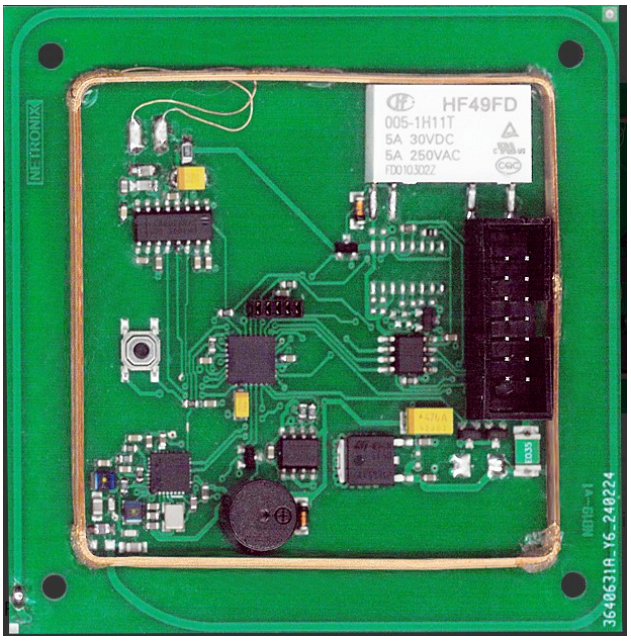
USER MANUAL

RFID READER

CTU-S5R



1 INTRODUCTION

The CTU-S series reader is an OEM RFID card reader operating at the rated frequency of 13.6MHz and 125kHz.

It has the following functionality:

- Supported transponder types:
  - 13.56MHz: Mifare S50, Mifare S70, Mifare Ultra Light, Mifare DesFire, I-CODE SLI, iCLASS,
  - 125kHz: EM4102, Hitag 1/s, HID PROX II, Hitag 2
  - NFC ID - unique ID from Android NFC ID application
  - USER ID – ID from secured parts of Mifare Plus, Desfire, Classic
- Built-in antenna
- Card memory with built-in lock driver
- A variety of communication interfaces depending on the version (table below)
- Addressability on the RS-485 bus
- Built-in relay, buzzer
- Built-in button for configuration / return to factory settings
- Configurable binary inputs / outputs
- Configuring the behavior of the buzzer, relay
- Control of binary outputs
- Reading binary inputs
- Ability to configure the format of the sent ID number
- Password-protected transmission
- Updating the software via the communication interface
- Available version (CTU-R5RL) with sleep mode

The CTU-S model family

Module model	GPIO	Card memory	Event memory	Relay	Supply voltage	INTERFACES							
						RS-232	RS-485	RS-232TTL	SPI	I2C	WIEGAND	1-WIRE	UART TTL
CTU-S2R	2	1000	3400	✓	5-16	✓							
CTU-S5R*	**2**	**1000**	**3400**	✓	**5-16**		✓	✓	✓	✓	✓	✓	✓
CTU-S5L	1	1000	3400	✓	5			✓	✓	✓	✓	✓	✓

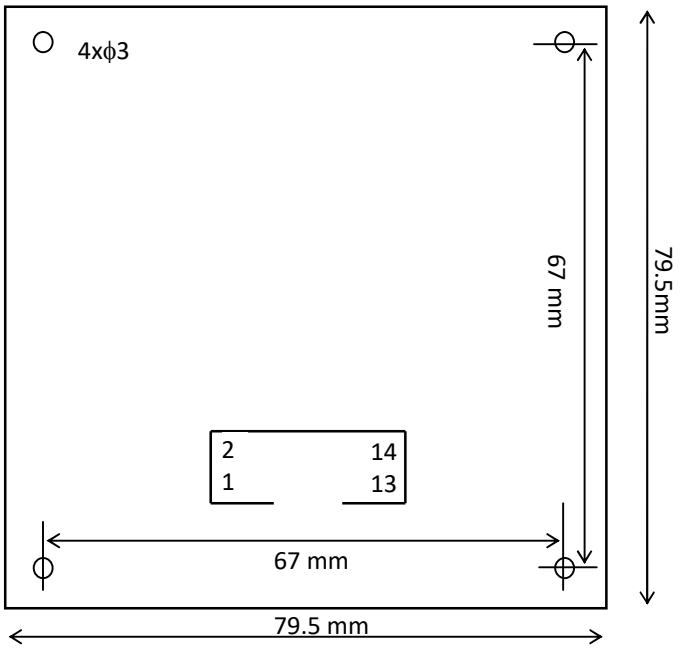
* - standard version, other versions on request

2 TECHNICAL DATA

Supported functionality depending on the type of transponder / card:		
Card type	Reading the ID number	Full writing and reading of memory blocks
MIFARE® Classic S50, S70, Ultralight	YES	YES
MIFARE® Plus	YES	SL0,SL1,SL3
MIFARE® DESFire	YES	YES
I-CODE SLI	YES	YES
ICLASS	YES (CSN only)	NO
EM4102	YES	-
Q5	YES	YES
Hitag-1/s	YES	YES (plain communication)

CTU-Sxx module parameters	
Supply voltage	5-16 V (CTU-R5RM / CTU-R2RM) 5V (CTU-R5RL)
Maximum current consumption	170 mA
Power consumption in sleep mode (CTU-S5RL)	16uA
Rated RF frequency of the module operation	13,56 MHz
Operating temperature	-20°C to +65°C
Permitted relay current	3A
Permissible voltage of the relay	30V
Read range	up to 15 cm
Max. GPIO current	20mA
Max. GPIO voltage	5V
RS232 / RS485 / RSTTL / UART TTL transmission parameters	2400, 4800, 9600, 19200, 38400, 57600, 115200 b/s, 8 data bits, 1 stop bit, no parity bit Netronix or MODBUS RTU protocol
Address on the I^2C bus	0xC0

3 DIMENSIONS, DESCRIPTION OF ELECTRICAL OUTPUTS



Pic.2 Top view

Pin number	Description of the connector
1	RS232RX, RS485B, RSTTL_RX, UART_RX, 1-WIRE, MOSI, SDA, WIEGAND1
2	RS232TX, RS485A, RSTTL_TX, UART_TX, MISO
3	SCK, SCL,WIEGAND0
4	CS
5	N.C.
6	GND
7	VCC
8	GPIO 1
9	GPIO 2 / SLEEP(CTU-R5RL)
10	GND
11	N.C.
12	N.C.
13	RELAY 1
14	RELAY 2

4 CONFIGURATION BY BUILT-IN BUTTON

There is a button on the board that have two functions:

1. Return to factory settings - pressing the button for at least 8 seconds
2. Selecting the interface according to the scheme below:

STEP	Number of presses:	1	2	3	4	5	6	7	8	9	10
1	MENU1 transponder selection	-	Mifare ISO14443A	EM4102 Unique	HID PROX II	iCLASS CSN	ISO15693 I-CODE	HITAG 1/S	HITAG 2	ISO14443B	all
2	Double buzzer confirmation signal										
3	MENU2 interface selection*	-	RS232 TTL	UART TTL	WIEGAND **	DALLAS ALL	DALLAS SINGLE	RS485	I2C	SPI	WIEGAND 26-SLAM CONFIG
4	Triple buzzer confirmation										

* - the interface type depends on the module type

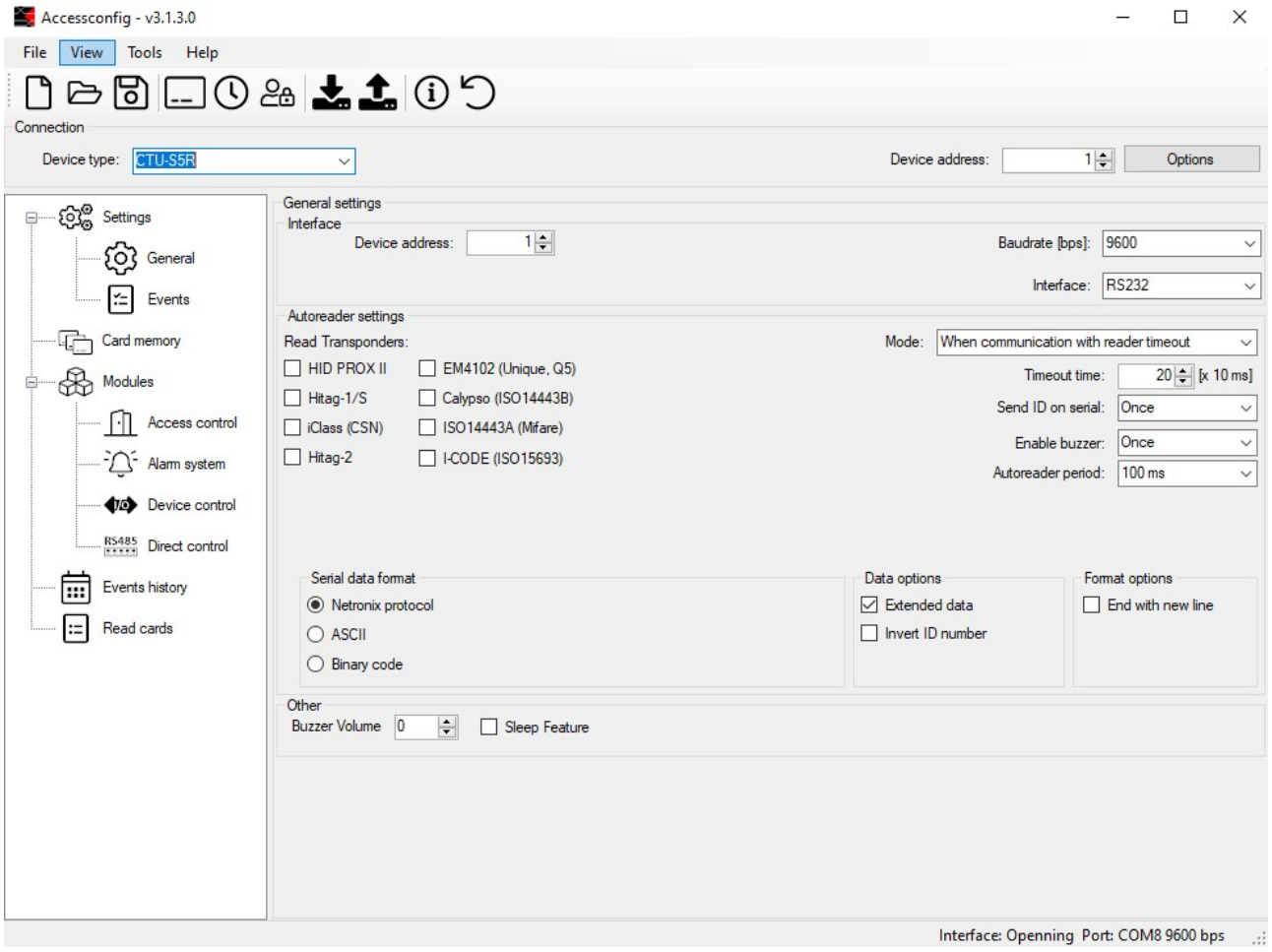
** - Wiegand 37 is default settings. Changes can be made using AccessConfig tool

Example:

To set Unique transponder and Wiegand interface:
-press button 3 times,
-wait for double buzzer signal,
-press button 4 times,
-wait for triple buzzer signal

5 CONFIGURATION BY ACCESSCONFIG TOOL

Reader can be configured using tool AccessConfig available on our webpage. Communication interface can be RS232, UART or RS485, depends on configuration choice by button. Default interface is RS232.



6 INTERFACE

6.1 RS232 / RS485/ UART INTERFACE

6.1.1 COMMUNICATION PROTOCOL FOR THE RS232 / RS485/ UART INTERFACE

The Netronix protocol or the MODBUS RTU protocol is used for communication via the RS232 / RS485 / UART interface.

In this documentation, the description of the RS-232/485 protocol has been limited to the description of commands and responses and their parameters. The header and the CRC checksum are always present and are consistent with the full „Netronix Protocol" documentation available at netronix.pl website. The default settings of communication parameters are 9600.8 bits, 1 stop bit, no parity bit. The baud rate can be changed with the C_SetInterfaceConfig command described later in the documentation.

Command frame:

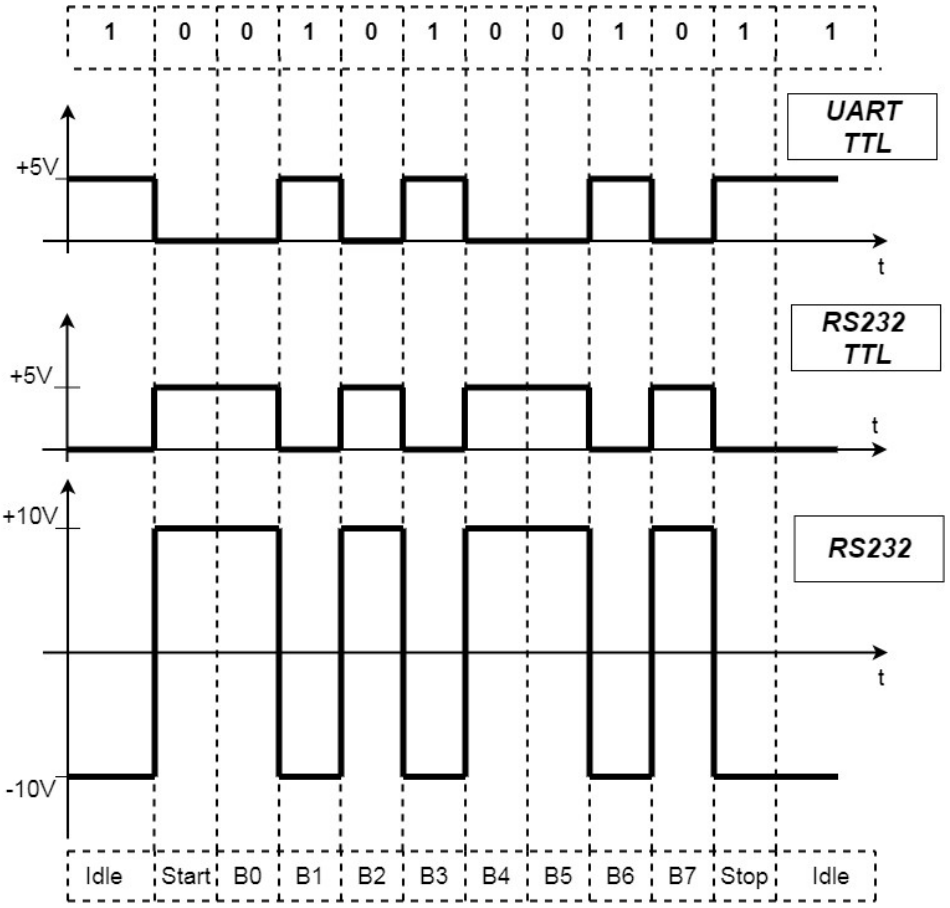
Adress	Lenght	C_CommandName	Command_parameter1…n	CRC

Response frame:

Adress	Lenght	C_CommandName +1	Response_parameter1…m	OperationCode	CRC

Operation with the RS protocol can be tested with the free utility software called FRAMER.

6.1.2 WAVEFORMS FOR RS232 AND UART SERIAL DATA TRANSMISSION



Pic 3. Waveforms for UART and RS232 interface outputs

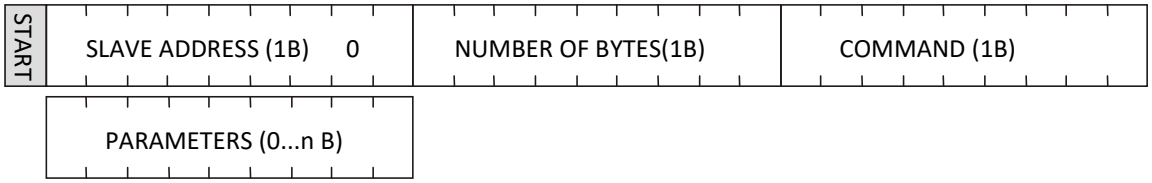
6.2 I²C INTERFACE

6.2.1 DATA EXCHANGE ALGORITHM

After configuring according to section 4, the CTU-R5RM module operates in the I2C interface mode in the following sequences:

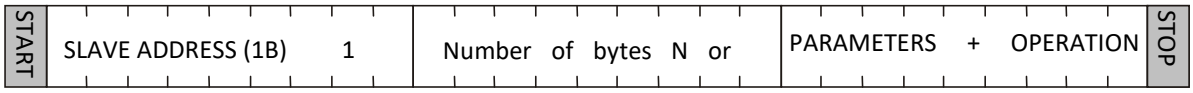
1. The master (external device) writes the command together with any parameters in the slave (CTU module)
2. The order is executed. (immediately after receiving the number of sent bytes declared in the frame)
3. The master reads the answer, its parameters and the operation code. In case of receiving 0xCB busy byte, you should try to read the response after about 1ms (commands related to reading / writing to transponders may take up to 100ms)

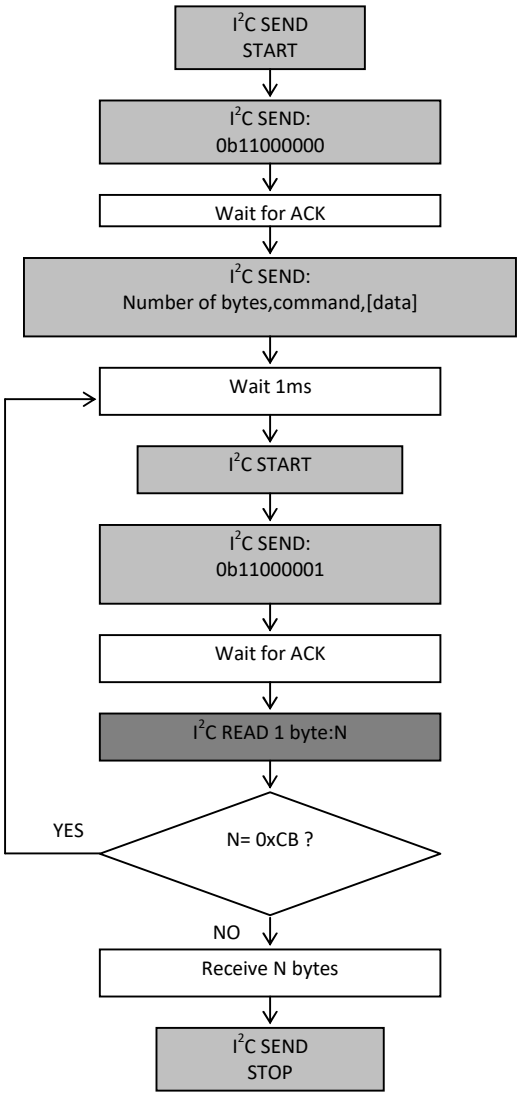
We write the command-question to the CTU module:



The „number of bytes" field must contain information about the number of bytes sent immediately after it, i.e. the sum of the „command" and „parameters" fields.

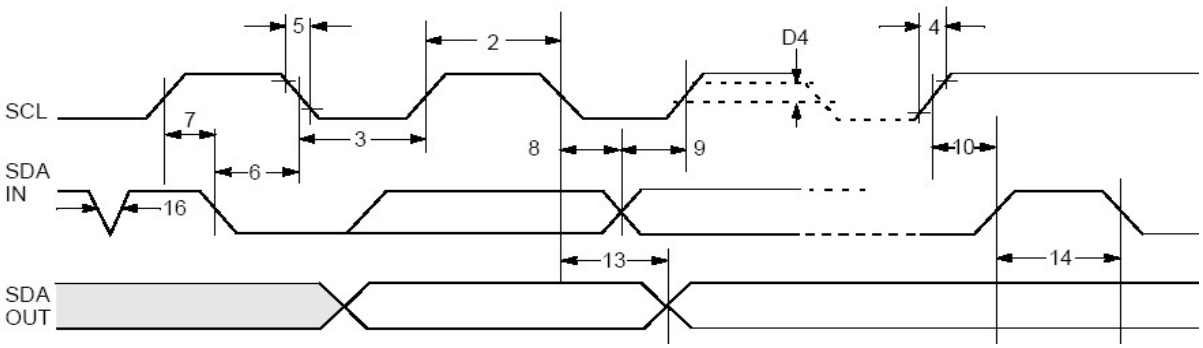
Then we get:





6.2.2 TIME DEPENDENCIES

The module sends and receives data with a clock frequency of up to 400KHz, taking the time dependencies presented below.



Param. No.	Sym.	Characteristic	Min.	Max.	Units
1	FCLK	Clock Frequency	— —	400 100	kHz
2	THIGH	Clock High Time	600 4000	— —	ns
3	TLOW	Clock Low Time	1300 4700	— —	ns
4	TR	SDA and SCL Rise Time **(Note 1)**	— —	300 1000	ns
5	TF	SDA and SCL Fall Time	—	300	ns
6	THD:STA	Start Condition Hold Time	600 4000	— —	ns
7	TSU:STA	Start Condition Setup Time	600 4700	— —	ns
8	THD:DAT	Data Input Hold Time	0	—	ns
9	TSU:DAT	Data Input Setup Time	100 250	— —	ns
10	TSU:STO	Stop Condition Setup Time	600 4000	— —	ns
11	TSU:WP	WP Setup Time	600 4000	— —	ns
12	THD:WP	WP Hold Time	1300 4700	— —	ns
13	TAA	Output Valid from Clock **(Note 2)**	— —	900 3500	ns
14	TBUF	Bus free time: Time the bus must be free before a new transmission can start	1300 4700	— —	ns
15	TOF	Output Fall Time from VIH Minimum to VIL Maximum	20+0.1CB —	250 250	ns

Note2: The reader holds the first clock pulse of each byte sent low until a valid state is displayed on the SDA line

6.3 1-WIRE INTERFACE

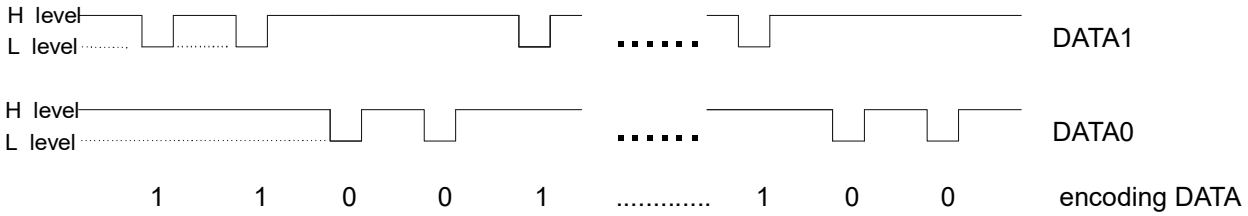
After configuring the device to work in the 1-WIRE mode, the reader emulates the behavior of the Dallas DS1990 series „chip". As long as the card is in the field, the reader will issue a unique number on the 1-WIRE bus. The time between successive readings of the transponder is 150ms, so presense pulses should occur at least once every 150ms. The format of the sent ID is as follows:

Family code	ID	Adress	CRC_DAL
01	ID1…ID5	01	XX

6.4 WIEGAND INTERFACE

The reader, after being configured to work in the WIEGAND mode, sends a unique ID number of the read card in accordance with the Wiegand 37 protocol with the following parameters:

Pulse duration (L level) 100us
Interval between pulses (H level) 1ms



7 COMMANDS AVAILABLE FOR THE RS232 / UART / RS485 INTERFACE - NETRONIX PROTOCOL

7.1 KEY MANAGEMENT

Key management comes down to saving the keys to the internal key memory. These keys cannot be read for security purposes. In order to maintain the highest data security, there is a certain correct philosophy of working with the keys. It consists in writing down the keys by individuals or persons having the highest level of trust. This record takes place only once or very rarely. The work of the reader in a specific application does not consist in using the key directly, but in calling the appropriate key number in order to log into the sector. Thus, in a particular application, the key does not actually appear on the data bus.
Additionally, the user should ensure that the key has the appropriate access rights to sectors. This is done through the card initialization process, where new secret keys are written to cards along with the appropriate access rights assigned to these keys.
Each sector of the transponder is assigned an A key and a B key. The C_LoadKeyToSKB and C_LoadKeyToDKB commands write the keys to the reader memory without any information about the key type (A or B). When logging into the sector, the user must enter 0xAA or 0xBB as the parameter if he wants the called key to be treated as A or B.

7.1.1 WRITING THE KEY TO THE DYNAMIC KEY MEMORY

Dynamic memory is characterized by an automatic erasing of its contents in the event of a power failure. Its content can be overwritten many times.

Command frame:

header	C_LoadKeyToDKB	Key1…6	CRC

Where:

Parameter name	Parameter description	Range of values
C_LoadKeyToDKB	Writing the key to the dynamic key memory	0x14
Key1…6	6 byte key	any

Response frame:

Header	C_LoadKeyToDKB +1		OperationCode	CRC

7.1.2 WRITING THE KEY TO THE STATIC KEY MEMORY

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

Command frame:

header	C_LoadKeyToSKB	Key1…6, KeyNo	CRC

Where:

Parameter name	Parameter description	Value range
C_LoadKeyToSKB	Writing the key to the static key memory	0x16
Key1…6	6 byte key	any
KeyNo	Key number. Up to 32 different keys can be saved in the reader.	0x00…0x1f

Response frame:

header	C_LoadKeyToSKB +1		OperationCode	CRC

7.2 COMMANDS FOR COMMUNICATION WITH TRANSPONDERS

7.2.1 ENABLING AND DISABLING A READER FIELD

Command frame:

Header	C_TurnOnAntennaPower	State	CRC

Where:

Parameter name	Parameter description	Value range
C_TurnOnAntennaPower	Enabling and disabling the reader field	0x10
State	Activation state	0x00 – disabling field and autoreader* 0x01 – enabling field and autoreader*

* if autoreader is in 'always on' mode

Response frame:

Header	C_TurnOnAntennaPower +1		OperationCode	CRC

7.2.2 SELECTION OF ONE MIFARE TRANSPONDER FROM MANY

Command frame:

Header	C_Select	RequestType	CRC

Where:

Parameter name	Parameter description	Value range
C_Select	Selection of one transponder from many	0x12
RequestType	Method of selecting a transponder	0x00 - Standard selection of transponders from the group of those that are not dormant, 0x01 - Selecting transponders from the group of all those in the reader field.

Response frame:

Header	C_Select +1	ColNo, CardType, ID1…….IDn	OperationCode	CRC

Where:

Parameter name	Parameter description	Value description
ColNo	Number of collisions when selecting one transponder. This number may indicate how many non-dormant transponders are in the field at the same time.	
CardType	Selected transponder type	For Primary Select: 0x50 – S50 0x70 – S70 0x10 – UltraLight 0xdf – DesFire 0xCA – iClass 0xE0 – I-CODE SLI 0x01 – EM4102 0x03 – Hitag-1/S 0x04 – HID PROX II 0x05 – Hitag-2
ID1…IDn	Unique transponder number	ID1 – LSB, IDn – MSB

7.2.3 READOUT OF RECENTLY READ TRANSPONDER ID

The command makes it possible to read the last 'seen' transponder ID buffered in the memory. The buffer is cleared after calling this command.

If this command is cyclically sending to obtain new ID, Aserial parameter in C_SetAutoreaderConfig command must be modified to 'off' to avoid collision on serial interface.

Command frame:

Header	C_GetHistID		CRC

Where:

Nazwa parametru	Parameter description	Value description
TagType	Selected transponder type	0x50 – S50 0x70 – S70 0x10 – UltraLight 0xDF – DesFire 0xCA – iClass 0xE0 – I-CODE SLI 0x01 – EM4102 0x03 – Hitag-1/S 0x04 – HID PROX II 0x05 – Hitag-2
ID1…IDn	Unique transponder number	ID1 – LSB, IDn – MSB

Response frame:

Header	C_ GetHistID +1	ID1...IDn	OperationCode	CRC

7.2.4 LOGGING INTO A SELECTED TRANSPONDER SECTOR USING THE DYNAMIC KEY BUFFER

For the login to be successful, it is necessary to reload the Dynamic Key Buffer each time the reader is turned on.

Command frame:

Header	C_LoginWithDKB	SectorNo, KeyType, DKNo	CRC

Where:

Parameter name	Parameter description	Value range
C_LoginWithDKB	Logging to the sector	0x18
SectorNo	Transponder sector number to which the user wants to log in	**BlockNumberingAndSectors
KeyType	The type of key that is contained in the internal Dynamic Key Buffer	0xAA –klucz typu A 0xBB – klucz typu B
DKNo	Dynamic key number	0x00

Response frame:

Header	C_LoginWithDKB +1		OperationCode	CRC

7.2.5 LOGGING INTO THE SELECTED TRANSPONDER SECTOR USING THE STATIC KEY BUFFER

For the login to be successful, it is necessary to load the Static Key Buffer first.

Command frame:

Header	C_LoginWithSKB	SectorNo, KeyType, SKNo	CRC

Where:

Parameter name	Parameter description	Value range
C_LoginWithSKB	Logging to the sector	0x1a
SectorNo	Transponder sector number to which the user wants to log in	**BlockNumberingAndSectors
KeyType	The type of key that is contained in the internal Dynamic Key Buffer	0xAA - key type A 0xBB - key type B
SKNo	Static key number	0x00…0x1F

Response frame:

Header	C_LoginWithSKB +1		OperationCode	CRC

7.2.6 READING THE CONTENTS OF A TRANSPONDER BLOCK

Command frame:

Header	C_ReadBlock	BlockNo	CRC

Where:

Parameter name	Parameter description	Value range
C_ReadBlock	Reading the contents of a transponder block	0x1e
BlockNo	Block number within a given sector	**BlockNumberingAndSectors

Response frame:

Header	C_ReadBlock +1	Data1….. Data16	OperationCode	CRC

Where:

Parameter name	Parameter description	Value range
Data1…. Data16	Data read from the transponder block	

7.2.7 WRITING THE CONTENTS OF THE TRANSPONDER BLOCK

Command frame:

Header	C_WriteBlock	BlockNo, Data1….. Data116	CRC

Where:

Parameter name	Parameter description	Value range
C_WriteBlock	Writing the contents of the transponder block	0x1c
BlockNo	Block number within a given sector	**BlockNumberingAndSectors
Data1…. Data16	Data to be written in the transponder block	any

Response frame:

Header	C_WriteBlock +1		OperationCode	CRC

7.2.8 COPYING THE CONTENTS OF A TRANSPONDER BLOCK TO ANOTHER BLOCK

Command frame:

Header	C_CopyBlock	SourceBlockNo, TargetBlockNo	CRC

Where:

Parameter name	Parameter description	Value range
C_CopyBlock	Copying the contents of a transponder block to another block	0x60
SourceBlockNo	Source block	**BlockNumberingAndSectors
TargetBlockNo	Target block for data	

Response frame:

Header	C_CopyBlock +1		OperationCode	CRC

7.2.9 WRITING VALUES TO THE TRANSPONDER BLOCK

Command frame:

Header	C_WriteValue	BlockNo, BackupBlockNo,Value1...4,	CRC

Where:

Parameter name	Parameter description	Value range
C_WriteValue	Writing values to the transponder block	0x34
BlockNo	Block number within a given sector in which the Value will be written	**BlockNumberingAndSectors
BackupBlockNo	Declared block number containing a copy of the Value. BackupBlockNo is not essential for the operation of the system and the user can / should see a copy of the Value himself.	**BlockNumberingAndSectors
Value1...4	Value written to the transponder block	any

Response frame:

Header	C_WriteValue +1		OperationCode	CRC

7.2.10 READING THE VALUES FROM THE TRANSPONDER BLOCK

Command frame:

Header	C_ReadValue	BlockNo	CRC

Where:

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

Response frame:

Header	C_ReadValue+1	Value1...4, BackupBlockNo	OperationCode	CRC

Where:

Parameter name	Parameter description	Value range
Value1...4	Value read from the transponder block	
BackupBlockNo	Block number that may contain a copy of the Value	**BlockNumberingAndSectors

7.2.11 INCREASE THE VALUE CONTAINED IN THE TRANSPONDER BLOCK

In order for the command execution to bring correct results, the data in the declared block must have the "Values" format.

Command frame:

Header	C_IncrementValue	BlockNo, Value1...4	CRC

Where:

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

Response frame:

Header	C_IncrementValue +1		OperationCode	CRC

7.2.12 DECREASE THE VALUE CONTAINED IN THE TRANSPONDER BLOCK

In order for the command execution to bring correct results, the data in the declared block must have the "Values" format.

Command frame:

Header	C_DecrementValue	BlockNo, Value1...4	CRC

Where:

Parameter name	Parameter description	Value range
C_DecrementValue	Decrease the value contained in the transponder block	0x32
BlockNo	Block number within a given sector in which the Value will be modified	**BlockNumberingAndSectors
Value1...4	Value subtracted from the existing actual value of the transponder block	any

Response frame:

Header	C_DecrementValue+1		OperationCode	CRC

7.2.13 SLEEP TRANSPONDER IN THE FIELD

In order to put the transponder to sleep, it must first be selected.

Command frame:

Header	C_Halt		CRC

Response frame:

Header	C_Halt+1		OperationCode	CRC

7.2.14 SAVING THE CONTENT OF THE PAGE IN MIFARE UL

Command frame:

Header	C_WritePage4B	PageAdr, Data1...4	CRC

Where:

Parameter name	Parameter description	Value range
C_WritePage4B	Saving the content of the page in MIFARE UL	0x26
PageAdr	Page number in the transponder	0x00...0x0f
Data1...4	Data to be saved	any

Response frame:

Header	C_WritePage4B +1		OperationCode	CRC

7.2.15 READING THE CONTENT OF PAGES IN MIFARE UL

Command frame:

Header	C_ReadPage16B	PageAdr	CRC

Where:

Parameter name	Parameter description	Value range
C_ReadPage16B	Reading the content of pages in MIFARE UL	0x28

PageAdr	The address of the page that should start reading the next 4 pages. If PageAdr> 0x ???? the pages at the beginning of memory will be read.	0x00…0x0f

Response frame:

Header	C_ReadPage16B +1	Data1…16	OperationCode	CRC

Where:

Parameter name	Parameter description	Value range
Data1…16	Data read from 4 consecutive pages	any

7.2.16 AUTHENTICATION FOR THE ULTRALIGHT C TRANSPONDERSPONDERA ULTRALIGHT C

Attention! Authentication is possible only after saving the keys in the transponder memory.

Command frame:

Header	C_ULC_Auth	KeyIdx	CRC

Where:

Parameter name	Parameter description	Value range
C_ULC_Auth		0x3C
KeyIdx	Index of the key stored in the reader	0x00…0x1f

Response frame:

Header	C_ ULC_Auth +1		OperationCode	CRC

7.3 COMMANDS FOR COMMUNICATION WITH THE MIFARE PLUS TRANSPONDERS

7.3.1 SL0 LEVEL COMMANDS

7.3.1.1 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	MIFARE Plus common command	0x3A
0xA8	Sub-command ‘Write Perso’	
AdrH, AdrL	Two-byte block or key number to write	According to the documentation of the MIFARE Plus transponder
Data{0..15}	Key or data to write	any

Response frame:

Header	C_ MfPlusCMD +1		OperationCode	CRC

7.3.1.2 COMMIT PERSO – SWITCH TO NEXT SL LEVEL

Command frame:

Header	C_MfPlusCMD	0xAA	CRC

Where:

Parameter name	Parameter description	Value range
C_MfPlusCMD	MIFARE Plus common command	0x3A
0xAA	Sub-command ‘Commit Perso’	

Response frame:

Header	C_ MfPlusCMD +1		OperationCode	CRC

7.3.2 SL1 LEVEL COMMAND SET

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

7.3.2.1 SL1 AES AUTHORIZATION

Command frame:

Header	C_MfPlusCMD	0x10, KeyIdx	CRC

Where:

Parameter name	Parameter description	Value range
C_MfPlusCMD	MIFARE Plus common command	0x3A
0x10	Sub-command 'Authentication SL1'	
KeyIdx	Index of the AES key stored in the reader	0x00-0x1F

Response frame:

Header	C_ MfPlusCMD +1		OperationCode	CRC

7.3.2.2 SWITCH TO NEXT SL LEVEL / AUTHENTICITY CHECK

Upgrading to a higher SL level or checking for authenticity takes place after 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	MIFARE Plus common command	0x3A
0x70	Sub-command 'First Auth'	
AdrH, AdrL	Two-byte block or key number to write	0x9002 – switch to SL2 level 0x9003 – switch to SL3 level 0x8000 – verification of the authenticity of the transponder
KeyIdx	Index of the AES key stored in the reader	0x00-0x1F

Response frame:

Header	C_ MfPlusCMD +1		OperationCode	CRC

7.3.3 SL3 LEVEL COMMAND SET

7.3.3.1 ESTABLISHING ISO14443-4 MODE

Each SL3-related command must be preceded by a single 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 identifier	0x00

Response frame:

Header	C_Init_ISO14443-4+1		OperationCode	CRC

7.3.3.2 LOGIN INTO SECTOR

Command frame:

Header	C_MfPlusCMD	0x1A, Sector, KeyType, KeyIdx	CRC

Where:

Parameter name	Parameter description	Value range
C_MfPlusCMD	MIFARE Plus common command	0x3A
0x1A	Sub-command 'Sector Login'	
Sector	Sector number	0x00-0x1f – MIFARE Plus 2K 0x00-0x27 – MIFARE Plus 4k
KeyType	Key type	0xAA – key A 0xBB – key B
KeyIdx	Index of the AES key stored in the reader	0x00-0x1F

Response frame:

Header	C_ MfPlusCMD +1		OperationCode	CRC

7.3.3.3 READ BLOCK CONTENT

Command frame:

Header	C_ MfPlusCMD	read_cmd, block	CRC

Where:

Parameter name	Parameter description	Value range
C_MfPlusCMD	MIFARE Plus common command	0x3A
read_cmd	Read mode type:	0x30-0x33
block	Block number	0-3 for sector<32 0-15 for sector>32

Command	MAC on command	MAC on resonse	Plain /encrypted
0x30	Yes	No	Encrypted*
0x31	Yes	Yes	Encrypted*
0x32	Yes	No	Plan
0x33	Yes	Yes	Plan

* - only MIFARE Plus X transponders

Response frame:

Header	C_ MfPlusCMD +1	Data1….. Data16	OperationCode	CRC

Where:

Parameter name	Parameter description	Value range
Data1…. Data16	Data read from the transponder block	

7.3.3.4 WRITE BLOCK CONTENT

Command frame:

Header	C_ MfPlusCMD	write_cmd, block, data0..data15	CRC

Where:

Parameter name	Parameter description	Value range
C_MfPlusCMD	MIFARE Plus common command	0x3A
write _cmd	Write mode type:	0xA0-0xA3

Command	MAC on	MAC on	Plain

		command	**resonse**	**/encrypted**	
	0xA0	Yes	No	Encrypted*	
	0xA1	Yes	Yes	Encrypted*	
	0xA2	Yes	No	Plain	
	0xA3	Yes	Yes	Plain	
block	Block number				0-3 for sector<32 0-15 for sector>32
data0..data15	Data for writing the transponder block				

* - only MIFARE Plux X transponders

Response frame:

Header	C_ MfPlusCMD +1		OperationCode	CRC

7.3.4 OPERATION DURATION FOR MIFARE PLUS

The table below defines the duration of individual operations, counted from the end of sending the command frame (RS) to the start of sending the 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

7.4 HANDLING OF DESFIRE, DESFIRE EV1 TRANSPONDERS

7.4.1 AUTHORIZATION, LOGIN TO THE CURRENTLY SELECTED APPLIZATION

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 related to the transponder	0x00..0x10
KeyIdx	Index of the AES key stored in the reader	0x00-0x1F
AuthType	Authorization type: 0x0A – DES 0x3A – 3DES 0xAA – AES	0x0A, 0x3A, 0xAA

Response frame:

Header	C_DesAuth +1		OperationCode	CRC

7.4.2 CHANGING THE MASTER KEY SETTINGS KEY SETTINGS OF THE CURRENTLY SELECTED APPLICATION

Command frame:

Header	C_DesChangeKeySett (0x44)	KeySettings	CRC

Where:

Parameter name	**Parameter description**	**Value range**
C_DesChangeKeySett	The command to change the key settings	0x44
KeySettings	Configuration byte	0x00..0x0f

Response frame:

Header	C_DesChangeKeySett+1		OperationCode	CRC

Structure of the configuration *KeySettings* byte:

Bit	Description
0	0 - PICC Master key is not modifiable
	1* - PICC Master key is modifiable
1	0 - Calling the C_DesGetAppIDs function requires authorization using the PICC Master key
	1* - Calling the C_DesGetAppIDs function does not require authorization
2	0 - Creating / deleting an application requires authorization using the PICC Master key
	1* - Creating a new application does not require authorization, removing the application requires authorization with the application key or the PICC Master key
3	0 - PICC Master Key configuration cannot be changed
	1* - PICC Master Key configuration change allowed in case of authorization with this key
4	RFU - 0
5	RFU - 0
6	RFU - 0
7	RFU - 0

* - default setting

7.4.3 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 the new key stored in the reader's memory	0x00..0x13
PrevEESavedKey	• If the changed key is not the one with the current authorization, we provide the index of the current key to be changed • If the changed key is the same as the current authorization, leave this parameter blank	0x00..0x13

Response frame:

Header	C_DesChangeKey+1		OperationCode	CRC

7.4.4 CREATING AN APPLICATION

Command frame:

Header	C_DesCreateApp (0x48)	AId1..3,KeySettings1, KeySettings2	CRC

Where:

Parameter name	Parameter description	Value range
C_DesCreateApp	Create application command	0x48
AId1..3	3-byte application identifier	0x00..0xFF
KeySettings1	Configuration byte (see below)	0x00..0x0F
KeySettings2	Bit3..Bit0: The number of keys assigned to a given application Bit7..Bit6: 00 - DES authorization for the entire application 10 - AES authorization for the entire application	0x00..0x0D

Response frame:

Header	C_DesCreateApp +1		OperationCode	CRC

Structure of the configuration *KeySettings* byte:

Bit	Description
0	0 - The Application Master key is immutable
	1* - The Application Master key is modifiable, it requires authorization using the existing AppMasterKey
1	0 - Calling the C_DesGetAppIDs function requires authorization using the PICC Master key
	1* - Calling the C_DesGetAppIDs function does not require authorization
2	0 - Creating / deleting a file requires authorization with AppMasterKey
	1* - Creating / deleting a file does not require authorization with AppMasterKey
3	0 - Unable to reconfigure the Application Master Key
	1* - Reconfiguration of the Application Master Key allowed for authorization with this key
4	Bit7 - Bit4: określają prawa do zmian parametrów klucza
5	0x0*: Klucz Master aplikacji jest niezbędny do zmiany ustawień kluczy
6	0x1-0xD : Autoryzacja przy pomocy klucza z tym indeksem jest konieczna do zmiany ustawień kluczy
7	0xE : Zmiana ustawień klucza wymaga autoryzacji z użyciem tego samego klucza

* - default setting

7.4.5 REMOVING THE APPLICATION

Command frame:

Header	C_DesDeleteApp (0x4a)	AId1..3	CRC

Where:

Parameter name	Parameter description	Value range
C_DesDeleteApp	Application removal command	0x4a
AId1..3	3-byte application identifier	0x00..0xFF

Response frame:

Header	C_DesCreateApp +1		OperationCode	CRC

7.4.6 GETTING APPLICATION LIST

Command frame:

Header	C_DesGetAppIDs (0x4c)		CRC

Where:

Parameter name	Parameter description	Value range
C_DesGetAppIDs	Application list get command	0x4c

Response frame:

Header	C_DesGetAppIDs +1	N*{Aid3,Aid2,Aid1}	OperationCode	CRC

A list of the Aid numbers of currently existing applications is returned.

7.4.7 APPLICATION SELECT

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 identifier	0x00-0xff

Response frame:

Header	C_DesSelectApp+1		OperationCode	CRC

7.4.8 TRANSPONDER FORMATING

Command frame:

Header	C_DesFormatPICC (0x60)		CRC

Where:

Parameter name	Parameter description	Value range
C_DesFormatPICC	The command to format the transponder	0x60

Execution of this command requires authorization using the PICC Master key.

Response frame:

Header	C_DesFormatPICC +1		OperationCode	CRC

7.4.9 INITIALIZATION OF TRANSMISSION PROTOCOL WITH MIFARE DESFIRE TRANSPONDERS (ISO14443-4)

Command frame:

Header	C_DesInitProtocol (0x3e)	CID	CRC

Where:

Parameter name	Parameter description	Value range
C_DesInitProtocol	ISO1444-4 initialization	0x3E
CID	Logical number of the selected transponder	0x00-0x0E

This command must appear immediately after selecting the transponder with the C_Select command. The current version of the reader allows you to work with one DESFire transponder at the same time. The logical CID number does not matter at present, it is recommended to enter the number 0.

Response frame:

Header	C_DesInitProtocol +1		OperationCode	CRC

7.4.10 DOWNLOADING THE FILE LIST OF THE CURRENTLY SELECTED APPLICATION

Command frame:

Header	C_DesGetFileIDs (0x64)		CRC

Where:

Parameter name	Parameter description	Value range
C_DesGetFileIDs	Command for retrieving the file list	0x64

Response frame:

Header	C_DesGetAppIDs +1	N*FileNo	OperationCode	CRC

A list of the file numbers currently existing in the selected application is returned.

7.4.11 READING FILE PROPERTIES

Command frame:

Header	C_DesGetFileSett (0x66)	FileNo	CRC

Where:

Parameter name	Parameter description	Value range
C_DesGetFileSett	Command to get file properties	0x66
FileNo	File ID	0x00-0x0f

Response frame:

Header	C_DesGetAppIDs +1	File params...	OperationCode	CRC

Depending on the file type, 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

7.4.12 CREATING *STANDARD DATA FILES*

Command frame:

Header	C_DesCreateSTDataFile (0x68)	FileNo,ComSett,AccRight1..2,FileSize1..3	CRC

Where:

Parameter name	Parameter description	Value range
C_DesCreateSTDataFile	STD file creation command	0x68
FileNo	File ID	0..0x0F
ComSett	Transmission type: 0x01 - unencrypted 0x03 - encrypted DES	0x00,0x03
AccRight1..2	See the table below for the file access rights	0x00..0xff
FileSize1..3	3 byte file size in bytes, in the order LSB..MSB	0x00-0xff

Access rights bytes:

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 4 4-bit fields. Each field can contain values in the range 0x0 - 0xF

- Values from the range 0x0 - 0xD define the number of the key which will be authorized to perform the given operation,
- The value 0xE means that the operation does not require authorization,
- The value 0xF means that the operation cannot be accessed, regardless of the key used.

Response frame:

Header	C_DesCreateSTDataFile +1		OperationCode	CRC

7.4.13 CREATING *BACKUP DATA FILE* TYPES

Command frame:

Header	C_DesCreateBACKDataFile (0x6a)	FileNo,ComSett,AccRight1..2,FileSize1..3	CRC

Where:

Parameter name	Parameter description	Value range
C_DesCreateBACKDataFile	BACKUP file creation command	0x6a
FileNo	File ID	0..0x07
ComSett	Transmission type: 0x01 - unencrypted 0x03 - encrypted DES	0x00,0x03
AccRight1..2	File access rights	0x00..0xff
FileSize1..3	3 byte file size in bytes, in the order 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*.

Saving a *Backup Data* file must end with the C_DesCommit command.

7.4.14 CREATING LINEAR/CYCLIC RECORD FILE TYPES

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* creation command	0x6c
FileNo	File ID	0..0x0F
ComSett	Transmission type: 0x01 - unencrypted 0x03 - encrypted DES	0x00,0x03
AccRight1..2	File access rights	0x00..0xff
RecSize1..3	3 byte record size in bytes, in the order LSB..MSB	0x00-0xff
RecNumb1..3	3-byte parameter specifying the number of records, order LSB..MSB	
Cy/Li	0x0c - cyclic 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*.

7.4.15 DELETING A FILE

Command frame:

Header	C_DesDeleteFile (0x6e)	FileNo	CRC

Where:

Parameter name	Parameter description	Value range
C_DesDeleteFile	File delete command	0x6e
FileNo	File ID	0x00..0x0F

Response frame:

Header	C_DesDeleteFile+1		OperationCode	CRC

7.4.16 CHANGING FILE SETTINGS

Command frame:

Header	C_DesChangeFileSett (0x80)	FileNo, ComSett, AccRight1..2	CRC

Where:

Parameter name	Parameter description	Value range
C_DesChangeFileSett	Command for changing file properties	0x80
FileNo	File ID	0..0x0F
ComSett	Transmission type: 0x01 - unencrypted 0x03 - encrypted DES	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 when creating *Standard Data Files*.

7.4.17 DATA READING FROM STD/BACK DATA FILE

Command frame:

Header	C_DesReadData (0x82)	FileNo, Offset1..3, Length1..3	CRC

Where:

Parameter name	Parameter description	Value range
C_DesReadData	Read command from file	0x82
FileNo	File ID	0..0x0F
Offset1..3	3-byte parameter specifying the place from which we start reading the file, sequence LSB..MSB	0x00-0xFF
Length1..3	3-byte parameter specifying the number of bytes we want to read, order LSB..MSB (up to 58 bytes can be read at once)	0x00-0x3A

Response frame:

Header	C_DesReadData +1	n Bytes	OperationCode	CRC

7.4.18 DATA WRITE TO STD/BACK DATA FILE

Command frame:

Header	C_DesWriteData (0x84)	FileNo, Offset1..3,Data1..58	CRC

Where:

Parameter name	Parameter description	Value range
C_DesWriteData	File write command	0x84
FileNo	File ID	0..0x0F
Offset1..3	3 bytes offset LSB..MSB	0x00-0xFF
Data1..58	Data to write	0x00-0xFF

Response frame:

Header	C_DesWriteData+1		OperationCode	CRC

7.4.19 RECORD WRITE TO RECORD DATA FILE

Command frame:

Header	C_DesWriteRecord (0x86)	FileNo, Offset1..3,Data1..58	CRC

Where:

Parameter name	Parameter description	Value range

C_DesWriteRecord	Komenda zapisu rekordu	0x86
FileNo	File ID	0..0x0F
Offset1..3	3-byte parameter specifying the place from which we start writing, the order LSB..MSB (this value must be smaller than the size of a single record)	0x00-0xFF
Data1..58	Data that we intend to save to the file, (you can save up to 58 bytes at one time, 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

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

Command frame:

Header	C_DesReadRecord (0x88)	FileNo, WhichRecord1..3,NoOfRecords1..3	CRC

Where:

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

Response frame:

Header	C_DesReadRecord +1	Record data...	OperationCode	CRC

The amount of data read cannot exceed 58 bytes, hence the rule: **{NoOfRecords1..3} * record_size <58bytes**

7.4.20 CLEANING RECORD DATA FILE

Command frame:

Header	C_DesClearRecordFile (0x8a)	FileNo	CRC

Where:

Parameter name	Parameter description	Value range
C_DesClearRecordFile	Record file cleanup command	0x8a
FileNo	File ID	0..0x0F

Response frame:

Header	C_DesClearRecordFile+1		OperationCode	CRC

Note: This operation must end with the C_DesCommit command.

7.4.21 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

7.4.22 TRANSPONDER DESELECTION

Command frame:

Header	C_DesDeselect (0x8e)		CRC

Where:

Parameter name	Parameter description	Value range
C_DesDeselect	Transponder de-selection command	0x8e

Response frame:

Header	C_DesDeselect+1		OperationCode	CRC

7.5 I-BLOCK DATA TRANSMISSION T=CL ISO14443-4 PROTOCOL

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

Response frame:

Header	C_TranscIBlock+1	data	OperationCode	CRC

7.6 HANDLING OF I-CODE SLI TRANSPONDERS

7.6.1 READING THE ID NUMBER OF THE I-CODE SLI TRANSPONDER

Command frame:

Header	C_Inventory		CRC

Where:

Parameter name	Parameter description	Value range
C_Inventory	Reading the ID number	0x04

Response frame:

Header	C_Inventory +1	0,CardType,ID1...ID8	OperationCode	CRC

7.6.2 READING THE PAGE OF THE SLI TRANSPONDER

Command frame:

Header	C_SLIReadPage	PageAdr	CRC

Where:

Parameter name	Parameter description	Value range
C_ SLIReadPage	Reading the content of pages in MIFARE Ultralight	0x2C
PageAdr	The page address is compatible with the supported transponder type	

Response frame:

Header	C_ SLIReadPage +1	Data1…4	OperationCode	CRC

Where:

Parameter name	Parameter description	Value range
Data1…4	Data read	any

7.6.3 SAVE THE CONTENTS OF THE PAGE IN SLI

Command frame:

Header	C_SLIWritePage	PageAdr, Data1...4	CRC

Where:

Parameter name	Parameter description	Value range
C_ SLIWritePage	Saving the page content in SLI	0x2E
PageAdr	Page number in the transponder	
Data1...4	Data to be saved	any

Response frame:

Header	C_ SLIWritePage +1		OperationCode	CRC

7.7 READING THE ID NUMBER OF THE ICLASS TRANSPONDER

Command frame:

Header	C_ICLASSInventory		CRC

Where:

Parameter name	Parameter description	Value range
C_ICLASSInventory	Odczyt numeru ID	0x0A

Response frame:

Header	C_ICLASSInventory +1	0,CardType,ID1...ID8	OperationCode	CRC

* - only the CSN number can be read

7.8 ELECTRICAL INPUTS AND OUTPUTS

The reader has configurable inputs / outputs.

7.8.1 STATUS OF THE OUTPUT

Command frame:

Header	C_WriteOutputs	IONo, State	CRC

Where:

Parameter name	Parameter description	Value range
C_WriteOutputs	Record of the output state	0x70
IONo	I/O port number. The port should be configured as output	0x00-0x03
State	Desired exit state	0x00 lub 0x01

Response frame:

Header	C_WriteOutputs +1		OperationCode	CRC

7.8.2 READING THE ENTRY STATE

Command frame:

Header	C_ReadInputs	IONo	CRC

Where:

Parameter name	Parameter description	Value range
C_ReadInputs	Reading the status of the input	0x72
IONo	I/O port number. The port should be configured as input	0x0..0x1

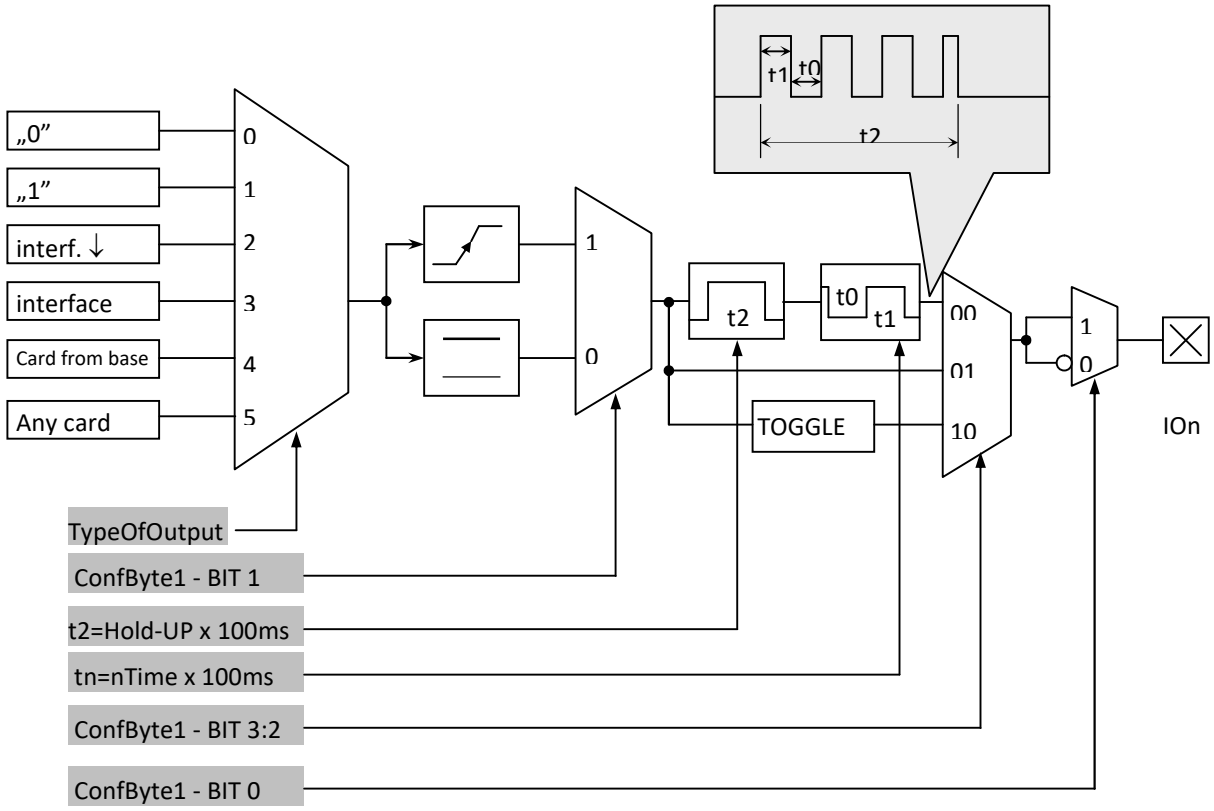
Response frame:

Header	C_ReadInputs +1	State,[COUNTER]	OperationCode	CRC

Where:

Parameter name	Parameter description	Value range
State	Read input state	
Counter	Counter status for the counter type input	

Configuration pattern for any port



Command frame:

C_SetIOConfig	IONo, IOConfigData1…n

If we configure the port as an output then the IOConfigData1 ... n parameters are as follows:
Dir, ConfByte1, TypeOfOutput, Hold-UP, 0Time, 1Time

Where:

Parameter name	Parameter description	Value range
C_SetIOConfig	Save anygo port configuration	0x50
IONo	I/O port number to be configured	0x0..0x3
Dir	Port direction	0x00 - exit
ConfByte1	One byte in which: BIT0: defines the output type as Normally Open or Normally Closed. BIT1: defines how a given output reacts as responsive to a change in arousal (slope responsive) or responsive to arousal (state responsive). BIT3: 2 defines how the output behaves in relation to the state of the trigger signal	ConfByte1 Bit 0 0 - normally Closed 1 - normally Open ConfByte1 Bit 1 0 - reacts to the level 1 - reacts to a slope ConfByte1 Bit 3: 2 00 - square wave generator 01 - direct 10 - change of output status
TypeOfOutput	Control signal source	0x00 - permanently disabled 0x01 - permanently on 0x02 - controlled through the serial interface, automatically returning to zero 0x03 - controlled through the serial interface 0x04 - RFU 0x05 - set when any card is applied to the reader

Hold	The time of maintaining the actuation state after the excitation ceases. This time is expressed as: Hold x 100ms During the "Hold" time, you can configure an output capable of generating a square wave. One time and zero time are set with the following parameters:
0Time	Logical zero time
1Time	Time of logical one

If we configure the port as input then the parameters IOConfigData1 ... n have the form:
Dir, Triger, TypeOfInput, RFU1, RFU2, RFU3

Where:

Parameter name	Parameter description	Value range
C_SetIOConfig	Save any port configuration	0x50
IONo	IO port number to be configured	0x0..0x3
Dir	Port direction	0x01 - input
Triger	A byte that specifies how the input is triggered	0x00 - not negated 0x01 - negated 0x02 - reaction to rising slope 0x03 - reaction to the falling slope
TypeOfInput	Input type: Standard - we get information about the input state taking into account the Triger setting Counter - a one-byte counter counts the number of edges that appeared at the input. The counter is read and reset with the C_ReadInputs command	0x00 "0" - permanently 0x01 "1" - permanently 0x02 - standard input 0x04 - counting input
RFU1-RFU3	Reserved	0x00

Not all CTU-Sxx ports have any direction.
For the correct configuration, the correct direction must be given for a given port.

List of existing ports that can be controlled in the CTU-R		
Port number	Direction	Description
0	Inuput/output	GPIO1
1	Inuput/output	GPIO2
2	output	RELAY
3	output	BUZZER

Response frame:

Header	C_SetIOConfig +1		OperationCode	CRC

7.8.3 READING OF ANY PORT CONFIGURATION

Command frame:

Header	C_GetIOConfig	IONo	CRC

Where:

Parameter name	Parameter description	Value range
C_GetIOConfig	Reading the configuration of each port	0x52
IONo	IO port number, the configuration of which is to be read	0x00…0x03

Response frame:

Header	C_GetIOConfig +1	IOConfigData1…n	OperationCode	CRC

Where:

Parameter name	Parameter description	Value range
IOConfigData1…n	Has the same form as when saving the configuration	

7.9 ACCESS PASSWORD

7.9.1 LOGGING TO THE READER

Command frame:

Header	C_LoginUser	Data1…n, 0x0	CRC

Where:

Parameter name	Parameter description	Value range
C_LoginUser	Login to the reader	0xb2
Data1…n	Is any byte string	Any in the range 0x01… 0xff. The string length can be 0 to 8 bytes
0x00	Zero ending the string	0x00

Response frame:

Header	C_LoginUser +1		OperationCode	CRC

7.9.2 CHANGE PASSWORD

Command frame:

Header	C_ChangeLoginUser	Data1…n, 0x0	CRC

Where:

Parameter name	Parameter description	Value range
C_ChangeLoginUser	Password change	0xb4
Data1…n	Is any byte string that will be valid password.	Any in the range 0x01…0xff. The string length can be 0 to 8 bytes
0x00	Zero ending the string	0x00

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

Response frame:

Header	C_ChangeLoginUser+1		OperationCode	CRC

7.9.3 LOG OUT FROM THE READER

This command will invalidate the last entered password.

Command frame:

Header	C_LogoutUser		CRC

Parameter name	Parameter description	Value range
C_LogoutUser	Logging out of the reader	0xd6

Response frame:

Header	C_LogoutUser +1		OperationCode	CRC

7.10 HANDLING OF INTERNAL TRANSPONDER MEMORY

7.10.1 READING THE TRANSPONDER NUMBER FROM MEMORY

Command frame:

Header	C_CardMemoryRead	AdrL, AdrH	CRC

Where:

Parameter name	Parameter description	Value range
C_CardMemoryRead	Reading the transponder number from the memory	0x20
AdrL, AdrH	The low and high byte of the address, respectively	0-999*

* The addresses from 0 to 989 are the addresses of the users' cards. The addresses 990-999 are the addresses of the Master cards.

Response frame:

Header	C_CardMemoryRead	+1	ID1(L)….ID5(H), Right	OperationCode	CRC

Where:

Parameter name	Parameter description	Value range
ID1(L)….ID5(H)	5 bytes of the transponder number	
Right	Access rights for a given transponder	0x01

7.10.2 SAVING THE TRANSPONDER NUMBER TO MEMORY

Command frame:

Header	C_CardMemoryWrite	AdrL, AdrH, ID1(L)….ID5(H), Right	CRC

Where:

Parameter name	Parameter description	Value range
C_CardMemoryWrite	Save transponder number to memory	0x22
AdrL, AdrH	The low and high byte of the address, respectively	0-999*
ID1(L)….ID5(H)	5 bytes of the transponder number	Any 5 bytes
Right	Access rights or the function performed by the transponder	0x01

* The addresses from 0 to 989 are the addresses of the users' cards. The addresses 990-999 are the addresses of the Master cards.

Response frame:

Header	C_CardMemoryWrite	+1		OperationCode	CRC

7.11 HANDLING OF BUILT-IN ACCESS CONTROL

7.11.1 ACCESS CONTROL CONFIGURATION RECORD

Command frame:

Header	C_AccesControllConfigWrite	Mode	CRC

Where:

Parameter name	Parameter description	Value range
C_AccesControllConfigWrite	Access control configuration record	0x74
Mode	Access control module operating mode	0x00 - module turned off 0x01 - module is on

Response frame:

Header	C_AccesControllConfigWrite+1		OperationCode	CRC

7.11.2 READING ACCESS CONTROL CONFIGURATION

Command frame:

Header	C_AccesControllConfigRead		CRC

Where:

Parameter name	Parameter description	Value range
C_ AccesControllConfigRead	Access control configuration readout	0x76

Response frame:

Header	C_AccesControllConfigRead+1	Mode	OperationCode	CRC

Where:

Parameter name	Parameter description	Value range
Mode	Access control module operating mode	0x00 - module turned off 0x01 - module is on

7.11.3 SAVING THE AUTOMATIC CONFIGURATION

This command configures the way of operation of the automatic unit reading the unique UID transponder number.

The described reader makes it possible to temporarily suspend the automatic unit operation in the case of correct transmission on the RS link.

If the reader will work in mixed mode, i.e..
- the UID readout machine is running, and:
- master device (computer, controller) communicates with the reader or by means of a reader with transponders

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

Command frame:

Nagłówek	C_SetAutoReaderConfig	ATrig, AOfflineTime, Aserial, AMode, Abuzz, AMulti	CRC

Where:

Parameter name	Parameter description	Value range
C_SetAutoReaderConfig **0x58**	Saving the automat configuration	0x58
ATrig	Bits:0-3 Defines when the automat UID readout is to run Bits: 4-7 Defines Tag scan period	Bits:0-3 0 - automat is turned off permanently 1 - automat switch is on permanently 2 - it turns on automatically when there is no transmission on RS / USB for longer than AOfflineTime 3 - it turns on automatically when there is no transmission on RS / USB for longer than AOfflineTime Bits:4-7 –*n value*, Tag scan period: For n=0: default 250ms, For n>0: period = 2^{n-1} * 50ms
AOfflineTime	Time of no transmission on RS / USB T= AofflineTime * [100 ms] Lack of transmission may concern any commands (Atrig = 2), or commands to	0x00…0xff

<table>
<tr><td></td><td>communicate with the transponder (Atrig = 3).
Commands to communicate with the transponder include:
C_TurnOnAntennaPower
C_Select
C_LoginWithDKB
C_LoginWithSKB)
C_ReadBlock
C_WriteBlock
C_CopyBlock
C_WritePage4B
C_ReadPage16B
C_IncrementValue
C_DecrementValue
C_WriteValue
C_ReadValue
C_Halt</td><td></td></tr>
<tr><td>ASerial</td><td>Automatic sending of the UID transponder number after its automatic reading from the transponder</td><td>0 - never
1 - only the first time the transponder is applied
2 - send all</td></tr>
<tr><td>AMode</td><td>Wybór formatu wysyłanego numeru
MSB LSB
<table><tr><td>R</td><td>R</td><td>R</td><td>CR</td><td>M</td><td>E</td><td>I</td><td>A</td></tr></table></td><td><table><tr><td>R</td><td>reserved, always 0</td></tr><tr><td>CR=1</td><td>the number ends with a line break CR + LF</td></tr><tr><td>M=1</td><td>the number begins with an "M"</td></tr><tr><td>E=1</td><td>information extended by the number of cards in the field and the card type (only UW-M4x</td></tr><tr><td>I=1</td><td>reverse order number</td></tr><tr><td>A=1</td><td>number sent in ASCII format</td></tr><tr><td>A=0</td><td>number sent in Nertonix format</td></tr></table></td></tr>
<tr><td>ABuzz</td><td>Automatic signaling of reading by means of a buzzer after the automatic reading of the UID from the transponder.</td><td>0 - never
1 - only the first time the transponder is applied
2 - signals all</td></tr>
<tr><td>AMulti</td><td>Selection of scanned transponders
MSB LSB
<table><tr><td>H2</td><td>HD</td><td>H1</td><td>U</td><td>S</td><td>I</td><td>B</td><td>M</td></tr></table></td><td>M- Mifare / NFC / ISO14443A
B- ISO14443B
I – IClass (CSN)
S – I-CODE (ISO15693)
U – Unique / EM4102 / Q5
H1 – Hitag-1/S
HD – HID Prox II
H2 – Hitag-2</td></tr>
</table>

Response frame:

Header	C_ SetAutoReaderConfig +1		OperationCode	CRC

7.11.4 READING OF AUTOMATIC CONFIGURATION

Command frame:

Header	C_ GetAutoReaderConfig		CRC

Where:

Parameter name	Parameter description	Value range
C_GetAutoReaderConfig	Readout of the machine configuration	0x5a

Response frame:

Header	C_ GetAutoReaderConfig +1	ATrig, AOfflineTime, ASerial, AMode, Abuzz, AMulti	OperationCode	CRC

Where:
The meaning of the response parameters is the same as described earlier.

7.11.5 DATE AND TIME SETTING

The following settings do not affect the operation of the reader.

Command frame:

Header	C_SetRtc	Year, Month, Day, Hour, Minute, Second	CRC

Where:

Parameter name	Parameter description	Value range
C_SetRtc	Date and time setting	0xb8
Year	year	0…99
Month	month	1…12
Day	day of the month	1…31
Hour	hour	0…23
Minute	minute	0…59
Second	second	0…59

Response frame:

Header	C_SetRtc +1		OperationCode	CRC

7.11.6 READING THE DATE AND TIME

Command frame:

Header	C_GetRtc		CRC

Where:

Parameter name	Parameter description	Value range
C_GetRtc	Reading the date and time	0xb6

Response frame:

Header	C_GetRtc+1	Year, Month, Day, Hour, Minute, Second	OperationCode	CRC

Where:
The meaning of the response parameters is the same as described earlier.

7.12 CONFIGURATION OF THE RS-232/485 SERIAL INTERFACE

7.12.1 SERIAL INTERFACE CONFIGURATION RECORD

Command frame:

Header	C_SetInterfaceConfig	Mode, Adr, Baudrate	CRC

Where:

Parameter name	Parameter description	Value range
C_SetInterfaceConfig	Saving the configuration of the serial interface	0x54
Mode		0x01
Adr	Address on the RS-485 bus	0x01…0xfe
Bodrate	Data speed on the RS-232/485 bus	0x01 = 2400 b/s 0x02 = 4800 b/s

	0x03 = 9600 b/s 0x04 = 19200 b/s 0x05 = 38400 b/s 0x06 = 57600 b/s 0x07 = 115200 b/s

Response frame:

	C_SetInterfaceConfig +1		OperationCode	CRC

7.12.2 READING OF THE SERIAL INTERFACE CONFIGURATION

Command frame:

	C_ GetInterfaceConfig		CRC

Where:

Parameter name	Parameter description	Value range
C_GetInterfaceConfig	Reading the configuration of the serial interface	0x56

Response frame:

	C_ GetInterfaceConfig +1	Mode, Adr, Baudrate	OperationCode	CRC

Where:
The meaning of the response parameters is the same as described earlier.

7.12.3 EVENT MANAGEMENT

The CTU-R5RM readers have an event memory with a capacity of 3400 records. The source of the event may be a card operation or a state change at the reader inputs. The readers do not have a battery-backed RTC clock. After a power failure, the clock is set to the default value: January 1, 2000, 00:00:00. The event counter is reset to zero.

7.12.4 EVENT RECORDER CONFIGURATION

Command frame:

Header	C_SetEventTrig	CardTrig, In4Trig, In3Trig, In2Trig, In1Trig	CRC

Where:

Parameter name	Parameter description	Value range
C_SetEventTrig	Event masking configuration	0x7C
CardTrig	Mask events related to the card (look down)	0x00 - 0xFF
In1Trig-In4Trig	Masking of input related events (look down)	0x00 - 0xFF

Response frame:

Header	C_SetEventTrig+1		OperationCode	CRC

• "Card" event masking byte

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
Reserve	No memory	Card removal	Card add	Reserve	Master card	Card outside the base	Card from the base

For example, byte 0x25 (00100101) means that events will be written when:

- the card present in the base will be read,
- the card saved as master will be read,
- the card was removed from the base,

• Bajty maskowania zdarzeń pochodzących od zmiany stanu na wejściach

Bajt	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
In1Trig	IO[3]F	IO[3]R	IO[2]F	IO[2]R	IO[1]F	IO[1]R	IO[0]F	IO[0]R
In2Trig	IO[7]F	IO[7]R	IO[6]F	IO[6]R	IO[5]F	IO[5]R	IO[4]F	IO[4]R
In3Trig	IO[11]F	IO[11]R	IO[10]F	IO[10]R	IO[9]F	IO[9]R	IO[8]F	IO[8]R
In4Trig	IO[15]F	IO[15]R	IO[14]F	IO[14]R	IO[13]F	IO[13]R	IO[12]F	IO[12]R

IO[n]R bits denote the response to the rising edge of input **n**,
IO[n]F bits denote the response to the trailing edge of input **n**.

For example, a sequence of configuration bytes In4Trig-In1Trig: 0x00,0x31,0x40,0x08, will cause the events to be written in the case of:

- Appearance of each state change at input with index 10
- Appearance of a rising edge at the entry with index 8
- Appearance of a rising edge at the entry with index 7
- Appearance of a falling edge at the input with index 1

When configuring the triggers of events, it must be determined which of the ports are configured as inputs. You should not configure events for those IOs that are outputs.

To ensure the correctness of the event recording, the time between successive triggers must be> 20ms.

7.12.5 READING THE EVENT RECORDER CONFIGURATION

Command frame:

Header	C_GetEventTrig		CRC

Where:

Parameter name	Parameter description	Value range
C_GetEventTrig	Readout of the event recorder configuration	0x7E

Response frame:

Header	C_GetEventTrig+1	CardTrig, In4Trig, In3Trig, In2Trig, In1Trig	OperationCode	CRC

Response bytes (CardTrig, In4Trig, In3Trig, In2Trig, In1Trig) correspond to the bytes from 10.1.

7.12.6 READING OF COUNTERS RELATED TO THE EVENT MEMORY

Command frame:

Header	C_GetEventParam		CRC

Where:

Parameter name	Parameter description	Value range
C_GetEventParam	Readout of the event recorder configuration	0x78

Response frame:

Header	C_GetEventParam+1	CapL, CapH, PointerL, PointerH, TotB3,TotB2,TotB1,TotB0	OperationCode	CRC

CapH:CapL – a two-byte value specifying the event memory capacity
PointerH:PointerL – a two-byte value that is a pointer to the first free event
TotB3:TotB2:TotB1:TotB0 – a four-byte value specifying the number of events recorded since the counter was reset. TotB3 is the youngest byte.
Events are written sequentially from index 0 to index Cap-1. When the memory is full, the counter "turns" and the oldest entries are overwritten.

Example:
If with the C_GetEventParam command we read that the event memory capacity is 4400 entries, the total number of recorded events is 5678, e.g. if we want to read the event No. 5660, the index of the event we are interested in will be 5660-4400-1 = 1259.
If we want to read the last event, we can use the Pointer value. The index of the last event will be Pointer-1.

7.12.7 READING EVENTS

Command frame:

Header	C_GetEvent	EvNoL, EvNoH	CRC

Where:

Parameter name	Parameter description	Value range
C_GetEvent	Reading the event	0x7a
EvNoL,EvNoH	Low and high byte of the event index	

Response frame:

Header	C_GetEvent+1	RR,MM,DD,gg,mm,ss,typ,B1,B2,B3,B4,B5	OperationCode	CRC

RR,MM,DD - event year, month, day
gg,mm,ss - event hour, minute, second
type - event type:

Depending on the value of the 8th bit of the "type" byte, 2 assignments are distinguished:

Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1
0 - card	No memory	Removed	Added	Reserved	Master	From outside the base	From the base
1 - inputs	Reserved	Reserved	Reserved	N4	N2	N1	N0

N4:N0 - binary input number from which the event was triggered.

- If the event was from a card, bytes B1-B5 contain the ID number of the card.

B1	B2	B3	B4	B5
UID1	UID2	UID3	UID4	UID5 (Unique)

- If the event is from an input change, bytes B1-B5 contain information about the state of the inputs in the format:

B1				B2				B3				B4				B5
IO0	IO1	IO2	IO3	IO4	IO5	IO6	IO7	IO8	IO9	IO10	IO11	IO12	IO13	IO14	IO15	Res

7.13 OTHER COMMANDS

7.13.1 CHANGE BUZZER VOLUME

This command changes the volume of the audible signal. The entered value will be saved in the non-volatile EEPROM memory.

Command frame:

Header	C_BuzzerConfig		CRC

Where:

Parameter name	Parameter description	Value range
C_BuzzerConfig	Changing the buzzer volume	0x00-0x0F

Response frame:

Header	C_BuzzerConfig +1		OperationCode	CRC

7.13.2 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

7.13.3 SLEEP MODE

The CTU-R5RL version can work in sleep mode, consuming low current. This functionality is disabled by default and must be enabled once with the C_SetSleepFeature command. When turned on, the IO2 pin becomes the control input for the sleep mode. Low state on IO2 puts the module in low current consumption. In order to obtain minimum power consumption, the IO1 port and communication inputs (RX / CS / SCK) should be polarized to ground or + 5V.

Command frame:

Header	C_SetSleepFeature	enable{0,1}	CRC

Where:

Parameter name	Parameter description	Value range
C_SetSleepFeature	Enable the SLEEP functionality	0x5C
enable	0 - disable 1 - enable	0,1

Response frame:

Header	C_SetSleepFeature +1		OperationCode	CRC

7.13.4 READING THE READER SOFTWARE VERSION

Command frame:

Header	C_FirmwareVersion		CRC

Where:

Parameter name	Parameter description	Value range
C_FirmwareVersion	Reading the reader software version	0xfe

Response frame:

Header	C_FirmwareVersion+1	Data1…..n	OperationCode	CRC

Where
Data1…n is a sequence of characters written in the form of ASCII codes.

7.14 MEANING OF OPERATION CODES IN ANSWER FRAMES

Operation code name	Description	Value
OC_Error	Error.	0x00
OC_ParityError	Parity error.	0x01
OC_RangeError	Parameter range error.	0x02
OC_LengthError	Data amount error.	0x03
OC_ParameterError	Parameter error.	0x04
OC_Busy	Instantaneous occupancy of internal modules.	0x05
OC_NoACKFromSlave	No internal communication.	0x22
OC_CommandUnknown	Unknown command.	0x07
OC_WrongPassword	The wrong password or the last password has expired, i.e. an automatic LogOut took place.	0x09
OC_NoCard	No transponder.	0x0a
OC_BadFormat	Zły format danych.	0x18
OC_FrameError	Transmission error. It may be a sign of existing disturbances.	0x19
OC_NoAnswer	No response from the transponder	0x1E
OC_TimeOut	Operation time exceeded. It may prove that there is no transponder in the reader field.	0x16
OC_Successful	Operation completed successfully.	0xff
Operation codes related to DESFIRE transponders		
OC_DesNoChanges	Operation Commit made no difference.	0x0c
OC_DesOutOfEEprom	Out of EEPROM memory.	0x0e
OC_DesIllegalCommand	Illegal command.	0x1c
OC_DesIntegrityError	CRC / transmission error with the card.	0x1e
OC_DesNoSuchKey	Invalid key numer.	0x40
OC_DesLengthError	Invalid command length.	0x7e
OC_DesPermisionDenied	You are not authorized to perform the operation.	0x9d
OC_DesParameterError	Command parameter error.	0x9e
OC_DesApplNotFound	No application about selected Aid	0xa0
OC_DesApplIntegrError	Application error, the application is blocked	0xa1
OC_DesAuthError	Authorization error / invalid key	0xae
OC_DesBoundaryError	Record write / read exceeded size	0xbe
OC_DesPICCIntegError	Internal transponder error is blocked	0xc1
OC_DesCountError	Exceeded limit of 28 applications	0xce
OC_DesDuplicateError	An application / file with this identifier already exists	0xde
OC_DesEepromError	Error writing / reading to EEPROM	0xee
OC_DesFileNotFound	The file with this ID does not exist	0xf0
OC_DesFileIntegrError	Unrecoverable file error, the file has been locked	0xf1

8 MODBUS RTU PROTOCOL

WARNING:
When using the MODBUS protocol, the device configuration should be changed so that it does not send the read ID by itself. For this purpose, write the value 0 to the register with the address 1022 (ASerial).

8.1 SUPPORTED FUNCTIONS OF MODBUS PROTOCOL

Funkcja	Opis
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 ADRESSES

8.2.1 ADDRESSES TO READ CARD ID

Adress	Type	R/W	Description
995	Holding Reg	R/W	Time from the last reading, after which the ID saved in the registers 1000 ... 1007 will be reset to zero (x100ms)
996	Holding Reg	R/W	The register is reset to 1 when a new transponder is read
997	Holding Reg	R	B<15:8> - transponder type B<7:0> - number of collisions
998	Holding Reg	R	ID lenght
999	Holding Reg	R	Time counter since last reading (x100ms)
1000	Holding Reg	R	Transponder code [0]
1001	Holding Reg	R	Transponder code [1]
1002	Holding Reg	R	Transponder code [2]
1003	Holding Reg	R	Transponder code [3]
1004	Holding Reg	R	Transponder code [4]
1005	Holding Reg	R	Transponder code [5]
1006	Holding Reg	R	Transponder code [6]
1007	Holding Reg	R	Transponder code [7]

8.2.2 AUTOREADER CONFIGURATION READ / WRITE ADDRESSES

Adress	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

The meaning of the registers is the same as for the parameters of the C_SetAutoreaderConfig command.

8.2.3 ADDRESSES FOR GPIO1 CONFIGURATION

Adress	Type	R/W	Description

1100	Holding Reg	R/W	Dir - determines the direction of the port, Dir=0 - output Dir=1 - input	
1101	Holding Reg	R/W	For **Dir=0** (output) **NormalOpen**	For **Dir=1** (input) **Trigger** - a value that defines the method of triggering the input 0x0000 - not negated 0x0001 - negated 0x0002 - reaction to rising edge 0x0003 - reaction to the falling edge
1102	Holding Reg	R/W	For **Dir=0** (output) **ChangeState**	For **Dir=1** (input) **TypeOfInput** - the type of the input 0x0000 - "0" permanently 0x0001 - "1" permanently 0x0002 - standard input 0x0003 - counting input
1103	Holding Reg	R/W	For **Dir=0** (output) **OtpMode**	Dla **Dir=1** (input) Delay
1104	Holding Reg	R/W	For **Dir=0** (output) TypeOfOutput - the source of the control signal 0x0000 - permanently disabled 0x0001 - permanently on 0x0002 - controlled through the serial interface, automatically returning to zero 0x0003 - controlled via the serial interface 0x0004 - set when you put the card in the internal base against the reader. 0x05 - set when any card is applied to the reader	
1105	Holding Reg	R/W	Dla **Dir=0** (wyjście) **Time** - the length of the generated waveform (x100ms). Value range: 0-255	
1106	Holding Reg	R/W	For **Dir=0** (output) **Time0** - logical zero time (x100ms) Value range: 0-255.	
1107	Holding Reg	R/W	For **Dir=0** (output) **Time1** - time of logical one (x100ms) Value range: 0-255.	

8.2.4 ADDRESSES FOR GPIO2 CONFIGURATION

Adress	Type	R/W	Opis	
1110	Holding Reg	R/W	Dir - determines the direction of the port, Dir=0 - output Dir=1 - input	
1111	Holding Reg	R/W		For Dir = 1 (input) Triger - a value that defines the method of triggering the input 0x0000 - not negated 0x0001 - negated 0x0002 - reaction to rising edge 0x0003 - reaction to the falling edge
1112	Holding Reg	R/W		For Dir = 1 (input) TypeOfInput - the type of the input 0x0000 - "0" permanently 0x0001 - "1" permanently 0x0002 - standard input 0x0003 - counting input

1113	Holding Reg	R/W	For Dir = 1 (input) Delay
1114	Holding Reg	R/W	For **Dir=0** (output) TypeOfOutput - the source of the control signal 0x0000 - permanently disabled 0x0001 - permanently on 0x0002 - controlled through the serial interface, automatically returning to zero 0x0003 - controlled via the serial interface 0x0004 - set when you put the card in the internal base against the reader. 0x0005 - set when any card is applied to the reader
1115	Holding Reg	R/W	For **Dir=0** (output) **Time** - the length of the generated waveform (x100ms). Value range: 0-255
1116	Holding Reg	R/W	For **Dir=0** (output) **Time0** - logical zero time (x100ms) Value range: 0-255.
1117	Holding Reg	R/W	For **Dir=0** (output) **Time1** - time of logical one (x100ms) Value range: 0-255.

8.2.5 ADDRESSES FOR RELAY CONFIGURATION

Adres	Type	R/W	Opis
1120	Holding Reg	R/W	Dir - determines the direction of the port, Dir=0 - output
1121	Holding Reg	R/W	
1122	Holding Reg	R/W	
1123	Holding Reg	R/W	
1124	Holding Reg	R/W	TypeOfOutput - the source of the control signal 0x0000 - permanently disabled 0x0001 - permanently on 0x0002 - controlled through the serial interface, automatically returning to zero 0x0003 - controlled via the serial interface 0x0004 - set when you put the card in the internal base against the reader. 0x0005 - set when any card is applied to the reader
1125	Holding Reg	R/W	**Time** - the length of the generated waveform (x100ms). Value range: 0-255
1126	Holding Reg	R/W	**Time0** - logical zero time (x100ms) Value range: 0-255.
1127	Holding Reg	R/W	**Time1** - time of logical one (x100ms) Value range: 0-255.

8.2.6 ADDRESSES FOR BUZZER CONFIGURATION

Adress	Type	R/W	Desctription
1130	Holding Reg	R/W	Dir - determines the direction of the port, Dir=0 - output
1131	Holding Reg	R/W	
1132	Holding Reg	R/W	
1133	Holding Reg	R/W	
1134	Holding Reg	R/W	TypeOfOutput - the source of the control signal 0x0000 - permanently disabled 0x0001 - permanently on 0x0002 - controlled through the serial interface, automatically returning to zero 0x0003 - controlled via serial interface 0x0004 - set when you put the card in the internal base against the reader.

			0x0005 - set in case of applying to the reader any card
1135	Holding Reg	R/W	**Time** - the length of the generated waveform (x100ms). Value range: 0-255
1136	Holding Reg	R/W	**Time0** - logical zero time (x100ms) Value range: 0-255.
1137	Holding Reg	R/W	**Time1** - time of logical one (x100ms) Value range: 0-255.

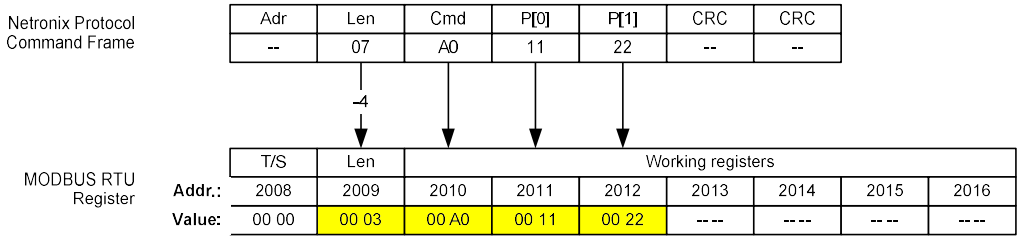
8.3 ENCAPSULATION OF THE NETRONIX PROTOCOL IN THE MODBUS RTU PROTOCOL

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

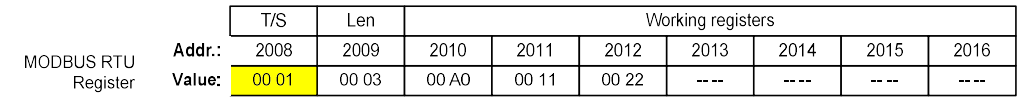
Adress	Type	R/W	Name	Description
2008	Holding Reg	R/W	Trigger/Status	This register is used to trigger command processing and to check the processing status. Allowed values: 0x0000 - module in Idle mode 0x0001 - processing triggered 0x00EE - error 0x00FF - command done. The answer is in the working registers.
2009	Holding Reg	R/W	Len	This register contains the length of the written command / response length (number of registers written / to be read)
2010-2073	Holding Reg	R/W	Working registers	These registers are used to write the command / read the response. One register holds the value of 1 command / response byte from the Netronix frame

8.3.1 SCHEME OF THE PROCEDURE

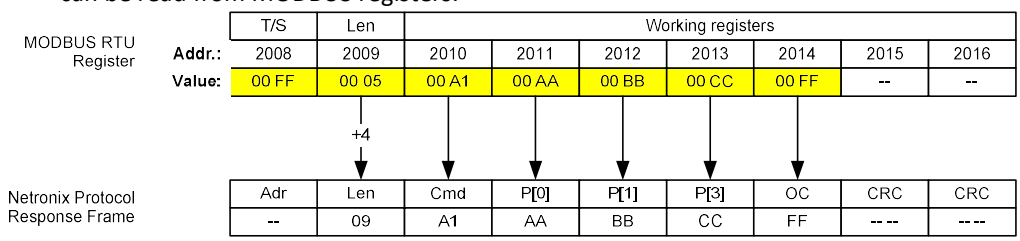
1. Enter the command from the Netronix protocol to the MODBUS registers according to the scheme below:



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



3. Read the Trigger / Status register until 0x00FF is displayed. Value 0x00FF means that the answer is ready and can be read from MODBUS registers.



8.3.2 EXAMPLE OF USE - READING THE SOFTWARE VERSION

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

1. Define what the frame should look like in the Netronix protocol. The Cmd field and any parameters are important. The whole frame for the command to read the firmware version looks as follows:

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

Cmd = 0xFE
Param - none

2. Enter the amount of data into the ***Len register***, and enter the command code (and any parameters) into 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. Enter the value 0x0001 into the ***Trigger / Status*** register. This will execute the command stored in ***Working Registers***.
RTU Tx > 01 06 07D7 0001 F946
RTU Rx > 01 06 07D7 0001 F946

Trigger/Status=0001

4. Next, the ***Trigger / Status*** register should be read, until the moment of reading the value 0x00FF. Value 0x00FF means that the command has been executed and there is a response in ***Working Registers***.
RTU Tx > 01 03 07D7 0001 3546
RTU Rx > 01 03 02 00FF F804

Trigger/Status=00FF

5. Next, the value of the Len register should be read. This register contains information on the number of registers in which the answer is saved
RTU Tx > 01 03 07D8 0001 0545
RTU Rx > 01 03 02 0011 7848

Len=0011

6. In the last step, read the Len of the 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 (answer to C_FirmwareVersion)
004D 0057 002D … 0031 002E 0035 – firmware version returned – „MW-R7-v3.2.A.1.5”
00FF – Operation code (success)

9 MASTERID MECHANISM

This mechanism is based on the possibility to quickly add / remove user cards to / from the reader memory by means of a "master card".

If you want to register the card as a "master card", you must first clear the card memory by returning to the factory settings. After clearing the memory, the selected card should be applied to the module at any time. This card automatically becomes the "master card". The master card cannot be removed or added with another card.

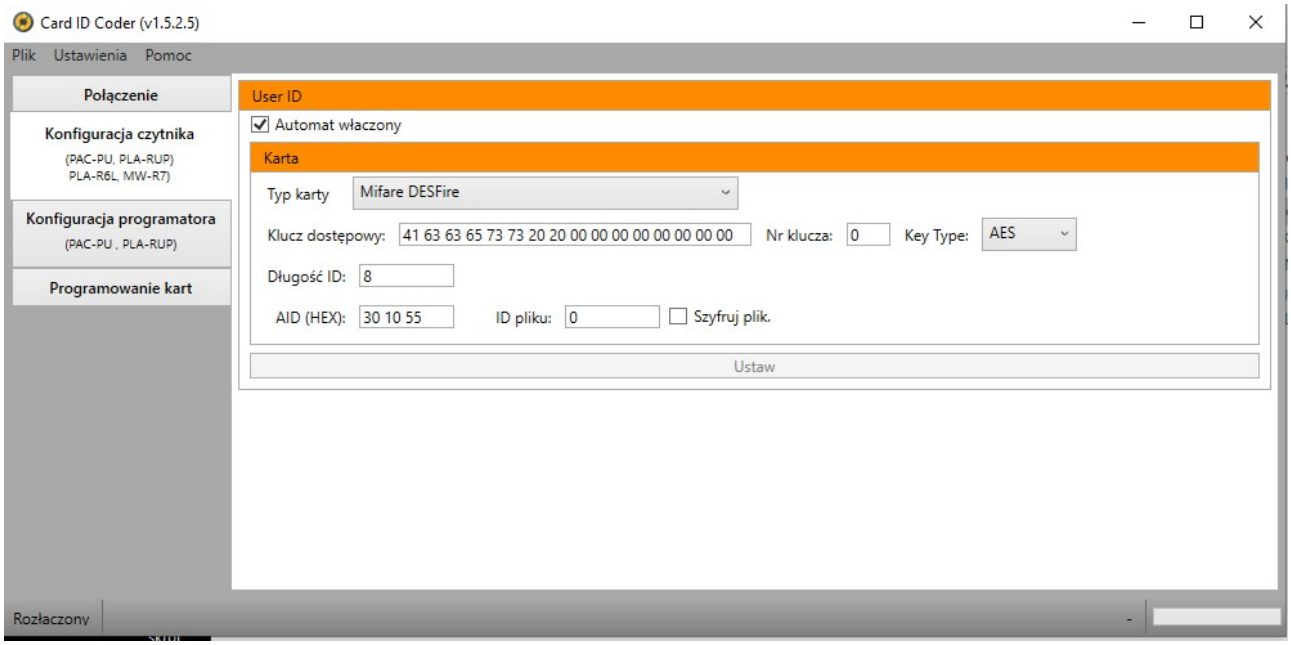
If we want to register the card as a "user card", first apply the "master card" to the reader and then, within approx. 5 seconds, apply the registered card to the reader.

If we want to remove the "user card" from the memory, first apply the "master card" to the reader and then, within approx. 5 seconds, apply the removed card to the reader.

After applying the "user card" to the reader, the reader activates the electrical output programmed as controlled by the internal access control mechanism.

10 USERID MECHANISM

This mechanism allows to read secured memory part of transponders and send its like UID. To setup this feature, use CardIDCoder tool.



11 CLEARING THE CARD MEMORY AND RETURN TO FACTORY SETTINGS

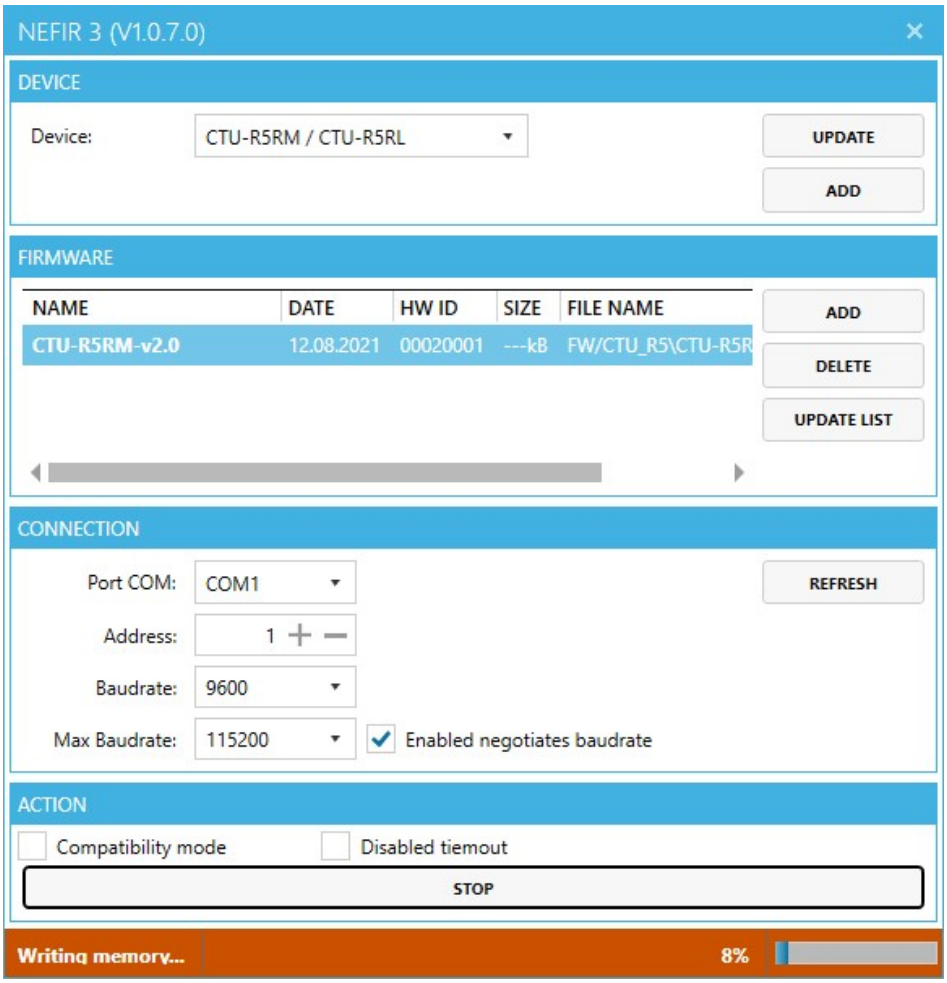
To return to the factory settings, press the return to factory settings button for about 8 seconds. When returning to factory settings, the following reader parameters are permanently set:

Parameter name or functionality	Value or setting	
Interface		
RS232 interface	Address: 0x01	
	Speed: 0x03	9600bps
RS485 interface	Address: 0x01	
	Speed: 0x03	9600bps
1-Wire interface	Family : 0x01	
	Address: 0x00	
WIEGAND interface	Number of bits: 37	
I2C interface	Address: 0xC0	
Autoreader config		
AAutoreader	Trigger : 0x02	
	Timeout : 0x14	2s
	ASerial : 0x01	At the first touchdown
	AMode : 0x04	Information extended by the number of cards in the field and the card type
	ABuzzer : 0x01	At the first touchdown
	AMulti : 0x11	MIFARE + EM4102
Master card	No master card In memory	
All internal transponder memory together with the Master card	0xff 0xff 0xff 0xff 0xff - memory cleared	
Wejścia/Wyjścia		
Port 0 – GPIO1	Entry of any destination	
Port 1 – GPIO2	Entry of any destination	
Port 2 – relay	Activation of the electric lock	
Port 3 – buzzer	Signaling of switching on the electric lock	
Buzzer volume	0x05	
Event		
Event configuration	Event log inactive	
Password		
Password	None	
SLEEP functionality	Disabled	

12 BOOTLOADER – CHANGING THE FIRMWARE OF THE DEVICE

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

1. Connect the device to the RS232 serial port on the computer,
2. Open the NEFIR3.exe program,
3. In the DEVICE section, select the appropriate device. If the device is not on the list, press the UPDATE buton,
4. In the FIRMWARE section, select the firmware version you want to program the device. Pressing the UPDATE LIST button causes the latest versions of files to be downloaded from the server,
5. In the CONNECTION section:
    a. Select the correct COM port,
    b. Set Baudrate on the baud rate at which the reader works (standard 9600),
    c. Optionally, check Enabled negotiates baudrate and select 115200 speed in the Max Baudrate field. Such settings speed up the reloading process.
6. In the ACTION section, press the START button. After pressing the button, the process of uploading the new firmware to the device will start.



Rysunek: View of the program window while reloading the firmware

13 EXAMPLE USAGE WITH TRANSPONDERS

13.1 EXAMPLE USAGE WITH MIFARE CLASSIC S50, S70

After correct connection of the reader and establishing mutual communication between reader and the master computer, you can start the operation of reading and writing the transponder memory.
The following operations assume that the reader has factory settings and that the S50 card used has factory settings, i.e. full access rights and both keys 0xff ff ff ff ff ff.

Since during manual attempts the time between successive commands sent via RS is relatively long and reaches from a few seconds to several minutes, the internal UID readout machine should be turned off.

This should be done with the command:
SetAutoReaderConfig z parametrami 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 .

To read the transponder, first load the key into the key memory.
So let's load the SKB key with
C_LoadKeyToSKB, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00

We turn on the field
TurnOnAntennaPower, 0x01

We put the transponder to the reader

We select the transponder
C_Select, 0x00

We log into, for example, sector 3.
C_LoginWithSKB, 0x03, 0xAA, 0x00

Read the contents of the 2nd block in the 3rd sector.
C_ReadBlock, 0x02

While all Operation Codes in response frames were OC_Successful, the received values are data read from the block.

13.2 EXAMPLE USAGE WITH MIFARE DESFIRE TRANSPONDERS

After correct connection of the reader and establishing mutual communication between it and the master computer, you can start the operation of reading and writing the transponder memory.
The following operations assume that the reader has factory settings and that the DESFire card used has factory settings, i.e. full access rights, and the PICC Master key has the value 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00.
The result of this example is creating a new application, changing the standard application key, creating a data file, saving and then reading the data from the file.

Since during manual attempts the time between successive commands sent via RS is relatively long and reaches from a few seconds to several minutes, the internal UID readout machine should be turned off.

This must be done with an command:

1. SetAutoReaderConfig 0x00, 0x00, 0x00, 0x00, 0x00, 0x00.

To read the transponder, first load the keys into the key memory.
So we load the standard key of DESFire transponders to the position of, for example, "3" in the reader's memory, and to position 4 we load our own key, which we will give to the new application:

2. C_DesSaveKey 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

3. C_DesSaveKey 0x04, 0x01, 0x02, 0x03, 0x04, 0x0a, 0x0b, 0x0c, 0x0d

We turn on the field

4. C_TurnOnAntennaPower 0x01

We put the transponder to the reader, we select the transponder

5. C_Select 0x00

We initiate the ISO data exchange protocol, with the logical number of the transponder 0

6. C_DesInitProtocol 0x00

We authorize using the "0" key, i.e. PICC Master key, this key is stored in the reader memory under the index "3"

7. C_DesAuth 0x00,0x03

We create an application with an identification number, e.g. 0x30, 0x10, 0x55, with default ApplicationMasterKey settings, with space reservation for 4 keys

8. C_DesCreateApp 0x30,0x10,0x55,0x0F,0x04

We change the default, newly created Application MasterKey to the one we have saved in the reader at position 4. Therefore, we select a new application:

9. C_DesSelectApp 0x30,0x10,0x55

We log in to the application using the Application Master Key, then change it and then log in again with the new key

10. C_DesAuth 0x00,0x03
11. C_DesChangeKey 0x00,0x04
12. C_DesAuth 0x00,0x04

Tworzymy standardowy plik z danymi, z pełnymi prawami dostępu dla Application Master Key, oraz prawami odczytu dla klucza „3". Plik będzie miał indeks „2", nieszyfrowaną wymianę danych oraz wielkość 1500 bajtów

13. C_DesCreateSTDataFile 0x02,0x00,0x30,0x00,0xDC,0x05,0x00

We now write data to the just created file from position 0

14. C_DesWriteData 0x02,0x00,0x00,0x00, $TuSaNaszeDaneDoZapisu

We read 21 bytes of the data just written

15. C_DesReadData 0x02,0x00,0x00,0x00, 0x15,0x00,0x00