

FCC / ISED approved RS485 RFID reader

Eccel part: 000470

User manual

Manual version: v1.1¹

14/01/2020

This product has been approved by the Federal Communications Commission (FCC) and Industry Canada (IC). It is accredited Limited Modular Approval for radio modules under regulations detailed in 47 CFR 15.225 and RSS-210 issue 10 Annex B6.

The product is only for use with a 5V dc (+/- 5%) power supply to it via the J1 connector shown in Picture 1 below.

The host power supply to the product should be rated to be able to deliver up to 300 milliamps of in rush current for up to 100 milliseconds and supply a constant current of up to 150 milliamps.

This product is only authorised for operation using the integral trace antenna contained on the product. It is not authorised for use with any other antenna.

RF exposure calculation has been provided with the module certification utilising a worst case 5mm distance to users, RF exposure in the host equipment may need to be re-evaluated.

Host equipment incorporating this product must have clearly affixed to it a permanent label stating, "Contains FCC ID: 2ALHY1356NFC-ICC-IC and Contains IC ID: 22592-RS485IC470".

The host equipment containing this product must itself be tested and verified that it does not exceed the FCC rules in 47 CFR part 15.225 including if the host equipment is powered by a mains AC supply, that it does not exceed the AC powerline conducted emissions limits specified in this FCC test specification. A host manufacturer has three options available, either:

- 1) A Class II permissive change must be requested through Eccel Technology Ltd in order to add the host equipment test evaluation data to the existing certification.
- 2) The host manufacturer can perform a change in ID followed by a class II permissive change.
- 3) The host manufacturer can apply for a new FCC ID.

This product is only authorised for the following rule: Part 47 CFR 15.225 as listed in the Grant of Equipment Authorisation.

The host product manufacturer is responsible for compliance to any other FCC and IC rules that apply to the host not covered by the modular transmitter Grant of Certification.

The RS485 RFID Reader v1 must be connected to the RS485 bus using standard CAT5 (not crossed) LAN cable. Pinout of the RJ45 connector is describe below. Devices can be connected in a chain, so one device can be connected to another one.

¹ The latest user manual can be found on our website: https://eccel.co.uk/wp-content/downloads/FCC_RS485_Reader_User_manual.pdf

Table of contents

1. Introduction	5
2. Electrical specification	6
2.1 ABSOLUTE MAXIMUM RATINGS	6
2.2 OPERATING CONDITIONS	6
2.3 DC CHARACTERISTICS ($V_{DD} = 5\text{ V}$, $T_S = 25\text{ }^\circ\text{C}$)	6
2.4 CURRENT CONSUMPTION (5V INPUT)	6
3. Installation	7
4. Communication interface	8
4.1 OVERVIEW	8
4.2 FRAME STRUCTURE	8
4.3 CRC CALCULATION	9
5. Commands list	11
5.1 GENERIC COMMANDS	11
5.1.1 Acknowledge frame (0x00)	11
5.1.2 Error response (0xFF)	11
5.1.3 Dummy command (0x01)	12
5.1.4 Get tag count (0x02)	13
5.1.5 Get tag UID (0x03)	13
5.1.6 Activate TAG (0x04)	14
5.1.7 Halt (0x05)	15
5.1.8 Set key (0x06)	15
5.1.9 Save keys (0x07)	16
5.1.10 Reboot (0x08)	16
5.1.11 Get version (0x09)	16
5.1.12 Get hardware version (0x0A)	17
5.1.13 Set communication settings (0x0B)	17
5.1.14 Get communication settings (0x0C)	18
5.1.15 Reset to factory defaults (0x0D)	19
5.1.16 Set LED (0x0E)	19
5.2 FIRMWARE COMMANDS	20
5.2.1 Jump to bootloader (0xF1)	20
5.2.2 Firmware start frame (0xF2)	20
5.2.3 Firmware frame (0xF4)	21
5.2.4 Firmware finish frame (0xF4)	21
5.3 MIFARE CLASSICS COMMANDS	22
5.3.1 Read block (0x20)	22
5.3.2 Write block (0x21)	22
5.3.3 Read value (0x22)	23
5.3.4 Write value (0x23)	24
5.3.5 Increment/decrement value (0x24)	24
5.3.6 Transfer value (0x25)	25
5.3.7 Restore value (0x26)	26
5.3.8 Transfer-Restore value (0x27)	26

5.4	MIFARE ULTRALIGHT COMMANDS.....	28
5.4.1	Read page (0x40).....	28
5.4.2	Write page (0x41).....	28
5.4.3	Get version (0x42).....	29
5.4.4	Read signature (0x43).....	29
5.4.5	Write signature (0x44).....	30
5.4.6	Lock signature (0x45).....	30
5.4.7	Read counter (0x46).....	30
5.4.8	Increment counter (0x47).....	31
5.4.9	Password auth (0x48).....	31
5.4.10	Ultralight-C authenticate (0x49).....	32
5.4.11	Check Tearing Event (0x4A).....	32
5.5	MIFARE DESFIRE COMMANDS.....	34
5.5.1	Get version (0x60).....	34
5.5.2	Select application (0x61).....	34
5.5.3	List application IDs (0x62).....	35
5.5.4	List files IDs (0x63).....	35
5.5.5	Authenticate (0x64).....	36
5.5.6	Authenticate ISO (0x65).....	36
5.5.7	Authenticate AES (0x66).....	37
5.5.8	Create application (0x67).....	37
5.5.9	Delete application (0x68).....	38
5.5.10	Change key (0x69).....	38
5.5.11	Get key settings (0x6A).....	39
5.5.12	Change key settings (0x6B).....	39
5.5.13	Create standard or backup data file (0x6C).....	39
5.5.14	Write data (0x6D).....	40
5.5.15	Read data (0x6E).....	40
5.5.16	Create value file (0x6F).....	41
5.5.17	Get value (0x70).....	42
5.5.18	Credit file (0x71).....	42
5.5.19	Credit file (0x72).....	42
5.5.20	Debit file (0x73).....	43
5.5.21	Create record file (0x74).....	43
5.5.22	Write record (0x75).....	44
5.5.23	Read record (0x76).....	44
5.5.24	Clear records (0x77).....	45
5.5.25	Delete file (0x78).....	45
5.5.26	Get free memory (0x79).....	46
5.5.27	Format memory (0x7A).....	46
5.5.28	Commit transaction (0x7B).....	46
5.5.29	Abort transaction (0x7C).....	47
5.6	ICODE (ISO15693) COMMANDS.....	48
5.6.1	Inventory start (0x90).....	48
5.6.2	Inventory next (0x91).....	48
5.6.3	Stay quiet (0x92).....	49
5.6.4	Read block (0x93).....	49
5.6.5	Write block (0x94).....	50
5.6.6	Lock block (0x95).....	50

5.6.7	Write AFI (0x96)	51
5.6.8	Lock AFI (0x97)	51
5.6.9	Write DSFID (0x98)	52
5.6.10	Lock DSFID (0x99)	52
5.6.11	Get System Information (0x9A)	52
5.6.12	Get multiple BSS (0x9B)	53
5.6.13	Password protect AFI (0x9C)	53
5.6.14	Read EPC (0x9D)	54
5.6.15	Get NXP System Information (0x9E)	54
5.6.16	Get random number (0x9F)	55
5.6.17	Set password (0xA0)	55
5.6.18	Write password (0xA1)	56
5.6.19	Lock password (0xA2)	56
5.6.20	Protect page (0xA3)	57
5.6.21	Lock page protection (0xA4)	58
5.6.22	Get multiple block protection status (0xA5)	58
5.6.23	Destroy (0xA6)	59
5.6.24	Enable privacy (0xA7)	59
5.6.25	Enable 64-bit password (0xA8)	59
5.6.26	Read signature (0xA9)	60
5.6.27	Read config (0xAA)	60
5.6.28	Write config (0xAB)	61
5.6.29	Pick random ID (0xAC)	61
6.	Compliance	63
7.	Mechanical dimensions	64

1. Introduction

Features

- Low cost RFID Reader with MIFARE® Classic® in 1K, 4K memory, ICODE, MIFARE Ultralight®, MIFARE DESFire® EV1/EV2, MIFARE Plus® support
- Over-the-Air lifetime updates
- Command interface via RS485
- Baud rate up to 115200 bps
- 1 constant current output (J3)
- High transponder read and write speed
- -25°C to 85°C operating range
- Multiple internal reference voltages
- RoHS compliant
- FCC / ISED limited modular approval



Description

This RS485 FCC/ISED/RED approved reader is CE compliant and has limited modular approval from the FCC and ISED for use in equipment supplied and fitted in the USA and Canada. Eccel provides free lifetime Over-the-Air (OTA) updates via a firmware file download from our website and programming of the unit via our free PC app using the RS485 communication port.

RS485 communications make the device suitable for industrial applications in noisy environments and/or where long communication lines are required. It provides multi-unit configurable addressing and integrated end of line termination impedance selection.

RS485 in and out ports are provided using standard RJ45 connectors. This enables users to connect the boards in a daisy chain arrangement using low cost readily available standard Cat 5 Ethernet cables.

The device provides “out of the box” RFID capability for integrating into customer equipment. A powerful 32-bit processor enables easy host communication to a wide range of NFC transponders with simple host commands and also powerful autonomous stand-alone capability.

Applications

- Access control
- Monitoring goods
- Approval and monitoring consumables
- Pre-payment systems
- Managing resources
- Contact-less data storage systems
- Evaluation and development of RFID systems

2. Electrical specification

2.1 Absolute maximum ratings

Stresses beyond the absolute maximum ratings listed in the table below may cause permanent damage to the device. These are stress ratings only, and do not refer to the functional operation of the device that should follow the recommended operating conditions.

Symbol	Parameter	Min	Max	Unit
T _S	Storage temperature	-40	+125	°C
T _A	Ambient temperature	-40	+85	°C
V _{DDMAX}	Supply voltage	3	5.5	V

Table 2-1. Absolute maximum ratings

2.2 Operating conditions

Symbol	Parameter	Min	Typ	Max	Unit
T _S	Operating temperature	-25	25	+85	°C
H	Humidity	5	60	95	%
V _{DD}	Supply voltage (J1 header)	4.75	5	5.25	V
I _{LED}	External LED current (at 5V)	13	15	17	mA

Table 2-2. Operating conditions

2.3 DC characteristics (V_{DD} = 5 V, T_S = 25 °C)

Symbol	Parameter	Min	Typ	Max	Unit
V _{ORS485}	V output RS485 (J1 header, RS485_TX pin)	-	5	-	V
V _{IRS485}	V input RS485 (J1 header, RS485_RX pin)	-15	-	+15	V

Table 2-3. DC characteristics

2.4 Current consumption (5V input)

Symbol	Parameter	Typ	Max	Unit
I _{PN_RFOFF}	RF field off	20		mA
I _{PN_RFON}	RF field on	60 ²	250 ³	mA

Table 2-4. Current consumption

² Dedicated NTag216 tag may increase current consumption up to 90mA and value depends on distance and location between tag and antenna (higher consumption is when tag is closer to the antenna).

³ Any metal object located in electromagnetic flux will increase current consumption.

3. Installation

The RS485 RFID Reader must be connected to the RS485 bus using standard CAT5 (not crossed) LAN cable. Pinout of the RJ45 connector is describe below. Devices can be connected in a chain, so one device can be connected to another one.

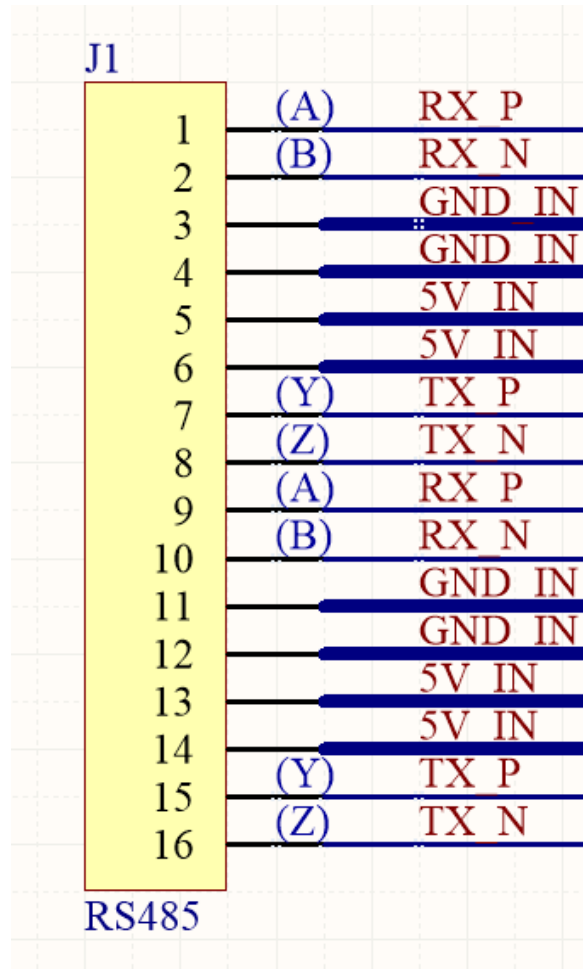


Figure 3-1. RJ45 connector pinout.

There is also the J3 header on the board. The user can connect there an external LED. The header is BM02B-SRSS-TB (JST). The reader delivers 15mA to drive a LED. This header fits the following plug: SHR-02V-S-B (JST) which can be bought here: <https://www.digikey.co.uk/product-detail/en/jst-sales-america-inc/SHR-02V-S-B/455-1377-ND/759866>

4. Communication interface

4.1 Overview

The RS485 RFID Reader can be controlled using a simple binary protocol available over RS485. This binary protocol was designed to be as simple as possible to implement on the host side whilst still providing robust communication.

The default baud rate is 115200 bps, and can be changed from 9600 up to 115200.

Other protocol settings are:

- data bits: 8,
- parity: None,
- Stop bits: 1,
- flow control: None.

4.2 Frame structure

Communication with the module is symmetric. So, frames sent to and received from the module are coded in the same way. All frames contain fields as described in the table below. The reader should answer only to commands with an address byte equal to its own device address.

The length of the command is the sum of the Address byte + Command body + 2 bytes CRC.

Frame STX	Command length	Command length XOR	Address byte	Command body		CRC16
1-byte	2-bytes	2-bytes	1-byte	1-byte	n-bytes	2-bytes
0xF5	Command body length, maximum value 1024, LSB	XOR with 0xFFFF of command length bytes, LSB	Address byte	Command	Command parameters	Address + Command body CRC, LSB

4.3 CRC calculation

CRC is a 16-bit CRC-CCITT with a polynomial equal to 0x1021. The initial value is set to 0xFFFF, the input data and the output CRC is not negated. In addition, no XOR is performed on the output value. Example C code is shown below.

```
static const uint16_t CCITTCRCTable [256] = {  
0x0000, 0x1021, 0x2042, 0x3063, 0x4084, 0x50a5,  
0x60c6, 0x70e7, 0x8108, 0x9129, 0xa14a, 0xb16b,  
0xc18c, 0xd1ad, 0xe1ce, 0xf1ef, 0x1231, 0x0210,  
0x3273, 0x2252, 0x52b5, 0x4294, 0x72f7, 0x62d6,  
0x9339, 0x8318, 0xb37b, 0xa35a, 0xd3bd, 0xc39c,  
0xf3ff, 0xe3de, 0x2462, 0x3443, 0x0420, 0x1401,  
0x64e6, 0x74c7, 0x44a4, 0x5485, 0xa56a, 0xb54b,  
0x8528, 0x9509, 0xe5ee, 0xf5cf, 0xc5ac, 0xd58d,  
0x3653, 0x2672, 0x1611, 0x0630, 0x76d7, 0x66f6,  
0x5695, 0x46b4, 0xb75b, 0xa77a, 0x9719, 0x8738,  
0xf7df, 0xe7fe, 0xd79d, 0xc7bc, 0x48c4, 0x58e5,  
0x6886, 0x78a7, 0x0840, 0x1861, 0x2802, 0x3823,  
0xc9cc, 0xd9ed, 0xe98e, 0xf9af, 0x8948, 0x9969,  
0xa90a, 0xb92b, 0x5af5, 0x4ad4, 0x7ab7, 0x6a96,  
0x1a71, 0x0a50, 0x3a33, 0x2a12, 0xdbfd, 0xcbdc,  
0xfbbf, 0xeb9e, 0x9b79, 0x8b58, 0xbb3b, 0xab1a,  
0x6ca6, 0x7c87, 0x4ce4, 0x5cc5, 0x2c22, 0x3c03,  
0x0c60, 0x1c41, 0xedae, 0xfd8f, 0xcdec, 0xddcd,  
0xad2a, 0xbd0b, 0x8d68, 0x9d49, 0x7e97, 0x6eb6,  
0x5ed5, 0x4ef4, 0x3e13, 0x2e32, 0x1e51, 0x0e70,  
0xff9f, 0xefbe, 0xdfdd, 0xcffc, 0xbf1b, 0xaf3a,  
0x9f59, 0x8f78, 0x9188, 0x81a9, 0xb1ca, 0xa1eb,  
0xd10c, 0xc12d, 0xf14e, 0xe16f, 0x1080, 0x00a1,  
0x30c2, 0x20e3, 0x5004, 0x4025, 0x7046, 0x6067,  
0x83b9, 0x9398, 0xa3fb, 0xb3da, 0xc33d, 0xd31c,  
0xe37f, 0xf35e, 0x02b1, 0x1290, 0x22f3, 0x32d2,  
0x4235, 0x5214, 0x6277, 0x7256, 0xb5ea, 0xa5cb,  
0x95a8, 0x8589, 0xf56e, 0xe54f, 0xd52c, 0xc50d,  
0x34e2, 0x24c3, 0x14a0, 0x0481, 0x7466, 0x6447,  
0x5424, 0x4405, 0xa7db, 0xb7fa, 0x8799, 0x97b8,  
0xe75f, 0xf77e, 0xc71d, 0xd73c, 0x26d3, 0x36f2,
```

```
0x0691, 0x16b0, 0x6657, 0x7676, 0x4615, 0x5634,  
0xd94c, 0xc96d, 0xf90e, 0xe92f, 0x99c8, 0x89e9,  
0xb98a, 0xa9ab, 0x5844, 0x4865, 0x7806, 0x6827,  
0x18c0, 0x08e1, 0x3882, 0x28a3, 0xcb7d, 0xdb5c,  
0xeb3f, 0xfb1e, 0x8bf9, 0x9bd8, 0xabbb, 0xbb9a,  
0x4a75, 0x5a54, 0x6a37, 0x7a16, 0x0af1, 0x1ad0,  
0x2ab3, 0x3a92, 0xfd2e, 0xed0f, 0xdd6c, 0xcd4d,  
0xbdaa, 0xad8b, 0x9de8, 0x8dc9, 0x7c26, 0x6c07,  
0x5c64, 0x4c45, 0x3ca2, 0x2c83, 0x1ce0, 0x0cc1,  
0xef1f, 0xff3e, 0xcf5d, 0xdf7c, 0xaf9b, 0xbfba,  
0x8fd9, 0x9ff8, 0x6e17, 0x7e36, 0x4e55, 0x5e74,  
0x2e93, 0x3eb2, 0x0ed1, 0x1ef0 };
```

```
static uint16_t GetCCITTCRC(const uint8_t* Data, uint32_t Size) {  
    uint16_t CRC;  
    uint16_t Temp;  
    uint32_t Index;  
    if (Size == 0) {  
        return 0;  
    }  
    CRC = 0xFFFF;  
    for (Index = 0; Index < Size; Index++){  
        Temp = (uint16_t)( (CRC >> 8) ^ Data[Index] ) & 0x00FF;  
        CRC = CCITTCRCtable[Temp] ^ (CRC << 8);  
    }  
    return CRC;  
}
```

5. Commands list

Commands are exchanged with the module using the protocol described above. All frames contain a command byte and command arguments. Depending upon the command, arguments can be optional, so a command length can be in the range from 1-1024 bytes.

5.1 Generic commands

5.1.1 Acknowledge frame (0x00)

This is the response message from the module to the host. This frame always contains 1-byte with command ID and optional arguments.

Command description			
Argument	Size	Value	Description
Command ID	1	0x00	
Related command ID	1	X	Related command code
Other parameters	n	X	Depending on the requested command this parameter is n-bytes long and contains parameters

Example:

```

HOST=>READER: 0x02 - GET_TAG_COUNT command
READER=>HOST: 0x00 - ACK byte
                0x02 - related command code GET_TAG_COUNT
                0x01 - argument for GET_TAG_COUNT - 0x01 - one tag
                    detected
    
```

5.1.2 Error response (0xFF)

In case of any problems with executing the command, the device can send back ERROR response with error number returned by the RFID chip. The most common errors are described below.

If the host sends an Error frame without argument to the reader, then repeat last frame.

Command description			
Argument	Size	Value	Description
ACK	1	0x00	
Command ID (optional)	1	0x01	DUMMY_COMMAND

Example:

```

HOST=>READER: 0xFF - Error byte
READER=>HOST: XX XX XX ... - last command send to the host
    
```

Example:

```

READER=>HOST: 0xFF - Error byte
                0x01 - related command code DUMMY_COMMAND
                0x02 - layer byte
                0x01 - Error number
    
```

Error list:

- 0x01 - No reply received, e.g. PICC removal
- 0x02 - Wrong CRC or parity detected
- 0x03 - A collision occurred
- 0x04 - Attempt to write beyond buffer size
- 0x05 - Invalid frame format
- 0x06 - Received response violates protocol
- 0x07 - Authentication error
- 0x08 - A Read or Write error occurred in RAM/ROM or Flash
- 0x09 - The RC sensors signal over heating
- 0x0A - Error due to RF.
- 0x0B - An error occurred in RC communication
- 0x0C - A length error occurred
- 0x0D - An resource error
- 0x0E - TX Rejected sanely by the counterpart
- 0x0F - RX request Rejected sanely by the counterpart
- 0x10 - Error due to External RF
- 0x11 - EMVCo EMD Noise Error
- 0x12 - Used when HAL ShutDown is called
- 0x20 - Invalid data parameters supplied (layer id check failed)
- 0x21 - Invalid parameter supplied
- 0x22 - Reading/Writing a parameter would produce an overflow
- 0x23 - Parameter not supported
- 0x24 - Command not supported
- 0x25 - Condition of use not satisfied
- 0x26 - key error occurred
- 0x7F - An internal error occurred

5.1.3 Dummy command (0x01)

This command takes no arguments. It is used to check that the module alive. The module replies to this command with an ACK response and no optional parameters.

Command description			
Argument	Size	Value	Description
Command ID	1	0x01	DUMMY_COMMAND
Response description			
ACK	1	0x00	
Command ID	1	0x01	DUMMY_COMMAND

Example:

```

HOST=>READER: 0x01 -DUMMY_COMMAND
READER=>HOST: 0x00 - ACK byte
              0x01 - related command code DUMMY_COMMAND
    
```

5.1.4 Get tag count (0x02)

The command send to the module to read how many TAGS are in range of the antenna no matter which technology of tag, so it returns the total amount present of all supported tag types. The maximum number for this standard discovery loop is 5. If you want to perform a full inventory command for ICODE tag types please refer to ICODE_INVENTORY_xxx commands.

After this command, the module holds all UID's and basic information about TAGs present in volatile memory and the user can read it using the GET_TAG_UID command.

Command description			
Argument	Size	Value	Description
Command ID	1	0x02	GET_TAG_COUNT
Response description			
ACK	1	0x00	
Command ID	1	0x02	GET_TAG_COUNT
TAG count	1	X	Maximum discovered tags is 5

Example:

```

HOST=>READER: 0x02 - GET_TAG_COUNT
READER=>HOST: 0x00 - ACK byte
                0x02 - related command code GET_TAG_COUNT
                0x01 - number of tags in range
    
```

5.1.5 Get tag UID (0x03)

This command should be executed after GET_TAG_COUNT frame to read information about the tag.

Command description			
Argument	Size	Value	Description
Command ID	1	0x03	GET_TAG_UID
TAG idx	1	X	TAG index in module memory, must me less than number of tags reported by GET_TAG_COUNT command
Response description			
ACK	1	0x00	
Command ID	1	0x03	GET_TAG_UID
TAG type	1	X	0x01 - MIFARE Ultralight 0x02 - MIFARE Ultralight-C 0x03 - MIFARE Classic 0x04 - MIFARE Classic 1k 0x05 - MIFARE Classic 4k 0x06 - MIFARE Plus 0x07 - MIFARE Plus 2k 0x08 - MIFARE Plus 4k 0x09 - MIFARE Plus 2k s12 0x0A - MIFARE Plus 4k s12 0x0B - MIFARE Plus 2k s13 0x0C - MIFARE Plus 4k s13 0x0D - MIFARE DESFire

			0x0F - JCOP 0x10 – MIFARE Mini 0x21 – ICODE Sli 0x22 – ICODE Sli-S 0x23 – ICODE Sli-L 0x24 – ICODE Slix 0x25 – ICODE Slix-S 0x26 – ICODE Slix-X 0x27 – ICODE Slix2 0x28 – ICODE DNA
TAG parameter	1	X	SAK - byte for MIFARE family tags DSFID - byte for ICODE family tags
UID	N	X	UID bytes. Max length is 8.

Example:

```

HOST=>READER: 0x03 - GET_TAG_UID
               0x00 - TAG idx

READER=>HOST: 0x00 - ACK byte
               0x03 - related command code GET_TAG_UID
               0x01 - MIFARE tag type
               0x20 - tag parameter:
                     SAK byte for MIFARE family tags
                     DSFID byte for ICODE family tags
               0x74 0x54 0x12 0x65 - tag UID bytes
  
```

5.1.6 Activate TAG (0x04)

The command activates a TAG after the discovery loop if more than one TAG is detected.

Command description			
Argument	Size	Value	Description
Command ID	1	0x04	ACTIVATE_TAG
TAG idx	1	X	TAG index in module memory, must be less than number of tags reported by GET_TAG_COUNT command
Response description			
ACK	1	0x00	
Command ID	1	0x04	ACTIVATE_TAG

Example:

```

HOST=>READER: 0x04 - ACTIVATE_TAG
               0x00 - TAG idx

READER=>HOST: 0x00 - ACK byte
               0x04 - related command code ACTIVATE_TAG
  
```

5.1.7 Halt (0x05)

The Halt command takes no arguments. It halts the tag and turns off the RF field. It must be executed at the end of each operation on a tag to disable the antenna and reduce the power consumption.

Command description			
Argument	Size	Value	Description
Command ID	1	0x05	HALT
Response description			
ACK	1	0x00	
Command ID	1	0x05	HALT

Example:

```

HOST=>READER: 0x05 - HALT
READER=>HOST: 0x00 - ACK byte
                0x05 - related command code HALT

```

5.1.8 Set key (0x06)

This command sets a KEY in Key Storage Memory on a selected slot. Set key can be used for all RFID functions needing authorization like e.g. READ/WRITE memory on the TAG etc. This command changes a key in volatile memory, so if the user wants to save it permanently and load automatically after boot-up, then the user should use the CMD_SAVE_KEYS command.

Command description			
Argument	Size	Value	Description
Command ID	1	0x06	SET_KEY
Key number	1	0-4	Key number in Key Storage Memory.
Key type	1	0 – 6	0x00 - AES 128 Key. (length = 16 bytes) 0x01 - AES 192 Key. (length = 24 bytes) 0x02 - AES 256 Key. (length = 32 bytes) 0x03 - DES Single Key. (length = 16 bytes) 0x04 - 2 Key Triple Des. (length = 16 bytes) 0x05 - 3 Key Triple Des. (length = 24 bytes) 0x06 - MIFARE (R) Key. (length = 12 bytes, key A+B)
Key	12-32	X	Key bytes. Length must match to the type.
Response description			
ACK	1	0x00	
Command ID	1	0x06	SET_KEY

Example:

```

HOST=>READER: 0x06 - SET_KEY
                0x00 - Key number
                0x06 - MIFARE key type
                0x00 0x00 0x00 0x00 0x00 0x00
                0xFF 0xFF 0xFF 0xFF 0xFF 0xFF - Key bytes

READER=>HOST: 0x00 - ACK byte
                0x06 - related command code SET_KEY

```

5.1.9 Save keys (0x07)

This command should be called if the user wants to save keys changed using the SET_KEY command in the module non-volatile memory. Saved keys will be automatically loaded after power up or reboot.

Command description			
Argument	Size	Value	Description
Command ID	1	0x07	SAVE_KEYS
Response description			
ACK	1	0x00	
Command ID	1	0x07	SAVE_KEYS

Example:

```
HOST=>READER: 0x07 - SAVE_KEYS
READER=>HOST: 0x00 - ACK byte
              0x07 - related command code SAVE_KEYS
```

5.1.10 Reboot (0x08)

This command requests a software reboot for the reader. After this command the device will not accept any protocol commands for 1 second.

Command description			
Argument	Size	Value	Description
Command ID	1	0x08	REBOOT
Response description			
ACK	1	0x00	
Command ID	1	0x08	REBOOT

Example:

```
HOST=>READER: 0x08 - REBOOT
READER=>HOST: 0x00 - ACK byte
              0x08 - related command code REBOOT
```

5.1.11 Get version (0x09)

This command requests a version string from the device.

Command description			
Argument	Size	Value	Description
Command ID	1	0x09	GET_VERSION
Response description			
ACK	1	0x00	
Command ID	1	0x09	GET_VERSION
Version string	X	X	Version string, contains major and minor version and build data and time e.g.: 1.1 Jan 18 2019 15:35:03

Example:

```

HOST=>READER: 0x09 - GET_VERSION
READER=>HOST: 0x00 - ACK byte
                0x09 - related command code GET_VERSION
                0x31 0x2e 0x31 0x20 0x4a 0x61 0x6e 0x20
                0x31 0x38 0x20 0x32 0x30 0x31 0x39 0x20
                0x31 0x35 0x3a 0x33 0x35 0x3a 0x30 0x33 - version
                string bytes
    
```

5.1.12 Get hardware version (0x0A)

This command requests a hardware version string from the device.

Command description			
Argument	Size	Value	Description
Command ID	1	0x0A	GET_HW_VERSION
Response description			
ACK	1	0x00	
Command ID	1	0x0A	GET_HW_VERSION
Version string	X	X	Version string, contains product name and major and minor hardware revision e.g.: RS485_v1.0

Example:

```

HOST=>READER: 0x0A - GET_HW_VERSION
READER=>HOST: 0x00 - ACK byte
                0x0A - related command code GET_HW_VERSION
                0x42 0x69 0x6f 0x52 0x61 0x64 0x5f 0x52 0x53 0x34
                0x35 0x5f 0x76 0x31 0x2e 0x30 - version string bytes
    
```

5.1.13 Set communication settings (0x0B)

This command sets the new communication settings of the RS485 bus. The reader sends an ACK using the old settings, then sets the new RS485 communication parameters. The command also includes information about bus termination. If the termination byte is set to 0x01, the reader must terminate the bus. The host application should wait at least 500ms before sending another command. The new RS485 module own address is stored in NVM.

Command description			
Argument	Size	Value	Description
Command ID	1	0x0B	SET_COMM_SETTINGS
Baudrate ID	1	X	0x00 - 4800kbps 0x01 - 9600kbps 0x02 - 19200 kbps 0x03 - 38400 kbps 0x04 - 57600 kbps 0x05 - 115200 kbps (factory default)

Bus address	1	X	Bus address byte. Default value is 0x80
Termination	1	X	0x00 – termination disabled 0x01 – termination enabled
Short name	4	X	Four ASCII bytes with device description
Response description			
ACK	1	0x00	
Command ID	1	0x0B	SET_COMM_SETTINGS

Example:

```

HOST=>READER: 0x0B - SET_COMM_SETTINGS
                0x05 - baudrate 115200
                0x81 - new RS485 module own address
                0x01 - termination
                0x31 0x32 0x33 -x34 - short name

READER=>HOST: 0x00 - ACK byte
                0x0B - related command code SET_COMM_SETTINGS

```

5.1.14 Get communication settings (0x0C)

This command gets communication settings.

Command description			
Argument	Size	Value	Description
Command ID	1	0x0C	GET_COMM_SETTINGS
Response description			
ACK	1	0x00	
Command ID	1	0x0C	GET_COMM_SETTINGS
Baudrate ID	1	X	0x00 – 4800kbps 0x01 – 9600kbps 0x02 - 19200 kbps 0x03 - 38400 kbps 0x04 - 57600 kbps 0x05 - 115200 kbps (factory default)
Bus address	1	X	Bus address byte. Default value is 0x80
Termination	1	X	0x00 – termination disabled 0x01 – termination enabled
Short name	4	X	Four ASCII bytes with device description

Example:

```

HOST=>READER: 0x0C - GET_COMM_SETTINGS

READER=>HOST: 0x00 - ACK byte
                0x0C - related command code GET_COMM_SETTINGS
                0x05 - baudrate 115200
                0x81 - bus address 0x81
                0x01 - termination enabled
                0x31 0x32 0x33 -x34 - short name

```

5.1.15 Reset to factory defaults (0x0D)

This command resets settings to factory default. The reader sends an ACK using the old settings, then reboots. The host application should wait at least 1000ms before sending another command.

Command description			
Argument	Size	Value	Description
Command ID	1	0x0D	FACTORY_RESET
Response description			
ACK	1	0x00	
Command ID	1	0x0D	FACTORY_RESET

Example:

```
HOST=>READER: 0x0D - FACTORY_RESET
READER=>HOST: 0x00 - ACK byte
                0x0D - related command code FACTORY_RESET
```

5.1.16 Set LED (0x0E)

This command turns on/off the external LED. If the On/Off flag is set to 2, then the host must provide a timeout for the LED (two bytes LSB). The device turns on the LED for the specified timeout and then turns it off.

Command description			
Argument	Size	Value	Description
Command ID	1	0x0E	SET_LED
On/Off flag	1	X	2 – turn on for specified timeout 1 – turn on LED 0 – turn off LED
On timeout	2	X	Unsigned 16-bit timeout value (LSB first) in milliseconds
Response description			
ACK	1	0x00	
Command ID	1	0x0E	SET_LED

Example:

```
HOST=>READER: 0x0E - SET_LED
                0x01 - LED on
READER=>HOST: 0x00 - ACK byte
                0x0E - related command code SET_LED
```

5.2 Firmware commands

The reader supports firmware upgrades. To perform a firmware upgrade, the user must execute the following commands.

5.2.1 Jump to bootloader (0xF1)

This command should be executed as the first frame in a firmware upgrade sequence. When the device receives this command, it reboots and stays in the bootloader application. After this step, the device will not be able to boot the main application until it is uploaded again to the device. The host application can execute the next command after 500ms. It is good practise to send the GET_VERSION command after this one to verify that the device is already in bootloader mode. The GET_VERSION string frame for the bootloader application contains the standard version string with an extra string "BOOTLOADER".

Command description			
Argument	Size	Value	Description
Command ID	1	0xF1	JUMP_TO_BOOTLOADER
Response description			
ACK	1	0x00	
Command ID	1	0xF1	JUMP_TO_BOOTLOADER

Example:

```
HOST=>READER: 0xF1 - JUMP_TO_BOOTLOADER
READER=>HOST: 0x00 - ACK byte
              0xF1 - related command code JUMP_TO_BOOTLOADER
```

5.2.2 Firmware start frame (0xF2)

This command is only supported when the device is running the bootloader application. This command must be executed to clear device flash before writing the new application firmware. The device responds with an ACK frame when the command is finished, and usually it takes about 3seconds.

Command description			
Argument	Size	Value	Description
Command ID	1	0xF2	FIRMWARE_START
Response description			
ACK	1	0x00	
Command ID	1	0xF2	FIRMWARE_START

Example:

```
HOST=>READER: 0xF2 - FIRMWARE_START
READER=>HOST: 0x00 - ACK byte
              0xF2 - related command code FIRMWARE_START
```

5.2.3 Firmware frame (0xF4)

When the device is running the bootloader application and FIRMWARE START has already been executed, the host application can upload binary firmware file in chunks that are 256 bytes long (the last frame can be smaller).

Command description			
Argument	Size	Value	Description
Command ID	1	0xF3	FIRMWARE_FRAME
Firmware bytes	Max. 256		Firmware bytes in chunks 256bytes long.
Response description			
ACK	1	0x00	
Command ID	1	0xF3	FIRMWARE_FRAME

Example:

```

HOST=>READER: 0xF3 - FIRMWARE_FRAME
               0x34 0x67 ... 0x45 - firmware bytes
READER=>HOST: 0x00 - ACK byte
               0xF3 - related command code FIRMWARE_START
  
```

5.2.4 Firmware finish frame (0xF4)

This command is only supported when the device is running the bootloader application. The command must be executed after all firmware frames are written to the device. The bootloader application checks the integrity of the application and runs it. The host application can execute the next command after 500ms. It is good practise to send the GET_VERSION command after this one to verify that the new version is already running on the device.

Command description			
Argument	Size	Value	Description
Command ID	1	0xF4	FIRMWARE_FINISH
Response description			
ACK	1	0x00	
Command ID	1	0xF4	FIRMWARE_FINISH

Example:

```

HOST=>READER: 0xF4 - FIRMWARE_FINISH
READER=>HOST: 0x00 - ACK byte
               0xF4 - related command code FIRMWARE_FINISH
  
```

5.3 MIFARE Classics commands

This set of commands should be performed on MIFARE Classics tags.

5.3.1 Read block (0x20)

The read block command should be used to read data from the tag. It takes as arguments the block number of the first block to read, the number of blocks to read, the key A or B parameter, and the key number in key storage. The returned ACK answer contains data read from the specified tag memory. The number of bytes of this data is MIFARE Classic block size (16) multiplied by the number of blocks to be read.

Command description			
Argument	Size	Value	Description
Command ID	1	0x20	MF_READ_BLOCK
Block number	1	X	
Number of blocks	1	Y	
Key A/B parameter	1	X	0x0A – Key A should be selected from key storage 0x0B – Key B should be selected from key storage
Key number	1	0-4	Key number in key storage
Response description			
ACK	1	0x00	
Command ID	1	0x20	MF_READ_BLOCK
Read data	Y*16	XXX	Bytes read from the tag. Number of bytes is number of requested blocks multiplied by 16.

Example:

```

HOST=>READER: 0x20 - MF_READ_BLOCK
               0x02 - block number 2
               0x02 - two blocks to read
               0x0A - key A should be selected from key storage
               0x00 - first key should be selected from key storage

READER=>HOST: 0x00 - ACK byte
               0x20 - related command code MF_READ_BLOCK
               0x01 0x2e 0x41 0x22 0x43 0x11 0x8e 0x20
               0x31 0x38 0x20 0x32 0x30 0x31 0x39 0x41
               0x81 0x23 0x42 0x28 0x33 0x01 0x8e 0x72
               0x31 0x35 0x3a 0x33 0x35 0x3a 0x30 0x33 - 32 bytes
               result
  
```

5.3.2 Write block (0x21)

The write block command should be used to write data to the tag. It takes as arguments the block number of the first block to write, the number of blocks to write, the key A or B parameter, the key number in key storage, and the bytes to be written. The number of bytes to be written must be exactly the number of blocks to write multiplied by 16. If the host receives an ACK without any errors means the write process was successful and the data was read back and verified as correct by the reader.

Command description			
Argument	Size	Value	Description
Command ID	1	0x21	MF_WRITE_BLOCK
Block number	1	X	
Number of blocks	1	Y	
Key A/B parameter	1	X	0x0A – Key A should be selected from key storage 0x0B – Key B should be selected from key storage
Key number	1	0-4	Key number in key storage
Bytes to write	Y*16	XXX	Bytes to write. Number of this bytes must be number of requested blocks multiplied by 16.
Response description			
ACK	1	0x00	
Command ID	1	0x21	MF_WRITE_BLOCK

Example:

```

HOST=>READER:  0x21 – MF_WRITE_BLOCK
                0x02 – block number 2
                0x02 – two blocks to write
                0x0A – key A should be selected from key storage
                0x00 – first key should be selected from key storage

                0x01 0x2e 0x41 0x22 0x43 0x11 0x8e 0x20
                0x31 0x38 0x20 0x32 0x30 0x31 0x39 0x41
                0x81 0x23 0x42 0x28 0x33 0x01 0x8e 0x72
                0x31 0x35 0x3a 0x33 0x35 0x3a 0x30 0x33 – 32 bytes to
                write

READER=>HOST:  0x00 – ACK byte
                0x21 – related command code MF_WRITE_BLOCK
  
```

5.3.3 Read value (0x22)

This command should be used to read a value from the tag. It takes as arguments the block number where the value is stored, the key A or B parameter, and the key number in key storage. The returned ACK response contains a value as a signed 32-bit value (LSB first) and an address byte as an unsigned 8bit value.

Command description			
Argument	Size	Value	Description
Command ID	1	0x22	MF_READ_VALUE
Block number	1	X	
Key A/B parameter	1	X	0x0A – Key A should be selected from key storage 0x0B – Key B should be selected from key storage
Key number	1	0-4	Key number in key storage
Response description			
ACK	1	0x00	
Command ID	1	0x22	MF_READ_VALUE
Value	4	X	Signed 32-bit value (LSB first)
Address	1	X	Address byte

Example:

```

HOST=>READER: 0x22 - MF_READ_VALUE
                0x02 - block number 2
                0x0A - key A should be selected from key storage
                0x00 - first key should be selected from key storage

READER=>HOST: 0x00 - ACK byte
                0x22 - related command code MF_READ_BLOCK
                0x00 0x00 0x00 0x01 - value
                0x01 - address byte
    
```

5.3.4 Write value (0x23)

This command should be used to write a value to the tag. It takes as arguments the block number where the value should be stored, the key A or B parameter, the key number in key storage, a value (signed 32-bit LSB first) as 4 bytes, and an address byte (unsigned 8-bit value).

Command description			
Argument	Size	Value	Description
Command ID	1	0x23	MF_WRITE_VALUE
Block number	1	X	
Key A/B parameter	1	X	0x0A – Key A should be selected from key storage 0x0B – Key B should be selected from key storage
Key number	1	0-4	Key number in key storage
Value	4	X	Signed 32-bit value (LSB first)
Address	1	X	Address byte
Response description			
ACK	1	0x00	
Command ID	1	0x23	MF_WRITE_VALUE

Example:

```

HOST=>READER: 0x23 - MF_WRITE_VALUE
                0x02 - block number 2
                0x0A - key A should be selected from key storage
                0x00 - first key should be selected from key storage
                0x00 0x00 0x00 0x01 - value
                0x01 - address byte

READER=>HOST: 0x00 - ACK byte
                0x23 - related command code MF_WRITE_BLOCK
    
```

5.3.5 Increment/decrement value (0x24)

This command should be used to increment or decrement a value stored in the tag memory. It takes as arguments the block number where the value is stored, the key A or B parameter, the key number in key storage, value (signed 32-bit LSB first) as 4 bytes to increment or decrement, and the increment/decrement flag.

Command description			
Argument	Size	Value	Description
Command ID	1	0x24	MF_INCREMENT_VALUE
Block number	1	X	
Key A/B parameter	1	X	0x0A – Key A should be selected from key storage 0x0B – Key B should be selected from key storage
Key number	1	0-4	Key number in key storage
Delta value	4	X	Signed 32-bit value (LSB first)
Increment/Decrement	1	X	0x00 – Decrement by delta value 0x01 – Increment by delta value
Response description			
ACK	1	0x00	
Command ID	1	0x24	MF_INCREMENT_VALUE

Example:

```

HOST=>READER: 0x24 – MF_INCREMENT_VALUE
                0x02 – block number 2
                0x0A – key A should be selected from key storage
                0x00 – first key should be selected from key storage
                0x00 0x00 0x00 0x01 – delta value
                0x01 – increment flag

READER=>HOST: 0x00 – ACK byte
                0x24 – related command code MF_INCREMENT_BLOCK
    
```

5.3.6 Transfer value (0x25)

This command should be used to transfer a value from a volatile register on the tag to the block being addressed. It takes as arguments the block number where the value should be stored, the key A or B parameter, the key number in key storage.

Command description			
Argument	Size	Value	Description
Command ID	1	0x25	MF_TRANSFER_VALUE
Block number	1	X	
Key A/B parameter	1	X	0x0A – Key A should be selected from key storage 0x0B – Key B should be selected from key storage
Key number	1	0-4	Key number in key storage
Response description			
ACK	1	0x00	
Command ID	1	0x25	MF_TRANSFER_VALUE

Example:

```

HOST=>READER: 0x25 – MF_TRANSFER_VALUE
                0x02 – block number 2
                0x0A – key A should be selected from key storage
                0x00 – first key should be selected from key storage

READER=>HOST: 0x00 – ACK byte
                0x25 – related command code MF_TRANSFER_BLOCK
    
```

5.3.7 Restore value (0x26)

This command should be used to restore a value to a volatile register on the tag from the block being addressed. It takes as arguments the block number where the value is stored, the key A or B parameter, key number in key storage.

Command description			
Argument	Size	Value	Description
Command ID	1	0x26	MF_RESTORE_VALUE
Block number	1	X	
Key A/B parameter	1	X	0x0A – Key A should be selected from key storage 0x0B – Key B should be selected from key storage
Key number	1	0-4	Key number in key storage
Response description			
ACK	1	0x00	
Command ID	1	0x26	MF_RESTORE_VALUE

Example:

```
HOST=>READER: 0x26 – MF_RESTORE_VALUE
                0x02 – block number 2
                0x0A – key A should be selected from key storage
                0x00 – first key should be selected from key storage
```

```
READER=>HOST: 0x00 – ACK byte
                0x26 – related command code MF_RESTORE_BLOCK
```

5.3.8 Transfer-Restore value (0x27)

This command performs a Restore-Transfer command sequence on the tag. It takes as arguments the block number to be decremented, the block number to be transferred to, the key A or B parameter, the key number in key storage. This command has the same functionality as the read value command, except that it can be used on a block which is corrupted – it tries to recover data from a corrupted block. The format of a value-type block allows for some bits to be corrupted and it still be possible to read and recover the proper value

Command description			
Argument	Size	Value	Description
Command ID	1	0x27	MF_TRANSFER_RESTORE_VALUE
Source block number	1	X	Block number to be decremented
Destination block number	1	X	Block number to be transferred to
Key A/B parameter	1	X	0x0A – Key A should be selected from key storage 0x0B – Key B should be selected from key storage
Key number	1	0-4	Key number in key storage
Response description			
ACK	1	0x00	
Command ID	1	0x27	MF_TRANSFER_RESTORE_VALUE

Example:

HOST=>READER: 0x27 - MF_TRANSFER_RESTORE_VALUE
0x02 - source block number 2
0x03 - destination block number 3
0x0A - key A should be selected from key storage
0x00 - first key should be selected from key storage

READER=>HOST: 0x00 - ACK byte
0x27 - related command code MF_TRANSFER_RESTORE_BLOCK

5.4 MIFARE Ultralight commands

This set of commands should be performed on MIFARE Ultralight tags.

5.4.1 Read page (0x40)

The read page command should be used to read data stored in tag pages. It takes as arguments the page number of the first page to be read, and the number of pages to be read. The returned ACK answer contains data read from the specified tag memory. The number of bytes of this data is MIFARE Ultralight page size (4) multiplied by the number of pages to be read.

Command description			
Argument	Size	Value	Description
Command ID	1	0x40	MFU_READ_PAGE
Page number	1	X	
Number of pages	1	Y	
Response description			
ACK	1	0x00	
Command ID	1	0x40	MFU_READ_PAGE
Read data	Y*4	XXX	Bytes read from the tag. Number of bytes is number of requested pages multiplied by 4.

Example:

```

HOST=>READER: 0x40 - MFU_READ_PAGE
               0x02 - page number 2
               0x02 - two pages to read

READER=>HOST: 0x00 - ACK byte
               0x40 - related command code MFU_READ_PAGE
               0x31 0x35 0x3a 0x33 0x35 0x3a 0x30 0x33 - 8 bytes result
  
```

5.4.2 Write page (0x41)

The write page command should be used to write data to the tag. It takes as arguments the page number of the first page to write, the number of pages to write, and the bytes to be written. The number of bytes to be written must be exactly the number of pages to write multiplied by 4. If the host receives an ACK without any errors means the write process was successful and the data was read back and verified as correct by the reader.

Command description			
Argument	Size	Value	Description
Command ID	1	0x41	MFU_WRITE_PAGE
Page number	1	X	
Number of pages	1	Y	
Bytes to write	Y*4	XXX	Bytes to write. Number of this bytes must be number of requested pages multiplied by 4.
Response description			
ACK	1	0x00	
Command ID	1	0x41	MFU_WRITE_PAGE

Example:

HOST=>READER: 0x41 – MFU_WRITE_PAGE
 0x02 – page number 2
 0x02 – two pages to write
 0x31 0x35 0x3a 0x33 0x35 0x3a 0x30 0x33 – 32 bytes to write

READER=>HOST: 0x00 – ACK byte
 0x41 – related command code MFU_WRITE_PAGE

5.4.3 Get version (0x42)

This command requests a version string from the TAG. The returned ACK answer consists of 8-bytes containing the version information defined by the NXP standard. Please refer to the NXP documentation for more information.

Command description			
Argument	Size	Value	Description
Command ID	1	0x42	MFU_GET_VERSION
Response description			
ACK	1	0x00	
Command ID	1	0x42	MFU_GET_VERSION
Version bytes	8	X	Version bytes from the TAG

Example:

HOST=>READER: 0x42 – MFU_GET_VERSION
 READER=>HOST: 0x00 – ACK byte
 0x42 – related command code MFU_GET_VERSION
 0x31 0x35 0x3a 0x33 0x35 0x3a 0x30 0x33 – version bytes

5.4.4 Read signature (0x43)

This command requests a version string from the device. The returned ACK answer contains 32-bytes with ECC signature defined by the NXP standard. Please refer to the NXP documentation for more information.

Command description			
Argument	Size	Value	Description
Command ID	1	0x43	MFU_READ_SIGNATURE
Response description			
ACK	1	0x00	
Command ID	1	0x43	MFU_READ_SIGNATURE
Version bytes	32	X	Signature bytes from the TAG

Example:

HOST=>READER: 0x43 – MFU_READ_SIGNATURE
 READER=>HOST: 0x00 – ACK byte
 0x43 – related command code MFU_READ_SIGNATURE
 0x01 0x2e 0x41 0x22 0x43 0x11 0x8e 0x20
 0x31 0x38 0x20 0x32 0x30 0x31 0x39 0x41
 0x81 0x23 0x42 0x28 0x33 0x01 0x8e 0x72
 0x31 0x35 0x3a 0x33 0x35 0x3a 0x30 0x33 – signature bytes

5.4.5 Write signature (0x44)

This command writes the signature information to the MIFARE Ultralight Nano TAG. It takes as arguments relative page location of the signature part to be written and four bytes of signature value to be written.

Command description			
Argument	Size	Value	Description
Command ID	1	0x44	MFU_WRITE_SIGNATURE
Relative page address	1	X	Relative page location of the signature part to be written
Bytes to write	4	XXX	Bytes of signature value to be written to the specified relative page address
Response description			
ACK	1	0x00	
Command ID	1	0x44	MFU_WRITE_SIGNATURE

Example:

```

HOST=>READER: 0x44 - MFU_WRITE_SIGNATURE
               0x00 - relative page number 0
               0x35 0x3a 0x30 0x33 - 4 bytes to write

READER=>HOST: 0x00 - ACK byte
               0x44 - related command code MFU_WRITE_SIGNATURE
    
```

5.4.6 Lock signature (0x45)

This command locks the signature temporarily or permanently based on the information provided in the API. The locking and unlocking of the signature can be performed using this command if the signature is not locked or temporary locked. If the signature is permanently locked, then unlocking can't be done.

Command description			
Argument	Size	Value	Description
Command ID	1	0x45	MFU_LOCK_SIGNATURE
Lock mode	1	X	0x00 – Unlock 0x01 – Lock 0x02 – Permanent lock
Response description			
ACK	1	0x00	
Command ID	1	0x45	MFU_LOCK_SIGNATURE

Example:

```

HOST=>READER: 0x45 - MFU_LOCK_SIGNATURE
               0x02 - permanent lock

READER=>HOST: 0x00 - ACK byte
               0x45 - related command code MFU_LOCK_SIGNATURE
    
```

5.4.7 Read counter (0x46)

This command should be used to read a counter from the TAG. It takes as arguments the counter number. The returned ACK response contains a value as a signed 24-bit value (LSB first).

Command description			
Argument	Size	Value	Description
Command ID	1	0x46	MFU_READ_COUNTER
Counter number	1	0-2	Counter number
Response description			
ACK	1	0x00	
Command ID	1	0x46	MFU_READ_COUNTER
Counter value	3	X	Unsigned 24-bit value, LSB first

Example:

```

HOST=>READER: 0x46 - MFU_READ_COUNTER
                0x01 - counter number

READER=>HOST: 0x00 - ACK byte
                0x46 - related command code MFU_READ_COUNTER
                0x00 0x00 0x01 - value
  
```

5.4.8 Increment counter (0x47)

This command should be used to increment a counter stored in the tag memory. It takes as arguments the counter number and increment value (24-bit value LSB first) as 3 bytes.

Command description			
Argument	Size	Value	Description
Command ID	1	0x47	MFU_INCREMENT_COUNTER
Counter number	1	0-2	Counter number
Increment value	3	X	Unsigned 24-bit value (LSB first)
Response description			
ACK	1	0x00	
Command ID	1	0x47	MFU_INCREMENT_COUNTER

Example:

```

HOST=>READER: 0x47 - MFU_INCREMENT_COUNTER
                0x02 - block number 2
                0x00 0x00 0x01 - increment value

READER=>HOST: 0x00 - ACK byte
                0x47 - related command code MFU_INCREMENT_COUNTER
  
```

5.4.9 Password auth (0x48)

This command tries to authenticate the tag using the chosen password. It takes as an argument a password as four bytes. The returned ACK response contains two bytes of password acknowledge (PACK).

Command description			
Argument	Size	Value	Description
Command ID	1	0x48	MFU_PASSWORD_AUTH
Counter number	4	X	4-bytes password
Response description			

ACK	1	0x00	
Command ID	1	0x48	MFU_PASSWORD_AUTH
PACK	2	X	Password acknowledge bytes

Example:

HOST=>READER: 0x48 – MFU_PASSWORD_AUTH
0x00 0x00 0x00 0x00 – password

READER=>HOST: 0x00 – ACK byte
0x48 – related command code MFU_PASSWORD_AUTH
0x00 0x00 – password acknowledge bytes

5.4.10 Ultralight-C authenticate (0x49)

This command tries to authenticate the MIFARE Ultralight-C tag using the password stored in the key storage. It takes as an argument one byte with the key number in the key storage.

Command description			
Argument	Size	Value	Description
Command ID	1	0x49	MFUC_AUTHENTICATE
Key number	1	0-4	Key number in key storage
Response description			
ACK	1	0x00	
Command ID	1	0x49	MFUC_AUTHENTICATE

Example:

HOST=>READER: 0x49 – MFUC_AUTHENTICATE
0x00 – key number

READER=>HOST: 0x00 – ACK byte
0x49 – related command code MFUC_AUTHENTICATE

5.4.11 Check Tearing Event (0x4A)

The Check Tearing Event command takes as arguments one byte with the counter number. This command checks whether there was a tearing event in the counter. The returned ACK response contains result byte. The value '0x00' is returned if there has been no tearing event, and '0x01' is returned if a tearing event occurred. Please refer to the NXP documentation for more information.

Command description			
Argument	Size	Value	Description
Command ID	1	0x4A	MFU_CHECKEVENT
Key number	1	0-4	Key number in key storage
Response description			
ACK	1	0x00	
Command ID	1	0x4A	MFU_CHECKEVENT

Example:

HOST=>READER: 0x4A – MFU_CHECKEVENT
0x00 – counter number

READER=>HOST: 0x00 – ACK byte

0x4A – related command code MFU_CHECKEVENT
0x01 – tearing event occurred

5.5 MIFARE DESFire commands

This set of commands should be performed on MIFARE DESFire tags.

5.5.1 Get version (0x60)

This command requests version information from the tag. The returned ACK answer contains 28-bytes with version information.

Command description			
Argument	Size	Value	Description
Command ID	1	0x60	MFDF_GET_VERSION
Response description			
ACK	1	0x00	
Command ID	1	0x60	MFDF_GET_VERSION
Read data	28	XXX	Version bytes read from the tag

Example:

```

HOST=>READER:    0x60 - MFDF_GET_VERSION

READER=>HOST:    0x00 - ACK byte
                  0x60 - related command code MFDF_GET_VERSION
                  0x01 0x2e 0x41 0x22 0x43 0x11 0x8e 0x20
                  0x31 0x38 0x20 0x32 0x30 0x31 0x39 0x41
                  0x81 0x23 0x42 0x28 0x33 0x01 0x8e 0x72
                  0x31 0x35 0x3a 0x33 - 28 bytes result
  
```

5.5.2 Select application (0x61)

This command requests select application operation on the tag. Takes as argument 3-bytes containing AID.

Command description			
Argument	Size	Value	Description
Command ID	1	0x61	MFDF_GET_VERSION
AID	3	X	Application ID
Response description			
ACK	1	0x00	
Command ID	1	0x61	MFDF_GET_VERSION

Example:

```

HOST=>READER:    0x61 - MFDF_SELECT_APP
                  0x01 0x02 0x03 - 3 bytes AID

READER=>HOST:    0x00 - ACK byte
                  0x61 - related command code MFDF_SELECT_APP
  
```

5.5.3 List application IDs (0x62)

This command requests lists application IDs from the TAG. The returned ACK answer contains the bytes with application IDs. Every ID is 3-bytes long.

Command description			
Argument	Size	Value	Description
Command ID	1	0x62	MFDF_LIST_APP_IDS
Response description			
ACK	1	0x00	
Command ID	1	0x62	MFDF_LIST_APP_IDS
Application IDs	X*3	X	Bytes with applications IDs

Example:

```

HOST=>READER:    0x62 - MFDF_LIST_APP_IDS

READER=>HOST:    0x00 - ACK byte
                  0x62 - related command code MFDF_LIST_APP_IDS
                  0x00 0x00 0x01 - first AID
                  0xAA 0xBB 0xCC - second AID
                  0x55 0x55 0x55 - third AID
                  ...

```

5.5.4 List files IDs (0x63)

This command returns the file IDs of all active files within the currently selected application. The returned ACK answer contains the bytes with file IDs. Every file ID is 3-bytes long.

Command description			
Argument	Size	Value	Description
Command ID	1	0x63	MFDF_LIST_FILE_IDS
Response description			
ACK	1	0x00	
Command ID	1	0x63	MFDF_LIST_FILE_IDS
Application IDs	X*3	X	Bytes with files IDs

Example:

```

HOST=>READER:    0x63 - MFDF_LIST_FILE_IDS

READER=>HOST:    0x00 - ACK byte
                  0x63 - related command code MFDF_LIST_FILE_IDS
                  0x00 0x00 0x01 - first file ID
                  0xAA 0xBB 0xCC - second file ID
                  0x55 0x55 0x55 - third file ID
                  ...

```

5.5.5 Authenticate (0x64)

This command tries to authenticate the MIFARE DESFire using the password stored in the key storage. It takes as an argument one byte with the key number in the key storage, and one byte with the key number on the card. This command can be used with DES and 2K3DES keys.

Command description			
Argument	Size	Value	Description
Command ID	1	0x64	MFDF_AUTHENTICATE
Key number in storage	1	0-4	Key number in key storage
Key number on card	1	x	Key number on card
Response description			
ACK	1	0x00	
Command ID	1	0x64	MFDF_AUTHENTICATE

Example:

```

HOST=>READER:  0x64 - MFDF_AUTHENTICATE
                0x00 - key number

READER=>HOST:  0x00 - ACK byte
                0x64 - related command code MFDF_AUTHENTICATE
  
```

5.5.6 Authenticate ISO (0x65)

This command tries to authenticate the MIFARE DESFire tag in ISO CBS send mode using the key stored in the key storage. It takes as an argument one byte with the key number in the key storage, and one byte with the key number on the card. This command can be used with DES, 3DES and 3K3DES keys.

Command description			
Argument	Size	Value	Description
Command ID	1	0x65	MFDF_AUTHENTICATE_ISO
Key number	1	0-4	Key number in key storage
Key number on card	1	X	Key number on card
Response description			
ACK	1	0x00	
Command ID	1	0x65	MFDF_AUTHENTICATE_ISO

Example:

```

HOST=>READER:  0x65 - MFDF_AUTHENTICATE_ISO
                0x00 - key number

READER=>HOST:  0x00 - ACK byte
                0x65 - related command code MFDF_AUTHENTICATE_ISO
  
```

5.5.7 Authenticate AES (0x66)

This command tries to authenticate the MIFARE DESFire using the key stored in the key storage, and one byte with the key number on the card. It takes as an argument one byte with the key number in the key storage. This command can be used with AES128 keys.

Command description			
Argument	Size	Value	Description
Command ID	1	0x66	MFDF_AUTHENTICATE_ISO
Key number	1	0-4	Key number in key storage
Response description			
ACK	1	0x00	
Command ID	1	0x66	MFDF_AUTHENTICATE_ISO

Example:

```

HOST=>READER:  0x66 - MFDF_AUTHENTICATE_AES
                0x00 - key number

READER=>HOST:  0x00 - ACK byte
                0x66 - related command code MFDF_AUTHENTICATE_AES
    
```

5.5.8 Create application (0x67)

This command tries to create application on the tag. It takes three arguments: 3-bytes of application ID, the keySettings1 byte and the keySettings2 byte. Please refer to the NXP documentation for more information about key settings bytes.

Command description			
Argument	Size	Value	Description
Command ID	1	0x67	MFDF_CREATE_APP
Application ID	3	X	Application ID bytes
Key settings 1	1	X	Please refer to the NXP documentation for more information
Key settings 2	1	X	Please refer to the NXP documentation for more information
Response description			
ACK	1	0x00	
Command ID	1	0x67	MFDF_CREATE_APP

Example:

```

HOST=>READER:  0x67 - MFDF_CREATE_APP
                0x00 - key number
                0x01 0x02 0x03 - application ID
                0xED 0x84 - key settings bytes

READER=>HOST:  0x00 - ACK byte
                0x67 - related command code MFDF_CREATE_APP
    
```

5.5.9 Delete application (0x68)

This command tries to delete an application from the tag. It takes one argument with the application ID.

Command description			
Argument	Size	Value	Description
Command ID	1	0x68	MFDF_DELETE_APP
Application ID	3	X	Application ID bytes
Response description			
ACK	1	0x00	
Command ID	1	0x68	MFDF_DELETE_APP

Example:

```

HOST=>READER:  0x68 - MFDF_DELETE_APP
                0x01 0x02 0x03 - application ID

READER=>HOST:  0x00 - ACK byte
                0x68 - related command code MFDF_DELETE_APP
  
```

5.5.10 Change key (0x69)

This command tries to change the key for the selected application. It takes three arguments: the old key number from key storage, the new key number in the key storage and the key number on the card. The key type of the application keys cannot be changed.

Command description			
Argument	Size	Value	Description
Command ID	1	0x69	MFDF_CHANGE_KEY
Old key number	1	0-4	Key number in key storage
New key number	1	0-4	Key number in key storage
Key number on card	1	X	Key number on the card
Response description			
ACK	1	0x00	
Command ID	1	0x69	MFDF_CHANGE_KEY

Example:

```

HOST=>READER:  0x69 - MFDF_CHANGE_APP
                0x00 - old key number
                0x01 - new key number
                0x00 - key number

READER=>HOST:  0x00 - ACK byte
                0x69 - related command code MFDF_CHANGE_APP
  
```

5.5.11 Get key settings (0x6A)

This command gets the key settings bytes from the tag. This command does not require any arguments but an application must be selected and authorized.

Command description			
Argument	Size	Value	Description
Command ID	1	0x6A	MFDF_GET_KEY_SETTINGS
Response description			
ACK	1	0x00	
Command ID	1	0x6A	MFDF_GET_KEY_SETTINGS
Key settings	2	X	Key settings bytes

Example:

```

HOST=>READER:    0x6A - MFDF_GET_KEY_SETTINGS

READER=>HOST:    0x00 - ACK byte
                  0x6A - related command code MFDF_GET_KEY_SETTINGS
                  0x01 0x02 - key settings bytes
  
```

5.5.12 Change key settings (0x6B)

This command changes the key settings bytes for the selected and authorized application. It takes one argument, 2-bytes long with key settings.

Command description			
Argument	Size	Value	Description
Command ID	1	0x6B	MFDF_CHANGE_KEY_SETTINGS
New key settings	2	X	Key settings bytes
Response description			
ACK	1	0x00	
Command ID	1	0x6B	MFDF_CHANGE_KEY_SETTINGS

Example:

```

HOST=>READER:    0x6B - MFDF_GET_KEY_SETTINGS
                  0x01 0x02 - key settings bytes

READER=>HOST:    0x00 - ACK byte
                  0x6B - related command code MFDF_GET_KEY_SETTINGS
  
```

5.5.13 Create standard or backup data file (0x6C)

This command creates a file for the storage of plain unformatted user data within the selected application. It takes four arguments listed in the table below.

Command description			
Argument	Size	Value	Description
Command ID	1	0x6C	MFDF_CREATE_DATA_FILE
File number	1	X	File number inside application

Access rights	2	X	Please refer to the NXP documentation for more information
File size	3	X	file size, LSB first
Backup file	1	X	0x00 – Standard file 0x01 – Backup file
Response description			
ACK	1	0x00	
Command ID	1	0x6C	MFDF_CREATE_DATA_FILE

Example:

```

HOST=>READER:  0x6C – MFDF_CREATE_DATA_FILE
                0x01 – file number
                0xEE 0xEE – access rights
                0x40 0x00 0x00 – file 64-bytes long
                0x01 – backup file

READER=>HOST:   0x00 – ACK byte
                0x6C – related command code MFDF_CREATE_DATA_FILE
  
```

5.5.14 Write data (0x6D)

This command writes data to standard data files or backup data files. It takes three arguments: the file number, the offset in the file where data should be stored, and the data bytes to be written. To store data on the TAG, a commit transaction command is required.

Command description			
Argument	Size	Value	Description
Command ID	1	0x6D	MFDF_WRITE_DATA
File number	1	X	File number inside application
File offset	3	X	file offset, 3-bytes LSB value
Data	N	X	Data bytes to write
Response description			
ACK	1	0x00	
Command ID	1	0x6D	MFDF_WRITE_DATA

Example:

```

HOST=>READER:  0x6D – MFDF_WRITE_DATA
                0x01 – file number
                0x00 0x00 0x00 – zero offset
                0x01 0x02 0x03 0x04 0x05 0x06 0x07 – data

READER=>HOST:   0x00 – ACK byte
                0x6D – related command code MFDF_WRITE_DATA
  
```

5.5.15 Read data (0x6E)

This command reads data from standard data files or backup data files. It takes three arguments: the file number, the offset in the file where data is stored, and the number of bytes to be read. The returned ACK response contains the data that has been read.

Command description			
Argument	Size	Value	Description
Command ID	1	0x6E	MFDF_READ_DATA
File number	1	X	File number inside application
File offset	3	X	file offset, 3-bytes LSB value
Data length	3	X	Read data length, 3-bytes LSB value
Response description			
ACK	1	0x00	
Command ID	1	0x6E	MFDF_READ_DATA

Example:

```

HOST=>READER:  0x6E - MFDF_READ_DATA
                0x01 - file number
                0x00 0x00 0x00 - zero offset
                0x07 0x00 0x00 - seven bytes to read
READER=>HOST:  0x00 - ACK byte
                0x6E - related command code MFDF_READ_DATA
                0x01 0x02 0x03 0x04 0x05 0x06 0x07 - data
    
```

5.5.16 Create value file (0x6F)

This command creates files for the storage and manipulation of 32bit signed integer values within an existing application on the TAG. It takes seven arguments listed in the table below.

Command description			
Argument	Size	Value	Description
Command ID	1	0x6F	MFDF_CREATE_VALUE_FILE
File number	1	X	File number inside application
Access rights	2	X	Please refer to the NXP documentation for more information
Low limit	4	X	Low limit as 4-bytes signed value, LSB first
Up limit	4	X	Up limit as 4-bytes signed value, LSB first
Initial value	4	X	Initial value as 4-bytes signed value, LSB first
Get free enabled	1	X	Please refer to the NXP documentation for more information
Limit credited	1	X	Please refer to the NXP documentation for more information
Response description			
ACK	1	0x00	
Command ID	1	0x6F	MFDF_CREATE_VALUE_FILE

Example:

```

HOST=>READER:  0x6F - MFDF_CREATE_VALUE_FILE
                0x02 - file number
                0xEE 0xEE - access rights
                0x00 0x00 0x00 0x00 - low limit
                0x80 0x00 0x00 0x00 - up limit
                0x00 0x00 0x00 0x00 - initial value
                0x01 - get free enabled
                0x01 - limited credit

READER=>HOST:  0x00 - ACK byte
                0x6F - related command code MFDF_CREATE_VALUE_FILE
    
```

5.5.17 Get value (0x70)

This command returns the value stored in a value file on the TAG. The returned ACK response contains 4 bytes of signed value, LSB-first.

Command description			
Argument	Size	Value	Description
Command ID	1	0x70	MFDF_GET_VALUE
File number	1	X	File number inside application
Response description			
ACK	1	0x00	
Command ID	1	0x70	MFDF_GET_VALUE
Value	4	X	4 bytes signed value, LSB first

Example:

```

HOST=>READER:    0x70 - MFDF_GET_VALUE
                  0x02 - file number

READER=>HOST:    0x00 - ACK byte
                  0x70 - related command code MFDF_GET_VALUE
                  0x05 0x00 0x00 0x00 - 4 bytes signed value, LSB
                                      first
  
```

5.5.18 Credit file (0x71)

This command increases a value stored in a value file on the TAG.

Command description			
Argument	Size	Value	Description
Command ID	1	0x71	MFDF_CREDIT
File number	1	X	File number inside application
Credit value	4	X	4 bytes signed value, LSB first
Response description			
ACK	1	0x00	
Command ID	1	0x71	MFDF_CREDIT

Example:

```

HOST=>READER:    0x71 - MFDF_CREDIT
                  0x02 - file number
                  0x05 0x00 0x00 0x00 - 4 bytes signed value, LSB
                                      first

READER=>HOST:    0x00 - ACK byte
                  0x71 - related command code MFDF_CREDIT
  
```

5.5.19 Credit file (0x72)

This command allows a limited increase of a value stored in a value file without having full credit permissions to the file. Please refer to the NXP documentation for more information.

Command description			
Argument	Size	Value	Description
Command ID	1	0x72	MFDF_LIMITED_CREDIT
File number	1	X	File number inside application
Credit value	4	X	4 bytes signed value, LSB first
Response description			
ACK	1	0x00	
Command ID	1	0x72	MFDF_LIMITED_CREDIT

Example:

HOST=>READER: 0x72 – MFDF_LIMITED_CREDIT
 0x02 – file number
 0x05 0x00 0x00 0x00 – 4 bytes signed value, LSB first

READER=>HOST: 0x00 – ACK byte
 0x72 – related command code MFDF_LIMITED_CREDIT

5.5.20 Debit file (0x73)

This command decreases a value stored in a value file on the TAG.

Command description			
Argument	Size	Value	Description
Command ID	1	0x73	MFDF_DEBIT
File number	1	X	File number inside application
Credit value	4	X	4 bytes signed value, LSB first
Response description			
ACK	1	0x00	
Command ID	1	0x73	MFDF_DEBIT

Example:

HOST=>READER: 0x73 – MFDF_DEBIT
 0x02 – file number
 0x05 0x00 0x00 0x00 – 4 bytes signed value, LSB first

READER=>HOST: 0x00 – ACK byte
 0x73 – related command code MFDF_DEBIT

5.5.21 Create record file (0x74)

This command creates files for multiple storage of structurally similar data within an existing application. If the cyclic flag is 0x00, then further writing is not possible unless it is cleared. If the cyclic flag is set to 0x01, then the new record overwrites the oldest record.

Command description			
Argument	Size	Value	Description
Command ID	1	0x74	MFDF_CREATE_RECORD_FILE
File number	1	X	File number inside application
Access rights	2	X	Please refer to the NXP documentation for more information
Record size	2	X	Record size, 16-bits LSB value
Number of records	2	X	Number of records, 16-bits LSB value
Cyclic flag	1	X	If cyclic file is full: 0x00 - further writing is not possible unless it is cleared 0x01 - the new record overwrites oldest record
Response description			
ACK	1	0x00	
Command ID	1	0x74	MFDF_CREATE_RECORD_FILE

Example:

```

HOST=>READER:  0x74 - MFDF_CREATE_RECORD_FILE
                0x03 - file number
                0xEE 0xEE - access rights
                0x08 0x00 - 8-bytes for every record
                0x40 0x00 - 64 records
                0x01 - cyclic flag

READER=>HOST:  0x00 - ACK byte
                0x74 - related command code MFDF_CREATE_VALUE_FILE
  
```

5.5.22 Write record (0x75)

This command writes data to a record file. It takes two arguments: the file number and the data bytes to be written. To store data on the TAG, a commit transaction command is required.

Command description			
Argument	Size	Value	Description
Command ID	1	0x75	MFDF_WRITE_RECORD_DATA
File number	1	X	File number inside application
Data	N	X	Data bytes to write
Response description			
ACK	1	0x00	
Command ID	1	0x75	MFDF_WRITE_DATA

Example:

```

HOST=>READER:  0x75 - MFDF_WRITE_DATA
                0x01 - file number
                0x01 0x02 0x03 0x04 0x05 0x06 0x07 - data

READER=>HOST:  0x00 - ACK byte
                0x75 - related command code MFDF_WRITE_RECORD_DATA
  
```

5.5.23 Read record (0x76)

This command reads data from a record file. It takes three arguments: the file number, the record number, and the number of bytes to be read. The returned ACK response contains the data that has been read.

Command description			
Argument	Size	Value	Description
Command ID	1	0x76	MFDF_READ_RECORD
File number	1	X	File number inside application
Record number	2	X	Record number, 2-bytes LSB value
Data length	2	X	Read data length, 2-bytes LSB value
Response description			
ACK	1	0x00	
Command ID	1	0x76	MFDF_READ_RECORD

Example:

```

HOST=>READER:  0x76 - MFDF_READ_RECORD
                0x01 - file number
                0x00 0x01 - record number
                0x08 0x00 - eighth bytes to read
READER=>HOST:  0x00 - ACK byte
                0x76 - related command code MFDF_READ_RECORD
                0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 - data
  
```

5.5.24 Clear records (0x77)

This command resets cyclic or lineal record files. It takes as an argument the file number.

Command description			
Argument	Size	Value	Description
Command ID	1	0x77	MFDF_CLEAR_RECORDS
File number	1	X	File number inside application
Response description			
ACK	1	0x00	
Command ID	1	0x77	MFDF_CLEAR_RECORDS

Example:

```

HOST=>READER:  0x77 - MFDF_CLEAR_RECORDS
                0x01 - file number
READER=>HOST:  0x00 - ACK byte
                0x77 - related command code MFDF_CLEAR_RECORDS
  
```

5.5.25 Delete file (0x78)

This command permanently deactivates a file within the file directory of the currently selected application. It takes as an argument the file number.

Command description			
Argument	Size	Value	Description
Command ID	1	0x78	MFDF_DELETE_FILE
File number	1	X	File number inside application
Response description			
ACK	1	0x00	
Command ID	1	0x78	MFDF_DELETE_FILE

Example:

```

HOST=>READER:  0x78 - MFDF_DELETE_FILE
                0x01 - file number

READER=>HOST:  0x00 - ACK byte
                0x78 - related command code MFDF_DELETE_FILE
    
```

5.5.26 Get free memory (0x79)

This command returns a value corresponding to the amount of free memory available on the TAG. No arguments are required. The available memory is returned as a 4 byte unsigned LSB value.

Command description			
Argument	Size	Value	Description
Command ID	1	0x79	MFDF_GET_FREE_MEM
Response description			
ACK	1	0x00	
Command ID	1	0x79	MFDF_GET_FREE_MEM
Free memory	4	X	Free memory, 4-bytes, LSB first

Example:

```

HOST=>READER:  0x79 - MFDF_GET_FREE_MEM

READER=>HOST:  0x00 - ACK byte
                0x79 - related command code MFDF_GET_FREE_MEM
                0x00 0x08 0x00 0x00 - free memory
    
```

5.5.27 Format memory (0x7A)

This command releases user memory in the TAG. No arguments are required.

Command description			
Argument	Size	Value	Description
Command ID	1	0x7A	MFDF_FORMAT
Response description			
ACK	1	0x00	
Command ID	1	0x7A	MFDF_FORMAT

Example:

```

HOST=>READER:  0x7A - MFDF_FORMAT

READER=>HOST:  0x00 - ACK byte
                0x7A - related command code MFDF_FORMAT
    
```

5.5.28 Commit transaction (0x7B)

This command validates all previous write access on backup data files, value files and record files within one application. No arguments are required.

Command description			
Argument	Size	Value	Description
Command ID	1	0x7B	MFDF_COMMIT_TRANSACTION
Response description			
ACK	1	0x00	
Command ID	1	0x7B	MFDF_COMMIT_TRANSACTION

Example:

HOST=>READER: 0x7B - MFDF_COMMIT_TRANSACTION

READER=>HOST: 0x00 - ACK byte
0x7B - related command code MFDF_COMMIT_TRANSACTION

5.5.29 Abort transaction (0x7C)

This command invalidates all previous write access on backup data files, value files and record files within one application. No arguments are required.

Command description			
Argument	Size	Value	Description
Command ID	1	0x7C	MFDF_ABORT_TRANSACTION
Response description			
ACK	1	0x00	
Command ID	1	0x7C	MFDF_ABORT_TRANSACTION

Example:

HOST=>READER: 0x7C - MFDF_ABORT_TRANSACTION

READER=>HOST: 0x00 - ACK byte
0x7C - related command code MFDF_ABORT_TRANSACTION

5.6 ICODE (ISO15693) commands

This set of commands should be performed on ICODE (ISO15693) TAGs.

5.6.1 Inventory start (0x90)

This command starts the inventory procedure on ISO 15693 TAGs. It activates the first TAG detected during collision resolution. If no TAGs are detected, then an error with a timeout flag is returned. This command takes one argument AFI - Application Family Identifier. Please refer to the NXP documentation for more information.

If any TAG(s) is/are detected, then the command returns an ACK message containing the UID (8-bytes), a DSFID byte, and 1-byte which contains information about any other tags detected in the field that are available to be read.

Because GET_TAG_COUNT command is limited to 5 tags only, ICODE_INVENTORY_START/ICODE_INVENTORY_NEXT commands should be used to detect all ICODE tags within range of the antenna.

Command description			
Argument	Size	Value	Description
Command ID	1	0x90	ICODE_INVENTORY_START
AFI	1	X	Application Family Identifier
Response description			
ACK	1	0x00	
Command ID	1	0x90	ICODE_INVENTORY_START
UID	8	XXX	Unique identifier
DSFID	1	X	Data Storage Format Identifier
More cards flag	1	X	0x00 – no more cards in range of antenna 0x01 – more cards in range of antenna

Example:

```

HOST=>READER:    0x90 – ICODE_INVENTORY_START
                  0x00 – Application Family Identifier

READER=>HOST:    0x00 – ACK byte
                  0x90 – related command code ICODE_INVENTORY_START
                  0x04 0x8F 0x7F 0x0A 0x01 0x24 0x16 0xE0 – UID
                  0x00 – DSFID
                  0x01 – more cards in range of antenna
  
```

5.6.2 Inventory next (0x91)

This command should be used to continue the inventory procedure on ISO 15693 TAGs. It activates the next TAG that was detected during the collision resolution. It takes one argument, AFI - Application Family Identifier. Please refer to the NXP documentation for more information. If a TAG or multiple tags is/are detected, then this command returns an ACK message containing the UID (8-bytes), a DSFID byte, and 1-byte which contains information about any other tags detected in the field that are available to be read.

Command description			
Argument	Size	Value	Description
Command ID	1	0x91	ICODE_INVENTORY_NEXT
AFI	1	X	Application Family Identifier
Response description			
ACK	1	0x00	
Command ID	1	0x91	ICODE_INVENTORY_NEXT
UID	8	XXX	Unique identifier
DSFID	1	X	Data Storage Format Identifier
More cards flag	1	X	0x00 – no more cards in range of antenna 0x01 – more cards in range of antenna

Example:

```

HOST=>READER: 0x91 – ICODE_INVENTORY_NEXT
                0x00 – Application Family Identifier

READER=>HOST: 0x00 – ACK byte
                0x91 – related command code ICODE_INVENTORY_NEXT
                0x04 0x8F 0x7F 0x0A 0x01 0x24 0x16 0xE0 – UID
                0x00 – DSFID
                0x00 – no more cards available for reading
    
```

5.6.3 Stay quiet (0x92)

This command performs an ISO15693 Stay Quiet command to the selected TAG. When the tag receives the Stay quiet command, it enters the quiet state and will not send back a response. The TAG exits the quiet state upon the execution of a reset (power off) or the command ICODE_INVENTORY_START . Please refer to the NXP documentation for more information.

Command description			
Argument	Size	Value	Description
Command ID	1	0x92	ICODE_STAY_QUIET
Response description			
ACK	1	0x00	
Command ID	1	0x92	ICODE_STAY_QUIET

Example:

```

HOST=>READER: 0x92 – ICODE_STAY_QUIET

READER=>HOST: 0x00 – ACK byte
                0x92 – related command code ICODE_STAY_QUIET
    
```

5.6.4 Read block (0x93)

The read block command should be used to read data stored in TAG blocks. It takes as arguments the block number of the first block to be read, and the number of blocks to be read. The returned ACK answer contains data read from the specified tag memory. The number of bytes of this data is ICODE block size (4) multiplied by the number of blocks to be read.

Command description			
Argument	Size	Value	Description
Command ID	1	0x93	ICODE_READ_BLOCK
Block number	1	X	
Block count	1	N	Number of block to read
Response description			
ACK	1	0x00	
Command ID	1	0x93	ICODE_READ_BLOCK
Read data	4*N	XXX	Bytes read from the tag.

Example:

HOST=>READER: 0x93 – ICODE_READ_BLOCK
 0x02 – block number 2
 0x01 – 1 block to read

READER=>HOST: 0x00 – ACK byte
 0x93 – related command code ICODE_READ_BLOCK
 0x35 0x3a 0x30 0x33 – 4 bytes block data

5.6.5 Write block (0x94)

The write block command should be used to write data to the tag. It takes as arguments the block number of the first block to write, the number of blocks to write, and the bytes to be written. The number of bytes to be written must be exactly the number of blocks to write multiplied by 4.

Command description			
Argument	Size	Value	Description
Command ID	1	0x94	ICODE_WRITE_BLOCK
Block number	1	X	
Block count	1	N	
Data to write	4*N	X	4-bytes data to write
Response description			
ACK	1	0x00	
Command ID	1	0x94	ICODE_WRITE_BLOCK

Example:

HOST=>READER: 0x94 – ICODE_WRITE_BLOCK
 0x02 – block number 2
 0x01 – block count 1
 0x35 0x3a 0x30 0x33 – 4 bytes to write

READER=>HOST: 0x00 – ACK byte
 0x94 – related command code ICODE_WRITE_BLOCK

5.6.6 Lock block (0x95)

This command performs a lock block command. Once it receives the lock block command, the TAG permanently locks the requested block. The command takes a one-byte argument representing the block number to be locked.

Command description			
Argument	Size	Value	Description
Command ID	1	0x95	ICODE_LOCK_BLOCK
Block number	1	X	
Response description			
ACK	1	0x00	
Command ID	1	0x95	ICODE_LOCK_BLOCK

Example:

```

HOST=>READER:  0x95 - ICODE_LOCK_BLOCK
                0x02 - block number 2

READER=>HOST:  0x00 - ACK byte
                0x95 - related command code ICODE_LOCK_BLOCK

```

5.6.7 Write AFI (0x96)

This command performs a write to Application Family Identifier value inside the TAG memory. The command takes a one-byte argument representing the AFI value.

Command description			
Argument	Size	Value	Description
Command ID	1	0x96	ICODE_WRITE_AFI
AFI value	1	X	
Response description			
ACK	1	0x00	
Command ID	1	0x96	ICODE_WRITE_AFI

Example:

```

HOST=>READER:  0x96 - ICODE_WRITE_AFI
                0xAA - new Application Family Identifier value

READER=>HOST:  0x00 - ACK byte
                0x96 - related command code ICODE_WRITE_AFI

```

5.6.8 Lock AFI (0x97)

This command performs a Lock AFI command on the TAG. When it receives the lock AFI request, the TAG locks the AFI value permanently into its memory.

Command description			
Argument	Size	Value	Description
Command ID	1	0x97	ICODE_LOCK_AFI
Response description			
ACK	1	0x00	
Command ID	1	0x97	ICODE_LOCK_AFI

Example:

HOST=>READER: 0x96 – ICODE_LOCK_AFI

READER=>HOST: 0x00 – ACK byte
0x96 – related command code ICODE_LOCK_AFI

5.6.9 Write DSFID (0x98)

This command performs a write to Data Storage Format Identifier value inside the TAG memory. This command takes a one-byte argument representing the DSFID value.

Command description			
Argument	Size	Value	Description
Command ID	1	0x98	ICODE_WRITE_DSFID
DSFID value	1	X	
Response description			
ACK	1	0x00	
Command ID	1	0x98	ICODE_WRITE_DSFID

Example:

HOST=>READER: 0x98 – ICODE_WRITE_DSFID
0xAA – new Data Storage Format Identifier value

READER=>HOST: 0x00 – ACK byte
0x98 – related command code ICODE_WRITE_DSFID

5.6.10 Lock DSFID (0x99)

This command performs a Lock DSIFD command on the TAG. When it receives the lock DSFID request, the TAG locks the DSFID value permanently into its memory.

Command description			
Argument	Size	Value	Description
Command ID	1	0x99	ICODE_LOCK_DSFID
Response description			
ACK	1	0x00	
Command ID	1	0x99	ICODE_LOCK_DSFID

Example:

HOST=>READER: 0x99 – ICODE_LOCK_DSFID

READER=>HOST: 0x00 – ACK byte
0x99 – related command code ICODE_LOCK_DSFID

5.6.11 Get System Information (0x9A)

This command performs get system information command on the TAG. No arguments are required. The ACK response contains bytes with system information. Please refer to the NXP documentation for more information.

Command description			
Argument	Size	Value	Description
Command ID	1	0x9A	ICODE_GET_SYSTEM_INFORMATION
Response description			
ACK	1	0x00	
Command ID	1	0x9A	ICODE_GET_SYSTEM_INFORMATION
System information	X	XXX	System information bytes

Example:

```

HOST=>READER:    0x9A - ICODE_GET_SYSTEM_INFORMATION

READER=>HOST:    0x00 - ACK byte
                  0x9A - related command code ICODE_GET_SYSTEM_INFORMATION
                  0x0F 0x04 0x8F 0x7F 0x0A 0x01 0x24
                  0x16 0xE0 0x00 0x00 0x33 0x03 0x02 - result bytes
  
```

5.6.12 Get multiple BSS (0x9B)

This command performs get multiple block security status command on the TAG. It takes as arguments the block number for which the status should be returned and the number of blocks to be used for returning the status. The ACK response contains bytes with block security status information. Please refer to the NXP documentation for more information.

Command description			
Argument	Size	Value	Description
Command ID	1	0x9B	ICODE_GET_MULTIPLE_BSS
First block number	1	X	
Number of blocks	1	N	
Response description			
ACK	1	0x00	
Command ID	1	0x9B	ICODE_GET_MULTIPLE_BSS
BSS information	N	X	Blocks security status information

Example:

```

HOST=>READER:    0x9B - ICODE_GET_MULTIPLE_BSS
                  0x00 - starting block number
                  0x08 - number of BSS to read

READER=>HOST:    0x00 - ACK byte
                  0x9B - related command code ICODE_GET_MULTIPLE_BSS
                  0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 - result bytes
  
```

5.6.13 Password protect AFI (0x9C)

This command enables the password protection for AFI. The AFI password has to be transmitted before with ICODE_SET_PASSWORD command.

Command description			
Argument	Size	Value	Description
Command ID	1	0x9C	ICODE_PASSWORD_PROTECT_AFI
Response description			
ACK	1	0x00	
Command ID	1	0x9C	ICODE_PASSWORD_PROTECT_AFI

Example:

HOST=>READER: 0x9C – ICODE_PASSWORD_PROTECT_AFI

READER=>HOST: 0x00 – ACK byte
0x9C – related command code ICODE_PASSWORD_PROTECT_AFI

5.6.14 Read EPC (0x9D)

This command reads EPC data from the TAG. The ACK response contains 12-bytes of EPC data. Please refer to the NXP documentation for more information.

Command description			
Argument	Size	Value	Description
Command ID	1	0x9D	ICODE_READ_EPC
Response description			
ACK	1	0x00	
Command ID	1	0x9D	ICODE_READ_EPC
EPC information	12	X	Please refer to the NXP documentation for more information.

Example:

HOST=>READER: 0x9D – ICODE_READ_EPC

READER=>HOST: 0x00 – ACK byte
0x9D – related command code ICODE_READ_EPC
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 – result bytes

5.6.15 Get NXP System Information (0x9E)

This command retrieves the NXP system information value from the TAG. No arguments are required. The ACK response contains bytes with the NXP system information. Please refer to the NXP documentation for more information.

Command description			
Argument	Size	Value	Description
Command ID	1	0x9E	ICODE_GET_NXP_SYSTEM_INFORMATION
Response description			
ACK	1	0x00	
Command ID	1	0x9E	ICODE_GET_NXP_SYSTEM_INFORMATION
System information	X	XXX	System information bytes

Example:

```

HOST=>READER:    0x9E – ICODE_GET_NXP_SYSTEM_INFORMATION

READER=>HOST:    0x00 – ACK byte
                  0x9E – related command code
                    ICODE_GET_NXP_SYSTEM_INFORMATION
                  0x0F 0x04 0x8F 0x7F 0x0A 0x01 0x24
                  0x16 0xE0 0x00 0x00 0x33 0x03 0x02 – result bytes
    
```

5.6.16 Get random number (0x9F)

This command requests a random number from the ICODE TAG. No arguments are required. The ACK response contains a 16-bit random number. This value should be used with ICODE_SET_PASSWORD command.

Command description			
Argument	Size	Value	Description
Command ID	1	0x9F	ICODE_GET_RANDOM_NUMBER
Response description			
ACK	1	0x00	
Command ID	1	0x9F	ICODE_GET_RANDOM_NUMBER
Random number	2	XXX	16-bits random number

Example:

```

HOST=>READER:    0x9F – ICODE_GET_RANDOM_NUMBER

READER=>HOST:    0x00 – ACK byte
                  0x9F – related command code ICODE_GET_RANDOM_NUMBER
                  0x7F 0x14 – result bytes
    
```

5.6.17 Set password (0xA0)

This command sets the password for the selected identifier. This command has to be executed just once for the related passwords if the TAG is powered. The password is calculated as XOR with the random number returned by the previously executed command ICODE_GET_RANDOM_NUMBER.

Here is an example how to calculate XOR password:

```

xorPassword[0] = password[0] ^ rnd[0];
xorPassword[1] = password[1] ^ rnd[1];
xorPassword[2] = password[2] ^ rnd[0];
xorPassword[3] = password[3] ^ rnd[1];
    
```

Command description			
Argument	Size	Value	Description
Command ID	1	0xA0	ICODE_SET_PASSWORD
Password Identifier	1	X	0x01 – Read password 0x02 – Write password 0x04 – Privacy password 0x08 – Destroy password

XOR Password	4	X	
Response description			
ACK	1	0x00	
Command ID	1	0xA0	ICODE_SET_PASSWORD

Example:

HOST=>READER: 0xA0 – ICODE_SET_PASSWORD
 0x02 – write password
 0x34 0x76 0x39 0x64 – calculated XOR password

READER=>HOST: 0x00 – ACK byte
 0xA0 – related command code ICODE_SET_PASSWORD

5.6.18 Write password (0xA1)

This command writes a new password to a selected identifier. With this command, a new password is written into the related memory. Note that the old password has to be transmitted before with ICODE_SET_PASSWORD. The new password takes effect immediately which means that the new password has to be transmitted with ICODE_SET_PASSWORD to get access to the protected blocks/pages. It takes as arguments the password identifier byte and the plain password 4-bytes long.

Command description			
Argument	Size	Value	Description
Command ID	1	0xA1	ICODE_WRITE_PASSWORD
Password Identifier	1	X	0x01 – Read password 0x02 – Write password 0x04 – Privacy password 0x08 – Destroy password
Password	4	X	Plain password
Response description			
ACK	1	0x00	
Command ID	1	0xA1	ICODE_WRITE_PASSWORD

Example:

HOST=>READER: 0xA1 – ICODE_WRITE_PASSWORD
 0x02 – write password
 0x34 0x76 0x39 0x64 – Plain password

READER=>HOST: 0x00 – ACK byte
 0xA1 – related command code ICODE_WRITE_PASSWORD

5.6.19 Lock password (0xA2)

This command locks the addressed password. Note that the addressed password has to be transmitted before with ICODE_SET_PASSWORD. A locked password can no longer be changed.

Command description			
Argument	Size	Value	Description
Command ID	1	0xA2	ICODE_LOCK_PASSWORD

Password Identifier	1	X	0x01 – Read password 0x02 – Write password 0x04 – Privacy password 0x08 – Destroy password
Response description			
ACK	1	0x00	
Command ID	1	0xA2	ICODE_LOCK_PASSWORD

Example:

HOST=>READER: 0xA2 – ICODE_LOCK_PASSWORD
 0x02 – write password

READER=>HOST: 0x00 – ACK byte
 0xA2 – related command code ICODE_LOCK_PASSWORD

5.6.20 Protect page (0xA3)

This command changes the protection status of a page. Note that the related passwords have to be transmitted before with ICODE_SET_PASSWORD if the page is not public. Please refer to the NXP documentation for more information.

Command description			
Argument	Size	Value	Description
Command ID	1	0xA3	ICODE_PAGE_PROTECT
Page address	1	X	<ul style="list-style-type: none"> • Page number to be protected in case of products that do not have pages characterized as high and Low. • Block number to be protected in case of products that have pages characterized as high and Low.
Protection status	1	X	<ul style="list-style-type: none"> • Protection status options for the products that do not have pages characterized as high and Low: 0x00: ICODE_PROTECT_PAGE_PUBLIC 0x01: ICODE_PROTECT_PAGE_READ_WRITE_READ_PASSWORD 0x10: ICODE_PROTECT_PAGE_WRITE_PASSWORD 0x11: ICODE_PROTECT_PAGE_READ_WRITE_PASSWORD_SEPERATE • Extended Protection status options for the products that have pages characterized as high and Low: 0x01: ICODE_PROTECT_PAGE_READ_LOW 0x02: ICODE_PROTECT_PAGE_WRITE_LOW 0x10: ICODE_PROTECT_PAGE_READ_HIGH 0x20: ICODE_PROTECT_PAGE_WRITE_HIGH
Response description			
ACK	1	0x00	
Command ID	1	0xA2	ICODE_PAGE_PROTECT

Example:

HOST=>READER: 0xA3 – ICODE_PAGE_PROTECT
 0x02 – second block selected
 0x01 – ICODE_PROTECT_PAGE_READ_LOW flag selected

READER=>HOST: 0x00 – ACK byte
 0xA3 – related command code ICODE_PAGE_PROTECT

5.6.21 Lock page protection (0xA4)

This command permanently locks the protection status of a page. Note that the related passwords have to be transmitted before with ref ICODE_SET_PASSWORD if the page is not public.

Command description			
Argument	Size	Value	Description
Command ID	1	0xA4	ICODE_LOCK_PAGE_PROTECTION
Page number	1	X	
Response description			
ACK	1	0x00	
Command ID	1	0xA4	ICODE_LOCK_PAGE_PROTECTION

Example:

HOST=>READER: 0xA4 – ICODE_LOCK_PAGE_PROTECTION
 0x02 – page number
 READER=>HOST: 0x00 – ACK byte
 0xA4 – related command code
 ICODE_LOCK_PAGE_PROTECTION

5.6.22 Get multiple block protection status (0xA5)

This instructs the label to return the block protection status of the requested blocks. It takes as arguments the first block number to get the block protection status and the number of blocks.

Command description			
Argument	Size	Value	Description
Command ID	1	0xA5	ICODE_GET_MULTIPLE_BPS
First block number	1	X	
Number of blocks	1	N	
Response description			
ACK	1	0x00	
Command ID	1	0xA5	ICODE_GET_MULTIPLE_BPS
BSS information	N	X	Blocks protection status information

Example:

HOST=>READER: 0xA5 – ICODE_GET_MULTIPLE_BPS
 0x00 – starting block number
 0x08 – number of BSS to read
 READER=>HOST: 0x00 – ACK byte
 0xA5 – related command code ICODE_GET_MULTIPLE_BPS
 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 – result
 bytes

5.6.23 Destroy (0xA6)

This command permanently destroys the label (tag). The destroy password has to be transmitted before with ICODE_SET_PASSWORD. This command is irreversible and the label will never respond to any command again. This command can take the XOR password argument for the ICODE products that requires this argument. The XOR password calculation method is described in the ICODE_SET_PASSWORD description.

Command description			
Argument	Size	Value	Description
Command ID	1	0xA6	ICODE_DESTROY
XOR password	4	X	Optional XOR password
Response description			
ACK	1	0x00	
Command ID	1	0xA6	ICODE_DESTROY

Example:

```
HOST=>READER: 0xA6 - ICODE_DESTROY
READER=>HOST: 0x00 - ACK byte
               0xA6 - related command code ICODE_DESTROY
```

5.6.24 Enable privacy (0xA7)

This command instructs the label to enter privacy mode. In privacy mode, the label will only respond to ICODE_GET_RANDOM_NUMBER and ICODE_SET_PASSWORD commands. To get out of the privacy mode, the Privacy password has to be transmitted before with ICODE_SET_PASSWORD.

Command description			
Argument	Size	Value	Description
Command ID	1	0xA7	ICODE_ENABLE_PRIVACY
XOR password	4	X	Optional XOR password
Response description			
ACK	1	0x00	
Command ID	1	0xA7	ICODE_ENABLE_PRIVACY

Example:

```
HOST=>READER: 0xA7 - ICODE_ENABLE_PRIVACY
READER=>HOST: 0x00 - ACK byte
               0xA7 - related command code ICODE_ENABLE_PRIVACY
```

5.6.25 Enable 64-bit password (0xA8)

This instructs the label that both Read and Write passwords are required for protected access. Note that both the Read and Write passwords have to be transmitted before with ICODE_SET_PASSWORD.

Command description			
Argument	Size	Value	Description
Command ID	1	0xA8	ICODE_ENABLE_64BIT_PASSWORD

Response description			
ACK	1	0x00	
Command ID	1	0xA8	ICODE_ENABLE_64BIT_PASSWORD

Example:

```

HOST=>READER:    0xA8 - ICODE_ENABLE_64BIT_PASSWORD

READER=>HOST:    0x00 - ACK byte
                  0xA8 - related command code
                  ICODE_ENABLE_64BIT_PASSWORD
  
```

5.6.26 Read signature (0xA9)

This command reads the signature bytes from the TAG. No arguments are required. The ACK response contains bytes containing the signature bytes. Please refer to the NXP documentation for more information.

Command description			
Argument	Size	Value	Description
Command ID	1	0xA9	ICODE_READ_SIGNATURE
Response description			
ACK	1	0x00	
Command ID	1	0xA9	ICODE_READ_SIGNATURE
Signature bytes	X	XXX	Signature bytes

Example:

```

HOST=>READER:    0xA9 - ICODE_READ_SIGNATURE

READER=>HOST:    0x00 - ACK byte
                  0xA9 - related command code ICODE_READ_SIGNATURE
                  0x0F 0x04 0x8F 0x7F 0x0A 0x01 0x24
                  0x16 0xE0 0x00 0x00 0x33 0x03 0x02 - result bytes
  
```

5.6.27 Read config (0xAA)

This command reads multiple 4-byte data chunks from the selected configuration block address. It takes two arguments, the first block number and the number of blocks to read the configuration data.

Command description			
Argument	Size	Value	Description
Command ID	1	0xAA	ICODE_READ_CONFIG
First block number	1	X	
Number of blocks	1	N	
Response description			
ACK	1	0x00	
Command ID	1	0xAA	ICODE_READ_CONFIG
Configuration bytes	N*4	X	

Example:

```

HOST=>READER: 0xAA - ICODE_READ_CONFIG
                0x00 - starting block number
                0x02 - number of blocks to read

READER=>HOST: 0x00 - ACK byte
                0xAA - related command code ICODE_READ_CONFIG
                0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 - result bytes
    
```

5.6.28 Write config (0xAB)

This command writes configuration bytes to addressed block data from the selected configuration block address. It takes three arguments: the option byte, the block number and the configuration bytes. Please refer to the NXP documentation for more information.

Command description			
Argument	Size	Value	Description
Command ID	1	0xAB	ICODE_WRITE_CONFIG
Option byte	1	X	0x01 – Enable option 0x00 – Disable option
Block number	1	X	
Configuration bytes	4	X	
Response description			
ACK	1	0x00	
Command ID	1	0xAB	ICODE_WRITE_CONFIG

Example:

```

HOST=>READER: 0xAB - ICODE_WRITE_CONFIG
                0x01 - option byte
                0x00 - block number
                0x00 0x00 0x00 0x00 - config bytes

READER=>HOST: 0x00 - ACK byte
                0xAB - related command code ICODE_WRITE_CONFIG
    
```

5.6.29 Pick random ID (0xAC)

This command enables the random ID generation in the tag. This interface is used to instruct the tag to generate a random number in privacy mode. Please refer to the NXP documentation for more information.

Command description			
Argument	Size	Value	Description
Command ID	1	0xAC	ICODE_PICK_RANDOM_ID
Response description			
ACK	1	0x00	
Command ID	1	0xAC	ICODE_PICK_RANDOM_ID

Example:

HOST=>READER: 0xAB - ICODE_PICK_RANDOM_ID

READER=>HOST: 0x00 - ACK byte
0xAB - related command code ICODE_PICK_RANDOM_ID

6. Compliance

- A. The end equipment that uses this RFID reader module should be tested in its fully operational form to see that it does not exceed the Transmitter RF emissions permitted by the FCC regulations.
- B. The Host device must be affixed with a permanent label that is clearly visible and states the following: *“Contains FCC ID: 2ALHY1356NFC-FCC-IC. Contains IC: 22592-1356NFC-IC”*. Please see *FCC KDB 784748 D01 Labelling Part 15 &18 Guidelines v08 section 8 and RSP-100 Issue 11 section 3.2* for further details on labelling requirements.
- C. The AC powerline conducted emissions test should be performed with the RFID reader module in operation in its host product if that host product connects to the AC powerline.
- D. Changes or modifications not expressly approved by Eccel Technology Ltd could void the user's authority to operate the equipment.
- E. This device complies with Part 15 of the FCC Rules and Industry Canada licence exempt RSS standard(s). Operation is subject to the following two conditions:
 - 1. This device may not cause interference, and
 - 2. This device must accept any interference, including interference that may cause undesired operation of the device.

Le présent appareil est conforme aux CNR d'Industrie Canada applicables aux appareils radio exempts de licence. L'exploitation est autorisée aux deux conditions suivantes:

- 1. l'appareil ne doit pas produire de brouillage, et
- 2. l'utilisateur de l'appareil doit accepter tout brouillage radioélectrique subi, même si le brouillage est susceptible d'en compromettre le fonctionnement.

7. Mechanical dimensions

All dimensions in millimeters.

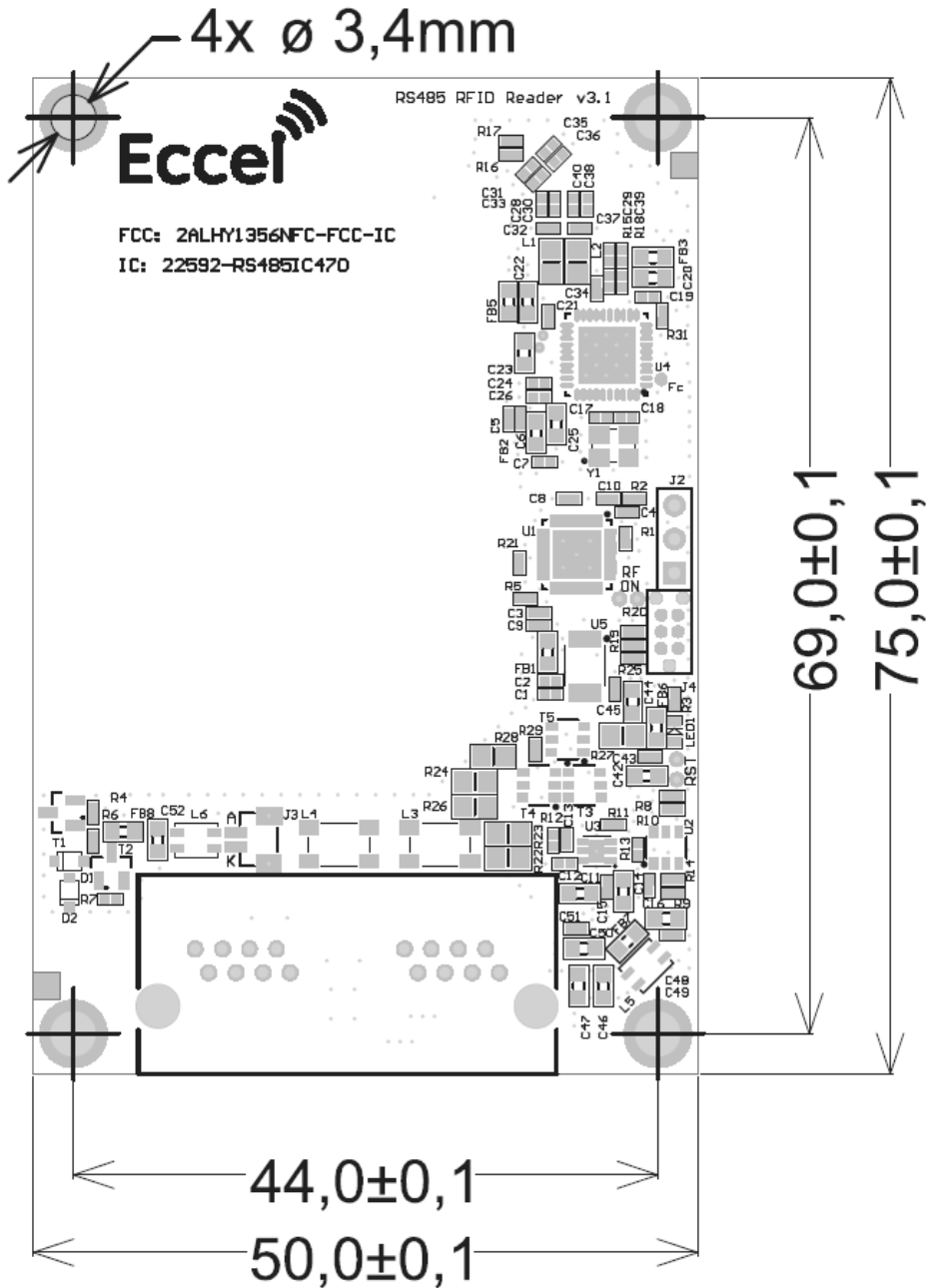


Figure 7-1. Mechanical dimensions

MIFARE, MIFARE Ultralight, MIFARE Plus, MIFARE Classic, and MIFARE DESFire are trademarks of NXP B.V.

No responsibility is taken for the method of integration or final use of the RS485 RFID readers

More information about the RS485 RFID reader and other products can be found at the Internet site:

<http://www.eccel.co.uk>

or alternatively contact ECCEL Technology (IB Technology) by e-mail at:

sales@eccel.co.uk