Advanced Card Systems Ltd.
Card & Reader Technologies

ACR120 Contactless Reader/Writer

Communication Protocol

Table of Contents

1.0. Scope

The ACR120 Serial Protocol is a reader interface that is used to communicate with ACR120 Mifare® and ISO 14443 type A and type B transponders. The major applications to be supported are:

- Access control, Identification: Reading the serial numbers of all cards in the field

- Data Storage: Performing encrypted read and write operations

- Ticketing: Performing read, write, increment and decrement operations in an encrypted environment

- Multi applications: Performing read, write, increment and decrement operations on various sectors of the Mifare® Standard tags using different encryption keys

2.0. Mifare® Data Structures

2.1. Definitions

Sector	Memory segment of the Mifare® Standard Card. Each segment consists of 4 blocks and has individual keys and access conditions. Typically in a multi-application environment each block is assigned to an application.
Key	6-byte structure assigned to each sector of the card. The reader may store up to 32 keys in its EEPROM or one key in its RAM.
Transport Key	Key as stored after delivery from the manufacturer. (i.e. `A0A1A2A3A4A5`, `B0B1B2B3B4B5` or `FFFFFFFFFFFF`)
Block	16-byte memory segment of the Mifare® Standard card.
Value	4-byte (unsigned long) variable stored in a special format in a block or page. Values are 2's complement numbers that can be negative also. Values are used for cashless payment. Values consume a complete block each using redundancy for integrity checking.
Card ID	4-byte unique serial number (single size type). Together with manufacturer code and check byte 16 bytes. Read-only. It Is stored in block 0 (sector 0) of each tag.

2.2. Mifare® Standard Card

The Mifare® Standard Card consists of 16 sectors. Each sector has 4 blocks, and each block has 16 bytes.

Mifare® Standard		Sectors
Sector 0 (Block: 0...3)		Block 8: Data or value (16 bytes)
Sector 1 (Block: 4...7)		Block 9: Data or value (16 bytes)
Sector 2 (Block 8...11)		Block 10: Data or value (16 bytes)
...		Block 11: Access Conditions, Key A,
Sector 15 (Block 60...63)		Key B (16 bytes)
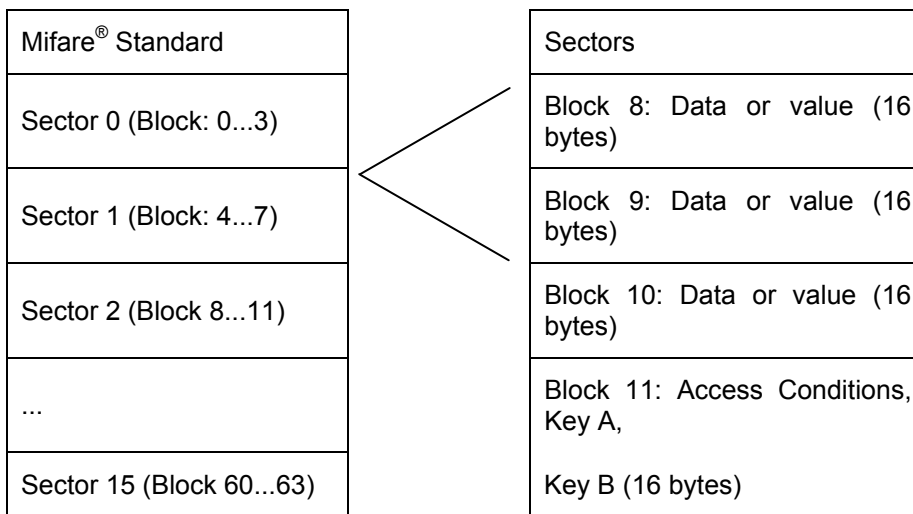
2.2.1. Sector 0 / Block 0

Serial Number (4 byte)	Check byte (1 byte)	Manufacturer data (11 byte)

This block is read only

2.2.2. Block 3, 7, 11, 15, …

Key A (6 byte)	Access Conditions (4 bytes)	Key B (6 byte)

Transport keys (keys after manufacturing, on delivery):

Key A: `A0 A1 A2 A3 A4 A5` (Infineon) or `FF FF FF FF FF FF` (new Philips cards)

Key B: `B0 B1 B2 B3 B4 B5` (Infineon) or `FF FF FF FF FF FF` (new Philips cards)

Access Conditions: `FF 07 80 xx` (key A used to read or write, the key A itself is not readable; key B is data only)

Note:

Remark 1: enabled keys read as `00 00 00 00 00 00`

Remark 2: using key B as data area will cause a security gap; due to the fact that it is necessary to rewrite key A and Access Conditions at every write access (Mifare® does only support read/write whole blocks). Therefore, this configuration is not recommended for security sensitive applications.

2.2.3. Mifare® States

The Mifare® Concept can handle multiple tags in the field and has encrypted memory access for secure cashless payment. Those features require a sequence of access as described below:
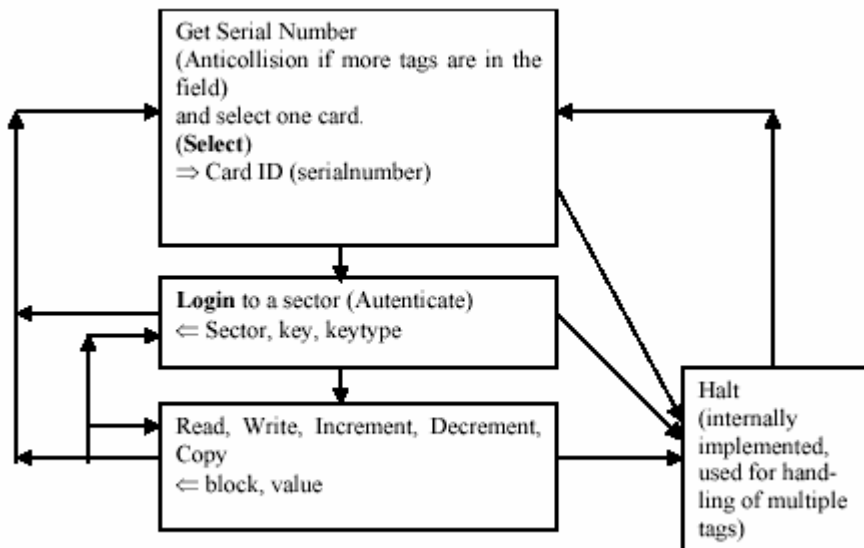
This state machine structure is mapped in application oriented reader commands:

Get Serial Number	"c" continuously reads serial numbers of all cards in the field "m<CR> displays a list of all tags in the field
Select	"s" reads a serial number and selects a single card in the field "m…<CR>" selects a specific card in the field
Login	"l" does the authentication procedure for a sector; always requires a select (using "s" or "m…") before
Read, Write, Increment, Decrement, Copy	"r", "w", "+", "-", "=" does the reading, writing and value handling; always requires a select and a login before

3.0. Software

As a default data is transmitted at 9600,n,8,1. Two protocol modes are supported. The protocol mode is configured in the reader EEPROM. As a factory default, the Binary protocol is used. For changing to ASCII Protocol see section 3.3.7.1.

3.1. ASCII Protocol

This protocol was designed for easy handling. The commands can be issued using a terminal program. Data is transmitted as ASCII Hexadecimal, which can be displayed on any terminal program (i.e. HyperTerminal).

Command	Data
M bytes	N bytes

3.2. Binary Protocol

This protocol was designed for industrial applications with synchronization and frame checking. Also, an addressing byte for party lines (master slave, multi drop) is included.

The protocol usually requires a device driver. Data is transmitted as binary, which is faster.

STX	Station ID	Length	Command/Data	BCC	ETX
1 byte	1 byte	1 byte	N bytes	1 byte	1 byte

3.2.1. STX

Start of Text (ASCII 02h)

3.2.2. Station ID

Unique ID of the station

ID 0: reserved for the bus master (controller device)

ID FFh: reserved reply all (Get ID Instruction)

3.2.3. Length

Data Length Indicator

Denotes the length of the Command/Data block

3.2.4. Command/Data

Instruction Block

The instruction block contains the command and data information. The command values are the same as in ASCII protocol mode ('x', 's', …). Data is transmitted binary instead of ASCII Hex.

The length of the command block depends on the instructions.

(Data bytes are denoted as nn in the Binary Frame Illustration in section 3.3)

3.2.5. BCC

Block Check Character

This Byte is used to detect transmission errors. It is calculated by XOR-ing each byte of the transmission frame excluding the STX/BCC and ETX character.

BCC=(Station ID) XOR (Length) XOR (Data/Command 1) XOR (Data/Command 2) XOR … XOR

(Data/Command N)

(BCC byte is denoted as cc in the Binary Frame Illustration in section 3.3)

> **Note:** If the reader receives an invalid instruction frame (i.e. BCC wrong) or the transmitted Station ID does not match the internal ID (see register table page 16) the reader resets the internal command interpreter (clear the receive buffer) and waits for the next STX transmitted by the controller.
>
> To prevent that the reader module hang up in an undefined state after receiving uncompleted instruction frames a binary timeout can be enabled (refer Protocol Configuration Register in section 3.3.7.1).
>
> The reader module answers in the same telegram format, with the ID-field set to 0. The Command/Data block of the answers in binary protocol mode does match the ASCII mode answers, with the only difference that data values are transmitted binary instead of ASCII Hex.

Examples:

02h	64h	01h	78h	1Dh	03h
STX	Station ID	Length	'x'	BCC	ETX

This instruction frame will reset the reader module with the station ID 64h. The reader module would not send an acknowledge for this command (see chapter "3.3.2 Reset")

02h	25h	02h	72h	04h	51h	03h
STX	Station ID	Length	'r'	block	BCC	ETX

This frame will read the fourth block of (the selected and authenticated) card in the field of the reader module 25h.

If successful the reader will reply for example the frame

02h	00h	0Fh	01h	00h	05h	03h	04h	02h	06h	07h	08h	0Fh	0Eh	0Bh	0Ch	0Dh	0Ah	09h	0Fh	03h
STX	Bus Master ID	Length	Data Byte Nr 0	Data Byte Nr 1	Data Byte Nr 2	Data Byte Nr 3	Data Byte Nr 4	Data Byte Nr 5	Data Byte Nr 6	Data Byte Nr 7	Data Byte Nr 8	Data Byte Nr 9	Data Byte Nr 10	Data Byte Nr 11	Data Byte Nr 12	Data Byte Nr 13	Data Byte Nr 14	Data Byte Nr 15	BCC	ETX

The Data in block 4 is: "01 00 05 03 04 02 06 07 08 0F 0E 0B 0C 0D 0A 09"

If the read operation fails because the TAG was removed the reader will reply.

02h	00h	01h	4Eh	4Fh	03h
STX	Master ID Length	'N'		BCC	ETX

3.3. Instruction Set

3.3.1. Overview

Command Byte	Mifare® Application Oriented Protocol Command	Mifare® Low Level Command
'x'	Reset	-
'c'	Continuous Read	Anticollision
's'	Select	Select
'm'	MultiTag Select / Tag list	Select / Anticollision
'l'	Login [sector, keytype, key]	Authenticate
'r'	Read [block]	Read
'rv'	Read value [block]	Read
're'	Read register [register]	-
'w'	Write [block, data]	Write
'wv'	Write value [block, value]	Write
'we'	Write register [register, data]	-
'wm'	Write key register [register, key]	-
'+'	Increment [block, value]	Increment
'-'	Decrement [block, value]	Decrement
'='	Copy [block, block]	Restore
'g'	Get ID of reader module	-
't'	Transfer data telegram [length, option, data]	various
'poff'/' '	Turn the antenna power on or off	-
'pr'/'pw	Read Write the 1 bit user Port	-

3.3.2. Reset

Command	
CMD	DATA
'x'	None
Binary Frame	02 01 01 78 78 03
Answer	
ANS	DATA
none	Binary Mode: None ASCII Mode: "ACR120 x.xx" + CR + LF, where x.xx is the major and minor version number of the firmware installed.

˙ Execute a power on (software) reset.

3.3.2.1. Start Up configuration

This command will reset the reader module and all TAGs in the field. After reset the continuous read command may be executed automatically (depending on the Protocol Configuration register).

3.3.2.2. Firmware Version

On Start Up the reader is transmitting a string (i.e. "ACR120 x.xx"). This string indicates the firmware provided by the reader module.

This string is only transmitted in ASCII mode.

This string can also be requested by the Version Request command "zv", described in section 3.3.18.

3.3.2.3. Hardware Reset using PIN 1

Alternatively you may use the Reader Enable Pin (PIN 1 of the OEM module to low) to reset the reader.

3.3.2.4. Reset Timing

The power up timing depends on environmental conditions such as voltage ramp-up. So for portables, the timing may depend on the charging state of the battery. It also may change due to future firmware version.

3.3.2.5. POWER ON Timing

1) at t=0 power is turned on

2) at t=136 msec the antenna field is activated

3) at t=155 msec the boot up message "ACR120 x.xx" is finished

4) at t=178 msec the first serial number has been sent to the PC (card touches antenna)

3.3.2.6. RESET (Pin 1) Timing

1) at t=0 reset pin 1 is released 0 V -> 5V

2) at t=75 msec the antenna field is activated

3) at t=93 msec the boot up message "ACR120 x.xx" is finished

4) at t=116 msec the first serial number has been sent to the PC (card touches antenna)

3.3.3. Continuous Read

Command	
CMD	DATA
'c'	none

Binary Frame   not supported in binary mode

Answer	
ANS	DATA
None	TAG Type Identification (1 byte) $0x01$ denotes a Mifare® Light Transponder (not supported) $0x02$ denotes a Mifare® Standard 1K Transponder MF1ICS50 $0x03$ denotes a Mifare® Standard 4K Transponder MF1ICS70 $0xFF$ denotes a unknown Transponder **Note**: The Tag type field is only available when the ExtendID bit is set in the Protocol Configuration Register (05h), as described in section 3.3.7.1. SN (4 byte) serial number of TAG

The serial number is repeated continuously while one or more tags remain in the field. This command is stopped by transmitting any character to the reader module.

3.3.3.1. Continuous read mode

By changing the "Cont. Mode" Flag in the Protocol Configuration register (see register table in section 3.3.7.1) the reader could be capable of continuously reading multiple tags.

3.3.3.2. Start Up configuration

This command is executed automatically at Start Up if the AutoStart Bit in the Protocol Configuration register is set (see register table page 17).

3.3.3.3. TAG ID Byte

This byte is only transmitted if the Extend ID in the Protocol Configuration register is set (refer table in section 3.3.7.1on page 17). Unknown Transponder types can be accessed as well if the transponder functionality is Mifare® compliant.

Disabled as factory default.

3.3.3.4. Binary mode

Continuous read is supported in ASCII protocol only. Since the binary protocol is strictly master slave with readers as slaves, it is not supported there. If the continuous read command is executed the command is only performed once.

3.3.3.5. Canceling the continuous read command

This command is not stopped if all tags leave the field. If the tags are removed while executing this command the reader stops transmitting serial numbers. If another tag enters the field before this command is cancelled (by transmitting any character) the reader module starts transmitting the new serial number.

3.3.3.6. Simple access control applications

Please consider that the serial number is not encrypted during the *Anti-collision/Select* procedure. Data encryption is activated after a successful login instruction.

For simple access control applications it is recommended to use read only blocks (see chapter 3.4 Sector Trailer and Access Conditions and datasheets of your Mifare® chip manufacturer) for the identification of the tag.

Reading any block (even the manufacturer block) of the transponder will ensure a successful Authentication procedure. This will increase your security.

3.3.4. Select

Command	
CMD	DATA
's'	none
Binary Frame:	`02 01 01 73 73 03`
Answer	
ANS	DATA
none	TAG Type Identification (1 byte) Please refer to section 3.3.7.1.1 for supported tag types
'N': no TAG	none

Binary Frame:	`02 00 01 4e 4f 03`
	`02 00 04 nn nn nn nn cc 03`
	`02 00 07 nn nn nn nn nn nn nn cc 03`
	`02 00 0a nn nn nn nn nn nn nn nn nn cc 03`

. Selects a single card and returns the card ID (Serial Number).

3.3.4.1. Select a single tag

`No previous continuous read is required.`

3.3.4.2. TAG ID Byte

This byte is only transmitted if the Extend ID in the Protocol Configuration register is set. Disabled as factory default.

3.3.4.3. Data encryption

Before a successful login instruction data encryption is inactive (serial number is transmitted as plain text).

3.3.4.4. Multiple tags

This command is designed for fast access to a single tag in the field. If you are using multiple cards in the field you have to use the 'm' instruction for the select procedure.

3.3.5. Login

Command	
CMD	DATA
'l'	sector (1 byte) 00…0F keytype (1 byte) `AA` authenticate with keytype A `FF` authenticate with keytype A, transport key `FFFFFFFFFFFF` `BB` authenticate with keytype B `10…2F` authenticate with keytype A using stored key (00..31) `30…4F` authenticate with keytype B using stored key (00..31) This parameter is optional. By transmitting <CR> (Carriage Return, ASCII 13d) instead the authentication is done with transport key 1 (`A0A1A2A3A4A5`, keytype A). key (6 bytes) By transmitting <CR> instead of the keydata authentication is done with manufacturers transport keys (`A0A1A2A3A4A5`, `B0B1B2B3B4B5`, `FFFFFFFFFFFF`).

Binary Frame:	sector 01, transport key A:
	`02 01 09 6C 01 AA A0 A1 A2 A3 A4 A5 CE 03`
	or
	`02 01 04 6C 01 AA 0D CF 03`
	or
	`02 01 03 6C 01 0D 62 03`
	with master key 0 (type A):
	`02 01 03 6C 01 10 7F 03`

Answer	
ANS	DATA
'L': login success	none
'N': no TAG	none
'F': login fail, key wrong	none
'E': invalid key format (stored key)	none

$\Rightarrow$ Performs an authentication to access one sector of the card. Only one sector can be accessed at a time. Optionally also keys stored in the reader EEPROM can be used.

To store keys in the EEPROM the write master key command ('wm') is used. It is possible to store up to 32 master keys in the reader EEPROM.

3.3.5.1. Login to a tag

Requires a select before (using the Select or MultiTag instruction).

3.3.5.2. No tag error 'N'

This means that the tag does not respond, because there is either no tag present or none of the tags in the field is selected ('s' or 'm' instruction).

> **Note:** By transmitting <CR> (carriage return, ASCII 13d) instead of an expected parameter the reader module uses standard parameters (transport keys) for the login procedure.

> Standard parameters are available for keytype and keydata (l<sector><CR>), which uses the transport key 1 (`A0A1A2A3A4A5`), or for the keydata only (l<sector>`AA`<CR>, l<sector>`BB`<CR>, l<sector>`FF`<CR>), which uses one of the three supported transport keys (key 1: `A0A1A2A3A4A5`, key 2: `B0B1B2B3B4B5`, key 3: `FFFFFFFFFFFF`).

3.3.5.3. Log in with keydata from EEPROM

Each key stored in the reader EEPROM can be used as keytype A or keytype B.

For using a specified key as keytype A the value for keytype character is the keynumber added to `10h`.

For using the keydata as key B `30h` is added to the keynumber.

So for a login with the key 0 from the EEPROM the instruction sequence is "l<sector>10" by interpreting data as key A, and "l<sector>30" by interpreting data as key B.

3.3.5.4. Usage of key A, key B

Mifare® cards support two different crypto keys for each sector. Each key is 32 bit long and stored in the sector trailer (last block of the sector) on the card. It is possible to set different access rights for the two keys. For more details of using different keys and access conditions (also stored in the sector trailer) refer the personalisation procedure.

Examples:

l01<CR>	authenticate for sector 1, using the transport key A (A0A1A2A3A4A5, keytype A)
l02AA<CR>	authenticate for sector 2, using the transport key A (A0A1A2A3A4A5, keytype A)
l3FBB<CR>	authenticate for sector 63, using the transport key 2 (B0B1B2B3B4B5, keytype B)
l04FF<CR>	authenticate for sector 4, using the transport key 3 (FFFFFFFFFFFF, keytype A)
l0FAAFFFFFFFFFFFF	Authenticate for sector 15, using key FFFFFFFFFFFF, keytype A
l0E14	Authenticate for sector 14, using EEPROM key 4, keytype A
l0530	Authenticate for sector 5, using EEPROM key 0, keytype B
l0732	Authenticate for sector 7, using EEPROM key 2, keytype B
l0110	Authenticate for sector 1, using EEPROM key 0, keytype A
l0ABBFF12FFFFFF35	Authenticate for sector 10, using key FF12FFFFFF35, keytype B

3.3.6. Read

Command	
CMD	DATA
'r' read block	block (1 byte) 00…3F (for a Mifare® Standard TAG)
'rv' read value block	block (1 byte) 00…3F
're' read reader EEPROM	register (1 byte) 00…13
Binary Frame:	read block 04h: 02 01 02 72 04 75 03 read value block 04h: 02 01 03 72 76 04 02 03 read register 10h(user data): 02 01 03 72 65 10 05 03

Answer	
ANS	DATA
none	read block: Data (16 bytes) read value: Value (4 bytes) read EEPROM: Data (1 byte)
'N': no TAG	none
'I': no value block	none
'F': read failure	none
Binary Frame:	Read block: 02 00 10 <16 bytes data> cc 03 Read value: 02 00 04 <4 bytes data> cc 03 Read EEPROM: 02 00 01 nn cc 03

$\Rightarrow$ Reads a block, value or register.

3.3.6.1. No value block 'I'

Specified block does not match the value format. The value block is corrupted; you may use a backup block for recovery (controlled by the application).

3.3.6.2. Reading blocks

The reading procedure requires a successful authentication, which requires a select and login before.

3.3.6.3. No tag error 'N'

The tag does not respond, because there is either no tag present or none of the tags in the field is not authenticated ('l' instruction).

3.3.6.4. Read failure 'F'

Additional to a data read error caused by bad transmission conditions; this error appears if you want to read a block, which is not in your currently authenticated sector.

I.e. if you try to read the block 02 (sector 0) after you just logged in into sector 1 (using the 'l01..' command).

3.3.7. Read Reader EEPROM

Reads the internal reader EEPROM. It contains non-volatile parameters.

Mifare® Reader Module EEPROM Memory Organization		
Number	Name	Description
00h…03h	unique device ID (32 bit)	This number is unique for each device and therefore read only.
04h	Station ID	Indicates the address ID for every station. The ID is used for addressing within a party line and is read only.
05h	Protocol Configuration	Set protocol type, power on behavior
06h	Baud Rate Selection	Defines communication speed
07h	Activation Duration LED/Relay	Defines the activation duration of the LED/Relay in units of 200ms in execution of portflip (pf) command.
08h	Activation Duration Buzzer	Defines the activation duration of the Buzzer in units of 200ms in execution of portflip (pf) command.
09h…0Fh	reserved	
10h…13h	user data	free usage

Station ID 0 and 255 are reserved for the bus master (controller) and the "get ID" instruction and are therefore not allowed for any reader module. As a factory default the Station ID is set to the value 01h.

It is possible to address up to 254 (01…254) different readers on a single RS485/RS422/RS232 bus.

At delivery the reader module is in ASCII protocol mode (9600,n,8,1) with AutoStart enabled and the Station ID set to 01.

> **Note:** In ASCII mode the Station ID is ignored.

> It is not recommended to use the Mifare® reader modules without an appropriate personalization in bus applications (binary protocol only).

> The register 10h to 13h are available for application specific user data and not interpreted by the reader module. The register locations 07h to 0Fh are reserved for future usage and not available for user data.

3.3.7.1. Protocol Configuration Register

Protocol Configuration Register (05h)							
Bit 8 (MSB)	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0 (LSB)
reserved	OPMODE (2 bit)		Cont. Mode	BinaryTime out	ExtendID	Binary	AutoStart

3.3.7.1.1. Extend ID: Extend TAG ID (serial number)

0 (factory default) The TAG ID Byte is not transmitted before the serial number

1 TAG ID Byte is transmitted before the serial number

This setting does only affect the commands continuous reading ('c'), select ('s') and `MultiTagSelect` ('m').

If set a unique serial number (4 bytes, 7 bytes or 10 bytes) of the transponder is extended by a single prefix byte.

The values for the prefix byte are:

TAG Type Identification:

Tag Type Value	Tag Type Description	Serial Number Length
0x01	Mifare Light	4
0x02	Mifare 1K	4
0x03	Mifare 4K	4
0x04	Mifare DESFire	7
0x05	Mifare UltrLight	7
0x06	JCOP30	4
0x07	Shanghai Transport	4
0x08	MPCOS Combi	4
0x80	ISO type B, Calypso	4

3.3.7.1.2. Binary: binary mode flag

0 (factory default) reader operates in ASCII protocol mode

1 reader operates in binary protocol mode

3.3.7.1.3. AutoStart:

0 reader is in command mode at start up

1(factory default) reader executes the command 'c' (read serial numbers continuously) at start up automatically. In binary protocol mode this bit is ignored (The binary protocol does not support continuous reading).

3.3.7.1.4. BinaryTimeout

0 (factory default) Binary Time-out disabled

1 Binary Time-out enabled

This flag is only interpreted if the reader operates in binary mode.

If the serial bus stays idle for more than 96 ms (no data is transmitted), the reader will clear its command buffer ant enter "Command Read" mode.

"Command Read" mode means that the reader is waiting for valid data frames (beginning with the STX code).

Example:

You transmit the sequence "`02h 00h FFh`" to a third party module (a door opener, turnstile control unit, …).

Since this sequence starts with a valid Mifare® reader protocol but does not terminate properly, the Mifare® reader would wait infinitely. Termination can be enforced using the binary timeout.

3.3.7.1.5.   Cont. Mode (Continuous Read Mode)

0 (factory default) continuous reading does only support single tags

1 continuous reading does support multiple tags

Using single tag reading is much faster than using multiple tag reading. If in single mode more than one card is in the field the reader does transmit none or only one of the serial numbers (depending on reading conditions and transponder positioning).

3.3.7.1.6.   OPMODE (Operation MODE, default 0 (ISO-A))

This 2 bit denotes the operation mode used at start up.

0x00	ISO-A mode
0x01	ISO-B mode
0x02	ISO-AB (toggle) mode
0x03	RFU

It is possible to change operation mode without changing these bits (and rebooting the reader) by using the Switch operation mode instruction.

3.3.7.2.   Baud Rate

Baud Rate Selection (06h)							
Bit 8 (MSB)	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0 (LSB)
reserved	reserved	reserved	reserved	reserved	BS2	BS1	BS0

3.3.7.2.1.   BS1, BS0

BS2	BS1	BS0	Baudrate
0	0	0	9600 baud (factory default)
0	0	1	19200 baud
0	1	0	38400 baud
0	1	1	57600 baud
1	0	0	115200 baud
1	0	1	9600 baud
1	1	0	9600 baud
1	1	1	9600 baud

Communication is always 8 data bits, no parity, 1 stop bit. At delivery the Communication speed is set to 9600 baud.

To take over changes in any of these registers the reader must be reset. It is recommended to clear reserved bits for ensure compatibility with further firmware versions.

Examples:

r05	reads block 4 (sector 1)
00112233445566778899AABBCCEEDDFF	reply from reader if Mifare® Standard block 5 contains "001122…"
r00	reads Manufacturer Code (sector 0)
rv04	reads value of block 4
00112233	reply from reader if Mifare® Standard value block 4 contains "00112233"
re04	reads register 4 (Station ID)
01	reply if Station ID is set to 01
re05	reads register 5 (Protocol Configuration)
01	reply if Protocol Configuration register is set to 01
re06	reads register 6 (Baud Rate Selection)
03	reply if baud rate is set to 57600 baud

3.3.8. Write

Command	
CMD	**DATA**
'w' write block	block (1 byte) `00…3F` data (16 bytes)
'wv' write value block	block (1 byte) `00…3F` value (4 bytes)
'we' write register	register (1 byte) `00..0F` data (1 byte)
'wm' write master key	key number (1 byte) `00…1F` key (6 bytes)

Binary Frame:	write block `04h`, data `001122…`:
	 `02 01 12 77 04 00 11 22 33 44 55`
	 `66 77 88 99 AA BB CC DD EE FF 60`
	 `03`
	 write value block `04h`, value `00112233`:
	 `02 01 07 77 76 04 00 11 22 33 03`
	 `03`
	 write register `10h` (user data), data `AAh`
	 `02 01 04 77 65 10 AA AD 03`
	 write master key 0, keydata `A0A1A2A3A4A5`:
	 `02 01 09 77 6D 00 A0 A1 A2 A3 A4`
	 `A5 13 03`

Answer	
ANS	**DATA**
None	write block: written data (16 bytes) write value: written value (4 bytes) write register: written value (1 byte) write master key: written key (6 bytes)
'X': unable to read after write	none
'U': read after write error	none
'N': no TAG	none
'F': write failure	none
'I': write failure	none
Binary Frame:	Write block: `02 00 10` (16 bytes of data) `cc 03` Write register: `02 00 01 nn cc 03`

⇒ Writes a block or sets a value. A read after write is performed automatically.

3.3.8.1. Writing blocks

The writing procedure requires a successful authentication, which requires a select and login before.

3.3.8.2. No tag error 'N'

This means that the tag does not respond, because there is either no tag present or none of the tags in the field is not authenticated (login instruction).

3.3.8.3. Invalid value 'I'

The value read back after the write value instruction is a not a value block. Data was written corruptly. In almost all cases you can handle this similar to a write failure 'F'.

3.3.8.4. Unable to read after write 'X'

TAG was removed from field immediately after the write instruction.

Data was written but the TAG did not response to the Read after Write Instruction (which is done automatically by the reader module). Data stored on the TAG may be corrupted.

3.3.8.5. Read after write error 'U'

After each write access to the TAG, a read is done automatically. This error occurs if the data read does not match the written one. This is caused when data was not stored on the TAG or if the sector is protected (i.e. keys of sector trailers are read protected and will always cause a read after write error although written correctly).

3.3.8.6. Write failure 'F'

Additional data write error caused by bad transmission conditions, this error appear if you want to write a block which is not in your currently authenticated sector.

i.e. if you try to read the block 02 (sector 0) after you just logged in into sector 1 (using the 'l01..' command).

3.3.8.7. Writing values

The write value block command is designed to create blocks, which match the value format. This command requires write access to specified block. It is not recommended to use this instruction for ticketing operations. For ticketing algorithms special instructions (Increment/Decrement/Copy) are supported.

3.3.8.8. Writing to the EEPROM

Writing to the internal reader EEPROM takes approximately 3 ms per byte. If the supply voltage is turned off while the reader is programming, data in the EEPROM may be corrupted. The reader is transmitting the write acknowledge (6 bytes key data) after successful programming of the EEPROM.

Reset reader to activate parameter changes.

3.3.8.9. Master keys

Master keys are not readable. Due to data security reasons master keys are stored redundant, consuming 12 bytes per key. The reader module allows to store up to 32 master keys. They are stored non-volatile.

3.3.8.9.1. Writing master keys

Due to the fact that master keys are stored in a write only area the reader module is not able to do a

read after write for this sections.

Nevertheless the reader does check if the programming cycle was executed correctly. Error codes are returned also.

To prove the master key stored is valid a login instruction (by using the EEPROM key, i.e. lxx20 and an appropriately formatted card) is needed.

3.3.8.9.2. Writing to the Sector Trailer

Because of some parts of the sector Trailer are write only (key areas) writing to the Sector Trailer will cause a "read after write" ('U') error.

If a part of the Sector Trailer is write protected (i.e. key A area), dummy data is required for a successful write operation.

Examples:

w08000102030405060708090A0B0C0D0E0F	writes a data stream (000102…) to block 8 (sector 2)
wv05010055EF	writes value 010055EF to block 5.
we0464	write the value 64h to the STATIONID register
we0502	set the reader in binary mode (write 02h into the Protocol Configuration register)
wm00112233445566	store key 112233445566 in EEPROM (key number 0)
wm02A0A1A2A3A4A5	store transport key 1 in EEPROM key 2
we0600	sets the reader to 9600 baud

3.3.9. Increment

Command	
CMD	**DATA**
'+'	Block (1 byte) 00…3F Value (4 bytes) 32 Bit value
Binary Frame:	Increment Block 04h, value 01010102 02 01 06 2B 04 01 01 01 02 2B 03

Answer	
ANS	DATA
None	new Value (4 bytes)
'X': unable to read after increment none	
'N': no TAG	none
'I': no value block none	
'F': increment failure	none

$\Rightarrow$ increments a value block by adding a value to the value stored. A read after write is performed automatically. Call fails, if source block is not in value format.

3.3.9.1. Incrementing values

Requires a select and login before.

3.3.9.2. No value block 'I'

Specified block does not match the value format.

Value block is corrupted; you may use a backup block for recovery (controlled by the application).

3.3.9.3. Unable to read after increment 'X'

TAG was removed from field immediately after the increment instruction.

Data was incremented but the TAG did not response to the read after increment instruction (which is done automatically by the reader module).

3.3.9.4. No tag error 'N'

This means that the tag does not respond, because there is either no tag present or none of the tags in the field is not authenticated ('l' instruction).

Examples:

+0400000001	adds 1 to value block 4
+0500000100	adds 256 to value block 5

3.3.10. Decrement

Command	
CMD	**DATA**
'-'	Block (1 byte) `00...3F` Value (4 bytes) 32 Bit value

Binary Frame	Decrement Block `04h`, value `01010102`
	 `02 01 06 2D 04 01 01 01`
	 `02 2D 03`

Answer	
ANS	DATA
none	new Value (4 bytes)
'X': unable to read after decrement	none
'N': no TAG	none
'I': no value block	none
'F': decrement failure	none
'E': decrement failure, empty	none

$\Rightarrow$ decrements a value block by subtracting a value to the value stored. A read after write is performed automatically. Call fails, if source block is not in value format.

3.3.10.1. Decrementing values

Requires a select and login before.

3.3.10.2. No tag error 'N', no value block 'I'

Same as Increment.

3.3.10.3. Unable to read after decrement 'X'

TAG was removed from field immediately after the decrement instruction.

Data was decremented but the TAG did not response to the read after decrement instruction (which is

done automatically by the reader module).

3.3.10.4. Decrement failure 'E'

The error return 'E' denotes, that the value block does not contain sufficient value to decrement. Some card types will return 'F' instead.

On a 2s complement representation the smallest value is `80000000h`, not `00000000h`!

Examples:

| -040000044B | subtract 1099 (=`0000044Bh`) from value block 4 |
| -0500000100 | subtract 256 from value block 5 |

3.3.11. Copy

Command	
CMD	**DATA**
'='	source block (1 byte) target block (1 byte)
Binary Frame	Copy value Block 4 to block 5 `02 01 03 3D 04 05 3E 03`
Answer	
ANS	DATA
none	new Value of target block (4 bytes)
'X': unable to read after copy	none
'N': no TAG	none
'I': no value block	none
'F': copy failure	none

 copies a value block to another block of the same sector. A read after write is performed automatically. Used for backup and error recovery.

3.3.11.1. Copy blocks

Requires a select and login before.

3.3.11.2. Target block

The target block need not be a valid value block. Call fails, if source block is not in value format.

3.3.11.3. Access conditions

This command requires the same access conditions than the decrement instruction.

3.3.11.4. Unable to read after copy 'X', no tag error 'N'

Same as Increment decrement.

3.3.11.5. No tag error 'N' no value block 'I'

Same as increment/decrement, but only source block has to be a value block, target block is overwritten.

Examples:

| =0405 | copy value block 4 to block 5 |
| =0506 | copy value block 5 to block 6 |

3.3.12. Turn on/off antenna power

Command	
CMD	**DATA**
'poff' 'pon'	none
Binary Frame	poff: `02 01 04 70 6F 66 66 1A 03` pon: `02 01 03 70 6F 6E 73 03`

Answer	
ANS	DATA
'P'	none

⇒ This command turns OFF or ON the antenna power. This reduces power consumption of the module.

3.3.12.1. 'pon'

A "power on" is done automatically by the reader if necessary (before TAG access commands like 's'/'m').

After executing the Power On command the reader is operational instantly.

3.3.12.2. 'poff'

Please keep in mind that turning off the antenna power will reset all TAGs in the field and therefore a new "select", and "login" is required before card access. It is not recommended to use this option while the host is processing data from the card (i.e. between the read and decrement instruction).

3.3.12.3. Sequential usage

Neither does a 'poff' instruction does affect the reader if the field is already turned off nor does the 'pon' if the antenna field is turned on.

3.3.13. Read/Write/Flip User Port

Command	
CMD	**DATA**
'pr' 'pw' 'pf'	none data (1 Byte)
Binary Frame:	Port Read: `02 01 02 70 72 cc 03` Port Write: `02 01 03 70 77 nn cc 03` Port Flip: `02 01 03 70 66 nn cc 03`

Answer	
ANS	DATA
	user port state (1 Byte)

⇒ This command returns the state of the user port (PIN 14 of the OEM module). This port may be

used as 1 bit I/O port.

3.3.13.1. Read port

(not available in the standard version of ACR120S)

3.3.13.2. Write port

For ACR120S, the user port is connected to the LED indicator. Changing the port will actually change the LED indicator's on/off state.

For ACR120S-SM, a relay is tied to the LED control. An additional control is made available for controlling the on board buzzer. This function sets the states of Relay (together with LED) and Buzzer.

> **Note**: The LED state of some readers may have been tied to indicate operation status by software option in factory default. In this case, the user may not be able to change the Relay/LED independently. To release this tie, please use the Write EEPROM message (3.3.8) to write a value of `0x00` to a special EEPROM address of `0xFE` then do a power reset to the reader. Doing this operation only once is enough to change the option permanently.

Examples:

For ACR120S and ACR120SM:

pw00	Turns the LED/Relay OFF and Buzzer OFF
pw01	Turns the LED/Relay ON and Buzzer OFF
pw02	Turns the LED/Relay OFF and Buzzer ON
pw03	Turns the LED/Relay ON and Buzzer ON

3.3.13.3. Flip port

This function is added to ease the LED/Relay flipping and Buzzer sounding operation. The Write port command only turns ON or OFF of the corresponding devices according to the user port state, it could be difficult for the controlling application program to time the activation duration precisely. This function call activates the LED/Relay and Buzzer for a precise duration defined in EEPROM values in address `0x07` and `0x08` respectively, in units of 200 ms. This function will not take any action when called if the value is zero (`0x00`) in the respective EEPROM locations.

Examples:

(Assume EEPROM location `0x07` and `0x08` are `0x0A` and `0x04`)

For ACR120S and ACR120SM:

pf00	No action
pf01	Activates the LED/Relay for 2 seconds (0x0A x 200ms)
pf02	Activates the Buzzer for 0.8 seconds (0x04 x 200ms)
pf03	Activates the LED/Relay and Buzzer for 2 and 0.8 seconds respectively.

* The Flip Port function is only available in firmware version 24S or later.

3.3.14. Get ID

Command	
CMD	**DATA**
'g'	none
Binary Frame:	`02 FF 01 67 99 03`
Answer	
ANS	DATA
none	Station ID (1 byte)

$\Rightarrow$ This command is used to get the Station ID's for all reader modules on the bus.

3.3.14.1. Time slotted answer

To avoid bus collisions, the answer of each reader is transmitted in a different timeslot. The Station ID of each device denotes the timeslot (see table below). In binary mode the Station ID is packed into a correct data frame (STX 0 1 ID BCC ETX), in ASCII mode data is terminated by CR+LF.

The length of a timeslot depends on the selected baud rate.

This command supports 256 different timeslots. After the last timeslot (256*T0) this command is done and all available readers should have transmitted their serial number.

$$T_0[s] = \frac{10}{Baudrate} * 6$$

It is not supported to use different baud rates on the same bus.
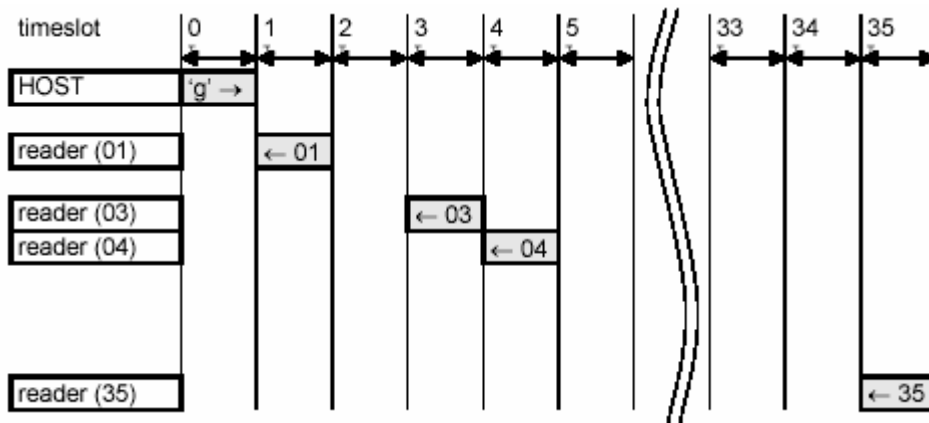
3.3.14.1.1. ASCII mode

If the reader is used in ASCII protocol mode the personalization procedure is not necessary. In ASCII mode the station ID does not affect reader's functionality (even if set to an reserved value like 0 or `FFh`).

3.3.15. Multi-Tag Selection

Command	
CMD	**DATA**
'm'	serial number (0, 4, 7 or 10 bytes)+<CR>
Binary Frame	Tag List: `02 01 02 6D 0D 63 03` Select Tag (F0 9F 34 08): `02 01 05 6D F0 9F 34 08 3A 03`

Answer	
ANS	DATA
none	TAG Type Identification (1 byte) Please refer to section 3.3.7.1.1 for supported tag types
'N': no TAG	none

⇒ This command is used instead of the Select ('s') instruction, when using multiple tags in the antenna field.

The serial number indicates the tag, which is selected. All other tags are not affected by this instruction.

3.3.15.1. m<CR>

By transmitting the carriage return code instead of the 4, 7 or 10 byte serial number the reader listens up the serial numbers of all tags which are in readable antenna range, terminated by a single data byte which indicates the number of tags found.

Each serial number is terminated by a CR/LF sequence in ASCII mode or put into a particular STX-ETX data frame in binary mode.

This command is finished after the reader transmits a single byte (the number of tags found) to the application host.

If the tag search is not successful, the reader will reply "00" (zero tags found) at command termination.

3.3.15.2. Reading distance

As each card in the field affects the antenna tuning, and reduce field strength the reading distance is reduced for each new transponder added to the antenna field.

Depending on the physical construction of each tag the interference between these tags may also affect reading quality.

Use this command instead of the select ('s') instruction if more tags are in reading range. This command replaces the 's' instruction, therefore a valid login sequence for multiple tags is:

m<CR>:	To display a tag list (you may alternatively use the 'c' instruction for a single tag).
m81635640<CR>:	To Select a specific tag.
l01<CR>:	To login into sector 1

After the MultiTag selection all other tag interfacing commands ('l', 'r', 'w', '+', …) are the same.

3.3.15.3. Maximum number of tags

The maximum number of tags in the field is only limited by physical characteristics of your antenna design.

The implementation detects up to 17 tags.

Examples:

m<CR>	tag list
00112233 85431557 81BF565D 03	reply from reader (three tags found)
m85431557 <CR>	select second tag (SN: 85431557)
85431557	reply: select successful

3.3.16. Transfer data telegram

<table>
<tr><td colspan="2" align="center">Command</td></tr>
<tr><td align="center">CMD</td><td align="center">DATA</td></tr>
<tr><td>'t'</td><td>length code (1 byte)
option byte (1byte)
data stream (various)</td></tr>
<tr><td>Binary Frame</td><td>REQA: <code>02 01 04 74 01 E3 26 B5 03</code>
Get ID: <code>02 01 05 74 02 03 93 20 C2 03</code> (single Tag)
Read Block 4: <code>02 01 05 74 02 0F 30 04 49 03</code></td></tr>
</table>

<table>
<tr><td colspan="2">Answer</td></tr>
<tr><td>ANS</td><td>DATA</td></tr>
<tr><td>none
'N'</td><td>received bytes (1 byte)
received data bytes (various)
none</td></tr>
</table>

$\Rightarrow$ This command is used to transfer user specific data frames to an ISO 14443 (Part 2) compliant Transponder, such as Mifare® PRO.

This command is extreme low level. If using Mifare® Light or Mifare® Standard cards there is absolutely no need for implementing this instruction in your application.

It is not recommended to use this command without detailed low-level knowledge of ISO 14443 protocols.

3.3.16.1. Length code

Number of Bytes you want to transmit. In Binary mode this byte is transferred too for compatibility reasons even though it could be calculated with the binary frame length code.

Due to internal buffer ranges the maximum length byte is 34 (maximal frame length is therefore 39 bytes).

If the transponder/host does transmit more than 34 bytes further data is ignored.

3.3.16.2. Option byte

These bytes holds transfer options.

Bit 0:   if set Parity generation is enabled

Bit 1:   if set Parity is odd, otherwise Parity bit is even

Bit 2:   if set CRC generation for transmission is enabled

Bit 3:   if set CRC checking for receiving is enabled

Bit 4:   if set Crypto unit is deactivated before transmission start

                Activation of the Crypto unit is only possible by using the login ('l') instruction

Bit 5,6,7:   Bit Framing (Number of Bits from last Byte transmitted)

3.3.16.3. CRC generation

CRC generation described in ISO 14443-3 Appendix B.

3.3.16.4. Receiving answer

Immediately after transmitting the reader is set into receive mode. If receiving no data the reader returns 'N' to the application host. Due to this timeout it is not recommended to use instructions with no response. In this case a acknowledge byte from the card may be used to speed up transmission (receiving a single byte is much faster than the timeout).

Examples (in ASCII mode)

select sequence for single tags in the field:

t01E326	answer:	020400 (REQA)
t02039320	answer:	05**81635640F4** (GetSN)
t070F9370**81635640F4**	answer:	0188 (Select card 81635640F4)

read block 4: (after login)

command code is `0x30`

t020F**30**04 answer: 10010203...

this means:

2 bytes transmitted, option byte set to `0x0F`, data bytes `0x30, 0x04` (read command code and block number)

It is not possible to activate the Crypto unit using the 't' instruction.

So if using a Mifare Classic compatible Operating system on a tag it is not possible to do an authentication (login).

This must be done using the regular login-instruction ('l').

3.3.17. Low Level Register Read/Write

This is a very low-level command, which changes the register of the contactless reader chip. This command should only be used through the advice of the reader manufacturer. Improper command may cause the reader to cease functioning permanently.

Command	
CMD	**DATA**
'zr'	Register Number (1 byte) 00…3F
'zw'	Register Number (1 byte) 00…3F Data (1 Byte)
Binary Frame:	Port Read: `02 01 03 7a 72 nn cc 03` Port Write: `02 01 04 7a 77 nn nn cc 03`
Answer	
ANS	DATA
	user port state (1 Byte)
Binary Frame:	Port Read: `02 00 01 nn cc 03` Port Write: `02 00 01 nn cc 03`

3.3.18. Request Version Information

This command returns the current firmware version of the reader.

Command	
CMD	**DATA**
'zv'	none
Binary Frame:	`02 01 02 7a 76 0f 03`
Answer	
ANS	DATA
none	A string indicating the current model and firmware version. "ACR120 x.xx", where x.xx is the major and minor version number of the firmware installed
Binary Frame:	`02 00 nn` <model & version identifier> `0d 0a cc 03` e.g. `02 00 0d 41 43 52 31 32 30 20 31 2e 33 31 0d 0a 54 03`

3.3.19. Alters Default Frame Waiting Index (FWI)

This function alters the default Frame Waiting Index (FWI) which the ISO 14443 cards reported during the initial card operation. The value of the reader is adopted through the Request for ATS operation in type A cards (sending $0xe0$, $0x00$ through Transfer data telegram command) and the Select operation in type B cards. In some instances, the frame waiting time may need to extend to wait for certain computation intensive operations on the card, which the card will request for a Waiting Time Extension (WTX) inside the ISO 14443 part 4 communication.

Command	
CMD	**DATA**
'zf'	data (1 Byte)
Binary Frame:	`02 01 03 7A 66 nn cc 03`
Answer	
ANS	DATA
	The new FWI
Binary Frame:	`02 01 03 7A 66` <new FWI> <checksum> `03` e.g. `02 01 03 7A 66 07 19 03`

The actual waiting time FWT is calculated by the following formula:

FWT = (256 * 16 / 13560000) * (2 ^ FWI)

which gives 4.94s if FWI = 14

This function is only available in firmware version 24S or later and should only be used by advanced programmers who understand the operation of the ISO14443 card accessing procedures.

3.4. Sector Trailer and Access Conditions

The last block of each sector contains configuration data for the sector. This configuration data includes key A, key B and the access conditions. The first six byte (byte 0…5) of the Sector Trailer contain key A data, the last six bytes (byte 10…15) contain key B data.

Byte number 6 to byte number 9 contains the access conditions for each block of the sector. It is possible to configure the access rights (read, write, increment, decrement, restore) different for each block in the sector and in dependence to the key used in the authentication (login) procedure.

Access conditions for the Sector Trailer himself are different form access conditions of a data block (increment, decrement, copy is never allowed for the Sector Trailer as of course this block never contains value data). The access conditions are stored redundant for data security reasons.

Examples for Sector Trailers / Access Conditions

Ticketing Applications

For ticketing applications it is recommended to use both keys of the Mifare® card. Key A as a field key with rights for read, copy and decrement only. Key B is used as master key with full access rights (including increment and changing the access conditions and keys).

Data Handling Applications

For data handling Applications it is recommended to disable ticketing operations (increment, decrement, copy). Key A is user as slave key with reading rights only. Key B is used as master key with read/write access to all blocks.

No Security, open configuration

For open configuration applications it always possible not to change the sector trailer (`FF 07 80 xx`) and use the configuration as defined by the card manufacturer.

Key A is set as master key with full access rights and key B is disabled.

Please consider that enabled keys are not readable and therefore return 00 on reading.

It is possible to configure each block of one sector as Value Block (Ticketing) or Data Block. As an example you may uses following values for the access rights.

Block 0	Block 1	Block 2	Sector Trailer
V	V	V	08 77 8F FF
V	V	D	48 77 8B FF
V	D	V	28 77 8D FF
V	D	D	68 77 89 FF
D	V	V	18 77 8E FF
D	V	D	58 77 8A FF
D	D	V	38 77 8C FF
D	D	D	78 77 88 FF

Where "D" denotes a data block and "V" value block. All access conditions are configured that way that key B has write access to the Sector Trailer and so may change the configuration.

Each of the 16 sectors consists of 4 blocks (including the sector trailer). Block 0 of sector 0 contains the serial number and some manufacturer data, it is read only.

For detailed description of the Sector Trailer /Access Conditions please refer to the datasheets of your chip manufacturer. As it is possible to destroy the tag (permanently make a block read and write protected) it is strongly recommended not to change the Sector Trailer without detailed knowledge and under safe environment with good reading/writing only.

Examples:

To use a whole sector number 4 for data storage with key A (read only key) set to 001122334455 and key B (read/write key) set to 66778899AABB the following instruction sequence should be used.

"s" select a single TAG

"l04<CR>" login to the sector (or appropriate login)

"w13001122334455787788FF66778899AABB" write keys and access conditions

This command will cause a "read after write" error, due to the fact that key areas are read only.

"poff" turn off the field (reset TAG)

"s" reselect the TAG

"l04…" do login with new keys (test writing)

To set the first two blocks of a sector for ticketing (i.e. value and backup block) and the third block to data mode and the keys same as above set the sector trailer to the value: "00112233445548778BFF66778899AABB"

4.0. Timing
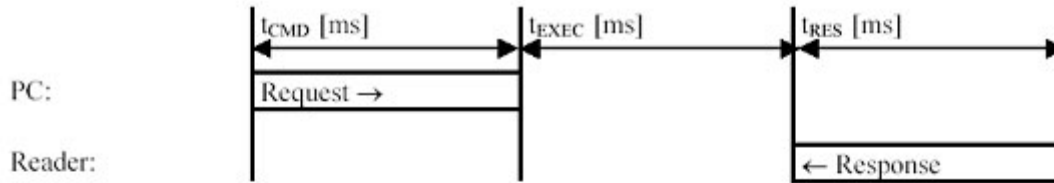
Command	Command			Execution			Response		
	t_{MIN}[1]	t_{TYP}[1]	t_{MAX}[1]	t_{FAIL}[2,3]	t_{SUCC}[4]	t_{NOTG}[5]	t_{MIN}[1]	t_{TYP}[1,4]	t_{MAX}[1]
Reset		1,0			67,6			30,0[6]	
Cont. Read		1,0		-[7]		∞[8]		10,4	
Select[9]		1,0		7,6	15,0	26,0	3,1	10,4	10,4
Multi Tag	2,1	2,1	5,2		various			various	
Login	4,2		17,7	16,6	5,4	15,0		3,1	
Read									
Data [r]		4,2		1,3	3,6	13,4	3,1	35,4	35,4
Value [rv]		4,2		3,8	3,8	13,4	3,1	10,4	10,4
EEPROM [re]		3,1			1,0			1,0	
Write[10]									
Data [w]		36,5		1,3	11,2	13,4	3,1	35,4	35,4
Value [wv]		12,5		1,3	11,2	13,4	3,1	10,4	10,4
EEPROM [we]		4,2			9,6			1,0	
Master Key [m]		9,4			115,0		1,0	6,3	6,3
Power ON		3,1			<0,3			1,0	
Power OFF		4,2			<0,4			1,0	
Port									

[1] This values are defined by tranmission speed (RS232 @ 9600/19200/38400/57600,n,8,1 ~ 1,041/0,52/0,26/0,17 ms/character). The timing data in this table is calculated for 9600 baud. No delay between cherechters is assumed and the timing data is calculated for ASCII mode.
When using binary mode this values will increase due to the binary mode frame, which requieres more data tranmitted (STX, …).

[2] Execution time on faulty communication or data failure (f.e. key wrong, no value block, access to unauthenticated sector). Timing data in this table is typical error detection time and will vary with error type.

[3] If the a data byte of the command instruction does not match the ASCII Hex Format (f.e. "r0G") the reader module is responding almost immediately (<170us) after the second nibble of the faulty data byte by transmitting a question mark, carriage return and linefeed ("?<CR><LF>").

[4] Execution time on operation success.

[5] Execution time on "no Tag"/timeout-error (max. processing time)

[6] In binary Mode the reader module does not send a response to the reset command. Nevertheless the reader is executing a power on reset and ready again in approximately 68 ms.

[7] If no TAGs is present the reader will wait until a TAG enters the field. If the TAGs remains in the field the serial numbers are repeated continuously.

[8] This command will not terminate automatically

[9] Select includes a card reset before (10 ms).

[10] All write operations include a read after write.

write [w]		3,1			<0,1			1,0	
read [r]		2,1			<0,1			1,0	
Get ID		1,0			various			1,0	
Transfer Telegram		various			various			various	
Increment[10]		11,5		1,3	18,0	13,4	3,1	10,4	10,4
Decrement[10]		11,5		1,3	18,0	13,4	3,1	10,4	10,4
Copy[10]		3,1		3,6	14,5	13,4	3,1	10,4	10,4

All values are ms. Grey marked cells are fixed values due to the fact that this instructions have a constant instruction/response length.

All timing data is advisory application information and does not form part of the specification. It may change in further firmware releases.

5.0. Frequently Asked Questions

5.1. Getting Started

To test and interface the Mifare® OEM Module, you do not need a sophisticated uP development system. All you need is a PC, a connection cable and a power supply for the reader. If you are using Windows 95/98/NT, take the following steps:

- Make sure, that your reader is RS232-interface type
- Start HyperTerminal
- Create a new connection (FILE/NEW CONNECTION)
- Enter name of connection as you like (i.e. 'Mifare')
- Select connect COM2 (COM1) direct connection
- Connection setup 9600,8,n,1,no handshake
- Connect your reader to COM2 (COM1) of the PC and apply appropriate the supply voltage. A string (i.e. "Mifare V 0.14") is transmitted to the PC by the reader. This String denotes the firmware provided by your reader module
- Put a tag to your reader. Serial numbers should be displayed properly
- Enter commands via keyboard. They should be transmitted to the reader and the reader should reply

5.2. How does ticketing work with Mifare®?

To get a quick impression, connect the reader to a terminal program, take a new card and try the following steps:

- Put the card in the field. The terminal program should show continuously the serial numbers of the card, for example "D1635640".
- Enter space. The transmission of serial numbers should stop.
- Enter "s" for select. A Mifare® card always has to be selected, before it can be accessed.
- Enter "l01<ENTER>" for login to sector 01. This uses key A and the transport key `A0A1A2A3A4A5`. Alternatively you can type in "L01AAA0A1A2A3A4A5", specifying that you want to use key, A which is A0A1A2A3A4A5 on a new card. A login is always needed before a sector can be accessed. For new Philips cards use "l01FF<ENTER>" since they have `FFFFFFFFFF` as transport key.
- Now you can access block 04, 05, 06, 07, which are on, sector 01. If you enter "w04000123456789AABBCCDDEEFFDDEE0375" then the value `000123456789AABBCCDDEEFFDDEE0375` gets written to block 04. To read it, enter "r04".
- To format block 04 as a value block and store 1500 points (`1500dec=000005DCh`) enter "wv04000005DC".
- To use up 100 points (`100dec=00000064h`) enter "-0400000064"
- To backup the value into block 05 enter "=0405"
- You also can add to the values on the card. To charge 500 points (`500dec=000001F4h`) enter "+04000001F4".

5.3. We would like to use Mifare® for cashless payment. How safe is it?

Security is always a property of the overall system, not of the components. It requires careful design.

A properly designed system will require **ALL** barriers to be hacked in order to be broken.

For good design start specifying feasible attacks. Then create barriers to block them.

Mifare® was specifically designed for cashless payment applications. The Mifare® concept provides following barriers:

- Anti-collision/-selection
- Atomic value transaction
- Ciphered communication
- Storage of values and data protected by mutual authentication
- Weak field keys that allow decrement only
- Stored keys in the reader that are not readable
- Keys in the card that are not readable
- A brute force attack by trying different keys is limited by the transaction time (several msec) of the card and would last virtually forever
- Etc.

5.3.1. The Application can and should provide more barriers

5.3.1.1. Sector access conditions

It is possible to assign access conditions in a way that only decrementing of values is allowed with the keys used in the field, so that even a manipulated field station cannot be used to charge cards with additional values. As a rule, key A is used as a field key, allowing decrement and read only, and key B to format the card or charge values.

Ensure these rules even for unused sectors!

5.3.1.2. Diversified keys

To make life even harder for attackers, keys can be modified using serial number and memory content of the card. So each card uses different keys and a listening attack on the reader interface would be hopeless.

5.3.1.3. Further improvements

- Limiting cash volume stored on a card
- Do not use the transport keys (keys as programmed after delivery) for ticketing applications!
- Ciphered and scrambled data storage
- Sabotage alarm
- Etc.

5.4. What happens if somebody pulls the card out of the field during a transaction?

Modifying memory content of a Mifare® card is an EEPROM write operation internally. It requires a sufficient energy supply to execute properly. If a card leaves the field during an EEPROM write, corrupted data may be left. However, a sophisticated transaction scheme inside the Mifare® tag reduces the chance of this happening significantly; maybe you will never encounter it in your tests. Incrementing or decrementing is safer than doing read-write explicitly. Additionally, the application can be designed in such a way that each value block is mirrored in a backup block. This allows automatic recovery of lost data resulting in very reliable systems. However, carefully designing and testing the application is recommended.

5.5. What type of Mifare® card should I use?

Mifare® Light was designed as a lean solution for a single application environment. It contains only one sector with 2 keys, access condition, 2 data blocks and one value block.

Mifare® Standard was designed for a multi-application environment. It contains 16 sectors each with 2 individual keys, access conditions, 3 data or value blocks. Some applications use the 1 KByte of the Mifare® Standard Card Memory just as storage.

5.6. How should Mifare® cards are personalized?

It depends on the application. Each sector of the Standard Mifare® card can be devoted to a specific application.

5.7. Identification, access control, data storage applications

Mifare® cards are delivered with transport keys (see section 2.2) and can be used already with those keys. Also the access conditions need not to be changed.

5.8. Cashless payment, ticketing, metering

Cashless payment is based on value blocks that can be incremented and decremented. Value blocks are generated easily using the Write Value command.

A sector containing a value block should also contain a backup block to restore values after a write crash. After each transaction the value block should be copied into the backup using the copy command. Care has to be taken, that a sector with a value block does not contain other value blocks. Otherwise a security gap arises from the possibility of copying value blocks within a sector.

In security sensitive applications it is recommended to change the access conditions and keys. Keys can be modified using the write command. It is strongly recommended to modify the sector trailer containing access conditions and keys (block 3, 7, 11, …) in a stable environment with good reading only. Otherwise sectors with badly formatted keys or access conditions never can be accessed again.

For a cashless payment block the recommended access conditions are:

08 77 8F xx

Which means, that key B (master key) can write values, access conditions and keys and key A (field key) can read and decrement values only.

6.0. References

ISO/IEC 14443 Part 1-4, Identification Cards – Contactless integrated circuit(s) cards – Proximity cards