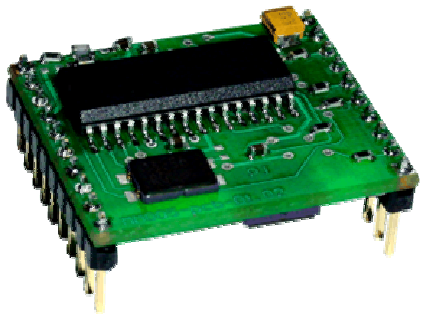
NETRONIX

Dokumentacja Techniczna

MM-005

MM005-doc-00.07
odniesienie do MM005-c-00.05+



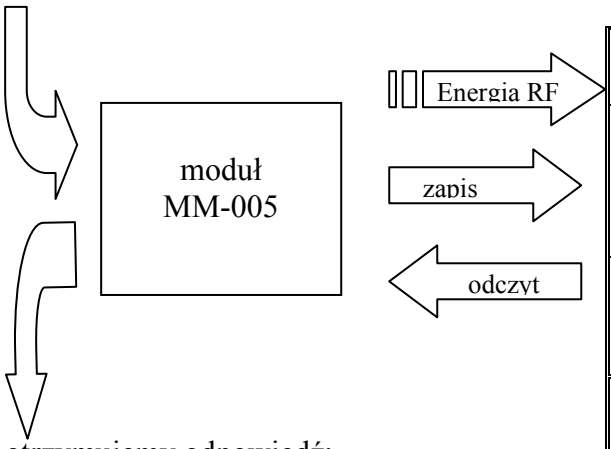
Spis Treści

Wprowadzenie

MM-005 jest modułem działającym na zasadzie bezstykowego odczytywania i zapisywania danych z/do transpondera Mifare® (RFID). Obsługiwany jest za pomocą łącza RS-232 z poziomami napięć zgodnymi z TTL.
Urządzenie działa na zasadzie:
pytanie (z urządzenia nadrzędnego - hosta) - akcja (modułu) - odpowiedź (modułu).

Do modułu MM-005 wysyłamy pytanie-rozkaz:

adres modułu	dł. ramki	rozkaz	dane	CRCH,CRCL
xx	xx	xx	xx xx xx	xx xx



sektor nr.	block nr.	bloki transpondera z zapisanymi danymi
0	0	B1.....B16
	1	B1.....B16
	2	B1.....B16
	3	B1.....B16
⋮	⋮	⋮
15 lub 39		

otrzymujemy odpowiedź:

adres modułu	dł. ramki	odpowiedź	dane	kod operacji	CRCH,CRCL
xx	xx	xx	xx xx	xx	xx xx

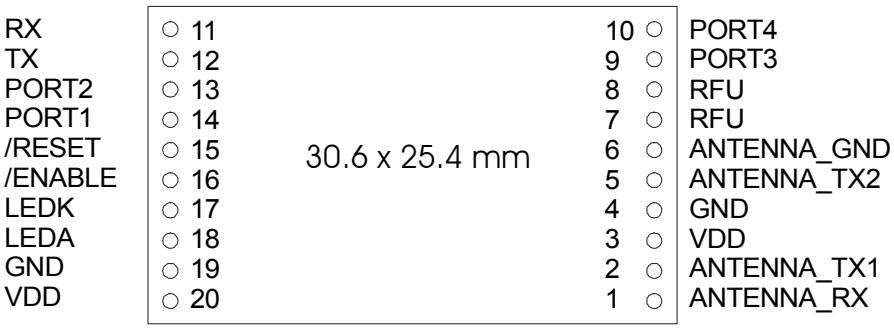
Moduł posiada 2 porty użytkownika (1-no bitowe) które mogą być odczytywane lub zapisywane.
Do MM-005 należy podłączyć antenę w postaci cewki powietrznej która emitowała będzie pole elektromagnetyczne a tym samym zasilała będzie transponder znajdujący się w tym polu.

Dane techniczne

Napięcie zasilające Uz:	4,5...5,5V
Prąd zasilający:	1...55 mA
Znamionowa częstotliwość RF pracy modułu	13,56 MHz
Prędkość danych odbieranych z transpondera	106 kb/s (10us/bit)
Czas zapisu bloku danych	6ms + transmisja RS
Czas odczytu bloku danych	2,5ms + transmisja RS
Czas typowej tranzakcji biletowej *	70ms + transmisja RS
Wydolność prądowa wyjść: port1...4 i RS-TX	10mA
Max. odległość odczytu/zapisu transponderów w zależności od zastosowanej anteny:	5...10 cm
Antena	Zewnętrzna wraz z kondensatorami ustalającymi rezonans dla 13,56MHz
Transmisja.	1200...115200 b/s, 8 bitów danych, 1 bit stopu, bez bitu parzystości, na poziomach napięciowych zgodnych z TTL (fabrycznie: 9600 b/s)

*odświeżenie 2 wartości i odczytanie 2 bloków

Opis wyprowadzeń

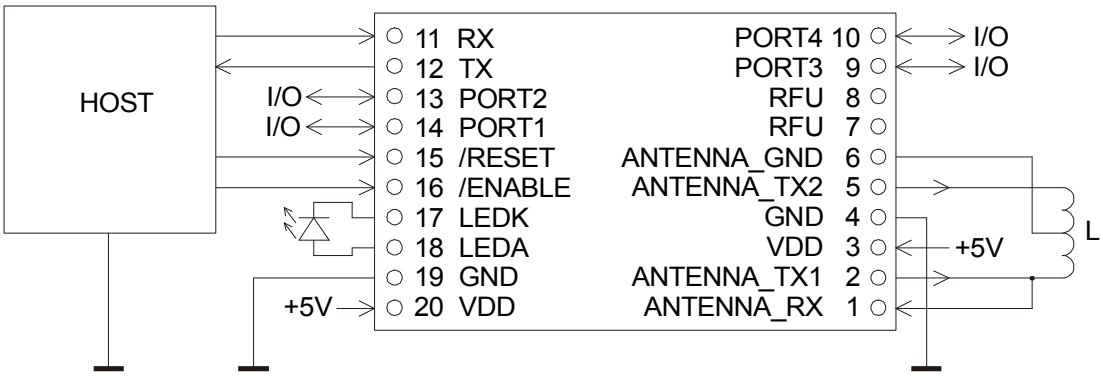


Wyprowadzenia widziane od strony elementów

1. ANTENNA_RX – wejście odbierające dane z transpondera, podłącza się do anteny
2. ANTENNA_TX1 – jedno z wyjść dostarczających energię do anteny
3. VDD – zasilanie +
4. GND – masa układu (zasilanie -)
5. ANTENNA_TX2 – jedno z wyjść dostarczających energię do anteny
6. ANTENNA_GND – masa anteny, podłącza się odczep anteny
7. NC (not connected)??
8. NC (not connected)??
9. PORT3 - wyjście/wejście użytkownika *
10. PORT4 - wyjście/wejście użytkownika *
11. RS232-RX – wejście łacza RS-232 na poziomach napięciowych zgodnych z TTL *
12. RS232-TX – wyjście łacza RS-232 na poziomach napięciowych zgodnych z TTL *
13. PORT2 - wyjście/wejście użytkownika *
14. PORT1 - wyjście/wejście użytkownika *
15. /RESET – wejście zewnętrznego resetu modułu , aktywny stan L *
16. /ENABLE – wejście aktywujące moduł , aktywny stan L *
17,18. LEDA, LEDK – wyjścia do podłączenia zewnętrznego LED, odpowiednio anoda i katoda
19. GND – masa układu (zasilanie -)
20. VDD – zasilanie +

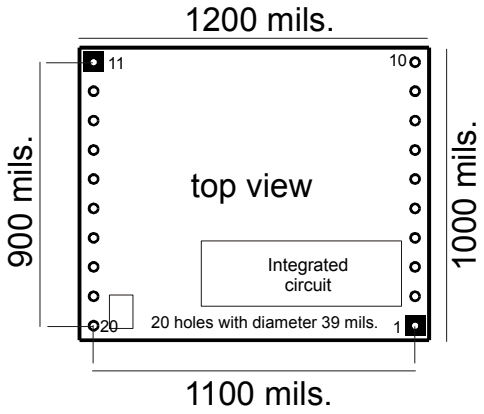
* Posiada rezystor zabezpieczający 100 Ohm'ów

Schemat połączeń



Schemat połączeń modułu z elementami zewnętrznymi

Wymiary modułu



Ogólny format ramki rozkazu dla czytnika

Adres modułu	Długość ramki	Rozkaz	Parametry 1..n	CRCH	CRCL
1 bajt	1 bajt	1 bajt	n * bajtów	1 bajt	1 bajt

Gdzie:

AdresModułu - unikalny adres modułu w systemie

Jeżeli:

AdresModułu = 0 to nie odpowie żaden moduł

AdresModułu = 0xFF to odpowiedzą wszystkie moduły w sieci

DługośćRamki - całkowita ilość bajtów ramki

Rozkaz - wartość parzysta

Parametry1...n - występują opcjonalnie i uzależnione są od rozkazu

CRCH, **CRCL** - odpowiednio starszy i młodszy bajt CRC16

Ogólny format ramki odpowiedzi z czytnika

Adres modułu	Długość ramki	Odpowiedź	Parametry 1..n	Kod operacji	CRCH	CRCL
1 bajt	1 bajt	1 bajt	n * bajtów	1 bajt	1 bajt	1 bajt

Gdzie:

AdresModułu - rzeczywisty nadany adres modułu odpowiadającego

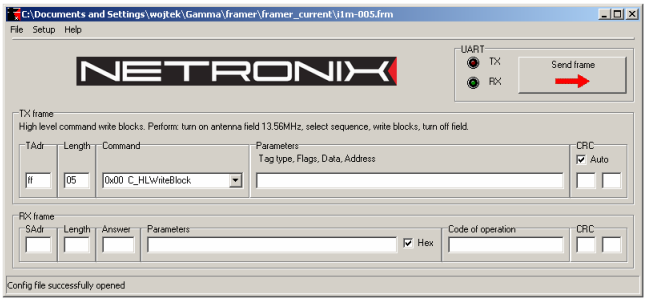
DługośćRamki - całkowita ilość bajtów ramki odpowiedzi

Odpowiedź = **Rozkaz** + 1 (wartość nieparzysta)

Parametry1...n - występują opcjonalnie i zależą od rozkazu

KodOperacji - informacja o poprawności wykonanego rozkazu

CRCH, **CRCL** - odpowiednio starszy i młodszy bajt CRC16



Moduł można przetestować za pomocą narzędziowego, bezpłatnego programu FRAMER ułatwiającego pracę z ramkami.

Opis rozkazów

Oznaczenia:
BlockNo – wartość=(0…3) dla MF1ICS50
SectorNo – wartość=(0…0x0F) czyli 16 sektorów dla MF1ICS50

Key1...6 – klucz za pomocą którego chcemy logować się do sektora
KeyType – typ klucza za pomocą którego chcemy się logować do sektora.
0xAA dla klucza typu A
0xBB dla klucza typu B.

Rozkazy wysokiego poziomu

Rozkazy wysokiego poziomu charakteryzują się tym, że mogą one przeprowadzić pełny proces komunikacji z kartą MIFARE®. Oznacza to, że załączenie pola, selekcja karty, autoryzacja, proces właściwy, i wyłączenie pola zostaną przeprowadzone automatycznie.

Zapis 16 bajtów do bloku

Nazwa rozkazu - zapytania	Kod rozkazu	Parametry
C_HLWriteBlock	0x00	Data1...16, SectorNo, BlockNo, Key1...6, KeyType

Data1…16 – dane do zapisu
SectorNo – docelowy sektor
BlockNo – docelowy blok w ramach sektora.

Nazwa rozkazu - odpowiedzi	Kod odpowiedzi	Parametry
A_HLWriteBlock	0x01	OperationKod

OperationKod - 0xff-zapis poprawny

Odczyt 16 bajtów z bloku

Nazwa rozkazu - zapytania	Kod rozkazu	Parametry
C_HLReadBlock	0x02	SectorNo, BlockNo, Key1...6, KeyType

SectorNo – źródłowy sektor
BlockNo – źródłowy blok w ramach sektora.

Nazwa rozkazu - odpowiedzi	Kod odpowiedzi	Parametry
A_HLReadBlock	0x03	Data1...16, OperationKod

Data1...16 – odczytane dane z bloku
OperationKod - 0xff-odczyt poprawny

Inkrementowanie wartości zapisanej w bloku

Rozkaz ten generuje sekwencję odczytu wartości z bloku do wewnętrznego rejestru operacyjnego transpondera „data register" , inkrementuje tą wartość i wynik zapisuje ponownie do bloku źródłowego.

Nazwa rozkazu - zapytania	Kod rozkazu	Parametry
C_HLIncrement	0x04	SectorNo, BlockNo, Value1...4, Key1...6, KeyType

SectorNo – sektor na którym będzie wykonana operacja
BlockNo – blok inkrementowany
Value1...4 – wartość którą chcemy dodać do wartości w bloku " BlockNo"

Nazwa rozkazu - odpowiedzi	Kod odpowiedzi	Parametry
A_HLIncrement	0x05	OperationKod

OperationKod - 0xff-inkrementacja poprawna
Aby ta operacja była przeprowadzona pomyślnie, blok BlockNo powinien być sformatowany jako "Value" .

Dekrementowanie wartości zapisanej w bloku

Rozkaz ten generuje sekwencję odczytu wartości z bloku do wewnętrznego rejestru operacyjnego transpondera „data register" , dekrementuje tą wartość i wynik zapisuje ponownie do bloku źródłowego.

Nazwa rozkazu - zapytania	Kod rozkazu	Parametry
C_HLDecrement	0x06	SectorNo, BlockNo, Value1...4, Key1...6, KeyType

SectorNo – sektor na którym będzie wykonana operacja
BlockNo – blok dekrementowany
Value1...4 – wartość którą chcemy odjąć od wartości w bloku " BlockNo"

Nazwa rozkazu - odpowiedzi	Kod odpowiedzi	Parametry
A_HLDecrement	0x07	

OperationKod - 0xff-dekrementowanie poprawne
Aby ta operacja była przeprowadzona pomyślnie, blok BlockNo powinien być sformatowany jako "Value" .

Rozkazy niskiego poziomu

Rozkazy niskiego poziomu charakteryzują się tym że można ich używać w dowolnych sekwencjach bez wielokrotnego załączania/wyłączania pola, wielokrotnej selekcji transpondera i wielokrotnego logowania się do jego sektora. Aplikacja używająca tych rozkazów jest w stanie mieć pełną kontrolę nad wieloma transponderami znajdującymi się w polu

Załączenie pola elektromagnetycznego anteny

Nazwa rozkazu - zapytania	Kod rozkazu	Parametry
C_TurnOnAntennaPower	0x10	-

Nazwa rozkazu - odpowiedzi	Kod odpowiedzi	Parametry
A_TurnOnAntennaPower	0x11	OperationKod

OperationKod – zawsze 0xff

Wyłączenie pola elektromagnetycznego anteny

Nazwa rozkazu - zapytania	Kod rozkazu	Parametry
C_TurnOffAntennaPower	0x44	-

Nazwa rozkazu - odpowiedzi	Kod odpowiedzi	Parametry
A_TurnOffAntennaPower	0x45	OperationKod

OperationKod – zawsze 0xff

Wywołanie ogólne dla kart

Rozkaz ten uruchamia pętlę antykolizyjną dla transponderów znajdujących się w polu anteny, wybiera jeden z nich i zwraca jego numer ID.

Nazwa rozkazu - zapytania	Kod rozkazu	Parametry
C_Select	0x12	TypeOfRequest

TypeOfRequest – decydyje o rodzaju wywołania transponderów

Gdy: TypeOfRequest=0xff to wywoływane są transpondery znajdujące się w stanie „Idle” oraz znajdujące się w stanie „Halt”.
Takie wywołanie określane jest jako „Request All”

TypeOfRequest=0x01 to wywoływane są tylko transpondery znajdujące się w stanie „Idle”.
Takie wywołanie określane jest jako „Request Standard”

Nazwa rozkazu - odpowiedzi	Kod odpowiedzi	Parametry
A_Select	0x13	ID1…4, OperationKod

ID1…4 – Numer karty , która została wyselekcjonowana

OperationKod - 0xff-selekcja poprawna

Załadowanie klucza do podręcznego bufora klucza

Bufor ten jest zawarty w MM-005 i jest w postaci pamięci RAM. W buforze tym można zapamiętać 1 klucz. Niemożliwe jest odczytanie tego klucza a jedynie logowanie się do sektorów przy jego użyciu.

Nazwa rozkazu - zapytania	Kod rozkazu	Parametry
C_LoadKeyBuffer	0x14	Key1…6

Key1…6 – klucz, który ma zostać załadowany do bufora klucza

Nazwa rozkazu - odpowiedzi	Kod odpowiedzi	Parametry
A_LoadKeyBuffer	0x15	OperationKod

OperationKod - 0xff-operacja poprawna

Załadowanie klucza do nie ulotnej pamięci kluczy

Bufor ten jest zawarty w MM-005 i jest w postaci pamięci EEPROM. W buforze tym, pod kolejnymi pozycjami, można zapamiętać do 32 kluczy. Niemożliwe jest odczytanie tych kluczy a jedynie logowanie się do sektorów przy ich użyciu.

Nazwa rozkazu - zapytania	Kod rozkazu	Parametry
C_LoadEEKeyBuffer	0x16	Key1…6, EEKeyNo

Key1…6 – klucz, który ma zostać załadowany do nie ulotnej pamięci kluczy
EEKeyNo – numer pozycji w pamięci =(0...0x1F)

Nazwa rozkazu - odpowiedzi	Kod odpowiedzi	Parametry
A_LoadEEKeyBuffer	0x17	OperationKod

OperationKod - 0xff-operacja poprawna

Logowanie do sektora przy użyciu podręcznego bufora klucza

Nazwa rozkazu - zapytania	Kod rozkazu	Parametry
C_LoginWithKB	0x18	SectorNo, KeyType

SectorNo – numer sektora do którego chcemy się zalogować
KeyType – decyduje o tym jak klucz znajdujący się w buforze klucza będzie traktowany podczas logowania.
Gdy KeyType =0xAA to klucz traktowany będzie jak typu A, jeżeli KeyType =0xBB to klucz traktowany będzie jak typu B

Nazwa rozkazu - odpowiedzi	Kod odpowiedzi	Parametry
A_LoginWithKB	0x19	OperationKod

OperationKod - 0xff-logowanie poprawne

Logowanie do sektora przy użyciu nie ulotnej pamięci kluczy

Nazwa rozkazu - zapytania	Kod rozkazu	Parametry
C_LoginWithEE	0x1a	SectorNo, KeyType, EEKeyNo

SectorNo – numer sektora do którego chcemy się zalogować
KeyType – decyduje o tym jak klucz z pozycji EEKeyNo będzie traktowany podczas logowania.
Gdy KeyType =0xAA to klucz traktowany będzie jak typu A, jeżeli KeyType =0xBB to klucz traktowany będzie jak typu B
EEKeyNo – numer pozycji w nie ulotnej pamięci kluczy (0...1F)

Nazwa rozkazu - odpowiedzi	Kod odpowiedzi	Parametry
A_LoginWithEE	0x1b	OperationKod

OperationKod - 0xff-logowanie poprawne

Zapis 16 bajtów do bloku

Nazwa rozkazu - zapytania	Kod rozkazu	Parametry
C_WriteBlock	0x1c	Data1...16, Blok nr

Data1...16 – dane do zapisu
Blok nr – numer bloku do którego będą zapisane dane

Nazwa rozkazu - odpowiedzi	Kod odpowiedzi	Parametry
A_WriteBlock	0x1d	OperationKod

OperationKod - 0xff-zapis poprawny

Odczyt 16 bajtów z bloku

Nazwa rozkazu - zapytania	Kod rozkazu	Parametry
C_ReadBlock	0x1e	Blok nr

Blok nr – numer bloku z którego będą odczytane dane

Nazwa rozkazu - odpowiedzi	Kod odpowiedzi	Parametry
A_ReadBlock	0x1f	OperationKod

OperationKod - 0xff-odczyt poprawny

Kopiowanie 16 bajtów z bloku do bloku

Nazwa rozkazu - zapytania	Kod rozkazu	Parametry
C_CopyBlock	0x20	SourceBlockNo, TargetBlockNo

SourceBlockNo – Numer bloku źródłowego
TargetBlockNo - Numer bloku docelowego

Nazwa rozkazu - odpowiedzi	Kod odpowiedzi	Parametry
A_CopyBlock	0x21	OperationKod

OperationKod - 0xff-odczyt poprawny
Kopiowanie można przeprowadzić w ramach tego samego sektora do którego jesteśmy zalogowani.

Zapisanie wartości do bloku

Rozkaz ten zamienia wartość 4-bajtową na zapis zgodny z formatem Value i zapisuje w postaci 16-u bajtów do bloku podanego jako BlockNo.

Nazwa rozkazu - zapytania	Kod rozkazu	Parametry
C_WriteValue	0x34	Value1...4, BackupBlockNo, BlockNo

Value1...4 – wartość zapisywana do bloku "BlockNo"
BackupBlockNo – adres bloku backup'u
BlockNo – numer docelowego bloku

Nazwa rozkazu - odpowiedzi	Kod odpowiedzi	Parametry
A_WriteValue	0x35	OperationKod

OperationKod - 0xff-odczyt poprawny

Odczytanie wartości z bloku

Nazwa rozkazu - zapytania	Kod rozkazu	Parametry
C_ReadValue	0x36	BlockNo

BlockNr – numer źródłowego bloku

Nazwa rozkazu - odpowiedzi	Kod odpowiedzi	Parametry
A_ReadValue	0x37	Value1...4, BackupBlockNo, OperationKod

Value1...4 – odczytana wartość
BackupBlockNo – odczytany adres bloku backup'u
OperationKod - 0xff-operacja poprawna
Aby OperationKod=0xff w bloku źródłowym musi znajdować się zapis zgodny z formatem „Value".

Inkrementowanie wartości zapisanej w bloku

Rozkaz ten dodaje argument Value1...4 do wartości zapisanej w bloku BlockNr. Rezultat działania pozostaje w wewnętrznym buforze danych „data register" w transponderze. Aby wynik został zapisany do sektora należy użyć rozkazu C_TransferValue. W ten sposób argument działania może być pobrany z pewnego bloku a wynik zapisany do innego co zwiększa bezpieczeństwo danych.

Nazwa rozkazu - zapytania	Kod rozkazu	Parametry
C_IncrementValue	0x30	BlockNo, Value1...4

BlockNo – numer bloku inkrementowanego
Value1...4 – wartość dodawana do bloku "BlockNo"

Nazwa rozkazu - odpowiedzi	Kod odpowiedzi	Parametry
A_IncrementValue	0x31	OperationKod

OperationKod - 0xff-operacja poprawna
Operacje można przeprowadzić w ramach tego samego sektora do którego jesteśmy zalogowani.

Dekrementowanie wartości zapisanej w bloku

Rozkaz ten odejmuje argument Value1...4 od wartości zapisanej w bloku BlockNr. Rezultat działania pozostaje w wewnętrznym buforze danych „data register" w transponderze. Aby wynik został zapisany do sektora należy użyć rozkazu C_TransferValue. W ten sposób argument działania może być pobrany z pewnego bloku a zapisany do innego co zwiększa bezpieczeństwo danych.

Nazwa rozkazu - zapytania	Kod rozkazu	Parametry
C_DecrementValue	0x32	BlockNo, Value1...4

BlockNo – numer bloku dekrementowanego
Value1...4 – wartość odejmowana od bloku "BlockNo"

Nazwa rozkazu - odpowiedzi	Kod odpowiedzi	Parametry
A_DecrementValue	0x33	OperationKod

OperationKod - 0xff-operacja poprawna
Operacje można przeprowadzić w ramach tego samego sektora do którego jesteśmy zalogowani.

Transfer danych z rejestru transpondera do wybranego bloku

Nazwa rozkazu - zapytania	Kod rozkazu	Parametry
C_TransferValue	0x38	BlockNo

BlockNo – numer docelowego bloku

Nazwa rozkazu - odpowiedzi	Kod odpowiedzi	Parametry
A_TransferValue	0x39	OperationKod

OperationKod - 0xff-operacja poprawna
Rozkaz ten jest wykorzystywany do przesłania wyniku obliczeń z wewnętrznego rejestru danych „data register" do wybranego bloku pamięci. Operację taką wykonuje się po wcześniejszym użyciu rozkazów C_IncrementValue lub C_DecrementValue.

Wprowadzenie transpondera w stan uśpienia

Rozkaz ten stosuje się gdy chcemy uśpić aktywny transponder (po poprawnym C_Select i/lub C_LoginWithEE i/lub C_LoginWithKB). Po takiej operacji możemy ponownie wyselekcjonować jeden z pozostałych transponderów w polu i przeprowadzać na nim żądane operacje. Aby wyselekcjonować jeden z pozostałych transponderów należy użyć rozkazu C_Select z parametrem 0x01. W taki sposób, w przypadku wielu transponderów znajdujących się w polu anteny, możemy obsłużyć wszystkie transpondery po kolei.

Nazwa rozkazu - zapytania	Kod rozkazu	Parametry
C_Halt	0x40	-

Nazwa rozkazu - odpowiedzi	Kod odpowiedzi	Parametry
A_Halt	0x41	OperationKod

OperationKod - 0xff-operacja poprawna
Aby ponownie przeprowadzić komunikację z transponderem znajdującym się w stanie „Halt" należy użyć rozkazu C_Select z parametrem 0xff.

Rozkazy zarządzające portami I/O użytkownika

Po wystąpieniu stanu reset modułu wszystkie porty I/O są zdefiniowane jako wejście czyli widziane z zewnątrz jako wysoka impedancja.

Ustawienie portu jako wyjście z jednoczesnym ustawieniem stanu

Nazwa rozkazu - zapytania	Kod rozkazu	Parametry
C_WriteUserPort	0x50	PortNo, Value

PortNo – numer portu (1...4)
Value – bit do zapisu

Nazwa rozkazu - odpowiedzi	Kod odpowiedzi	Parametry
A_WriteUserPort	0x51	OperationKod

OperationKod – zawsze 0xff

Ustawienie portu jako wejście z jednoczesnym odczytem stanu

Nazwa rozkazu - zapytania	Kod rozkazu	Parametry
C_ReadUserPort	0x52	PortNo

PortNo – numer portu (1...4)

Nazwa rozkazu - odpowiedzi	Kod odpowiedzi	Parametry
A_ReadUserPort	0x53	Value , OperationKod

Value – odczytana wartość portu
OperationKod – zawsze 0xff

Rozkazy dodatkowye

Ustawienie wzmocnienia układu odbierającego dane z transpondera

Nazwa rozkazu - zapytania	Kod rozkazu	Parametry
C_SetGain	0x60	NewGain

NewGain – nowa wartość wzmocnienia układu odbierającego dane z transpondera (0…3)

Nazwa rozkazu - odpowiedzi	Kod odpowiedzi	Parametry
A_SetGain	0x61	OperationKod

OperationKod - 0xff-operacja poprawna

Ustawienie prędkości UART'a

Nazwa rozkazu - zapytania	Kod rozkazu	Parametry
C_SetUartSpeed	0x62	UartSpeed

UartSpeed – prędkość portu UART =(1…8) gdzie 1=1200b/s …8=115200b/s

Nazwa rozkazu - odpowiedzi	Kod odpowiedzi	Parametry
A_SetUartSpeed	0x63	OperationKod

OperationKod – zawsze 0xff
MM-005 odpowiada na prędkości dotychczasowej po czym zmienia prędkość na „nową" i oczekuje przez czas 10 sekund. W tym czasie MM-005 powinien odebrać jakąkolwiek ramkę z poprawnym CRC aby pozostać na nowej prędkości. W przypadku przeciwnym urządzenie ponownie powróci do prędkości poprzedniej.

Nadanie modułowi adresu na magistrali RS

Nazwa rozkazu - zapytania	Kod rozkazu	Parametry
C_SetSlaveAdres	0x64	NewSlaveAdr

NewSlaveAdr – nowy adres jaki nadajemy urządzeniu

Nazwa rozkazu - odpowiedzi	Kod odpowiedzi	Parametry
A_SetSlaveAdres	0x65	OperationKod

OperationKod – zawsze 0xff

Odczyt numeru wersji oprogramowania modułu

Nazwa rozkazu - zapytania	Kod rozkazu	Parametry
C_SoftVersion	0xfe	-

Nazwa rozkazu - odpowiedzi	Kod odpowiedzi	Parametry
A_SoftVersion	0xff	Data1...n, OperationKod

Data1...n – zapisana w kodach ascii wersja oprogramowania
OperationKod - zawsze 0xff

Obliczanie CRC

CRC obliczane jest jako x^16+x^12+x^5+1 z wartością początkową 0x0000. CRC obliczane jest na podstawie wszystkich bajtów z wyjątkiem CRCH i CRCL
Przykładowa procedura obliczania CRC napisana w języku C:

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

}
```

gdzie:
*Zadr -jest wskaźnikiem do pierwszego bajtu danych
Ile -mówi ile jest bajtów danych z których liczone będzie CRC
*DoAdr -jest wskaźnikiem do obliczonego CRC

Przykłady obsługi transpondera Mifare® przy pomocy MM-005

Założenia:

- Komunikaty wysyłane są jako rozgłoszeniowe (do wszystkich modułów w sieci, AdresModułu=ff)

Typowa ramka rozkazu:

adres modułu	dł.ramki	rozkaz	dane	CRCH,CRCL
ff	XX	XX	XX XX XX XX	XX XX

- Zakładamy , że czytnikowi wcześniej został nadany adres 01 poprzez funkcję C_SlaveAddressSet. Oznacza to że odpowiadał będzie czytnik o adresie 01.

Typowa ramka odpowiedzi:

adres modułu	dł.ramki	odpowiedź	dane	kod operacji	CRCH,CRCL
01	XX	XX	XX XX XX XX	XX	XX XX

- Transponder używany do testów ma ustawione bloki konfiguracyjne (sector trailer) tak jak nowy transponder produkcji Philips'a czyli
  ff ff ff ff ff ff ff 07 80 69 ff ff ff ff ff ff
  Oznacza to że transponder zezwoli na zapisywanie i odczytywanie bloków danych oraz inkrementowanie, dekrementowanie i przeprowadzanie operacji „transfer" na wartościach. Przy takiej konfiguracji wszystkie te operacje można przeprowadzać przy użyciu haseł typu A lub B.
  (Jeżeli posiadamy transponder innego producenta to klucze typu A oraz B mogą wyglądać następująco: a0 a1 a2 a3 a4 a5 oraz b0 b1 b2 b3 b4 b5)

Praca z funkcjami wysokiego poziomu

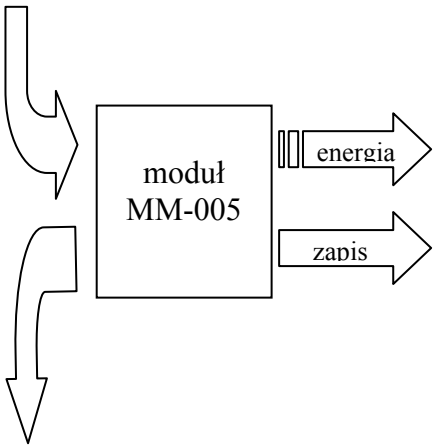
Przykład 1 Zapis 16 bajtów do bloku

Chcemy zapisać 16 bajtów do wybranego bloku i sprawdzić poprawność tego zapisu.
Do tego celu można użyć 2 funkcji wysokiego poziomu C_HLWriteBlock i C_HLReadBlock

Dla nowej kary Philips'a hasła transportowe mają postać ff ff ff ff ff ff.
Zapiszmy sekwencję 60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f do sektora nr. 4 do bloku nr.2.

Do modułu MM-005 wysyłamy sekwencję:

adres modułu	dł. ramki	roz-kaz	parametry	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



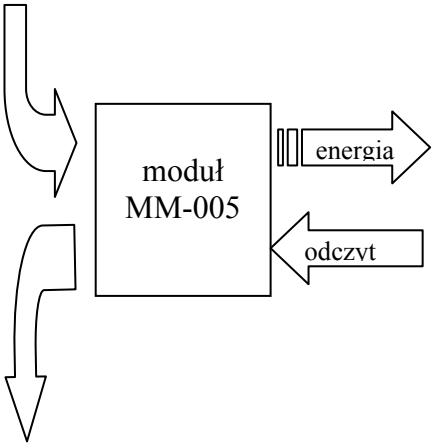
sektor nr.	blok nr.	bloki transpondera z zapisanymi danymi
⋮	⋮	⋮
4	0	
4	1	
4	2	60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f
4	3	xx xx xx xx xx xx ff 07 80 69 ff ff ff ff ff ff
⋮	⋮	⋮

otrzymujemy odpowiedź:

adres modułu	dł. ramki	odpowiedź	parametry	kod operacji	CRCH, CRCL
01	06	01	-	ff	e9 d5

Aby zweryfikować poprawność zapisu należy wysłać sekwencję:

adres modułu	dł. ramki	rozkaz	parametry	CRCH, CRCL
ff	0e	02	04 02 ff ff ff ff ff ff bb	99 a5



sektor nr.	blok nr.	bloki transpondera z zapisanymi danymi
⋮	⋮	⋮
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
⋮	⋮	⋮

otrzymujemy odpowiedź:

adres modułu	dł. ramki	odpowiedź	parametry	kod operacji	CRCH, CRCL
01	16	03	60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f	ff	2f df

Czyli odczytaliśmy to co poprzednio było zapisane.

<u>Przykład 2</u> Dekrementowanie wartości zapisanej w bloku

Aby dekrementować wartość zapisaną w bloku musimy najpierw zapisać ją w odpowiednim formacie akceptowanym przez transponder. Wartości jakie możemy przechowywać w blokach transpondera są 4-o bajtowe. Zapiszmy zatem wartość dziesiętną 41 394 czyli 00 00 a1 b2 w zapisie hexadecymalnym. Zgodnie z formatem wartości w bloku powinniśmy zapisać ciąg bajtów w postaci: 00 00 a1 b2 ff ff 5e 4d 00 00 a1 b2 00 ff 00 ff.

Zapiszmy tą sekwencję do sektora nr.4 do bloku nr.2 korzystając ze znanego już rozkazu C_HL_WriteBlock.

adres modułu	dł. ramki	rozkaz	parametry	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

otrzymujemy odpowiedź:

adres modułu	dł. ramki	odpowiedź	parametry	kod operacji	CRCH, CRCL
01	06	01		ff	e9 d5

Zweryfikujmy zapis za pomocą rozkazu C_HL_ReadBlock:

adres modułu	dł. ramki	rozkaz	parametry	CRCH, CRCL
ff	0e	02	04 02 ff ff ff ff ff ff bb	99 a5

otrzymujemy odpowiedź:

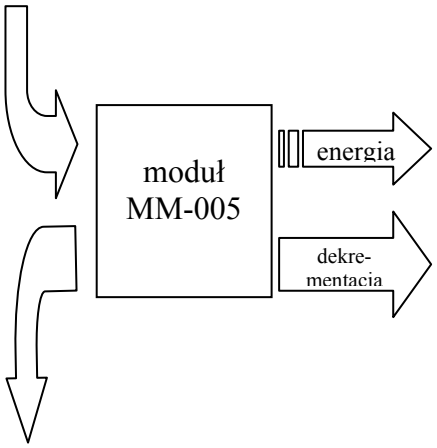
adres modułu	dł. ramki	odpowiedź	parametry	kod operacji	CRCH, CRCL
01	16	03	00 00 a1 b2 ff ff 5e 4d 00 00 a1 b2 00 ff 00 ff	ff	b7 73

Czyli w bloku nr. 5 mamy zapisaną wartość 00 00 a1 b2.

Teraz możemy przystąpić do dekrementacji wartości 00 00 a1 b2. Od tej wartości odejmijmy 258 w zapisie dziesiętnym czyli 00 00 01 02 w zapisie hexadecymalnym.
Do tego celu można użyć funkcji wysokiego poziomu C_HL_Decrement

Do modułu MM-005 wysyłamy sekwencję:

adres modułu	dł. ramki	rozkaz	parametry	CRCH, CRCL
ff	11	06	04 02 00 00 01 02 ff ff ff ff ff ff bb	cd 45



sektor nr.	blok nr.	bloki transpondera z zapisanymi danymi
⋮	⋮	⋮
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
⋮	⋮	⋮

otrzymujemy odpowiedź:

adres modułu	dł. ramki	odpowiedź	parametry	kod operacji	CRCH, CRCL
01	06	07	-	ff	43 73

Zweryfikujmy dekrementację za pomocą rozkazu C_HL_ReadBlock:

adres modułu	dł. ramki	rozkaz	parametry	CRCH, CRCL
ff	0d	02	04 02 ff ff ff ff ff ff bb	99 a5

otrzymujemy odpowiedź:

adres modułu	dł. ramki	odpowiedź	parametry	kod operacji	CRCH, CRCL
01	16	03	00 00 a0 b0 ff ff 5f 4f 00 00 a0 b0 00 ff 00 ff	ff	da af

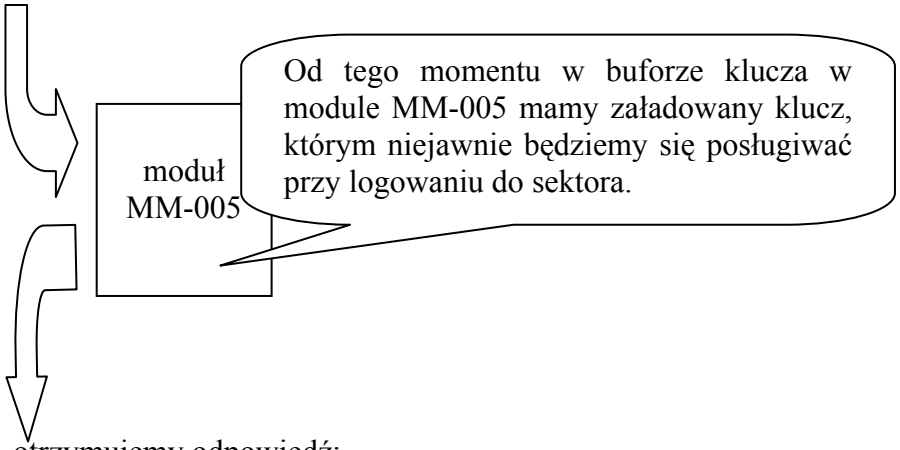
Widzimy, że w bloku nr. 5 mamy zapisaną wartość 00 00 a0 b0 czyli dekrementacja zapisanej w bloku liczby odbyła się prawidłowo.

Podobne działanie można przeprowadzić używając rozkazu C_HL_Increment inkrementującego wartość zapisaną w transponderze.

Przykład 3 Dekrementowanie wartości zapisanej w bloku

Ładujemy klucz do podręcznego bufora klucza przy użyciu C_LoadKeyBuffer.
Do modułu MM-005 wysyłamy sekwencję:

adres modułu	dł. ramki	rozkaz	parametry	CRCH, CRCL
ff	0b	14	ff ff ff ff ff ff	3b f0

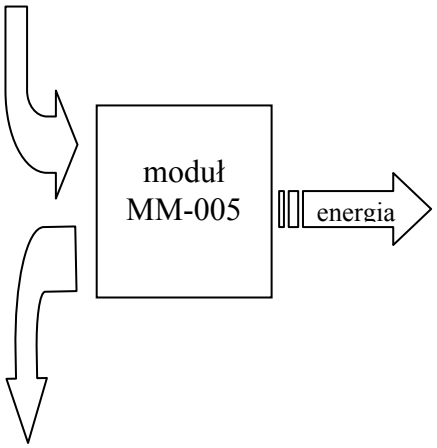


otrzymujemy odpowiedź:

adres modułu	dł. ramki	odpowiedź	parametry	kod operacji	CRCH, CRCL
01	06	15	-	ff	26 62

Za pomocą rozkazu C_TurnOnAntennaPower załączamy pole elektromagnetyczne anteny.

adres modułu	dł. ramki	rozkaz	parametry	CRCH, CRCL
ff	05	10	-	22 A7



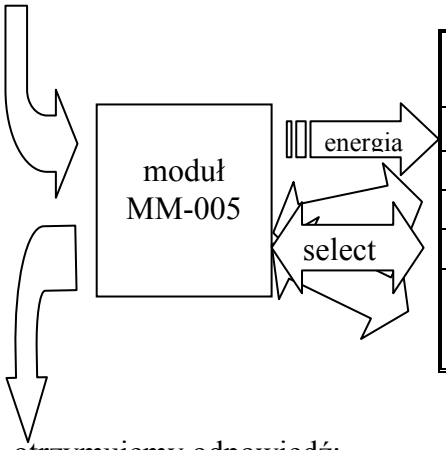
sektor nr.	blok nr.	bloki transpondera z zapisanymi danymi
⋮	⋮	⋮
4	0	
4	1	
4	2	00 00 a0 b0 ff ff 5f 4f 00 00 a0 b0 00 ff 00 ff
4	3	xx xx xx xx xx xx ff 07 80 69 ff ff ff ff ff ff
⋮	⋮	⋮

otrzymujemy odpowiedź:

adres modułu	dł. ramki	odpowiedź	parametry	kod operacji	CRCH, CRCL
01	06	11	-	ff	ea a6

Przy użyciu rozkazu C_Select uruchamiamy pętlę antykolizyjną z wyborem jednego ze znajdujących się w polu transponderów.

adres modułu	dł. ramki	rozkaz	parametry	CRCH, CRCL
ff	06	12	ff	82 e2



sektor nr.	blok nr.	bloki transpondera z zapisanymi danymi
0	0	ID1 ID2 ID3 ID4 xx xx xx xx xx xx xx xx xx xx xx xx
0	1	
0	2	
0	3	
⋮	⋮	⋮

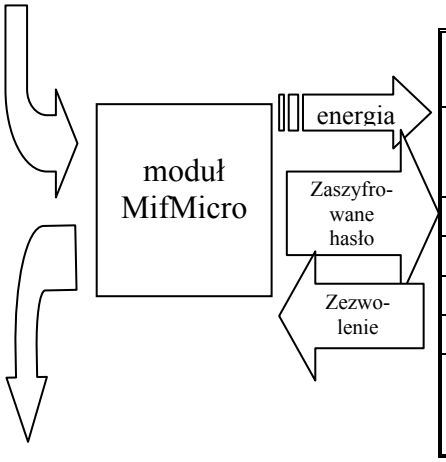
otrzymujemy odpowiedź:

adres modułu	dł. ramki	odpowiedź	parametry	kod operacji	CRCH, CRCL
01	0a	13	ID1 ID2 ID3 ID4	ff	XX XX

Gdzie ID1...ID4 to unikalny numer ID transpondera, który został wyselekcjonowany w polu anteny. Numer ten jest zapisany w sektorze nr.0 transpondera.

Przy użyciu hasła zapamiętanego w podręcznym buforze klucza logujemy się do sektora nr.4. Transponder udostępni wtedy bloki od 0 do 3.
Dzięki parametrowi bb, klucz znajdujący się w podręcznym buforze klucza traktowany jest jak typu B.
Posłużmy się rozkazem LoginWithKB

adres modułu	dł. ramki	rozkaz	parametry	CRCH, CRCL
ff	07	18	04 bb	3b 34



sektor nr.	blok nr.	bloki transpondera z zapisanymi danymi
⋮	⋮	⋮
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**
⋮	⋮	⋮

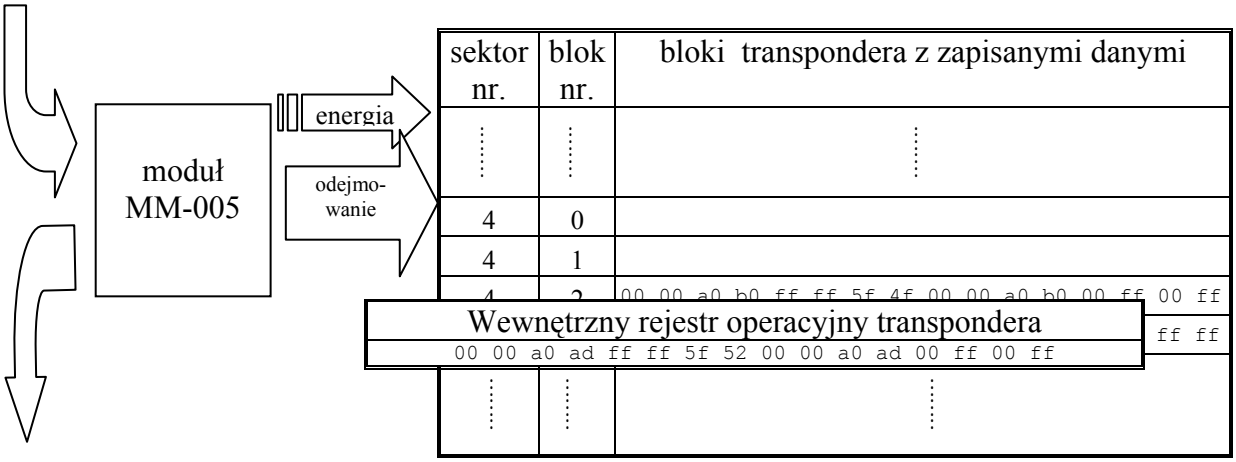
otrzymujemy odpowiedź:

adres modułu	dł. ramki	odpowiedź	parametry	kod operacji	CRCH, CRCL
01	06	19	-	ff	63 0f

Kod operacji ff oznacza poprawne zalogowanie do czwartego sektora.

W bloku nr.2 mamy zapisany ciąg wartości: 00 00 a0 b0 ff ff 5f 4f 00 00 a0 b0 00 ff 00 ff czyli wartość 00 00 a0 b0 od której odejmiemy 3.
Przystąpmy więc do dekrementacji bloku nr.2 za pomocą C_DecrementValue.

adres modułu	dł. ramki	rozkaz	parametry	CRCH, CRCL
ff	0a	32	02 00 00 00 03	b7 2d



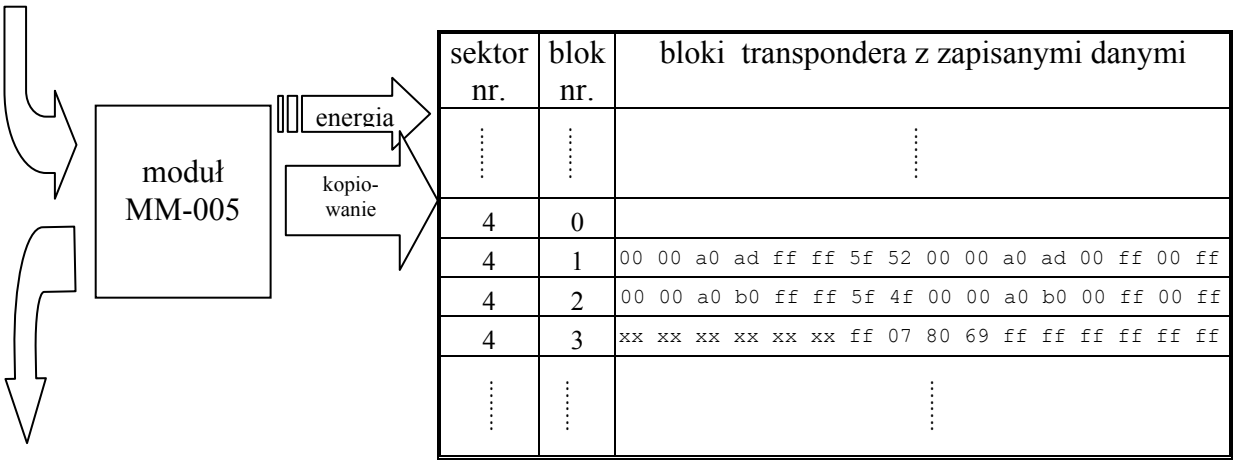
otrzymujemy odpowiedź:

adres modułu	dł. ramki	odpowiedź	parametry	kod operacji	CRCH, CRCL
01	06	33	-	ff	8a 22

Kod operacji ff oznacza poprawną dekrementację wartości. Jest ona jednak przechowywana w wewnętrznym rejestrze operacyjnym transpondera a nie w źródłowym bloku. W zależności od potrzeb, zgodnie z uprawnieniami, użytkownik może przesłać ją do wybranego bloku pamięci w ramach tego samego sektora.

Przepiszmy rejestr operacyjny do bloku nr.1 za pomocą C_TransferValue

adres modułu	dł. ramki	rozkaz	parametry	CRCH, CRCL
ff	0a	38	01	65 1e

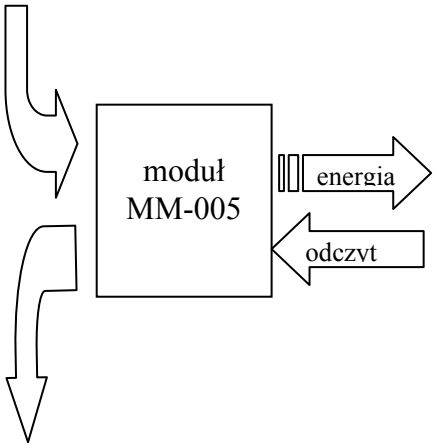


otrzymujemy odpowiedź:

adres modułu	dł. ramki	odpowiedź	parametry	kod operacji	CRCH, CRCL
01	06	39	-	ff	65 e9

Odczytajmy wartość za pomocą C_ReadValue

adres modułu	dł. ramki	rozkaz	parametry	CRCH, CRCL
ff	06	36	01	46 11



sektor nr.	blok nr.	bloki transpondera z zapisanymi danymi
⋮	⋮	⋮
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
⋮	⋮	⋮

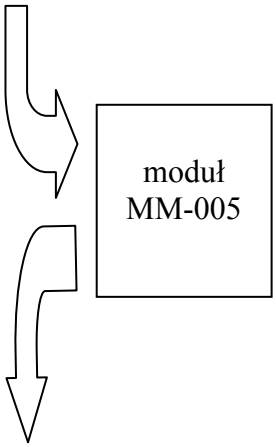
otrzymujemy odpowiedź:

adres modułu	dł. ramki	odpowiedź	parametry	kod operacji	CRCH, CRCL
01	0b	37	00 00 a0 ad 00	ff	76 0e

Odczytana wartość to 00 00 a0 ad czyli wszystkie operacje przebiegły pomyślnie.

Wyłączamy pole anteny używając C_TurnOffAntennaPower

adres modułu	dł. ramki	rozkaz	parametry	CRCH, CRCL
ff	05	44	-	38 d6



sektor nr.	blok nr.	bloki transpondera z zapisanymi danymi
⋮	⋮	⋮
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
⋮	⋮	⋮

otrzymujemy odpowiedź:

adres modułu	dł. ramki	odpowiedź	parametry	kod operacji	CRCH, CRCL
01	0b	45	-	ff	28 dd

Najnowsze wiadomości dotyczące produktów firmy NETRONIX
http://www.netronix.pl/