ND67-v16 APB D281021

TECHNICAL DATA SHEET MW-R7B / MW-R7G MW-R4B / MW-R4G

MW-R7G and MW-R7B

Documentation version:	ND67-v16
Date:	28.10.2021
Valid from the firmware version:	MW-R7x-v4.8.A1.6

Contents:

1 Introduction 5
2 Technical data 6
3 Wires 7
3.1 MW-R4B / MW-R4G 7
3.2 MW-R7B / MW-R7G 7
4 Input / Output 8
4.1 Physical inputs 8
4.1.1 PinINx parameters 8
4.2 Physical outputs 8
4.2.1 PinOut parameters 8
4.3 RGB LED 8
4.4 Sources of SIGNALS controlling outputs 8
4.4.1 Source „0” and source „1” 9
4.4.2 Source button 9
4.4.3 Source AnyCard 9
4.4.4 Source RSx 9
4.4.5 Source PinINx 9
4.4.6 Source SIG_Ax 10
4.4.7 Source SIG_Bx 10
4.4.8 Source SIG_Cx 10
5 Dimensions 11
6 Interface 12
6.1 Interface 1-WIRE 12
6.2 Interface WIEGAND 12
6.3 Interface RS232 / RS485 / CAN 12
6.3.1 Communication protocol for RS232 / 485 13
6.3.2 Communication protocol for CAN interface 13
6.3.3 Signal levels for serial RS232(TTL) 13
7 Netronix protocol - Available commands for RS232/RS485/CAN interface 14
7.1 Serial interface configuration 14
7.1.1 Writing serial interface configuration 14
7.1.2 Reading-out serial interface configuration 15
7.2 Communication orders with transponders 15
7.2.1 Key management 15
7.2.1.1 Writing mifare classic key to the dynamic key memory 16
7.2.1.2 Writing mifare classic key to the static key memory 16
7.2.1.3 Writing AES / 3DES key to the static key memory 16
7.2.2 Common commands for communication with transponders 17
7.2.2.1 Enabling and disabling reader field 17
7.2.2.2 Selection of one transponder from many 17
7.2.2.3 Get transponder to sleep mode in the field 17
7.2.3 Commands for communication with mifare classic transponders 18
7.2.3.1 Logging into transponder sector using dynamic key 18
7.2.3.2 Logging into transponder sector using static key buffer 18
7.2.3.3 Reading-out contents of transponder block 18
7.2.3.4 Writing content of transponder block 19
7.2.3.5 Copying content of transponder block to another block 19
7.2.3.6 Writing values to transponder block 19
7.2.3.7 Reading-out valueS from transponder block 19
7.2.3.8 Increasing value contained in transponder block 20
7.2.3.9 Decreasing value contained in transponder block 20
7.2.4 Commands for communication for Mifare Ultralight transponders 20
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 21
7.2.5.1 SL0 level commands 21
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 24
7.2.6.1 Authorization, logging into the currently selected application 24
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 26
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 27
7.2.6.11 Downloading file properties 28
7.2.6.12 Creating Standard Data Files type 28
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 32
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 33
7.2.9 Mifare Application Directory - MAD 33
7.2.9.1 Formatting MAD cards 33
7.2.9.2 Adding application to MAD directory 34
7.2.9.3 Searching for sector for given application 34
7.2.9.4 Searching for next application sector 35
7.3 Electrical sources, inputs and outputs 35
7.3.1 Writing Rsx source status 35
7.3.2 Reading-out source status 35
7.3.3 Writing port configuration 36
7.3.4 Reading-out port configuration 36
7.3.5 SIG_A block configuration 37
7.3.6 SIG_B block configuration 38
7.3.7 SIG_C block configuration 38
7.3.8 Colour configuration 39
7.4 Access password 39
7.4.1 Logging into the reader 39
7.4.2 Password change 39
7.4.3 Logging out from reader 40
7.5 AutoReader mechanism 40
7.5.1 Writing configuration of machine 40
7.5.2 Reading-out configuration of machine 42
7.6 Support for ID stored in transponder memor 42
7.6.1 UserID configuration 42
7.7 Other commands 42
7.7.1 Remote reader reset 43
7.7.2 Reading-out software version from reader 43
7.8 Meaning of operation codes in response frames 43
8 MODBUS RTU protocol 45
8.1 Supported MODBUS protocol functions 45
8.2 MODBUS adress 45

8.2.1 Register Addresses for reading transponder ID 45
8.2.2 Register addresses for reading/saving autoreader configuration 45
8.2.3 Register addresses for reading/saving RS232 / RS48 configuration 45
8.2.4 Register addresses for reading/saving block SIG_A configurations 46
8.2.5 Register addresses for reading/saving block SIG_B configurations 46
8.2.6 Register addresses for reading/saving block SIG_C configurations 46
8.2.7 Register addresses for reading/saving LED configurations 47
8.2.8 Register addresses for reading/saving i/o configurations 47
8.2.9 Register addresses for reading/saving UserID configurations 47
8.2.10 Register addresses for saving values of rsx sources 47
8.2.11 Register addresses for reading source values 47
8.3 Encapsulation Netronix protocol inside MODBUS RTU 48
8.3.1 Workflow 48
8.3.2 Example of use - reading the firmware version 49
9 Return to factory settings 51
10 Bootloader – changing device’s firmware 53

1 INTRODUCTION

MW-R7x / MW-R4x is a wall-mounted reader of RFID cards which works on 13,56 kHz rated frequency.
Main features:

	MW-R4B / MW-R4G	MW-R7B / MW-R7G
Support for transponders:	MIFARE® Classic, Plus, Ultralight C, DESFire, I-CODE SLI, iClass (tylko CSN)	
Interfaces:	RS485	RS232, RS485, 1-WIRE, WIEGAND CAN
Buzzer	YES	
Built-in LED RGB of common purpose dowolnego przeznaczenia	YES	
Button	YES	
Available in colors:	black (MW-R4B) beige (MW-R4G)	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

Reader parameters	MW-R4x	MW-R7x
Supply voltage	8-15V	
Maximal supply currenht	120 mA	
Rated operation RF frequency of module	13,56MHz	
Reading-out distance of transponders	do 8 cm	
Dimentions (wid.* len. * height)	44x83x14 mm	
IP rating	IP54	
Button	Yes - capacitive	
Interfaces	RS485	RS232 (TTL), RS485, 1-WIRE, WIEGAND, CAN
Input / output	Anticollision (In/Out) PinOUT (Out) PinIN0 (In) PinIN1 (In)	

3 WIRES

3.1 MW-R4B / MW-R4G

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 (RS485_B)
Yellow	PinINTERFACE2	Serial interface line (RS232_ A)
Grey	PinIN0	Input for any purpose
Pink	PinIN1	Input for any purpose

3.2 MW-R7B / MW-R7G

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 (RS232_TX, RS485_B, CAN_H, WIEGAND0, 1WIRE)
Yellow	PinINTERFACE2	Serial interface line (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-R4x / MW-R7x reader has got three physical inputs:

1. PinIN0
2. PinIN1
3. Button

4.1.1 PININX PARAMETERS

Logic level „0”	0 …0.2V
Logic level „1”	3.75…25V

Inputs has internal 10kOhm pull up to +5V.

4.2 PHYSICAL OUTPUTS

MW-R7x reader has got six physical outputs:

1. Colour0 (RGB LED)
2. Colour1 (RGB LED)
3. Colour2 (RGB LED)
4. Colour3 (RGB LED)
5. Buzzer
6. PinOUT

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

4.2.1 PINOUT PARAMETERS

Output type	Open collector
Maximum voltage at high impedance	40V
Maximum current	400mA

4.3 RGB LED

MW-R4x / 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 ColourX output can be done with *Colour configuration* command. When deter-minig which colour is to be displayed, Colour0 input has the highest priority, Colour3 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 con-tains 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	Button	Source reflecting the status of button. It has value of 1 when button is pressed and value of 0 otherwise
3	Any Card	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 5 6 7	RS_0 RS_1 RS_2 RS_3	Sources controlled via RS232 serial interface. See C_WriteSourceRSx command
8 9	PinIN0 PinIN1	Sources controlled by physical input pin using the INPUT block
10 11 12 13	SigA0 SigA1 SigA2 SigA3	Sources controlled by SIG_Ax block outputs
14 15 16 17	SigB0 SigB1 SigB2 SigB3	Sources controlled by SIG_Bx block outputs
18 19 20 21	SigC0 SigC1 SigC2 SigC3	Sources controlled by SIG_Cx block outputs

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.

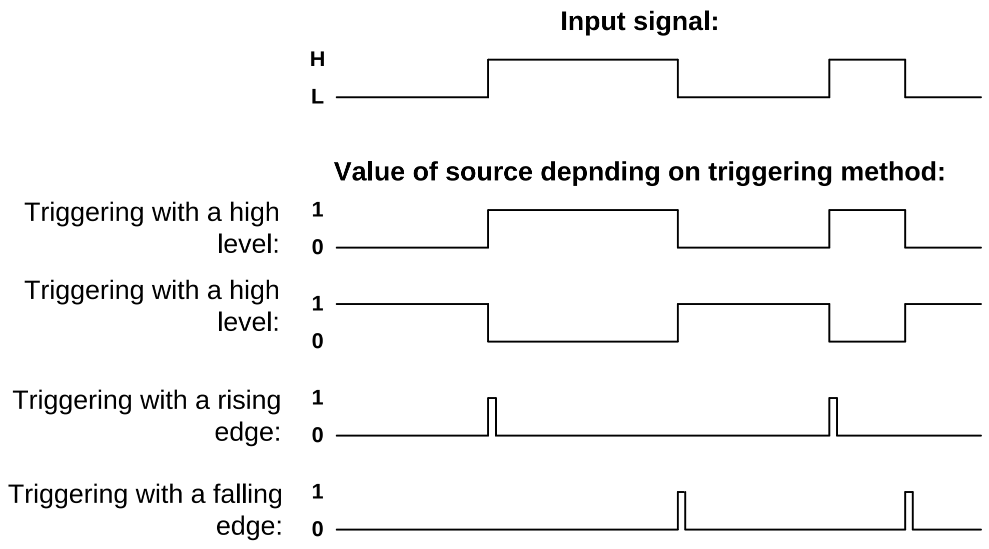
4.4.3 SOURCE ANYCARD
Source reflecting information about the presence of card in the field of the reader. It has value 1 when the card is in the field and value 0 in the opposite case.

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_WriteSourceRSx command.

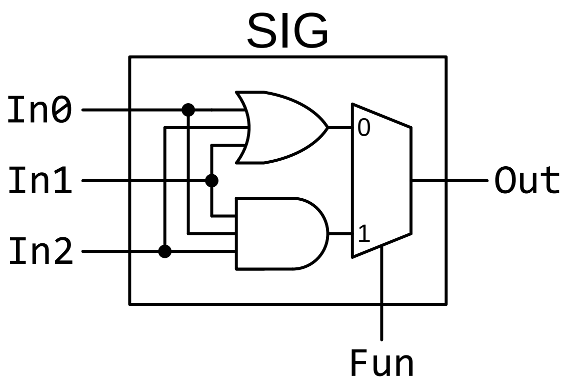
4.4.5 SOURCE PININX
PinINx sources are controlled through physical inputs. Depending on configuration, source has value:


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.

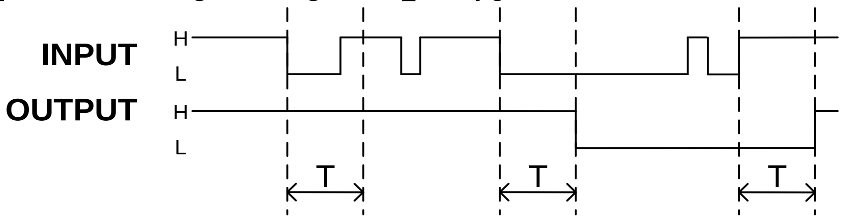


4.4.7 SOURCE SIG_BX

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

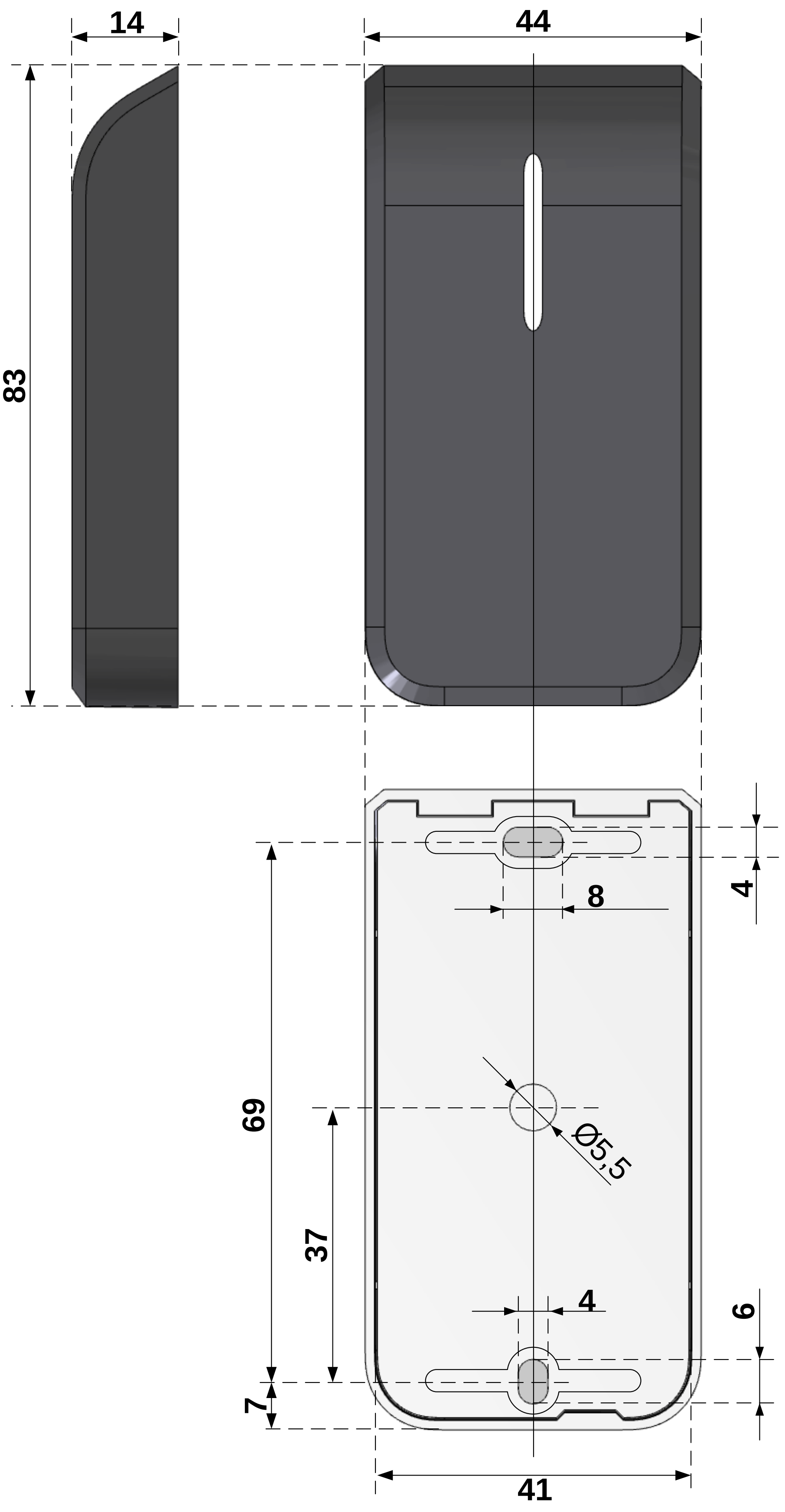
MW-R7x reader has 4 virtual SIG_C blocks that allow you to filter logic signals. The state at the SIG_C output will change to the same as at the input if the input state remains constant for the time defined by the Time parameter. The SIG_C blocks are configured using the *SIG_C Configuration block*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

If they occur, the RS-232 and RS-485/CAN interfaces listen all the time while waiting for a command. AutoReader sends the read-out ID via the interface selected in the AutoReader configuration.

6.1 INTERFACE 1-WIRE

NOTE:
1-WIRE interface is only available in MW-R7B / MW-R7G version

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

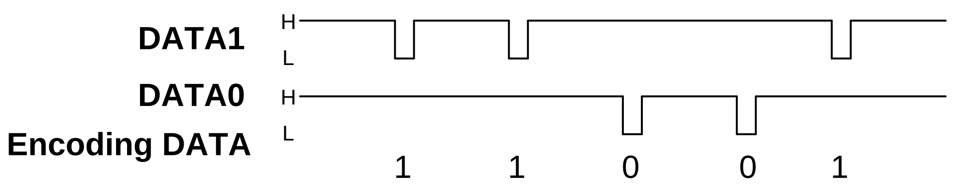
To change a parameter *Address* or *Family code*, please send a *C_SetInterfaceConfig* command to the reader.

6.2 INTERFACE WIEGAND

NOTE:
WIEGAND interface is only available in MW-R7B / MW-R7G version

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 parameters	**Card ID / responding WIEGAND frame**	
P1=26, P2=0	0b0001001000110100010101100111100010011010 P00010010001101000101011 0N	Card ID WIEGAND frame
P1=37, P2=0	0b0001001000110100010101100111100010011010 P00010010001101000101011001111000100N	Card ID WIEGAND frame
P1=26, P2=1	0b0001001000110100010101100111100010011010 P01010110011110001001101 0N	Card ID 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

NOTE:
RS232(TTL) and CAN interfaces are only available in MW-R7B / MW-R7G

MW-R 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.

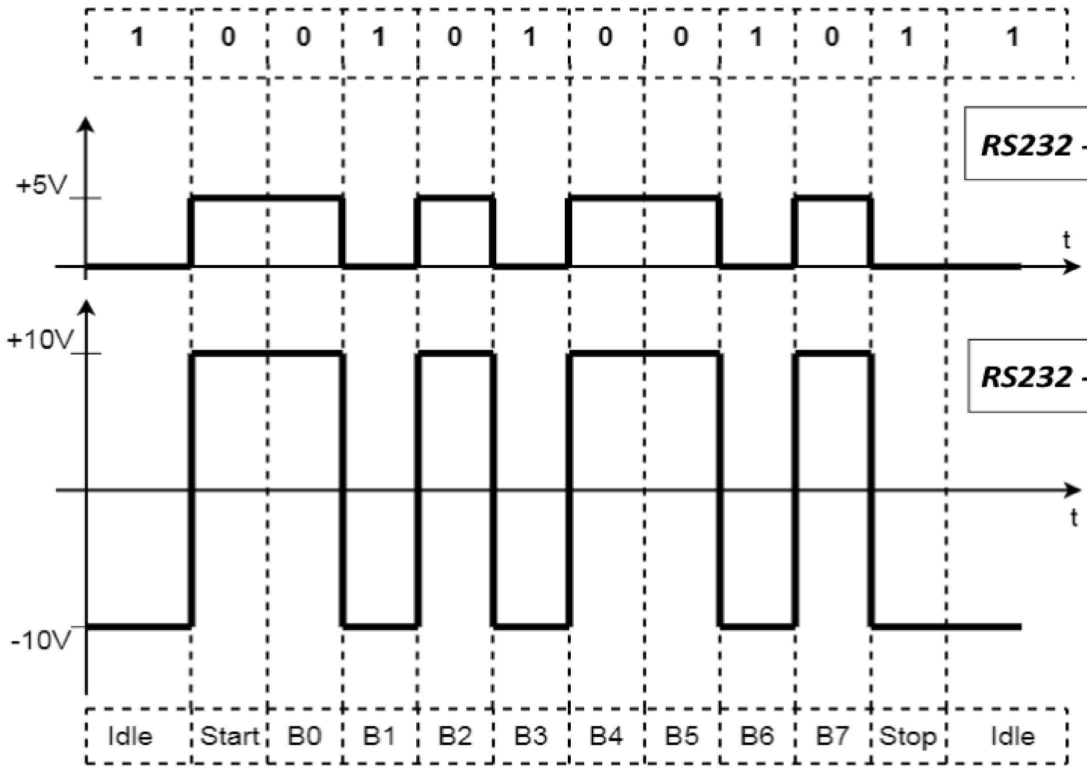
6.3.2 COMMUNICATION PROTOCOL FOR CAN INTERFACE

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.

6.3.3 SIGNAL LEVELS FOR SERIAL RS232(TTL)



7 NETRONIX PROTOCOL - 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 1 – RS485[1] 2 – 1-WIRE 3 – WIEGAND 4 – CAN
P1, P2, [P3][2]	Parameters depending on P0 field value: For Typ=0 P1 – Logical address (RS232) P2 – Transmission speed (RS232) [P3] – Optional parameter. Force TX line initialization at device start. No parameter does not change the current value. For Typ=1 or Typ=4 P1 – Logical address (RS485 / CAN) P2 – Transmission speed (RS485) [P3] – Optional parameter. RS485/CAN interface switch For Typ=2 P1 – ConfAdr (7th byte of Dallas frame) P2 – ConfFC (1st byte of Dallas frame) Dla Typ=3 P1 – Number of bits P2 – L/M. This switch determines which part of the card ID will be sent in the WIEGAND frame	 P1: 0x01 - 0xFE P2: See Tabela 7.3 P3: 0 – Don't initialize TX, 1 –initializeTX. P1: 0x01 - 0xFE P2: See Tabela 7.3 P3: 0 – RS485 (default value), 1 - CAN P1: 0x00-0xFF P2: 0x00-0xFF P1: 26 - 37 P2: 0-1

(1) – Only allowable value for MW-R4B / MW-R4G
(2) – Parameter only exists in MW-R7B / MW-R7G version

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 1 – RS485 2 – 1-WIRE 3 – WIEGAND 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 1 – RS485 2 – 1-WIRE 3 – WIEGAND 4 – CAN
P1, P2, [P3]	Parameters depending on P0 field value: For Typ=0 P1 – Logical address (RS232) P2 – Transmission speed (RS232) For Typ=1 or Typ=4 P1 – Logical address (RS485 / CAN) P2 – Transmission speed (RS485) [P3] – Optional parameter. RS485/CAN interface switch For Typ=2 P1 – ConfAdr (7th byte of Dallas frame) P2 – ConfFC (1st byte of Dallas frame) Dla Typ=3 P1 – Number of bits P2 – L/M. This switch determines which part of the card ID will be sent in the WIEGAND frame	 P1: 0x01 - 0xFE P2: – See Tabela 7.3 P1: 0x01 - 0xFE P2: – See Tabela 7.3 P3: 0 – RS485 (default value), 1 - CAN P1: 0x00-0xFF P2: 0x00-0xFF P1: 26 - 37 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 overwrit-ten 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. 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 overwrit-ten 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.	0x00…0x1f

	Reader can store up to 32 different keys.
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 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 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 3 - HITAG 4 - HID
ID1…IDn	Unique transponder number	ID1 – LSB, 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 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 0xBB – B type key
SKNo	Static Key Number	0x00…0x1F

Response frame:

header	C_LoginWithSKB +1		OperationCode	CRC

7.2.3.3 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

7.2.3.4 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	Data to be saved in transponder block	any

Response frame:

header	C_WriteBlock +1		OperationCode	CRC

7.2.3.5 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

7.2.3.6 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 BackupBlockNo does not have a significant im-pact 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

7.2.3.7 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

7.2.3.8 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

7.2.3.9 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 read-ing-out. If PageAdr> 0x ???? this will read-out pages at the be-ginning 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

Where:

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 0x9003 – transition to level SL3 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 0x00-0x27 – card Plus 4k
KeyType	Key type	0xAA – klucz A 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:

Parameter name	Parameter description	Value range
C_MfPlusCMD	MFPlus support command	0x3A

read_cmd	Read-out procedure type:	0x30-0x33

cmd.	MAC on command	MAC on resonse	Plain /en-crypted
0x30	Yes	No	Encrypted*
0x31	Yes	Yes	Encrypted*
0x32	Yes	No	Plan
0x33	Yes	Yes	Plan

block	Block number for reading-out	0-3 for sectors<32 0-15 for sectors>32

*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

Where:

Parameter name	Parameter description	Value range
C_MfPlusCMD	MFPlus support command	0x3A
write _cmd	Writing procedure type:	0xA0-0xA3

cmd.	MAC on command	MAC on resonse	Plain /en-crypted
0xA0	Yes	No	Encrypted*
0xA1	Yes	Yes	Encrypted*
0xA2	Yes	No	Plain
0xA3	Yes	Yes	Plain

block	Block number to read-out	0-3 for sectors<32 0-15 for sectors>32
data0..data15	Data for writing transponder block	

*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

Where:

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 : 0x0A – DES 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, 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

Where:

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	0x00..0x13

	If changed key is the same in which current authorization took place, this parameter is left blank

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: Number of keys assigned to the application Bit7..Bit6: 00 – DES authorization for entire application 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 re-cords		Current number of re-cords	
		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: 0x01 – unencrypted 0x03 – DES encrypted	0x00,0x03

AccRight1..2	Access rights to file, see table below	0x00..0xff
FileSize1..3	3-byte file size in bytes, 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_DesCreateBACKDataFile	Command to create a BACKUP file	0x6a
FileNo	File ID	0..0x07
ComSett	Transmission type: 0x01 – unencrypted 0x03 – DES encrypted	0x00,0x03
AccRight1..2	File access rights	0x00..0xff
FileSize1..3	3 byte file size in bytes 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: 0x01 – unecrypted 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 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: 0x01 – unecrypted 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 (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, (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, (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_TransclBlock	data	CRC

Where:

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

Response frame:

header	C_TransclBlock+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 0xa8	Formatting to MAD	0xa8
Type	1 - MAD1 (15sectors)	0x01,0x02

	2 – MAD2 (30sectors)	
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 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 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 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 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 See the chapter:: 9 PinINx	0 – not negated 1 – negated 2 – reaction to increasing slope 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:

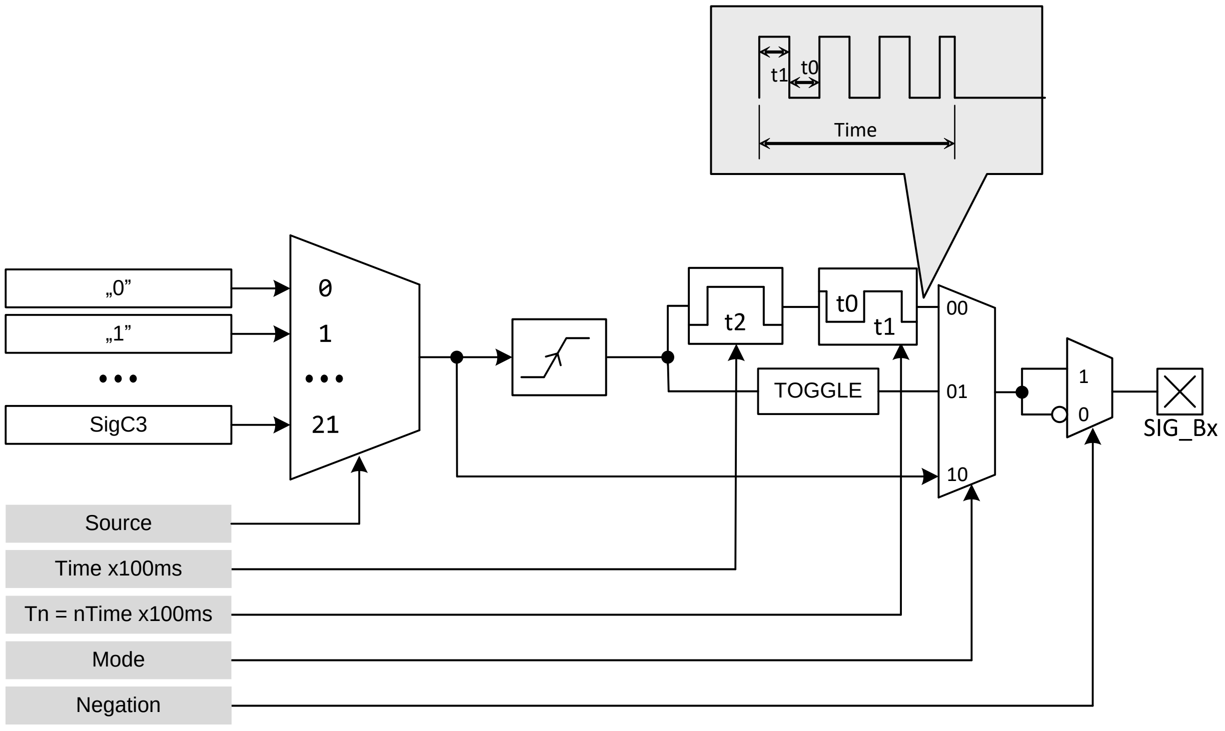
Parameter name	Parameter description	Value range
C_ConfigSIG_A	SIG_A block read-out / write configuration	0x5C
SigNo	Block number SIG_A, whose configuration is to be read-out / written	0x00…0x03
Function	Optional parameter - if present, command writes new configuration. Specifies the type of function executed by SIG_A block.	0 – function OR 1 – function AND
In1, In2, In3	Optional parameters - if present, command writes new configuration. 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 01 – change in the output status to opposite of the previous state 10 – directly
Negation	Output negation.	0 – Negated output 1 – Direct output
Time	Time of maintaining switching state after the activation stops. This time is expressed as: Maintaining x 100ms 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. C0 – kolor0 code, priority 1 (highest) C1 – kolor1 code, priority 2 C2 – kolor2 code, priority 3 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	Value: 0x00..0xFF The maximum length is 8 bytes
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. 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

7.5 AUTOREADER MECHANISM

7.5.1 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, [AModeParam], A Abuzz, AMulti, AInterface	CRC

Where:

Parameter name	Parameter description	Value range
C_SetAutoReader-Config	Writing machine configuration	0x58
ATrig	Defines when the UID reading machine needs to work	0 - machine is permanently off 1 - machine is permanently on

		2 - automatically activated when there is no transmission to RS for longer than AOfflineTime 3 - automatically activated when there are no calls for commands to communicate with transponder for a longer period than AOfflineTime
AOfflineTime	Time of lack of transmission on RS / USB T = AofflineTime * [100 ms] No transmission can refer to any commands (Atrig = 2), or commands to communicate with the transponder (Atrig = 3). Communication commands with the transponder are: C_TurnOnAntennaPower C_Select	0x00...0xff
ASerial	Automatic sending of the UID transponder number after automatic reading-out from the transponder	0 - never 1 - only for first application of the transponder 2 – sends all
AMode	Configuration byte specifying the format of the sent ID. Format: MSB … LSB: I | E | F<1,0> | C<1,0> | D | ID NOTE: Bity E and F<1,0> have meaning only for AInterface=0 or AInterface=1. Bity C1, C0, D have meaning only for ASCII format (F<1,0>=1)	I=1 - Number in reverse order E=1 - extended information about collision signaling and card type F<1,0>=0 - ID in the Nertonix frame format F<1,0>=1 - ID in ASCII format F<1,0>=2 - ID in binary format F<1,0>=3 - ID in ASCII format (DEC) C<1,0>=0 - No end of line sign C<1,0>=1 - CR end sign C<1,0>=2 – LF end sign C<1,0>=3 – CRLF end sign D=1 - convert to decimal format, only for ASCII mode ID – extended information about the reader's address set for the RS485 bus
[AModeParam]	Optional parameter. If it is not pressent, its value is dixed at 0.	For AMode.F<1,0>=3 Specifies the numer of ID characters.
ABuzz	Automatic signaling read-out by a buzzer after automatic UID read-out from transponder.	0 - never 1 - only for first application of the transponder 2 – signals everything
AMulti	Reading-out mode for many types of transponders Format: MSB … LSB: - | - | - | - | S | I | - | M	M – transponders MIFARE family I – iClass (CSN) S – I-CODE (ISO15693)
AInterface	Choosing interface after which the auto-reader machine sends the read ID	0 – RS232 1 – RS485 / CAN(1) 2 – 1-WIRE 3 – WIEGAND

		4 – RS485 / CAN(1)

(1) – depending on which interface is set as active

Response frame:

header	C_ SetAutoReaderConfig +1		OperationCode	CRC

7.5.2 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.

7.6 SUPPORT FOR ID STORED IN TRANSPONDER MEMOR

The MW-R7 reader can read the ID saved in the transponder's memory by the user.Write/read configurations.

7.6.1 USERID CONFIGURATION

Command frame:

header	C_ConfigUserID	[CardType, ID_Len, ID_Offset, Param[…]]	CRC

Where:

Parameter name	Parameter description	Value range
C_ConfigUserID	Command code	0xBA
CardType	Card type	0x50 – Mifare S50 0x70 – Mifare S70 0xDF - DESFire
ID_Len	Length of saved ID	0x01…0x14
ID_Offset	Offset ID relative to 1 byte in the sector.	0x20-ID_Len
Parameters for CardType=0x50 or CardType=0x70 (Mifare Classic)		
Param[0] – SecNo	Sector number	
Param[1] – KeyType	Key type	0xAA – key A 0xBB – key B
Param[2] – SKBKeyNo	The key number in the static key memory (SKB) that is used to log on to the sector.	0-0x1F
Parameters for CardType=0xDF (DES Fire)		
Param[0] – FileNo	File number	0x00 – 0xFF
Param[1] – AuthType	Typ klucza	0x0A – DES 0x3A – 3DES 0xAA - AES
Param[2] – KeyNo	The number at which the key is stored in memory	0-7
Param[3..5] – AppId	3 bajtowe ID aplikacji	0x000000 – 0xFFFFFF

Response frame:

	C_ConfigUserID+1	CardType, ID_Len, ID_Offset, SecNo, Param[]	OperationCode	

Meaning of response parameters is identical to those described above.

7.7 OTHER COMMANDS

7.7.1 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

7.7.2 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.

7.8 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_BadCommand		0x08
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_NoAntennaPower	Antenna power is turn off	0x30
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 MODBUS RTU PROTOCOL

NOTE:
When using the MODBUS protocol, the reader should not send automatically the readed ID.
To disable automatic ID sending, enter the value 0 in the register with the address 1022 (ASerial).

8.1 SUPPORTED MODBUS PROTOCOL FUNCTIONS

Function	Description
0x01	Read Coils
0x02	Read Discrete Inputs
0x03	Read Holding Regs
0x04	Read Input Regs
0x05	Write Single Coil
0x06	Write Single Reg
0x10	Write Multiple Regs

8.2 MODBUS ADRESS

8.2.1 REGISTER ADDRESSES FOR READING TRANSPONDER ID

Address	Type	R/W	Description
996	Holding Reg	R/W	New card state Reading: 1-new transponder detect Writing: 0 – clear this flag
997	Holding Reg	R	B<15:8> - Transponder type B<7:0> - Number of collisions
998	Holding Reg	R	ID Length
999	Holding Reg	R	Time counter since last reading (x100ms)
1000	Holding Reg	R	Transponder ID [0]
1001	Holding Reg	R	Transponder ID [1]
1002	Holding Reg	R	Transponder ID [2]
1003	Holding Reg	R	Transponder ID [3]
1004	Holding Reg	R	Transponder ID [4]
1005	Holding Reg	R	Transponder ID [5]
1006	Holding Reg	R	Transponder ID [6]
1007	Holding Reg	R	Transponder ID [7]

8.2.2 REGISTER ADDRESSES FOR READING/SAVING AUTOREADER CONFIGURATION

Address	Type	R/W	Description
1020	Holding Reg	R/W	ATrig
1021	Holding Reg	R/W	AOfflineTimer
1022	Holding Reg	R/W	ASerial
1023	Holding Reg	R/W	B<15:8> - AModeParam, B<7:0> - AMode
1024	Holding Reg	R/W	ABuzz
1025	Holding Reg	R/W	AMulti
1026	Holding Reg	R/W	AInterface

The meaning of registers is the same as for the C_SetAutoReaderConfig command.

8.2.3 REGISTER ADDRESSES FOR READING/SAVING RS232 / RS48 CONFIGURATION

Address	Type	R/W	Description
1030	Holding Reg	R/W	Device address on the RS232 bus
1031	Holding Reg	R/W	Baudrate on the RS232 bus (See Tabela 7.3)
1032	Holding Reg	R/W	Device address on the RS485 bus
1033	Holding Reg	R/W	Baudrate on the RS485 bus (See Tabela 7.3)

8.2.4 REGISTER ADDRESSES FOR READING/SAVING BLOCK SIG_A CONFIGURATIONS

Address	Type	R/W	Description
1040	Holding Reg	R/W	SigA_0 – Function
1041	Holding Reg	R/W	SigA_0 – In0
1042	Holding Reg	R/W	SigA_0 – In1
1043	Holding Reg	R/W	SigA_0 – In2
1044	Holding Reg	R/W	SigA_1 – Function
1045	Holding Reg	R/W	SigA_1 – In0
1046	Holding Reg	R/W	SigA_1 – In1
1047	Holding Reg	R/W	SigA_1 – In2
1048	Holding Reg	R/W	SigA_2 – Function
1049	Holding Reg	R/W	SigA_2 – In0
1050	Holding Reg	R/W	SigA_2 – In1
1051	Holding Reg	R/W	SigA_2 – In2
1052	Holding Reg	R/W	SigA_3 – Function
1053	Holding Reg	R/W	SigA_3 – In0
1054	Holding Reg	R/W	SigA_3 – In1
1055	Holding Reg	R/W	SigA_3 – In2

8.2.5 REGISTER ADDRESSES FOR READING/SAVING BLOCK SIG_B CONFIGURATIONS

Address	Type	R/W	Description
1060	Holding Reg	R/W	SigB_0 – Source
1061	Holding Reg	R/W	SigB_0 – Mode
1062	Holding Reg	R/W	SigB_0 – Negation
1063	Holding Reg	R/W	SigB_0 – Time
1064	Holding Reg	R/W	SigB_0 – 0Time
1065	Holding Reg	R/W	SigB_0 – 1Time
1066	Holding Reg	R/W	SigB_1 – Source
1067	Holding Reg	R/W	SigB_1 – Mode
1068	Holding Reg	R/W	SigB_1 – Negation
1069	Holding Reg	R/W	SigB_1 – Time
1070	Holding Reg	R/W	SigB_1 – 0Time
1071	Holding Reg	R/W	SigB_1 – 1Time
1072	Holding Reg	R/W	SigB_2 – Source
1073	Holding Reg	R/W	SigB_2 – Mode
1074	Holding Reg	R/W	SigB_2 – Negation
1075	Holding Reg	R/W	SigB_2 – Time
1076	Holding Reg	R/W	SigB_2 – 0Time
1077	Holding Reg	R/W	SigB_2 – 1Time
1078	Holding Reg	R/W	SigB_3 – Source
1079	Holding Reg	R/W	SigB_3 – Mode
1080	Holding Reg	R/W	SigB_3 – Negation
1081	Holding Reg	R/W	SigB_3 – Time
1082	Holding Reg	R/W	SigB_3 – 0Time
1083	Holding Reg	R/W	SigB_3 – 1Time

8.2.6 REGISTER ADDRESSES FOR READING/SAVING BLOCK SIG_C CONFIGURATIONS

Address	Type	R/W	Description
1090	Holding Reg	R/W	SigC_0 – Source
1091	Holding Reg	R/W	SigC_0 – Time
1092	Holding Reg	R/W	SigC_1 – Source
1093	Holding Reg	R/W	SigC_1 – Time
1094	Holding Reg	R/W	SigC_2 – Source
1095	Holding Reg	R/W	SigC_2 – Time
1096	Holding Reg	R/W	SigC_3 – Source
1097	Holding Reg	R/W	SigC_3 – Time

8.2.7 REGISTER ADDRESSES FOR READING/SAVING LED CONFIGURATIONS

Address	Type	R/W	Description
1100	Holding Reg	R/W	C0 – color code 0 (See Table 4.1)
1101	Holding Reg	R/W	C1 – color code 1 (See Table 4.1)
1102	Holding Reg	R/W	C2 – color code 2 (See Table 4.1)
1103	Holding Reg	R/W	C3 – color code 3 (See Table 4.1)

8.2.8 REGISTER ADDRESSES FOR READING/SAVING I/O CONFIGURATIONS

Address	Type	R/W	Description
1104	Holding Reg	R/W	Source – PinOut
1105	Holding Reg	R/W	Source – Colour0
1106	Holding Reg	R/W	Source – Colour1
1107	Holding Reg	R/W	Source – Colour2
1108	Holding Reg	R/W	Source – Colour3
1109	Holding Reg	R/W	Source – Buzzer
1110	Holding Reg	R/W	Source – PinIn0
1111	Holding Reg	R/W	Source – PinIn1

8.2.9 REGISTER ADDRESSES FOR READING/SAVING USERID CONFIGURATIONS

Adres	Typ	R/W	Opis
1150	Holding Reg	R/W	CardType
1151	Holding Reg	R/W	ID_Len
1152	Holding Reg	R/W	ID_Offset
1153	Holding Reg	R/W	Param[0]
1154	Holding Reg	R/W	Param[1]
1155	Holding Reg	R/W	Param[2]
1156	Holding Reg	R/W	Param[3]
1157	Holding Reg	R/W	Param[4]
1158	Holding Reg	R/W	Param[5]

The meaning of registers is the same as for the C_ConfigUserID command.

8.2.10 REGISTER ADDRESSES FOR SAVING VALUES OF RSX SOURCES

Address	Type	R/W	Description
1200	Holding Reg	W	RS0
1201	Holding Reg	W	RS1
1202	Holding Reg	W	RS2
1203	Holding Reg	W	RS3

8.2.11 REGISTER ADDRESSES FOR READING SOURCE VALUES

Address	Type	R/W	Description
100	Coil / Discrete Inputs	R	„0”
101	Coil / Discrete Inputs	R	„1”
102	Coil / Discrete Inputs	R	Button
103	Coil / Discrete Inputs	R	AnyCard
104	Coil / Discrete Inputs	R	RS_0
105	Coil / Discrete Inputs	R	RS_1
106	Coil / Discrete Inputs	R	RS_2
107	Coil / Discrete Inputs	R	RS_3
108	Coil / Discrete Inputs	R	PinIn0
109	Coil / Discrete Inputs	R	PinIn1
110	Coil / Discrete Inputs	R	SigA0
111	Coil / Discrete Inputs	R	SigA1

112	Coil / Discrete Inputs	R	SigA2
113	Coil / Discrete Inputs	R	SigA3
114	Coil / Discrete Inputs	R	SigB0
115	Coil / Discrete Inputs	R	SigB1
116	Coil / Discrete Inputs	R	SigB2
117	Coil / Discrete Inputs	R	SigB3
118	Coil / Discrete Inputs	R	SigC0
119	Coil / Discrete Inputs	R	SigC1
120	Coil / Discrete Inputs	R	SigC2
121	Coil / Discrete Inputs	R	SigC3

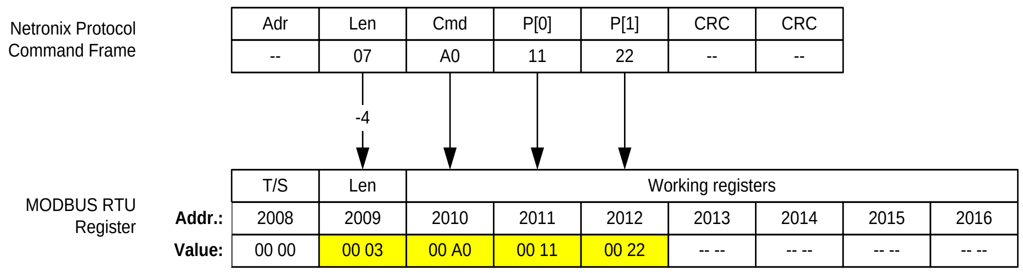
8.3 ENCAPSULATION NETRONIX PROTOCOL INSIDE MODBUS RTU

Any command from the Netronix protocol can be executed using the appropriate registers from the MODBUS protocol.

Addr.	**Type**	**R/W**	**Name**	**Description**
2008	Holding Reg	R/W	Trigger/Status	This register is used to trigger command processing and to check the processing status. Allowed values: 0x0000 – Module in IDLE mode 0x0001 – Triggering processing 0x00EE – Error 0x00FF – Command completed. The answer is in the work registers.
2009	Holding Reg	R/W	Len	Rejestr ten zawiera długość zapisanej komendy / długość odpowiedzi (ilość rejestrów zapisanych/do odczytania)
2010-2073	Holding Reg	R/W	Working registers	Rejestry te służą do zapisania komendy / odczytania odpowiedzi. Jeden rejestr przechowuje wartość 1 bajta komendy/odpowiedzi z ramki Netronix

8.3.1 WORKFLOW

1. Write to the MODBUS registers the command from the Netronix protocol according to the following scheme:



Netronix Protocol Command Frame

Adr	Len	Cmd	P[0]	P[1]	CRC	CRC
--	07	A0	11	22	--	--

MODBUS RTU Register

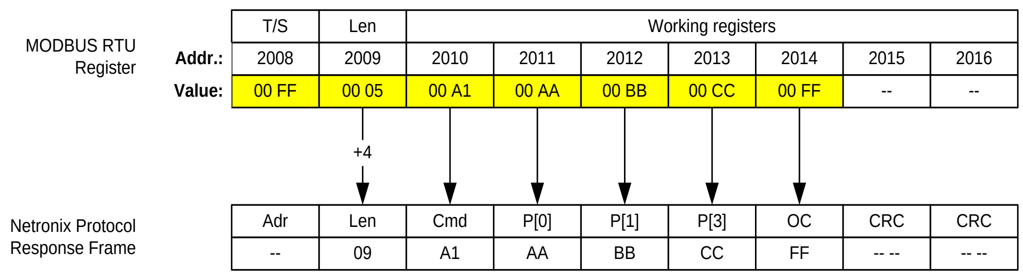
	T/S	Len	Working registers						
Addr.:	2008	2009	2010	2011	2012	2013	2014	2015	2016
Value:	00 00	00 03	00 A0	00 11	00 22	-- --	-- --	-- --	-- --

2. Write the value 0x0001 to the Trigger/Status registry

MODBUS RTU Register

	T/S	Len	Working registers						
Addr.:	2008	2009	2010	2011	2012	2013	2014	2015	2016
Value:	00 01	00 03	00 A0	00 11	00 22	-- --	-- --	-- --	-- --

3. Read the Trigger/Status register until the value 0x00FF appears in it. The value 0x00FF means that the answer is ready and can be read from the MODBUS registers.



	T/S	Len	Working registers						
Addr.:	2008	2009	2010	2011	2012	2013	2014	2015	2016
Value:	00 FF	00 05	00 A1	00 AA	00 BB	00 CC	00 FF	--	--

Adr	Len	Cmd	P[0]	P[1]	P[3]	OC	CRC	CRC
--	09	A1	AA	BB	CC	FF	-- --	-- --

8.3.2 EXAMPLE OF USE - READING THE FIRMWARE VERSION

Assumptions:
Reader logical address (Netronix / Modbus RTU protocol) - 0x01.

1. Specify what the frame should look like in the Netronix protocol. Only the Cmd field and parameters are relevant. The entire frame for the C_FirmwareVerison command looks like this:

Adr	Len	Cmd	CRC	
0x01	0x05	0xFE	0xC6	0x14

   Cmd = 0xFE
   Param – none

2. The amount of data should be entered in the ***Len*** register and the command code (and optional parameters) should be entered in the ***Working Registers***:

```
 RTU Tx > 01 10 07D8 0002 04 0001 00FE 0925
 RTU Rx > 01 10 07D8 0002 C087

Len = 0001
WorkingRegister[0] = 00FE
```

3. Write the value 0x0001 to the ***Trigger / Status*** register. This will trigger the execution of the command stored in the Working Registers.

```
 RTU Tx > 01 06 07D7 0001 F946
 RTU Rx > 01 06 07D7 0001 F946

Trigger/Status=0001
```

4. Then read the ***Trigger/Status*** register until the value 0x00FF is read. The value 0x00FF means that the command has been carried out and the ***Working Registers*** have the answer.

```
 RTU Tx > 01 03 07D7 0001 3546
 RTU Rx > 01 03 02 00FF F804

Trigger/Status=00FF
```

5. Then read the ***Len*** register. This register contains information about the number of registers in which the response is stored.

```
 RTU Tx > 01 03 07D8 0001 0545
 RTU Rx > 01 03 02 0011 7848

Len=0011
```

6. In the last step, read the Len first working registers.

```
 RTU Tx > 01 03 07D9 0011 5549
 RTU Rx > 01 03 22
          00FF
          004D 0057 002D 0052 0037 002D 0056 0033 002E 0032 002E 0041 0031 002E 0035
          00FF
          8E C6

00FF - Command code +1 (response to C_FirmwareVersion)
004D 0057 002D … 0031 002E 0035 – firmware version – „MW-R7-v3.2.A.1.5”
00FF - Operation code (success)
```

9 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

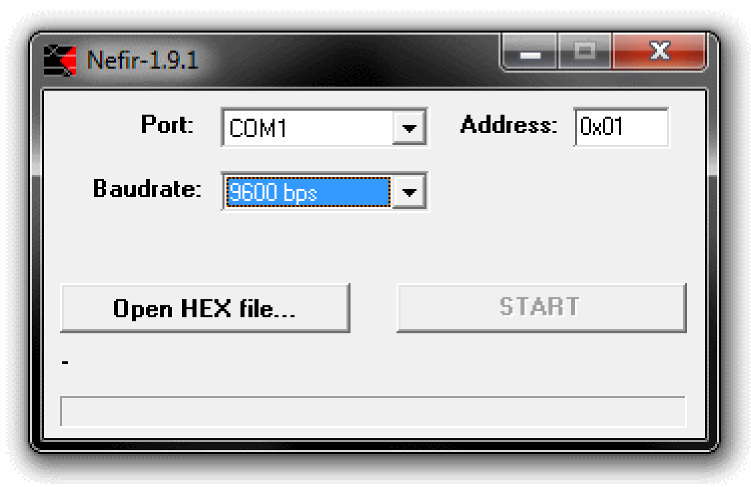
Parameter name or functionality	Value or setting	
Interface		
RS232 interface	Address: 0x01 Speed: 0x03	 9600bps
RS485/CAN interface	Adress: 0x01 Speed: 0x03 Type: 0x00	 9600bps RS-485
1-WIRE interface	Family: 0x01 Address: 0x00	
Wiegand interface	Number of bits 37	
Read-out transponder		
AAutoreader	Triger: 0x02 Timeout: 0x14 Mode: 0xFF ASerial: 0x01 AMode: 0x40 ABuzzer: 0x01 AMulti: 0x09 AInterface: 0x00	 2s all supported types for the first touch Netronix format, extended information about collision signaling and card type For the first touch MIFARE + I-CODE 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 C1: BLUE C2: WHITE 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 Time: 0, Time0: 0, Time1: 0	
SigB1	Source: „0”, Mode: 2, Negation: 1 Time: 0, Time0: 0, Time1: 0	
SigB2	Source: „0”, Mode: 2, Negation: 1 Time: 0, Time0: 0, Time1: 0	
SigB3	Source: „0”, Mode: 2, Negation: 1 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

10 BOOTLOADER – CHANGING DEVICE'S FIRMWARE

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

1. Connect device to serial port on computer
   - RS232 – for MW-R7x
   - RS485 – for MW-R4x
2. Open NEFIR.exe program
3. Set the appropriate COM port and transmission speed to 9600bps
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.



Port: COM1
Address: 0x01
Baudrate: 9600 bps

Drawing 9.2 Program window view when reloading firmware