

Roger Mifare Library

1.0.0.0

Generated by Doxygen 1.7.4

Fri Jul 29 2011 10:33:43



# Contents

<b>1</b>	<b>Roger Mifare Library.</b>	<b>1</b>
1.1	Introduction . . . . .	1
1.2	Includes . . . . .	2
<b>2</b>	<b>Mifare Classic</b>	<b>3</b>
<b>3</b>	<b>Mifare UltraLight</b>	<b>5</b>
<b>4</b>	<b>Roger Secure Sector</b>	<b>7</b>
<b>5</b>	<b>Read Card Buffer</b>	<b>9</b>
<b>6</b>	<b>Reader Settings</b>	<b>11</b>
<b>7</b>	<b>Roger Mifare Library Version</b>	<b>13</b>
<b>8</b>	<b>Module Documentation</b>	<b>15</b>
8.1	Functions supporting Mifare Classic . . . . .	15
8.1.1	Function Documentation . . . . .	16
8.1.1.1	mfClassicReadHost . . . . .	16
8.1.1.2	mfClassicReadTerminal . . . . .	16
8.1.1.3	mfClassicWriteHost . . . . .	17
8.1.1.4	mfClassicWriteTerminal . . . . .	18
8.1.1.5	mfCreateValueBlockHost . . . . .	18
8.1.1.6	mfCreateValueBlockTerminal . . . . .	19
8.1.1.7	mfIncrementHost . . . . .	19
8.1.1.8	mfIncrementTerminal . . . . .	20
8.1.1.9	mfDecrementHost . . . . .	20

8.1.1.10	mfDecrementTerminal . . . . .	21
8.1.1.11	mfRestoreHost . . . . .	22
8.1.1.12	mfRestoreTerminal . . . . .	22
8.2	Functions supporting Mifare UltraLight . . . . .	23
8.2.1	Function Documentation . . . . .	23
8.2.1.1	mfULReadHost . . . . .	23
8.2.1.2	mfULReadTerminal . . . . .	24
8.2.1.3	mfULWriteHost . . . . .	24
8.2.1.4	mfULWriteTerminal . . . . .	25
8.3	Functions supporting Roger Secure Sector . . . . .	25
8.3.1	Function Documentation . . . . .	25
8.3.1.1	mfRssSetHost . . . . .	25
8.3.1.2	mfRssSetTerminal . . . . .	26
8.3.1.3	mfRssDelHost . . . . .	27
8.3.1.4	mfRssDelTerminal . . . . .	27
8.3.1.5	mfRssReadHost . . . . .	28
8.3.1.6	mfRssReadTerminal . . . . .	28
8.3.1.7	mfRssWriteHost . . . . .	29
8.3.1.8	mfRssWriteTerminal . . . . .	29
8.4	Functions supporting Read Card Buffer . . . . .	30
8.4.1	Function Documentation . . . . .	30
8.4.1.1	readCardBufferHost . . . . .	30
8.4.1.2	readCardBufferTerminal . . . . .	30
8.4.1.3	readCardIDHost . . . . .	31
8.4.1.4	readCardIDTerminal . . . . .	31
8.5	Functions supporting - Reader Settings . . . . .	32
8.5.1	Function Documentation . . . . .	32
8.5.1.1	readCardSettingsHost . . . . .	32
8.5.1.2	readCardSettingsTerminal . . . . .	33
8.5.1.3	writeCardSettingsHost . . . . .	33
8.5.1.4	writeCardSettingsTerminal . . . . .	34
8.6	Flags returned by library functions . . . . .	34
8.6.1	Define Documentation . . . . .	34
8.6.1.1	MIFARE_OK . . . . .	35

8.6.1.2	MIFARE_ERROR	35
8.6.1.3	MIFARE_WRONG_EPSO_FRAME	35
8.6.1.4	MIFARE_NEW_CARD_NOT_FOUND	35
8.6.1.5	MIFARE_WRONG_KEY	35
8.6.1.6	MIFARE_WRONG_BLOCK	35
8.6.1.7	MIFARE_WRONG_PAGE	35
8.6.1.8	MIFARE_WRONG_SECTOR	35
8.7	Settings for Mifare Classic cards	35
8.7.1	Define Documentation	36
8.7.1.1	MF_CLASSIC_KEY_TYPE_A	36
8.7.1.2	MF_CLASSIC_KEY_TYPE_B	36
8.7.1.3	VALUE_BLOCK_INCREMENT	36
8.7.1.4	VALUE_BLOCK_DECREMENT	36
8.7.1.5	VALUE_BLOCK_RESTORE	36
8.8	Reader configuration parameters - only RUD-3.	36
8.8.1	Typedef Documentation	36
8.8.1.1	CARD_SETTINGS	36
8.8.1.2	PCARD_SETTINGS	36
8.8.2	Enumeration Type Documentation	36
8.8.2.1	readSequence_e	36
8.8.2.2	accessKey_e	37
8.8.2.3	outputOrder_e	37
8.9	Function - Roger Mifare Library Version	37
8.9.1	Function Documentation	37
8.9.1.1	mifareLibVer	37
<b>9</b>	<b>Data Structure Documentation</b>	<b>39</b>
9.1	_CARD_NUMBER_SETTINGS Struct Reference	39
9.1.1	Detailed Description	39
9.1.2	Field Documentation	39
9.1.2.1	Csn	40
9.1.2.2	Msn	40
9.1.2.3	Ssn	40
9.1.2.4	UserKey	40

---

9.1.2.5	Step	40
9.2	<a href="#">_CARD_NUMBER_SETTINGS::_Csn_s Struct Reference</a>	40
9.2.1	Detailed Description	40
9.2.2	Field Documentation	40
9.2.2.1	Order	40
9.3	<a href="#">_CARD_NUMBER_SETTINGS::_Msn_s Struct Reference</a>	40
9.3.1	Detailed Description	41
9.3.2	Field Documentation	41
9.3.2.1	Aid	41
9.3.2.2	Block	41
9.3.2.3	UsesKey	41
9.3.2.4	Order	41
9.4	<a href="#">_CARD_NUMBER_SETTINGS::_Ssn_s Struct Reference</a>	41
9.4.1	Detailed Description	41
9.4.2	Field Documentation	42
9.4.2.1	Sector	42
9.4.2.2	Block	42
9.4.2.3	UsesKey	42
9.4.2.4	Order	42

# Chapter 1

## Roger Mifare Library.

The EPSO protocol parser for Roger Mifare proximity readers.

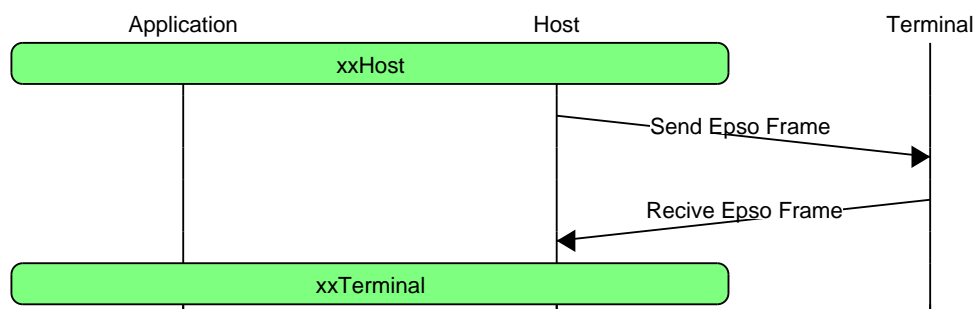
### 1.1 Introduction

The EPSO protocol defines how devices can exchange data. The unit which initializes the communication and send commands is called **Host** (PC or special microprocessor) while the unit which responses the commands is called **Terminal** (Roger proximity reader configured or programmed to work on EPSO protocol).

The Roger library has the same approach. Each supported functionality has two functions.

- *xxHost* - Function that creates EPSO frame
- *xxTerminal* - Function that converts EPSO frame to data

User application calls Mifare function - *xxHost* from Roger Mifare library. The library returns Epso frame ready to send to Roger Mifare proximity reader. The reader returns an EPSO answer frame. User application can decode the frame using function - *xxTerminal*



## 1.2 Includes

- RogerMifare.dll - Dynamic-Link Library
- RogerMifare.lib - Provides linking to dynamic library
- RogerMifare.h - Defines library function prototypes and constants
- CardNumberSettings.h - Defines reader settings struct



## Chapter 2

# Mifare Classic

Functions supporting Mifare Classic.

- *Read block*
- *Write to block*
- *Increment value block*
- *Decrement value block*
- *Restore value block*

[Functions supporting Mifare Classic](#)

[Settings for Mifare Classic cards](#)

[Flags returned by library functions](#)



## Chapter 3

# Mifare UltraLight

Functions supporting Mifare UltraLight.

- *Read page*
- *Write to page*

[Functions supporting Mifare UltraLight](#)

[Flags returned by library functions](#)



## Chapter 4

# Roger Secure Sector

Sector of data of the Mifare Classic card. The data are protected with a Roger secret key, which is not published. Only Roger reader have to access to data in Roger Secure Sector.

### Application:

- The main functions with data access in the Mifare Classic cards,
- Storing a unique identification number - *UID* - see [Functions supporting Read Card Buffer](#)

### Functions:

- *Set*
- *Delete*
- *Read*
- *Write*

[Functions supporting Roger Secure Sector](#)

[Flags returned by library functions](#)



## Chapter 5

# Read Card Buffer

Card Buffer - Roger Mifare proximity readers can read one of unique identification number - *UID* stored in Mifare cards.

- *CSN* - Card Serial Number
- *MSN* - Mad Serial Number - *UID* stored in the data in the sector. The sector is pointed by *AID* - Application Directory - only Mifare Classic.
- *SSN* - Sector Serial Number - *UID* stored in the data in the sector - only Mifare Classic.

Which number is read depends on the reader configuration - see [Reader Settings](#)

### Functions:

- *Read Card Buffer*
- *Read Card Serial Number*

[Functions supporting Read Card Buffer](#)

[Flags returned by library functions](#)

### Note

Functions Read Card Buffer returns a unique identification number - *UID*, only the first call for a particular card. Re-call returns [MIFARE\\_NEW\\_CARD\\_NOT\\_FOUND](#).





## Chapter 6

# Reader Settings

Roger Mifare proximity readers can read one of unique identification number - *UID* stored in Mifare cards.

- *CSN* - Card Serial Number
- *MSN* - Mad Serial Number - *UID* stored in the data in the sector. The sector is pointed by *AID* - Application Directory - only Mifare Classic.
- *SSN* - Sector Serial Number - *UID* stored in the data in the sector - only Mifare Classic.

Configuration settings for individual numbers describe structure- [\\_CARD\\_NUMBER\\_SETTINGS](#).

### Functions:

- *Read Settings*
- *Write Settings*

[Functions supporting - Reader Settings](#)

[Reader configuration parameters - only RUD-3.](#)

[Flags returned by library functions](#)



## Chapter 7

# Roger Mifare Library Version

Function - Roger Mifare Library Version

Flags returned by library functions



## Chapter 8

# Module Documentation

### 8.1 Functions supporting Mifare Classic

#### Functions

- DWORD [mfClassicReadHost](#) (\_\_in BYTE Id, \_\_in BYTE Sector, \_\_in BYTE Block, \_\_in PBYTE Key, \_\_in BYTE KeyType, \_\_out PCHAR Frame, \_\_out PBYTE FrameLength)
- DWORD [mfClassicReadTerminal](#) (\_\_in BYTE Id, \_\_in PCHAR Frame, \_\_in BYTE FrameLength, \_\_out PBYTE Data)
- DWORD [mfClassicWriteHost](#) (\_\_in BYTE Id, \_\_in BYTE Sector, \_\_in BYTE Block, \_\_in PBYTE Data, \_\_in PBYTE Key, \_\_in BYTE KeyType, \_\_out PCHAR Frame, \_\_out PBYTE FrameLength)
- DWORD [mfClassicWriteTerminal](#) (\_\_in BYTE Id, \_\_in PCHAR Frame, \_\_in BYTE FrameLength)
- DWORD [mfCreateValueBlockHost](#) (\_\_in BYTE Id, \_\_in BYTE Sector, \_\_in BYTE Block, \_\_in PBYTE ValueBlock, \_\_in BYTE Adr, \_\_in PBYTE Key, \_\_in BYTE KeyType, \_\_out PCHAR Frame, \_\_out PBYTE FrameLength)
- DWORD [mfCreateValueBlockTerminal](#) (\_\_in BYTE Id, \_\_in PCHAR Frame, \_\_in BYTE FrameLength)
- DWORD [mfIncrementHost](#) (\_\_in BYTE Id, \_\_in BYTE vbSector, \_\_in BYTE vbBlock, \_\_in PBYTE vbOperand, \_\_in PBYTE vbKey, \_\_in BYTE vbKeyType, \_\_in BYTE tranSector, \_\_in BYTE tranBlock, \_\_in PBYTE tranKey, \_\_in BYTE tranKeyType, \_\_out PCHAR Frame, \_\_out PBYTE FrameLength)
- DWORD [mfIncrementTerminal](#) (\_\_in BYTE Id, \_\_in PCHAR Frame, \_\_in BYTE FrameLength)
- DWORD [mfDecrementHost](#) (\_\_in BYTE Id, \_\_in BYTE vbSector, \_\_in BYTE vbBlock, \_\_in PBYTE vbOperand, \_\_in PBYTE vbKey, \_\_in BYTE vbKeyType, \_\_in BYTE tranSector, \_\_in BYTE tranBlock, \_\_in PBYTE tranKey, \_\_in BYTE tranKeyType, \_\_out PCHAR Frame, \_\_out PBYTE FrameLength)
- DWORD [mfDecrementTerminal](#) (\_\_in BYTE Id, \_\_in PCHAR Frame, \_\_in BYTE FrameLength)

- DWORD [mfRestoreHost](#) ( \_\_in BYTE *Id*, \_\_in BYTE *vbSector*, \_\_in BYTE *vbBlock*, \_\_in PBYTE *vbKey*, \_\_in BYTE *vbKeyType*, \_\_in BYTE *tranSector*, \_\_in BYTE *tranBlock*, \_\_in PBYTE *tranKey*, \_\_in BYTE *tranKeyType*, \_\_out PCHAR *Frame*, \_\_out PBYTE *FrameLength* )
- DWORD [mfRestoreTerminal](#) ( \_\_in BYTE *Id*, \_\_in PCHAR *Frame*, \_\_in BYTE *FrameLength* )

### 8.1.1 Function Documentation

8.1.1.1 **DWORD** [mfClassicReadHost](#) ( \_\_in BYTE *Id*, \_\_in BYTE *Sector*, \_\_in BYTE *Block*, \_\_in PBYTE *Key*, \_\_in BYTE *KeyType*, \_\_out PCHAR *Frame*, \_\_out PBYTE *FrameLength* )

Mifare Classic - Read block in sector. Host frame.

#### Parameters

<i>Id</i>	ID Number of Terminal - <i>TerminalID</i> .
<i>Sector</i>	Sector number.
<i>Block</i>	Block number in the <i>Sector</i> .
<i>Key</i>	A byte array containing the key to the <i>Block</i> . Size 6 bytes.
<i>KeyType</i>	Determines if the <i>Key</i> contains key type A <a href="#">MF_CLASSIC_KEY_TYPE_A</a> or type B <a href="#">MF_CLASSIC_KEY_TYPE_B</a> .
<i>Frame</i>	Returns a byte array containing an output frame of EPSO. Ready to send.
<i>FrameLength</i>	Returns a pointer to the size of <i>Frame</i> in bytes.

#### Returns

FLAGS:

- [MIFARE\\_OK](#).
- [MIFARE\\_ERROR](#)
- [MIFARE\\_WRONG\\_KEY](#)
- [MIFARE\\_WRONG\\_BLOCK](#)
- [MIFARE\\_WRONG\\_EPSO\\_FRAME](#)

#### Note

The byte array *Frame* must be large enough to return full frame of EPSO. Maximum frame size is 64 bytes

8.1.1.2 **DWORD** [mfClassicReadTerminal](#) ( \_\_in BYTE *Id*, \_\_in PCHAR *Frame*, \_\_in BYTE *FrameLength*, \_\_out PBYTE *Data* )

Mifare Classic - Read block in sector. Terminal frame.

#### Parameters

<i>Id</i>	ID Number of Terminal - <i>TerminalID</i> .
-----------	---

<i>Frame</i>	A byte array containing the input frame of EPSO. Received from the Terminal.
<i>Frame-Length</i>	The size of the <i>Frame</i> in bytes.
<i>Data</i>	Returns a byte array containing 16 bytes of data read from the <i>Block</i> .

**Returns**

FLAGS:

- [MIFARE\\_OK](#)
- [MIFARE\\_ERROR](#)

8.1.1.3 `DWORD mfClassicWriteHost ( _in BYTE Id, _in BYTE Sector, _in BYTE Block, _in PBYTE Data, _in PBYTE Key, _in BYTE KeyType, _out PCHAR Frame, _out PBYTE FrameLength )`

Mifare Classic - Write block in sector. Host frame.

**Parameters**

<i>Id</i>	ID Number of Terminal - <i>TerminalID</i> .
<i>Sector</i>	Sector number.
<i>Block</i>	Block number in the <i>Sector</i> .
<i>Data</i>	A byte array containing 16 bytes of data to write on the <i>Block</i> .
<i>Key</i>	A byte array containing the key to the <i>Block</i> . Size 6 bytes.
<i>KeyType</i>	Determines if the <i>Key</i> contains key type A <a href="#">MF_CLASSIC_KEY_TYPE_A</a> or type B <a href="#">MF_CLASSIC_KEY_TYPE_B</a> .
<i>Frame</i>	Returns a byte array containing an output frame of EPSO. Ready to send.
<i>Frame-Length</i>	Returns a pointer to the size of <i>Frame</i> in bytes.

**Returns**

FLAGS:

- [MIFARE\\_OK](#)
- [MIFARE\\_ERROR](#)
- [MIFARE\\_WRONG\\_KEY](#)
- [MIFARE\\_WRONG\\_BLOCK](#)
- [MIFARE\\_WRONG\\_EPSO\\_FRAME](#)

**Note**

The byte array *Frame* must be large enough to return full frame of EPSO. Maximum frame size is 64 bytes

#### 8.1.1.4 `DWORD mfClassicWriteTerminal ( __in BYTE Id, __in PCHAR Frame, __in BYTE FrameLength )`

Mifare Classic - Write block in sector. Terminal frame.

##### Parameters

<i>Id</i>	ID Number of Terminal - <i>TerminalID</i> .
<i>Frame</i>	A byte array containing the input frame of EPSO. Received from the Terminal.
<i>FrameLength</i>	The size of the <i>Frame</i> in bytes.

##### Returns

FLAGS:

- [MIFARE\\_OK](#)
- [MIFARE\\_ERROR](#)

#### 8.1.1.5 `DWORD mfCreateValueBlockHost ( __in BYTE Id, __in BYTE Sector, __in BYTE Block, __in PBYTE ValueBlock, __in BYTE Adr, __in PBYTE Key, __in BYTE KeyType, __out PCHAR Frame, __out PBYTE FrameLength )`

Mifare Classic - Create *Value Block*. Host frame.

##### Parameters

<i>Id</i>	ID Number of Terminal - <i>TerminalID</i> .
<i>Sector</i>	Sector number.
<i>Block</i>	Block number in the <i>Sector</i> .
<i>ValueBlock</i>	A Pointer to the 4 bytes value to write on the <i>Block</i> .
<i>Adr</i>	Address - <i>storage address</i> .
<i>Key</i>	A byte array containing the key to the <i>Block</i> . Size 6 bytes.
<i>KeyType</i>	Determines if the <i>Key</i> contains key type A <a href="#">MF_CLASSIC_KEY_TYPE_A</a> or type B <a href="#">MF_CLASSIC_KEY_TYPE_B</a> .
<i>Frame</i>	Returns a byte array containing an output frame of EPSO. Ready to send.
<i>FrameLength</i>	Returns a pointer to the size of <i>Frame</i> in bytes.

##### Returns

FLAGS:

- [MIFARE\\_OK](#)
- [MIFARE\\_ERROR](#)
- [MIFARE\\_WRONG\\_KEY](#)
- [MIFARE\\_WRONG\\_BLOCK](#)
- [MIFARE\\_WRONG\\_EPSO\\_FRAME](#)



**Note**

The byte array *Frame* must be large enough to return full frame of EPSO. Maximum frame size is 64 bytes

#### 8.1.1.6 `DWORD mfCreateValueBlockTerminal ( _in BYTE Id, _in PCHAR Frame, _in BYTE FrameLength )`

Mifare Classic - Create *Value Block*. Terminal frame.

**Parameters**

<i>Id</i>	ID Number of Terminal - <i>TerminalID</i> .
<i>Frame</i>	A byte array containing the input frame of EPSO. Received from the Terminal.
<i>FrameLength</i>	The size of the <i>Frame</i> in bytes.

**Returns**

FLAGS:

- [MIFARE\\_OK](#)
- [MIFARE\\_ERROR](#)

#### 8.1.1.7 `DWORD mfIncrementHost ( _in BYTE Id, _in BYTE vbSector, _in BYTE vbBlock, _in PBYTE vbOperand, _in PBYTE vbKey, _in BYTE vbKeyType, _in BYTE tranSector, _in BYTE tranBlock, _in PBYTE tranKey, _in BYTE tranKeyType, _out PCHAR Frame, _out PBYTE FrameLength )`

Mifare Classic - Increment *Value Block* Operation. Host frame.

**Parameters**

<i>Id</i>	ID Number of Terminal - <i>TerminalID</i> .
<i>vbSector</i>	Sector number that contains the <i>Value Block</i> .
<i>vbBlock</i>	Block number in the <i>vbSector</i> that contains the <i>Value Block</i> .
<i>vbOperand</i>	A Pointer to the 4 bytes value to increment the <i>Value Block</i> .
<i>vbKey</i>	A byte array containing the key to the <i>vbBlock</i> . Size 6 bytes.
<i>vbKeyType</i>	Determines if the <i>vbKey</i> contains key type A <a href="#">MF_CLASSIC_KEY_TYPE_A</a> or type B <a href="#">MF_CLASSIC_KEY_TYPE_B</a> .
<i>tranSector</i>	Sector number to transfer the <i>Value Block</i> after increment.
<i>tranBlock</i>	Block number in the <i>tranSector</i> to transfer the <i>Value Block</i> after increment.
<i>tranKey</i>	A byte array containing the key to the <i>tranBlock</i> . Size 6 bytes.
<i>tranKeyType</i>	Determines if the <i>tranKey</i> contains key type A <a href="#">MF_CLASSIC_KEY_TYPE_A</a> or type B <a href="#">MF_CLASSIC_KEY_TYPE_B</a> .
<i>Frame</i>	Returns a byte array containing an output frame of EPSO. Ready to send.
<i>FrameLength</i>	Returns a pointer to the size of <i>Frame</i> in bytes.

**Returns**

FLAGS:

- [MIFARE\\_OK](#)
- [MIFARE\\_ERROR](#)
- [MIFARE\\_WRONG\\_KEY](#)
- [MIFARE\\_WRONG\\_BLOCK](#)
- [MIFARE\\_WRONG\\_EPSO\\_FRAME](#)

**Note**

The byte array *Frame* must be large enough to return full frame of EPSO. Maximum frame size is 64 bytes

8.1.1.8 **DWORD** mfIncrementTerminal ( *\_in* BYTE *Id*, *\_in* PCHAR *Frame*, *\_in* BYTE *FrameLength* )

Mifare Classic - Increment *Value Block* Operation. Terminal frame.

**Parameters**

<i>Id</i>	ID Number of Terminal - <i>TerminalID</i> .
<i>Frame</i>	A byte array containing the input frame of EPSO. Received from the Terminal.
<i>FrameLength</i>	The size of the <i>Frame</i> in bytes.

**Returns**

FLAGS:

- [MIFARE\\_OK](#)
- [MIFARE\\_ERROR](#)

8.1.1.9 **DWORD** mfDecrementHost ( *\_in* BYTE *Id*, *\_in* BYTE *vbSector*, *\_in* BYTE *vbBlock*, *\_in* PBYTE *vbOperand*, *\_in* PBYTE *vbKey*, *\_in* BYTE *vbKeyType*, *\_in* BYTE *tranSector*, *\_in* BYTE *tranBlock*, *\_in* PBYTE *tranKey*, *\_in* BYTE *tranKeyType*, *\_out* PCHAR *Frame*, *\_out* PBYTE *FrameLength* )

Mifare Classic - Decrement *Value Block* Operation. Host frame.

**Parameters**

<i>Id</i>	ID Number of Terminal - <i>TerminalID</i> .
<i>vbSector</i>	Sector number that contains the <i>Value Block</i> .
<i>vbBlock</i>	Block number in the <i>vbSector</i> that contains the <i>Value Block</i> .
<i>vbOperand</i>	A Pointer to the 4 bytes value to decrement the <i>Value Block</i> .
<i>vbKey</i>	A byte array containing the key to the <i>vbBlock</i> . Size 6 bytes.

<i>vbKeyType</i>	Determines if the <i>vbKey</i> contains key type A <a href="#">MF_CLASSIC_KEY_TYPE_A</a> or type B <a href="#">MF_CLASSIC_KEY_TYPE_B</a> .
<i>tranSector</i>	Sector number to transfer the <i>Value Block</i> after decrement.
<i>tranBlock</i>	Block number in the <i>tranSector</i> to transfer the <i>Value Block</i> after decrement.
<i>tranKey</i>	A byte array containing the key to the <i>tranBlock</i> . Size 6 bytes.
<i>tranKeyType</i>	Determines if the <i>tranKey</i> contains key type A <a href="#">MF_CLASSIC_KEY_TYPE_A</a> or type B <a href="#">MF_CLASSIC_KEY_TYPE_B</a> .
<i>Frame</i>	Returns a byte array containing an output frame of EPSO. Ready to send.
<i>Frame- Length</i>	Returns a pointer to the size of <i>Frame</i> in bytes.

**Returns**

FLAGS:

- [MIFARE\\_OK](#)
- [MIFARE\\_ERROR](#)
- [MIFARE\\_WRONG\\_KEY](#)
- [MIFARE\\_WRONG\\_BLOCK](#)
- [MIFARE\\_WRONG\\_EPSO\\_FRAME](#)

**Note**

The byte array *Frame* must be large enough to return full frame of EPSO. Maximum frame size is 64 bytes

8.1.1.10 **DWORD** mfDecrementTerminal ( *\_\_in* BYTE *Id*, *\_\_in* PCHAR *Frame*, *\_\_in* BYTE *FrameLength* )

Mifare Classic - Decrement *Value Block* Operation. Terminal frame.

**Parameters**

<i>Id</i>	ID Number of Terminal - <i>TerminalID</i> .
<i>Frame</i>	A byte array containing the input frame of EPSO. Received from the Terminal.
<i>Frame- Length</i>	The size of the <i>Frame</i> in bytes.

**Returns**

FLAGS:

- [MIFARE\\_OK](#)
- [MIFARE\\_ERROR](#)

8.1.1.11 `DWORD mfRestoreHost ( _in BYTE Id, _in BYTE vbSector, _in BYTE vbBlock, _in  
 PBYTE vbKey, _in BYTE vbKeyType, _in BYTE tranSector, _in BYTE tranBlock,  
 _in PBYTE tranKey, _in BYTE tranKeyType, _out PCHAR Frame, _out PBYTE  
FrameLength )`

Mifare Classic - Restore *Value Block* Operation. Host frame.

#### Parameters

<i>Id</i>	ID Number of Terminal - <i>TerminalID</i> .
<i>vbSector</i>	Sector number that contains the <i>Value Block</i> .
<i>vbBlock</i>	Block number in the <i>vbSector</i> that contains the <i>Value Block</i> .
<i>vbKey</i>	A byte array containing the key to the <i>vbBlock</i> . Size 6 bytes.
<i>vbKeyType</i>	Determines if the <i>vbKey</i> contains key type A <a href="#">MF_CLASSIC_KEY_TYPE_A</a> or type B <a href="#">MF_CLASSIC_KEY_TYPE_B</a> .
<i>tranSector</i>	Sector number to transfer the <i>Value Block</i> .
<i>tranBlock</i>	Block number in the <i>tranSector</i> to transfer the <i>Value Block</i> .
<i>tranKey</i>	A byte array containing the key to the <i>tranBlock</i> . Size 6 bytes.
<i>tranKeyType</i>	Determines if the <i>tranKey</i> contains key type A <a href="#">MF_CLASSIC_KEY_TYPE_A</a> or type B <a href="#">MF_CLASSIC_KEY_TYPE_B</a> .
<i>Frame</i>	Returns a byte array containing an output frame of EPSO. Ready to send.
<i>Frame- Length</i>	Returns a pointer to the size of <i>Frame</i> in bytes.

#### Returns

FLAGS:

- [MIFARE\\_OK](#)
- [MIFARE\\_ERROR](#)
- [MIFARE\\_WRONG\\_KEY](#)
- [MIFARE\\_WRONG\\_BLOCK](#)
- [MIFARE\\_WRONG\\_EPSO\\_FRAME](#)

#### Note

The byte array *Frame* must be large enough to return full frame of EPSO. Maximum frame size is 64 bytes

8.1.1.12 `DWORD mfRestoreTerminal ( _in BYTE Id, _in PCHAR Frame, _in BYTE FrameLength  
 )`

Mifare Classic - Restore *Value Block* Operation. Terminal frame.

#### Parameters

<i>Id</i>	ID Number of Terminal - <i>TerminalID</i> .
<i>Frame</i>	A byte array containing the input frame of EPSO. Received from the Terminal.
<i>Frame- Length</i>	The size of the <i>Frame</i> in bytes.

**Returns**

FLAGS:

- [MIFARE\\_OK](#)
- [MIFARE\\_ERROR](#)

## 8.2 Functions supporting Mifare UltraLight

**Functions**

- DWORD [mfULReadHost](#) (\_\_in BYTE *Id*, \_\_in BYTE *Page*, \_\_out PCHAR *Frame*, \_\_out PBYTE *FrameLength*)
- DWORD [mfULReadTerminal](#) (\_\_in BYTE *Id*, \_\_in PCHAR *Frame*, \_\_in BYTE *FrameLength*, \_\_out PBYTE *Data*)
- DWORD [mfULWriteHost](#) (\_\_in BYTE *Id*, \_\_in BYTE *Page*, \_\_in PBYTE *Data*, \_\_out PCHAR *Frame*, \_\_out PBYTE *FrameLength*)
- DWORD [mfULWriteTerminal](#) (\_\_in BYTE *Id*, \_\_in PCHAR *Frame*, \_\_in BYTE *FrameLength*)

### 8.2.1 Function Documentation

8.2.1.1 DWORD [mfULReadHost](#) ( \_\_in BYTE *Id*, \_\_in BYTE *Page*, \_\_out PCHAR *Frame*, \_\_out PBYTE *FrameLength* )

Mifare UltraLight - Read page. Host frame.

**Parameters**

<i>Id</i>	ID Number of Terminal - <i>TerminalID</i> .
<i>Page</i>	Page number
<i>Frame</i>	Returns a byte array containing an output frame of EPSO. Ready to send.
<i>Frame-Length</i>	Returns a pointer to the size of <i>Frame</i> in bytes.

**Returns**

FLAGS:

- [MIFARE\\_OK](#)
- [MIFARE\\_WRONG\\_PAGE](#)
- [MIFARE\\_WRONG\\_EPSO\\_FRAME](#)
- [MIFARE\\_ERROR](#)

**Note**

The byte array *Frame* must be large enough to return full frame of EPSO. Maximum frame size is 64 bytes

8.2.1.2 **DWORD** mfULReadTerminal ( *\_in* BYTE *Id*, *\_in* PCHAR *Frame*, *\_in* BYTE *FrameLength*, *\_out* PBYTE *Data* )

Mifare UltraLight - Read page. Terminal frame.

#### Parameters

<i>Id</i>	ID Number of Terminal - <i>TerminalID</i> .
<i>Frame</i>	A byte array containing the input frame of EPSO. Received from the Terminal.
<i>FrameLength</i>	The size of the <i>Frame</i> in bytes.
<i>Data</i>	Returns a byte array containing 4 bytes of data read from the <i>Page</i> .

#### Returns

FLAGS:

- [MIFARE\\_OK](#)
- [MIFARE\\_ERROR](#)

8.2.1.3 **DWORD** mfULWriteHost ( *\_in* BYTE *Id*, *\_in* BYTE *Page*, *\_in* PBYTE *Data*, *\_out* PCHAR *Frame*, *\_out* PBYTE *FrameLength* )

Mifare UltraLight - Write page. Host frame.

#### Parameters

<i>Id</i>	ID Number of Terminal - <i>TerminalID</i> .
<i>Page</i>	Page number.
<i>Data</i>	A byte array containing 4 bytes of data to write on the <i>Page</i> .
<i>Frame</i>	Returns a byte array containing an output frame of EPSO. Ready to send.
<i>FrameLength</i>	Returns a pointer to the size of <i>Frame</i> in bytes.

#### Returns

FLAGS:

- [MIFARE\\_OK](#)
- [MIFARE\\_WRONG\\_PAGE](#)
- [MIFARE\\_WRONG\\_EPSO\\_FRAME](#)
- [MIFARE\\_ERROR](#)

#### Note

The byte array *Frame* must be large enough to return full frame of EPSO. Maximum frame size is 64 bytes

8.2.1.4 **DWORD** mfULWriteTerminal ( \_\_in BYTE *Id*, \_\_in PCHAR *Frame*, \_\_in BYTE *FrameLength* )

Mifare UltraLight - Write page. Terminal frame.

#### Parameters

<i>Id</i>	ID Number of Terminal - <i>TerminalID</i> .
<i>Frame</i>	A byte array containing the input frame of EPSO. Received from the Terminal.
<i>FrameLength</i>	The size of the <i>Frame</i> in bytes.

#### Returns

FLAGS:

- [MIFARE\\_OK](#)
- [MIFARE\\_ERROR](#)

## 8.3 Functions supporting Roger Secure Sector

### Functions

- **DWORD** [mfRssSetHost](#) ( \_\_in BYTE *Id*, \_\_in BYTE *Sector*, \_\_in PBYTE *Key*, \_\_in BYTE *KeyType*, \_\_out PCHAR *Frame*, \_\_out PBYTE *FrameLength* )
- **DWORD** [mfRssSetTerminal](#) ( \_\_in BYTE *Id*, \_\_in PCHAR *Frame*, \_\_in BYTE *FrameLength* )
- **DWORD** [mfRssDelHost](#) ( \_\_in BYTE *Id*, \_\_in BYTE *Sector*, \_\_out PCHAR *Frame*, \_\_out PBYTE *FrameLength* )
- **DWORD** [mfRssDelTerminal](#) ( \_\_in BYTE *Id*, \_\_in PCHAR *Frame*, \_\_in BYTE *FrameLength* )
- **DWORD** [mfRssReadHost](#) ( \_\_in BYTE *Id*, \_\_in BYTE *Sector*, \_\_in BYTE *Block*, \_\_out PCHAR *Frame*, \_\_out PBYTE *FrameLength* )
- **DWORD** [mfRssReadTerminal](#) ( \_\_in BYTE *Id*, \_\_in PCHAR *Frame*, \_\_in BYTE *FrameLength*, \_\_out PBYTE *Data* )
- **DWORD** [mfRssWriteHost](#) ( \_\_in BYTE *Id*, \_\_in BYTE *Sector*, \_\_in BYTE *Block*, \_\_in PBYTE *Data*, \_\_out PCHAR *Frame*, \_\_out PBYTE *FrameLength* )
- **DWORD** [mfRssWriteTerminal](#) ( \_\_in BYTE *Id*, \_\_in PCHAR *Frame*, \_\_in BYTE *FrameLength* )

### 8.3.1 Function Documentation

8.3.1.1 **DWORD** mfRssSetHost ( \_\_in BYTE *Id*, \_\_in BYTE *Sector*, \_\_in PBYTE *Key*, \_\_in BYTE *KeyType*, \_\_out PCHAR *Frame*, \_\_out PBYTE *FrameLength* )

Roger Secure Sector - Create RSS. Host frame.

**Parameters**

<i>Id</i>	ID Number of Terminal - <i>TerminalID</i> .
<i>Sector</i>	Sector number
<i>Key</i>	A byte array containing the key to the <i>Sector Trailer</i> . Size 6 bytes.
<i>KeyType</i>	Determines if the <i>Key</i> contains key type A <a href="#">MF_CLASSIC_KEY_TYPE_A</a> or type B <a href="#">MF_CLASSIC_KEY_TYPE_B</a> .
<i>Frame</i>	Returns a byte array containing an output frame of EPSO. Ready to send.
<i>Frame-Length</i>	Returns a pointer to the size of <i>Frame</i> in bytes.

**Returns**

FLAGS:

- [MIFARE\\_OK](#)
- [MIFARE\\_WRONG\\_KEY](#)
- [MIFARE\\_WRONG\\_SECTOR](#)
- [MIFARE\\_WRONG\\_EPSO\\_FRAME](#)
- [MIFARE\\_ERROR](#)

**Note**

*Sector Trailer* - last block in sector containing:

- secret key A and B,
- the access conditions for the block of that sector

The byte array *Frame* must be large enough to return full frame of EPSO. Maximum frame size is 64 bytes

8.3.1.2 `DWORD mfRssSetTerminal ( __in BYTE Id, __in PCHAR Frame, __in BYTE FrameLength )`

Roger Secure Sector - Create RSS. Terminal frame.

**Parameters**

<i>Id</i>	ID Number of Terminal - <i>TerminalID</i> .
<i>Frame</i>	A byte array containing the input frame of EPSO. Received from the Terminal.
<i>Frame-Length</i>	The size of the <i>Frame</i> in bytes.

**Returns**

FLAGS:

- [MIFARE\\_OK](#)
- [MIFARE\\_ERROR](#)



### 8.3.1.3 `DWORD mfRssDelHost ( __in BYTE Id, __in BYTE Sector, __out PCHAR Frame, __out PBYTE FrameLength )`

Roger Secure Sector - Delete RSS. Host frame.

#### Parameters

<i>Id</i>	ID Number of Terminal - <i>TerminalID</i> .
<i>Sector</i>	Sector number.
<i>Frame</i>	Returns a byte array containing an output frame of EPSO. Ready to send.
<i>Frame- Length</i>	Returns a pointer to the size of <i>Frame</i> in bytes.

#### Returns

FLAGS:

- [MIFARE\\_OK](#)
- [MIFARE\\_WRONG\\_SECTOR](#)
- [MIFARE\\_WRONG\\_EPSO\\_FRAME](#)
- [MIFARE\\_ERROR](#)

#### Note

Function does not erase data in RSS. Only change is *Sector Trailer*, which is set to default value - transport configuration

The byte array *Frame* must be large enough to return full frame of EPSO. Maximum frame size is 64 bytes

### 8.3.1.4 `DWORD mfRssDelTerminal ( __in BYTE Id, __in PCHAR Frame, __in BYTE FrameLength )`

Roger Secure Sector - Delete RSS. Terminal frame.

#### Parameters

<i>Id</i>	ID Number of Terminal - <i>TerminalID</i> .
<i>Frame</i>	A byte array containing the input frame of EPSO. Received from the Terminal.
<i>Frame- Length</i>	The size of the <i>Frame</i> in bytes.

#### Returns

FLAGS:

- [MIFARE\\_OK](#)
- [MIFARE\\_ERROR](#)

### 8.3.1.5 `DWORD mfRssReadHost ( __in BYTE Id, __in BYTE Sector, __in BYTE Block, __out PCHAR Frame, __out PBYTE FrameLength )`

Roger Secure Sector - Read RSS. Host frame.

#### Parameters

<i>Id</i>	ID Number of Terminal - <i>TerminalID</i> .
<i>Sector</i>	Sector number.
<i>Block</i>	Block number in the <i>Sector</i> .
<i>Frame</i>	Returns a byte array containing an output frame of EPSO. Ready to send.
<i>Frame-Length</i>	Returns a pointer to the size of <i>Frame</i> in bytes.

#### Returns

FLAGS:

- [MIFARE\\_OK](#).
- [MIFARE\\_WRONG\\_BLOCK](#)
- [MIFARE\\_WRONG\\_EPSO\\_FRAME](#)
- [MIFARE\\_ERROR](#)

#### Note

The byte array *Frame* must be large enough to return full frame of EPSO. Maximum frame size is 64 bytes

### 8.3.1.6 `DWORD mfRssReadTerminal ( __in BYTE Id, __in PCHAR Frame, __in BYTE FrameLength, __out PBYTE Data )`

Roger Secure Sector - Read RSS. Terminal frame.

#### Parameters

<i>Id</i>	ID Number of Terminal - <i>TerminalID</i> .
<i>Frame</i>	A byte array containing the input frame of EPSO. Received from the Terminal.
<i>Frame-Length</i>	The size of the <i>Frame</i> in bytes.
<i>Data</i>	Returns a byte array containing 16 bytes of data read from the <i>Block</i> .

#### Returns

FLAGS:

- [MIFARE\\_OK](#)
- [MIFARE\\_ERROR](#)

8.3.1.7 **DWORD** mfRssWriteHost ( *\_in* BYTE *Id*, *\_in* BYTE *Sector*, *\_in* BYTE *Block*, *\_in* PBYTE *Data*, *\_out* PCHAR *Frame*, *\_out* PBYTE *FrameLength* )

Roger Secure Sector - Write RSS. Host frame.

#### Parameters

<i>Id</i>	ID Number of Terminal - <i>TerminalID</i> .
<i>Sector</i>	Sector number.
<i>Block</i>	Block number in the <i>Sector</i> .
<i>Data</i>	A byte array containing 16 bytes of data to write on the <i>Block</i> .
<i>Frame</i>	Returns a byte array containing an output frame of EPSO. Ready to send.
<i>Frame-Length</i>	Returns a pointer to the size of <i>Frame</i> in bytes.

#### Returns

FLAGS:

- [MIFARE\\_OK](#)
- [MIFARE\\_WRONG\\_BLOCK](#)
- [MIFARE\\_WRONG\\_EPSO\\_FRAME](#)
- [MIFARE\\_ERROR](#)

#### Note

The byte array *Frame* must be large enough to return full frame of EPSO. Maximum frame size is 64 bytes

8.3.1.8 **DWORD** mfRssWriteTerminal ( *\_in* BYTE *Id*, *\_in* PCHAR *Frame*, *\_in* BYTE *FrameLength* )

Roger Secure Sector - Write RSS. Terminal frame.

#### Parameters

<i>Id</i>	ID Number of Terminal - <i>TerminalID</i> .
<i>Frame</i>	A byte array containing the input frame of EPSO. Received from the Terminal.
<i>Frame-Length</i>	The size of the <i>Frame</i> in bytes.

#### Returns

FLAGS:

- [MIFARE\\_OK](#)
- [MIFARE\\_ERROR](#)

## 8.4 Functions supporting Read Card Buffer

### Functions

- DWORD [readCardBufferHost](#) ( \_\_in BYTE *Id*, \_\_out PCHAR *Frame*, \_\_out PBYTE *FrameLength* )
- DWORD [readCardBufferTerminal](#) ( \_\_in BYTE *Id*, \_\_in PCHAR *Frame*, \_\_in BYTE *FrameLength*, \_\_out PCHAR *CardCode* )
- DWORD [readCardIDHost](#) ( \_\_in BYTE *Id*, \_\_out PCHAR *Frame*, \_\_out PBYTE *FrameLength* )
- DWORD [readCardIDTerminal](#) ( \_\_in BYTE *Id*, \_\_in PCHAR *Frame*, \_\_in BYTE *FrameLength*, \_\_out PBYTE *CardID*, \_\_out PBYTE *CardIDLength* )

### 8.4.1 Function Documentation

8.4.1.1 **DWORD** [readCardBufferHost](#) ( \_\_in BYTE *Id*, \_\_out PCHAR *Frame*, \_\_out PBYTE *FrameLength* )

Read card buffer. Host frame.

#### Parameters

<i>Id</i>	ID Number of Terminal - <i>TerminalID</i> .
<i>Frame</i>	Returns a byte array containing an output frame of EPSO. Ready to send.
<i>FrameLength</i>	Returns a pointer to the size of <i>Frame</i> in bytes.

#### Returns

FLAGS:

- [MIFARE\\_OK](#).
- [MIFARE\\_ERROR](#)
- [MIFARE\\_WRONG\\_EPSO\\_FRAME](#)

#### Note

Functions Read Card Buffer returns a unique identification number - *UID*, only the first call for a particular card. Re-call returns [MIFARE\\_NEW\\_CARD\\_NOT\\_FOUND](#). The byte array *Frame* must be large enough to return full frame of EPSO. Maximum frame size is 64 bytes

8.4.1.2 **DWORD** [readCardBufferTerminal](#) ( \_\_in BYTE *Id*, \_\_in PCHAR *Frame*, \_\_in BYTE *FrameLength*, \_\_out PCHAR *CardCode* )

Read card buffer. Terminal frame.

#### Parameters

<i>Id</i>	ID Number of Terminal - <i>TerminalID</i> .
<i>Frame</i>	A byte array containing the input frame of EPSO. Received from the Terminal.
<i>Frame-Length</i>	The size of the <i>Frame</i> in bytes.
<i>CardCode</i>	Returns a byte array containing 17 bytes of unique identification number - <i>UID</i> . The first byte containing reader localization sign (letter "R" or "T")

**Returns**

FLAGS:

- [MIFARE\\_OK](#)
- [MIFARE\\_ERROR](#)

8.4.1.3 `DWORD readCardIDHost ( __in BYTE Id, __out PCHAR Frame, __out PBYTE FrameLength )`

Read card serial number. Host frame.

**Parameters**

<i>Id</i>	ID Number of Terminal - <i>TerminalID</i> .
<i>Frame</i>	Returns a byte array containing an output frame of EPSO. Ready to send.
<i>Frame-Length</i>	Returns a pointer to the size of <i>Frame</i> in bytes.

**Returns**

FLAGS:

- [MIFARE\\_OK](#)
- [MIFARE\\_ERROR](#)
- [MIFARE\\_WRONG\\_EPSO\\_FRAME](#)

**Note**

The byte array *Frame* must be large enough to return full frame of EPSO. Maximum frame size is 64 bytes

8.4.1.4 `DWORD readCardIDTerminal ( __in BYTE Id, __in PCHAR Frame, __in BYTE FrameLength, __out PBYTE CardID, __out PBYTE CardIDLength )`

Read card serial number. Terminal frame.

**Parameters**

<i>Id</i>	ID Number of Terminal - <i>TerminalID</i> .
-----------	---

<i>Frame</i>	A byte array containing the input frame of EPSO. Received from the Terminal.
<i>Frame-Length</i>	The size of the <i>Frame</i> in bytes.
<i>CardID</i>	Returns a byte array containing card serial number - <i>CSN</i> .
<i>CardIDLength</i>	The size of the <i>CardID</i> in bytes.

### Returns

FLAGS:

- [MIFARE\\_OK](#)
- [MIFARE\\_ERROR](#)
- [MIFARE\\_WRONG\\_EPSO\\_FRAME](#)
- [MIFARE\\_NEW\\_CARD\\_NOT\\_FOUND](#)

### Warning

*CardID* may be 4 to 10 bytes depending on the type of card.

## 8.5 Functions supporting - Reader Settings

### Functions

- DWORD [readCardSettingsHost](#) (\_\_in BYTE *Id*, \_\_out PCHAR *Frame*, \_\_out PBYTE *FrameLength*)
- DWORD [readCardSettingsTerminal](#) (\_\_in BYTE *Id*, \_\_in PCHAR *Frame*, \_\_in BYTE *FrameLength*, \_\_out PCARD\_SETTINGS *CardSettings*)
- DWORD [writeCardSettingsHost](#) (\_\_in BYTE *Id*, \_\_in PCARD\_SETTINGS *CardSettings*, \_\_out PCHAR *Frame*, \_\_out PBYTE *FrameLength*)
- DWORD [writeCardSettingsTerminal](#) (\_\_in BYTE *Id*, \_\_in PCHAR *Frame*, \_\_in BYTE *FrameLength*)

### 8.5.1 Function Documentation

8.5.1.1 **DWORD** [readCardSettingsHost](#) ( \_\_in BYTE *Id*, \_\_out PCHAR *Frame*, \_\_out PBYTE *FrameLength* )

Read Card Number Settings. Host frame.

### Parameters

<i>Id</i>	ID Number of Terminal - <i>TerminalID</i> .
<i>Frame</i>	Returns a byte array containing an output frame of EPSO. Ready to send.
<i>Frame-Length</i>	Returns a pointer to the size of <i>Frame</i> in bytes.

**Returns**

FLAGS:

- [MIFARE\\_OK](#)
- [MIFARE\\_ERROR](#)
- [MIFARE\\_WRONG\\_EPSO\\_FRAME](#)

**Note**

The byte array *Frame* must be large enough to return full frame of EPSO. Maximum frame size is 64 bytes

**8.5.1.2** `DWORD readCardSettingsTerminal ( __in BYTE Id, __in PCHAR Frame, __in BYTE FrameLength, __out PCARD_SETTINGS CardSettings )`

Read Card Number Settings. Terminal frame.

**Parameters**

<i>Id</i>	ID Number of Terminal - <i>TerminalID</i> .
<i>Frame</i>	A byte array containing the input frame of EPSO. Received from the Terminal.
<i>FrameLength</i>	The size of the <i>Frame</i> in bytes.
<i>CardSettings</i>	Returns a pointer to the structure describes reader settings - <a href="#">_CARD_NUMBER_SETTINGS</a>

**Returns**

FLAGS:

- [MIFARE\\_OK](#)
- [MIFARE\\_ERROR](#)

**8.5.1.3** `DWORD writeCardSettingsHost ( __in BYTE Id, __in PCARD_SETTINGS CardSettings, __out PCHAR Frame, __out PBYTE FrameLength )`

Write Card Number Settings. Host frame.

**Parameters**

<i>Id</i>	ID Number of Terminal - <i>TerminalID</i> .
<i>CardSettings</i>	A pointer to the structure describes reader settings - <a href="#">_CARD_NUMBER_SETTINGS</a>
<i>Frame</i>	Returns a byte array containing an output frame of EPSO. Ready to send.
<i>FrameLength</i>	Returns a pointer to the size of <i>Frame</i> in bytes.

**Returns**

FLAGS:

- [MIFARE\\_OK](#)
- [MIFARE\\_ERROR](#)
- [MIFARE\\_WRONG\\_EPSO\\_FRAME](#)

**Note**

The byte array *Frame* must be large enough to return full frame of EPSO. Maximum frame size is 64 bytes

#### 8.5.1.4 `DWORD writeCardSettingsTerminal ( __in BYTE Id, __in PCHAR Frame, __in BYTE FrameLength )`

Write Card Number Settings. Terminal frame.

**Parameters**

<i>Id</i>	ID Number of Terminal - <i>TerminalID</i> .
<i>Frame</i>	A byte array containing the input frame of EPSO. Received from the Terminal.
<i>FrameLength</i>	The size of the <i>Frame</i> in bytes.

**Returns**

FLAGS:

- [MIFARE\\_OK](#)
- [MIFARE\\_ERROR](#)

## 8.6 Flags returned by library functions

- `#define MIFARE\_OK 0x00`
- `#define MIFARE\_ERROR 0x01`
- `#define MIFARE\_WRONG\_EPSO\_FRAME 0x04`
- `#define MIFARE\_NEW\_CARD\_NOT\_FOUND 0x05`
- `#define MIFARE\_WRONG\_KEY 0x06`
- `#define MIFARE\_WRONG\_BLOCK 0x07`
- `#define MIFARE\_WRONG\_PAGE 0x08`
- `#define MIFARE\_WRONG\_SECTOR 0x09`

### 8.6.1 Define Documentation



8.6.1.1 `#define MIFARE_OK 0x00`

Return OK

8.6.1.2 `#define MIFARE_ERROR 0x01`

Error

8.6.1.3 `#define MIFARE_WRONG_EPSO_FRAME 0x04`

Error - wrong data in the EPSO frame

8.6.1.4 `#define MIFARE_NEW_CARD_NOT_FOUND 0x05`

Warning - not detected the new card or no card - [readCardBufferTerminal\(\)](#)

8.6.1.5 `#define MIFARE_WRONG_KEY 0x06`

Error - wrong key or key type

8.6.1.6 `#define MIFARE_WRONG_BLOCK 0x07`

Error - wrong block number

8.6.1.7 `#define MIFARE_WRONG_PAGE 0x08`

Error - wrong page number - Mifare UltraLight

8.6.1.8 `#define MIFARE_WRONG_SECTOR 0x09`

Error - wrong sector number

## 8.7 Settings for Mifare Classic cards

- `#define MF_CLASSIC_KEY_TYPE_A 0x00`
- `#define MF_CLASSIC_KEY_TYPE_B 0x01`
- `#define VALUE_BLOCK_INCREMENT 0x00`
- `#define VALUE_BLOCK_DECREMENT 0x01`
- `#define VALUE_BLOCK_RESTORE 0x02`

### 8.7.1 Define Documentation

#### 8.7.1.1 `#define MF_CLASSIC_KEY_TYPE_A 0x00`

Access key to block - key type A

#### 8.7.1.2 `#define MF_CLASSIC_KEY_TYPE_B 0x01`

Access key to block - key type B

#### 8.7.1.3 `#define VALUE_BLOCK_INCREMENT 0x00`

Function - Mifare Classic Increment Value Block

#### 8.7.1.4 `#define VALUE_BLOCK_DECREMENT 0x01`

Function - Mifare Classic Increment Value Block

#### 8.7.1.5 `#define VALUE_BLOCK_RESTORE 0x02`

Function - Mifare Classic Increment Value Block

## 8.8 Reader configuration parameters - only RUD-3.

- enum `readSequence_e` { `skip` = 0, `readCsn` = 1, `readMsn` = 2, `readSsn` = 3 }
- enum `accessKey_e` { `useRogerKey` = 0, `useUserKey` = 1 }
- enum `outputOrder_e` { `normalOrder` = 0, `reverseOrder` = 1 }
- typedef struct `_CARD_NUMBER_SETTINGS` `CARD_SETTINGS`
- typedef struct `_CARD_NUMBER_SETTINGS * PCARD_SETTINGS`

### 8.8.1 Typedef Documentation

#### 8.8.1.1 typedef struct `_CARD_NUMBER_SETTINGS` `CARD_SETTINGS`

Describes reader settings.

#### 8.8.1.2 typedef struct `_CARD_NUMBER_SETTINGS * PCARD_SETTINGS`

### 8.8.2 Enumeration Type Documentation

#### 8.8.2.1 enum `readSequence_e`

Card number reading sequence.

**Enumerator:**

- skip*** Skip
- readCsn*** Card Serial Number
- readMsn*** Mad Serial Number
- readSsn*** Sector Serial Number

8.8.2.2 enum **accessKey\_e**

Type of key used for reading *MSN*, and *SSN*.

8.8.2.3 enum **outputOrder\_e**

Data output order.

## 8.9 Function - Roger Mifare Library Version

**Functions**

- DWORD [mifareLibVer](#) ( \_\_inout LPTSTR *Version*, \_\_in size\_t *Length* )

### 8.9.1 Function Documentation

8.9.1.1 DWORD [mifareLibVer](#) ( \_\_inout LPTSTR *Version*, \_\_in size\_t *Length* )**Parameters**

<i>Version</i>	Returns a pointer to a buffer with library version string
<i>Length</i>	The size of <i>Version</i> in bytes.

**Returns**

FLAGS:

- [MIFARE\\_OK](#)
- [MIFARE\\_ERROR](#)



## Chapter 9

# Data Structure Documentation

### 9.1 `_CARD_NUMBER_SETTINGS` Struct Reference

Describes reader settings.

#### Data Structures

- struct [\\_Csn\\_s](#)  
*Settings - Card Serial Number.*
- struct [\\_Msn\\_s](#)  
*Settings - MAD Serial Number.*
- struct [\\_Ssn\\_s](#)  
*Settings - Sector Serial Number.*

#### Data Fields

- struct [\\_CARD\\_NUMBER\\_SETTINGS::\\_Csn\\_s](#) Csn
- struct [\\_CARD\\_NUMBER\\_SETTINGS::\\_Msn\\_s](#) Msn
- struct [\\_CARD\\_NUMBER\\_SETTINGS::\\_Ssn\\_s](#) Ssn
- BYTE [UserKey](#) [6]
- enum [readSequance\\_e](#) Step [3]

#### 9.1.1 Detailed Description

Describes reader settings.

#### 9.1.2 Field Documentation

9.1.2.1 struct `_CARD_NUMBER_SETTINGS::_Csn_s`  
`_CARD_NUMBER_SETTINGS::Csn`

9.1.2.2 struct `_CARD_NUMBER_SETTINGS::_Msn_s`  
`_CARD_NUMBER_SETTINGS::Msn`

9.1.2.3 struct `_CARD_NUMBER_SETTINGS::_Ssn_s`  
`_CARD_NUMBER_SETTINGS::Ssn`

9.1.2.4 `BYTE _CARD_NUMBER_SETTINGS::UserKey[6]`

A byte array containing the key defined by user. Size 6 bytes

9.1.2.5 enum `readSequance_e _CARD_NUMBER_SETTINGS::Step[3]`

Card number reading sequence

## 9.2 `_CARD_NUMBER_SETTINGS::_Csn_s` Struct Reference

Settings - Card Serial Number.

### Data Fields

- enum [outputOrder\\_e](#) `Order`

### 9.2.1 Detailed Description

Settings - Card Serial Number.

### 9.2.2 Field Documentation

9.2.2.1 enum `outputOrder_e _CARD_NUMBER_SETTINGS::_Csn_s::Order`

Data output order

## 9.3 `_CARD_NUMBER_SETTINGS::_Msn_s` Struct Reference

Settings - MAD Serial Number.

### Data Fields

- UINT16 [Aid](#)

- BYTE [Block](#)
- enum [accessKey\\_e](#) [UsesKey](#)
- enum [outputOrder\\_e](#) [Order](#)

### 9.3.1 Detailed Description

Settings - MAD Serial Number.

### 9.3.2 Field Documentation

#### 9.3.2.1 UINT16 \_CARD\_NUMBER\_SETTINGS::\_Msn\_s::Aid

Application Directory

#### 9.3.2.2 BYTE \_CARD\_NUMBER\_SETTINGS::\_Msn\_s::Block

Block number in sector pointed by Application Directory

#### 9.3.2.3 enum [accessKey\\_e](#) \_CARD\_NUMBER\_SETTINGS::\_Msn\_s::UsesKey

Type of key used for reading

#### 9.3.2.4 enum [outputOrder\\_e](#) \_CARD\_NUMBER\_SETTINGS::\_Msn\_s::Order

Data output order

## 9.4 \_CARD\_NUMBER\_SETTINGS::\_Ssn\_s Struct Reference

Settings - Sector Serial Number.

### Data Fields

- BYTE [Sector](#)
- BYTE [Block](#)
- enum [accessKey\\_e](#) [UsesKey](#)
- enum [outputOrder\\_e](#) [Order](#)

### 9.4.1 Detailed Description

Settings - Sector Serial Number.

## 9.4.2 Field Documentation

### 9.4.2.1 BYTE\_CARD\_NUMBER\_SETTINGS::\_Ssn\_s::Sector

Sector number

### 9.4.2.2 BYTE\_CARD\_NUMBER\_SETTINGS::\_Ssn\_s::Block

Block number in sector

### 9.4.2.3 enum accessKey\_e\_CARD\_NUMBER\_SETTINGS::\_Ssn\_s::UsesKey

Type of key used for reading

### 9.4.2.4 enum outputOrder\_e\_CARD\_NUMBER\_SETTINGS::\_Ssn\_s::Order

Data output order



# Index

- [\\_CARD\\_NUMBER\\_SETTINGS, 39](#)
  - [Csn, 39](#)
    - [Msn, 40](#)
    - [Ssn, 40](#)
    - [Step, 40](#)
    - [UserKey, 40](#)
  - [\\_CARD\\_NUMBER\\_SETTINGS::\\_Csn\\_s, 40](#)
  - [\\_CARD\\_NUMBER\\_SETTINGS::\\_Msn\\_s, 40](#)
  - [Aid, 41](#)
    - [Block, 41](#)
    - [Order, 41](#)
    - [UsesKey, 41](#)
  - [\\_CARD\\_NUMBER\\_SETTINGS::\\_Ssn\\_s, 41](#)
  - [Block, 42](#)
    - [Order, 42](#)
    - [Sector, 42](#)
    - [UsesKey, 42](#)
- [accessKey\\_e](#)
  - Reader configuration parameters - only
  - [RUD-3., 37](#)
- [Aid](#)
  - [\\_CARD\\_NUMBER\\_SETTINGS::\\_Msn\\_s, 41](#)
- [Block](#)
  - [\\_CARD\\_NUMBER\\_SETTINGS::\\_Msn\\_s, 41](#)
  - [\\_CARD\\_NUMBER\\_SETTINGS::\\_Ssn\\_s, 42](#)
- [CARD\\_SETTINGS](#)
  - Reader configuration parameters - only
  - [RUD-3., 36](#)
- [Csn](#)
  - [\\_CARD\\_NUMBER\\_SETTINGS, 39](#)
- [Flags returned by library functions, 34](#)
  - [MIFARE\\_ERROR, 35](#)
  - [MIFARE\\_NEW\\_CARD\\_NOT\\_FOUND, 35](#)
- [MIFARE\\_OK, 34](#)
- [MIFARE\\_WRONG\\_BLOCK, 35](#)
- [MIFARE\\_WRONG\\_EPSO\\_FRAME, 35](#)
- [MIFARE\\_WRONG\\_KEY, 35](#)
- [MIFARE\\_WRONG\\_PAGE, 35](#)
- [MIFARE\\_WRONG\\_SECTOR, 35](#)
- [Function - Roger Mifare Library Version, 37](#)
- [mifareLibVer, 37](#)
- [Functions supporting - Reader Settings, 32](#)
  - [readCardSettingsHost, 32](#)
  - [readCardSettingsTerminal, 33](#)
  - [writeCardSettingsHost, 33](#)
  - [writeCardSettingsTerminal, 34](#)
- [Functions supporting Mifare Classic, 15](#)
  - [mfClassicReadHost, 16](#)
  - [mfClassicReadTerminal, 16](#)
  - [mfClassicWriteHost, 17](#)
  - [mfClassicWriteTerminal, 17](#)
  - [mfCreateValueBlockHost, 18](#)
  - [mfCreateValueBlockTerminal, 19](#)
  - [mfDecrementHost, 20](#)
  - [mfDecrementTerminal, 21](#)
  - [mfIncrementHost, 19](#)
  - [mfIncrementTerminal, 20](#)
  - [mfRestoreHost, 21](#)
  - [mfRestoreTerminal, 22](#)
- [Functions supporting Mifare UltraLight, 23](#)
  - [mfULReadHost, 23](#)
  - [mfULReadTerminal, 23](#)
  - [mfULWriteHost, 24](#)
  - [mfULWriteTerminal, 24](#)
- [Functions supporting Read Card Buffer, 30](#)
  - [readCardBufferHost, 30](#)
  - [readCardBufferTerminal, 30](#)
  - [readCardIDHost, 31](#)
  - [readCardIDTerminal, 31](#)
- [Functions supporting Roger Secure Sector, 25](#)
  - [mfRssDelHost, 26](#)
  - [mfRssDelTerminal, 27](#)

- mfRssReadHost, [27](#)
- mfRssReadTerminal, [28](#)
- mfRssSetHost, [25](#)
- mfRssSetTerminal, [26](#)
- mfRssWriteHost, [28](#)
- mfRssWriteTerminal, [29](#)
- MF\_CLASSIC\_KEY\_TYPE\_A
  - Settings for Mifare Classic cards, [36](#)
- MF\_CLASSIC\_KEY\_TYPE\_B
  - Settings for Mifare Classic cards, [36](#)
- mfClassicReadHost
  - Functions supporting Mifare Classic, [16](#)
- mfClassicReadTerminal
  - Functions supporting Mifare Classic, [16](#)
- mfClassicWriteHost
  - Functions supporting Mifare Classic, [17](#)
- mfClassicWriteTerminal
  - Functions supporting Mifare Classic, [17](#)
- mfCreateValueBlockHost
  - Functions supporting Mifare Classic, [18](#)
- mfCreateValueBlockTerminal
  - Functions supporting Mifare Classic, [19](#)
- mfDecrementHost
  - Functions supporting Mifare Classic, [20](#)
- mfDecrementTerminal
  - Functions supporting Mifare Classic, [21](#)
- mfIncrementHost
  - Functions supporting Mifare Classic, [19](#)
- mfIncrementTerminal
  - Functions supporting Mifare Classic, [20](#)
- mfRestoreHost
  - Functions supporting Mifare Classic, [21](#)
- mfRestoreTerminal
  - Functions supporting Mifare Classic, [22](#)
- mfRssDelHost
  - Functions supporting Roger Secure Sector, [26](#)
- mfRssDelTerminal
  - Functions supporting Roger Secure Sector, [27](#)
- mfRssReadHost
  - Functions supporting Roger Secure Sector, [27](#)
- mfRssReadTerminal
  - Functions supporting Roger Secure Sector, [28](#)
- mfRssSetHost
  - Functions supporting Roger Secure Sector, [25](#)
- mfRssSetTerminal
  - Functions supporting Roger Secure Sector, [26](#)
- mfRssWriteHost
  - Functions supporting Roger Secure Sector, [28](#)
- mfRssWriteTerminal
  - Functions supporting Roger Secure Sector, [29](#)
- mfULReadHost
  - Functions supporting Mifare UltraLight, [23](#)
- mfULReadTerminal
  - Functions supporting Mifare UltraLight, [23](#)
- mfULWriteHost
  - Functions supporting Mifare UltraLight, [24](#)
- mfULWriteTerminal
  - Functions supporting Mifare UltraLight, [24](#)
- MIFARE\_ERROR
  - Flags returned by library functions, [35](#)
- MIFARE\_NEW\_CARD\_NOT\_FOUND
  - Flags returned by library functions, [35](#)
- MIFARE\_OK
  - Flags returned by library functions, [34](#)
- MIFARE\_WRONG\_BLOCK
  - Flags returned by library functions, [35](#)
- MIFARE\_WRONG\_EPSO\_FRAME
  - Flags returned by library functions, [35](#)
- MIFARE\_WRONG\_KEY
  - Flags returned by library functions, [35](#)
- MIFARE\_WRONG\_PAGE
  - Flags returned by library functions, [35](#)
- MIFARE\_WRONG\_SECTOR
  - Flags returned by library functions, [35](#)
- mifareLibVer

- Function - Roger Mifare Library Version, [37](#)
- Msn
  - `_CARD_NUMBER_SETTINGS`, [40](#)
- Order
  - `_CARD_NUMBER_SETTINGS::_Csn_` - [s, 40](#)
  - `_CARD_NUMBER_SETTINGS::_Msn_readSsn` - [s, 41](#)
  - `_CARD_NUMBER_SETTINGS::_Ssn_` - [s, 42](#)
- outputOrder\_e
  - Reader configuration parameters - only RUD-3., [37](#)
- PCARD\_SETTINGS
  - Reader configuration parameters - only RUD-3., [36](#)
- readCardBufferHost
  - Functions supporting Read Card Buffer, skip [30](#)
- readCardBufferTerminal
  - Functions supporting Read Card Buffer, Ssn [30](#)
- readCardIDHost
  - Functions supporting Read Card Buffer, [31](#)
- readCardIDTerminal
  - Functions supporting Read Card Buffer, [31](#)
- readCardSettingsHost
  - Functions supporting - Reader Settings, [32](#)
- readCardSettingsTerminal
  - Functions supporting - Reader Settings, [33](#)
- readCsn
  - Reader configuration parameters - only RUD-3., [37](#)
- Reader configuration parameters - only RUD-3., [36](#)
  - `accessKey_e`, [37](#)
  - `CARD_SETTINGS`, [36](#)
  - `outputOrder_e`, [37](#)
  - `PCARD_SETTINGS`, [36](#)
  - `readCsn`, [37](#)
  - `readMsn`, [37](#)
  - `readSequence_e`, [36](#)
  - `readSsn`, [37](#)
  - `skip`, [37](#)
  - `readMsn`
    - Reader configuration parameters - only RUD-3., [37](#)
  - `readSequence_e`
    - Reader configuration parameters - only RUD-3., [36](#)
  - Reader configuration parameters - only RUD-3., [37](#)
- Sector
  - `_CARD_NUMBER_SETTINGS::_Ssn_` - [s, 42](#)
  - Settings for Mifare Classic cards, [35](#)
    - `MF_CLASSIC_KEY_TYPE_A`, [36](#)
    - `MF_CLASSIC_KEY_TYPE_B`, [36](#)
    - `VALUE_BLOCK_DECREMENT`, [36](#)
    - `VALUE_BLOCK_INCREMENT`, [36](#)
    - `VALUE_BLOCK_RESTORE`, [36](#)
  - Reader configuration parameters - only RUD-3., [37](#)
- Settings for Mifare Classic cards, [35](#)
  - `MF_CLASSIC_KEY_TYPE_A`, [36](#)
  - `MF_CLASSIC_KEY_TYPE_B`, [36](#)
  - `VALUE_BLOCK_DECREMENT`, [36](#)
  - `VALUE_BLOCK_INCREMENT`, [36](#)
  - `VALUE_BLOCK_RESTORE`, [36](#)
- Step
  - `_CARD_NUMBER_SETTINGS`, [40](#)
  - `_CARD_NUMBER_SETTINGS`, [40](#)
- UserKey
  - `_CARD_NUMBER_SETTINGS`, [40](#)
- UsesKey
  - `_CARD_NUMBER_SETTINGS::_Msn_` - [s, 41](#)
  - `_CARD_NUMBER_SETTINGS::_Ssn_` - [s, 42](#)
- VALUE\_BLOCK\_DECREMENT
  - Settings for Mifare Classic cards, [36](#)
- VALUE\_BLOCK\_INCREMENT
  - Settings for Mifare Classic cards, [36](#)
- VALUE\_BLOCK\_RESTORE
  - Settings for Mifare Classic cards, [36](#)
- writeCardSettingsHost
  - Functions supporting - Reader Settings, [33](#)
- writeCardSettingsTerminal
  - Functions supporting - Reader Settings, [34](#)