PrevEESavedKey | • If changed key is not the one in which current authorization occurred, we give index of current key that will be changed<br>• If changed key is the same in which current authorization took place, this parameter is left blank | 0x00..0x13 |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_DesChangeKey+1 | | OperationCode | CRC |

### 7.2.6.4 CREATING APPLICATION

Command frame:

| | | | |
|---|---|---|---|
| header | C_DesCreateApp (0x48) | AId1..3,KeySettings1, KeySettings2 | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_DesCreateApp | Creating application command | 0x48 |
| AId1..3 | 3-byte application ID | 0x00..0xFF |
| KeySettings1 | Configurational byte (see below) | 0x00..0x0F |
| KeySettings2 | *Bit3..bit0:*<br>Number of keys assigned to the application<br>*Bit7..Bit6*:<br>00 – DES authorization for entire application<br>10- AES authorization for entire application | 0x00..0x0D |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_DesCreateApp +1 | | OperationCode | CRC |

Struktura bajtu konfiguracyjnego *KeySettings*:

| Bit | Meaning |
|---|---|
| 0 | 1 * - Application Master key is modifiable, requires authorization using existing AppMasterKey key |
| | |
| 1 | 0 – calling C_DesGetAppIDs function requires authorization using PICC Master key |
| | 1* – calling C_DesGetAppIDs does not require authorization |
| 2 | 0 – creating / deleting file requires authorization using AppMasterKey |
| | 1* -creation / deletion of the file does not require authorization using AppMasterKey |
| 3 | 0 – it is not possible to change the configuration of Application Master Key |
| | 1* - change of Application Master Key configuration allowed in case of authorization using this key |
| 4 | Bit7-Bit4: determine rights to change key parameters |
| 5 | 0x0*:Application master key is necessary to change key settings |
| 6 | 0x1-0xD : authorization with a key with this index is necessary to change key settings |
| 7 | 0xE :changing key settings requires authorization using the same key |

* - default setting

### 7.2.6.5 DELETING APPLICATION

Command frame:

| | | | |
|---|---|---|---|
| header | C_DesDeleteApp (0x4a) | AId1..3 | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_DesDeleteApp | Application deletion command | 0x4a |
| AId1..3 | 3-byte application ID | 0x00..0xFF |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_DesCreateApp +1 | | OperationCode | CRC |

### 7.2.6.6 DOWNLOADING LIST OF APPLICATIONS

Command frame:

| | | | |
|---|---|---|---|
| header | C_DesGetAppIDs (0x4c) | | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_DesGetAppIDs | Downloading list of applications command | 0x4c |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_DesGetAppIDs +1 | N*{Aid3,Aid2,Aid1} | OperationCode | CRC |

List of Aid numbers, currently existing applications, is returned

### 7.2.6.7 APPLICATION SELECTION

Command frame:

| | | | |
|---|---|---|---|
| header | C_DesSelectApp (0x4e) | Aid1..3 | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_DesSelectApp | Application selection command | 0x4e |
| Aid1..3 | 3-byte application ID | 0x00-0xff |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_DesSelectApp+1 | | OperationCode | CRC |

### 7.2.6.8 TRANSPONDER FORMATTING

Command frame:

| | | | |
|---|---|---|---|
| header | C_DesFormatPICC (0x70) | | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_DesFormatPICC | Transponder formatting command | 0x60 |

Execution of this command requires authorization using PICC Master key.

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_DesFormatPICC +1 | | OperationCode | CRC |

### 7.2.6.9 INITIALIZATION OF TRANSMISSION PROTOCOL WITH DESFIRE TRANSPONDER

Command frame:

| | | | |
|---|---|---|---|
| header | C_DesInitProtocol (0x3E) | CID | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_DesInitProtocol | Transponder formatting command | 0x3E |
| CID | Logical number of selected transponder | 0x00-0x0E |

This command must appear immediately after selecting the transponder with C_Select command. Current version of reader allows you to work with one Desfire transponder simultaneously. CID logical number does not currently matter, it is recommended to enter the number 0.

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_DesInitProtocol +1 | | OperationCode | CRC |

### 7.2.6.10 DOWNLOADING LIST OF FILES OF CURRENTLY SELECTED APPLICATION

Command frame:

| | | | |
|---|---|---|---|
| header | C_DesGetFileIDs (0x64) | | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_DesGetFileIDs | Downloading list of applications command | 0x64 |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_DesGetAppIDs +1 | N*FileNo | OperationCode | CRC |

List of file numbers ,currently existing in the selected application, is returned.

### 7.2.6.11 DOWNLOADING FILE PROPERTIES

Command frame:

| | | | |
|---|---|---|---|
| header | C_DesGetFileSett (0x66) | FileNo | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_DesGetFileSett | File properties download command | 0x66 |
| FileNo | File ID | 0x00-0x0f |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_DesGetAppIDs +1 | File params... | OperationCode | CRC |

Depending on type of file, information is returned in the following format:

- For *Standard Data Files* and *Backup Data Files*

| 1 byte | 1 byte | 2 bytes | 3 bytes |
|---|---|---|---|
| File type | Comm. Sett. | Access right | File size |
| | | LSB MSB | LSB MSB |

- For *Value Files* (this type is currently not implemented)

| 1 byte | 1 byte | 2 bytes | 4 bytes | 4 bytes | 4 bytes | 1 byte |
|---|---|---|---|---|---|---|
| File type | Comm. Sett. | Access right | Lower limit | Upper limit | Limited credit value | Limited credit enable |
| | | LSB MSB | LSB MSB | LSB MSB | LSB MSB | |

- For *Linear/Cyclic record files*

| 1 byte | 1 byte | 2 bytes | 3 bytes | 3 bytes | 3 bytes |
|---|---|---|---|---|---|
| File type | Comm. Sett. | Access right | Record size | Maximum number of records | Current number of records |
| | | LSB MSB | LSB MSB | LSB MSB | LSB MSB |

### 7.2.6.12 CREATING *STANDARD DATA FILES* TYPE

Command frame:

| | | | |
|---|---|---|---|
| header | C_DesCreateSTDataFile (0x68) | FileNo,ComSett,AccRight1..2,FileSize1..3 | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_DesCreateSTDataFile | Creating STD file command | 0x68 |
| FileNo | File ID | 0..0x0F |
| ComSett | Transmission type:<br>0x01 – unencrypted<br>0x03 – DES encrypted | 0x00,0x03 |
| AccRight1..2 | Access rights to file, see table below | 0x00..0xff |
| FileSize1..3 | 3-byte file size in bytes,<br>in the order of LSB..MSB | 0x00-0xff |

Bytes specifying access rights:

| 15 12 | 11 8 | 7 4 | 3 0 |
|---|---|---|---|
| **Read Access** | **Write Access** | **Read & Write Access** | **Change Right Access** |
| MBS *1st byte* | | *2nd byte* | LSB |

Two bytes of access rights are divided into four 4-bit fields. Each field can contain values from range 0x0 - 0xF

- Values in range 0x0 - 0xD specify key number, which will have the right to perform given operation,
- Value 0xE means that the operation does not require authorization
- Value 0xF means that there is no access to operation, regardless of key used

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_DesCreateSTDataFile +1 | | OperationCode | CRC |

### 7.2.6.13 CREATING *BACKUP DATA FILES* TYPE

Command frame:

| | | | |
|---|---|---|---|
| header | C_DesCreateBACKDataFile (0x6a) | FileNo,ComSett,AccRight1..2,FileSize1..3 | CRC |

Where:

| **Parameter name** | Parameter description | Value range |
|---|---|---|
| C_DesCreateBACKData-File | Command to create a BACKUP file | 0x6a |
| FileNo | File ID | 0..0x07 |
| ComSett | Transmission type:<br>0x01 – unencrypted<br>0x03 – DES encrypted | 0x00,0x03 |
| AccRight1..2 | File access rights | 0x00..0xff |
| FileSize1..3 | 3 byte file size in bytes<br>in order of LSB..MSB | 0x00-0xff |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_DesCreateBACKDataFile +1 | | OperationCode | CRC |

Access rights are defined in the same way as for *Standard Data Files*

Writing *Backup Data file* must end with issuance of C_DesCommit command.

### 7.2.6.14 CREATING LINEAR/CYCLIC RECORD FILES TYPE

Command frame:

| | | | |
|---|---|---|---|
| header | C_DesCreateRecordFile (0x6c) | FileNo, ComSett, AccRight1..2, RecSize1..3, RecNumb1..3, Cy/Li{0x0C,0x01} | CRC |

Where:

| **Parameter name** | Parameter description | Value range |
|---|---|---|
| C_DesCreateRecordFile | *Record File* type creation command | 0x6c |
| FileNo | File ID | 0..0x0F |
| ComSett | Transmision type:<br>0x01 – unecrypted<br>0x03 –DES encrypted | 0x00,0x03 |
| AccRight1..2 | File access rights | 0x00..0xff |
| RecSize1..3 | 3-byte record size in bytes, in order of LSB..MSB | 0x00-0xff |
| RecNumb1..3 | 3-byte parameter specifying number of records, order of LSB..MSB | |
| Cy/Li | 0x0c- cyclical type<br>0x0l – linear type | 0x0C,0x01 |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_DesCreateRecordFile+1 | | OperationCode | CRC |

Access rights are defined in the same way as for *Standard Data Files.*

### 7.2.6.15 DELETING FILE

Command frame:

| | | | |
|---|---|---|---|
| header | C_DesDeleteFile (0x6e) | FileNo | CRC |

Where:

| **Parameter name** | Parameter description | Value range |
|---|---|---|
| C_DesDeleteFile | File deletion command | 0x6e |
| FileNo | File ID | 0x00..0x0F |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_DesDeleteFile+1 | | OperationCode | CRC |

### 7.2.6.16 CHANGE FILE SETTINGS

Command frame:

| | | | |
|---|---|---|---|
| header | C_DesChangeFileSett (0x80) | FileNo, ComSett, AccRight1..2 | CRC |

Where:

| **Parameter name** | **Parameter description** | **Value range** |
|---|---|---|
| C_DesChangeFileSett | Command to change the properties of file | 0x80 |
| FileNo | File ID | 0..0x0F |
| ComSett | Transmision type:<br>0x01 – unecrypted<br>0x03 –DES encrypted | 0x00,0x03 |
| AccRight1..2 | File access rights | 0x00..0xff |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_DesChangeFileSett+1 | | OperationCode | CRC |

Access rights are defined in the same way as for *Standard Data Files.*

### 7.2.6.17 READING-OUT DATA FROM *STD/BACK DATA FILE* TYPE

Command frame:

| | | | |
|---|---|---|---|
| header | C_DesReadData (0x82) | FileNo, Offset1..3, Length1..3 | CRC |

Where:

| **Parameter name** | Parameter description | Value range |
|---|---|---|
| C_DesReadData | Reading-out from file command | 0x82 |
| FileNo | File ID | 0..0x0F |
| Offset1..3 | 3-byte parameter specifying place from which we start to read-out file, order of LSB..MSB | 0x00-0xFF |
| Length1..3 | 3-byte parameter specifying number of bytes to be read-out, order of LSB..MSB<br>(once read-out can be up to 58 bytes) | 0x00-0x3A |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_DesReadData +1 | n Bytes | OperationCode | CRC |

### 7.2.6.18 WRITING DATA TO *STD/BACK DATA FILE* TYPE

Command frame:

| | | | |
|---|---|---|---|
| header | C_DesWriteData (0x84) | FileNo, Offset1..3,Data1..58 | CRC |

Where:

| **Parameter name** | Parameter description | Value range |
|---|---|---|
| C_DesWriteData | Writing file command | 0x84 |
| FileNo | File ID | 0..0x0F |
| Offset1..3 | 3-byte parameter specifying the place from which we start to write, order of LSB..MSB | 0x00-0xFF |
| Data1..58 | Data that we intend to write to a file,<br>(one time you can write up to 58Byte) | 0x00-0xFF |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_DesWriteData+1 | | OperationCode | CRC |

### 7.2.6.19 WRITING RECORD TO *RECORD DATA FILE* TYPE

Command frame:

| | | | |
|---|---|---|---|
| header | C_DesWriteRecord (0x86) | FileNo, Offset1..3,Data1..58 | CRC |

Where:

| **Parameter name** | Parameter description | Value range |
|---|---|---|
| C_DesWriteRecord | Record writing command | 0x86 |
| FileNo | File ID | 0..0x0F |
| Offset1..3 | 3-byte parameter specifying the place from which we start to write, order of LSB..MSB (this value must be smaller than size of a single record) | 0x00-0xFF |
| Data1..58 | Data that we intend to write to a file,<br>(one time you can write up to 58 bytes, the sum of this value and the offset must be smaller than the size of a single record) | 0x00-0xFF |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_DesWriteRecord+1 | | OperationCode | CRC |

Note: Writing a record to a *Record File* type file must end with the issuance of the *C_DesCommit* command.

### 7.2.6.20 READING-OUT RECORD FROM *RECORD DATA FILE* TYPE

Command frame:

| | | | |
|---|---|---|---|
| header | C_DesReadRecord (0x88) | FileNo, WhichRecord1..3,NoOfRecords1..3 | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_DesReadRecord | Reading-out record command | 0x88 |
| FileNo | File ID | 0..0x0F |
| WhichRecord1..3 | 3-byte parameter specifying record from which we start to read-out, order of LSB..MSB | 0x00-0xFF |
| NoOfRecords1..3 | 3-byte parameter specifying number of records to read-out, order of LSB..MSB | 0x00-0xFF |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_DesReadRecord +1 | Record data... | OperationCode | CRC |

Number of read-out data can not be more than 58 bytes, so keep the rule: {NoOfRecords1..3} * size_crumb <58bytes

### 7.2.6.21 CLEARING OUT *RECORD DATA FILE* TYPES

Command frame:

| | | | |
|---|---|---|---|
| header | C_DesClearRecordFile (0x8a) | FileNo | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_DesClearRecordFile | Clearing out record file command | 0x8a |
| FileNo | File ID | 0..0x0F |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_DesClearRecordFile+1 | | OperationCode | CRC |

Note: This operation must end with the issuance of *C_DesCommit* command.

### 7.2.6.22 CONFIRMATION COMMAND - *DESCOMMIT*

Command frame:

| | | | |
|---|---|---|---|
| header | C_DesCommit (0x8c) | | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_DesCommit | Confirmation command | 0x8c |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_DesCommit+1 | | OperationCode | CRC |

### 7.2.6.23 TRANSPONDER DESELECTION

Command frame:

| | | | |
|---|---|---|---|
| header | C_DesDeselect (0x8e) | | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_DesDeselect | Transponder deselection command | 0x8e |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_DesDeselect+1 | | OperationCode | CRC |

### 7.2.7 I-BLOCK DATA TRANSMISSION OF T=CL ISO14443-4 PROTOCOL

This command allows you to send data to the transponder in ISO14443-4 mode, and at the same time returns information from the transponder. Before executing this command, it is necessary to enter ISO14443-4 mode with the command C_Init_ISO14443-4.

Command frame:

| | | | |
|---|---|---|---|
| header | C_TranscIBlock | data | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_TranscIBlock | | 0xC8 |
| data | Data of I-Block package | any |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_TranscIBlock+1 | data | OperationCode | CRC |

### 7.2.8 SUPPORT FOR I-CODE SLI FAMILY TRANSPONDERS

#### 7.2.8.1 READING-OUT ID NUMBER OF I-CODE SLI TRANSPONDER

Command frame:

| | | |
|---|---|---|
| header | C_Inventory | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_Inventory | Reading-out ID number | 0x04 |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_Inventory +1 | 0,CardType,ID1...ID8 | OperationCode | CRC |

#### 7.2.8.2 READING-OUT SLI TRANSPONDER PAGE

Command frame:

| | | | |
|---|---|---|---|
| header | C_SLIReadPage | PageAdr | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_ SLIReadPage | Reading-out content of websites at Mifare UL | 0x2C |
| PageAdr | Website address matches type of supported transponder | |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_ SLIReadPage +1 | Data1...4 | OperationCode | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| Data1…4 | Read-out data | any |

#### 7.2.8.3 WRITING PAGE CONTENT IN SLI

Command frame:

| | | | |
|---|---|---|---|
| header | C_SLIWritePage | PageAdr, Data1...4 | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_ SLIWritePage | Writing page content in SLI | 0x2E |
| PageAdr | Page number in transponder | |
| Data1...4 | Data to be written | any |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_ SLIWritePage +1 | | OperationCode | CRC |

### 7.2.9 MIFARE APPLICATION DIRECTORY - MAD

#### 7.2.9.1 FORMATTING MAD CARDS

Command frame:

| | | | |
|---|---|---|---|
| header | C_FormatMad | Type, Infobyte | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_FormatMad<br>0xa8 | Formatting to MAD | 0xa8 |
| Type | 1 - MAD1 (15sectors)<br>2 – MAD2 (30sectors) | 0x01,0x02 |
| Infobyte | Indicator on sector of issuer (default 0x00) | 0x00-0x1F |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_FormatMad+1 | | OperationCode | CRC |

Notes:

Before you execute the C_FormatMad command, you need to:

- disable AutoReader mode (via the C_SetAutoReaderConfig command)
- load keys (default 0xff, 0xff, 0xff, 0xff, 0xff, 0xff)
- turn on antenna power (via C_TurnOnAntennaPower command)
- select card (via C_Select command)
- log in to sector 0 using an AA type key

#### 7.2.9.2 ADDING APPLICATION TO MAD DIRECTORY

Command frame:

| | | | |
|---|---|---|---|
| header | C_AddApplication | LSB, MSB, Sector | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_AddApplication<br>0xaa | Adding application | 0xaa |
| LSB | less significant byte of application number | 0x00 - 0xFF |
| MSB | more significant byte of application number | 0x00 - 0xFF |
| Sector | Sector number, where application should be located | 0x01-0x0F :MAD1<br>0x01-0x1F :MAD2 |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_AddApplication+1 | | OperationCode | CRC |

Notes:
Application number must be different from 0x0000
Before you execute the C_AddApplication command, you need to:

- disable AutoReader mode (via the C_SetAutoReaderConfig command)
- load keys (default 0xff, 0xff, 0xff, 0xff, 0xff, 0xff)
- turn on antenna power (via C_TurnOnAntennaPower command)
- select card (via C_Select command)
- log in to sector 0 using an AA type key

### 7.2.9.3 SEARCHING FOR SECTOR FOR GIVEN APPLICATION

Command frame:

| | | | |
|---|---|---|---|
| header | C_GetSectorMad | LSB, MSB | CRC |

Where:

| **Parameter name** | Parameter description | Value range |
|---|---|---|
| C_GetSectorMad<br>0xac | Searching for sector | 0xac |
| LSB | less significant byte of application number | 0x00 - 0xFF |
| MSB | more significant byte of application number | 0x00 - 0xFF |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_GetSectorMad+1 | Sector | OperationCode | CRC |

Notes:
Before you execute C_GetSectorMad command, you need to:

- disable AutoReader mode (via the C_SetAutoReaderConfig command)
- load keys (default 0xff, 0xff, 0xff, 0xff, 0xff, 0xff)
- turn on antenna power (via C_TurnOnAntennaPower command)
- select card (via C_Select command)
- log in to sector 0 using an AA type key

If response byte is 0x00, then application is not in the MAD directory.

### 7.2.9.4 SEARCHING FOR NEXT APPLICATION SECTOR

Command frame:

| | | | |
|---|---|---|---|
| header | C_GetSectorMadNext | LSB, MSB | CRC |

Where:

| **Parameter name** | Parameter description | Value range |
|---|---|---|
| C_GetSectorMad<br>0xae | Searching for next sector | 0xae |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_GetSectorMadNext+1 | Sector | OperationCode | CRC |

Notes:
Before you can execute C_GetSectorMadNext command, perform a lookup of the sector with the C_GetSector-Mad command whose search result was different from 0.

If response byte is 0x00, then no more sectors were found for application

## 7.3 ELECTRICAL SOURCES, INPUTS AND OUTPUTS

### 7.3.1 WRITING RSX SOURCE STATUS

Command frame:

| | | | |
|---|---|---|---|
| header | C_WriteSourceRSx | Source, State, [Time] | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_WriteSourceRSx | Writing RSx source status | 0x70 |
| Source | RSx source number. | 0x04-0x07 |
| State | Desired exit status | 0x00 lub 0x01 |
| [Time] | Optional parameter. Time after which RSx source will return to state 0 (x10ms) | 0x00-0xFF |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_WriteSourceRSx +1 | | OperationCode | CRC |

### 7.3.2 READING-OUT SOURCE STATUS

Command frame:

| | | | |
|---|---|---|---|
| header | C_ReadSource | Source | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_ReadSource | Reading-out source status | 0x72 |
| Source | Source | See ID number from Table 4.2 |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_ReadSource +1 | State | OperationCode | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_ReadSource+1 | Reading-out source status | 0x73 |
| State | Source status | 0x04-0x07 |

### 7.3.3 WRITING PORT CONFIGURATION

Command frame:

| | |
|---|---|
| C_SetIOConfig | IONo, Dir, P0 |

Where:

**If we configure port as an output:**

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_SetIOConfig | Writing configuration of any port | 0x50 |
| IONo | IO port number to be configured | 0x00..0x05 |
| Dir | Port direction | 0x00 – output |
| P0 | Source of control signal | See ID number from Table 4.2 |

**If we configure port as an input:**

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_SetIOConfig | Writing configuration of any port | 0x50 |
| IONo | IO port number to be configured | 0x06 – 0x07 |
| Dir | Port direction | 1 – input |
| P0 | Byte specifying the triggering method<br>See the chapter:: 8 PinINx | 0 – not negated<br>1 – negated |

| | | |
|---|---|---|
| | | 2 – reaction to increasing slope<br>3 – reaction to decreasing slope |

Not all MW-R7x ports have any direction. For correct configuration, correct direction should be given for given port.

Table 7.4 List of existing ports that can be controlled in MW-R7x

| Port number | Direction | Description |
|---|---|---|
| 0 | output | Physical output PinOUT |
| 1 | output | KOLOR0 |
| 2 | output | KOLOR1 |
| 3 | output | KOLOR2 |
| 4 | output | KOLOR3 |
| 5 | output | BUZZER |
| 6 | output | Physical input PinIN0 |
| 7 | output | Physical input PinIN1 |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_SetIOConfig +1 | | OperationCode | CRC |

### 7.3.4 READING-OUT PORT CONFIGURATION

Command frame:

| | | | |
|---|---|---|---|
| header | C_GetIOConfig | IONo | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_GetIOConfig | Reading-out configuration of any port | 0x52 |
| IONo | IO port number whose configuration is to be read-out | 0x00…0x07 |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_GetIOConfig +1 | Dir, P0 | OperationCode | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| Dir, P0 | Parameters have the same meaning as C_SetIOConfig command | |

### 7.3.5 SIG_A BLOCK CONFIGURATION

Command frame:

| | | | |
|---|---|---|---|
| header | C_ConfigSIG_A | SigNo, [Function, In0, In1, In2] | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_ConfigSIG_A | SIG_A block read-out / write configuration | 0x5C |
| SigNo | Block number SIG_A, whose configuration is to be read-out / written | 0x00…0x03 |
| Function | Optional parameter - if present, command writes new configuration.<br>Specifies the type of function executed by SIG_A block. | 0 – function OR<br>1 – function AND |
| In1, In2, In3 | Optional parameters - if present, command writes new configuration.<br>Sources of input signals | See ID number from Table 4.2 |

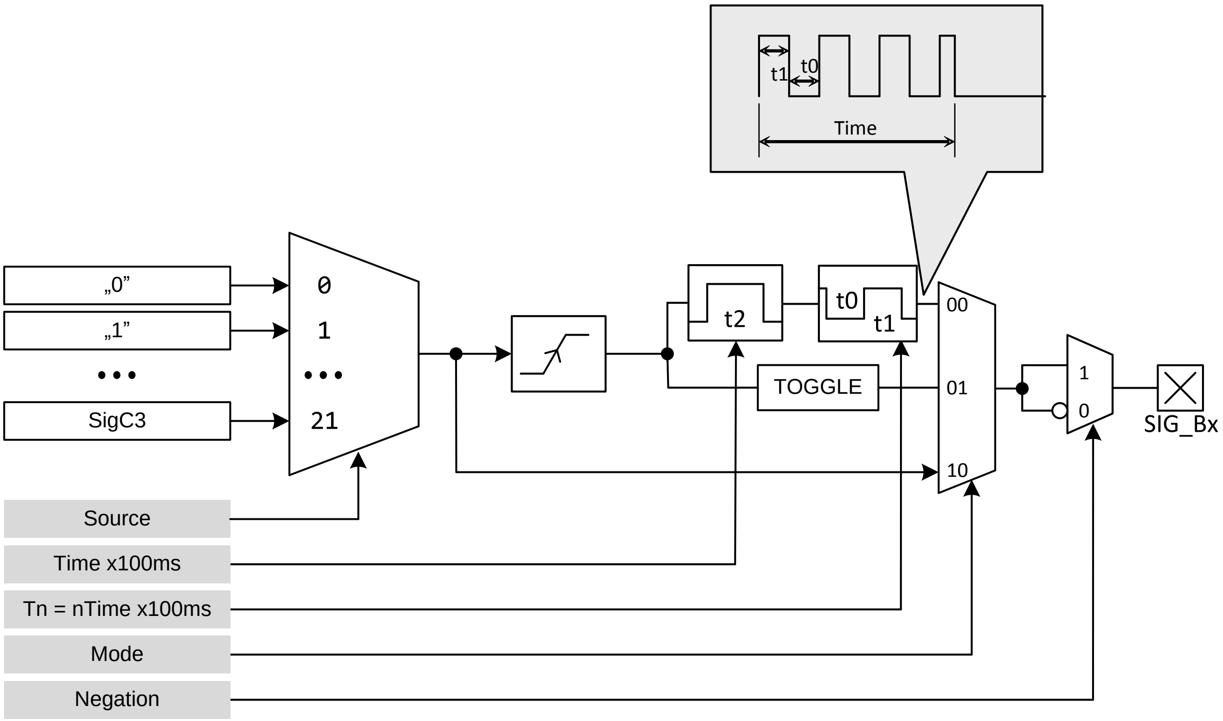
Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_ConfigSIG_A +1 | Function, In0, In1, In2 | OperationCode | CRC |

Where:

Meaning of the response parameters is identical to those described above.

### 7.3.6 SIG_B BLOCK CONFIGURATION



Command frame:

| | |
|---|---|
| C_ConfigSIG_B | No, [Source, Mode, Negation, Time, 0Time, 1Time] |

Parameters: *Source, Mode, Negation, Time, 0Time, 1Time* are optional and if they exist, a new configuration will be written.

| **Parameter name** | Parameter description | Value range |
|---|---|---|
| C_ConfigSIG_B | SIG_B block read-out / write configuration | 0x60 |
| No | Block number SIG_B | 0x00..0x03 |
| Source | Source of control signal | See ID number from Table 4.2 |
| Mode | Specifies the behaviour of output. | 00 – square wave generator<br>01 – change in the output status to opposite of the previous state<br>10 – directly |
| Negation | Output negation. | 0 – Negated output<br>1 – Direct output |
| Time | Time of maintaining switching state after the activation stops. This time is expressed as: Maintaining x 100ms<br>During the "Hold up" time, you can configure the output that can generate a square wave. Time of one and zero is set by following parameters: 0Time and 1Time | 0-255 |
| 0Time | Logical „0” time | 0-255 |
| 1Time | Logical „1” time | 0-255 |

Response frame:

| | |
|---|---|
| C_ConfigSIG_B+1 | No, Source, Mode, Negation, Time, 0Time, 1Time |

Where:
Meaning of response parameters is identical to those described above.

### 7.3.7 SIG_C BLOCK CONFIGURATION

Command frame:

| | |
|---|---|
| C_ConfigSIG_C | No, [Source, Time] |

Parameters: *Source, Time* are optional and if they exist, a new configuration will be written.

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_ConfigSIG_C | SIG_C block read-out / write configuration | 0x62 |
| No | Numer bloku SIG_C | 0x00..0x03 |
| Source | Source of control signal | See ID number from Table 4.2 |
| Time | Filtering time (x100ms) | 0-255 |

Response frame:

| | |
|---|---|
| C_ConfigSIG_C+1 | No, Source, Time |

Where:
Meaning of response parameters is identical to those described above.

### 7.3.8 COLOUR CONFIGURATION

Command frame:

| | | | |
|---|---|---|---|
| header | C_ConfigLed | [C0, C1, C2, C3] | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_ConfigLed | Write/ read-out configuration of displayed colors | 0x5E |
| [C0, C1, C2, C3] | Optional parameters - if present, command writes new configuration.<br>C0 – kolor0 code, priority 1 (highest)<br>C1 – kolor1 code, priority 2<br>C2 – kolor2 code, priority 3<br>C3 – kolor3 code, priority 4 (lowest) | See: Table 4.1 |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_ConfigLed +1 | C0, C1, C2, C3 | OperationCode | CRC |

Where:
Meaning of response parameters is identical to those described above.

## 7.4 ACCESS PASSWORD

### 7.4.1 LOGGING INTO THE READER

Command frame:

| | | | |
|---|---|---|---|
| header | C_LoginUser | Data1...n, 0x0 | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_LoginUser | Logging into the reader | 0xb2 |
| Data1…n | Is any string of bytes | Dowolne z zakresu 0x01…0xff. Długość łańcucha może wynosić od 0 do 8 bajtów |
| 0x00 | Zero terminating string | 0x00 |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_LoginUser +1 | | OperationCode | CRC |

### 7.4.2 PASSWORD CHANGE

Command frame:

| | | | |
|---|---|---|---|
| header | C_ChangeLoginUser | Data1…n, 0x0 | CRC |

Gdzie:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_ChangeLoginUser | Password change | 0xb4 |
| Data1…n | is any string of bytes that will be valid access password. | Any of the ranges 0x01 … 0xff.<br>String length can be from 0 to 8 bytes |
| 0x00 | Zero terminating string | 0x00 |

If Data1 = 0x00 then reader will not be password protected. You can set a new password at any time so that the reader is protected by a password.

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_ChangeLoginUser+1 | | OperationCode | CRC |

### 7.4.3 LOGGING OUT FROM READER

This command will void the last password you provided.

Command frame:

| | | | |
|---|---|---|---|
| header | C_LogoutUser | | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_LogoutUser | Logging out from reader | 0xd6 |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_LogoutUser +1 | | OperationCode | CRC |

## 1.1 AUTOREADER MECHANISM

### 7.4.4 WRITING CONFIGURATION OF MACHINE

C_SetAutoReaderConfig command configures the operating mode of the machine reading unique transponder number.

Described reader gives you the opportunity to temporarily suspend the operation of the machine in case of the correct transmission on the RS link.

If reader works in mixed mode, i.e.

- UID reading machine is being started, and:
- master device (computer, controller) communicates with the reader or with the use of a transponder reader

then:
it is necessary to properly configure the reader so that in case of transmissions with a reader or with a transponder, the reading machine suspends its work.

Command frame:

| | | | |
|---|---|---|---|
| header | C_SetAutoReaderConfig | ATrig, AOfflineTime, Aserial, AMode, Abuzz, AMulti, AInterface | CRC |

Where:

<table>
<tr><th>Parameter name</th><th>Parameter description</th><th>Value range</th></tr>
<tr><td>C_SetAutoReaderConfig</td><td>Writing machine configuration</td><td>0x58</td></tr>
<tr><td>ATrig</td><td>Defines when the UID reading machine needs to work</td><td>0 - machine is permanently off<br>1 - machine is permanently on<br>2 - automatically activated when there is no transmission to RS for longer than AOfflineTime<br>3 - automatically activated when there are no calls for commands to communicate with transponder for a longer period than AOfflineTime</td></tr>
<tr><td>AOfflineTime</td><td>Time of lack of transmission on RS / USB<br>T = AofflineTime * [100 ms]<br>No transmission can refer to any commands (Atrig = 2), or commands to communicate with the transponder (Atrig = 3).<br><br>Communication commands with the transponder are:<br>C_TurnOnAntennaPower<br>C_Select</td><td>0x00…0xff</td></tr>
<tr><td>ASerial</td><td>Automatic sending of the UID transponder number after automatic reading-out from the transponder</td><td>0 - never<br>1 - only for first application of the transponder<br>2 – sends all</td></tr>
<tr><td>AMode</td><td>Configuration byte specifying the format of the sent ID.<br><br>Format:<br><table><tr><th colspan="3">MSB</th><th colspan="3">LSB</th></tr><tr><td>I</td><td>E</td><td>F<1,0></td><td>C<1,0></td><td>D</td><td>ID</td></tr></table><br>UWAGA:<br>Bity E and F<1,0> have meaning only for AInterface=0 or AInterface=1.<br><br>Bity C1, C0, D have meaning only for ASCII format (F<1,0>=1)</td><td>I=1 - Number in reverse order<br><br>E=1 - extended information about collision signaling and card type<br><br>F<1,0>=0 - ID in the Nertonix frame format<br>F<1,0>=1 - ID in ASCII format<br>F<1,0>=2 - ID in binary format<br><br>C<1,0>=0 - No end of line sign<br>C<1,0>=1 - CR end sign<br>C<1,0>=2 – LF end sign<br>C<1,0>=3 – CRLF end sign<br><br>D=1 - convert to decimal format, only for ASCII mode<br><br>ID – extended information about the reader's address set for the RS485 bus</td></tr>
<tr><td>ABuzz</td><td>Automatic signaling read-out by a buzzer after automatic UID read-out from transponder.</td><td>0 - never<br>1 - only for first application of the transponder<br>2 – signals everything</td></tr>
<tr><td>AMulti</td><td>Reading-out mode for many types of transponders<br>Format:<br><table><tr><th colspan="4">MSB</th><th colspan="4">LSB</th></tr><tr><td>-</td><td>-</td><td>-</td><td>-</td><td>S</td><td>I</td><td>C</td><td>M</td></tr></table></td><td>M – transponders MIFARE family<br>C – Calypso (ISO14443B)<br>I – iClass (CSN)<br>S – I-CODE (ISO15693)</td></tr>
<tr><td>AInterface</td><td>Choosing interface after which the autoreader machine sends the read ID</td><td>0 – RS232<br>1 – RS485 / CAN(1)</td></tr>
</table>

| | | |
|---|---|---|
| | | 2 – 1-WIRE<br>3 – WIEGAND<br>4 – RS485 / CAN[1] |

[1] – depending on which interface is set as active

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_ SetAutoReaderConfig +1 | | OperationCode | CRC |

### 7.4.5 READING-OUT CONFIGURATION OF MACHINE

Command frame:

| | | | |
|---|---|---|---|
| header | C_ GetAutoReaderConfig | | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_GetAutoReaderConfig | Reading-out machine configuration | 0x5a |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_ GetAutoReaderConfig +1 | ATrig, AOfflineTime, ASerial, AMode, Abuzz, AMulti | OperationCode | CRC |

Where:

Meaning of response parameters is identical to those described above.

## 7.5 OTHER COMMANDS

### 7.5.1 REMOTE READER RESET

Command frame:

| | | | |
|---|---|---|---|
| header | C_Reset | | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_Reset | Remote reader reset | 0xd0 |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_Reset +1 | | OperationCode | CRC |

### 7.5.2 READING-OUT SOFTWARE VERSION FROM READER

Command frame:

| | | | |
|---|---|---|---|
| header | C_FirmwareVersion | | CRC |

Where:

| Parameter name | Parameter description | Value range |
|---|---|---|
| C_FirmwareVersion | Reading-out reader software version | 0xfe |

Response frame:

| | | | | |
|---|---|---|---|---|
| header | C_FirmwareVersion+1 | Data1…..n | OperationCode | CRC |

Where:

Data1 ... n is a string of characters stored in the form of ASCII codes.

## 7.6 MEANING OF OPERATION CODES IN RESPONSE FRAMES

**Table 7.5 Operation codes**

| Name of operation code | Description | Value |
|---|---|---|
| OC_Error | error | 0x00 |
| OC_ParityError | Parity error | 0x01 |
| OC_RangeError | Parameter range error | 0x02 |
| OC_LengthError | Data length error | 0x03 |

| | | |
|---|---|---|
| OC_ParameterError | Parameter error | 0x04 |
| OC_Busy | Momentary occupancy of internal modules | 0x05 |
| OC_NoACKFromSlave | Lack of internal communication | 0x22 |
| OC_CommandUnknown | Unknown command | 0x07 |
| OC_WrongPassword | Wrong password or last password has expired, i.e. an automatic | 0x09 |
| OC_NoCard | No transponder | 0x0a |
| OC_BadFormat | Bad data format | 0x18 |
| OC_FrameError | Transmission error. It may be indicative of existing interference. | 0x19 |
| OC_NoAnswer | No response from transponder | 0x1E |
| OC_TimeOut | Operation time exceeded. It may indicate a lack of a transponder in field of reader | 0x16 |
| OC_Successful | Operation completed correctly | 0xff |
| **Operation codes related to DESFIRE transponders** | | |
| OC_DesNoChanges | Commit operation did not bring any changes | 0x0c |
| OC_DesOutOfEEprom | No eeprom memory | 0x0e |
| OC_DesIllegalCommand | Illegal command | 0x1c |
| OC_DesIntegrityError | CRC error/transmission with card | 0x1e |
| OC_DesNoSuchKey | Invalid key number | 0x40 |
| OC_DesLengthError | Invalid command length | 0x7e |
| OC_DesPermisionDenied | No permission to perform operation | 0x9d |
| OC_DesParameterError | Command parameter error | 0x9e |
| OC_DesApplNotFound | No application for selected Aid | 0xa0 |
| OC_DesApplIntegrError | Application error, application is blocked | 0xa1 |
| OC_DesAuthError | Authorization error/incorrect key | 0xae |
| OC_DesBoundaryError | Writing/reading-out from record went beyond the size | 0xbe |
| OC_DesPICCIntegError | Internal transponder error, is blocked | 0xc1 |
| OC_DesCountError | 28 applications limit have been exceeded | 0xce |
| OC_DesDuplicateError | Application / File with this identifier already exists | 0xde |
| OC_DesEepromError | Error during reading-out / writing to EEPROM memory | 0xee |
| OC_DesFileNotFound | File with this ID does not exist | 0xf0 |
| OC_DesFileIntegrError | Irreversible file error, the file is blocked | 0xf1 |

# 8 RETURN TO FACTORY SETTINGS

To return to factory settings, within 3 to 10 seconds after starting the device, press the front button for approx. 3 seconds. When returning to factory settings, the following parameters are permanently set:

Tabela 8.6 Ustawienia fabryczne

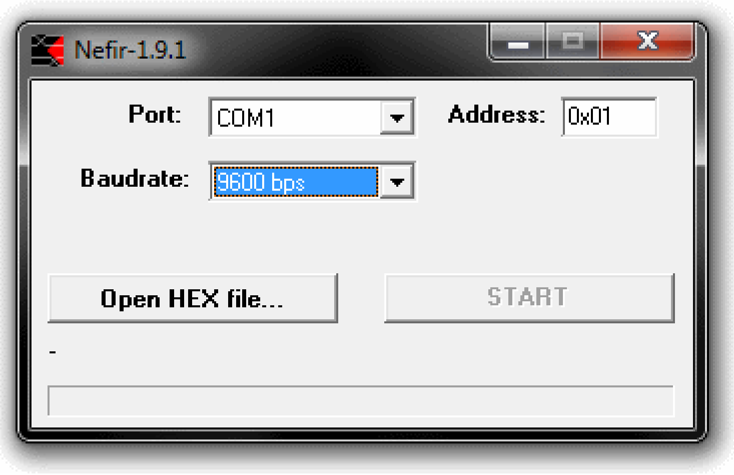
| **Parameter name or functionality** | **Value or setting** | |
|---|---|---|
| Interface | | |
| RS232 interface | Address: 0x01<br>Speed: 0x03 | <br>9600bps |
| RS485/CAN interface | Adress: 0x01<br>Speed: 0x03<br>Type: 0x00 | <br>9600bps<br>RS-485 |
| 1-WIRE interface | Family: 0x01<br>Address: 0x00 | |
| Wiegand interface | Number of bits 37 | |
| Read-out transponder | | |
| AAutoreader | Triger: 0x02<br>Timeout: 0x14<br>Mode: 0xFF<br>ASerial: 0x01<br>AMode: 0x40<br>ABuzzer: 0x01<br>AMulti: 0x09<br>AInterface: 0x00 | <br>2s<br>all supported types<br>for the first touch<br>Netronix format, extended information about collision signaling and card type<br>For the first touch<br>MIFARE + I-CODE<br>RS232 |
| **Inputs/Outputs** | | |
| PinIN0 input | Trigger: Low state | |
| Wejście PinIN1 | Trigger: Low state | |
| Wyjście PinOUT | Source control: Button | |
| Wyjście Kolor0 | Source control: PinIN1 | |
| Wyjście Kolor1 | Source control: Button | |
| Wyjście Kolor2 | Source control: „0” | |
| Wyjście Kolor3 | Source control: „1” | |
| Wyjście Buzzer | Source control: PinIN0 | |
| Color setting | | |
| LED configuration | C0: GREEN<br>C1: BLUE<br>C2: WHITE<br>C3: RED | |
| **SIGNAL blocks** | | |
| SigA0 | In0: „0”; In1: „0”; In2: „0”; Function: OR | |
| SigA1 | In0: „0”; In1: „0”; In2: „0”; Function: OR | |
| SigA2 | In0: „0”; In1: „0”; In2: „0”; Function: OR | |
| SigA3 | In0: „0”; In1: „0”; In2: „0”; Function: OR | |
| SigB0 | Source: „0”, Mode: 2, Negation: 1<br>Time: 0, Time0: 0, Time1: 0 | |
| SigB1 | Source: „0”, Mode: 2, Negation: 1<br>Time: 0, Time0: 0, Time1: 0 | |
| SigB2 | Source: „0”, Mode: 2, Negation: 1<br>Time: 0, Time0: 0, Time1: 0 | |
| SigB3 | Source: „0”, Mode: 2, Negation: 1<br>Time: 0, Time0: 0, Time1: 0 | |
| SigC0 | Source: „0”, Time: 0 | |
| SigC1 | Source: „0”, Time: 0 | |
| SigC2 | Source: „0”, Time: 0 | |
| SigC3 | Source: „0”, Time: 0 | |
| **Password** | | |

| | | |
|---|---|---|
| Password | „”, 0x3C | No password, 60s |

# 9 BOOTLOADER – CHANGING DEVICE'S FIRMWARE

In order to upload a new firmware to device, follow procedure below:

1. Connect device to RS232 serial port on computer
2. Open NEFIR.exe program
3. Set the appropriate COM port and transmission speed to 57600bps
4. Press *Open HEX File* button and load file with new firmware
5. Press START button, which will start firmware reloading
6. Wait for end of reloading process.



Nefir-1.9.1

Port: COM1

Address: 0x01

Baudrate: 9600 bps

-

Drawing 9.2 Program window view when reloading firmware