Communication protocol uFR -VCOM for uFR Reader device

Reader communication type:
– RS232
– 1Mb-8-N-1
Memory space for the usage:
·Mifare Classic Mini: 224 bytes
·Mifare Classic 1k: 752 bytes
·Mifare Classic 4k: 3440 bytes.

Maximum data transferred by command is 64 bytes.
All communication is initiated by the host (PC) to which the device is connected.
The communication is initiated as followed ("CMD" 7 byte package):

CMD_HEADER	command code	CMD_TRAILER	cmd_ext_length (if 0 than the "CMD EXT" is not used)	cmd_param0	cmd_param1	checksum

Checksum represents xor of the whole package (excluding cheksum itself) with addition of 7 on the end. This calculation of cheksum is standard for communication between the host and the device.

Example: checksum = (CMD_HEADER **XOR** command_code **XOR** CMD_TRAILER **XOR** cmd_ext_length **XOR** cmd_param0 **XOR** cmd_param1) **+** 7
XOR is Exclusive disjunction

The device can respond as follows:
• If the command is been sent by the host it requires sending the additional set of data from the host and the devise respond with following "ACK" 7 byte set:

ACK_HEADER	command code	CMD_TRAILER	irrelevant	irrelevant	irrelevant	checksum

After that the device expects "CMD EXT" byte set which size (including the cheksum of that set) is defined in "CMD" set. After successful receiving "CMD EXT" set the device starts data processing and executes commands.
• If the communication error exists or while executing the command the device responds with following "ERR" 7 byte set:

ERR_HEADER	error code	ERR_TRAILER	Number of bytes follows in ERR_EXT set	err_val0	err_val1	checksum

• At the end as a respond to a successfully executed command the device sends "RESPONSE" set and potentially "RESPONSE_EXT" set after that:

RESPONSE_ HEADER	command code	RESPONSE_ TRAILER	response_ext_length "RESPONSE_ EXT" set (if 0 than the "RESPONSE_ EXT" is not used)	response_val0	response_val1	checksum

Communication constants:

```
#define CMD_HEADER 0x55
#define CMD_TRAILER 0xAA
#define ACK_HEADER 0xAC
#define ACK_TRAILER 0xCA
#define RESPONSE_HEADER 0xDE
#define RESPONSE_TRAILER 0xED
#define ERR_HEADER 0xEC
#define ERR_TRAILER 0xCE
```

Command codes are as follows:

```
#define GET_READER_TYPE 0x10
#define GET_READER_SERIAL 0x11
#define READER_KEY_WRITE 0x12
#define GET_CARD_ID 0x13
#define LINEAR_READ 0x14
#define LINEAR_WRITE 0x15
#define BLOCK_READ 0x16
#define BLOCK_WRITE 0x17
#define BLOCK_IN_SECTOR_READ 0x18
#define BLOCK_IN_SECTOR_WRITE 0x19
#define SECTOR_TRAILER_WRITE 0X1A
#define USER_DATA_READ 0x1B
#define USER_DATA_WRITE 0x1C
#define VALUE_BLOCK_READ 0x1D
#define VALUE_BLOCK_WRITE 0x1E
#define VALUE_BLOCK_IN_SECTOR_READ 0x1F
#define VALUE_BLOCK_IN_SECTOR_WRITE 0x20
#define VALUE_BLOCK_INC 0x21
#define VALUE_BLOCK_DEC 0x22
#define VALUE_BLOCK_IN_SECTOR_INC 0x23
#define VALUE_BLOCK_IN_SECTOR_DEC 0x24
#define LINEAR_FORMAT_CARD 0x25
#define SECTOR_TRAILER_WRITE_UNSAFE 0x2F
#define SELF_RESET 0x30
#define GET_CARD_ID_EX 0x2C
#define GET_DLOGIC_CARD_TYPE 0x3C
#define SET_CARD_ID_SEND_CONF       0x3D
#define GET_CARD_ID_SEND_CONF       0x3E
```

Error codes are as follows:

```
#define OK 0x00
#define COMMUNICATION_ERROR 0x01
#define CHKSUM_ERROR 0x02
#define READING_ERROR 0x03
#define WRITING_ERROR 0x04
#define BUFFER_OVERFLOW 0x05
#define MAX_ADDRESS_EXCEEDED 0x06
#define MAX_KEY_INDEX_EXCEEDED 0x07
#define NO_CARD 0x08
#define COMMAND_NOT_SUPPORTED 0x09
#define FORBIDEN_DIRECT_WRITE_IN_SECTOR_TRAILER 0x0A
#define ADDRESSED_BLOCK_IS_NOT_SECTOR_TRAILER 0x0B
#define WRONG_ADDRESS_MODE 0x0C
#define WRONG_ACCESS_BITS_VALUES 0x0D
#define AUTH_ERROR 0x0E
#define PARAMETERS_ERROR 0x0F
#define WRITE_VERIFICATION_ERROR 0x70
```

```
#define BUFFER_SIZE_EXCEEDED 0x71
#define VALUE_BLOCK_INVALID 0x72
#define VALUE_BLOCK_ADDR_INVALID 0x73
#define VALUE_BLOCK_MANIPULATION_ERROR 0x74
```

GET_READER_TYPE

It gives device (reader) type in size of 4 bytes which is hard coded in the firmware.
For IS21 the set value is 0xD1150021.
The CMD_EXT set is not in use.
The cmd_param0 and cmd_param1 are not in use
If everything operates as expected the RESPONSE set is sent and after that also the RESPONSE EXT set of 5 bytes which contains 4 byte DeviceType values (little-endian) and at the end one checksum byte.

GET_READER_SERIAL

It gives the device (reader) serial number with length of 4 bytes. This serial number is been read from EEPROMA MF RC chip of the device.
The CMD_EXT set is not in use.
The cmd_param0 and cmd_param1 are not in use.
If everything operates as expected the RESPONSE set is sent and after that also the RESPONSE EXT set of 5 bytes which contains 4 byte ReaderSerialNumber values (little-endian) and at the end one checksum byte.

Functions for operation with detected cards

For all the functions for operations with cards the following applies:
• They operates only with one card in the device field
• If there is no card in the field it gives error NO_CARD.
• If there is more than one card in the field the behavior of the device is unpredictable but some of the next cases are possible:

- Gives NO_CARD error or
- Just one card is detected and the device gives its type (this is due to the lack of a cascade of selection and the collision process as described in the ISO14443 standard).

GET_CARD_ID

Gives the serial number of the card which is currently in the readers field and the one byte value that represents its type. For Mifare Classic 1K the type is 0x08, Mifare Classic 4k type is 0x18 and Mifare Classic Mini cards type is 0x09.
The CMD_EXT set is not in use.
The cmd_param0 and cmd_param1 are not in use.
If everything operates as expected the RESPONSE set is sent and after that also the RESPONSE EXT set of 5 bytes which contains 4 byte CardSerialNumber values (little endian) and at the end one checksum byte.
response_val0 contains value of the card type.

Functions for reading and writing the data into the card

The parameter AUTH_MODE affects the all functions and determines authorization before reading or entering data in the card sector. This parameter can have the following values:
• RKA_AUTH1A 0x00
• RKA_AUTH1B 0x01

- AKM1_AUTH1A 0x20
- AKM1_AUTH1B 0x21
- AKM2_AUTH1A 0x40
- AKM2_AUTH1B 0x41
- PK_AUTH1A 0x60
- PK_AUTH1B 0x61

From the names of each of these constants can be concluded that the suffixes 1A and 1B indicate that you want to perform authentication key A or key B. As for the prefix in the names of constants they are represent the following:
RKA – shortcut of Reader Key Authentication. This means that authentication will be done with one of the 32 keys that are pre-enrolled in the EEPROM MF RC chip in the reader. It is assumed that as one of the command parameter that is sent to the reader is the index of the desired key.

AKM1 (AKM2) – shortcut of Automatic Key Modes. This means that the authentication will be done automatically with the keys that are pre-enrolled in the EEPROM MF RC chip in the reader and they are indexed on the basis of the block or sector address where the writing or reading is currently done. This applies to any function for card writing and reading, even for linear modes. In AKM1 mode the keys in the reader are indexed so that the authentication with key A for the sectors 0 to 15 takes the keys index by order of 0 to 15 in the reader, and B for the authentication key for the sectors 0 to 15 takes the keys index by order of 16 to 31 in the reader. In AKM2 mode keys in the reader are indexed so that the authentication key A for the sectors 0 to 15 takes the steam keys index in the reader (0, 2, ..., 30), and B for the authentication key for the sectors 0 to 15 takes the odd keys index in the reader(1, 3, ..., 31).

For 4k cards, which have 24 sectors more than 1k cards (total 40) for sectors 16 to 31 is used the same method as for indexing sectors 0 to 15 and for sectors 32 to 39 used the same method of indexing and for sectors 0 to 8.
PK - short for Provided Key refers to the authentication which is performed with key that is sent as a command parameter. Generally, this mode of authentication should be avoided due to the low level of security it provides.

LINEAR_READ

Linear read data from the card that is currently in the field of the reader. Maximum capacity for user data in this mode of reading for different card types are as follows:
Mifare Mini: 224 bytes (out of 320 byte that contains a maximum reading time of entire contents with the fragmentation must be under 85 ms)
Mifare Classic 1k: 752 bytes (maximum reading time of entire contents with the fragmentation must be under 250 ms - cca. 30 ms for 94 bytes)
Mifare Classic 4k: 3440 bytes (problematic for continuous reading because it is inevitable that duration of the reading the complete card content exceeds 1s).
The CMD_EXT set is used whose length depends on the mode of authentication that is used.
cmd_param0 contains AUTH_MODE.

Depending on AUTH_MODE, CMD and CMD_EXT sets contains:

- RKA_AUTH1x:
- cmd_param1 in CMD set contains key index in the
- 1st and 2nd byte of CMD_EXT set contains linear_address (little endian)
- 3th and 4th byte of CMD_EXT set contains data_length (little endian)
- 5th byte contains checksum

- AKMy_AUTH1x:
- cmd_param1 is not used.
- 1st and 2nd byte of CMD_EXT set contains linear_address (little endian)
- 3th and 4th byte of CMD_EXT set contains data_length (little endian)
- 5th byte contains checksum

- PK_AUTH1x:
- cmd_param1 is not used.
- 1st and 2nd byte of CMD_EXT set contains linear_address (little endian)
- 3th and 4th byte of CMD_EXT set contains data_length (little endian)
- array from 5th do 10th byte contains 6-byte key.
- 11st byte contains checksum.

If everything operates as expected the RESPONSE set is sent and after that also the RESPONSE EXT with number of bytes according to the data_length command with checksum at the end.
In case the card is removed from the field or in case of wrong authentication including that some block is read anyway, it turns ERR set with NO_CARD error code or AUTH_ERROR and then the ERR_EXT set which contains the array of the read bytes and checksum at the end.

LINEAR_WRITE
Linear data writing into the card which is currently in the field of the reader. The verification of each written block is done during the writing.
The CMD_EXT set is used and its length depends on the authentication mode that is used cmd_param0 contains AUTH_MODE.
Depending on AUTH_MODE, CMD and CMD_EXT sets contains:

- RKA_AUTH1x:
- cmd_param1 in CMD set contains key index in the reader
- 1st and 2nd byte of CMD_EXT set contains linear_address (little endian)
- 3th and 4th byte of CMD_EXT set contains data_length (little endian)
- from 5th byte up (data_length + 4) contains data array for writing
- (data_length + 5) byte contains checksum

- AKMy_AUTH1x:
- cmd_param1 is not used.
- 1st and 2nd byte of CMD_EXT set contains linear_address (little endian)
- 3th and 4th byte of CMD_EXT set contains data_length (little endian)
- from 5th byte up (data_length + 4) contains data array for writing
- (data_length + 5) byte contains checksum

• PK_AUTH1x:
• cmd_param1 is not used.
• 1st and 2nd byte of CMD_EXT set contains linear_address (little endian)
• 3th and 4th byte of CMD_EXT set contains data_length (little endian)
• array from 5th to 10th byte contains 6- byte key
• 11st byte and up to (data_length + 10) contains data array for writing
• (data_length + 11) byte contains checksum.
If everything went as expected it turns the RESPONSE set.
RESPONSE_EXT is not used.
In error case it turns the ERR set where the response_val0 contains the number of eventual written bytes.

BLOCK_READ
Reads the whole data block from the card which is in the reader field. Address mode is used for so called block addressing where for example the first block on Mifare Classic 1k has an address 0 and the last one has the address 63.
The CMD_EXT set is used and its length depends on authentication mode that is used.

cmd_param0 contains AUTH_MODE.
Depending on AUTH_MODE, CMD and CMD_EXT set contains:

• RKA_AUTH1x:
• cmd_param1 in CMD set contains key index in the reader
• 1st byte of CMD_EXT set contains block_address
• 2nd, 3rd and 4th byte of CMD_EXT set contains dummy data
• 5th byte contains checksum

• AKMy_AUTH1x:
• cmd_param1 is not used.
• 1st byte of CMD_EXT set contains block_address
• 2nd, 3rd and 4th byte of CMD_EXT set contains dummy data
• 5th byte contains checksum

• PK_AUTH1x:
• cmd_param1 is not used.
• 1st byte of CMD_EXT set contains block_address
• 2nd, 3rd and 4th byte of CMD_EXT set contains dummy data
• array from 5th to 10th byte contains 6-byte key.
• 11st byte contains checksum
If all operates as it should it turns the RESPONSE set and the RESPONSE_EXT is following with 16 read bytes and checksum at the end.

BLOCK_WRITE
Writes the whole data block into the card that is currently in the readers field. Address mode is used for so called block addressing where for example the first block on Mifare Classic 1k has an address 0 and the last one has the address 63. This command doesn’t allow the direct writing into the sector

trailer and in the case of its addressing it gives back the FORBIDEN_DIRECT_WRITE_IN_SECTOR_TRAILER.
The CMD_EXT set is used and its length depends on the authentication mode that is in use.
cmd_param0 contains AUTH_MODE.
Depending on AUTH_MODE, CMD and CMD_EXT set contains:

- RKA_AUTH1x:
- cmd_param1 in CMD set contains readers index key
- 1st byte of CMD_EXT set contains block_address
- 2nd, 3rd and 4th byte of CMD_EXT set contains dummy data
- in 5th to 20th byte of set are placed data for writing into the data block
- 21st byte contains checksum

- AKMy_AUTH1x:
- cmd_param1 is not used.
- 1st byte of CMD_EXT set contains block_address
- 2nd, 3rd and 4th byte of CMD_EXT set contains dummy data
- in 5th to 20th byte of the set are placed the data for writing into the data block
- 21st byte contains checksum

- PK_AUTH1x:
- cmd_param1 is not used.
- 1st byte of CMD_EXT set contains block_address
- 2nd, 3rd and 4th byte CMD_EXT set contains dummy data
- array from 5th to 10th byte contains 6-byte key.
- in 11th too 26th byte are placed the data for writing into the data block
- 27th byte contains checksum.

If everything is done as it should it returns the RESPONSE set.
RESPONSE_EXT is not used.

BLOCK_IN_SECTOR_READ

It has the same function as the BLOCK_READ but uses the different address mode for so called sector addressing where is always given the address of the sector and the sector block (as specified in the NXP documentation for Mifare Classic cards). The first sector of the Mifare Classic 1k card for example has the address 0 and the last one has 15. The block addresses of the sector are defined in the interval from 0 to 3 (3rd block of each sector is sector trailer) excluding Mifare Classic 4k cards for which in its second line of address space (the second 2k that is 32nd up to 39th sector) have the block addresses in sector 0 to 15 and the 15th is sector trailer.

Communication command protocol is the same as with BLOCK_READ with following exception:

- 1st byte of the CMD_EXT set contains block_in_sector_address
- 2nd byte of the CMD_EXT set contains sector_address
- 3rd and 4th byte of the CMD_EXT set contains dummy data

BLOCK_IN_SECTOR_WRITE

Has the same function as the BLOCK_WRITE but uses the different address mode, so called sector addressing where the sector address and the address of the block in the sector is always given (as mentioned in NXP documentation for Mifare Classic cards). For example the first sector on Mifare

Classic 1k card has the address 0 and the last one has the address 15. The block addresses in sector are in the interval from 0 to 3 (3rd block of each sector is sector trailer) excluding Mifare Classic 4k cards for which in its second line of address space (the second 2k that is 32nd up to 39th sector) have the block addresses in sector 0 to 15 and the 15th is sector trailer.
Communication command protocol is the same as with BLOCK_WRITE with following exception:
• 1st byte of CMD_EXT set contains block_in_sector_address
• 2nd byte of CMD_EXT set contains sector_address
• 3rd and 4th byte of CMD_EXT set contains dummy data

SECTOR_TRAILER_WRITE

It reads keys and access bits into the trailers of the sector. It could be used or sector address mode (without need for block_in_sector_address to be sent because the given sector is always known) either the block address mode that determines the addressing_mode u CMD_EXT set parameter which can have the following values:

BLOCK_ADDRESS_MODE = 0
SECTOR_ADDRESS_MODE = 1
Access bits are sent separately as 4 bytes that has possible values 0 up to 7.
The device Firmware is formatting the access bits according to the cards specification irreversible blocking of that sector.

The CMD_EXT set is used and its length depends on the authentication mode that is in use.
cmd_param0 contains AUTH_MODE.
Depending on AUTH_MODE, CMD and CMD_EXT set contains:
• RKA_AUTH1x:
• cmd_param1 in CMD set contains readers index key
• 1st byte of the set contains sector_(block_)address
• 2nd byte of the set contains dummy value
• 3rd byte of the set contains addressing mode
• 4th byte contains 9-byte sector trailer value (anything could be written)
• in 5th to 10th byte of the set is an unencrypted key A for writing
• in 11th to 14th byte are the access bits values for 0 to 3 blocks inside the sector respectively (for Classic 4k cards also the second half of their address space – the rest 2K of space, 11th byte of CMD_EXT set determines the access bits values for the blocks 0 to 4, the 12th byte for blocks 5 to 9 and the 13th byte for blocks 10 to 14 and at the end 14th byte for sector trailer)
• the 15th to 20th byte of the set contains an unencrypted key B for writing
• 21st byte contains checksum

• AKMy_AUTH1x:
• cmd_param1 is not used.
• 1st byte of the set contains sector_(block_)address
• 2nd byte of the set contains dummy value
• 3rd byte of the set contains addressing mode
• 4th byte contains 9-byte sector trailer value (anything could be written)
• in 5th to 10th byte of the set is an unencrypted key A for writing
• in 11th to 14th byte are the access bits values for 0 to 3 blocks inside the sector respectively (for Classic 4k cards also the second half of their address space – the rest 2K of space, 11th byte of CMD_EXT set determines the access bits values for the blocks 0 to 4, the 12th byte for blocks 5

to 9 and the 13th byte for blocks 10 to 14 and at the end 14th byte for sector trailer)
• the 15th to 20th byte of the set contains an unencrypted key B for writing
• 21st byte contains checksum

• PK_AUTH1x:
• cmd_param1 is not used.
• 1st byte of the set contains sector_(block_)address
• 2nd byte of the set contains dummy value
• 3rd byte of the set contains addressing_mode
• 4th byte contains 9-byte sector trailer value (anything could be written)
• array from 5th up to 10th byte contains 6-byte key.
• in 11th to 16th byte of the set is an unencrypted key A for writing
• in 17th to 20th byte are the access bits values for 0 to 3 blocks inside the sector respectively (for Classic 4k cards also the second half of their address space – the rest 2K of space, 11th byte of CMD_EXT set determines the access bits values for the blocks 0 to 4, the 12th byte for blocks 5 to 9 and the 13th byte for blocks 10 to 14 and at the end 14th byte for sector trailer)
• the 21st do 26th byte of the set contains an unencrypted key B for writing
• 27th byte contains checksum
If everything is done as it should it returns the RESPONSE set.
RESPONSE_EXT is not used.

SECTOR_TRAILER_WRITE_UNSAFE
It operates as SECTOR_TRAILER_WRITE except it send already formatted sector trailer block to be written without the access bits value check. The command is unsafe because it could lead to irreversible blocking of the entire sector of the card due to improperly formatted value of access bits. Made only for advanced users.
The CMD_EXT set is used and its length depends on the authentication mode that is in use.
cmd_param0 contains AUTH_MODE.

Depending on AUTH_MODE, CMD and CMD_EXT set contains:
• RKA_AUTH1x:
• cmd_param1 u CMD set contains readers index key
• 1st byte of the set contains sector_(block_)address
• 2nd byte of the set contains dummy value
• 3rd byte of the set contains addressing_mode
• 4th byte of the set contains dummy value
• in 5th to 20th byte of the set is the content of the sector trailer for writing
• 21st byte contains checksum

• AKMy_AUTH1x:
• cmd_param1 is not used.
• 1st byte of the set contains sector_(block_)address
• 2nd byte of the set contains dummy value
• 3rd byte of the set contains addressing_mode
• 4th byte of the set contains dummy value
• in 5th to 20th byte of the set is the content of the sector trailer for writing
• 21st byte contains checksum

• PK_AUTH1x:
• cmd_param1 is not used.
• 1st byte of the set contains sector_(block_)address
• 2nd byte of the set contains dummy value
• 3rd byte of the set contains addressing_mode
• 4th byte of the set contains dummy value
• array from 5th up to 10th bytes contains 6-byte key.
• in 11th to 26th byte of the set is the content of the sector trailer for writing
• 27th byte contains checksum
If everything is done as it should it returns the RESPONSE set.
RESPONSE_EXT is not used.

VALUE_BLOCK_READ
Reads the 4-byte value of the "value block" of the card which is currently in the reading field. Address mode that is used is so called block addressing where for example the first block of Mifare Classic 1k card has the address 0 and the last one has the address 63.

The CMD_EXT set is used and its length depends on the authentication mode that is used. cmd_param0 contains AUTH_MODE.

Depending on AUTH_MODE, CMD and CMD_EXT set contains:
• RKA_AUTH1x:
• cmd_param1 in CMD set contains readers index key
• 1st byte of the CMD_EXT set contains block_address
• 2nd, 3rd and 4th byte of the CMD_EXT set contains dummy data
• 5th byte contains checksum

• AKMy_AUTH1x:
• cmd_param1 is not used.
• 1st byte of the CMD_EXT set contains block_address
• 2nd, 3rd and 4th byte of the CMD_EXT set contains dummy data
• 5th byte contains checksum

• PK_AUTH1x:
• cmd_param1 is not used.
• 1st byte of the CMD_EXT set contains block_address
• 2nd, 3rd and 4th byte of the CMD_EXT set contains dummy data
• array from 5th up to 10th byte contains 6-byte key.
• 11th byte contains checksum

If everything is done as it should it returns the RESPONSE set and the RESPONSE_EXT follows with 4-byte value with checksum at the end.

response_val0 contains block address (read from block value for powerful backup as mentioned in the Mifare card documentation).
In the case of error the VALUE_BLOCK_ADDR_INVALID (read value of the value block is formatted properly but the address bytes aren't) it returns ERR_EXT set which contains the value of the value block.

VALUE_BLOCK_WRITE

Enters 4-byte value of the "value block" into the card that is currently in the reading field. Address mode that is used for so called block addressing where for example Mifare Classic 1k card has the address 0 for the first and the address 63for the last one.

This command disallow the writing into the trailers of the sector and in case of their addressing it returns the FORBIDEN_DIRECT_WRITE_IN_SECTOR_TRAILER.
The CMD_EXT set is used and its length depends on the authentication mode that is used.
cmd_param0 contains AUTH_MODE.
Depending on AUTH_MODE, CMD and CMD_EXT set contains:

- RKA_AUTH1x:
- cmd_param1 in CMD set contains readers index key
- 1st byte of the CMD_EXT set contains block_address
- 2nd and 3rd byte of the CMD_EXT set contains dummy data
- 4th byte contains value address
- in 5th to 8th byte of the set is placed the data for writing into the value block
- 9th byte contains checksum

- AKMy_AUTH1x:
- cmd_param1 is not used.
- 1st byte of the CMD_EXT set contains block_address
- 2nd and 3rd byte of the CMD_EXT set contains dummy data
- 4th byte contains value address
- in 5th to 8th byte of the set is placed the data for writing into the value block
- 9th byte contains checksum

- PK_AUTH1x:
- cmd_param1 is not used.
- 1st byte of the CMD_EXT set contains block_address
- 2nd and 3rd byte of the CMD_EXT set contains dummy data
- 4th byte contains value address
- array from 5th up to 10th byte contains 6-byte key.
- in 11th to 14th byte of the set is placed the data for writing into the value block
- 15th byte contains checksum

If everything is done as it should it returns the RESPONSE set.
RESPONSE_EXT is not used.

VALUE_BLOCK_IN_SECTOR_READ

It operates as VALUE_BLOCK_READ but uses the different address mode, so-called sector addressing where are always given the sector address and the block address in the sector (as mentioned in NXP documentation for Mifare Classic cards). For example the first sector of the Mifare Classic 1k card has the 0 and the last one has the address 15. Block addresses in the sector are in the

interval from 0 to 3 (3rd block of each sector is sector trailer) excluding Mifare Classic 4k cards for which in its second half of address space (second 2k with 32 to 39 sector) the addresses of the blocks in sector 0 to 15 and the block 15 is sector trailer.
Communication command protocol is the same as with VALUE_BLOCK_READ with following exception:

- 1st byte of the CMD_EXT set contains block_in_sector_address
- 2nd byte of the CMD_EXT set contains sector_address
- 3rd and 4th byte of the CMD_EXT set contains dummy data.

VALUE_BLOCK_IN_SECTOR_WRITE

It operates as VALUE_BLOCK_WRITE but uses different address mode, so-called sector addressing where are always given the sector address and the block address in the sector (as mentioned in NXP documentation for Mifare Classic cards). For example the first sector of the Mifare Classic 1k card has the 0 and the last one has the address 15. Block addresses in the sector are in the interval from 0 to 3 (3rd block of each sector is sector trailer) excluding Mifare Classic 4k cards for which in its second half of address space (second 2k with 32 to 39 sector) the addresses of the blocks in sector 0 to 15 and the block 15 is sector trailer.
Communication command protocol is the same as with VALUE_BLOCK_READ with following exception:

- 1st byte of the CMD_EXT set contains block_in_sector_address
- 2nd byte of the CMD_EXT set contains sector_address
- 3rd and 4th byte of the CMD_EXT set contains dummy data
- 4th byte contains value address.

VALUE_BLOCK_INC

It increases the value of the addressed value block for the 4-byte value **increment_val** that is send as a command parameter and is been used for so-called block address mode.

The CMD_EXT set is used and its length depends on the authentication mode that is used.
cmd_param0 contains AUTH_MODE.
Depending on AUTH_MODE, CMD and CMD_EXT set contains:

- RKA_AUTH1x:
- cmd_param1 in CMD set contains readers index key
- 1st byte of the CMD_EXT set contains block_address
- 2nd, 3rd and 4th byte of the CMD_EXT set contains dummy data
- in 5th to 8th byte set is **increment_val**
- 9th byte contains checksum

- AKMy_AUTH1x:
- cmd_param1 is not used.
- 1st byte of the CMD_EXT set contains block_address
- 2nd, 3rd and 4th byte of the CMD_EXT set contains dummy data
- in 5th to 8th byte set is **increment_val**
- 9th byte contains checksum

- PK_AUTH1x:
- cmd_param1 is not used.
- 1st byte of the CMD_EXT set contains block_address

- 2nd, 3rd and 4th byte of the CMD_EXT set contains dummy data
- array from 5th up to 10th byte contains 6-byte key
- in 11th to 14th bytes of the set is **increment_val**
- 15th byte contains checksum.

If everything is done as it should it returns the RESPONSE set.
RESPONSE_EXT is not used.

VALUE_BLOCK_IN_SECTOR_INC

It operates as VALUE_BLOCK_IN_SECTOR_INC but uses the different address mode, so-called sector addressing where are always given the sector address and the block address in the sector (as mentioned in NXP documentation for Mifare Classic cards). For example the first sector of the Mifare Classic 1k card has the 0 and the last one has the address 15. Block addresses in the sector are in the interval from 0 to 3 (3rd block of each sector is sector trailer) excluding Mifare Classic 4k cards for which in its second half of address space (second 2k with 32 to 39 sector) the addresses of the blocks in sector 0 to 15 and the block 15 is sector trailer.
Communication command protocol is the same as with : VALUE_BLOCK_INC with following exception:

- 1st byte of the CMD_EXT set contains block_in_sector_address
- 2nd byte of the CMD_EXT set contains sector_address
- 3rd and 4th byte of the CMD_EXT set contains dummy data.

VALUE_BLOCK_DEC

Decrement the value of the addressed value block for 4-byte value **decrement_val** which is sent as the command parameter. The so-called block address mode is used.

The CMD_EXT set is used and the length of the authentication mode is used.

cmd_param0 contains AUTH_MODE.
Depending on AUTH_MODE, CMD and CMD_EXT set contains:
- RKA_AUTH1x:
- cmd_param1 in CMD set contains readers index key
- 1st byte of the CMD_EXT set contains block_address
- 2nd, 3rd and 4th byte CMD_EXT set contains dummy data
- in 5th to 8th byte of the set is **decrement_val**
- 9th byte contains checksum

- AKMy_AUTH1x:
- cmd_param1 is not used.
- 1st byte of the CMD_EXT set contains block_address
- 2nd, 3rd and 4th byte CMD_EXT set contains dummy data
- in 5th to 8th byte of the set is **decrement_val**
- 9th byte contains checksum

- PK_AUTH1x:
- cmd_param1 is not used.
- 1st byte of the CMD_EXT set contains block_address
- 2nd, 3rd and 4th byte of the CMD_EXT set contains dummy data

- array from 5th up to 10th byte contains 6-byte key.
- in 11th to 14th byte of the set is **decrement_val**
- 15th byte contains checksum.

If everything is done as it should it returns the RESPONSE set.
RESPONSE_EXT is not used.

VALUE_BLOCK_IN_SECTOR_DEC

It operates as VALUE_BLOCK_IN_SECTOR_DEC but uses different address mode, so-called sector addressing where are always given the sector address and the block address in the sector (as mentioned in NXP documentation for Mifare Classic cards). For example the first sector of the Mifare Classic 1k card has the 0 and the last one has the address 15. Block addresses in the sector are in the interval from 0 to 3 (3rd block of each sector is sector trailer) excluding Mifare Classic 4k cards for which in its second half of address space (second 2k with 32 to 39 sector) the addresses of the blocks in sector 0 to 15 and the block 15 is sector trailer.
Communication command protocol is the same as with VALUE_BLOCK_DEC with following exception:

- 1st byte of the CMD_EXT set contains block_in_sector_address
- 2nd byte of the CMD_EXT set contains sector_address
- 3rd and 4th byte of the CMD_EXT set contains dummy data

LINEAR_FORMAT_CARD

The CMD_EXT set is used and its length depends on the authentication mode that is used.
cmd_param0 contains AUTH_MODE.
Depending on AUTH_MODE, CMD and CMD_EXT set contains:

- RKA_AUTH1x:
- cmd_param1 in CMD set contains readers index key
- 1st byte of the set contains access bits value for blocks in sector
- 2nd byte of the set contains access bits value for sector trailers
- 3rd byte of the set contains dummy value
- 4th byte of the set has 9-byte sector trailer value (anything could be written)
- in 5th to 10th byte of the set is new key A
- in 11th to 16th byte of the set is new key B
- 17th byte contains checksum

- AKMy_AUTH1x:
- cmd_param1 is not used.
- 1st byte of the set contains access bits value for blocks in sector
- 2nd byte of the set contains access bits value for sector trailers
- 3rd byte of the set contains dummy value
- 4th byte of the set has 9-byte sector trailer value (anything could be written)
- in 5th to 10th byte of the set is new key A
- in 11th to 16th byte of the set is new key B
- 17th byte contains checksum
- PK_AUTH1x:
- cmd_param1 is not used.
- 1st byte of the set contains access bits value for blocks in sector

- 2nd byte of the set contains access bits value for sector trailers
- 3rd byte of the set contains dummy value
- 4th byte of the set has 9-byte sector trailer value (anything could be written)
- array from 5th up to 10th byte contains 6-byte key for authentication (previous)
- in 11th to 16th byte of the set is new key A
- in 17th to 22nd byte of the set is new key B
- 23rd byte contains checksum

If everything is done as it should it returns the RESPONSE set.
RESPONSE_EXT is not used.

GET_CARD_ID_EX

Gives the serial number of the card which is currently in the readers field, length of serial number (4 (UID size: single), 7 (UID size: double) or 10 (UID size: triple)), and the one byte value that represents its type. For Mifare Classic 1K the type is 0x08, Mifare Classic 4k type is 0x18 and Mifare Classic Mini cards type is 0x09.
The CMD_EXT set is not in use.
The cmd_param0 and cmd_param1 are not in use.
If everything operates as expected the RESPONSE set is sent and after that also the RESPONSE EXT set of 11 bytes which contains card serial number and at the end one checksum byte.
response_val0 contains value of the card type.
response_val1 contains length of card serial number.

GET_DLOGIC_CARD_TYPE

Returns the card type whose values are in accordance with a list:

```
//DLOGIC CARD TYPE
#define DL_MIFARE_ULTRALIGHT              0x01
#define DL_MIFARE_ULTRALIGHT_EV1_11       0x02
#define DL_MIFARE_ULTRALIGHT_EV1_21       0x03
#define DL_MIFARE_ULTRALIGHT_C            0x04
#define DL_NTAG_203                       0x05
#define DL_NTAG_210                       0x06
#define DL_NTAG_212                       0x07
#define DL_NTAG_213                       0x08
#define DL_NTAG_215                       0x09
#define DL_NTAG_216                       0x0A

#define DL_MIFARE_MINI                    0x20
#define DL_MIFARE_CLASSIC_1K              0x21
#define DL_MIFARE_CLASSIC_4K              0x22
#define DL_MIFARE_PLUS_S_2K               0x23
#define DL_MIFARE_PLUS_S_4K               0x24
#define DL_MIFARE_PLUS_X_2K               0x25
#define DL_MIFARE_PLUS_X_4K               0x26
#define DL_MIFARE_DESFIRE                 0x27
#define DL_MIFARE_DESFIRE_EV1_2K          0x28
```

#define DL_MIFARE_DESFIRE_EV1_4K 0x29
#define DL_MIFARE_DESFIRE_EV1_8K 0x2A

SET_CARD_ID_SEND_CONF

Set the asynchronously card ID sending parameters.
- cmd_param0 contains send enable flag (bit 0) and prefix enable flag (bit 1)
- 1st byte of the CMD_EXT contains prefix character
- 2nd byte of the CMD_EXT contains suffix character
- array from 3rd byte up to 6th byte of the CMD_EXT contains baud rate value
- 7th byte of the CMD_EXT contains internal crc (xor of bytes cmd_param0 to 6th byte + 7)
- 8th byte of the CMD_EXT contains checksum

If everything is done as it should it returns the RESPONSE set.
RESPONSE_EXT is not used.

GET_CARD_ID_SEND_CONF

Get the asynchronously card ID sending parameters.
The CMD_EXT set is not in use.
The cmd_param0 and cmd_param1 are not in use.
If everything operates as expected the RESPONSE set is sent and after that also the RESPONSE EXT set of 9 bytes.
response_val0 and response_val1 are not in use.
- 1st byte of the RESPONSE_EXT contains send enable flag (bit 0) and prefix enable flag (bit 1)
- 2nd byte of the RESPONSE_EXT contains prefix character
- 3rd byte of the RESPONSE_EXT contains suffix character
- array from 4th byte up to 7th byte of the RESPONSE_EXT contains baud rate value
- 8th byte of the RESPONSE_EXT contains internal crc
- 9th byte of the RESPONSE_EXT contains checksum