MIFARE CLASSIC 1K/4K USER MANUAL

Release 1.1.0

SonMicro Elektronik

Nov 07, 2017

CONTENTS

CHAPTER

ONE

INTRODUCTION

Note: This document focuses quick understanding of the MIFARE® Classic EV1 (and MIFARE® Classic) cards and their usage. For more information about the MIFARE® Classic EV1 family of tags please refer NXP Semiconductor datasheets.

The MIFARE® Classic family is the most widely used contactless smart card ICs operating in the 13.56 MHz frequency range with read/write capability and ISO/IEC 14443 A compliance. Smart cards based on MIFARE® Classic ICs are a commonly known solution in various applications such as:

- Access Control
- Public Transportation
- Electronic Toll Collection
- Loyalty Cards
- Event Ticketing
- Car Parking
- and many more...

MIFARE® Classic EV1, is succeeding the **MIFARE® Classic**, is available with the future proof 7-byte unique identifier and 4-byte non-unique identifiers.

Note: In the past MIFARE® Classic cards were limited to 4-byte UIDs only. Due to the limited number of UIDs in the single size range all new MIFARE® related products are supporting 7-byte UIDs.

MIFARE® Classic family of tags is being used in short range (up to 10 centimeters) RFID applications where higher security and fast data reading systems are required. This family of tags have fast contactless communication speed (106 Kbit/s) between the card and the reader and uses CRYPTO1, a proprietary encryption algorithm created by NXP Semiconductors. It also supports anti-collision feature so that multiple cards in the field can work.

- **MIFARE® Classic EV1 1K (MF1 S50)**
  - Available as 4-byte NUID or 7-byte UID
  - **1K EEPROM Size** (16 sectors with 4 blocks)
  - Supports credit/value operations i.e. Increment/Decrement 32-bit Signed Integer value
  - Anti-collision support
  - 2 CRYPTO1 Keys for per sector with wide variety of access conditions
  - Write Endurance: 200 000 Cycle, 10 years data retention

- **MIFARE® Classic EV1 4K (MF1 S70)**
  - Available as 4-byte NUID or 7-byte UID
  - **4K EEPROM Size** (32 sectors with 4 blocks and 8 sectors with 16 blocks)
  - Supports credit/value operations i.e. Increment/Decrement 32-bit Signed Integer value
  - Anti-collision support
  - 2 CRYPTO1 Keys for per sector with wide variety of access conditions
  - Write Endurance: 200 000 Cycle, 10 years data retention
- **MIFARE® Ultralight EV1(MF0 ULx1)**
  - 7-byte UID (serial number)
  - 384 and 1024 Bits user memory product variant (20 pages a 4-byte, 41 pages a 4-byte)
  - 32-bit OTP (One Time Programmable) area
  - Lock bits, Configurable Counters
  - Three independent 24-bit one-way counters
  - Protected data access through 32-bit password
  - Anti-collision support
  - Memory overwrite protection
  - Write Endurance: 100 000 Cycle, 10 years data retention
  - Write Endurance for one-way counters 1.000.000 cycles

MIFARE® Ultralight differs from MIFARE® Classic family. The contactless communication is not encrypted. However, it is perfectly designed for single or limited-use tickets in public transport, event ticketing (i.e. stadiums, exhibitions, leisure parks etc.) and loyalty applications. Please refer SonMicro's MIFARE® ULTRALIGHT EV1 USER MANUAL document for more information.

Attention: MIFARE® Classic 1K/4K Security

- For improved security it is strongly recommended to change the factory default keys (0x FF FF FF FF FF FF) of the unused sectors.
- For improved security it is strongly recommended to use derived authentication keys from the card UID. Thus, revealing the authentication keys of a single card does not affect overall system security.

CHAPTER

TWO

MIFARE CLASSIC MEMORY ORGANIZATION

2.1 MIFARE 1K MEMORY MAP

		Byte Number within Block																
Sector	**Block**	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	**Description**
0	[0] 0																	Manufacturer Block
	[1] 1																	Data Block
	[2] 2																	Data Block
	[3] 3	KeyA						Access Bits				KeyB						Sector Trailer 0
1	[0] 4																	Data Block
	[1] 5																	Data Block
	[2] 6																	Data Block
	[3] 7	KeyA						Access Bits				KeyB						Sector Trailer 1
...	[0]																	Data Block
	[1]																	Data Block
	[2]																	Data Block
	[3]	KeyA						Access Bits				KeyB						Sector Trailer ...
...	[0]																	Data Block
	[1]																	Data Block
	[2]																	Data Block
	[3]	KeyA						Access Bits				KeyB						Sector Trailer ...
14	[0] 56																	Data Block
	[1] 57																	Data Block
	[2] 58																	Data Block
	[3] 59	KeyA						Access Bits				KeyB						Sector Trailer 14
15	[0] 60																	Data Block
	[1] 61																	Data Block
	[2] 62																	Data Block
	[3] 63	KeyA						Access Bits				KeyB						Sector Trailer 15

Table 2.1 *MIFARE Classic 1K - Memory Organization*

- Mifare 1K EEPROM is arranged of 16 sectors. Each sector has 4 blocks and each block has 16-byte.
- Block 0 is a special read-only data block keeps the manufacturer data and the UID of the tag.
- Sector Trailer block, the last block of the sector, holds the access conditions and two of the authentication keys for that particular sector.

2.2 MIFARE 4K MEMORY MAP

		Byte Number within Block																
Sector	**Block**	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	**Description**
0	[0] 0																	Manufacturer Block
	[1] 1																	Data Block
	[2] 2																	Data Block
	[3] 3	KeyA						Access Bits				KeyB						Sector Trailer 0
1	[0] 4																	Data Block
	[1] 5																	Data Block
	[2] 6																	Data Block
	[3] 7	KeyA						Access Bits				KeyB						Sector Trailer 1
...	...																	...
...	...																	...
31	[0] 124																	Data Block
	[1] 125																	Data Block
	[2] 126																	Data Block
	[3] 127	KeyA						Access Bits				KeyB						Sector Trailer 31
32	[0] 128																	Data Block
	[1] 129																	Data Block
	[2] 130																	Data Block
	[3] 131																	Data Block
	[4] 132																	Data Block
	[.]																	Data Block
	[.]																	Data Block
	[15] 143	KeyA						Access Bits				KeyB						Sector Trailer 32
...	...																	...
...	...																	...
39	[0] 240																	Data Block
	[1] 241																	Data Block
	[2] 242																	Data Block
	[3] 243																	Data Block
	[4] 244																	Data Block
	[.]																	Data Block
	[.]																	Data Block
	[15] 255	KeyA						Access Bits				KeyB						Sector Trailer 39

Table 2.2 *MIFARE Classic 4K - Memory Organization*

- Mifare 4K EEPROM is arranged of 40 sectors. From Sector 0 to 31, memory organization is similar to Mifare 1K, each sector has 4 blocks. From Sector 32 to 39, each sector has 16 blocks.
- Block 0 is a special read-only data block keeps the manufacturer data and the UID of the tag.
- Sector Trailer block, the last block of the sector, holds the access conditions and two of the authentication keys for that particular sector.

CHAPTER

THREE

MIFARE CLASSIC BLOCKS

MIFARE 1K/4K blocks can be categorized into 4 types: Manufacturer Block, Data Blocks, Value Blocks and Sector Trailer Blocks.

3.1 MANUFACTURER BLOCK

Manufacturer block, first block (Block 0) of the first sector (Sector 0), is a read-only block holds UID and IC manufacturer data.

Remark: Generally the UID is retrieved at card activation sequence instead of reading this block directly.

3.1.1 MIFARE 1K/4K Manufacturer Block with 4-byte NUID

Byte Number within Block 0/Sector0															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NUID[0..3]				Manufacturer Data											

Table 3.1.1 *MIFARE 1K/4K Manufacturer Block with 4-byte NUID*

3.1.2 MIFARE 1K/4K Manufacturer Block with 7-byte UID

Byte Number within Block 0/Sector0															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
UID[0..6]							Manufacturer Data								

Table 3.1.2 *MIFARE 1K/4K Manufacturer Block with 7-byte UID*

3.2 DATA BLOCKS

Mifare 1K: All sectors contain 3 blocks of 16 bytes for storing data (Sector 0 contains only two data blocks and the read-only manufacturer block).

Mifare 4K: The first 32 sectors contain 3 blocks and the last 8 sectors contain 15 blocks for storing data (Sector 0 contains only two data blocks and the read-only manufacturer block).

The data blocks can be configured by the access bits present in Sector Trailer block as

- **read/write blocks** for general purpose data storage

- **value blocks** for electronic purse applications where additional commands like increment and decrement for direct control of the stored 32-bit signed value are provided.

A successful authentication has to be performed to allow any memory operation

Remark: The default content of the data blocks at delivery is not defined

Attention:

IMPORTANT

For critical applications where card data blocks are written frequently, it is recommended to backup copy of the card block in another sector and design your application accordingly, dealing with two sectors. There can be MIFARE cards from different manufacturers in the market with poor writing performance. These cards may erase the block content (full of 0x00 or 0xFF) while the write operation is interrupted during the milliseconds time frame. This may happen when the card is being moved from out of RF field while the data is being written to the flash memory of the card. Card may have no power to sustain write operation.

3.3 VALUE BLOCKS

Value blocks allow to perform an electronic purse application easier. Value blocks can be used as "credit" on the card.

- A value block is a specially formatted data block which permits error detection. Value signifies a signed 32-bit integer. Value can be written/read and incremented/decremented with the given amount.
- A value block can only be generated through a write operation in value block format. *(See CmdWriteValueBlock command in relevant Firmware Manual document)*

Byte Number within value block															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
value				~value				value				adr	~adr	adr	~adr

Table 3.3 *MIFARE 1K/4K Value Block format. Inverted sign denotes 2's complement.*

Note: Value commands: CmdWriteValueBlock, CmdReadValueBlock, CmdIncrementValueBlock, CmdDecrementValueBlock handles data block formatting automatically. Please refer Firmware & User Manual document for the related application's firmware (i.e. stdMifare V2.x.x)

Attention:

IMPORTANT

For critical applications where card data blocks are written frequently, it is recommended to backup copy of the card block in another sector and design your application accordingly, dealing with two sectors. There can be MIFARE cards from different manufacturers in the market with poor writing performance. These cards may erase the block content (full of 0x00 or 0xFF) while the write operation is interrupted during the milliseconds time frame. This may happen when the card is being moved from out of RF field while the data is being written to the flash memory of the card. Card may have no power to sustain write operation.

3.4 SECTOR TRAILER BLOCKS

The sector trailer is the last block of a sector. Sector trailer contains the

- Authentication keys A (mandatory) and B (optional)
- access conditions for the blocks of that sector. The access bits also specify the type (data or value) of the data blocks.

Byte Number within the Sector Trailer Block															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
KeyA						Access Bits				KeyB (Optional)					

Table 3.4 *MIFARE 1K/4K Sector Trailer Block.*

- If KeyB is not needed, the last 6 bytes of the sector trailer can be used as data bytes. The access bits for the sector trailer have to be configured accordingly, see *Access Bits & Conditions*
- When the sector trailer is read, the key bytes are blanked out by returning logical zeros. If KeyB is configured to be readable, the data stored in bytes 10 to 15 is returned.
- All keys are set to 0xFF FF FF FF FF FF (6 times FFh) at chip delivery and the bytes 6, 7 and 8 are set to 0xFF0780 (See *Transport Configuration*)
- Byte-9 is used for general purpose 1 byte user data. Factory default value is 0x69.

Attention:

- Access bits (stored in bytes 6,7,8) should present in the sector trailer in proper format. See Figure 4 *(Format of the Access Bits)*
  If Sector trailer is written with improper format then that sector won't be accessible again and no further read and write operation on that sector will be possible.
  Provided software tools, and SDK prepares this format automatically. However, care must be taken to make sure the proper format is used when writing to sector trailer directly with the provided write block command/API.
- It is also important user should keep record of the passwords/keys used for the sectors otherwise it is impossible to reset the card or access it again without knowing the correct key(s).

CHAPTER

FOUR

ACCESS BITS & CONDITIONS

The access conditions for every data block and sector trailer are defined by 3 bits, which are stored non-inverted and inverted in the sector trailer of the specified sector. The access bits control the rights of memory access using the secret keys A and B. The access conditions may be altered, provided one knows the relevant key and the current access condition allows this operation.

Attention: With each memory access the internal logic verifies the format of the access conditions. If it detects a format violation the whole sector is irreversibly blocked.

Access Bits	Valid Commands	Block	Description
C13,C23,C33	read/write	3	Sector Trailer
C12,C22,C32	read/write/increment/decrement	2	Data Block
C11,C21,C31	read/write/increment/decrement	1	Data Block
C10,C20,C30	read/write/increment/decrement	0	Data Block

Table 4 *Access Bits*

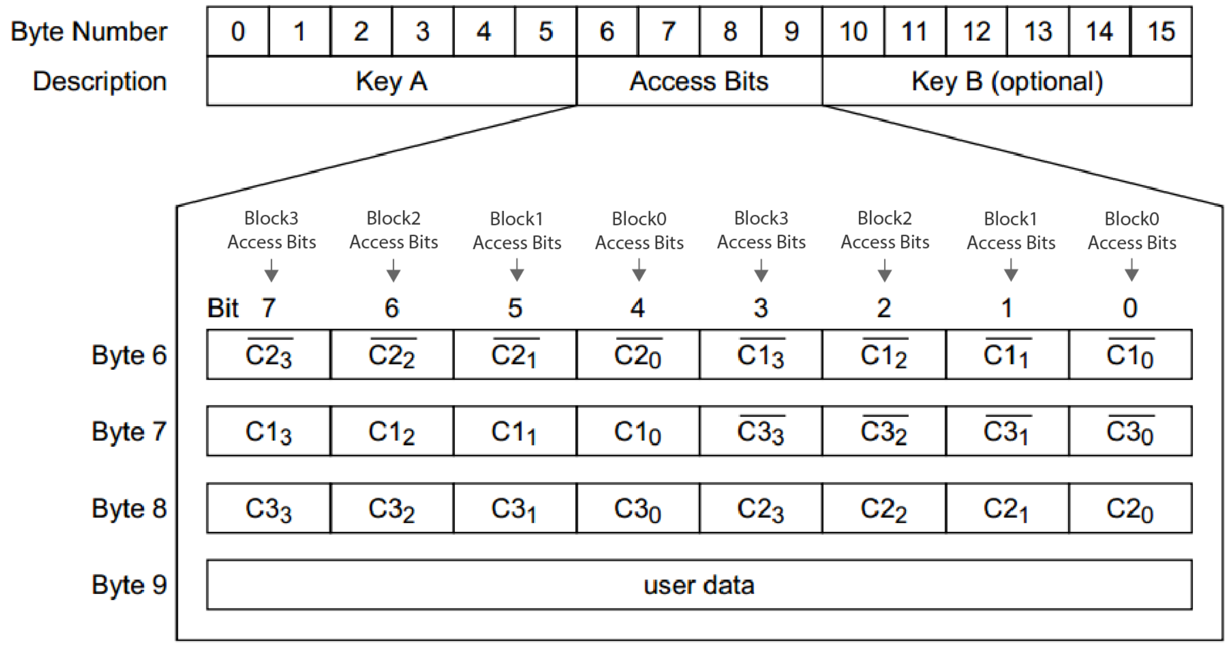

Byte Number
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Description
Key A
Access Bits
Key B (optional)
Block3 Access Bits
Block2 Access Bits
Block1 Access Bits
Block0 Access Bits
Block3 Access Bits
Block2 Access Bits
Block1 Access Bits
Block0 Access Bits
Bit 7
6
5
4
3
2
1
0
Byte 6
$\overline{C2_3}$ $\overline{C2_2}$ $\overline{C2_1}$ $\overline{C2_0}$ $\overline{C1_3}$ $\overline{C1_2}$ $\overline{C1_1}$ $\overline{C1_0}$
Byte 7
$C1_3$ $C1_2$ $C1_1$ $C1_0$ $\overline{C3_3}$ $\overline{C3_2}$ $\overline{C3_1}$ $\overline{C3_0}$
Byte 8
$C3_3$ $C3_2$ $C3_1$ $C3_0$ $C2_3$ $C2_2$ $C2_1$ $C2_0$
Byte 9
user data

Figure 4 *Format of the Access Bits*

4.1 ACCESS CONDITIONS FOR THE SECTOR TRAILER

- On chip delivery the access conditions for the sector trailers and KeyA are predefined as *Transport Configuration*
- Since KeyB may be read in the transport configuration, it cannot be used as authentication key and new cards must be authenticated with KeyA.
- Since the access bits themselves can also be blocked, special care has to be taken during the personalization of cards.

Access Bits			**Access Conditions for**						**Remark**
			KeyA		**Access Bits**		**KeyB**		
C1	**C2**	**C3**	**Read**	**Write**	**Read**	**Write**	**Read**	**Write**	
0	0	0	Never	KeyA	KeyA	Never	KeyA	KeyA	KeyB may be read[1]*
0	1	0	Never	Never	KeyA	Never	KeyA	Never	KeyB may be read[1]*
1	0	0	Never	KeyB	KeyA|B	Never	Never	KeyB	
1	1	0	Never	Never	KeyA|B	Never	Never	Never	
0	0	1	Never	KeyA	KeyA	KeyA	KeyA	KeyA	KeyB may be read, transport configuration[1]*
0	1	1	Never	KeyB	KeyA|B	KeyB	Never	KeyB	
1	0	1	Never	Never	KeyA|B	KeyB	Never	Never	
1	1	1	Never	Never	KeyA|B	Never	Never	Never	

Table 4.1 *Access Conditions for the Sector Trailer Block*

[1] - For this access condition KeyB is readable and may be used for data. KeyB cannot be used authentication key if can be read.

4.2 ACCESS CONDITIONS FOR THE DATA BLOCKS

Access conditions for each data block within the sector can be defined as in the table below.

- **Read/write block:**
    - the operations read and write are allowed.
- **Value block:**
    - Allows the additional value operations increment, decrement, transfer and restore.
    - Ex: With access condition '001' only read and decrement are possible for the relevant data block which reflects a non-rechargeable card.
    - Ex: With access condition '110' recharging is possible by using key B for the relevant data block.

Note:

- Manufacturer block: the read-only condition is not affected by the access bits setting.
- In transport configuration KeyA must be used for authentication.

Access Bits			**Access Conditions for**				**Application**
C1	**C2**	**C3**	**Read**	**Write**	**Increment**	**Decrement Transfer Restore**	
0	0	0	KeyA|B	KeyA|B	KeyA|B	KeyA|B	transport configuration[1]*
0	1	0	KeyA|B	Never	Never	Never	read/write block[1]*
1	0	0	KeyA|B	KeyB	Never	Never	read/write block[1]*
1	1	0	KeyA|B	KeyB	KeyB	KeyA|B	value block with recharging[1]*
0	0	1	KeyA|B	Never	Never	KeyA|B	value block non-rechargeable[1]*
0	1	1	KeyB	KeyB	Never	Never	read/write block[1]*
1	0	1	KeyB	Never	Never	Never	read/write block[1]*
1	1	1	Never	Never	Never	Never	read/write block

Table 4.2 *Access Conditions for the data blocks*

[1] - If KeyB may be read in the corresponding Sector Trailer it cannot serve for authentication (see [1] marked lines in Table 4.1).

4.3 TRANSPORT CONFIGURATION

Transport configuration is the name for factory default keys and configuration.

- **KeyA:** 0x FF FF FF FF FF FF (Default Key - Cannot be readable)
- **KeyB:** 0x FF FF FF FF FF FF (Default Data) KeyB is used as data in transport configuration because it is readable. It cannot be used as authentication key.
- **Access Bits:** 0xFF0780

Access Bits: 0xFF0780 meaning:

- KeyA never be read, but can write(change) itself.
- KeyA can read/write Access Bits and KeyB.
- Notice that the KeyB is read-able by KeyA. Thus, KeyB cannot be used as an authentication key. It can be used for general purpose user data.
- KeyA is allowed to read/write/increment/decrement for the data blocks.

See the next title for detailed analysis of the access bits of Transport Configuration.

Note: For basic secure applications you can use the Transport Configuration Access Bits (FF0780h) in your application by just changing the KeyA. It gives full control with a single key (KeyA) and the configuration does not lock any block access irreversibly. (Remember it is always recommended to use derived key from card UID for best practices)

For different applications, i.e. rechargeable cards, Transport Configuration may not be best option. In this case you may consider using KeyA and KeyB for different purposes i.e. recharging at credit loading stations/desks and for credit/value decrement in the field.

4.4 ACCESS BITS ANALYSIS EXAMPLE

Following example analysis the Transport Configuration Access Bits (0xFF0780):

See also Figure 4 *Format of the Access Bits*

- Byte6 = 0xFF
- Byte7 = 0x07
- Byte8 = 0x80

	Block3 Bits	Block2 Bits	Block1 Bits	Block0 Bits	Block3 Bits	Block2 Bits	Block1 Bits	Block0 Bits
Byte6	~C2(3)	~C2(2)	~C2(1)	~C2(0)	~C1(3)	~C1(2)	~C1(1)	~C1(0)
0xFF	1	1	1	1	1	1	1	1
Byte7	C1(3)	C1(2)	C1(1)	C1(0)	~C3(3)	~C3(2)	~C3(1)	~C3(0)
0x07	0	0	0	0	0	1	1	1
Byte8	C3(3)	C3(2)	C3(1)	C3(0)	C2(3)	C2(2)	C2(1)	C2(0)
0x80	1	0	0	0	0	0	0	0

Table 4.4 *Transport configuration analysis table. Inverted sign denotes 2's complement.*

Block(3)(Sector Trailer):

- C1 = 0 *C1(3)*
- C2 = 0 *C2(3)*
- C3 = 1 *C3(3)*

It's the fifth row(001) on Table 4.1 *Access Conditions for the Sector Trailer Block*

Meaning:

- KeyA never be read, but can write(change) itself.
- KeyA can read/write Access Bits and KeyB.
- Notice that KeyB is read-able by KeyA. Thus, KeyB cannot be used as Authentication Key. It can be used for general purpose user data.

Block(2)(Data Block):

- C1 = 0 *C1(2)*
- C2 = 0 *C2(2)*
- C3 = 0 *C3(2)*

Block(1)(Data Block):

- C1 = 0 *C1(1)*
- C2 = 0 *C2(1)*
- C3 = 0 *C3(1)*

Block(0)(Data Block):

- C1 = 0 *C1(0)*
- C2 = 0 *C2(0)*

- C3 = 0 *C3(0)*

All data blocks (2,1,0) are configured same, it's the first row(000) on Table 4.2 *Access Conditions for the data blocks*

Meaning:

- KeyA and KeyB can do all operations (read/write/increment/decrement) on the target data block. However, because the KeyB can be read in Block3 (Sector Trailer) configuration, it cannot be used as an authentication key.

CHAPTER

FIVE

CARD ACTIVATION & BLOCK ACCESS

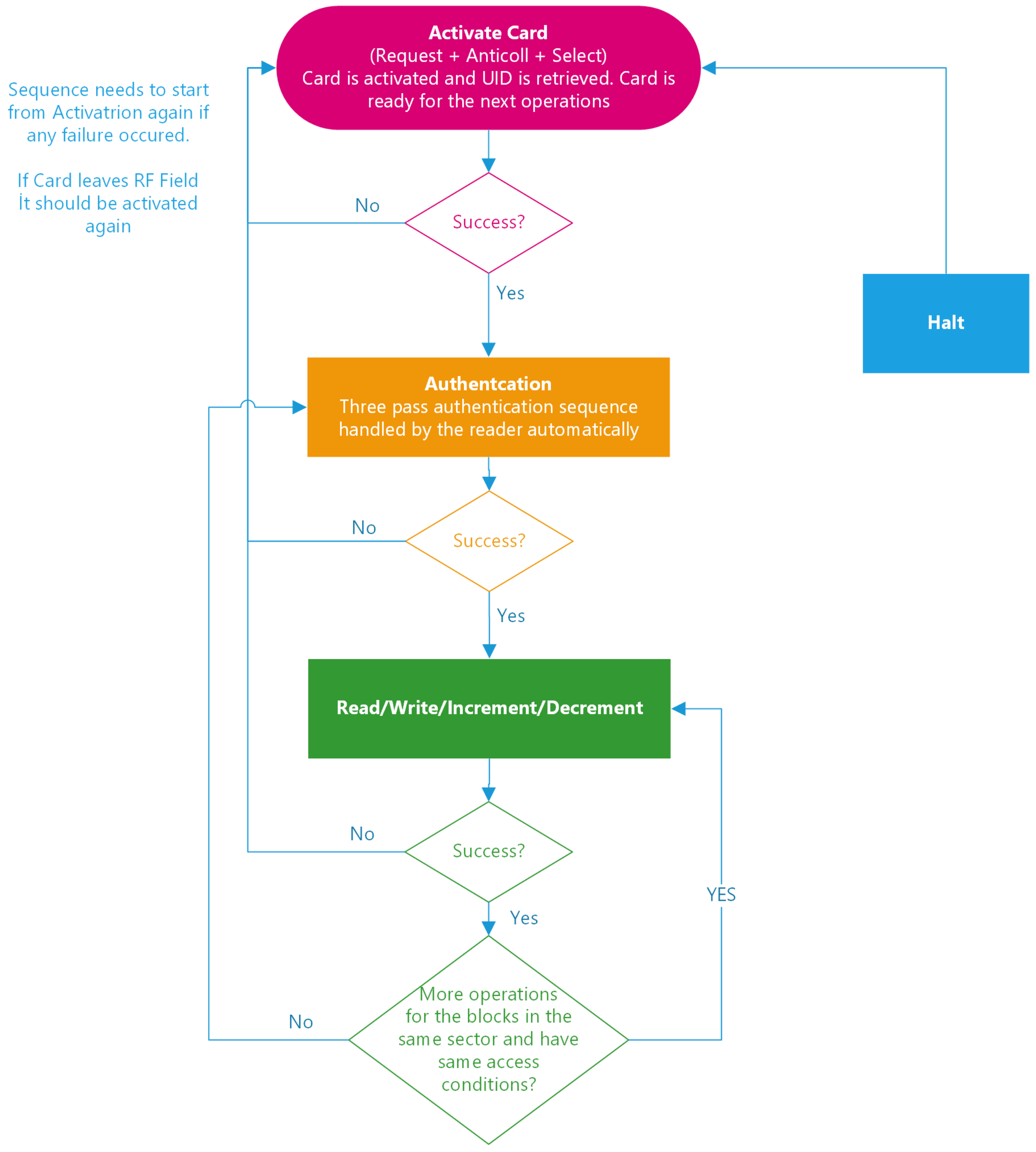
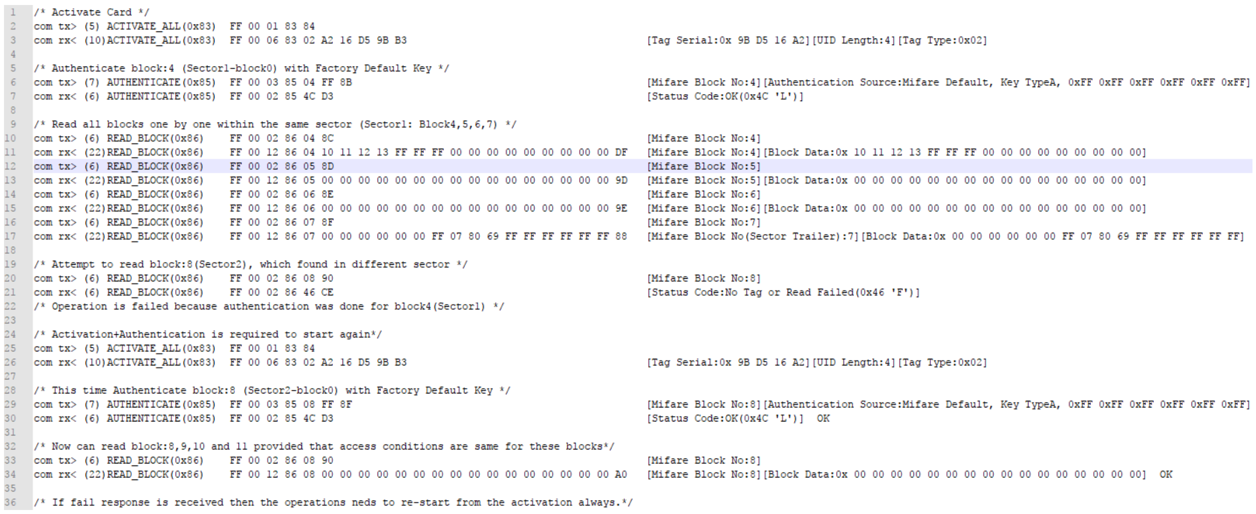

Sequence needs to start from Activatrion again if any failure occured.
If Card leaves RF Field İt should be activated again
Activate Card
(Request + Anticoll + Select)
Card is activated and UID is retrieved. Card is ready for the next operations
Success?
No
Yes
Halt
Authentcation
Three pass authentication sequence handled by the reader automatically
Success?
No
Yes
Read/Write/Increment/Decrement
Success?
No
Yes
More operations for the blocks in the same sector and have same access conditions?
No
YES

Figure 5 *Flow diagram to perform Card operations*

5.1 CARD ACTIVATION & BLOCK ACCESS EXAMPLE



```
/* Activate Card */
com tx> (5) ACTIVATE_ALL(0x83)  FF 00 01 83 84
com rx< (10)ACTIVATE_ALL(0x83)  FF 00 06 83 02 A2 16 D5 9B B3                                 [Tag Serial:0x 9B D5 16 A2][UID Length:4][Tag Type:0x02]

/* Authenticate block:4 (Sector1-block0) with Factory Default Key */
com tx> (7) AUTHENTICATE(0x85)  FF 00 03 85 04 FF 8B                                          [Mifare Block No:4][Authentication Source:Mifare Default, Key TypeA, 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF]
com rx< (6) AUTHENTICATE(0x85)  FF 00 02 85 4C D3                                             [Status Code:OK(0x4C 'L')]

/* Read all blocks one by one within the same sector (Sector1: Block4,5,6,7) */
com tx> (6) READ_BLOCK(0x86)    FF 00 02 86 04 8C                                             [Mifare Block No:4]
com rx< (22)READ_BLOCK(0x86)    FF 00 12 86 04 10 11 12 13 FF FF FF 00 00 00 00 00 00 00 00 00 DF   [Mifare Block No:4][Block Data:0x 10 11 12 13 FF FF FF 00 00 00 00 00 00 00 00 00]
com tx> (6) READ_BLOCK(0x86)    FF 00 02 86 05 8D                                             [Mifare Block No:5]
com rx< (22)READ_BLOCK(0x86)    FF 00 12 86 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 9D   [Mifare Block No:5][Block Data:0x 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00]
com tx> (6) READ_BLOCK(0x86)    FF 00 02 86 06 8E                                             [Mifare Block No:6]
com rx< (22)READ_BLOCK(0x86)    FF 00 12 86 06 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 9E   [Mifare Block No:6][Block Data:0x 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00]
com tx> (6) READ_BLOCK(0x86)    FF 00 02 86 07 8F                                             [Mifare Block No:7]
com rx< (22)READ_BLOCK(0x86)    FF 00 12 86 07 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF 88   [Mifare Block No(Sector Trailer):7][Block Data:0x 00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF]

/* Attempt to read block:8(Sector2), which found in different sector */
com tx> (6) READ_BLOCK(0x86)    FF 00 02 86 08 90                                             [Mifare Block No:8]
com rx< (6) READ_BLOCK(0x86)    FF 00 02 86 46 CE                                             [Status Code:No Tag or Read Failed(0x46 'F')]
/* Operation is failed because authentication was done for block4(Sector1) */

/* Activation+Authentication is required to start again*/
com tx> (5) ACTIVATE_ALL(0x83)  FF 00 01 83 84
com rx< (10)ACTIVATE_ALL(0x83)  FF 00 06 83 02 A2 16 D5 9B B3                                 [Tag Serial:0x 9B D5 16 A2][UID Length:4][Tag Type:0x02]

/* This time Authenticate block:8 (Sector2-block0) with Factory Default Key */
com tx> (7) AUTHENTICATE(0x85)  FF 00 03 85 08 FF 8F                                          [Mifare Block No:8][Authentication Source:Mifare Default, Key TypeA, 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF]
com rx< (6) AUTHENTICATE(0x85)  FF 00 02 85 4C D3                                             [Status Code:OK(0x4C 'L')]  OK

/* Now can read block:8,9,10 and 11 provided that access conditions are same for these blocks*/
com tx> (6) READ_BLOCK(0x86)    FF 00 02 86 08 90                                             [Mifare Block No:8]
com rx< (22)READ_BLOCK(0x86)    FF 00 12 86 08 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 A0   [Mifare Block No:8][Block Data:0x 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00]  OK

/* If fail response is received then the operations neds to re-start from the activation always.*/
```

Figure 5.1 *Communication log illustrating card activation, authentication and read block*

CHAPTER SIX

TRADEMARKS

- MIFARE® is a registered trademark of NXP B.V. and is used under license.

CHAPTER SEVEN

DOCUMENT REVISON HISTORY

Version 1.1.0 (07 Nov 2017)

Minor document fixes.

Version 1.0.0 (21 Oct 2017)

Initial release.