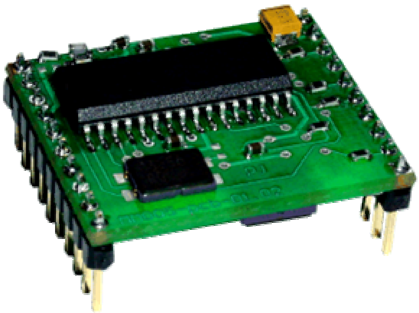
Technical Data Sheet

MM-005

MM005-doc-00.07

In reference to the MM005-c-00.05+



Contents

Contents 2
Introduction 3
General specifications 3
The pin diagram 4
Connection diagram 5
The module dimensions 5
General command frame format for the reader 6
General response frame format for the reader 6
Command description 7
High level commands 7
  Write of 16 bytes to the block 7
  Reading out the 16 bytes from the block 7
  Incrementation the value written in the block 8
  Decrementation the value written in the block 8
Low level commands 9
  Switching the antenna electromagnetic field on 9
  Switching the antenna electromagnetic field off 9
  General call out for cards 9
  Loading the key to the cache key buffer 10
  Loading the key to the non-volatile key memory 10
  Logging to the sector with the cache key buffer 10
  Logging to the sector with non-volatile key memory 11
  Writing the16 bytes to the block 11
  Reading-out the 16 bytes from the block 11
  Copying the 16 bytes from block to block 11
  Writing the value to the block 12
  Reading-out the value from the block 12
  Incrementation the value written in the block 12
  Decrementation the value written in the block 13
  Data transfer from transponder registry to the chosen block 13
  Setting the transponder into the stand-by state 13
Commands managing the user I/O ports 14
  Setting up the port as a output with simultaneous state setting 14
  Setting up the port as a input with simultaneous state setting 14
Additional commands 14
  Setting up the gain of the circuit receiving data from the transponder 14
  Setting up the baud rate of the UART interface 14
  Assigning the RS bus address to the module 15
  Reading-out the module software version number 15
Calculating the CRC value 15
Examples of operation of the Mifare® transponder by means the MM-005 module 16
Operation the module with the high level functions 17
  Example 1 Writing the 16 bytes to the block 17
  Example 2 Decrementation the value written in the block 18
Operation the module with high level functions 20
  Example 3 Decrementation the value written in a block 20

Introduction

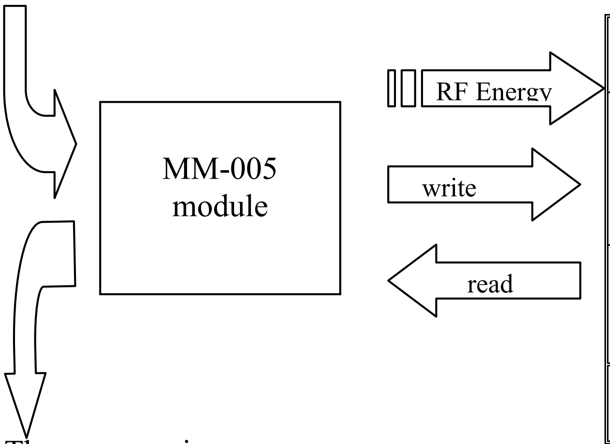
Module MM-005 operates on principle of the contact-less information writing and reading from and to the transponder Mifare® (RFID). Data is transmitted via RS-232 interface compatible with TTL voltage level.

The principle of operation:

Query from (master unit- host) - (module) action - (module) response.

The query is sent to the module MM-005:

module address	frame length	command	data	CRCH,CRCL
xx	xx	xx	xx xx xx	xx xx



Sector no.	Block no.	Transponder block with written data
0	0	B1......................B16
	1	B1......................B16
	2	B1......................B16
	3	B1......................B16
⋮	⋮	⋮
15 or 39		

The response is:

Module address	frame length	response	data	operation code	CRCH,CRCL
xx	xx	xx	xx xx	xx	xx xx

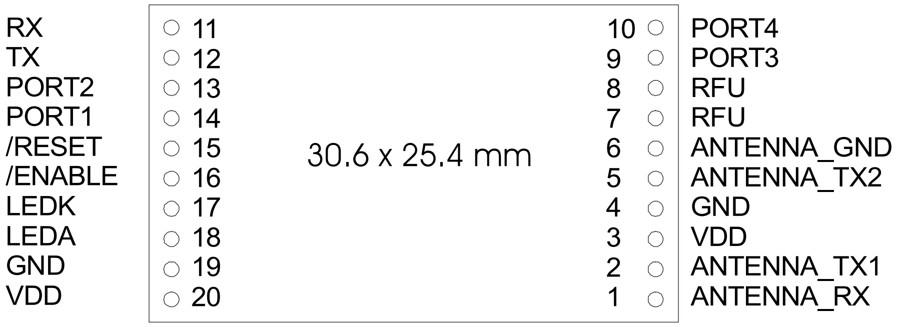
The module is equipped with two 1-bit user ports, which can be used for reading and writing. Connect an air coil antenna to the MM-005. The antenna will produce an electromagnetic field and supply a transponder located in the field.

General specifications

Supply voltage Uz:	4.5...5.5 V
Supply current:	1...55 mA
Module rated operating radio frequency:	13.56 MHz
Baud rate of data received from transponder:	106 kbps (10 µs/b)
Write time to data block:	6 ms + RS transmission
Read time from data block:	2.5 ms + RS transmission
Time of typical ticket transaction:*	70 ms + RS transmission
Output current capacity: port1 to port4 and RS-TX:	10 mA
Transporter read / write distance (depending on antenna used):	5...10 cm
Antenna	External with resonance adjusting capacitors for 13.56 MHz
Transmission:	1200 to 115200 bps, 8 data bits, 1 stop bit, no parity bit, with voltages comply TTL levels (default: 9600 bps)

*refresh of two values and write in of two blocks:

The pin diagram



Module pins – element side view

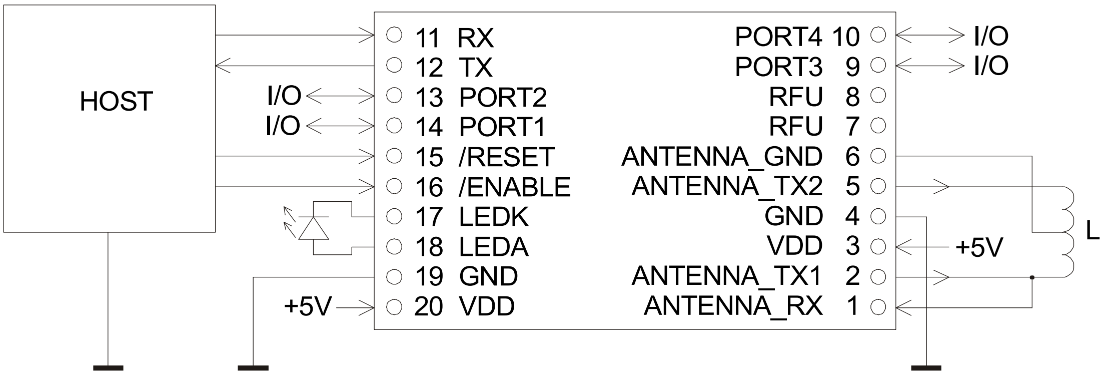
1. ANTENNA_RX – input receiving the data from transponder – connect to antenna
2. ANTENNA_TX1 – one of outputs that supply the antenna with energy
3. VDD – plus of supply voltage
4. GND – module earth (minus of supply voltage)
5. ANTENNA_TX2 – one of outputs that supply the antenna with energy
6. ANTENNA_GND – antenna earth – the tap of connected antenna
7. NC (not connected)??
8. NC (not connected)??
9. PORT3 – user output/input *
10. PORT4 - user output/input *
11. RS232-RX – RS-232 input with voltages comply TTL level *
12. RS232-TX – RS-232 output with voltages comply TTL level *
13. PORT2 – user output/input *
14. PORT1 - user output/input *
15. /RESET – output of external module reset signal, active state L *
16. /ENABLE – input of module enable signal, active state L *

17,18. LEDA, LEDK – outputs for connecting the external LED - anode, cathode respectively

19. GND – module earth (minus of supply voltage)
20. VDD – plus of supply voltage

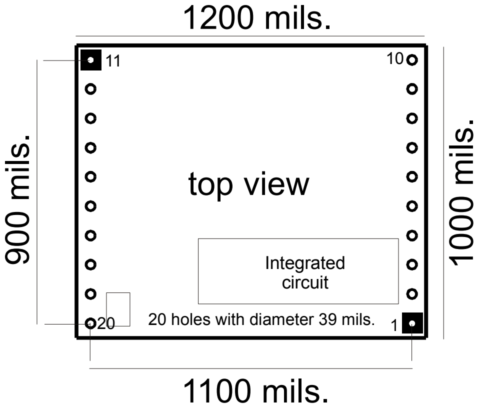
* With 100 Ohm safety resistor

Connection diagram



Connection diagram with external elements

The module dimensions



General command frame format for the reader

Module address	Frame length	Command	Parameters 1 to n	CRCH	CRCL
1 byte	1 byte	1 byte	n * bytes	1 byte	1 byte

Where:

Module address - unique module address in the system

If:

Module address = 0 any module will not respond

Module address = 0xFF all modules in net will answer

Frame length – total number of frame bytes

Command – even value

Parameters 1 to n – occur optionally and depend on command

CRCH, **CRCL** - MSByte and LSByte of CRC16 respectively

General response frame format for the reader

Module address	Frame length	Response	Parameters 1 to n	Operation code	CRCH	CRCL
1 byte	1 byte	1 byte	n * bytes	1 byte	1 byte	1 byte

Where:

Module address - assigned the real address of responding module

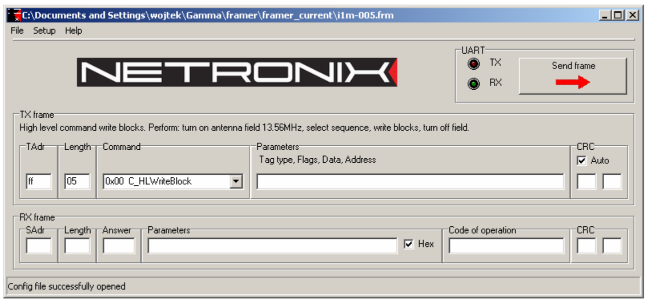
Frame length - total number of response frame bytes

Response = Command + 1 (odd value)

Parameters 1 to n - exist optionally and depend on command

Operation code - informs about correctness of executed command

CRCH, CRCL - MSByte and LSByte of CRC16 respectively



C:\Documents and Settings\wojtek\Gamma\framer\framer_current\i1m-005.frm

TX frame

High level command write blocks. Perform: turn on antenna field 13.56MHz, select sequence, write blocks, turn off field.

TAdr: ff

Length: 05

Command: 0x00 C_HLWriteBlock

Parameters: Tag type, Flags, Data, Address

CRC: ☑ Auto

RX frame

Parameters: ☑ Hex

Config file successfully opened

Module can be tested with free of charge FRAMER software tool, which makes work with frames easier.

Command description

Assignments:
BlockNo – value=(0…3) for MF1ICS50
SectorNo – value=(0…0x0F) it means 16 sectors MF1ICS50

Key1...6 – key we are to use for logging to the sector
KeyType – key type we are to use for logging to the sector
0xAA – for type A key
0xBB – for type B key

High level commands

The transponder can fully communicate with MIFARE® card by means of high level commands. It means, that field switching on, card selection, authorization, proper process and switching off the field is carried on automatically.

Write of 16 bytes to the block

Name of command – query	Command code	Parameters
C_HLWriteBlock	0x00	Data1...16, SectorNo, BlockNo, Key1...6, KeyType

Data1…16 – data for write
SectorNo – the target sector
BlockNo – the target block within the sector.

Name of command – query	Response code	Parameters
A_HLWriteBlock	0x01	OperationKod

OperationKod - 0xff-the write is correct

Reading out the 16 bytes from the block

Name of command – query	Command code	Parameters
C_HLReadBlock	0x02	SectorNo, BlockNo, Key1...6, KeyType

SectorNo – source sector
BlockNo – source block within the sector

Name of command – query	Response code	Parameters
A_HLReadBlock	0x03	Data1...16, OperationKod

Data1...16 – read-out from the block
OperationKod - 0xff- read-out is correct

Incrementation the value written in the block

This command generates read-out sequence of values from the block to the internal operational registry „data register”, increments this value and writes the result into to the source block once more.

Name of command – query	Command code	Parameters
C_HLIncrement	0x04	SectorNo, BlockNo, Value1...4, Key1...6, KeyType

SectorNo – the sector on which we are to carry out the operation
BlockNo – the incremented block
Value1...4 – value we want to add to the value in block “ BlockNo”

Name of command – query	Response code	Parameters
A_HLIncrement	0x05	OperationKod

OperationKod - 0xff- the incementation is correct
To carry out the operation successfully, the block should be formatted as ”Value”.

Decrementation the value written in the block

This command generates read-out sequence of values from the block to the internal operational registry „data register”, decrements this value and writes the result into to the source block once more.

Name of command – query	Command code	Parameters
C_HLDecrement	0x06	SectorNo, BlockNo, Value1...4, Key1...6, KeyType

SectorNo – the sector on which we are to carry out the operation
BlockNo – the decremented block
Value1...4 – the value we want to subtract from value I block “ BlockNo”

Name of command – response	Response code	Parameters
A_HLDecrement	0x07	

OperationKod - 0xff- the decrementation is correct
To carry out the operation successfully, the block should be formatted as ”Value”.

Low level commands

Low level commands can be used in freely arranged sequences without multiple switching on/off of field, multiple transponder selection and multiple login to its sector. An application using these commands can control fully numerous transponder located in the field.

Switching the antenna electromagnetic field on

Name of command – query	Command code	Parameters
C_TurnOnAntennaPower	0x10	-

Name of command – response	Response code	Parameters
A_TurnOnAntennaPower	0x11	OperationKod

OperationKod –0xff always

Switching the antenna electromagnetic field off

Name of command – query	Command code	Parameters
C_TurnOffAntennaPower	0x44	-

Name of command – response	Response code	Parameters
A_TurnOffAntennaPower	0x45	OperationKod

OperationKod – 0xff always

General call out for cards

This command initiates the anti-collision loop for transponders situated in the antenna field, chooses the one of them and returns its ID.

Name of command – query	Command code	Parameters
C_Select	0x12	TypeOfRequest

TypeOfRequest – decides on call out type of the transponders.

If: TypeOfRequest=0xff – the transponders which are in state „Idle" and „Halt" are called out. The call out of such type is named „Request All"

TypeOfRequest=0x01 – only transponders which are in state „Idle" are called out. The call out of such type is named „Request Standard"

Name of command – response	Response code	Parameters
A_Select	0x13	ID1…4, OperationKod

ID1…4 – Card ID which has been selected

OperationKod - 0xff- proper selection

Loading the key to the cache key buffer

This buffer is included in the MM-005 module as a RAM memory. It is possible to memorize one key in this buffer. It is impossible to read-out this key, but by means of it to log to the sectors only.

Name of command – query	Command code	Parameters
C_LoadKeyBuffer	0x14	Key1…6

Key1…6 – key, which is to be loaded to the buffer.

Name of command – response	Response code	Parameters
A_LoadKeyBuffer	0x15	OperationKod

OperationKod - 0xff- the operation is correct

Loading the key to the non-volatile key memory

This buffer is included in the MM-005 module as an EEPROM memory. It is possible to memorize up to 32 keys, as subsequent locations, in this memory. It is impossible to read-out this keys, but by means of them to log to the sectors only.

Name of command – query	Command code	Parameters
C_LoadEEKeyBuffer	0x16	Key1…6, EEKeyNo

Key1…6 – the key, which is to be loaded to the non-volatile key memory

EEKeyNo – the memory position number =(0...0x1F)

Name of command – response	Response code	Parameters
A_LoadEEKeyBuffer	0x17	OperationKod

OperationKod - 0xff- the operation is correct

Logging to the sector with the cache key buffer

Name of command – query	Command code	Parameters
C_LoginWithKB	0x18	SectorNo, KeyType

SectorNo – sector no. to which we want to log

KeyType – decides how a key which is present in the key buffer, will be treated during logging

If KeyType =0xAA, the key will be treated as a type A key, if KeyType =0xBB, the key will be treated as a type B key

Name of command – response	Response code	Parameters
A_LoginWithKB	0x19	OperationKod

OperationKod - 0xff - logging is correct

Logging to the sector with non-volatile key memory

Name of command – query	Command code	Parameters
C_LoginWithEE	0x1a	SectorNo, KeyType, EEKeyNo

SectorNo – sector no., to which we want to log
KeyType – decides how a key which is present in EEKeyNo location, will be treated during logging
If KeyType=0xAA the key will be treated as a type A key, if KeyType=0xBB, the key will be treated as a type B key
EEKeyNo – location no. in the non-volatile key memory (0...1F)

Name of command – response	Response code	Parameters
A_LoginWithEE	0x1b	OperationKod

OperationKod - 0xff - logging is correct

Writing the16 bytes to the block

Name of command – query	Command code	Parameters
C_WriteBlock	0x1c	Data1...16, Blok nr

Data1...16 – data for writing
Blok nr – block no., to which data will be written in

Name of command – response	Response code	Parameters
A_WriteBlock	0x1d	OperationKod

OperationKod - 0xff – write is correct

Reading-out the 16 bytes from the block

Name of command – query	Command code	Parameters
C_ReadBlock	0x1e	Blok nr

Blok nr – block no., from which data will be red-out

Name of command – response	Response code	Parameters
A_ReadBlock	0x1f	OperationKod

OperationKod - 0xff – read-out is correct

Copying the 16 bytes from block to block

Name of command – query	Command code	Parameters
C_CopyBlock	0x20	SourceBlockNo, TargetBlockNo

SourceBlockNo – The source block no.
TargetBlockNo – The target block no.

Name of command – response	Response code	Parameters
A_CopyBlock	0x21	OperationKod

OperationKod - 0xff – read-out is correct
Copying can be done within the same sector we are logged to.

Writing the value to the block

This command converts 4-byte value in 16-byte value, which conforms Value format and write this value to the block denoted as BlockNo.

Name of command – query	Command code	Parameters
C_WriteValue	0x34	Value1...4, BackupBlockNo, BlockNo

Value1...4 – value written to the block "BlockNo"
BackupBlockNo – the back-up block address
BlockNo – the target block no.

Name of command – response	Response code	Parameters
A_WriteValue	0x35	OperationKod

OperationKod - 0xff – read-out is correct

Reading-out the value from the block

Name of command – query	Command code	Parameters
C_ReadValue	0x36	BlockNo

BlockNr – the source block no.

Name of command – response	Response code	Parameters
A_ReadValue	0x37	Value1...4, BackupBlockNo, OperationKod

Value1...4 – red-out value
BackupBlockNo – red-out address of back-up block
OperationKod - 0xff – the operation is correct
To OperationKod=0xff there should be write conforming "Value" format in the source block.

Incrementation the value written in the block

This command adds Value1...4 argument to the value written in the block "BlockNr." The result remains in the transponder internal data buffer „data register". To write the result to the sector, use the command C_TransferValue. This way we can get the argument from one block and write the result to the other one. It increases data protection level.

Name of command – query	Command code	Parameters
C_IncrementValue	0x30	BlockNo, Value1...4

BlockNo – the incremented block number.
Value1...4 – value which is being added to the block "BlockNo"

Name of command – response	Response code	Parameters
A_IncrementValue	0x31	OperationKod

OperationKod - 0xff – operation is correct
We can perform operations within the same sector we are logged in.

Decrementation the value written in the block

This command subtracts Value1...4 argument from the value written in the block “BlockNr.” The result remains in the transponder internal data buffer „data register”. To write the result to the sector, use the command C_TransferValue. This way we can get the argument from one block and write the result to the other one. It increases data protection level.

Name of command – query	Command code	Parameters
C_DecrementValue	0x32	BlockNo, Value1...4

BlockNo – the decremented block number
Value1...4 – value subtracted from the block “BlockNo”

Name of command – response	Response code	Parameters
A_DecrementValue	0x33	OperationKod

OperationKod - 0xff – the operation is correct
We can perform operations within the same sector we are logged in.

Data transfer from transponder registry to the chosen block

Name of command – query	Command code	Parameters
C_TransferValue	0x38	BlockNo

BlockNo – the target block number

Name of command – response	Response code	Parameters
A_TransferValue	0x39	OperationKod

OperationKod - 0xff - the operation is correct
This command is used to transfer the results of calculation from the internal data registry „data register” to the chosen memory block. This operation is performed after execution of C_IncrementValue or C_DecrementValue command.

Setting the transponder into the stand-by state

The command we use when we want to set-up the active transponder into the stand-by state (after correct execution of command: C_Select and/or C_LoginWithEE and/or C_LoginWithKB). When such operation is completed, we can select one of the transponders remained in the field once again and carry required operations. To select one of the remained transponders, use the C_Select command with parameter 0x01. In case of many transponders present in the antenna field, we can use all transponders one after another.

Name of command – query	Command code	Parameters
C_Halt	0x40	-

Name of command – response	Response code	Parameters
A_Halt	0x41	OperationKod

OperationKod - 0xff - the operation is correct
To communicate with the transponder once again, which is in state “Halt”, use the C_Select command with 0xff parameter.

Commands managing the user I/O ports

After the module-reset state had occurred, all I/O ports are defined as inputs with high input impedance.

Setting up the port as a output with simultaneous state setting

Name of command – query	Command code	Parameters
C_WriteUserPort	0x50	PortNo, Value

PortNo – (1...4) port no.
Value – bit for writing

Name of command – response	Response code	Parameters
A_WriteUserPort	0x51	OperationKod

OperationKod – 0xff always

Setting up the port as a input with simultaneous state setting

Name of command – query	Command code	Parameters
C_ReadUserPort	0x52	PortNo

PortNo – (1...4) port no.

Name of command – response	Response code	Parameters
A_ReadUserPort	0x53	Value, OperationKod

Value – written-in port value
OperationKod – 0xff value

Additional commands

Setting up the gain of the circuit receiving data from the transponder

Name of command – query	Command code	Parameters
C_SetGain	0x60	NewGain

NewGain – the new gain value of the circuit receiving data from transponder (0…3)

Name of command – response	Response code	Parameters
A_SetGain	0x61	OperationKod

OperationKod - 0xff- the operation is correct

Setting up the baud rate of the UART interface

Name of command – query	Command code	Parameters
C_SetUartSpeed	0x62	UartSpeed

UartSpeed – baud rate of UART port = (1…8), where 1=1200 b/s …8=115200 b/s

Name of command – response	Response code	Parameters
A_SetUartSpeed	0x63	OperationKod

OperationKod – 0xff always

The MM-005 responds with preceding baud rate next switches to a “new” one and then waits for 10 seconds. During this, to maintain the new baud rate the MM-005 should receive whichever frame with correct CRC code. In other case the module will return to the previous one.

Assigning the RS bus address to the module

Name of command – query	Command code	Parameters
C_SetSlaveAdres	0x64	NewSlaveAdr

NewSlaveAdr – the new address we assign to the unit

Name of command – response	Response code	Parameters
A_SetSlaveAdres	0x65	OperationKod

OperationKod – 0xff always

Reading-out the module software version number

Name of command – query	Command code	Parameters
C_SoftVersion	0xfe	-

Name of command – response	Response code	Parameters
A_SoftVersion	0xff	Data1...n, OperationKod

Data1...n – the software version written in SCII code
OperationKod - 0xff always

Calculating the CRC value

The CRC value is calculated from equation x^16+x^12+x^5+1 with initial value equal to 0x0000. This value is calculated in virtue of all the bytes except of CRCH and CRCL. Example of calculation of CRC value, written in C language:

```
void LiczCRC2(unsigned char *FromAddr, unsigned short *ToAddr, unsigned char Many)
{
int     i,NrBajtu;
 unsigned short C;
        *ToAddr=0;
        for (NrBajtu=1;NrBajtu<=Many;NrBajtu++,FromAddr++)
        {
                C=((*ToAddr>>8)^*FromAddr)<<8;
                for (i=0;i<8;i++)
                        if (C&0x8000) C=(C<<1)^0x1021;
                        else C=C<<1;
                *ToAddr=C^(*ToAddr<<8);
        }

}
```

Where:
*FromAddr - is the data first byte flag
Many - informs how many data bytes will be used for calculation
*ToAddr - is the flag for the calculated CRC value

Examples of operation of the Mifare® transponder by means the MM-005 module

Assuming that:

- Messages are sent as a broadcasting ones (to the all modules in the network, AdresModułu=ff)

Typical command frame:

module address	frame length	command	data	CRCH,CRCL
ff	XX	XX	XX XX XX XX	XX XX

- We assume that, address 01 has been assigned to the reader earlier by means C_SlaveAddressSet function. It means, that responding reader will have the address 01.

Typical response frame:

module address	frame length	response	data	operation code	CRCH,CRCL
01	XX	XX	XX XX XX XX	XX	XX XX

- The transponder used for tests has set the configuration blocks (sector trailer), just like the new transponder of the Philips production, that is to say:
  ff ff ff ff ff ff ff 07 80 69 ff ff ff ff ff ff
  It means that, the transponder will allow to write and read-out of the data blocks and to increment, decrement and transfer of values. With this configuration, it is possible to perform all these operation by means of A or B password.
  (If we have a transponder of other producer, the A or B keys can be like that: a0 a1 a2 a3 a4 a5 and b0 b1 b2 b3 b4 b5)

Operation the module with the high level functions

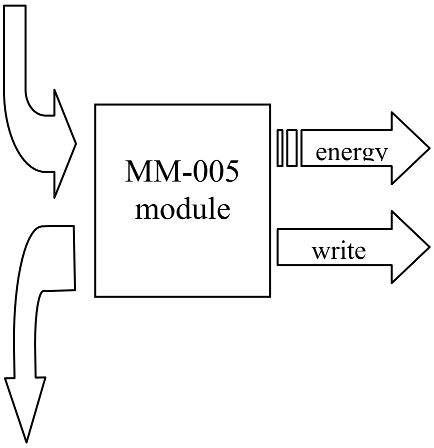
Example 1 Writing the 16 bytes to the block

We want to write 16 bytes to the chosen block, and then check correctness of that write. For this, we can use two high level function: C_HLWriteBlock and C_HLReadBlock

For the new Philips card, the transmission password will be like that: ff ff ff ff ff ff. Write this sequence 60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f to the block no.22, sector no.4.

We send the sequence to the module

module address	frame length	command	parameters	CRCH, CRCL
ff	1e	00	60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 04 02 ff ff ff ff ff ff bb	1b a0



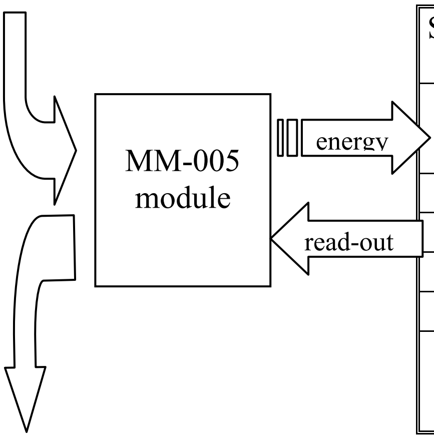
Sector no.	Block no.	Transponder blocks with written data
⋮	⋮	⋮
4	0	
4	1	
4	2	60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f
4	3	xx xx xx xx xx xx ff 07 80 69 ff ff ff ff ff ff
⋮	⋮	⋮

We receive the response:

module address	frame length	response	data	operation code	CRCH, CRCL
01	06	01	-	ff	e9 d5

To verify correctness of the write, send the sequence:

module address	frame length	command	parameters	CRCH, CRCL
ff	0e	02	04 02 ff ff ff ff ff ff bb	99 a5



Sector no.	Block no.	Transponder blocks with written data
⋮	⋮	⋮
4	0	
4	1	
4	2	60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f
4	3	xx xx xx xx xx xx ff 07 80 69 ff ff ff ff ff ff
⋮	⋮	⋮

We receive the response:

module address	frame length	response	data	operation code	CRCH, CRCL
01	16	03	60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f	ff	2f df

It means, that we red-out what was written before.

Example 2 Decrementation the value written in the block

To decrement a value written in a block, we should first to write it in with proper format accepted by the transponder. The values we can to save in the blocks have length of 4 bytes. So let us write in the decimal value of 41 394 (A00 00 a1 b2 in hexadecimal format). According to value format in block, we should write in the sequence of bytes: 00 00 a1 b2 ff ff 5e 4d 00 00 a1 b2 00 ff 00 ff.

Let us write this sequence to the sector no. 4 of the block no. 2, using known C_HL_WriteBlock command:

module address	frame length	command	parameters	CRCH, CRCL
ff	1e	00	00 00 a1 b2 ff ff 5e 4d 00 00 a1 b2 00 ff 00 ff 04 02 ff ff ff ff ff ff bb	52 2b

We receive the response:

module address	frame length	response	data	operation code	CRCH, CRCL
01	06	01		ff	e9 d5

Let us verify this write by means of C_HL_ReadBlock command:

module address	frame length	command	parameters	CRCH, CRCL
ff	0e	02	04 02 ff ff ff ff ff ff bb	99 a5

We receive the response:

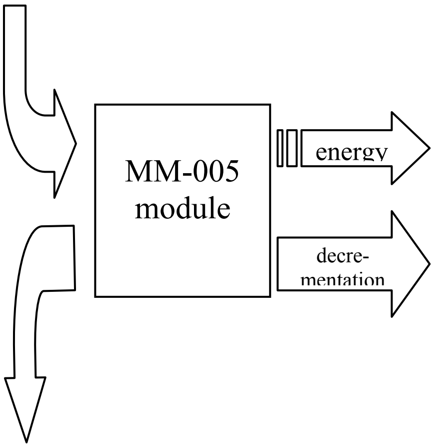
module address	frame length	response	data	operation code	CRCH, CRCL
01	16	03	00 00 a1 b2 ff ff 5e 4d 00 00 a1 b2 00 ff 00 ff	ff	b7 73

It means that, in block np. 5 there is the written 00 00 a1 b2 value.

We can decrement the 00 00 a1 b2 value now. Let us subtract from this value the value of 258 in decimal format, it means 00 00 01 02 in hexadecimal format.
To do this, we can use high level function: C_HL_Decrement.

We send the sequence to the MM-005 module:

module address	frame length	command	parameters	CRCH, CRCL
ff	11	06	04 02 00 00 01 02 ff ff ff ff ff ff bb	cd 45



Sector no.	Block no.	Transponder blocks with written data
⋮	⋮	⋮
4	0	
4	1	
4	2	00 00 a0 b0 ff ff 5f 4f 00 00 a0 b0 00 ff 00 ff
4	3	xx xx xx xx xx xx ff 07 80 69 ff ff ff ff ff ff
⋮	⋮	⋮

We receive the response:

module address	frame length	response	data	operation code	CRCH, CRCL
01	06	07	-	ff	43 73

Let us verify the decrementation by means the C_HL_ReadBlock command:

module address	frame length	command	parameters	CRCH, CRCL
ff	0d	02	04 02 ff ff ff ff ff ff bb	99 a5

We receive the response:

module address	frame length	response	data	operation code	CRCH, CRCL
01	16	03	00 00 a0 b0 ff ff 5f 4f 00 00 a0 b0 00 ff 00 ff	ff	da af

As we can see now, there is the value 00 00 a0 b0 written in the block no. 5. It means, that decrementation process has been performed correctly.

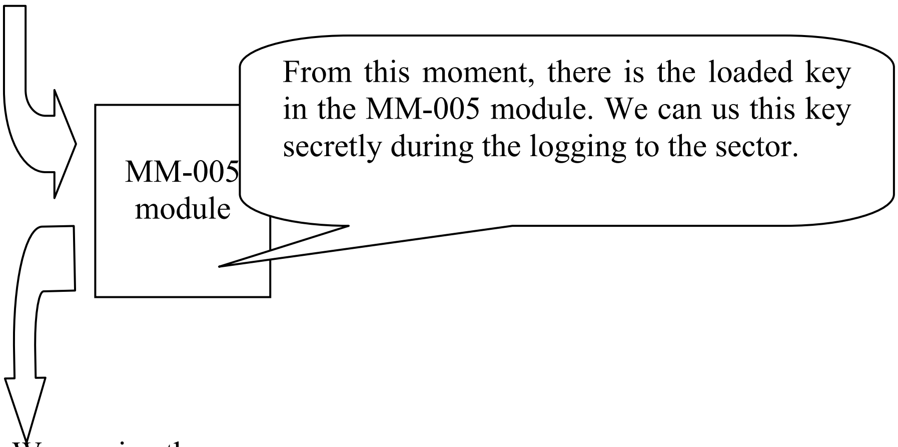
Similar procedure we can perform by means of the C_HL_Increment command. With this command we can increment a value written in the transponder.

Operation the module with high level functions

Example 3 Decrementation the value written in a block

Load a key to the cache key buffer with a help of C_LoadKeyBuffer.
We send the sequence to the MM-005 module:

module address	frame length	command	parameters	CRCH, CRCL
ff	0b	14	ff ff ff ff ff ff	3b f0



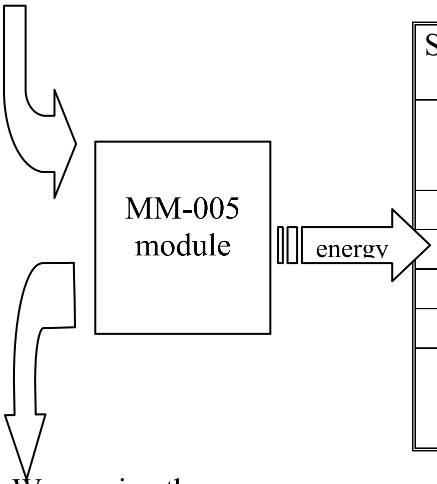
From this moment, there is the loaded key in the MM-005 module. We can us this key secretly during the logging to the sector.

We receive the response:

module address	frame length	response	data	operation code	CRCH, CRCL
01	06	15	-	ff	26 62

With C_TurnOnAntennaPower command, we turn the antenna electromagnetic field on.

module address	frame length	command	parameters	CRCH, CRCL
ff	05	10	-	22 A7



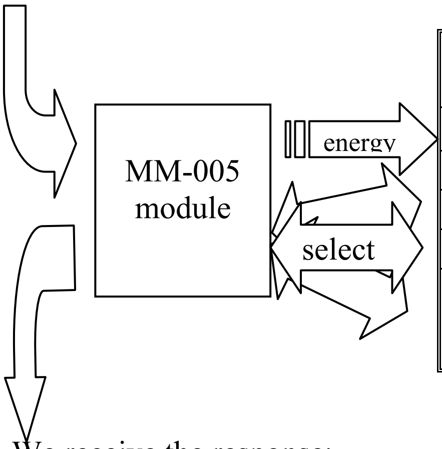
Sector no.	Block no.	Transponder blocks with written data
⋮	⋮	⋮
4	0	
4	1	
4	2	00 00 a0 b0 ff ff 5f 4f 00 00 a0 b0 00 ff 00 ff
4	3	xx xx xx xx xx xx ff 07 80 69 ff ff ff ff ff ff
⋮	⋮	⋮

We receive the response:

module address	frame length	response	data	operation code	CRCH, CRCL
01	06	11	-	ff	ea a6

With C_Select command we trigger anti-collision loop and we choose one of the transponders located in the antenna field.

module address	frame length	command	parameters	CRCH, CRCL
ff	06	12	ff	82 e2



Sector no.	Block no.	Transponder blocks written with data
0	0	ID1 ID2 ID3 ID4 xx xx xx xx xx xx xx xx xx xx xx xx
0	1	
0	2	
0	3	
⋮	⋮	⋮

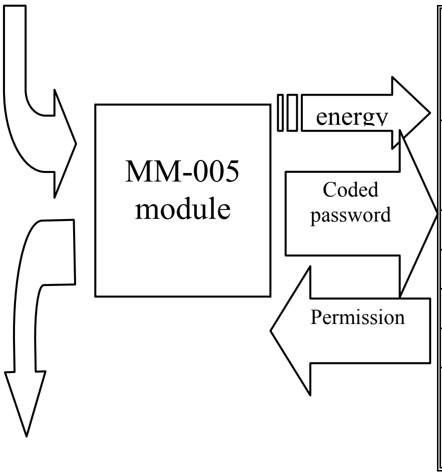
We receive the response:

module address	frame length	response	data	operation code	CRCH, CRCL
01	0a	13	ID1 ID2 ID3 ID4	ff	XX XX

Where: ID1...ID4 – the transponder unique ID number, which has been selected in the antenna field. This number is written in the sector no.0 of the transponder.

Entering the password saved in the cache key buffer, we are logging to the sector no. 4.
The transponder renders the blocks 0 to 3 accessible.
Thanks to the bb parameter, the key inserted in the cache key buffer is treated as a type B key.
Let us use the LoginWithKB command:

module address	frame length	command	parameters	CRCH, CRCL
ff	07	18	04 bb	3b 34



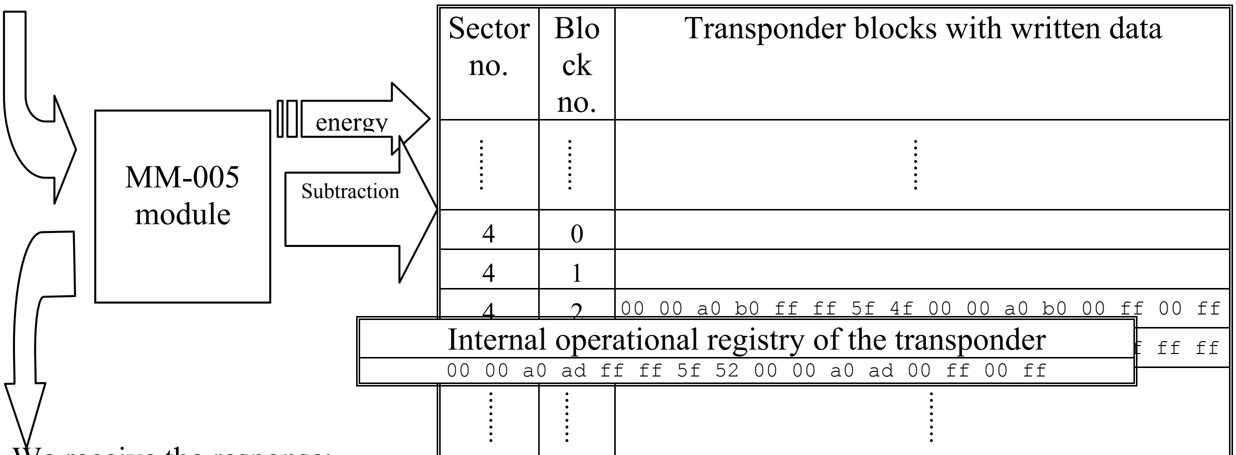
Sector no.	block no.	Transponder blocks with written data
⋮	⋮	⋮
4	0	
4	1	
4	2	`00 00 a0 b0 ff ff 5f 4f 00 00 a0 b0 00 ff 00 ff`
4	3	`xx xx xx xx xx xx ff 07 80 69` **`ff ff ff ff ff ff`**
⋮	⋮	⋮

We receive the response:

module address	frame length	response	data	operation code	CRCH, CRCL
01	06	19	-	ff	63 0f

The ff operation code means, that the fourth sector logging operation has been performed properly.
There is values sequence: 00 00 a0 b0 ff ff 5f 4f 00 00 a0 b0 00 ff 00 ff written in the block no. 2.
It means there is the 00 00 a0 b0 value from which, we substract 3.
Let us begin the decrementation of the block no 2 with the C_DecrementValue command.

module address	frame length	command	parameters	CRCH, CRCL
ff	0a	32	02 00 00 00 03	b7 2d



Sector no.	Block no.	Transponder blocks with written data
⋮	⋮	⋮
4	0	
4	1	
4	2	00 00 a0 b0 ff ff 5f 4f 00 00 a0 b0 00 ff 00 ff
		f ff ff
⋮	⋮	⋮

Internal operational registry of the transponder

```
00 00 a0 ad ff ff 5f 52 00 00 a0 ad 00 ff 00 ff
```

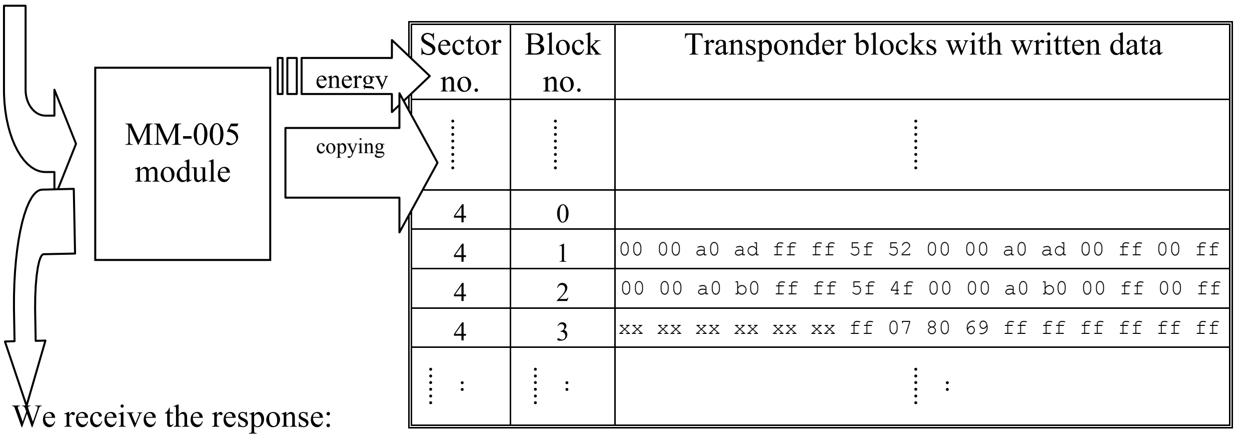
We receive the response:

module address	frame length	response	data	operation code	CRCH, CRCL
01	06	33	-	ff	8a 22

The ff operation code means proper decrementation of the value. This value is saved in the internal operational registry of the transponder but not in the source block. Depending on requirements and conforming to authorization, a user can send the value to the chosen memory block within the same sector.

Let us overwrite the operation registry to the block no.1 with the C_TransferValue command.

module address	frame length	command	parameters	CRCH, CRCL
ff	0a	38	01	65 1e



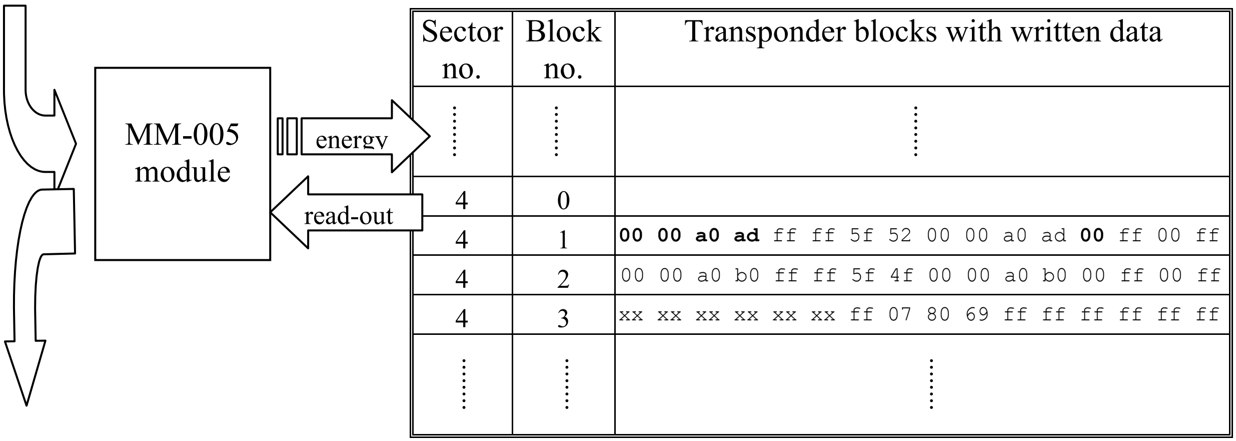
Sector no.	Block no.	Transponder blocks with written data
⋮	⋮	⋮
4	0	
4	1	00 00 a0 ad ff ff 5f 52 00 00 a0 ad 00 ff 00 ff
4	2	00 00 a0 b0 ff ff 5f 4f 00 00 a0 b0 00 ff 00 ff
4	3	xx xx xx xx xx xx ff 07 80 69 ff ff ff ff ff ff
⋮ :	⋮ :	⋮ :

We receive the response:

module address	frame length	response	data	operation code	CRCH, CRCL
01	06	39	-	ff	65 e9

Let us read the value with the C_ReadValue command

module address	frame length	command	parameters	CRCH, CRCL
ff	06	36	01	46 11



Sector no.	Block no.	Transponder blocks with written data
⋮	⋮	⋮
4	0	
4	1	**00 00 a0 ad** ff ff 5f 52 00 00 a0 ad **00** ff 00 ff
4	2	00 00 a0 b0 ff ff 5f 4f 00 00 a0 b0 00 ff 00 ff
4	3	xx xx xx xx xx xx ff 07 80 69 ff ff ff ff ff ff
⋮	⋮	⋮

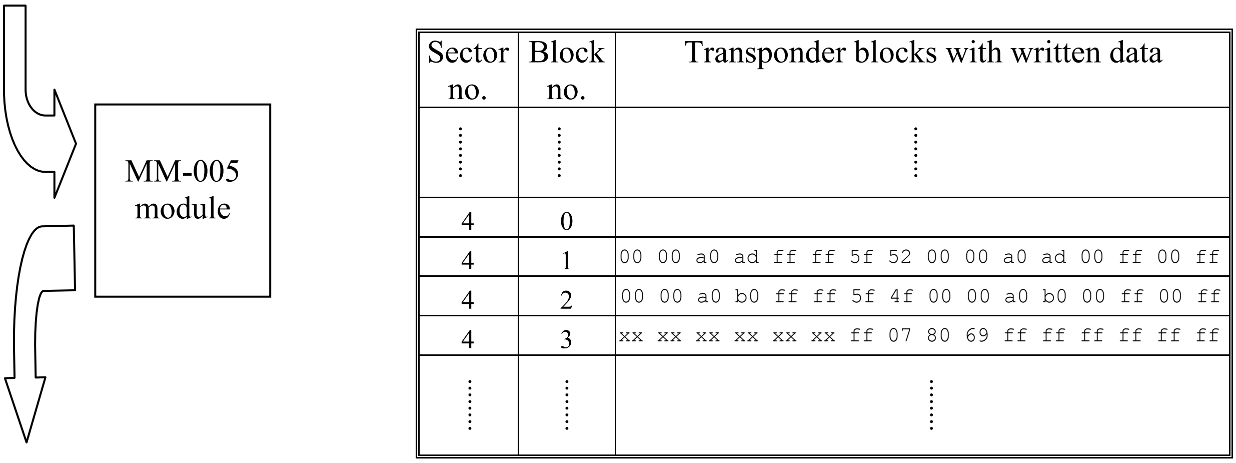
We receive the response:

module address	frame length	response	data	operation code	CRCH, CRCL
01	0b	37	00 00 a0 ad 00	ff	76 0e

The read-out value is 00 00 a0 ad. It means that all operations have been performed successfully.

Switching the antenna field off using the C_TurnOffAntennaPower command:

module address	frame length	command	parameters	CRCH, CRCL
ff	05	44	-	38 d6



Sector no.	Block no.	Transponder blocks with written data
⋮	⋮	⋮
4	0	
4	1	00 00 a0 ad ff ff 5f 52 00 00 a0 ad 00 ff 00 ff
4	2	00 00 a0 b0 ff ff 5f 4f 00 00 a0 b0 00 ff 00 ff
4	3	xx xx xx xx xx xx ff 07 80 69 ff ff ff ff ff ff
⋮	⋮	⋮

We receive the response:

module address	frame length	response	data	operation code	CRCH, CRCL
01	0b	45	-	ff	28 dd

NETRONIX product overview is available on website:
http://www.netronix.pl/