UFR Series MIFARE® card reader - Win API (Application Programming Interface)

uFR series of readers operates with Mifare® Classic contactless card series which communication interface is compliant to ISO / IEC 14443 A standard.

Since the MIFARE® cards have a lot of specifics that are not supported by any standard Windows API, IS21 Series MIFARE® card readers are supplied with separate API interface placed in the DLL called is21.dll.

Alternatively this reader can communicate over a virtual COM port using appropriate available IS21-VCOM protocol. In this way, IS21 readers can be used on any platform for which there is still no direct software support (Mac OS X, Linux, Linux x86_64, Windows CE (4.2, 5.2, 6.0 for a range of processors), Windows Mobile (version 5 and 6 for x86), Pocket PC 2003 (x86 and ARM / XScale processor)).

API specification (applies to the IS21-VCOM protocol) contains functions that:

● emulate linear address space on the MIFARE® cards,
● directly addressing blocks on the MIFARE® card - the block address mode,
● indirectly addressing blocks on the MIFARE® cards, combining sector and blocks addresses within the sector - the sector address mode.

This way of data addressing is performed in accordance with the manufacturer's documentation for addressing the MIFARE® card.

Allows four methods of authentication for card data access:

● "**Reader key authentication" -** the default authentication mode. For this mode the keys are stored into the reader (with a maximum of 32 key with indexes from 0 to 31) and the key index is sent with related functions. In the case of functions that emulate linear address space in this method of authentication, the use of the same key for all sectors (or at least for those who are in default range for linear addressing) is default.

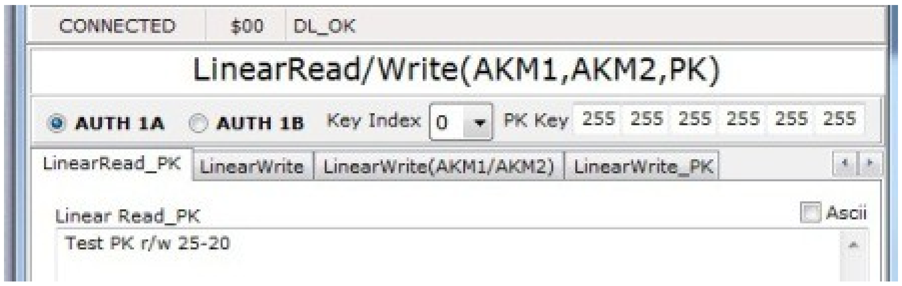
● "**Automatic key mode 1" (AKM1) and "Automatic key mode 2" (AKM2)** are optional, automatic modes of authentication. This modes enable automatic selection of keys stored in the reader on the basis of the block address or a combination of block and sector address within the sector. These modes could be used in emulation of a linear address space because after the address conversion in the readers software it is performed automatic keys selection for the authentication of a block or sector. The difference between AKM1 and AKM2 is only in the way of automatic selection of A and B keys performing.

When using **AKM1** mode it is accepted that the index keys in the reader from 0 to 15 are appropriate with A sectors keys from 0 to 15 and the index keys in the reader from 16 to 31 are appropriate with B sectors keys from 0 to 15.

When used **AKM2** mode, even key indexes in the reader (0, 2, ..., 28, 30) are accepted as a sectors keys from 0 to 15 respectively and the odd key indexes in the reader are accepted as B sectors keys from 0 to 15. This is certainly true for Mifare ® 1K.

For MIFARE ® card MINI only the first five keys A and B can be used (in AKM1 key index from 0 to 4 for A keys and from 16 to 20 for B keys, in AKM2 mode for A key index 0, 2, 4, 6 and 8 and for B keys 1, 3, 5, 7 and 9) because this cards contain only that much sectors.

On MIFARE® 4K there are 40 sectors so the lower and upper address space are organized into the 2K. With these cards AKM1 and AKM2 modes are organized in such a way that the same keys indexes from the reader corresponding sectors 0 to 15 and 16 to 32. For the last 8 sectors (sectors 32 to 39) the same readers keys are used that correspond to sectors 0 to 7 and 16 to 23.


CONNECTED $00 DL_OK

LinearRead/Write(AKM1,AKM2,PK)

AUTH 1A (selected), AUTH 1B, Key Index: 0, PK Key: 255 255 255 255 255 255

Linear Read_PK

Test PK r/w 25-20

The last method of authentication is **"Provided key" (PK)**. In this mode, you do not use keys stored in the reader but the keys are sent directly from the code with API functions. This mode does not provide any security, so its use is not recommended except under strictly controlled conditions or for testing purposes.

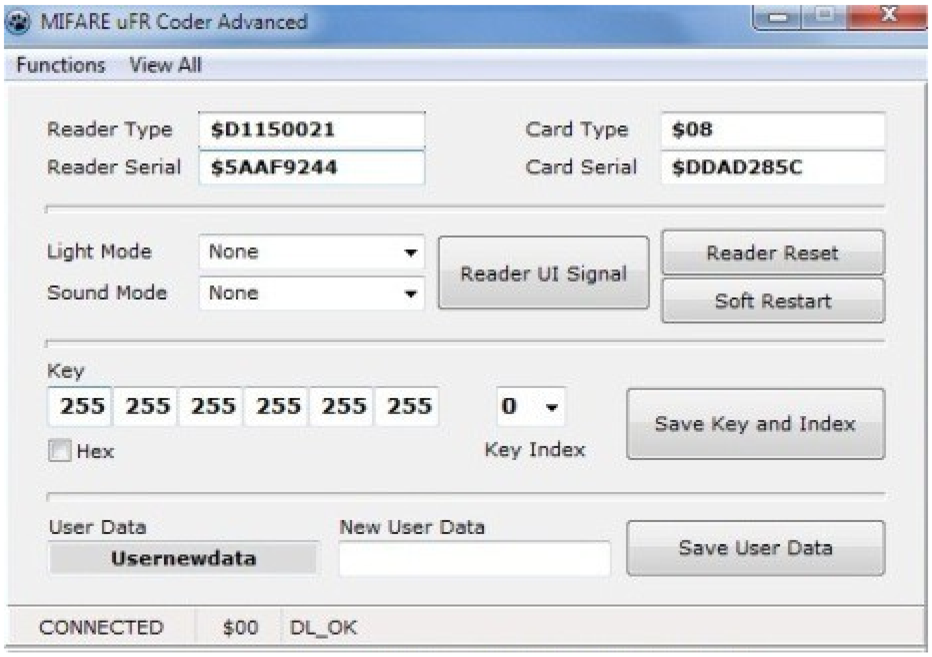
A special function for the sector trailer blocks card entry is implemented which is a very simplified calculation of the bytes values containing the access bits. This avoids the danger of permanently blocking the entire sectors of the card due to wrong bits format which controls access to blocks of a sector.

For those with more experience in working with MIFARE cards, the so-called unsafe option is left for the sector trailer blocks manipulation.

● There is a method for linear emulation mode, which formats the card sector trailer blocks in the same way ie. sets a unique keys and access bits for the entire card. This is a very simplified way of the card initialization for linear approach.

● API contains a set of functions for manipulating the cards value blocks. Four-byte values read and write are supported with a value blocks automatically formatted for the appropriate specification. The increment and decrement blocks value is also supported.

General functions for working with the reader



Reader Type: $D1150021
Reader Serial: $5AAF9244
Card Type: $08
Card Serial: $DDAD285C
Light Mode: None
Sound Mode: None
Key: 255 255 255 255 255 255
Hex
Key Index: 0
User Data: Usernewdata
New User Data:
CONNECTED $00 DL_OK

ReaderOpen: Opens a port of connected reader. In the case of multi-thread applications, developers must be careful to synchronize access to readers resources to avoid unforeseen situations.

GetReaderType: Returns the device type identifier. On IS21 readers this value is 0xD1150021.

GetReaderSerialNumber: Returns the device serial number.

ReaderSoftRestart: Reader is restarted by software. This function sets all readers operating parameters to the default values and resets the close RF field, which practically resets all the cards in the field.

ReaderReset: Resets all the digital logic of readers hardware. This function can generally caled in the event that ReaderSoftRestart did not gave the desired results.

ReaderClose: Closes readers port. This enables access to the reader from other processes.

ReaderKeyWrite: Sets the keys for authentication to the reader when reading and manipulating data on the cards. The keys are entered in a special readers area in EEPROM that can not be read anymore which provided protection against unauthorized access.

Function declaration (C language):

```
unsigned long ReaderKeyWrite(const unsigned char *aucKey,
                             unsigned char ucKeyIndex);
```

35/17 *aucKey Pointer* to an array of 6 bytes containing the key. Key bytes can have any value in the range 0 to 255. The transport keys on the new cards should have all the bits degrades gracefully (all key bytes have a value of 255)

- *ucKeyIndex Index* in the reader where the user intends to store the new key. Possible values are 0 to 31.

ReaderUISignal: The function is used to control the reader light and sound signal. There are four modes of light signals and five sound modes.

Function declaration (C language):

```
unsigned long ReaderUISignal(unsigned char ucLightSignalMode,
                             unsigned char ucBeepSignalMode);
```

- ucLightSignalMode Defines the light signals mode. It can have values from 0 to 4. A value of 0 indicates light signals inactivity.
- ucBeepSignalMode Defines the sound signals mode. It can have values from 0 to 5. A value of 0 indicates sound signals inactivity.

Functions for working with cards

By type of data they work with, the functions are classified in:

- Functions for manipulating card **data blocks**
- Functions for manipulating card **value blocks.**

According to the card data addressing method, this function are divided into:

- Functions that **emulate the linear address space**
- Functions that use **bloc addressing**
- Functions that use **sector addressing**

Functions for cards data manipulating are sorted according to the authentication method into the function sets recognizable by the suffix of the authentication method:

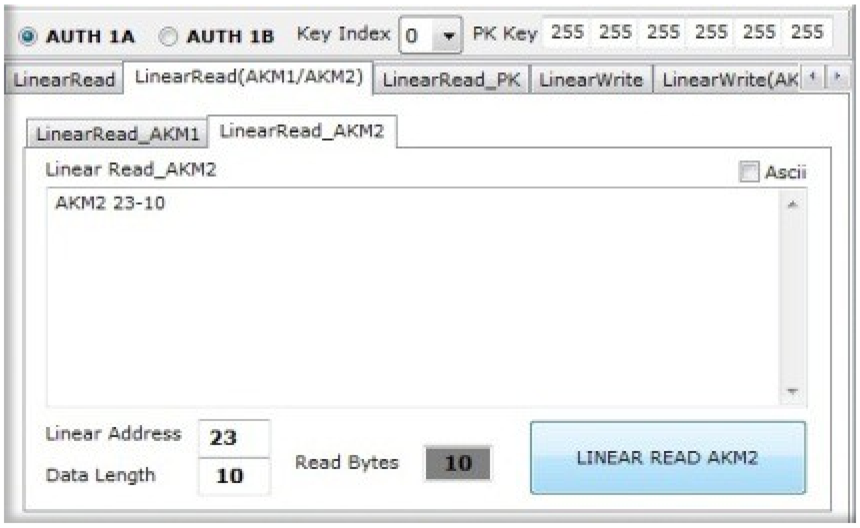
- *"Reader key authentication"* is the default authentication method so function of this group do not have any suffix
- Functions with the **_AKM1** suffix use *"Automatic key mode 1"*
- Functions with the **_AKM2** suffix use "Automatic key mode 2"

Functions with the **_PK** suffix use *"Provided key"* method.

General functions for working with cards

GetCardId: This function returns the type identifier and card serial number placed into the reader. Reader supports only cards that have 4 byte serial number (UID size: single) according to the standard ISO / IEC 14443 A.

Functions that emulate the linear address space



Key Index: 0
PK Key: 255 255 255 255 255 255

Linear Read_AKM2

AKM2 23-10

Linear Address: 23
Data Length: 10
Read Bytes: 10

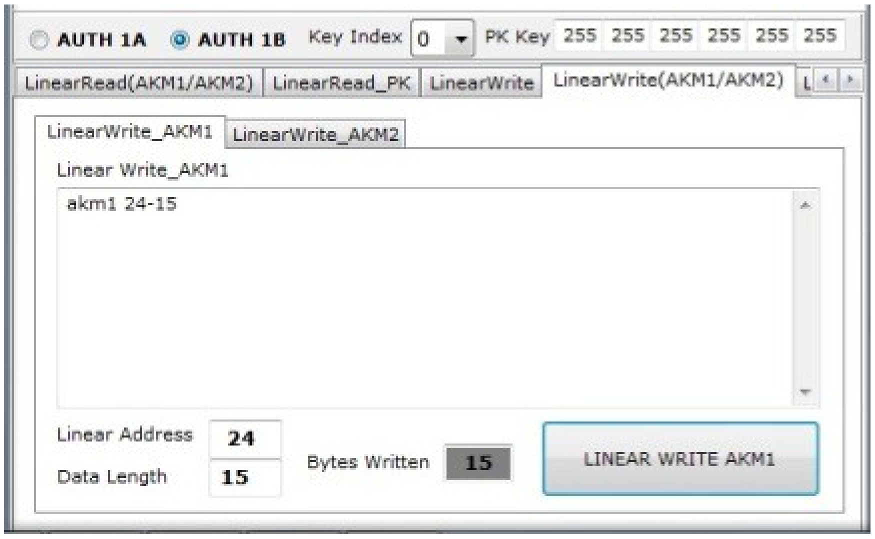
- ***LinearRead***
- ***LinearRead_AKM1***
- ***LinearRead_AKM2***
- ***LinearRead_PK***

Functions declaration (C language):

```
unsigned long LinearRead(unsigned char *aucData,
                 unsigned short usLinearAddress,
                 unsigned short usDataLength,
                 unsigned short *lpusBytesReturned,
                 unsigned char ucAuthMode,
                 unsigned char ucReaderKeyIndex);
unsigned long LinearRead_AKM1(unsigned char *aucData,
                          unsigned short usLinearAddress,
                          unsigned short usDataLength,
                          unsigned short *lpusBytesReturned,
                          unsigned char ucAuthMode);
unsigned long LinearRead_AKM2(unsigned char *aucData,
                          unsigned short usLinearAddress,
                          unsigned short usDataLength,
                          unsigned short *lpusBytesReturned,
                          unsigned char ucAuthMode);
unsigned long LinearRead_PK(unsigned char *aucData,
                        unsigned short usLinearAddress,
                        unsigned short usDataLength,
                        unsigned short *lpusBytesReturned,
                        unsigned char ucAuthMode,
                        unsigned char *aucProvidedKey);
```

These functions are used for card data reading by using the linear address space emulation. The method for proving authenticity is determined by the suffix in the functions names:

- *aucData* - Pointer to the sequence of bytes where read data will be stored.
- *usLinearAddress* - Linear address on the card from which the data want to read
- *usDataLength* - Number of bytes for reading. For aucData a minimum usDataLength bytes must be allocated before calling the function
- *lpusBytesReturned* - Pointer to "unsigned short" type variable, where the number of successfully read bytes from the card is written. If the reading is fully managed this data is equal to the usDataLength parameter. If there is an error reading some of the blocks, the function returns all successfully read data in the aucData before the errors occurrence and the number of successfully read bytes is returned via this parameter
- *ucAuthMode* - This parameter defines whether to perform authentication with key A or key B. It can have two values, namely: AUTHENT1A (0x60) or AUTHENT1B (0x61).
- ucReaderKeyIndex - The default method of authentication (when the functions without a suffix is used) performs the authenticity proving by using the selected key index from the reader. In the linear address mode, this applies to all sectors that are read
- *aucProvidedKey* - Pointer to the six-byte string containing the key for authenticity proving in the "Provided Key" method. _PK Suffix in the name of the function indicates this method usage.


AUTH 1A [✓ AUTH 1B] Key Index 0 PK Key 255 255 255 255 255 255

Linear Write_AKM1

akm1 24-15

Linear Address 24
Data Length 15
Bytes Written 15

- ***LinearWrite,***
- ***LinearWrite_AKM1,***

- ***LinearWrite_AKM2,***
- ***LinearWrite_PK***

Functions declaration (C language):

```
unsigned long LinearWrite(const unsigned char *aucData,
                          unsigned short usLinearAddress,
                          unsigned short usDataLength,
                          unsigned short *lpusBytesWritten,
                          unsigned char ucAuthMode,
                          unsigned char ucReaderKeyIndex);

unsigned long LinearWrite_AKM1(const unsigned char *aucData,
                               unsigned short usLinearAddress,
                               unsigned short usDataLength,
                               unsigned short *lpusBytesWritten,
                               unsigned char ucAuthKey);

unsigned long LinearWrite_AKM2(const unsigned char *aucData,
                               unsigned short usLinearAddress,
                               unsigned short usDataLength,
                               unsigned short *lpusBytesWritten,
                               unsigned char ucAuthKey);

unsigned long LinearWrite_PK(const unsigned char *aucData,
                             unsigned short usLinearAddress,
                             unsigned short usDataLength,
                             unsigned short *lpusBytesWritten,
                             unsigned char ucAuthKey,
                             unsigned char *aucProvidedKey);
```

These functions are for writing data to the card using the emulation of linear address space. The method for proving authenticity is determined by the suffix in the functions names:

- *aucData* - Pointer to the sequence of bytes containing data for writing on the card
- *usLinearAddress* - Linear address of the card where the data writing is intend
- *usDataLength* - - Number of bytes for the entry. In aucData a minimum usDataLength bytes must be allocated before calling the function
- *lpusBytesWritten* - Pointer to a "unsigned short" type variable, where the number of successfully read bytes from the card is written. If the entry is a successfully completed this data is equal to the usDataLength parameter. If there was an error in writing some of the blocks, the function returns the number of successfully written bytes over this parameter

- *ucAuthKey* - This parameter defines whether to perform authentication with A key or key B. It can have two values, namely: AUTHENT1A (0x60) or AUTHENT1B (0x61)
- *ucReaderKeyIndex* - The default method of authentication (when the functions without a suffix is used) performs the authenticity proving by using the selected key index from the reader. In the linear address mode, this applies to all sectors that are written
- aucProvidedKey - Pointer to the sixth byte string containing the key for authenticity proving in the "Provided Key" method. _PK Suffix in the name of the function indicates this method usage.

- **LinearFormatCard,**
- **LinearFormatCard_AKM1,**
- **LinearFormatCard_AKM2,**
- **LinearFormatCard_PK**

Functions description:

These functions are used for new keys A and B writing as well as access bits in the trailers of all card sectors. The setting of ninth trailers bytes is enabled (a general-purpose byte where any value can be entered). In all the card sector trailers the same value is set for the entire card so the same keys and access rights are valid. As it is necessary to prove the authenticity on the base of previous keys before writing into the sector trailers, these functions are potentially suitable to initialize the new card (the authentication is performed with transportation keys, all the key bytes are 0xFF) or to re-initialize the card with the same keys and access rights for all sectors. Certainly, there must always be careful about the previously set access rights (access bits) on the cards in case the changing of some keys or bits for access rights control is disabled.

Function declaration (C language):

```
unsigned long LinearFormatCard(const unsigned char *aucNewKeyA,
                               unsigned char ucBlocksAccessBits,
                               unsigned char ucSectorTrailersAccessBits,
                               unsigned char ucSectorTrailersByte9,
                               const unsigned char *aucNewKeyB,
                               unsigned char *lpucSectorsFormatted,
                               unsigned char ucAuthMode,
                               unsigned char ucReaderKeyIndex);
unsigned long
LinearFormatCard_AKM1(const unsigned char *aucNewKeyA,
                      unsigned char ucBlocksAccessBits,
                      unsigned char ucSectorTrailersAccessBits,
                      unsigned char ucSectorTrailersByte9,
                      const unsigned char *aucNewKeyB,
                      unsigned char *lpucSectorsFormatted,
                      unsigned char ucAuthMode);
```

```
unaigned long
LinearFormatCard_AKM2(const unsigned char *aucNewKeyA,
                      unsigned char ucBlocksAccessBits,
                      unsigned char ucSectorTrailersAccessBits,
                      unsigned char ucSectorTrailersByte9,
                      const unsigned char *aucNewKeyB,
                      unsigned char *lpucSectorsFormatted,
                      unsigned char ucAuthMode);
```

```
 unsigned long
 LinearFormatCard(const unsigned char *aucNewKeyA,
                  unsigned char ucBlocksAccessBits,
                  unsigned char ucSectorTrailersAccessBits,
                  unsigned char ucSectorTrailersByte9,
                  const unsigned char *aucNewKeyB,
                  unsigned char *lpucSectorsFormatted,
                  unsigned char ucAuthMode,
                  unsigned char *aucProvidedKey);
```

Greater flexibility in sector trailers initiating is offered by **SectorTrailerWrite** functions group :

- *aucNewKeyA* - Pointer on 6 bytes array containing a new A key
- *ucBlocksAccessBits* - The access bits values that define permissions for all data blocks on the card. It can have values 0 to 7
- *ucSectorTrailersAccessBits* - The access bits value that define access permissions for all the card sector trailers. It can have values 0 to 7
- *ucSectorTrailersByte9* - The ninth byte value of all card sectors trailers. It can contain any value
- *aucNewKeyB* - *P*ointer on 6 bytes array containing a new B key
- *lpucSectorsFormatted* - Pointer to a "unsigned char" type variable through which the number of successfully formatted sectors trailers returns. Eg. if all the sectors trailers are successfully initialized, on the MIFARE® 1K, through this parameter it returns the value 16 which represents the number of sectors on this card. In case of error the parameter is an indication of the number of successfully initialized sectors starting from zero.

- *ucAuthMode* - This parameter defines whether to perform authentication A key or B key. It can have two values, namely: AUTHENT1A (0x60) or AUTHENT1B (0x61).
- *ucReaderKeyIndex* - The default method of authentication (when the functions without a suffix is used) performs the authenticity proving by using the selected key index from the reader. In the linear address mode, this applies to all sectors that are written

aucProvidedKey - Pointer to the sixth byte string containing the key for authenticity proving in the "Provided Key" method. _PK Suffix in the name of the function indicates this method usage.

Functions for working with data blocks

- **BlockRead,**
- **BlockRead_AKM1,**
- **BlockRead_AKM2,**
- **BlockRead_PK,**

Functions description:

This functions group is used for card block content reading. Always reads the entire block (16 bytes of the block). Functions use the so-called bloc addressing (the first card block has the address 0; first sector trailer has address 3, the next one 7, etc. until the last Mifare ® 1K block which is also a trailer of the last sector, has an address 63). These functions also allows reading of the sector trailers contents (its available part for reading, depending on the access rights set).

Functions declaration (C language):

```
unsigned long BlockRead(unsigned char *aucData,
                        unsigned char ucBlockAddress,
                        unsigned char ucAuthMode,
                        unsigned char ucKeyIndex);

unsigned long BlockRead_AKM1(unsigned char *aucData,
                             unsigned char ucBlockAddress,
                             unsigned char ucAuthMode);

unsigned long BlockRead_AKM2(unsigned char *aucData,
                             unsigned char ucBlockAddress,
                             unsigned char ucAuthMode);

 unsigned long BlockRead_PK(unsigned char *aucData,
                            unsigned char block_address,
                            unsigned char ucAuthMode,
                            const unsigned char *aucProvidedKey);
```

- *aucData* - Pointer to the number of bytes where read data will be stored. Must be allocated at least 16 bytes before calling the function.
- *ucBlockAddress* - ucAuthMode block address. This parameter defines whether to perform authentication A key or B key. It can have two values, namely: AUTHENT1A (0x60) or AUTHENT1B (0x61).
- *ucReaderKeyIndex* - The default method of authentication (when the functions without a suffix is used) performs the authenticity proving by using the selected key index from the reader. In the linear address mode, this applies to all sectors that are written
- *aucProvidedKey* - Pointer to the sixth byte array containing the key for authenticity proving in the "Provided Key" method. _PK Suffix in the name of the function indicates this method usage.

These functions work the same as BlockRead group functions and are made for card block content reading. The only difference is that the sectoral addressing is used. That includes separately sending sector addresses and block addresses within a sector. For MIFARE® 1K card sector address may be in the range 0 to 15, and blocks address within the sector ranging from 0 to 3. For MIFARE ® 4k sector address may be in the range of 0 to 39 and since the second half of the address space organization is different (above 2 MB) blocks address in the last 8 sectors (sectors 32 to 39) may be in the range of 0 to 15. The entire block (16-byte block) is always read.

These functions can read the sector trailers contents (its available part for reading, depending on the access rights set).

- *aucData* - Pointer to the bytes array where read data are going to be stored. At least 16 bytes must be allocated before the function is called
- *ucSectorAddress* - Sector Address
- *ucBlockInSectorAddress* - Block address within a sector
- *ucAuthMode* - This parameter defines whether to perform authentication with A key or B key. It can have two values, namely: AUTHENT1A (0x60) or AUTHENT1B (0x61).
- *ucReaderKeyIndex* - The default method of authentication (when the functions without a suffix is used) performs the authenticity proving by using the selected key index from the reader. In the linear address mode, this applies to all sectors that are written
- *aucProvidedKey* - Pointer to the sixth byte array containing the key for authenticity proving in the "Provided Key" method. _PK Suffix in the name of the function indicates this method usage.
- **BlockWrite,**
- **BlockWrite_AKM1,**
- **BlockWrite_AKM2,**
- **BlockWrite_PK**

Functions description:

These functions are used for data entry (16 bytes at a time) into the card blocks. Functions use the so-called bloc addressing (the first card block has the address 0; first sector trailer has address 3, the next one 7, etc. until the last Mifare ® 1K block which is also a trailer of the last sector, has an address 63). This functions group don't allow direct data enter into the sector trailers. To do so, use the special functions SectorTrailerWrite and SectorTrailerWriteUnsafe.

Functions declaration (C language):

```
unsigned long BlockWrite(const unsigned char *aucData,
                     unsigned char ucBlockAddress,
                     unsigned char ucAuthMode,
                     unsigned char ucKeyIndex);

unsigned long BlockWrite_AKM1(const unsigned char *aucData,
                          unsigned char ucBlockAddress,
                          unsigned char ucAuthMode);

unsigned long BlockWrite_AKM2(const unsigned char *aucData,
                          unsigned char ucBlockAddress,
                          unsigned char ucAuthMode);

unsigned long BlockWrite_PK(const unsigned char *aucData,
                        unsigned char ucBlockAddress,
                        unsigned char ucAuthMode,
                        const unsigned char *aucProvidedKey);
```

- *aucData* - Pointer to the number of bytes where read data will be stored. Must be allocated at least 16 bytes before calling the function
- *ucBlockAddress* - Cards block address
- *ucAuthMode* - This parameter defines whether to perform authentication with key A or key B. It can have two values, namely: AUTHENT1A (0x60) or AUTHENT1B (0x61).
- *ucReaderKeyIndex* - The default method of authentication (when the functions without a suffix is used) performs the authenticity proving by using the selected key index from the reader. In the linear address mode, this applies to all sectors that are read
- *aucProvidedKey* - Pointer to the sixth byte array containing the key for authenticity proving in the "Provided Key" method. _PK Suffix in the name of the function indicates this method usage.

FORBIDEN_DIRECT_WRITE_IN_SECTOR_TRAILER.

- **BlockInSectorWrite,**
- **BlockInSectorWrite_AKM1,**
- **BlockInSectorWrite_AKM2,**
- **BlockInSectorWrite_PK**

Function description:

These functions work the same as BlockWrite group functions, they are used for data entry (16 bytes at a time) into card blocks. The only difference is the use of sector addressing. Sector addressing means separate sending sector and block addresses within a sector. For MIFARE® 1K card sector address may be in the range 0 to 15, and blocks address within the sector ranging from 0 to 3. For MIFARE ® 4k sector address may be in the range of 0 to 39 and since the second half of the address space organization is different (above 2 MB) blocks address in the last 8 sectors (sectors 32 to 39) may be in the range of 0 to 15. This functions group don't allow direct data enter into the sector trailers. To do so, use the special functions SectorTrailerWrite and SectorTrailerWriteUnsafe

Function declaration (C language)

```
unsigned long BlockWrite(const unsigned char *aucData,
                     unsigned char ucSectorAddress,
                     unsigned char ucBlockInSectorAddress,
                     unsigned char ucAuthMode,
                     unsigned char ucKeyIndex);

unsigned long BlockWrite_AKM1(const unsigned char *aucData,
                          unsigned char ucSectorAddress,
                          unsigned char ucBlockInSectorAddress,
                          unsigned char ucAuthMode);

unsigned long BlockWrite_AKM2(const unsigned char *aucData,
                          unsigned char ucSectorAddress,
                          unsigned char ucBlockInSectorAddress,
                          unsigned char ucAuthMode);

unsigned long BlockWrite_PK(const unsigned char *aucData,
                        unsigned char ucSectorAddress,
                        unsigned char ucBlockInSectorAddress,
                        unsigned char ucAuthMode,
                        const unsigned char *aucProvidedKey);
```

- *aucData* - Pointer to the number of bytes where read data will be stored. Must be allocated at least 16 bytes before calling the function
- *ucSectorAddress* - Sector address
- *ucBlockInSectorAddress* - Block address in the sector
- *ucAuthMode* - This parameter defines whether to perform authentication with A key or B key. It can have two values, namely: AUTHENT1A (0x60) or AUTHENT1B (0x61)
- *ucReaderKeyIndex* - The default method of authentication (when the functions without a suffix is used) performs the authenticity proving by using the selected key index from the reader. In the linear address mode, this applies to all sectors that are written

- *aucProvidedKey* - Pointer to the sixth byte array containing the key for authenticity proving in the "Provided Key" method. _PK Suffix in the name of the function indicates this method usage.

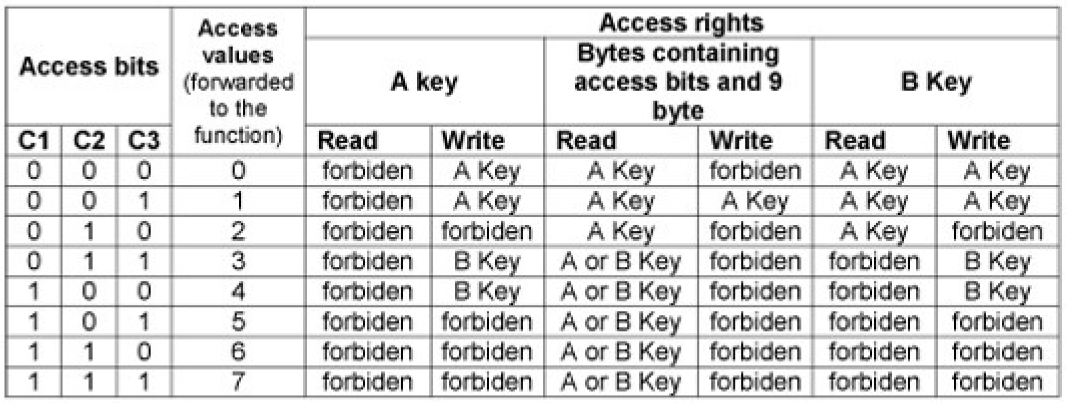
FORBIDEN_DIRECT_WRITE_IN_SECTOR_TRAILER.

- **SectorTrailerWrite,**
- **SectorTrailerWrite_AKM1,**
- **SectorTrailerWrite_AKM2,**
- **SectorTrailerWrite_PK**

Functions description:

These functions are used for data writing in the card sector trailers. Functions can also be used for sector trailers block addressing as well as for the sector addressing which is determined by the ucAddressingMode parameter. In the case of block addressing, the first card block has the address 0; trailer has a first sector address 3 and the next 7, etc. until the last block of Mifare® 1k which is also a trailer of the last sector and has an address 63. This group of functions simplifies the bits manipulation for blocks access rights setting (access bits) and minimizes the possibility of permanent blocking of the whole sector due to incorrect formatting of these bits. Formatting the access bits is made by the reader before the writing. API users can choose the appropriate blocks access rights which are represented by values 0 to 7 and to transmit them to these functions.

For sector trailers the following access rights are valid:



Access bits			Access values (forwarded to the function)	Access rights					
				A key		Bytes containing access bits and 9 byte		B Key	
C1	C2	C3		Read	Write	Read	Write	Read	Write
0	0	0	0	forbiden	A Key	A Key	forbiden	A Key	A Key
0	0	1	1	forbiden	A Key	A Key	A Key	A Key	A Key
0	1	0	2	forbiden	forbiden	A Key	forbiden	A Key	forbiden
0	1	1	3	forbiden	B Key	A or B Key	forbiden	forbiden	B Key
1	0	0	4	forbiden	B Key	A or B Key	forbiden	forbiden	B Key
1	0	1	5	forbiden	forbiden	A or B Key	forbiden	forbiden	forbiden
1	1	0	6	forbiden	forbiden	A or B Key	forbiden	forbiden	forbiden
1	1	1	7	forbiden	forbiden	A or B Key	forbiden	forbiden	forbiden

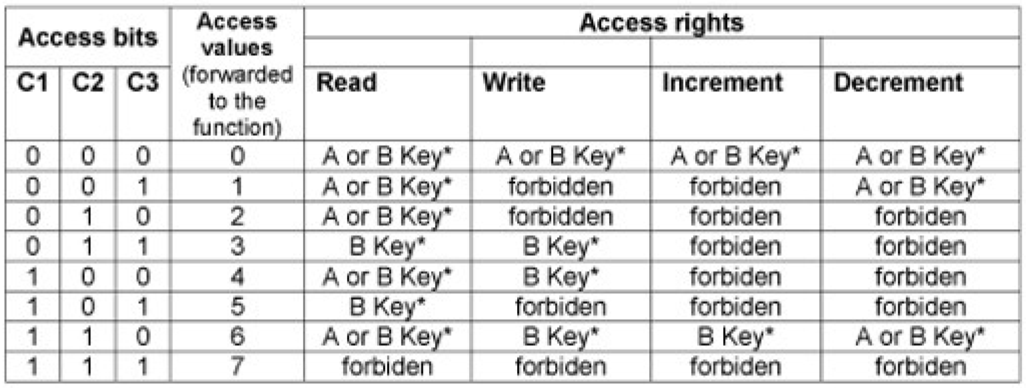
Table 1: Access rights for the sector trailers

For sector trailers following access rights are valid:

- Access bits C1 C2 C3
- Access values (submitted to the function)

- Access rights
- Key A bytes containing the access bits and the nine byte key B
- Reading and writing

For blocks the following access rights are valid:



Access bits			Access values (forwarded to the function)	Access rights			
C1	C2	C3		Read	Write	Increment	Decrement
0	0	0	0	A or B Key*	A or B Key*	A or B Key*	A or B Key*
0	0	1	1	A or B Key*	forbidden	forbiden	A or B Key*
0	1	0	2	A or B Key*	forbidden	forbiden	forbiden
0	1	1	3	B Key*	B Key*	forbiden	forbiden
1	0	0	4	A or B Key*	B Key*	forbiden	forbiden
1	0	1	5	B Key*	forbiden	forbiden	forbiden
1	1	0	6	A or B Key*	B Key*	B Key*	A or B Key*
1	1	1	7	forbiden	forbiden	forbiden	forbiden

Table 2: Access rights for the blosks

*) If the access rights for the sector trailer of an appropriate sector set up so that it is possible to readB Key, it can not be used for authentication in any of the cases. These functions also sets new sector keys if it is permitted to access rights.

For blocks the fallowing access rights are valid:

- Access bits C1 C2 C3
- Access values (submitted to the function)
- Access rights
- Reading, writing, increment, decrement

Functions declaration (C language):

```
unsigned long
SectorTrailerWrite(unsigned char ucAddressingMode,
                   unsigned char ucAddress,
                   const unsigned char *aucNewKeyA,
                   unsigned char ucBlock0AccessBits,
                   unsigned char ucBlock1AccessBits,
                   unsigned char ucBlock2AccessBits,
                   unsigned char ucSectorTrailerAccessBits,
                   unsigned char ucSectorTrailerByte9,
                   const unsigned char *aucNewKeyB,
                   unsigned char ucAauthMode,
                   unsigned char ucKeyIndex);

unsigned long
SectorTrailerWrite_AKM1(unsigned char ucAddressingMode,
                        unsigned char ucAddress,
                        const unsigned char *aucNewKeyA,
                        unsigned char ucBlock0AccessBits,
                        unsigned char ucBlock1AccessBits,
                        unsigned char ucBlock2AccessBits,
                        unsigned char ucSectorTrailerAccessBits,
                        unsigned char ucSectorTrailerByte9,
                        const unsigned char *aucNewKeyB,
                        unsigned char ucAauthMode);

unsigned long
SectorTrailerWrite_AKM2(unsigned char ucAddressingMode,
                        unsigned char ucAddress,
                        const unsigned char *aucNewKeyA,
                        unsigned char ucBlock0AccessBits,
                        unsigned char ucBlock1AccessBits,
                        unsigned char ucBlock2AccessBits,
                        unsigned char ucSectorTrailerAccessBits,
                        unsigned char ucSectorTrailerByte9,
                        const unsigned char *aucNewKeyB,
                        unsigned char ucAauthMode);
unsigned long
SectorTrailerWrite_PK(unsigned char ucAddressingMode,
                      unsigned char ucAddress,
                      const unsigned char *aucNewKeyA,
                      unsigned char ucBlock0AccessBits,
                      unsigned char ucBlock1AccessBits,
                      unsigned char ucBlock2AccessBits,
                      unsigned char ucSectorTrailerAccessBits,
                      unsigned char ucSectorTrailerByte9,
                      const unsigned char *aucNewKeyB,
                      unsigned char ucAauthMode,,
                      const unsigned char *aucProvidedKey);
```

- *ucAddressingMode* - Specifies the address mode. Possible values of this parameter are BLOCK_ADDRESS_MODE (0x00) or SECTOR_ADDRESS_MODE (0x01). If any other value is sent the function returns an error code WRONG_ADDRESS_MODE
- *ucAddress* - Sectors or sector trailers blocks address, depending on ucAddressingMode. When using a sector-address mode, then, for instance, the MIFARE Classic 1K card, the range can be from 0 to 15 (this card has 16 sectors). The same card type in the block addressing mode can use the values from 0 to 63 provided that an error occurs if the addressed block is not also the sector trailer.
- *aucNewKeyA* - Pointer to the 6 byte array that represents a new A key for a specified sector which will be set if that is previously allowed with the access rights
- *aucNewKeyB* - Pointer to the six-byte array that represents a new B key for a specified sector which will be set if that is previously allowed with the access rights
- *ucBlock0AccessBits* - Access value for the 0 block of a sector.

MIFARE ® 4k has a different organization for the last 8 sectors, the second half of the address space. Therefore, in these sectors the access rights are set as follows:

- access rights to the first 5 blocks - ucBlock1AccessBits Access value block for the first sector
- prava pristupa drugih 5 blokova - ucBlock2AccessBits Access value block for the first sector
- access rights to the last 5 blocks:
    -
    - *ucSectorTrailerAccessBits* - Access value for a sector trailer
    - *ucSectorTrailerByte9* - The ninth sector trailers byte is a byte for general purpose where any single-byte value can be entered

- *ucAuthMode* - This parameter defines whether to perform authentication with key A or key B. It can have two values, namely: AUTHENT1A (0x60) or AUTHENT1B (0x61)
- *ucReaderKeyIndex* - The default method of authentication (when the functions without a suffix is used) performs the authenticity proving by using the selected key index from the reader. In the linear address mode, this applies to all sectors that are written
- *aucProvidedKey* - Pointer to the sixth byte array containing the key for authenticity proving in the "Provided Key" method. _PK Suffix in the name of the function indicates this method usage.

- **SectorTrailerWriteUnsafe,**
- **SectorTrailerWriteUnsafe_AKM1,**
- **SectorTrailerWriteUnsafe_AKM2,**
- **SectorTrailerWriteUnsafe_PK**

Functions description:

These functions have the same purpose as the function of the SectorTrailerWrite group with the difference in sending the "raw" sector trailers content and the errors are possible while formatting access bits values for entering. These functions are intended for developers with experience in working with MIFARE cards. All rules mentioned for the SectorTrailerWrite group functions applies to these functions, except the option of the "raw" data for sector trailer entry.

Functions declaration (C language):

```
unsigned long
SectorTrailerWriteUnsafe(unsigned char ucAddressingMode,
                         unsigned char ucAddress,
                         const unsigned char *aucSectorTrailer,
                         unsigned char ucAauthMode,
                         unsigned char key_index);

unsigned long
SectorTrailerWriteUnsafe_AKM1(unsigned char ucAddressingMode,
                              unsigned char ucAddress,
                          const unsigned char *aucSectorTrailer,
                              unsigned char ucAauthMode);

nsigned long
SectorTrailerWriteUnsafe_AKM2(unsigned char ucAddressingMode,
                              unsigned char ucAddress,
                       const unsigned char *aucSectorTrailer,
                              unsigned char ucAauthMode);
unsigned long
SectorTrailerWriteUnsafe_PK(unsigned char ucAddressingMode,
                            unsigned char ucAddress,
                      const unsigned char *aucSectorTrailer,
                            unsigned char ucAauthMode,
                        const unsigned char *aucProvidedKey);
```

- *ucAddressingMode* - Specifies the address mode. Possible values of this parameter are BLOCK_ADDRESS_MODE (0x00) or SECTOR_ADDRESS_MODE (0x01). If any other value is been sent the function returns an error code WRONG_ADDRESS_MODE.
- *ucAddress* - Sectors or sector trailers block address, depending on ucAddressingMode.

When using a sector address mode, then,in the case of MIFARE ® 1K card, the range can be from 0 to 15 (this card has 16 sectors) and the same card type in block addressing mode can use the values 0 to 63 with the possible error if the addressed block isn't also the sector trailer.

- *aucSectorTrailer* - Pointer to 6 byte array that contains the "raw" data for the address sector trailer entry
- *ucAuthMode* - This parameter defines whether to perform authentication key A or key B. It can have two values, namely: AUTHENT1A (0x60) or AUTHENT1B (0x61)
- *ucReaderKeyIndex* - The default method of authentication (when the functions without a suffix is used) performs the authenticity proving by using the selected key index from the reader. In the linear address mode, this applies to all sectors that are written

aucProvidedKey - Pointer to the sixth byte array containing the key for authenticity proving in the "Provided Key" method. _PK Suffix in the name of the function indicates this method usage.

Functions for working with value blocks

Value blocks represents an optional MIFARE® card functionality. This is actually a mode in which the entire block of data on the card (16 bytes) represents one four-byte value. In this mode, you can add any data block on the card (except of course, block 0, the zero sector and sector trailer). The values in the value blocks are formatted in a special way and in addition to value records contains the one byte address value, which gives users the added ability to implement the backup system.

D-Logic card readers takes care of the proper value blocks formatting so the set of functions that handle only with four byte values are available to users. It should be mentioned that the use of value blocks makes sense if the access rights to desired block are set on values 1, 6 or 0 (the default in new cards) which allows their values increment and decrement. First of all, value blocks must be initiated, value and associated address must be in compliance with the appropriate format of sixteen byte records. The best and easiest way for value blocks initialization is with a set of Windows API functions IS21 ValueBlockWrite or ValueBlockInSectorWrite.

- **ValueBlockRead,**
- **ValueBlockRead_AKM1,**
- **ValueBlockRead_AKM2,**
- **ValueBlockRead_PK**

Functions description:

These functions are used to read the fourth byte value of value blocks. In addition they are returning the associated address stored in the value block. Functions used block addressing (the first card block has the address 0; first sector trailer has address 3, the next one 7, etc. until the last Mifare ® 1K block which is also a trailer of the last sector, has an address 63)

Function declaration (C language):

```
unsigned long ValueBlockRead(long *lValue,
                         unsigned char *ucValueAddr,
                         unsigned char ucBlockAddress,
                         unsigned char ucAuthMode,
                         unsigned char ucKeyIndex);

unsigned long ValueBlockRead_AKM1(long *lValue,
                              unsigned char *ucValueAddr,
                              unsigned char ucBlockAddress,
                              unsigned char ucAuthMode);

unsigned long ValueBlockRead_AKM1(long *lValue,
                              unsigned char *ucValueAddr,
                              unsigned char ucBlockAddress,
                              unsigned char ucAuthMode);

unsigned long ValueBlockRead_PK(long *lValue,
                            unsigned char *ucValueAddr,
                            unsigned char ucBlockAddress,
                            unsigned char ucAuthMode,
                            const unsigned char *key);
```

- *lValue* - Pointer to a variable of a type "long" over which the block value returns
- *ucValueAddr* - Pointer to a variable of unsigned char type is returned via the one byte address which gives the added ability for a backup system implementation
- *ucBlockAddress* - Block address

- *ucAuthMode* - This parameter defines whether to perform authentication key A or key B. It can have two values, namely: AUTHENT1A (0x60) or AUTHENT1B (0x61).
- *ucReaderKeyIndex* - The default method of authentication (when the functions without a suffix is used) performs the authenticity proving by using the selected key index from the reader. In the linear address mode it applies to all sectors for writing
- *aucProvidedKey* - Pointer to the six-byte array that contains the key for authentication of the "Provided Key" method. _PK function name suffix indicates to the use of this method.

- **ValueBlockInSectorRead,**
- **ValueBlockInSectorRead_AKM1,**
- **ValueBlockInSectorRead_AKM2,**
- **ValueBlockInSectorRead_PK**

Functions description

These functions do the same as ValueBlockRead group functions and are proper for reading 4 byte values of the value blocks. In addition they return the associated address stored in the value block. The only difference is the use of so-called sectoral addressing. Sectoral addressing means separately sending sector and block addresses within a sector. For MIFARE® 1K card sector address may be in the range 0 to 15, and blocks address within the sector ranging from 0 to 3. For MIFARE ® 4k sector address may be in the range of 0 to 39 and since the second half of the address space organization is different (above 2 MB) blocks address in the last 8 sectors (sectors 32 to 39) may be in the range of 0 to 15.

Function declaration (C language):

```
unsigned long ValueBlockInSectorRead(long *lValue,
                                 unsigned char *ucValueAddr,
                                 unsigned char ucSectorAddress,
                          unsigned char ucBlockInSectorAddress,
                                 unsigned char ucAuthMode,
                                 unsigned char ucKeyIndex);

unsigned long ValueBlockInSectorRead_AKM1(long *lValue,
                                      unsigned char *ucValueAddr,
                                   unsigned char ucSectorAddress,
                            unsigned char ucBlockInSectorAddress,
                                      unsigned char ucAuthMode);
unsigned long ValueBlockInSectorRead_AKM1(long *lValue,
                                              unsigned char *ucValueAddr,
                                           unsigned char ucSectorAddress,
                                    unsigned char ucBlockInSectorAddress,
                                          unsigned char ucAuthMode);

unsigned long ValueBlockInSectorRead_PK(long *lValue,
                                           unsigned char *ucValueAddr,
                                        unsigned char ucSectorAddress,
                                 unsigned char ucBlockInSectorAddress,
                                           unsigned char ucAuthMode,
                                 const unsigned char *aucProvidedKey);
```

- *lValue* - Pointer to a variable of a long type over which the value block returns
- *ucValueAddr* - Pointer to a variable of unsigned char type is returned via the one byte address which gives the added ability for a backup system implementation
- *ucSectorAddress* - Sector address
- *ucBlockInSectorAddress* - Block address in a sector
- *ucAuthMode* - This parameter defines whether to perform authentication key A or key B. It can have two values, namely: AUTHENT1A (0x60) or AUTHENT1B (0x61)
- *ucReaderKeyIndex* - e default method of authentication (when the functions without a suffix is used) performs the authenticity proving by using the selected key index from the reader. In the linear address mode it applies to all sectors for writing
- *aucProvidedKey* - Pointer to the six-byte array that contains the key for authentication of the "Provided Key" method. _PK function name suffix indicates to the use of this method.
- **ValueBlockWrite,**
- **ValueBlockWrite_AKM1,**
- **ValueBlockWrite_AKM2,**
- **ValueBlockWrite_PK**

Functions description:

These functions are used to initialize and write fourth byte value blocks values and store the associated address in the value block. Functions using the so-called block addressing (the first card block has the address 0; trailer has a first sector address 3 and the next 7, etc. until the last block of Mifare® 1k which is also a trailer of the last sector and has an address 63).

Function declaration (C language):

```c
unsigned long ValueBlockWrite(long lValue,
                          unsigned char ucValueAddr,
                          unsigned char ucBlockAddress,
                          unsigned char ucAuthMode,
                          unsigned char ucKeyIndex);

unsigned long ValueBlockWrite_AKM1(long lValue,
                               unsigned char ucValueAddr,
                               unsigned char ucBlockAddress,
                               unsigned char ucAuthMode);

unsigned long ValueBlockWrite_AKM2(long lValue,
                               unsigned char ucValueAddr,
                               unsigned char ucBlockAddress,
                               unsigned char ucAuthMode);

unsigned long ValueBlockWrite_PK(long lValue,
                             unsigned char ucValueAddr,
                             unsigned char ucBlockAddress,
                             unsigned char ucAuthMode,
                      const unsigned char *aucProvidedKey);
```

- *lValue* - Value for the value block entry
- *ucValueAddr* - Value block associated address
- *ucBlockAddress* - Block address
- *ucAuthMode* - This parameter defines whether to perform authentication with A key or B key. It can have two values, namely: AUTHENT1A (0x60) or AUTHENT1B (0x61).
- *ucReaderKeyIndex* - The default method of authentication (when the functions without a suffix is used) performs the authenticity proving by using the selected key index from the reader. In the linear address mode, this applies to all sectors that are written
- *aucProvidedKey* - Pointer to the sixth byte array containing the key for authenticity proving in the "Provided Key" method. _PK Suffix in the name of the function indicates this method usage.

- **ValueBlockInSectorWrite,**
- **ValueBlockInSectorWrite_AKM1,**
- **ValueBlockInSectorWrite_AKM2,**
- **ValueBlockInSectorWrite_PK**

Functions description:

These functions are similar to the ValueBlockWrite group functions. They use for entry, value blocks 4 bytes values initialization. In addition, stores the associated address into the block value. The only difference is the sectoral addressing usage. Sectoral addressing means separately sending sector and block addresses within a sector. For MIFARE® 1K card sector address may be in the range 0 to 15, and blocks address within the sector ranging from 0 to 3. For MIFARE ® 4k sector address may be in the range of 0 to 39 and since the second half of the address space organization is different (above 2 MB) blocks address in the last 8 sectors (sectors 32 to 39) may be in the range of 0 to 15.

Functions declaration (C language):

```
unsigned long ValueBlockInSectorWrite(long lValue,
                                      unsigned char ucValueAddr,
                                      unsigned char ucSectorAddress,
                                      unsigned char ucBlockInSectorAddress,
                                      unsigned char ucAuthMode,
                                      unsigned char ucKeyIndex);

unsigned long ValueBlockInSectorWrite_AKM1(long lValue,
                                           unsigned char ucValueAddr,
                                           unsigned char ucSectorAddress,
                                     unsigned char ucBlockInSectorAddress,
                                           unsigned char ucAuthMode);

unsigned long ValueBlockInSectorWrite_AKM2(long lValue,
                                           unsigned char ucValueAddr,
                                           unsigned char ucSectorAddress,
                                     unsigned char ucBlockInSectorAddress,
                                           unsigned char ucAuthMode);

 unsigned long ValueBlockInSectorWrite_PK(long lValue,
                                          unsigned char ucValueAddr,
                                          unsigned char ucSectorAddress,
                                    unsigned char ucBlockInSectorAddress,
                                                 unsigned char ucAuthMode,
                                 const unsigned char *aucProvidedKey);
```

- *lValue* - Values for the value block entry
- *ucValueAddr* - Value block associated address
- *ucSectorAddress* - Sector address
- *ucBlockInSectorAddress* - Block address of a sector
- *ucAuthMode* - This parameter defines whether to perform authentication key A or key B. It can have two values, namely: AUTHENT1A (0x60) or AUTHENT1B (0x61)

$^{35}_{17}$ *ucReaderKeyIndex* - The default method of authentication (when the functions without a suffix is used) performs the authenticity proving by using the selected key index from the reader. In the linear address mode it applies to all sectors for writing

$^{35}_{17}$ *aucProvidedKey* - Pointer to the six-byte array that contains the key for authentication of the "Provided Key" method. _PK function name suffix indicates to the use of this method.

$^{35}_{17}$ **ValueBlockIncrement,**

$^{35}_{17}$ **ValueBlockIncrement_AKM1,**

$^{35}_{17}$ **ValueBlockIncrement_AKM2,**

$^{35}_{17}$ **ValueBlockIncrement_PK**

Functions description:

This feature set is used to increment the value 4 byte value blocks. The value of value block increment is sent as a parameter of these functions. Functions use block addressing (the first card block has the address 0; first sector trailer has address 3, the next one 7, etc. until the last Mifare ® 1K block which is also a trailer of the last sector, has an address 63).

Funcution declaration (C language):

```
unsigned long
ValueBlockInSectorIncrement(long lIncrementValue,
                            unsigned char ucSectorAddress,
                            unsigned char ucBlockInSectorAddress,
                            unsigned char ucAuthMode,
                            unsigned char ucKeyIndex);

unsigned long
ValueBlockInSectorIncrement_AKM1(long lIncrementValue,
                                 unsigned char ucSectorAddress,
                          unsigned char ucBlockInSectorAddress,
                                  unsigned char ucAuthMode);

unsigned long
ValueBlockInSectorIncrement_AKM2(long lIncrementValue,
                                 unsigned char ucSectorAddress,
                          unsigned char ucBlockInSectorAddress,
                                 unsigned char ucAuthMode);
 unsigned long
 ValueBlockInSectorIncrement_PK(long lIncrementValue,
                                unsigned char ucSectorAddress,
                         unsigned char ucBlockInSectorAddress,
                                     unsigned char ucAuthMode,
                          const unsigned char *aucProvidedKey);
```

- *lIncrementValue* - The value of value block increment
- *ucBlockAddress* - Block adress in a sector
- *ucAuthMode* - This parameter defines whether to perform authentication key A or key B. It can have two values, namely: AUTHENT1A (0x60) or AUTHENT1B (0x61)
- *ucReaderKeyIndex* - The default method of authentication (when the functions without a suffix is used) performs the authenticity proving by using the selected key index from the reader. In the linear address mode it applies to all sectors for writing
- *aucProvidedKey* - Pointer to the six-byte array that contains the key for authentication of the "Provided Key" method. _PK function name suffix indicates to the use of this method.

- **ValueBlockInSectorIncrement,**
- **ValueBlockInSectorIncrement_AKM1,**
- **ValueBlockInSectorIncrement_AKM2,**
- **ValueBlockInSectorIncrement_PK**

Functions description:

These functions has the same purpose as ValueBlockIncrement group functions and are used for reading 4 byte values of the value blocks. The value of value block increment is sent as a parameter of these functions. The only difference is the sectoral addressing usage. Sectoral addressing means separately sending sector and block addresses within a sector. For MIFARE® 1K card sector address may be in the range 0 to 15, and blocks address within the sector ranging from 0 to 3. For MIFARE ® 4k sector address may be in the range of 0 to 39 and since the second half of the address space organization is different (above 2 MB) blocks address in the last 8 sectors (sectors 32 to 39) may be in the range of 0 to 15.

Functions declaration (C language):

```
unsigned long
ValueBlockInSectorIncrement(long lIncrementValue,
                            unsigned char ucSectorAddress,
                            unsigned char ucBlockInSectorAddress,
                            unsigned char ucAuthMode,
                            unsigned char ucKeyIndex);
unsigned long
ValueBlockInSectorIncrement_AKM1(long lIncrementValue,
                                 unsigned char ucSectorAddress,
                          unsigned char ucBlockInSectorAddress,
                              unsigned char ucAuthMode);

unsigned long
ValueBlockInSectorIncrement_AKM2(long lIncrementValue,
                                 unsigned char ucSectorAddress,
                          unsigned char ucBlockInSectorAddress,
                                 unsigned char ucAuthMode);

unsigned long
ValueBlockInSectorIncrement_PK(long lIncrementValue,
                                 unsigned char ucSectorAddress,
                          unsigned char ucBlockInSectorAddress,
                                      unsigned char ucAuthMode,
                          const unsigned char *aucProvidedKey);
```

- *lIncrementValue* - The value of value block increment
- *ucSectorAddress* - Sector address
- *ucBlockInSectorAddress* - Block address within a sector
- *ucAuthMode* - This parameter defines whether to perform authentication key A or key B. It can have two values, namely: AUTHENT1A (0x60) or AUTHENT1B (0x61)
- *ucReaderKeyIndex* - The default method of authentication (when the functions without a suffix is used) performs the authenticity proving by using the selected key index from the reader. In the linear address mode it applies to all sectors for writing
- *aucProvidedKey* - Pointer to the six-byte array that contains the key for authentication of the "Provided Key" method. _PK function name suffix indicates to the use of this method.

- **ValueBlockDecrement,**
- **ValueBlockDecrement_AKM1,**
- **ValueBlockDecrement_AKM2,**
- **ValueBlockDecrement_PK**

Functions description:

This set of functions is used to decrement 4 byte value of value blocks. The value of the value block decrement is sent as a parameter of these functions. Functions use block addressing (the first card block has the address 0; first sector trailer has address 3, the next one 7, etc. until the last Mifare ® 1K block which is also a trailer of the last sector, has an address 63).

Functions declaration (C language):

```c
unsigned long ValueBlockDecrement(long lDecrementValue,
                                  unsigned char ucBlockAddress,
                                  unsigned char ucAuthMode,
                                  unsigned char ucKeyIndex);

unsigned long ValueBlockDecrement_AKM1(long lDecrementValue,
                                       unsigned char ucBlockAddress,
                                       unsigned char ucAuthMode);

unsigned long ValueBlockDecrement_AKM2(long lDecrementValue,
                                       unsigned char ucBlockAddress,
                                       unsigned char ucAuthMode);

unsigned long ValueBlockDecrement_PK(long lDecrementValue,
                                     unsigned char ucBlockAddress,
                                         unsigned char ucAuthMode,
                                 const unsigned char *aucProvidedKey);
```

- *lDecrementValue* - The value of value block decrement
- *ucBlockAddress* - Block address within a sector
- *ucAuthMode* - This parameter defines whether to perform authentication key A or key B. It can have two values, namely: AUTHENT1A (0x60) or AUTHENT1B (0x61)
- *ucReaderKeyIndex* - The default method of authentication (when the functions without a suffix is used) performs the authenticity proving by using the selected key index from the reader. In the linear address mode it applies to all sectors for writing
- *aucProvidedKey* - Pointer to the six-byte array that contains the key for authentication of the "Provided Key" method. _PK function name suffix indicates to the use of this method.

- **ValueBlockInSectorDecrement,**
- **ValueBlockInSectorDecrement_AKM1,**
- **ValueBlockInSectorDecrement_AKM2,**
- **ValueBlockInSectorDecrement_PK**

Functions description:

These functions work the same as ValueBlockDecrement group functions and are made for the value blocks 4 byte values decrement. The value of the value block decrement is sent as a parameter to these functions. Only difference is the sectoral addressing usage. That includes separately sending sector addresses and block addresses within a sector. For MIFARE® 1K card sector address may be in the range 0 to 15, and blocks address within the sector ranging from 0 to 3. For MIFARE ® 4k sector address may be in the range of 0 to 39 and since the second half of the address space organization is different (above 2 MB) blocks address in the last 8 sectors (sectors 32 to 39) may be in the range of 0 to 15

Function declaration (C language):

```
unsigned long
ValueBlockInSectorDecrement(long lDecrementValue,
                            unsigned char ucSectorAddress,
                            unsigned char ucBlockInSectorAddress,
                            unsigned char ucAuthMode,
                            unsigned char ucKeyIndex);

unsigned long
ValueBlockInSectorDecrement_AKM1(long lDecrementValue,
                                 unsigned char ucSectorAddress,
                          unsigned char ucBlockInSectorAddress,
                                unsigned char ucAuthMode);

unsigned long
ValueBlockInSectorDecrement_AKM2(long lDecrementValue,
                                 unsigned char ucSectorAddress,
                          unsigned char ucBlockInSectorAddress,
                              unsigned char ucAuthMode);

unsigned long
ValueBlockInSectorDecrement_PK(long lDecrementValue,
                               unsigned char ucSectorAddress,
                        unsigned char ucBlockInSectorAddress,
                               unsigned char ucAuthMode,
                         const unsigned char *aucProvidedKey);
```

- *lDecrementValue* - The value of value block decrement
- *ucSectorAddress* - Sector address
- *ucBlockInSectorAddress* - Block address within a sector
- *ucAuthMode* - This parameter defines whether to perform authentication key A or key B. It can have two values, namely: AUTHENT1A (0x60) or AUTHENT1B (0x61)
- *ucReaderKeyIndex* - The default method of authentication (when the functions without a suffix is used) performs the authenticity proving by using the selected key index from the reader. In the linear address mode it applies to all sectors for writing

aucProvidedKey - Pointer to the six-byte array that contains the key for authentication of the "Provided Key" method. _PK function name suffix indicates to the use of this method.

Additional general functions for working with the cards

- **GetDlogicCardType**

Function description:

This is a function that return a value that corresponds to the type of the detected card. Values of the card type defined in the following list:

```
#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
```

If the card type is not supported, function return the value zero.

```
#define TAG_UNKNOWN                          0x00
```

Function declaration (C language):

```
unsigned long GetDlogicCardType(unsigned char *lpucCardType);
```

Parameter description:

- lpucCardType Pointer to the variable in which the value of the card type written.

- **GetCardIdEx**

Function description:

Function writes the card serial number card in a byte array, and also gives the length of the serial number of the card. Length of the serial number can be 4 (UID size: single), 7 (UID size: double) or 10 (UID size: triple) bytes, so it is necessary to define an array of 10 bytes which will be used to enter the card serial number.

Function declaration (C language):

```
GetCardIdEx(unsigned char *lpuSak,
         unsigned char *aucUid,
         unsigned char *lpucUidSize);
```

Parameters description:

- lpuSak Pointer to the variable in which the SAK (Select Acknowledge) written.
- AucUid Pointer to array of bytes in which the card UID (Unique Identifier) written.
- LpucUidSize Pointer to variable for the card UID size.

Function for DLL version reading

- **GetDllVersion**

Function description:

The function returns the number of type unsigned long (4 bytes). The meaning of bytes in this number are presented in the table below.

0	1	2	3
DLL Major Version	DLL Minor Version	DLL Build	

Example:

Function returns the number 0x00080202.

DLL Major Version = 2

DLL Minor Version = 2

DLL Build = 8

Function declaration (C language):

```
unsigned long GetDllVersion(void);
```

Functions that support NDEF records

- **get_ndef_record_count**

Function description:

Function returns the number of NDEF messages that have been read from the card, and number of NDEF records, number of NDEF empty messages. Also, function returns array of bytes containing number of messages pairs. First byte of pair is message ordinal, and second byte is number of NDEF records in that message. Message ordinal starts from 1.

Function declaration (C language)

```
unsigned long get_ndef_record_count(
               unsigned char *ndef_message_cnt,
               unsigned char *ndef_record_cnt,
               unsigned char *ndef_record_array,
               unsigned char *empty_ndef_message_cnt);
```

Parameters description:

- ndef_message_cnt pointer to the variable containing number of NDEF messages
- ndef_record_cnt pointer to the variable containing number of NDEF records
- ndef_record_array pointer to the array of bytes containing pairs (message ordinal – number of records)
- empty_ndef_message_cnt pointer to the variable containing number of empty messages

- **read_ndef_record**

Function description:

Function returns TNF, type of record, ID and payload from the NDEF record. NDEF record shall be elected by the message ordinal and record ordinal in this message.

Function declaration (C language)

```
unsigned long read_ndef_record(unsigned char message_nr,
                              unsigned char record_nr,
                              unsigned char *tnf,
                              unsigned char *type_record,
                              unsigned char *type_length,
                              unsigned char *id,
                              unsigned char *id_length,
                              unsigned char *payload,
                              unsigned long *payload_length);
```

Parameters description:

• message_nr	NDEF message ordinal (starts form 1)
• record_nr	NDEF record ordinal (in message)
• tnf	pointer to the variable containing TNF of record
• type_record	pointer to array containing type of record
• type_length string	pointer to the variable containing length of type of record
• id	pointer to array containing ID of record
• id_length	pointer to the variable containing length of ID of record string
• payload	pointer to array containing payload of record
• payload_length	pointer to the variable containing length of payload

- **write_ndef_record**

Function description:

Function adds a record to the end of message, if one or more records already exist in this message. If current message is empty, then this empty record will be replaced with the record. Parameters of function are: ordinal of message, TNF, type of record, ID, payload. Function also returns pointer to the variable which reported that the card formatted for NDEF using (card does not have a capability container, for example new Mifare Ultralight, or Mifare Classic card).

Function declaration (C language)

```
write_ndef_record(unsigned char message_nr,
unsigned char *tnf,
unsigned char *type_record,
unsigned char *type_length,
unsigned char *id,
unsigned char *id_length,
unsigned char *payload,
unsigned long *payload_length,
unsigned char *card_formated);
```

Parameters description:

• message_nr	NDEF message ordinal (starts form 1)
• tnf	pointer to the variable containing TNF of record
• type_record	pointer to array containing type of record
• type_length string	pointer to the variable containing length of type of record
• id	pointer to array containing ID of record
• id_length	pointer to the variable containing length of ID of record string
• payload	pointer to array containing payload of record
• payload_length	pointer to the variable containing length of payload
• card_formated for NDEF using.	pointer to the variable which shows that the card formatted

- **erase_last_ndef_record**

Function description:

Function deletes the last record of selected message. If message contains one record, then it will be written empty message.

Function declaration (C language)

```
unsigned long erase_last_ndef_record(unsigned char message_nr);
```

Parameter description:

- message_nr NDEF message ordinal (starts form 1)

- **erase_all_ndef_records**

Function description:

Function deletes all records of message, then writes empty message.

Function declaration (C language)

```
unsigned long erase_all_ndef_records(unsigned char message_nr);
```

Parameter description:

- message_nr NDEF message ordinal (starts form 1)

- **ndef_card_initialization**

Function description:

Function prepares the card for NDEF using. Function writes Capability Container (CC) if necessary, and writes empty message. If card is MIFARE CLASSIC or MIFARE PLUS, then function writes MAD (MIFARE Application Directory), and default keys and access bits for NDEF using.

Function declaration (C language)

```
unsigned long ndef_card_initialization(void);
```

ERROR CODES OF NDEF FUNCTIONS

```
UFR_WRONG_NDEF_CARD_FORMAT = 0x80,
UFR_NDEF_MESSAGE_NOT_FOUND = 0x81,
UFR_NDEF_UNSUPPORTED_CARD_TYPE = 0x82,
UFR_NDEF_CARD_FORMAT_ERROR = 0x83,
UFR_MAD_NOT_ENABLED = 0x84,
UFR_MAD_VERSION_NOT_SUPPORTED = 0x85,
```