NFC & MIFARE & ISO14443A&B & ISO15693 & ISO18000 & ISO7816 IC CARD MODULE

QM-ABCM7 Series IC Card Module

General Technical Manual

(Revision 6.21)

Quick Ohm Küpper & Co. GmbH

February 17, 2022

Please read this manual carefully before using. If any problem, please feel free to contact us, we will offer a satisfied answer ASAP.

Contents

Document update records

Revision	Date	Update information
V4.51	May. 15, 2015	Support MIFARE Ultralight EV1
V4.62	Jan. 15, 2015	Modify MIFARE Plus and MIFARE DESFire translation. Add sample to SR series cards.
V4.63	Jan. 29, 2015	Modify spelling errors.
V5.00	Feb. 2, 2015	Module authentication 5.3 added
V5.10	Mar. 18, 2016	Support encryption communication, chapter 5.3.4
V5.20	Mar. 24, 2016	Support NFC functions, chapter 5.14
V5.21	Apr. 21, 2016	Support NFC Tag functions, chapter 5.15
V5.23	Jun. 7, 2016	Modification grammar.
V5.30	Sep. 30, 2016	Support FeliCa
V5.31	Apr. 22, 2018	Reset module settings to default build in a reset operation Support multi-block read of Ultralight series
V5.50	Nov.28 2018	Added SPI com port
V5.52	Sep.23 2019	Support ISO15693 Inventory for all tags Support ISO14443A request for all cards
V5.55	Oct.9 2019	Modification of MIFARE PLUS command format
V6.00	July 7, 2020	Support ISO18000-3M3
V6.02	July 15, 2020	Fix spell errors
V6.10	February 1, 2021	Add command for none EMV standard smart card. Chapter 5.4.3 & 5.4.4
V6.20	May 6, 2021	Add Chapter 5.2.22 and Fix spell errors
V6.21	November 29, 2021	Correct Chapter 5.2.19

1 Introduction

This file describes “working modes” and “communication protocol” of QM-ABCM7 series IC card module in details. It's suitable for the programmers who are using Series QM-ABCM7 RFID module to do the development.

If you have any question during the programming, please feel free to contact our technical support via kontakt@quio-rfid.de.

2 Function

QM-ABCM7 series IC card module is high efficient product. We have built in a lot of advanced functions. However, if you are still not satisfied with these functions, please contact us.

2.1 Basic Mode

QM-ABCM7 is a slave device; the "ask & answer" is the basic working mode loop. That means if QM-ABCM7 receives a command from master machine, it will execute the command and answer to the master machine. This is a command cycle. A new command won't be accepted while the module is executing a command. So when you develop the application program, you MUST be sure the last command cycle is finished in order to send the next command.

2.2 Automatically Detecting Card

QM-ABCM7 supports automatic detecting cards that are based on ISO14443A and ISO15693. When the automatic detecting cards function is open, QM-ABCM7 will continuously send searching card commands. Once the card enters into the effective electrical field, the ICC pin will show low level. In this situation, you could directly do operation to the card. If the automatic detecting function is close, when you want to do operation to the card, you need to send searching card command firstly.

The default automatic detecting cards function could be set via 0x1D command. This setting is saved in FLASH, which means that this setting will be effective on next power on. While module is working, the automatic detecting cards function could be temporarily open or closed via 0x11 command. However, this setting won't be saved. So after turning on the power again, it will be back to the default setting.

If the module operate card mode (via 0x70 command) is set to ISO15693, then QM-ABCM7 will just only auto detect ISO15693 cards. If the module operate card mode (via 0x70 command) is set to ISO14443A, then QM-ABCM7 will just only detect ISO14443A cards. While automatic detecting card is open, any other type of card operation command will change the module operate card mode, and results in the fail of the automatic detecting.

Automatic detecting cards function support ISO15693 cards.

Automatic detecting cards function support MIFARE 1K/4K and MIFARE Ultra Light cards.

ISO14443A T=CL card could be detected when the automatic detecting cards function is open. If you need to operate the card, you need to send RATS command (0x30) to the module firstly. After the module got successful response from the CPU card, then the automatic detecting cards function will be closed automatically, please notice the information above.

Automatic detecting cards functions won't report card detected while multi same type cards are in the field. The card data may be erroneous, if multi cards are in the electric field. And also multi cards operation will be forced to close when the automatic detecting cards function is open.

2.3 Automatically detecting card and outputing the card UID

Users could set automatic detecting card and output the card UID. Under this mode, the card's serial number could be output from UART or RS232C ports when swiping the card. The supported RF protocols are ISO14443A and ISO15693. The card output method could be single time or continuously until the card is moved away from the RF antenna field.The output format could be HEX or ASCII. In ASCII, it can only output UID; HEX output format is according to 0x20 and 0x5C commands return format.

When entering into automatically detecting card mode, the read/write card could not be operated because the card will enter into halt status once when the card is detected. If need to read/write card, automatically output the card UID function must be shut temporarily via 0x11 command and then go on with the read/write card operations. For the temporarily setting, you could refer to "Module working mode set".

This function couldn't work in I²C, USB and SPI interfaces. Once received the command from I²C, USB or SPI interfaces, the “Automatically detecting card and outputing the card SNR" will be prohibited.

2.4 Module Idle Status

This function design is aim to get low power consumption of the module. It isn't suitable for USB interface. In idle mode, the module of RF output will turn OFF, so the power consumption will reduce to be about 3mA. Sending the next following command to module will wake up the module, and then the RF output will be ON. Please refer to "Set Module idle".

2.5 Module Disable Mode

Some of the QM-ABCM7 modules have CE pins. When CE pin is in low level, the module is working. But if CE pin is in high level, the module will enter into disable mode, so the power consumption will reduce to about 30 to 100uA (depending on the RF chip, RC522/RC523 is lower consumption). Letting the CE pin be in low level again, the module will be woken up. The waking up time is about 50ms.

2.6 EMV/PBOC Certification Mode

Some of the QM-ABCM7 modules have PBOC and EMV certification modes. These modes support EMV& PBOC (LEVEL 1) protocol certification and electrical characteristics certification. This mode could be open via sending the commands.

We are capability of designing the Reader which could pass the EMV/PBOC certification. If you need the help, please feel free to contact us.

2.7 Operations Default Setting

QM-ABCM7 modules support the following default operations: setting automatically detecting card ON/OFF; setting automatically output card SNR ON/OFF; setting automatically output card SNR ONE TIME/CONTINUOUS; setting SNR output by HEX/ASCII; setting under automatically detecting mode, ISO15693 card could be set AFI; setting automatically detecting card interval time; setting multi-card operation ON/OFF; setting UART communication address and baud rate; setting I^{2}C communication address; setting RF output power. About the detailed above commands, please refer to “Module reset to factory default".

2.8 ISP Function

QM-ABCM7 supports In System Program. The firmware of QM-ABCM7 could be updated via USB or UART/RS232C interfaces.

2.9 Reader Authentication

Quio Rfid offers a method to authenticate reader module in order to ensure that our customer is using Quio Rfid products. It is especially useful in some security sensitive environment.

NDA (None Distribute Agreement) is needed to obtain the detail operation steps. Please contact us to sign the NDA.

2.10 Encryption Communication

Encryption communication is useful in data sensitive environment. Quio Rfid offers an encrypted communication method to protect the sensitive data.

NDA (None Distribute Agreement) is needed to obtain the detail operation steps. Please contact us to sign the NDA.

2.11 NFC Target Functions

Some PCD supports NFC functions, like PN512, and the reader module could support NFC target functions. This mode could support active mode too. It could work with other NFC devices which support active mode. If the initiator only supports passive mode, passive mode is the only mode could be chosen.

2.12 NFC Tag Simulation Functions

The reader module of supporting NFC target mode could be simulated as a NFC Tag. It is a FLASH data storage space with 128 bytes. The memory space organization is according to Ultralight card, 4 bytes as a page, total 32 pages.

The data in the NFC Tag could be initializing over communication port.

In order to WRITE to NFC Tag over RF interface, users MUST use the special command of Quio Rfid reader module.

3 Communincation Protocol

3.1 Test Software

We supply test software "TransPort”. New users could use this tool to understand the protocol of JCP04 & JCP05. The reader module could be directly connected with the PC via RS232C or USB HID interface. And we supply MT500 (JMY IC Card Module Tester) for test I^2C and UART interface modules. MT500 could convert RS232C signal to UART or I^2C signal to operate the module. Please contact us for details.

Please refer to our sample code while developing application. For I^2C, time sequence is very important to get the Max. communication speed.

3.2 JCP04 Communication Protocol

JCP04 communication protocol is used in early products. JCP05 communication protocol is improved based on JCP04. QM-ABCM7 supports both JCP04 and JCP05 communication protocols. We recommended using JCP05 communication protocol in new products development.

3.2.1 Data sent format

Length	Command	Data	Checksum

- Length: 1 byte, number of bytes from Length byte to the last byte of Data.
- Command: 1 byte, Application-layer command, please refers to Application-layer protocol in detailed.
- Data: length depends on the command type, from 0x00 to 0xFC bytes.
- Checksum: 1 byte, Exclusive OR (XOR) results from length byte to the last byte of data.

3.2.2 Data returned format

- Success:

Length	Command	Data	Checksum

- Failure:

Length	Invert Command	Checksum

NOTE: "Failure" means that the communication between module and card failed.

3.3 JCP05 Communication Protocol

3.3.1 Data sent format

- Host send:

Length	C.A.	Command	Data	Checksum

- Length: 2 bytes, number of bytes from Length byte to the last byte of Data, MSB first, length from 0x0004 to 0x01FE.
- C.A. (communication address): 1 byte, the address of UART or RS232C multi-device communication, default address: 0x01; broadcast address: 0x00.
- Command: 1 byte, Application-layer command, please refers to Application-layer protocol in detailes.
- Data: length depends on the command type, length from 0 to 510 bytes; depending on the processor, and some models will be less than 510 bytes.
- Checksum: 1 byte, Exclusive OR (XOR) results from length byte to the last byte of data.

3.3.2 Data returned format

- Success:

Length	C.A.	Command	Data	Checksum

- Failure:

Length	C.A.	Invert Command	Checksum

NOTE: "Failure" means that the communication between module and card failed.

3.4 Data Returned Time

Slaves begin to execute the command once received host's commands. The executive time is normally less than 100ms, depending on the command type. Some command executive time maybe longer (say 60s, e.g. some CPU cards to generate a key pair command). The waiting time is decided by the card and the command type. So the hosts need to set the waiting time according to the different commands.

4 Communication Interface

4.1 UART and RS232C interface

4.1.1 Physical Interface

The Universal Asynchronous Receiver/Transmitter (UART) takes bytes of data and transmits the individual bits in a sequential fashion. And the interface level is TTL-level specification. The communication between UART and PC is via TX, RX and GND pins. When using, the host's TX pin should be connected to the device's RX pin; at the same time the host's RX pin should ne connected to the device's TX pin.

RS232C is the UART interface which belongs to RS232C-level. The communication between RS232C and PC is via TXD, RXD and GND pins. When using, the host's TXD pin should be connected to the device's RXD pin, at the same time the host's RXD pin should be connected to the device's TXD pin.

The communication protocol is byte oriented. Both sending and receiving bytes are in hexadecimal format. The communication parameters are as follows:

Baud rate: 19200bps (default), 115200bps, 9600bps, 38400bps and 57600bps.
Start bits: 1 bit
Data bits: 8 bits
Stop bits: 1 bit
Parity check: None
Flow control: None

4.1.2 Communication Process

Host sends command to the slave, and the slave will execute the command once received it; then the slave will send the result to the host. This is a command cycle.

It is meaningless to issue a new command before receiving the return message from the previous command

4.2 I²C Interface

4.2.1 Physical Interface

I²C interface is a two-wire synchronous serial communication interface which belongs to TTL-level specification. It uses only two bidirectional open-drain lines, Serial Data Line (SDA) and Serial Clock (SCL), and could be pulled up with resistors 2.7K~10K to choose. The pulling up is very important to get a stable communication and faster speed. The host's SDA pin should connect to the device's SDA pin; meanwhile the host's SCL pin should connect to the device's SCL pin.

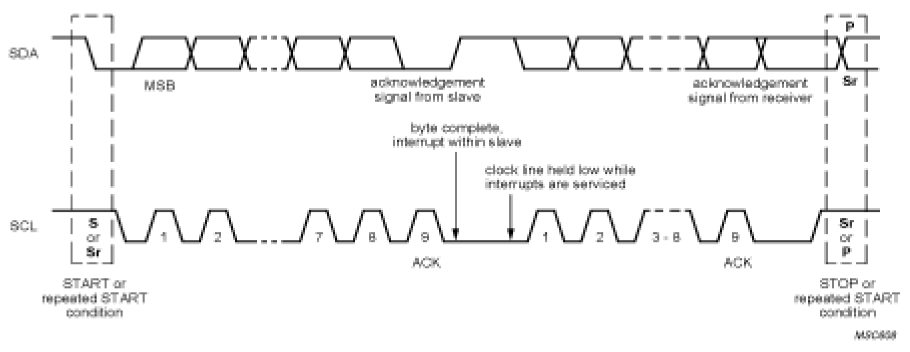
I² C bus is able to connect with 128 devices. The I² C address of module is default 0xA0. Users may change the address setting via sending the command (0x19), so that user could connect multi modules on the same I² C bus.

4.2.2 Data Links

The following is I²C communication link. It shows JCP04 protocol data format.

4.2.2.1 Clock and Data Transaction

The SDA pin is normally pulled high with an external device. Data on the SDA pin may change only during SCL low time periods. Data changes during SCL high periods will indicate a start or stop condition as defined below.


SDA
MSB
acknowledgement signal from slave
acknowledgement signal from receiver
P
Sr
byte complete, interrupt within slave
clock line held low while interrupts are serviced
SCL
S or Sr
1
2
7
8
9
ACK
1
2
3 - 8
9
ACK
Sr or P
START or repeated START condition
STOP or repeated START condition
MSC608

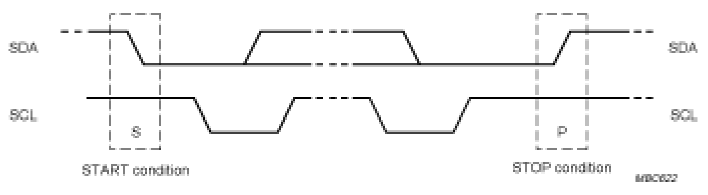
Data transfer on the I²C-bus.

4.2.2.2 Start Condition

A high-to-low transition of SDA with SCL high is a start condition, which must precede any other command.

4.2.2.3 Stop Condition

A low-to-high transition of SDA with SCL high is a stop condition.



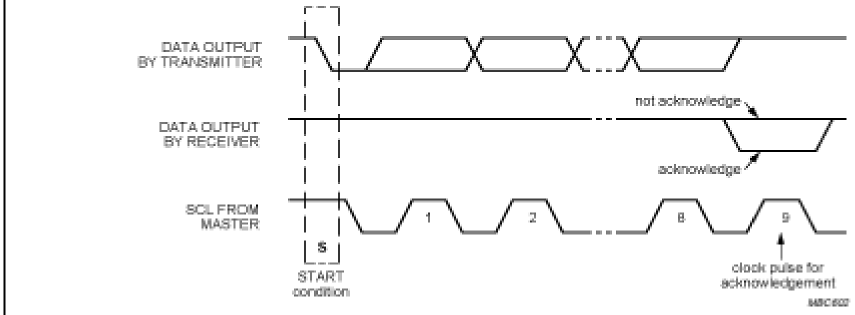
START and STOP conditions.

4.2.2.4 Acknowledge (ACK)

All addresses and data words are serially transmitted to and from the module in 8-bit words. The module sends a zero to acknowledge that it is not busy and has received each word. This happens during the ninth clock cycle.

4.2.2.5 Bus Status

When the module has received command, it doesn't acknowledge I²C bus until ends with the card communication.



Acknowledge on the I²C-bus.

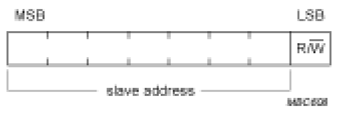
4.2.2.6 Device Address

The module requires a 8-bit device address following a start condition to enable the chip for a read or write operation.

The device address word consists of 7 addressing bits and 1 operation select bit.

The first 7 bits of the module address are 1010000 (0xA0 in hex)

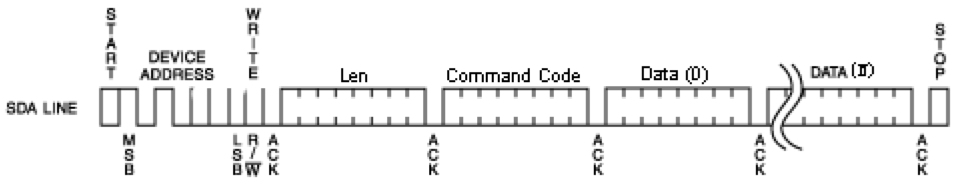
The eighth bit of the device address is the read/write operation select bit. A read operation is initiated if this bit is high and a write operation is initiated if this bit is low.



The first byte after the START procedure.

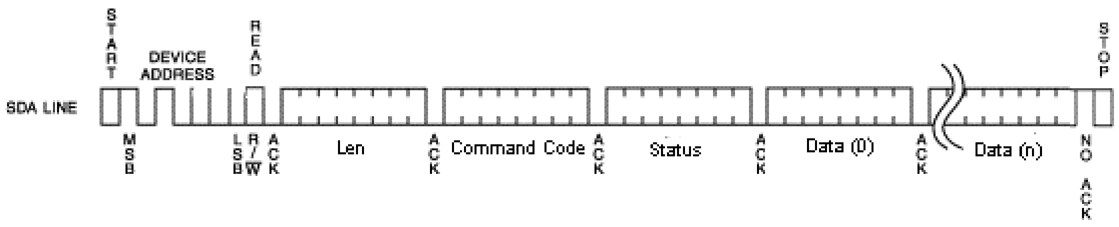
4.2.2.7 Write Data Operation

The host device sends a command to module via writing operation.



4.2.2.8 Read Data Operation

The host device gets result via reading operation.



4.2.3 Data Transaction

The module is a slave device of the I²C bus, so the host needs to write the command package to module. The module will execute the command. Then the host needs to poll the status of the module while it is working by sending out the command of "read" continuously. If the module send an ACK to a read operation, then the last command execution were finished. At this time the host could read the result and/or data from the module.

4.2.4 Description of Command Transaction

E.g.: To read the block 1 of MIFARE card; as "JCP04 Application layer protocol" an example, the steps:

Send command: 0A210001FFFFFFFFFFFF2A

There are steps here:

A. Write command to module
   1. Start condition.
   2. Send control byte, it is 0xA0, the meaning is: address 0xA0 + write control 0x00.
   3. Send module command: 0x0A210001FFFFFFFFFFFF.
   4. Send command checksum: 0x2A.
   5. Stop condition.

B. Send read command. If module no ACK, then the module is working. Repeat this step.
   1. Start condition.
   2. Send control byte 0xA1, it is I²C slave address 0xA0 + read control 0x01.
   3. If module is no ACK, go to step B. if yes, go to step C.

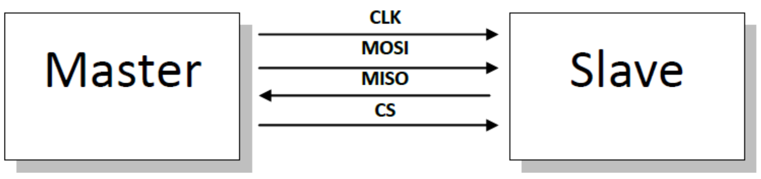
C. Get the data bytes from module
   1. Get the first byte and send ACK, if the data is 0x12, the meaning is there are 18 bytes useful bytes in this package.
   2. Get the else 17 bytes (0x12-1=0x11) data and send ACK after every byte.
   3. Get the checksum and send NACK.
   4. Stop condition.

D. Verify the checksum. If it is ok, the communication is ok.

4.3 SPI interface

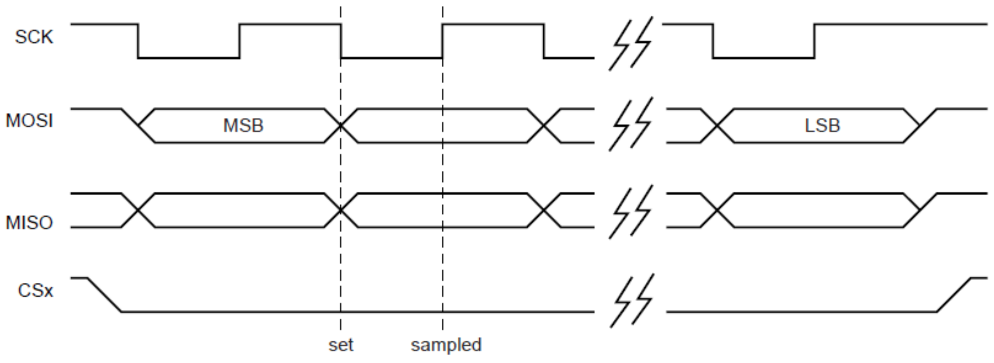
4.3.1 Physical Interface

SPI (Serial Peripheral Interface) is a high speed, full duplex, sychoronos system bus. It works in Master and Slave mode. The connection is show in below.



4.3.1.1 Bus working mode

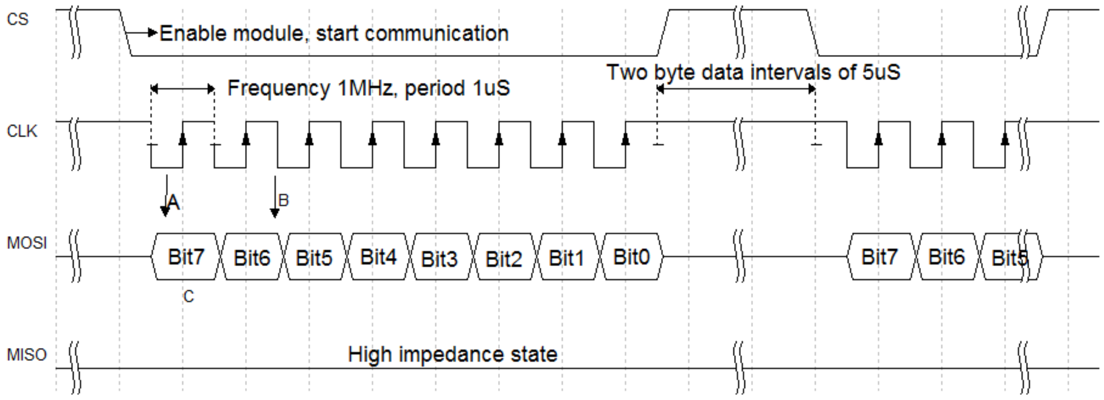
It is standard SPI working mode 3.



SCK idle high. Data sample at 2nd edge of SCK.

4.3.1.2 Write operation

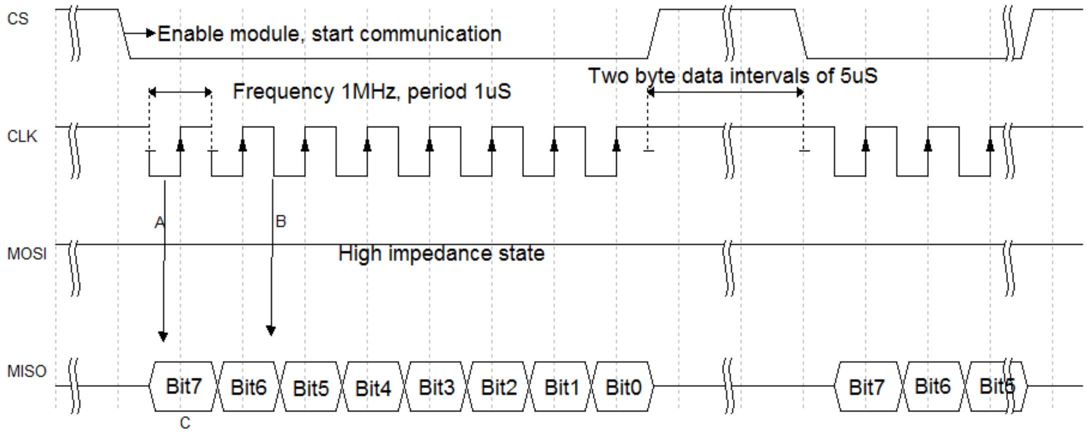
Master writes a byte to Slave.


CS
Enable module, start communication
Two byte data intervals of 5uS
Frequency 1MHz, period 1uS
CLK
A
B
MOSI
Bit7
Bit6
Bit5
Bit4
Bit3
Bit2
Bit1
Bit0
Bit7
Bit6
Bit5
C
MISO
High impedance state

The data is msb first.

4.3.1.3 Read operation

Master read a byte from Slave.


CS
Enable module, start communication
Two byte data intervals of 5uS
Frequency 1MHz, period 1uS
CLK
A
B
MOSI
High impedance state
MISO
Bit7
Bit6
Bit5
Bit4
Bit3
Bit2
Bit1
Bit0
Bit7
Bit6
Bit5
C

4.3.2 Data links

4.3.2.1 SPI instructions

SPI inquire:	0x04	inquire the state of SPI Slave device
SPI write data:	0x10	send data to SPI Slave device
SPI read data:	0x20	get data from SPI Slave device

4.3.2.2 SPI state code

This code is answer to SPI inquire command.

0x42 Slave device is ready to receive new command

0x08 Slave device is ready for read result of previous command or ready to receive new command.

4.3.3 Description of Command Transaction

The procdure of communication

A. Pull CS to low to enable SPI
B. Send inquire command 0x04
C. Read a byte from SPI Salve device
D. If the data is 0x42 or 0x08 then continue. Else to B.
E. Send "SPI write data" command (0x10).
F. Send JCP04 (or JCP05) command package. E.g. request command 03 20 00 23
G. Send inquire command and get the operation result. If the result is 0x08 then continue. Else repeate G. This operation is waiting the Slave device working. The time is determin by the command type. Maybe over 300 seconds.
H. It is failed if no response over waiting time. Pull CS to high and return.
I. If Slave device answer a byte of 0x08, then the Slave device is finish the command and ready to send result to Master device.
J. Send "SPI read data" command to Slave device, a byte of 0x20.
K. Read data from SPI Slave device. This is a package of JCP04 (or JCP05). The length is in the header of the package.
L. Restore SCK to high after read data.
M. Pull CS to high. A communication cycle is finished.

4.4 USB interface

The USB interface accords with USB 2.0 HID specification. The same name pin between host and device could be connected.

In Windows OS, you do not need to add any driver. In Windows OS, the reader sends command and gets response via windows API "SetReport" and "GetReport".

The data length is within 64 bytes because of USB HID in Windows API restriction. In other operation system, the command length is unrestricted.

5 Application-layer Protocol

5.1 Overview

Chapter 5 includes the whole contents of JCP04 and JCP05. Some of the QM-ABCM7 doesn't support the whole parts. For example: some no SAM slots; some don't support some kind of card types (ISO14443B or ISO15693). So users need to make sure the command could operate the card. Otherwise, the wrong information will be returned from the module. The module details could be gotten from the Manual.

This chapter will introduce the communication protocol application level commands and data structures in details; the application level protocol only introduces the commands and the data. The whole data packet composition format could be JPC04 and JCP05, but the command code and data have to be the same.

We illustrate each command in the following format:

Frame	Command	Data	Checksum

Here, "Frame" means JCP04 or JCP05's Frame Header, as follows:

JCP04 Frame Header: 1 byte length information, all the bytes except Checksum byte.

JCP05 Frame Header: Totally 3 bytes, 2 bytes length information (MSB first) and 1 byte UART or RS232C communication address. Length information is all the bytes except Checksum byte. Serial communication address is used for multi communication to select specified equipment to communicate.

Checksum: Exclusive OR (XOR) results from length byte to the last byte of data.

For example, we explain separately with the following command.

Frame	0x11	Mode	Checksum

It's the command to control the module working status, now we need to close the antenna with this command, so the "mode" is 0x00, and the command is:

JCP04: 0x03 11 00 12; in it 0x03 is Frame Header, all the bytes except Checksum byte, the length is 0x03, so take the value 0x03; 0x11 is the command; 0x00 is parameter (meaning close automatic detecting card, close the antenna); 0x12 is Checksum byte, the front 3 bytes XOR result is 0x12.

JCP05: 0x00 05 00 11 00 14; 0x00 05 00 is Frame Header, 0x00 05 is length, 0x00 is UART/RS232C communication address; 0x11 is command; 0x00 is parameter, meaning close automatic detecting card, close antenna; 0x14 is Checksum byte,the front 5 bytes XOR result is 0x14.

Each command has examples following. The example command is JPC05. It is the result of testing JMY6801H. About parts of JCP04, do not recommend using in the new product development.

5.2 System commands

5.2.1 Module reset to factory default

Function: Reset all configurations of the module to factory default setting. The new setting will effect after re-power on. The module will be reset by system watch dog after this command.

Host send:

Frame	0x0F	52 45 53 45 54	Checksum

Success:

Frame	0x0F	Checksum

Failure:

Frame	0xF0	Checksum

We use JMY6801H to do the test. Under JCP04 and JCP05 protocols, the report is as follows:

JCP04 send: 0x07 0F 52 45 53 45 54 5D

JCP04 return: 0x02 0F 0D

In the following commands, we just only use JCP05 protocol as an example.

JCP05 send: 0x00 09 00 0F 52 45 53 45 54 53

JCP05 return: 0x00 04 01 0F 0A

5.2.2 Read product information

Function: read the product information of CURRENT PRODUCT, including product name, firmware version, firmware date and configuration information.

Host send:

Frame	0x10	Checksum

Success:

Frame	0x10	Information	Checksum

Information: 30bytes; 8bytes product name, 4bytes firmware version,8bytes firmware date, 1byte UART baud rate code, 1byte UART Multi-device communication address,1byte I²C address, 1byte multi-card operation enable status,1byte ISO15693 automatic detecting card AFI,1byte ISO15693 automatic detecting card AFI enable status, 1byte automatic detecting card interval,1byte default automatically detecting card status when power on, 1byte default automatically output SNR set when power on, 1byteRF output power

Failure:

Frame	0xEF	Checksum

Example:

Send: 0x00 04 00 10 14

Return: 0x00 22 01 10 4A 4D 59 36 38 30 31 48 34 2E 33 30 32 30 31 33 31 31 32 33 00 01 A0 00 00 00 0A 00 00 00 99

5.2.3 Read PCD information (including UID)

Function: read the PCD information, including Product identification code, UID, and configuration information.

Host send:

Frame	0x03	Checksum

Success:

Frame	0x03	Info.	Checksum

Info.: Products Information, 16 bytes, they are 5 bytes product identification code, 3bytes RFU, 4bytes UID, 3bytes RFU, 1byte CRC)

Failure:

Frame	0xFC	Checksum

Example:

Send: 0x00 04 00 03 07

Return: 0x00 14 01 03 30 FF FF 0F 04 00 00 00 25 54 75 5B A6 57 5E 38 E5

5.2.4 Module Working Mode Set

Function: The temporarily settings for the module, which means that the settings will be lost after power off. Set the antenna RF output ON/OFF; set the automatic detecting card ON/OFF. Automatically detect card and output UID ON/OFF. Under the automatic detecting card and output UID status, after detected the card then output the UID via RS232 or UART, finally make the detected card enter into idle status. If setting the continuously output card UID, after card be detected, it will continuously output the UID until moving the card away from the antenna field. The UID output format could be set as ASCII format.

Host sends:

Frame	0x11	Mode	Checksum

Mode: 1 byte

Antenna status:	BIT0 = 0: OFF;	BIT0 = 1: ON (default ON)
Auto request:	BIT1 = 0: OFF;	BIT1 = 1: ON (default OFF)
Auto request and output UID:	BIT2 = 0: OFF;	BIT2 = 1: ON (default OFF)
Auto request and continuously output UID:	BIT3 = 0: OFF;	BIT2 = 1: ON (default OFF)
Auto request and output UID format:	BIT4 = 0: HEX;	BIT2 = 1: ASCII (default HEX)

Success:

Frame	0x11	Checksum

Failure:

Frame	0xEE	Checksum

Example:

Send: 0x00 05 00 11 03 17

Return: 0x00 04 01 11 14

5.2.5 Set Module Idle

Function: set the module idle. In idle mode, the module of RF output turn to OFF, PCD power down, and CPU in idle mode, so the power consumption reduces to about 100uA. Sending the next command to module will wake up the module, and then the RF output ON and automatic detecting card restore the settings. The module will enter into idle mode after the answer procedure is finished for this command. In I²C and SPI mode, host needs to get the answer, and then the module goes into idle mode. The module with USB interface doesn't support idle mode.

Host sends:

Frame	0x12	Random	Checksum

Random: 1 byte random data, for example: 0x55

Success:

Frame	0x12	Checksum

Failure:

Frame	0xED	Checksum

Example:

Send: 0x00 05 00 12 55 42

Return: 0x00 04 01 12 17

5.2.6 Set LED

Function: set the LED ON or OFF.

Host sends:

Frame	0x13	Status	Checksum

Status: 1byte

LED1	BIT0=0: OFF;	BIT0=1: ON
LED2	BIT1=0: OFF;	BIT1=1: ON
LED3	BIT2=0: OFF;	BIT2=1: ON
LED4	BIT3=0: OFF;	BIT3=1: ON

Success:

Frame	0x13	Checksum

Failure:

Frame	0xEC	Checksum

Example:

Send: 0x00 05 00 13 0F 19

Return: 0x00 04 01 13 16

5.2.7 Set Buzzer

Function: set buzzer to beep.

Host sends:

Frame	0x14	Time	Checksum

Time: 1 byte time, time unit is 10mS. If time is 0x0A, the beep time is 100mS.

Success:

Frame	0x14	Checksum

Failure:

Frame	0xEB	Checksum

Example:

Send: 0x00 05 00 14 AA BB

Return: 0x00 04 01 14 11

5.2.8 Read Data from FLASH in MCU

Function: read data from FLASH in MCU of the module.

Host sends:

Frame	0x15	Address	Bytes	Checksum

Address:	2 bytes, read start address, address from 0x0000 to 0x01FF, MSB first
Bytes:	1 byte, number of bytes to read

Success:

Frame	0x15	Data	Checksum

Data: data to be read.

Failure:

Frame	0xEA	Checksum

Example:

Send: 0x00 07 00 15 00 00 10 02

Return: 0x00 14 01 15 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 00

5.2.9 Write Data into FLASH in MCU

Function: write data into FLASH in MCU of the module.

Host sends:

Frame	0x16	Address	Bytes	Data	Checksum

Address:	2 bytes, read start address, address from 0x0000 to 0x01FF, MSB first.
Bytes:	2 byte, number of bytes to be written.

Data: "Bytes" data to be written.

Success:

Frame	0x16	Checksum

Failure:

Frame	0xE9	Checksum

Example:

Send: 0x00 17 00 16 00 00 10 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 11

Return: 0x00 04 01 16 13

5.2.10 Read Data from FLASH on Module

Function: read data from FLASH on module. This FLASH size is 512KB. The FLASH is 264 bytes per block and 2048 blocks totally. The operations are with in a BLOCK. If you need read data in 2 blocks, then you must send read command twice.

Host sends:

Frame	0x05	BlockNo	Address	Bytes	Checksum

BlockNo: 2 bytes, the reading block number: 0x0000 ~ 0x07FF, MSB first.

Address: 2 bytes, the address in block: 0x0000 ~ 0x0107, MSB first.

Bytes: 2 bytes, the number of bytes to be read: 0x0001 ~ 0x0108, MSB first.

Success:

Frame	0x05	Data	Checksum

Data: the data read result.

Failure:

Frame	0xFA	Checksum

Example:

Send: 0x00 0A 00 05 00 02 00 10 00 08 15

Return: 0x00 0C 01 05 00 01 02 03 04 05 06 07 08

5.2.11 Write Data into FLASH on Module

Function: write data into the FLASH on Module. This FLASH size is 512KB. The FLASH is 264 bytes per block and 2048 blocks totally. The operations are with in a BLOCK. If you need to write data in 2 blocks, you must send write command twice.

Host sends:

Frame	0x06	BlockNo	Address	Bytes	Data	Checksum

BlockNo: 2 bytes, the writing block number: 0x0000 ~ 0x07FF, MSB first.

Address: 2 bytes, the address in block: 0x0000 ~ 0x0107, MSB first.

Bytes: 2 bytes, the number of bytes to be write: 0x0001 ~ 0x0108, MSB first.

Data: the data to be written.

Success:

Frame	0x06	Checksum

Failure:

Frame	0xF9	Checksum

Example:

Send: 0x00 12 00 06 00 02 00 10 00 08 00 01 02 03 04 05 06 07 0E

Return: 0x00 04 01 06 03

5.2.12 Set UART Communication Baud Rate

Function: set UART communication baud rate of the module. After module received the command, it will first save the new setting, and then send the executive result according to the previous baud rate. At last it will set to the new baudrate. UART communication baud rate is default 19200bps. Settings will SAVE in the module, and it won't be lost after power OFF.

Host sends:

Frame	0x17	Baud rate	Checksum

Baud rate: 1 byte, baud rate code; 0: 19200bps; 1: 115200bps; 2: 9600bps; 3: 38400bps; 4: 57600bps.

Success:

Frame	0x17	Checksum

Failure:

Frame	0xE8	Checksum

Example:

Send: 0x00 05 00 17 00 12

Return: 0x00 04 01 17 12

5.2.13 Set UART Multi-device Communication Address

Function: set UART Multi-device communication address of the module. The address is default 1. Settings will SAVE in the module and won't be lost after power OFF.

Host sends:

Frame	0x18	Address	Checksum

Address: 1 byte, UART Muti-device communication address: 1 ~ 0xFF.

Success:

Frame	0x18	Checksum

Failure:

Frame	0xE7	Checksum

Example:

Send: 0x00 05 00 18 02 1F

Return: 0x00 04 02 18 1E

5.2.14 Set I²C Communication Address

Function: set I²C communication address of the module. After module received the command, it will first save the new address, and then send the executive result to the host. At last it will set the new address. The I²C address of the module is 1 byte HEX data. Lsb. is 0; the address of module must be the even number, and the invalid address will NOT be accepted. Settings will save in the module, and it won't be lost after power OFF. The module defult address is 0xA0.

Host sends:

Frame	0x19	Address	Checksum

Address: 1 byte, Lsb. is 0; address must be the even number.

Success:

Frame	0x19	Checksum

Failure:

Frame	0xE6	Checksum

Example:

Send: 0x00 05 00 19 A0 BC

Return: 0x00 04 01 19 1C

5.2.15 Set Multi-card Operation

Function: set multi-card operation. If users need to operate one card from multi-card, they need to use the multi-card operation. If users set the automatic detecting card, the multi-card operation will be prohibited. If there is more than one card in the RF effective field then the operation will fail. Settings will save in the module; it will be not lost after power OFF. Multi-card operation default enables. This function is suitable for ISO14443A.

Host sends:

Frame	0x1A	Enable	Checksum

Enable: 1 byte, 0: Disable multi-card; 1: Enable multi-card; other values: RFU.

Success:

Frame	0x1A	Checksum

Failure:

Frame	0xE5	Checksum

Example:

Send: 0x00 05 00 1A 00 1F

Return: 0x00 04 01 1A 1F

5.2.16 Set ISO15693 Automatic Detecting Card AFI and AFI Enable

Function: set automatic detecting card AFI and AFI enables in ISO15693 mode. If users set AFI and AFI enables, then automatic detecting card only detects the AFI of the card which is equal to the set AFI. Settings will save in the module; it won't be lost after power OFF. AFI is default 0, AFI function is Disable.

Host sends:

Frame	0x1B	AFI	AFI enable	Checksum

AFI:	1 byte, AFI, 0~0xFF.
AFI enable:	1 byte, 0: Disable; 1: Enable; other value: RFU.

Success:

Frame	0x1B	Checksum

Failure:

Frame	0xE4	Checksum

Example:

Send: 0x00 06 00 1B 0A 01 16

Return: 0x00 04 01 1B 1E

5.2.17 Set Automatic Detecting Card Interval Time

Function: set interval time of automatic detecting card function. The default is 100ms. Settings will save in the module; it will be not lost after power OFF.

Host sends:

Frame	0x1C	Time	Checksum

Time: 1 byte, 0x00 to 0xFF, unit is 10mS, 0x01 means 10mS.

Success:

Frame	0x1C	Checksum

Failure:

Frame	0xE3	Checksum

Example:

Send: 0x00 05 00 1C 01 18

Return: 0x00 04 01 1C 19

5.2.18 Set the Default of Automatic Detecting Card

Function: Set the default state of automatic detecting card when turned on device. Settings will save in the module; it will be not lost after power OFF. For temporarily open or close automatically detect card, please use the 0x11 command.

Host sends:

Frame	0x1D	Status	Checksum

Status: 1 byte, 0x00: OFF; 0x01: ON, other value: RFU

Success:

Frame	0x1D	Checksum

Failure:

Frame	0xE2	Checksum

Example:

Send: 0x00 05 00 1D 01 19

Return: 0x00 04 01 1D 18

5.2.19 Set Default of Automatic Detecting Card and Output SNR

Function: Set default state of automatically detecting card and output SNR when turned on device. Settings will save in the module; it will be not lost after power OFF.

Host sends:

Frame	0x1E	Status	Checksum

Status: 1 byte.

Auto request and output UID:	BIT0 = 0: OFF;	BIT0 = 1: ON
Continuously output UID:	BIT1 = 0: OFF;	BIT1 = 1: ON
ASCII output format:	BIT2 = 0: OFF;	BIT2 = 1: ON

Output Communication Protocol Choose:

	BIT4:BIT3 = 0:0 JCP04
	BIT4:BIT3 = 0:1 JCP05
	BIT4:BIT3 = 1:0 JCP02
	BIT4:BIT3 = 1:1 JCP03
Process ISO15693:	BIT5 = 0: ON; BIT5 = 1: OFF
Process ISO14443 TYPE A:	BIT6 = 0: ON; BIT6 = 1: OFF
RFU:	BIT7 = 1(MUST BE 1);

Success:

Frame	0x1E	Checksum

Failure:

Frame	0xE1	Checksum

Example:

Send: 0x00 05 00 1E 00 1B

Return: 0x00 04 01 1E 1B

5.2.20 Set the RF Output Level

Function: To set the RF output level. When the RF output power is reduced, the card operation distance will be reduced too. The customer could set it according to the concrete needs. Settings will save in the module; it will be not lost after power OFF. This command does not support in CL RC663 devices.

Host sends:

Frame	0x02	Power	Checksum

Power: 1 byte, 0x00: the strongest; 0x01: the stronger; 0x02: the weak; 0x03: the weakest; other values: RFU.

Success:

Frame	0x02	Checksum

Failure:

Frame	0xFD	Checksum

Example:

Send: 0x00 05 00 02 00 07

Return: 0x00 04 01 02 07

5.2.21 Module Contactless Protocol Set

Function: The default for setting module contactless protocol is ISO14443A. If the module doesn't support ISO14443A, the default protocol is ISO15693. The setting won't be saved and will return to the default status at next power on. The firmware version 6.00 or later was build in automatic protocol set function.

Host sends:

Frame	0x70	Mode	Checksum

Mode: 1 byte, 0: ISO14443A; 1: ISO14443B; 2: ISO15693; 3: I.CODE 1; 4: ISO18000-3M3; other value: RFU

Success:

Frame	0x70	Checksum

Failure:

Frame	0x8F	Checksum

Example:

Send: 0x00 05 00 70 00 75

Return: 0x00 04 01 70 75

5.2.22 Set Current Antenna

Function: Set the Current Antenna for Multi Antenna readers.

Host sends:

Frame	0x04	Number	Checksum

Number: 1 byte, 0~255, the number of the antenna.

Success:

Frame	0x04	Checksum

Failure:

Frame	0xFB	Checksum

Example:

Send: 0x00 05 00 04 01 00

Return: 0x00 04 01 04 01

5.3 Module security

5.3.1 Get authentication code

Function: Get authentication code from module; this is the 1st step of module authentication.

Host sends:

Frame	0x07	Code	Checksum

Code: 8 bytes, ciphered authentication code.

Success:

Frame	0x07	Checksum

Example:

Send: 0x00 04 00 07 03

Return: 0x00 0C 01 07 D0 9B 09 33 2C B7 9F FB 84

5.3.2 Module authentication

Function: This is the 2nd step of module authentication; after this step the controller could be sure the module is the certified products.

NDA is needed for the authentication method. Contact us please.

Host sends:

Frame	0x08	AuthData	Checksum

AuthData: 16 bytes, ciphered authentication data.

Success:

Frame	0x08	Result	Checksum

Result: 8 bytes authentication result.

Failure:

Frame	0xF7	Checksum

Example:

Send: 0x00 14 00 08 83 68 42 DB 08 9B FD CD 92 25 31 1A EE 04 31 FC 76

Return: 0x00 0C 01 08 1F 7A 61 4D 9D 65 35 1D 9C

5.3.3 Modify authentication key

Function: Modify the authentication key of the module; authentication is needed for this operation.

Host sends:

Frame	0x09	KeyData	Checksum

KeyData: 32 bytes, ciphered key modificatiaon data.

Success:

Frame	0x09	Checksum

Failure:

Frame	0xF6	Checksum

Example:

Send: 0x00 24 00 09 1F 7A 61 4D 9D 65 35 1D B4 D0 A7 45 41 14 72 61 82 2B BF ED C0 AD 98 86 B0 0B 5F 5A 80 0C 18 50 86

Return: 0x00 04 01 09 0C

5.3.4 Encrypted Communication

Function: send commands to reader with encryption.

Host sends:

Frame	0xFC	Ciphertext	Checksum

Ciphertext: command to reader with encryption

Success:

Frame	0xFC	Ciphertext	Checksum

Ciphertext: result from reader with encryption

Failure:

Frame	0x03	Checksum

Example:

Send: 0x00 0C 00 FC B0 A5 56 01 11 46 D7 F1 C3

Return: 0x00 14 01 FC 69 B6 C4 D2 1C 70 CA 6A 0F 1D 64 2B D6 D1 68 36 E8

5.4 ISO14443A/B CPU Card Commmands

5.4.1 ISO14443 TYPE A Request

Function: ISO14443A request cards, cards include MIFARE and other ISO14443A cards. In the returned results, user could judge the length of serial number via the returned data package length, and judge the card type by ATQA, also users can judge whether the card supports ISO14443-4 by SAK. If automatic detecting card function is opened, then this command is only to read the result of automatic detecting card.

Host sends:

Frame	0x20	Mode	Checksum

Mode: 1 byte, 0: WUPA; 1: REQA; other value: RFU

Success:

Frame	0x20	Data	Checksum

Data: 4, 7 or 10 bytes card serial number + 2 bytes ATQA + 1 byte SAK

Failure:

Frame	0xDF	Checksum

Example:

Send: 0x00 05 00 20 00 25

Return: 0x00 0B 01 20 32 41 00 21 04 00 28 54

5.4.2 Reuqest all ISO14443 TYPE A Card

Function: ISO14443A request cards and output all the UIDs.

Host sends:

Frame	0x7D	Checksum

Mode: 1 byte, 0: WUPA; 1: REQA; other value: RFU

Success:

Frame	0x7D	N * CI	Checksum

N: number of cards detected

CI: card information. 10byte UID + 2 bytes ATQA + 1 byte SAK + 1byte actual UID length

Failure:

Frame	0x82	Checksum

Example:

Send: 0x00 04 00 7D 79

Return: 0x00 20 01 7D 76 3F 82 A2 00 00 00 00 00 00 00 00 08 04 52 90 32 22 00 00 00 00 00 00 00 00 28 04 C7

5.4.3 ISO14443-4 TYPE A Card RATS

Function: send RATS to ISO14443-4 TYPE-A card. Before executing this command, it needs to request card and verify the card support ISO14443-4 via SAK of card. If the automatic detecting card function is on, after a successful implementation of the RATS command, the automatic detecting card function will be forced OFF.

Host sends:

Frame	0x30	EN_NADCID	Checksum

EN_ NADCID: 1: enable NADCID; 0: disable NADCID (EMV is disable)

Success:

Frame	0x30	ATS	Checksum

ATS: ATS, length depends on card.

Failure:

Frame	0xCF	Checksum

Example:

Send: 0x00 05 00 30 00 35

Return: 0x00 14 01 30 10 78 80 90 02 20 90 00 00 00 00 00 21 00 41 32 BD

5.4.4 ISO14443-4 TYPE B Request

Function: ISO14443-4 TYPE B card request and set attribute.

Host sends:

Frame	0x60	Mode	AFI	EN_ NADCID	Checksum

Mode: 1 byte, 0: WUPB; 1: REQB; other values: RFU

AFI: 1 byte, the AFI to request, if request all AFI, please use 0x00.

EN_ NADCID: 1: enable NADCID; 0: disable NADCID (EMV is disable)

Success:

Frame	0x60	Info.	Checksum

Info: total 13 bytes, 12 bytes of ATQB: 0x50 (1 byte), PUPI (4 bytes), application data (4 bytes), protocol information (3 bytes), 1 byte answer to Attribute.

For more details, please refer to ISO14443-3 "ATQB Response" part.

Failure:

Frame	0x9F	Checksum

Example:

Send: 0x00 07 00 60 00 00 00 67

Return: 0x00 12 01 60 50 18 12 02 23 00 00 00 00 00 00 81 00 00 89

5.4.5 Request Card according to EMV and PBOC

Function: Card Request according to EMV and PBOC standards, and then to set the communication parameters between the module and card. This card request command is aim to CPU card (T=CL). It contains ISO14443A&B. After requesting the card via this command, you could operate the CPU card via sending APDU commands.

Host sends:

Frame	0x32	Checksum

Success:

Frame	0x32	Type	Info	Checksum

Type:	0x41:	ISO14443 TYPE A;
	0x42:	ISO14443 TYPE B;
	0x4D:	Multi card in the Antenna field, request failed.

Info: TYPE A card returned data:

0x41, 1 byte UID Length; Length bytes UID; 2 bytes ATQA; 1 byte SAK; ATS (ATS is not fixed. Please reference the Datasheet of the card from the suppliers.).

TYPE B card returned data:

0x42, 1 byte 0x50, 4 bytes PUPI, 4 bytes Application data, 3 bytes Protocol information, 1 byte answer to ATTRIB.

Failure:

Frame	0xCD	Checksum

Example:

Send: 0x00 04 00 32 36

Return: 0x00 12 01 32 42 50 18 12 02 23 00 00 00 00 00 81 81 00 18

5.4.6 Send APDU to ISO14443-4 Card

Function: Send APDU to an ISO14443-4 card. Before executing the command, it needs to reset the card. If operating ISO14443-4 card, the automatic detect fuction will need to be turned OFF. That’s because the ISO14443-4 card’s status will be lost in automatic detecting card.

Host sends:

Frame	0x31	APDU	Checksum

APDU: APDU to send

Success:

Frame	0x31	Response	Checksum

Response: card response, length depends on the detailed command

Failure:

Frame	0xCE	Checksum

Example:

Send: 0x00 09 00 31 00 84 00 00 08 B4

Return: 0x00 0E 01 31 B9 89 3A B0 16 40 7E D0 90 00 EC

5.4.7 ISO14443-4 TYPE B Card Halt

Function: To let the current ISO14443B card enter into halt status. Not all of the cards support this command, most of them don't support, especially the new card.

Host sends:

Frame	0x62	PUPI	Checksum

PUPI: 4 bytes, PUPI of the card that will be halt.

Success:

Frame	0x62	Checksum

Failure:

Frame	0x9D	Checksum

Example:

Send: 0x00 08 01 62 00 00 00 00 6B

Return: 0x00 04 01 62 67

5.5 MIFARE 1K/4K/mini Card Commands

5.5.1 MIFARE Request

MIFARE series cards request, please refer to ISO14443 TYPE A Request.

5.5.2 MIFARE 1K/4K Data Block Read

Function: Read MIFARE 1K/4K one block data.

Host sends:

Frame	0x21	Key ID	Block	Key	Checksum

Key ID: 1 byte, Key identifier

BIT0 = 0:Key A; BIT0 = 1: Key B;

BIT1=0: using the key in the command; BIT1=1: using the key downloaded by command 0x2D;

BIT6:BIT5:BIT4:BIT3:BIT2: if use the downloaded key, this is the index of the key;

BIT7=0: The block need to be certified via using the above key;

BIT7=1: The block has been authenticated and passed. Do not need authentication again. (This operation and automatic detecting card could not be used at the same time);

(IMPORTANT: more information please refers to Chapter 5.3 about Key identifier).

Block: 1 byte, Block number to read, 0 to 0x3F for S50; 0 to 0xFF for S70;

Key: 6 bytes, the key of the card.

Success:

Frame	0x21	Data	Checksum

Data: 16 bytes card data

Failure:

Frame	0xDE	Checksum

Example:

Send: 0x00 0C 00 21 00 01 FF FF FF FF FF FF 2C

Return: 0x00 14 01 21 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 34

5.5.3 MIFARE 1K/4K Multi-Blocks Read

Function: Read multi data blocks in the same sector. This function is supported only in the same sector. If crossing sectors, the reading will fail.

Host sends:

Frame	0x2A	Key ID	Start Block	Blocks	Key	Checksum

Key ID: 1 byte, key identifier;

Start Block: 1 byte, the start block to be read;

Blocks: 1 byte, number of blocks to be read. All blocks need in same sector.

Key: 6 bytes, the key of the card.

Success:

Frame	0x2A	Data	Checksum

Data: blocks * 16 bytes card data per block

Failure:

Frame	0xD5	Checksum

Example:

Send: 0x00 0D 00 2A 00 01 02 FF FF FF FF FF FF 24

Return: 0x00 24 01 2A 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 0F

5.5.4 MIFARE 1K/4K Data Block Write

Function: Write the data to a block of MIFARE 1K/4K.

Host sends:

Frame	0x22	Key ID	Block	Key	Data	Checksum

Key ID:	1 byte, Key identifier;
Block:	1 byte, Block number to be written;
Key:	6 bytes, the key of the card;
Data:	16 bytes data to be written.

Success:

Frame	0x22	Checksum

Failure:

Frame	0xDD	Checksum

Example:

Send: 0x00 1C 00 22 00 01 FF FF FF FF FF FF 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 3F

Return: 0x00 04 01 22 27

5.5.5 MIFARE 1K/4K Multi-Blocks Write

Function: Write multi data blocks. The function is supported only in the same sector. If crossing sector, it will fail while writing the first block in the next sector and then prompt the error in the returned result.

Host sends:

Frame	0x2B	Key ID	Start Block	Blocks	Key	Data	Checksum

Key ID:	1 byte, key identifier;
Start Block:	1 byte, the start block number to be written;
Blocks:	1 byte, number of blocks to be written;
Key:	6 bytes, the key of the card;
Data:	blocks * 16 bytes data to write per block

Success:

Frame	0x2B	Checksum

Failure:

Frame	0xD4	Checksum

Example:

Send: 0x00 2D 00 2B 00 01 02 FF FF FF FF FF FF 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 05

Return: 0x00 04 01 2B 2E

5.5.6 MIFARE 1K/4K Purse Block Initialization

Function: Initialize a block of MIFARE 1K/4K as a purse. The format of purse uses MIFARE 1K/4K's default. The card's key block and block 0 could not be used as a purse. For more details about MIFARE 1K/4K card, please refer to the datasheet.

Host sends:

Frame	0x23	Key ID	Block	Key	Value	Checksum

Key ID:	1 byte, Key identifier;
Block:	1 byte, Block number to be initialized;
Key:	6 bytes, the key of the card;
Value:	4 bytes, initialized value, LSB first.

Success:

Frame	0x23	Checksum

Failure:

Frame	0xDC	Checksum

Example:

Send: 0x00 10 00 23 00 01 FF FF FF FF FF FF 04 03 02 01 36

Return: 0x00 04 01 23 26

5.5.7 MIFARE 1K/4K Purse Read

Function: Read a purse of MIFARE 1K/4K. The format of purse uses MIFARE 1K/4K's default. Module will read the data in the block and check if it is a purse format. If the purse format is incorrect, the respose will show failure.

Host sends:

Frame	0x24	Key ID	Block	Key	Checksum

Key ID:	1 byte, Key identifier;
Block:	1 byte, block number of the value to be read;
Key:	6 bytes, the key of the card.

Success:

Frame	0x24	Data	Checksum

Data: 4 bytes value data, LSB first.

Failure:

Frame	0xDB	Checksum

Example:

Send: 0x00 0C 00 24 00 01 FF FF FF FF FF FF 29

Return: 0x00 08 01 24 04 03 02 01 29

5.5.8 MIFARE 1K/4K Purse Increment

Function: Purse increment of MIFARE 1K/4K. The format of the purse uses MIFARE 1K/4K's default. Purse increment means the increment on the basis of the original value.

Host sends:

Frame	0x25	Key ID	Block	Key	Value	Checksum

Key ID:	1 byte, Key identifier;
Block:	1 byte, block number of purse to be increment;
Key:	6 bytes, the key of the card;
Value:	4 bytes, increment value, LSB first.

Success:

Frame	0x25	Checksum

Failure:

Frame	0xDA	Checksum

Example:

Send: 0x00 10 00 25 00 01 FF FF FF FF FF FF 01 00 00 00 35

Return: 0x00 04 01 25 20

5.5.9 MIFARE 1K/4K Purse Decrement

Function: Purse decrement of MIFARE 1K/4K. The format of the purse uses MIFARE 1K/4K's default. Purse decrement means the decrement on the basis of the original number. Purse decrement only needs the "read authority" of the key.

Host sends:

Frame	0x26	Key ID	Block	Key	Value	Checksum

Key ID:	1 byte, Key identifier;
Block:	1 byte, block number of purse to be decrement;
Key:	6 bytes, the key of the card;
Value:	4 bytes, decrement value, LSB first

Success:

Frame	0x26	Checksum

Failure:

Frame	0xD9	Checksum

Example:

Send: 0x00 10 00 26 00 01 FF FF FF FF FF FF 02 00 00 00 35

Return: 0x00 04 01 26 23

5.5.10 MIFARE 1K/4K Purse Backup

Function: Copy the MIFARE 1K/4K purse to another block in the same sector. The format of the purse uses MIFARE 1K/4K's default.

Host sends:

Frame	0x27	Key ID	Source	Target	Key	Checksum

Key ID:	1 byte, Key identifier;
Source:	1 byte, block number of purse to copy;
Target:	1 byte, copy the purse to this block (source and target need in same sector);
Key:	6 bytes, the key of the card.

Success:

Frame	0x27	Checksum

Failure:

Frame	0xD8	Checksum

Example:

Send: 0x00 0D 00 27 00 01 02 FF FF FF FF FF FF 29

Return: 0x00 04 01 27 22

5.5.11 ISO14443A Card Halt

Function: Set the current operating ISO14443A card (including MIFARE series cards) into halt status.

Host sends:

Frame	0x28	Checksum

Success:

Frame	0x28	Checksum

Failure:

Frame	0xD7	Checksum

Example:

Send: 0x00 04 00 28 2C

Return: 0x00 04 01 28 2D

5.5.12 Download MIFARE 1K/4K Card Key into Module

Function: Download the MIFARE 1K/4K card key into module. There are 32 key memory spaces in the module that could storage 32 different keys. While using the downloaded key in the module, this key wouldn't appear on the pin-outs of the PCD. So it could be safer. Because the written time of EEPROM is limited, please do not use this command frequently. Lose efficacy EEPROM could not be work.

Host sends:

Frame	0x2D	Key Index	Key	Checksum

Key Index: 1 byte, Key Index (0 ~ 0x1F) in the module.

Key:	6 bytes, the key of the card to be stored in module.

Success:

Frame	0x2D	Checksum

Failure:

Frame	0xD2	Checksum

Example:

Send: 0x00 0B 00 2D 00 FF FF FF FF FF FF 26

Return: 0x00 04 01 2D 28

5.5.13 About KEY Identifier

There is a byte of KEY identifier in command of MIFARE 1K/4K cards. This byte will identify the way to get the card key.

KeyIdentifier							
BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0

BIT0 = 0: KEY A; authenticate Key A of the card.
BIT0 = 1: KEY B; authenticate Key B of the card.
BIT1 = 0: Using the following 6 bytes Key in command.
BIT1 = 1: Using the downloaded Key by command.
BIT6:BIT5:BIT4:BIT3:BIT2: Index of the Key already downloaded (0 to 31).
BIT7=0: The block need to authenticate with the above key.
BIT7=1: The block has been authenticated. This operation do not need to authenticate again (this operation and automatic detecting card could not be used at the same time).

If BIT1 is 0, then these 5 bits (BIT6 to BIT2) are unused. If BIT1 is 1, then use the already downloaded key. Users need to download key(s) first; and then the 6 bytes key in the command are left unused, but the 6-byte is necessary in the command sequence.

E.g.: key Identifier is 0x00; binary system is 00000000, here:
BIT0 = 0; authenticate Key A of the card
BIT1 = 0; using the key in command
BIT6:BIT5:BIT4:BIT3:BIT2: 00000, because not use the already downloaded key, the index key is useless in this command.

E.g.: key Identifier is 0x33; binary system is 00110011, here:
BIT0 = 1; authenticate Key B of the card
BIT1 = 1; using the downloaded Key in the module
BIT6:BIT5:BIT4:BIT3:BIT2:01100, then use the already downloaded key 01100, and hexadecimal is 0x0C, decimal is 12.

5.6 MIFARE Ultralight/Ultralight C/Ultralight EV1 Card Commands

5.6.1 MIFARE Ultralight/Ultralight C/Ultralight EV1 Request

For MIFARE UltraLight/UltraLight C card request, please refer to ISO14443 TYPE A Request.

5.6.2 MIFARE Ultralight/Ultralight C/Ultralight EV1 Card Read

Function: Read the data from MIFARE UltraLight/UltraLight C cards. A read command will read 4 blocks data from the card. If read start block is the last block (0x0F), then these 4 blocks data are the 15th, 0th, 1st and 2nd block.

Host sends:

Frame	0x41	Start Block	Checksum

Or Host sends:

Frame	0x41	Start Block	NOB	Checksum

Start Block: 1 byte, the start block number to be read.

NOB: number of blocks, if no this part is default 4 blocks. Only support 4, 8, 12, 16 blocks of N times of 4.

Success:

Frame	0x41	Data	Checksum

Data: 16 bytes card data of 4 blocks, a read operation read 4 blocks from the start block.

Failure:

Frame	0xBE	Checksum

Example:

Send: 0x00 05 00 41 05 41

Return: 0x00 14 01 41 00 06 01 10 11 FF 00 00 00 00 00 00 88 88 88 88 AD

Send: 0x00 06 00 41 05 08 4A (read 8 blocks, total 32 byes)

Return: 0x00 24 01 41 00 06 01 10 11 FF 00 00 00 00 00 00 88 88 88 88 00 06 01 10 11 FF 00 00 00 00 00 00 88 88 88 88 AD

5.6.3 MIFARE Ultralight/Ultralight C/Ultralight EV1 Card Write

Function: Write data to MIFARE UltraLight/UltraLight C cards. Each for one block data.

Host sends:

Frame	0x42	Block	Data	Checksum

Block: 1 byte, block number to be written.

Data: 4 bytes data to be written.

Success:

Frame	0x42	Checksum

Failure:

Frame	0xBD	Checksum

Example:

Send: 0x00 09 00 42 05 55 55 55 55 4E

Return: 0x00 04 01 42 47

5.6.4 MIFARE UltraLight C Key Authentication

Function: Inputting UltraLight C key, the device directly authenticate the key. This process of authentification is controled by the module.

Host sends:

Frame	0x43	Key	Checksum

Key: 16 bytes UltraLight C key

Success:

Frame	0x43	Checksum

Failure:

Frame	0xBC	Checksum

Example:

Send: 0x00 14 00 43 49 45 4D 4B 41 45 52 42 21 4E 41 43 55 4F 59 46 21

Return: 0x00 04 01 43 46

5.6.5 MIFARE UltraLight C Ek (RndB) Read

Function: To read encrypted RndB that is generated by Ultralight C card. Command 0x44 and 0x45 are the separation commands to authenticate Ultralight C. Because the micro-controller calculates 3DES is slower, so authentication will be more time-consuming, and therefore part of the separation of authentication allows users to calculate 3DES to save authentication time.Users could first consider using 0x43 to authenticate, if required authentication in speed, you could contact us using separate authentication commands for technical support.

Host sends:

Frame	0x44	Checksum

Success:

Frame	0x44	Ek (RndB)	Checksum

Ek (RndB): the card returned RndB encrypted data. The RndB was done DES decryption via using the card key. After be decrypted, byte shifted---the first byte is moved to the end, then got RndB'. At this time Ek (RndB) is the subsequent 3DES CBC algorithm initial vector.

Failure:

Frame	0xBB	Checksum

Example:

Send: 0x00 04 00 44 40

Return: 0x00 0C 01 44 93 AB 8B F9 42 56 AF 14 AC

5.6.6 MIFARE Ultralight C Ek (RndA + RndB') Authentication

Function: Input the "RndA + RndB'" which have already been encrypted.

Host sends:

Frame	0x45	Ek (RndA＋RndB')	Checksum

Ek (RndA＋RndB'): 16 bytes' result which "RndA + RndB'" be encrypted via using DES CBC. RndA is 8bytes random number specified by the user. RndB is obtained by the 0x44 command. RndB obtained by decrypting the shift (the first byte to be shifted the last).

Success:

Frame	0x45	Ek(RndA)	Checksum

Failure:

Frame	0xBA	Checksum

Ek(RndA): The card returned encrypted RndA. After decrypted and shifted via using 3DES CBC, then to compare result with RndA. If equality, authentication is passed.

Example:

Send: 0x00 14 00 45 6E 30 F6 C1 17 05 C0 BE 48 40 DE 68 71 6C 9F F0 98

Return: 0x00 0C 01 45 30 F6 C1 17 05 C0 BE 6E 4D

5.6.7 Ultralight EV1 GET_VERSION

Function: The GET_VERSION command is used to retrieve information on the MIFARE family, product version, storage size and other product data required to identify the Ultralight EV1 card.

Host sends:

Frame	0x46	Checksum

Success:

Frame	0x46	Version	Checksum

Version: 8bytes Ultralight EV1 Card version.

Failure:

Frame	0xB9	Checksum

Example:

Send: 0x00 04 00 46 42

Return: 0x00 0C 01 46 00 04 03 01 01 00 0B 03 44

5.6.8 Ultralight EV1 FAST_READ

Function: The FAST_READ command requires a start page address and an end page address and returns the all n*4 bytes of the addressed pages.

Host sends:

Frame	0x47	Start Block	End Block	Checksum

Start Block: 1byte.

End Block: 1byte.

Success:

Frame	0x47	Card data	Checksum

Card data: Blocks * 4 bytes card data.

Failure:

Frame	0xB8	Checksum

Example:

Send: 0x00 06 00 47 00 05 44

Return: 0x00 1C 01 47 04 52 7F A1 42 F9 38 80 03 48 00 00 00 00 00 00 00 00 00 00 00 00 00 00 9A

5.6.9 Ultralight EV1 READ_CNT

Function: The READ_CNT command is used to read the current value of one of the 3 one-way counters of the Ultralight EV1.

Host sends:

Frame	0x48	Address	Checksum

Address: 1byte, Ultralight EV1 counters address.

Success:

Frame	0x48	Data	Checksum

Data: 3 bytes Ultralight EV1 counter data.

Failure:

Frame	0xB7	Checksum

Example:

Send: 0x00 05 00 48 00 4D

Return: 0x00 07 01 48 06 00 00 48

5.6.10 Ultralight EV1 INCR_CNT

Function: The INCR_CNT command is used to increment one of the 3 one-way counters of the Ultralight EV1. The two arguments are the counter number and the increment value.

Host sends:

Frame	0x49	Address	Data	Checksum

Address: 1byte, Ultralight EV1 counters address.

Data: 3 bytes, Ultralight EV1 INCR_CNT data.

Success:

Frame	0x49	Checksum

Failure:

Frame	0xB6	Checksum

Example:

Send: 0x00 08 00 49 00 01 00 00 40

Return: 0x00 04 01 49 4C

5.6.11 Ultralight EV1 PWD_AUTH

Function: A protected memory area can be accessed only after a successful password authentication using the PWD_AUTH command.

Host sends:

Frame	0x4A	PWD	Checksum

PWD: 4byte, Ultralight EV1 card password.

Success:

Frame	0x4A	Checksum

Failure:

Frame	0xB5	Checksum

Example:

Send: 0x00 08 00 4A FF FF FF FF 42

Return: 0x00 06 01 4A 00 00 4D

5.6.12 Ultralight EV1 READ_SIG

Function: The READ_SIG command returns an IC-specific, 32-byte ECC signature, to verify NXP Semiconductors as the silicon vendor. The signature is programmed at chip production and cannot be changed afterwards.

Host sends:

Frame	0x4B	Checksum

Success:

Frame	0x4B	Signature	Checksum

Signature: 32 bytes signature data.

Failure:

Frame	0xB4	Checksum

Example:

Send: 0x00 04 00 4B 4F

Return: 0x00 24 01 4B 6F 44 89 DA 45 59 EF C9 87 1A DB D0 CF 66 4D 47 F7 0A C9 EC 2E EC 7C FD 81 FF 74 4B 2E 28 97 1F 4D

5.6.13 Ultralight EV1 CHECK_TEARING_EVENT

Function: The CHECK_TEARING_EVENT command enables the application to identify if a tearing event happened on a specified counter element. It takes the counter number as single argument and returns a specified valid flag for this counter. If the returned valid flag is not equal to the predefined value, a tearing event happened. Note, although a tearing event might have happened on the counter, a valid value corresponding to the last valid counter status is still available using the READ_CNT command.

Host sends:

Frame	0x8C	Address	Checksum

Address: 1byte, Ultralight EV1 counters address.

Success:

Frame	0x8C	Flag	Checksum

Flag: 1byte, valid flag for this counter.

Failure:

Frame	0x73	Checksum

Example:

Send: 0x00 05 00 8C 00 89

Return: 0x00 05 01 8C BD 35

5.6.14 Ultralight EV1 VCSL

Function: The VCSL command is used to enable a unique identification and selection process across different MIFARE cards and card implementations on mobile devices. The command requires a 16-byte installation identifier IID and a 4-byte PCD capability value as parameters. The parameters are present to support compatibility to other MIFARE devices but are not used or checked inside the MF0ULx1. Nevertheless, the number of bytes is checked for correctness. The answer to the VCSL command is the virtual card type identifier VCTID. This identifier indicates the type of card or ticket. Using this information, the reader can decide whether the ticket belongs to the installation or not.

Host sends:

Frame	0x8D	IID	PCDCAPS	Checksum

IID:	16bytes, installation identifier.
PCDCAPS:	4bytes, PCD capabilities.

Success:

Frame	0x8C	VCTID	Checksum

VCTID: 1byte, virtual Card Type Identifier.

Failure:

Frame	0x72	Checksum

Example:

Send: 0x00 18 00 8D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 95

Return: 0x00 05 01 8D 05 8C

5.7 MIFARE Plus Card Commands

QM-ABCM7 series reader module support MIFARE Plus card operation. NXP MIFARE Plus cards are used to instead MIFARE 1 card. We provide application commands which are based on Level 3. In the card level 3, the authentication use AES encryption algorithm. In the communication process between module and card, all are using encrypted data + command with MAC + response with MAC mode. So the security of RF communication is extremely high.

Use the following command allows the user to quickly start MIFARE Plus R & D works. But for advanced user, also could use APDU to implement the card.

5.7.1 MIFARE Plus Prepare Commands

5.7.1.1 MIFARE Plus Request

For MIFARE Plus card request, please refer to ISO14443 TYPE A Request.

5.7.1.2 MIFARE Plus RATS

Please refer to ISO14443-4 TYPE-A card reset (RATS).

5.7.1.3 MIFARE Plus Request and RATS

Please refer to Card Request according to EMV and PBOC.

5.7.2 MIFARE Plus Initialization Commands

5.7.2.1 MIFARE Plus Write Perso

Function: Initialization of the AES key and all other blocks. About these blocks address and the default value, please refer to the MIFARE Plus datasheet or contact us.

Host sends:

Frame	0x33	Address	Data	Checksum

Address: 2 bytes block address, MSB first.

Data: 16 bytes.

Success:

Frame	0x33	Status	Checksum

Status: status code returned from the card, the communication between the card and module is successful, but it may not meet the conditions for the implementation.

Please reference: MIFARE Plus Returned Status Code.

Failure:

Frame	0xCC	Checksum

Example:

Send: 0x00 16 00 33 40 00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 65

Return: 0x00 04 01 33 36

5.7.2.2 MIFARE Plus Commit Perso

Function: Level 0 command, to switch Level0 to Level1 or Level3. Target Level depends on the card. If need switch to Level 1 or Level 3, please tell the suppliers when purchasing. Before using this command, please use MIFARE Plus Write Perso command to write all AES key and the initial value of all the blocks, then make the changed data effective.

Host sends:

Frame	0x34	Checksum

Success:

Frame	0x34	Status	Checksum

Status: status code returned from the card.

Please reference: MIFARE Plus Returned Status Code.

Failure:

Frame	0xCB	Checksum

Example:

Send: 0x00 04 00 34 30

Return: 0x00 04 01 34 31

5.7.2.3 MIFARE Plus Switch to Level2/3

Function: Level 1 or Level 2 command, switch to Level2 or Level3

Host sends:

Frame	0x35	Level	Key	Checksum

Level: 1 byte, level to be switched, 2: Level 2; 3: Level 3.

Key: 16 bytes.

Success:

Frame	0x35	Status	Checksum

Status: status code returned from the card.

Please reference: MIFARE Plus Returned Status Code.

Failure:

Frame	0xCA	Checksum

Example:

Send: 0x00 15 00 35 03 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 23

Return: 0x00 04 01 35 30

5.7.3 MIFARE Plus Application Layer Commands

5.7.3.1 MIFARE Plus Data Block Authenticate

Function: Level 3 command, authentication for data block.

Host sends:

Frame	0x36	Key Type	Address	Key	Checksum

Key Type: 0: key A; 1: key B

Address: 2 bytes (MSB first).

Key: 16 bytes.

Success:

Frame	0x36	Status	Checksum

Status: status code returned from the card.

Please reference: MIFARE Plus Returned Status Code.

Failure:

Frame	0xC9	Checksum

Example:

Send: 0x00 17 00 36 01 00 04 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 24

Return: 0x00 04 01 36 33

5.7.3.2 MIFARE Plus Data Block Read

Function: Level 3 command, reading operation of data block; before reading, the relevant block need to be authorized.

Host sends:

Frame	0x37	Start Block	Blocks	Checksum

Start Block:	2 bytes (MSB first).
Blocks:	1 byte, blocks to be read

Success:

Frame	0x37	Status	Data	Checksum

Status: status code returned from the card.

Please reference: MIFARE Plus Returned Status Code.

Data: block * 16 bytes

Failure:

Frame	0xC8	Checksum

Example:

Send: 0x00 07 00 37 00 05 01 34

Return: 0x00 14 01 37 00 00 00 00 FF FF FF FF 00 00 00 00 05 FA 05 FA 22

5.7.3.3 MIFARE Plus Data Block Write

Function: Level 3 command, writing operation of data block; before writing, the relevant block need to be authorized.

Host sends:

Frame	0x38	Start Block	Blocks	Data	Checksum

Start Block:	2 bytes (MSB first).
Blocks:	1 byte, blocks to be writen
Data:	block * 16 bytes data to be writen

Success:

Frame	0x38	Status	Checksum

Status: status code returned from the card.

Please reference: MIFARE Plus Returned Status Code.

Failure:

Frame	0xC7	Checksum

Example:

Send: 0x00 17 00 38 00 04 01 00 00 00 00 FF FF FF FF 00 00 00 00 05 FA 05 FA 2A

Return: 0x00 04 01 38 3D

5.7.3.4 MIFARE Plus Purse Create

Function: Level 3 command, creating a block of MIFARE Plus as a purse.

Host sends:

Frame	0x39	Block	Value	Checksum

Block: 2 bytes (MSB first), block number.

Value: 4 bytes (LSB first), purse initial value.

Success:

Frame	0x39	Status	Checksum

Status: status code returned from the card.

Please reference: MIFARE Plus Returned Status Code.

Failure:

Frame	0xC6	Checksum

Example:

Send: 0x00 0A 00 39 00 04 00 01 00 00 36

Return: 0x00 04 01 39 3C

5.7.3.5 MIFARE Plus Purse Read

Function: Level 3 command, reading the balance of the purse.

Host sends:

Frame	0x3A	Block	Checksum

Block: 2 bytes (MSB first), block number.

Success:

Frame	0x3A	Status	Value	Checksum

Status: status code returned from the card.

Please reference: MIFARE Plus Returned Status Code.

Value: 4 bytes balance of the purse.

Failure:

Frame	0xC5	Checksum

Example:

Send: 0x00 06 01 3A 00 04 39

Return: 0x00 08 01 3A 00 01 00 00 32

5.7.3.6 MIFARE Plus Purse Increment

Function: Level 3 command, purse increment of MIFARE Plus.

Host sends:

Frame	0x3B	Block	Value	Checksum

Block: 2 bytes (MSB first), block number.

Value: 4 bytes (LSB first), value to increase.

Success:

Frame	0x3B	Status	Checksum

Status: status code returned from the card.

Please reference: MIFARE Plus Returned Status Code.

Failure:

Frame	0xC4	Checksum

Example:

Send: 0x00 0A 00 3B 00 04 00 01 00 00 34

Return: 0x00 04 01 3B 3E

5.7.3.7 MIFARE Plus Purse Decrement

Function: Level 3 command, purse decrement of MIFARE Plus.

Host sends:

Frame	0x3C	Block	Value	Checksum

Block: 2 bytes (MSB first), block number.

Value: 4 bytes (LSB first), value to decrease

Success:

Frame	0x3C	Status	Checksum

Status: status code returned from the card.

Please reference: MIFARE Plus Returned Status Code.

Failure:

Frame	0xC3	Checksum

Example:

Send: 0x00 0A 00 3C 00 04 00 01 00 00 33

Return: 0x00 04 01 3C 39

5.7.3.8 MIFARE Plus Purse Copy

Function: Level 3 command, copy the MIFARE Plus purse to another block in the same sector.

Host sends:

Frame	0x3D	Source	Target	Checksum

Source: 2 bytes (MSB first), source block number

Target: 2 bytes (MSB first), target block number

Success:

Frame	0x3D	Status	Checksum

Status: status code returned from the card.

Please reference: MIFARE Plus Returned Status Code.

Failure:

Frame	0xC2	Checksum

Example:

Send: 0x00 08 00 3D 00 04 00 05 34

Return: 0x00 04 01 3D 38

5.7.3.9 MIFARE Plus First Authenticate

Function: Level 1/3 Command. In Level 3, this command is use to authorize for data block, configuration block and AES key block before reading and writing.

Host sends:

Frame	0x3E	Address	Key	Checksum

Address: 2 bytes (MSB first), AES key address.

Key: 16 bytes, AES key

Success:

Frame	0x3E	Status	Checksum

Status: status code returned from the card.

Please reference: MIFARE Plus Returned Status Code.

Failure:

Frame	0xC1	Checksum

Example:

Send: 0x00 16 00 3E 40 02 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 6A

Return: 0x00 04 01 3E 3B

5.7.3.10 MIFARE Plus Following Authenticate

Function: Level 1/3 Command. In Level 3, this command is use to authorize for none data block before reading and writing. It is use to authorize again after first authentication.

Host sends:

Frame	0x3F	Address	Key	Checksum

Address: 2 bytes (MSB first), AES key address.

Key: 16 bytes, AES key

Success:

Frame	0x3F	Status	Checksum

Status: status code returned from the card.

Please reference: MIFARE Plus Returned Status Code.

Failure:

Frame	0xC0	Checksum

Example:

Send: 0x00 16 00 3F 40 02 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 6B

Return: 0x00 04 01 3F 3A

5.7.4 MIFARE Plus Returned Status Code

There is a status code after MIFARE Plus card response; this table indicates the possible value.

HEX	Status	Explanation
0x90	OPERATION_SUCCESS	Normal operation ends
0x06	AUTHENTICATION_ERROR	Authentication conditions are not met; No exist block; The block is visited in a numeric
0x07	COMMAND_OVERFLOW	Plaintext R&W in a task Overflow.
0x08	INVALID_MAC	MAC error.
0x88	INVALID_MAC of return	MAC error. of return from card
0x09	INVALID_BLOCK_NUMBER	Illegal block number
0x0A	NOT_EXIST_BLOCK_NUMBER	Block number does not exist.
0x0B	CONDITIONS_NOT_SATISFIED	Use conditions are not met.
0x0C	LENGTH_ERROR	Length error.
0x0F	GENERAL_MANIPULATION_ERROR	Cards Internal error.

5.8 DESFire Card Commands

We are here to provide a separate operation for DESFire card. DESFire card authentication and communication use DES encryption algorithm. The encryption of communication between QM-ABCM7 and DESFire cards is set by users. If the user sets the RF communication process is encrypted then the card data security is extremely high.

Use the following command allows the user to quickly start DESFire card R&D works. But for advanced user, also could use APDU to implement the card.

5.8.1 DESFire Prepare Commands

5.8.1.1 DESFire Request

DESFire card request, Please reference ISO14443 TYPE A Request.

5.8.1.2 DESFire RATS

DESFire card RATS, Please reference: ISO14443-4 TYPE-A card reset (RATS).

5.8.1.3 DESFire Request and RATS

This command support DESFire Request and RATS. Please reference: Card Request according to EMV and PBOC.

5.8.1.4 DESFire Authenticate

Function: Triple mutual authentication between DESFire and PCD. The authentication key number could be master Key or any other key.

The command means the host sends the key to the module. The module will process the authentication and send back results.

Advanced users could control the authentication process by themselves to improve security. We provide additional authentication interface. For details, please reference: 0x8E: DESFire Authenticate first step Get ekNo (RndB) and 0x8F: DESFire Authenticate second step get ekNo (RndA'). For all encryption and decryption methods related to DESFire refer to datasheet please. We also provide a tool to calculate the encryption and decryption. The source code of the tool is helpfull for users, if you need any assistance, please contact us.

Host sends:

Frame	0x90	KeyNo	Key	Checksum

KeyNo: 1 byte, the number of the key

Key: 16 bytes.

Success:

Frame	0x90	Status	SenssionKey	Checksum

Status: status code returned from the card, the communication between the card and module is successful, but it may not meet the conditions for the implementation.

Please reference: DESFire Returned State Code.

SenssionKey: 16 bytes. The senssion key will be sent back only after a successful authentication. The session key will be used in the following card operations. It is the key to decrypt the encrypted data in encrypted communication process.

Failure:

Frame	0x6F	Checksum

Example:

Send: 0x00 15 00 90 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 85

Return: 0x00 15 01 90 00 CC 6C E1 74 46 42 09 8D 1B 78 17 03 49 4C 67 A1 85

5.8.1.5 DESFire Authenticate first step Get ekNo (RndB)

Function: Authentication is initiated by the module. Get the ekNo (RndB) from the card.

Host sends:

Frame	0x8E	KeyNo	Checksum

KeyNo: 1 byte.

Success:

Frame	0x8E	Status	ekNo (RndB)	Checksum

Status: status code returned from the card.

Please reference: DESFire Returned State Code.

If the status code is 0xAF, it's correct. The host will offer further data, the following command must be: DESFire Authenticate second step get ekNo (RndA') then may go on authentication.

ekNo (RndB): 8 bytes, the result of random number encrypted by specified key. Use correct key to decrypt could get the RndB.

Failure:

Frame	0x71	Checksum

Example:

Send: 0x00 05 00 8E 00 8B

Return: 0x00 0D 01 8E AF 28 EA 37 7B 60 A0 DC F8 47

5.8.1.6 DESFire Authenticate second step Get ekNo (RndA')

Function: Random number RndA generated by the host. Host encryptes the assembled "RndA and RndB", and then send to card and get ekNo (RndA') from card, decrypting it to get RndA', reassembling to obtain RndA, if it is equal to RndA of generated by host, the authentication is passed.

Host sends:

Frame	0x8F	dkNo (RndA＋RndB')	Checksum

dkNo(RndA＋RndB'): 16bytes.

Success:

Frame	0x8F	Status	ekNo (RndA')	Checksum

Status: status code returned from the card.

Please reference: DESFire Returned State Code.

ekNo (RndA'): encrypted host random number. After decrypted with the correct key and reassemble, if equal to RndA, then the authentication is passed.

Sensssion Key: 16 bytes, Combination of RndA and RndB:

Sensssion Key = RndA[0..3]＋RndB[0..3]＋RndA[4..7]＋RndB[4..7]

The sensssion key will be used in the following card operations. It is the key to decrypt the encrypted data in encrypted communication process.

Failure:

Frame	0x70	Checksum

Example:

Send: 0x00 14 00 8F 42 FF CF C7 87 A1 90 32 B9 CC B8 A5 E7 70 C6 F5 66

Return: 0x00 0D 01 8F 00 28 4C 45 14 2A 60 17 67 8C

5.8.1.7 DESFire Select Application

Function: Select the specified card application. The following operation will be effective to this application.

Host sends:

Frame	0x98	AID	Checksum

AID: 3 bytes (LSB in first).

Success:

Frame	0x98	Status	Checksum

Status: status code returned from the card.

Please reference: DESFire Returned State Code.

Failure:

Frame	0x67	Checksum

Example:

Send: 0x00 07 00 98 12 34 56 EF

Return: 0x00 05 01 98 00 9C

5.8.2 DESFire Initialization Commands

5.8.2.1 DESFire Format Card

Function: Format card, all the card application and application files will be deleted.

Host sends:

Frame	0x99	Checksum

Success:

Frame	0x99	Status	Checksum

Status: status code returned from the card.

Please reference: DESFire Returned State Code.

Failure:

Frame	0x67	Checksum

Example:

Send: 0x00 04 00 99 9D

Return: 0x00 05 01 99 00 9D

5.8.2.2 DESFire Create Application

Function: Create new application.

Host sends:

Frame	0x95	AID	KeySett	NumOfKeys	Checksum

AID:	3 bytes (LSB in first).
KeySett:	1 byte, Key Setting
NumOfKeys:	1 byte.

Success:

Frame	0x95	Status	Checksum

Status: status code returned from the card.

Please reference: DESFire Returned State Code.

Failure:

Frame	0x6A	Checksum

Example:

Send: 0x00 09 00 95 12 34 56 EF 03 00

Return: 0x00 05 01 95 00 91

5.8.2.3 DESFire Change Key Settings

Function: Modify the master key/application master key configuration setting. DES/3DES encryption and CRC checksum will be used in the process of instruction execution.

Host sends:

Frame	0x91	KeySettings	Checksum

KeySettings: 8 bytes encrypted key settings.

Success:

Frame	0x91	Status	Checksum

Status: status code returned from the card.

Please reference: DESFire Returned State Code.

Failure:

Frame	0x6E	Checksum

Example:

Send: 0x00 0C 00 91 84 87 0F C4 44 83 B1 C9 EA

Return: 0x00 05 01 91 00 95

5.8.2.4 DESFire Get Key Settings

Function: Get the master key/appalication master key configureation settings.

Host sends:

Frame	0x92	Checksum

Success:

Frame	0x92	Status	KeySetting	Max.KeyNo	Checksum

Status: status code returned from the card.

Please reference: DESFire Returned State Code.

KeySetting: 1 byte

Max.KeyNo: 1byte, Max. Key numbers of current application

Failure:

Frame	0x6D	Checksum

Example:

Send: 0x00 04 00 92 96

Return: 0x00 07 01 92 00 EF 0C 77

5.8.2.5 DESFire Change Key

Function: Modify the key stored in the card. DES/3DES encryption and CRC checksum will be use in the process of instruction execution.

Host sends:

Frame	0x93	KeyID	ekKey	Checksum

KeyID: 1 byte.

ekKey: 24bytes (Refer to the datasheet for encryption calculations, or use the tools we provide and refer to source code).

Success:

Frame	0x93	Status	Checksum

Status: status code returned from the card.

Please reference: DESFire Returned State Code.

Failure:

Frame	0x6C	Checksum

Example:

Send: 0x00 1D 00 93 01 B5 30 7D 8F 42 7E D4 E3 C2 9B 0E 2B F0 A8 9D 49 59 35 9E 62 1F FE C8 00 BA

Return: 0x00 05 01 93 00 97

5.8.2.6 DESFire Get Key Version

Function: Get the key version information.

Host sends:

Frame	0x94	KeyID	Checksum

KeyID: 1 byte.

Success:

Frame	0x94	Status	Version	Checksum

Status: status code returned from the card.

Please reference: DESFire Returned State Code.

Version: 1 byte.

Failure:

Frame	0x6B	Checksum

Example:

Send: 0x00 05 00 94 01 90

Return: 0x00 06 01 94 00 F0 63

5.8.2.7 DESFire Delete Application

Function: Delete the specified application.

Host sends:

Frame	0x96	AID	Checksum

AID: 3 bytes (LSB in first).

Success:

Frame	0x96	Status	Checksum

Status: status code returned from the card.

Please reference: DESFire Returned State Code.

Failure:

Frame	0x69	Checksum

Example:

Send: 0x00 07 00 96 4A 4D 59 CF

Return: 0x00 05 01 96 00 92

5.8.2.8 DESFire Get Version

Function: Get card manufacturer and production information.

Host sends:

Frame	0x9A	Checksum

Success:

Frame	0x9A	Status	Data	Checksum

Status: status code returned from the card.

Please reference: DESFire Returned State Code.

Data: 28 bytes card manufacturer and production information.

Failure:

Frame	0x65	Checksum

Example:

Send: 0x00 04 00 9A 9E

Return: 0x00 21 01 9A 00 04 01 01 01 00 16 05 04 01 01 01 04 16 05 04 0B 30 9A 4F 22 80 BA 24 17 A9 20 07 11 E0

5.8.2.9 DESFire Get Application IDs

Function: Get all application identifier of the card.

Host sends:

Frame	0x97	Checksum

Success:

Frame	0x97	Status	AID	Checksum

Status: status code returned from the card.

Please reference: DESFire Returned State Code.

AID: Application identifier length is 3 bytes, the length is 3* identification number.

Failure:

Frame	0x68	Checksum

Example:

Send: 0x00 04 00 97 93

Return: 0x00 0B 01 97 00 4A 4D 07 4A 4D 59 C3

5.8.2.10 DESFire Get File IDs

Function: Get all file identifier of current application.

Host sends:

Frame	0x9B	Checksum

Success:

Frame	0x9B	Status	FID	Checksum

Status: status code returned from the card.

Please reference: DESFire Returned State Code.

FID: File ldentifier length is 1 byte, the total length is file number * 1 bytes.

Failure:

Frame	0x64	Checksum

Example:

Send: 0x00 04 00 9B 9F

Return: 0x00 08 01 9B 00 03 04 01 94

5.8.2.11 DESFire Get File Settings

Function: Get specified file setting in current application.

Host sends:

Frame	0x9C	FID	Checksum

FID: 1byte.

Success:

Frame	0x9C	Status	Data	Checksum

Status: status code returned from the card.

Please reference: DESFire Returned State Code.

Data: according to different types of files with different length, details as follows:

- Data file: 1 byte file type + 1 byte comm. setting + 2 bytes access right + 3 bytes file size.
- Value file: 1 byte file type + 1 byte comm. setting + 2 bytes access right + 4 bytes lower limit + 4 bytes upper limit + 4 bytes limited credit value + 1 byte limited credit enable.
- Record file: 1 byte file type + 1 byte comm. setting + 2 bytes access right + 3 bytes record size + 3 bytes Max record + 3 bytes current number of records.

  Note: The above multi-byte data are all LSB first.

Failure:

Frame	0x63	Checksum

Example:

Send: 0x00 05 00 9C 01 98

Return: 0x00 16 01 9C 00 02 00 EE EE 00 00 00 00 77 77 77 77 00 00 00 00 00 89

5.8.2.12 DESFire Change File Settings

Function: Modify specified file setting in current application.

Host sends:

Plaintext:

Frame	0x9D	File ID	Comm.Sett	AccessRight	Checksum

Cryptograph:

Frame	0x9D	File ID	EncryptedSetting	Checksum

File ID: 1 byte.

Comm.Sett: 1byte, 0: Plaintext; 1: MAC code checksum; 3: DES/3DES encryption.

AccessRight: 2 bytes (LSB in first).

EncryptedSetting: 8 bytes, 1 byte communication setting + 2 bytes file permission + 2 bytes CRC + 3 bytes 0x00 got via encryption.

Success:

Frame	0x9D	Status	Checksum

Status: status code returned from the card.

Please reference: DESFire Returned State Code.

Failure:

Frame	0x62	Checksum

Example:

Send: 0x00 08 00 9D 03 00 11 11 96

Return: 0x00 05 01 9D 00 99

5.8.2.13 DESFire Create STD Data File

Function: Create Standard Data File in current application.

Host sends:

Frame	0x9E	FID	Comm.Sett	AccessRight	Size	Checksum

FID:	1 byte.
Comm.Sett:	1 byte, 0: Plaintext; 1: MAC code checksum; 3: DES/3DES encryption.
AccessRight:	2 bytes (LSB in first).
Size:	3 bytes (LSB in first).

Success:

Frame	0x9E	Status	Checksum

Status: status code returned from the card.

Please reference: DESFire Returned State Code.

Failure:

Frame	0x61	Checksum

Example:

Send: 0x00 0B 00 9E 03 00 EE EE 00 01 00 97

Return: 0x00 05 01 9E 00 9A

5.8.2.14 DESFire Create Backup Data File

Function: Create Data File in current application, support backup mechanism (mirror). Then the file actual size is greater than or equal to DOUBLE size of specify file size and it is multiple of 32 bytes.

Host sends:

Frame	0x9F	FID	Comm.Sett	AccessRight	Size	Checksum

File ID:	1 byte.
Comm.Sett:	1byte, 0: Plaintext; 1:MAC code checksum; 3: DES/3DES encryption.
AccessRight:	2 bytes (LSB in first).
Size:	3 bytes (LSB in first).

Success:

Frame	0x9F	Status	Checksum

Status: status code returned from the card.

Please reference: DESFire Returned State Code.

Failure:

Frame	0x60	Checksum

Example:

Send: 0x00 0B 00 9F 04 00 EE EE 00 01 00 91

Return: 0x00 05 01 9F 00 9B

5.8.2.15 DESFire Create Value File

Function: Create Value File in current application, support backup mechanism.

Host sends:

Frame	0xA0	FID	Comm. Sett	Access Right	Lower limit	Upper limit	Value	Limited Credit enable	Checksum

FID:	1 byte.
Comm.Sett:	1byte, 0: Plaintext; 1: MAC code checksum; 3: DES/3DES encryption.
AccessRight:	2 bytes (LSB in first).
Lower limit:	4 bytes (Signed int, LSB in first).
Upper limit:	4 bytes (Signed int, LSB in first).
Value:	4 bytes (Signed int, LSB in first).
Limited Credit enable:	1 byte, 0: disable; 1: enable.

Success:

Frame	0xA0	Status	Checksum

Status: status code returned from the card.

Please reference: DESFire Returned State Code.

Failure:

Frame	0x5F	Checksum

Example:

Send: 0x00 15 00 A0 01 00 EE EE 00 00 00 00 77 77 77 77 22 22 22 22 00 B4

Return: 0x00 05 01 A0 00 A4

5.8.2.16 DESFire Create Linear Record File

Function: Create Linear Record File in current application, support backup mechanism.

Host sends:

Frame	0xA1	FID	Comm. Sett	Access Right	Record Size	Max Records	Checksum

FID:	1 byte.
Comm.Sett:	1byte, 0: Plaintext; 1: MAC code checksum; 3: DES/3DES encryption.
AccessRight:	2 bytes (LSB in first).
Record Size:	3 bytes (LSB in first), bytes of single record.
Max Records:	3 bytes (LSB in first), total record numbers of the file.

Success:

Frame	0xA1	Status	Checksum

Status: status code returned from the card.

Please reference: DESFire Returned State Code.

Failure:

Frame	0x5E	Checksum

Example:

Send: 0x00 0E 00 A1 08 00 00 00 20 00 00 20 00 00 A7

Return: 0x00 05 01 A1 00 A5

5.8.2.17 DESFire Create Cyclic Record File

Function: Create Cyclic Record File in the current application.

Host sends:

Frame	0xA2	FID	Comm. Sett	Access Right	Record Size	Max Records	Checksum

FID:	1 byte.
Comm.Sett:	1byte, 0: Plaintext; 1: MAC code checksum; 3: DES/3DES encryption.
AccessRight:	2 bytes (LSB in first).
Record Size:	3 bytes (LSB in first), bytes of single record.
Max Records:	3 bytes (LSB in first), total record numbers of the file.

Success:

Frame	0xA2	Status	Checksum

Status: status code returned from the card.

Please reference: DESFire Returned State Code.

Failure:

Frame	0x5D	Checksum

Example:

Send: 0x00 0E 00 A2 06 00 EE EE 10 00 00 10 00 00 AA

Return: 0x00 05 01 A2 00 A6

5.8.2.18 DESFire Delete File

Function: Delete specified file in current application.

Host sends:

Frame	0xA3	FID	Checksum

FID: 1 byte.

Success:

Frame	0xA3	Status	Checksum

Status: status code returned from the card.

Please reference: DESFire Returned State Code.

Failure:

Frame	0x5C	Checksum

Example:

Send: 0x00 05 00 A3 06 A0

Return: 0x00 05 01 A3 00 A7

5.8.3 DESFire Application Layer Commands

5.8.3.1 DESFire Read Data

Function: Read specified Data File (Standard Data File or Backup Fata File) in current application.

Host sends:

Frame	0xA4	FID	Offset	Length	Checksum

FID:	1 byte.
Offset:	3 bytes (LSB in first), offset in the file.
Length:	3 bytes (LSB in first), bytes need to be read.

Success:

Frame	0xA4	Status	Data	Checksum

Status: status code returned from the card.

Please reference: DESFire Returned State Code.

Data: data returned from the card.

Failure:

Frame	0x5B	Checksum

Example:

Send: 0x00 0B 00 A4 03 00 00 00 10 00 00 BC

Return: 0x00 15 01 A4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 B0

5.8.3.2 DESFire Write Data

Function: Write specified Data File (Standard Data File or Backup Fata File) in current application. For Backup Data File, Commit is needed to take effect after write, refer to: DESFire Commit Transaction please.

Host sends:

Frame	0xA5	FID	Offset	Length	Data	Checksum

FID: 1 byte.

Offset: 3 bytes (LSB in first), offset in the file.

Length: 3 bytes (LSB in first), bytes need to be written.

Data: The data to be written.

Success:

Frame	0xA5	Status	Checksum

Status: status code returned from the card.

Please reference: DESFire Returned State Code.

Failure:

Frame	0x5A	Checksum

Example:

Send: 0x00 14 00 A5 07 00 00 00 08 00 00 00 11 22 33 44 55 66 77 88 36

Return: 0x00 05 01 A5 00 A1

5.8.3.3 DESFire Get Value

Function: Read current value of specified Value File in current application.

Host sends:

Frame	0xA6	FID	Checksum

FID: 1 byte.

Success:

Frame	0xA6	Status	Data	Checksum

Status: status code returned from the card.

Please reference: DESFire Returned State Code.

Data: There are two lengths, depending on whether it is encrypted.

Plaintext: 4 bytes value (LSB in first).

Encryption: 8 bytes encrypted data, After decryption: 4 bytes value (LSB first) + 2 bytes CRC + 2 bytes 0x00.

Failure:

Frame	0x59	Checksum

Example:

Send: 0x00 05 00 A6 01 A2

Return: 0x00 09 01 A6 00 22 22 22 22 AE

5.8.3.4 DESFire Credit

Function: Increase value in specified Value File in current application. Commitment is needed to take effect after this operation, refer to: DESFire Commit Transaction please.

Host sends:

Frame	0xA7	FID	Data	Checksum

FID: 1 byte.

Data: There are two lengths, depending on whether it is encrypted.

Plaintext: 4 bytes value (LSB in first).

Encryption: 8 bytes encrypted data, After decryption: 4 bytes value (LSB first) + 2 bytes CRC + 2 bytes 0x00.

Success:

Frame	0xA7	Status	Checksum

Status: status code returned from the card.

Please reference: DESFire Returned State Code.

Failure:

Frame	0x58	Checksum

Example:

Send: 0x00 09 00 A7 01 01 00 00 00 AE

Return: 0x00 05 01 A7 00 A3

5.8.3.5 DESFire Debit

Function: Decrease value in specified Vale File in current application. Commitment is needed to take effect after this operation, refer to: DESFire Commit Transaction please.

Host sends:

Frame	0xA8	FID	Data	Checksum

FID: 1 byte.

Data: There are two lengths, depending on whether it is encrypted.

Plaintext: 4 bytes value (LSB in first).

Encryption: 8 bytes encrypted data, After decryption: 4 bytes value (LSB first) + 2 bytes CRC + 2 bytes 0x00.

Success:

Frame	0xA8	Status	Checksum

Status: status code returned from the card.

Please reference: DESFire Returned State Code.

Failure:

Frame	0x57	Checksum

Example:

Send: 0x00 09 00 A8 01 01 00 00 00 A1

Return: 0x00 05 01 A8 00 AC

5.8.3.6 DESFire Limited Credit

Function: Increase a limited value in specified Value File in current application without having full Read&Write permissions to the file. Commitment is needed to take effect after this operation, refer to: DESFire Commit Transaction please.

Host sends:

Frame	0xA9	FID	Data	Checksum

FID: 1 byte.

Data: There are two lengths, depending on whether it is encrypted.

Plaintext: 4 bytes value (LSB in first).

Encryption: 8 bytes encrypted data, After decryption: 4 bytes value (LSB first) + 2 bytes CRC + 2 bytes 0x00.

Success:

Frame	0xA9	Status	Checksum

Status: status code returned from the card.

Please reference: DESFire Returned State Code.

Failure:

Frame	0x56	Checksum

Example:

Send: 0x00 09 00 A9 01 01 00 00 00 A0

Return: 0x00 05 01 A9 00 AD

5.8.3.7 DESFire Write Record

Function: Write data to specified Data File in current application. The data file could be Linear Record or Cyclic Record file. This command appends one record at the end of the record file. The status will show an error when the linear record file is full. In case of cyclic record file is already full, it erases and overwrites the oldest record. Commitment is needed to take effect after this operation, refer to: DESFire Commit Transaction please.

Host sends:

Frame	0xAA	FID	Offset	Length	Data	Checksum

FID: 1 byte.

Offset: 3 bytes (LSB in first), offset in the record.

Length: 3 bytes (LSB in first), greater than 0 and less than or equal Record Size subtract the offset in the record.

Data: The data to be written.

Success:

Frame	0xAA	Status	Checksum

Status: status code returned from the card.

Please reference: DESFire Returned State Code.

Failure:

Frame	0x55	Checksum

Example:

Send: 0x00 1B 00 AA 06 00 00 00 10 00 00 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF A7

Return: 0x00 05 01 AA 00 AE

5.8.3.8 DESFire Read Record

Function: Read one or multi records from specified Record File in current application.

Host sends:

Frame	0xAB	FID	Offset	Length	Checksum

FID: 1 byte.

Offset: 3 bytes (LSB in first), offset of the record.

Length: 3 bytes (LSB in first), number of records to be read.

Success:

Frame	0xAB	Status	Data	Checksum

Status: status code returned from the card.

Please reference: DESFire Returned State Code.

Data: data returned from the card.

Failure:

Frame	0x54	Checksum

Example:

Send: 0x00 0B 00 AB 06 00 00 00 01 00 00 A7

Return: 0x00 15 01 AB 00 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF BF

5.8.3.9 DESFire Clear Record File

Function: Clear specified Record File of current application. Commitment is needed to take effect after this operation, refer to: DESFire Commit Transaction please.

Host sends:

Frame	0xAC	FID	Checksum

FID: 1 byte.

Success:

Frame	0xAC	Status	Checksum

Status: status code returned from the card.

Please reference: DESFire Returned State Code.

Failure:

Frame	0x53	Checksum

Example:

Send: 0x00 05 00 AC 05 AC

Return: 0x00 05 01 AC 00 A8

5.8.3.10 DESFire Commit Transaction

Function: Submit all WRITE operation of Backup Data file, Value file and Record file in current application. The modifications will be take effect after this operation.

Host sends:

Frame	0xAD	Checksum

Success:

Frame	0xAD	Status	Checksum

Status: status code returned from the card.

Please reference: DESFire Returned State Code.

Failure:

Frame	0x52	Checksum

Example:

Send: 0x00 04 00 AD A9

Return: 0x00 05 01 AD 00 A9

5.8.3.11 DESFire Abort Transaction

Function: Abort all WRITE operation of Backup Data file, Value file and Record file in current application.

Host sends:

Frame	0xAE	Checksum

Success:

Frame	0xAE	Status	Checksum

Status: status code returned from the card.

Please reference: DESFire Returned State Code.

Failure:

Frame	0x51	Checksum

Example:

Send: 0x00 04 00 AE AA

Return: 0x00 05 01 AE 00 AA

5.8.4 DESFire Returned State Code

Coding of Status and Error Codes of DESFire card

HEX code	Status	Explanation
0x00	OPERATION_OK	Successful operation
0x0C	NO_CHANGES	No changes done to backup files, CommitTransaction / AbortTransaction not necessary
0x0E	OUT_OF_EEPROM_ERROR	Insufficient NV-Memory to complete command
0x1C	ILLEGAL_COMMAND_CODE	Command code not supported
0x1E	INTEGRITY_ERROR	CRC or MAC does not matchdata Padding bytes not valid
0x40	NO_SUCH_KEY	Invalid key number specified
0x7E	LENGTH_ERROR	Length of command string invalid
0x9D	PERMISSION_DENIED	Current configuration / status does not allow the requested command
0x9E	PARAMETER_ERROR	Value of the parameter(s) invalid
0xA0	APLICATION_NOT_FOUND	Requested AID not present on PICC
0xA1	APPL_INTEGRITY_ERROR	Unrecoverable error within application, application will be disdabled *
0xAE	AUTHENTICATION_ERROR	Current authentication status does not allow the requested command
0xAF	ADDITIONAL_FRAME	Additional data frame is expected to be sent
0xBE	BOUNDARY_ERROR	Attempt to read/write data from/to beyond the files'/record's limits Attempt to exceed the limits of value file
0xC1	PICC_INTEGRITY_ERROR	Unrecoverable error within PICC, PICC will be disabled *
0xCA	COMMAND_ABOUTED	Previous Command was not fully completed Not all Frames were requested or provided by the PCD
0xCD	PICC_DISABLED_ERROR	PICC was disabled by an unrecoverable error *
0xCE	COUNT_ERROR	Number of Applications limited to 28, no additional CreatApplication possible
0xDE	DUPLICATE_ERROR	Creation of file/application failed because file/application with same number already exists
0xEE	EEPROM_ERROR	Could not complete NV-write operation due to loss of power, internal backup/rollback mechanism active *
0xF0	FILE_NOT_FOUND	Specified file number does not exist
0xF1	FILE_INTEGRITY_ERROR	Unrecoverable error within file, file will be disabled *

* These errors are not expected to appear during normal operation.

5.9 SR176 Card Commands

5.9.1 SR Serial Cards 1 Slot Initiate Card

Function: SR serial cards (SR176/SRI512/SRI1K/SRI2K/SRI4K/SRIX4K, the same below) single channel initiate card. Before read/write card, it needs to use the command of "SR serial cards select" to select the card. For more detailed card operations please refer to the card manual please.

Host sends:

Frame	0x63	Checksum

Success:

Frame	0x63	Card ID	Checksum

Card ID: 1 byte. It is a random ID.

Failure:

Frame	0x9C	Checksum

Example:

Send: 0x00 04 00 63 67

Return: 0x00 05 01 63 EE 89

5.9.2 SR Serial Cards Select

Function: Select a SR card as the CURRENT CARD.

Host sends:

Frame	0x65	Card ID	Checksum

Card ID: 1 byte, the card ID to select.

Success:

Frame	0x65	Card ID	Checksum

Card ID: 1 byte, the selected card ID

Failure:

Frame	0x9A	Checksum

Example:

Send: 0x00 05 00 65 EE 8E

Return: 0x05 01 65 EE 8F

5.9.3 SR Serial Cards Completion

Function: Set the CURRENT CARD into the completion status. If want to operate the card again, need to move the card out of the antenna RF effective field and initiate the card again.

Host sends:

Frame	0x67	Checksum

Success:

Frame	0x67	Checksum

Failure:

Frame	0x98	Checksum

Example:

Send: 0x00 04 00 67 63

Return: 0x00 04 01 67 62

5.9.4 SR176 Card Read

Function: Read data block of SR176 card.

Host sends:

Frame	0x68	StartBlock	BlockNumbers	Checksum

StartBlock:	1 byte.
BlockNumbers:	1 byte; the quantity of blocks to be read.

Success:

Frame	0x68	Data	Checksum

Data: 2 bytes * BlockNumbers, data from the card.

Failure:

Frame	0x97	Checksum

Example:

Send: 0x00 06 00 68 00 08 66

Return: 0x00 14 01 68 93 39 E9 0F 08 92 D0 02 EE EE EE EE EE EE EE EE 79

5.9.5 SR176 Card Write

Function: Write into the data block of SR176 card. After written, module will read the data to compare. If it is not equal, it will return failure.

Host sends:

Frame	0x69	StartBlock	BlockNumbers	Data	Checksum

StartBlock:	1 byte.
BlockNumbers:	1 byte; the quantity of blocks to be written.
Data:	2 bytes * BlockNumbers, data to write to the card.

Success:

Frame	0x69	Checksum

Failure:

Frame	0x96	Checksum

Example:

Send: 0x00 0A 00 69 08 02 88 88 99 99 69

Return: 0x00 04 01 69 6C

5.9.6 SR176 Block Lock

Function: Write Lock Register of SR176 card. The module will check the lock result after written.

Host sends:

Frame	0x6A	LOCK_REG	Checksum

LOCK_REG: 1byte, the lock register values to be written.

Success:

Frame	0x6A	Checksum

Failure:

Frame	0x95	Checksum

Example:

Send: 0x00 05 01 6A EF 81

Return: 0x00 04 01 6A 6F

5.10 SRI512/1K/2K/4K Card Commands

5.10.1 SRI Serial Cards 1 Slot Initiate Card

Please reference: SR serial cards 1 slot initiate card.

5.10.2 SRI Serial Cards 16 Slots Initiate Card

Function: SR serial cards (SRI512/SRI1K/SRI2K/SRI4K/SRIX4K, the same below) 16 channels initiate card.

Host sends:

Frame	0x64	Checksum

Success:

Frame	0x64	Status	Card ID	Checksum

Status: 16 bytes, the initiate result of 16 channels (0~15), 0x00: current channel success; 0xE8: current channel collision; 0xFF: current channel no card.

Card ID: 16 bytes; card ID of 16 channels; it is valid while the initial result of current channel is successful.

Failure:

Frame	0x9B	Checksum

Example:

Send: 0x00 04 00 64 60

Return: 0x00 24 01 64 FF FF FF FF FF FF FF FF FF FF FF FF 00 FF FF FF 00 00 00 00 00 00 00 00 00 00 00 00 7C 00 00 00 C2

5.10.3 SR Serial Cards Select

Please reference: SR serial cards select.

5.10.4 SRI Serial Cards Return to Inventory

Function: Set a selected SRI card returning to inventory status.

Host sends:

Frame	0x66	Checksum

Success:

Frame	0x66	Checksum

Failure:

Frame	0x99	Checksum

Example:

Send: 0x00 04 00 66 62

Return: 0x00 04 01 66 63

5.10.5 SR Serial Cards Completion

Please reference: SR serial cards completion.

5.10.6 SRI Serial Cards Read

Function: Read data block of SRI serial card.

Host sends:

Frame	0x6B	StartBlock	BlockNumbers	Checksum

StartBlock:	1 byte.
BlockNumbers:	1 byte; the quantity of blocks to be read.

Success:

Frame	0x6B	Data	Checksum

Data: 4 bytes * BlockNumbers, data from the card.

Failure:

Frame	0x94	Checksum

Example:

Send: 0x00 06 00 6B 08 04 61

Return: 0x00 14 01 6B 00 00 00 00 11 11 11 11 22 22 22 22 33 33 33 33 7E

5.10.7 SRI Serial Cards Write

Function: Write data block of SRI serial card. After written, module will read the data to compare. If it is not equal, it will return failure.

Host sends:

Frame	0x6C	StartBlock	BlockNumbers	Data	Checksum

StartBlock:	1 byte.
BlockNumbers:	1 byte; the quantity of blocks to be written.
Data:	4 bytes * BlockNumbers, data to be written to the card.

Success:

Frame	0x6C	Checksum

Failure:

Frame	0x93	Checksum

Example:

Send: 0x00 16 00 6C 08 04 00 00 00 00 11 11 11 11 22 22 22 22 33 33 33 33 76

Return: 0x00 04 01 6C 69

5.10.8 SRI Serial Cards Block Lock

Function: Write Lock Register of SRI serial card. The module will check the lock result after written.

Host sends:

Frame	0x6D	LOCK_REG	Checksum

LOCK_REG: 1byte, the lock register values to be written.

Success:

Frame	0x6D	Checksum

Failure:

Frame	0x92	Checksum

Example:

Send: 0x00 05 00 6D FF 97

Return: 0x00 04 01 6D 68

5.10.9 SRI Serial Cards Read UID

Function: Read UID of SRI serial card.

Host sends:

Frame	0x6E	Checksum

Success:

Frame	0x6E	UID	Checksum

UID: 8 bytes, UID of CURRENT CARD.

Failure:

Frame	0x91	Checksum

Example:

Send: 0x00 04 00 6E 6A

Return: 0x00 0C 01 6E D0 02 0C B3 E3 CC E9 7F B7

5.10.10 SRIX Serial Cards Authentication

Function: SRIX serial card authentication; Anti clone function of the SRIX serial card.

Host sends:

Frame	0x6F	Data	Checksum

Data: 6 bytes, data input.

Success:

Frame	0x6F	Result	Checksum

Result: 3 bytes, result return.

Failure:

Frame	0x90	Checksum

Example:

Send: 0x 00 0A 00 6F 12 34 56 78 90 AB 56

Return: 0x00 07 01 6F 0C B3 E3 35

5.11 SAM or CPU Card Commands

5.11.1 SAM or CPU Card Reset

Function: Reset SAM or CPU card, get ATR and set the relevant communication parameters.

Host sends:

Frame	0x4D	SAM.No	BaudRate	Checksum

SAM.No:	1 byte, 0: CPU card, other value: SAM slot number.
BaudRate:	1 byte, reset baud rate. 0: 9600bps; 1: 19200bps; 2: 38400 bps; 3: 55800 bps; 4: 57600 bps; 5: 115200 bps; 6: 230400 bps; other value: RFU.

Success:

Frame	0x4D	SAM.No	ATR	Checksum

ATR:	SAM card reset information, length depending on the card

Failure:

Frame	0xB2	Checksum

Example:

Send: 0x00 06 00 4D 01 00 4A

Return: 0x00 15 01 4D 01 3B 6C 00 02 43 21 86 38 07 54 42 00 16 0E 5A 2F AD

5.11.2 Set PPS of SAM or CPU Card

Function: Using PPS to modify the communication baud rate after SAM reset.

Host sends:

Frame	0x4E	SAM.No	BaudRate	Checksum

SAM.No:	1 byte, 0: CPU card, other value: SAM slot number.
BaudRate:	1 byte, reset baud rate. 0: 9600bps; 1: 19200bps; 2: 38400 bps; 3: 55800 bps; 4: 57600 bps; 5: 115200 bps; 6: 230400 bps; other value: RFU.

Success:

Frame	0x4E	SAM.No	Checksum

Failure:

Frame	0xB1	Checksum

Example:

Send: 0x00 06 00 4E 01 05 4C

Return: 0x00 05 01 4E 01 4B

5.11.3 Send APDU to SAM or CPU Card

Function: send APDU (COS command) to SAM and get result.

Host sends:

Frame	0x4F	SAM.No	APDU	Checksum

SAM.No:	1 byte, 0: CPU card, other value: SAM slot number.

APDU: APDU need to send.

Success:

Frame	0x4F	SAM.No	Response	Checksum

Response: response of SAM, length depending on the type of APDU.

Failure:

Frame	0xB0	Checksum

Example:

Send: 0x00 0A 00 4F 01 00 84 00 00 08 C8

Return: 0x00 0F 01 4F 01 58 CE 18 13 43 E3 6B 10 90 00 96

5.11.4 Power Down and Eject CPU Card

Function: Power down and eject CPU card from slot 0.

Host sends:

Frame	0x4C	SAM.No	Checksum

SAM.No: 1 byte, 0: CPU card, other value: SAM slot number.

Success:

Frame	0x4C	SAM.No	Checksum

Failure:

Frame	0xB3	Checksum

Example:

Send: 0x00 05 00 4C 00 49

Return: 0x00 05 00 4C 00 49

5.12 ISO15693 Operation Commands

5.12.1 ISO15693 Inventory

Function: Find a card in RF effective field. If it is successful, set the tag as CURRENT TAG.

If automatic detecting card function is turned on, this command will take the result of automatic detecting card, it won't to detect card after received the command.

Host sends:

Frame	0x5C	AFI	Checksum

AFI: 1byte AFI, detect card equal to AFI only.

If not use AFI, then host sends:

Frame	0x5C	Checksum

Success:

Frame	0x5C	DSFID	UID	Checksum

DSFID: 1 byte, DSFID of CURRENT TAG.

UID: 8 bytes (LSB in first), UID of CURRENT TAG.

Failure:

Frame	0xA3	Checksum

Example:

Send: 0x00 05 00 5C 00 59

Return: 0x00 0D 01 5C 33 E3 DB CF 19 00 00 07 E0 6A

5.12.2 ISO15693 Inventory all Tags

Function: Find all tags and output the UIDs. And the tags will in quiet mode after this command.

Host sends:

Frame	0x7C	AFI	Checksum

AFI: 1byte AFI, detect card equal to AFI only.

If not use AFI, then host sends:

Frame	0x7C	Checksum

Success:

Frame	0x7C	N * (DSFID + UID)	Checksum

N: number of tags found.

DSFID: 1 byte

UID: 8 bytes (LSB in first)

Failure:

Frame	0x83	Checksum

Example:

Send: 0x00 05 00 7C 00 79

Return: 0x00 16 01 7C 00 3B 8B FC A3 00 01 04 E0 00 3B A1 D9 12 00 01 04 E0 D5

5.12.3 ISO15693 Stay Quiet

Function: Set the CURRENT TAG stay quiet. This command is only for "Inventory" and "get system information". Read and write card commands are based on the address, so even with this command; it could also read and write operations.

Host sends:

Frame	0x5D	Checksum

Success:

Frame	0x5D	Checksum

Failure:

Frame	0xA2	Checksum

Example:

Send: 0x00 04 00 5D 59

Return: 0x00 04 01 5D 58

5.12.4 ISO15693 Get System Information

Function: Get the system information of CURRENT TAG.

Host sends:

Frame	0x5E	Checksum

Success:

Frame	0x5E	Data	Checksum

Data: system information, the content to depend on the functions of the card, please refers to the data sheet of the card.

Failure:

Frame	0xA1	Checksum

Example:

Send: 0x00 04 00 5E 5A

Return: 0x00 12 01 5E 0F E3 DB CF 19 00 00 07 E0 33 23 3F 03 8B EC

5.12.5 ISO15693 Reset to Ready

Function: Set a stay quiet TAG reset to ready.

Host sends:

Frame	0x5F	UID	Checksum

Data: 8 bytes, UID of the tag to reset to ready.

Success:

Frame	0x5F	Checksum

Failure:

Frame	0xA0	Checksum

Example:

Send: 0x00 0C 00 5F E3 DB CF 19 00 00 07 E0 5A

Return: 0x00 04 01 5F 5A

5.12.6 ISO15693 Read Blocks

Function: Read data blocks of CURRENT TAG.

Host sends:

Frame	0x54	StartBlock	BlockNumbers	Checksum

StartBlock:	1 byte, the start block number to be read.
BlockNumbers:	1 byte, number of blocks to be read, Max. 62.

Success:

Frame	0x54	Data	Checksum

Data:	Blocks * bytes per block (depend on the cards).

Failure:

Frame	0xAB	Checksum

Example:

Send: 0x00 06 00 54 08 02 58

Return: 0x00 0C 01 54 88 88 88 88 99 99 99 99 59

5.12.7 ISO15693 Write Blocks

Function: Write data blocks of CURRENT TAG.

Host sends:

Frame	0x55	StartBlock	BlockNumbers	Data	Checksum

StartBlock:	1 byte, start block number to be written.
BlockNumbers:	1 byte, number of blocks to be written, Max. 62.
Data:	Blocks * 4 bytes..

Success:

Frame	0x55	Checksum

Failure:

Frame	0xAA	Checksum

Example:

Send: 0x00 0E 00 55 08 02 11 11 11 11 22 22 22 22 51

Return: 0x00 04 01 55 50

5.12.8 ISO15693 Lock Block

Function: Lock a block of CURRENT TAG.

Host sends:

Frame	0x56	BlockNumber	Checksum

BlockNumber: 1 byte, block number to be locked.

Success:

Frame	0x56	Checksum

Failure:

Frame	0xA9	Checksum

Example:

Send: 0x00 05 00 56 08 5B

Return: 0x00 04 01 56 53

5.12.9 ISO15693 Write AFI

Function: Write AFI to CURRENT TAG.

Host sends:

Frame	0x57	AFI	Checksum

AFI: 1 byte, AFI value to be written.

Success:

Frame	0x57	Checksum

Failure:

Frame	0xA8	Checksum

Example:

Send: 0x00 05 00 57 08 5A

Return: 0x00 04 01 57 52

5.12.10 ISO15693 Lock AFI

Function: Lock AFI of CURRENT TAG.

Host sends:

Frame	0x58	Checksum

Success:

Frame	0x58	Checksum

Failure:

Frame	0xA7	Checksum

Example:

Send: 0x00 04 00 58 5C

Return: 0x00 04 01 58 5D

5.12.11 ISO15693 Write DSFID

Function: Write DSFID of CURRENT TAG.

Host sends:

Frame	0x59	DSFID	Checksum

DSFID: 1 byte, DSFID value to be written.

Success:

Frame	0x59	Checksum

Failure:

Frame	0xA6	Checksum

Example:

Send: 0x00 05 00 59 09 55

Return: 0x00 04 01 59 5C

5.12.12 ISO15693 Lock DSFID

Function: Lock DSFID of CURRENT TAG.

Host sends:

Frame	0x5A	Checksum

Success:

Frame	0x5A	Checksum

Failure:

Frame	0xA5	Checksum

Example:

Send: 0x00 04 00 5A 5E

Return: 0x00 04 01 5A 5F

5.12.13 ISO15693 Get Blocks Security

Function: Get blocks security information of CURRENT TAG.

Host sends:

Frame	0x5B	StartBlock	BlockNumbers	Checksum

StartBlock:	1 byte, the start block number to be gotten security.
BlockNumbers:	1 byte, number of blocks to be gotten security.

Success:

Frame	0x5B	Data	Checksum

Data: bytes equal to BlockNumbers, the locked information of data block.

Failure:

Frame	0xA4	Checksum

Example:

Send: 0x00 06 00 5B 00 40 1D

Return: 0x00 44 01 5B 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00 00 00 1F

5.13 I.CODE 1 Operation Commands

5.13.1 I.CODE1 Inventory

Function: Search I.CODE1 card in RF effective field.

Host sends:

Frame	0x80	Checksum

Success:

Frame	0x80	SNR	Checksum

SNR: 8 bytes.

Failure:

Frame	0x7F	Checksum

Example:

Send: 0x00 04 00 80 84

Return: 0x00 0C 01 80 5D 9A D4 0F 00 00 00 01 90

5.13.2 I.CODE 1 Read

Function: Read data from I.CODE1.

Host sends:

Frame	0x81	BlockNumber	Checksum

BlockNumber: 1byte, value: 0x00 to 0x0F.

Success:

Frame	0x81	Data	Checksum

Data: 4bytes.

Failure:

Frame	0x7E	Checksum

Example:

Send: 0x00 05 00 81 08 8C

Return: 0x00 08 01 81 FF FF FF FF 88

5.13.3 I.CODE 1 Write

Function: To write data into I.CODE1.

Host sends:

Frame	0x82	BlockNumber	Data	Checksum

BlockNumber:	1 byte.
Data:	4 bytes.

Success:

Frame	0x82	Checksum

Failure:

Frame	0x7D	Checksum

Example:

Send: 0x00 09 00 82 08 12 34 56 78 8B

Return: 0x00 04 01 82 87

5.13.4 I.CODE 1 Stay Quiet

Function: I.CODE1 stays quiet.

Host sends:

Frame	0x83	Checksum

Success:

Frame	0x83	Checksum

Failure:

Frame	0x7C	Checksum

Example:

Send: 0x00 04 00 83 87

Return: 0x00 04 01 83 86

5.14 NFC Functions

5.14.1 Set NFC Device Working Mode

Function: Set NFC device working mode.

Host sends:

Frame	0xC0	MODE	Checksum

Success:

Frame	0xC0	Checksum

MODE: 1 byte

= 0x00 passive initiator (default)

= 0x01 passive target

= 0x02 active initiator

= 0x03 active target

= 0x04 NFC Tag simulation

Failure:

Frame	0x3F	Checksum

Example:

Send: 0x00 05 00 C0 01 C4

Return: 0x00 04 01 C0 C5

5.14.2 Initiator Commands

5.14.2.1 Initiator Request Targets

Function: initiator request target attributes over RF interface.

Host sends:

Frame	0xC1	Checksum

Success:

Frame	0xC1	DATA	Checksum

DATA: returned message from target

Failure:

Frame	0x3E	Checksum

Example:

Send: 0x00 04 00 C1 C5

Return: 0x00 20 01 C1 31 32 33 34 35 31 32 33 34 35 00 00 00 0C 32 46 66 6D 01 01 11 03 02 00 13 04 01 96 03

5.14.2.2 Data Exchange

Function: initiator exchange data with target over RF interface.

Host sends:

Frame	0xC2	DTS	Checksum

DTS: Data to exchange

Success:

Frame	0xC2	DATA	Checksum

DATA: returned data from target

Failure:

Frame	0x3D	Checksum

Example:

Send: 0x00 0C 00 C2 11 22 33 44 55 66 77 88 46

Return: 0x00 08 01 C2 12 34 56 78 C3

5.14.3 Target Commands

5.14.3.1 Target Prepare Exchange Data

Function: store exechange data to target to wait initiator exchange command from RF interface.

Host sends:

Frame	0xC3	DATA	Checksum

DATA: data to exchange

Success:

Frame	0xC3	Checksum

Failure:

Frame	0x3C	Checksum

Example:

Send: 0x00 08 00 C3 12 34 56 78 C3

Return: 0x00 04 01 C3 C6

5.14.3.2 Target Read Status and Exchange Result

Function: read RF communication status, if data exchange finished then return exchange data.

Host sends:

Frame	0xC4	Checksum

Success:

Frame	0xC4	DATA	Checksum

DATA: returned message from target

Failure:

Frame	0x3B	Checksum

Example:

Send: 0x00 04 00 C4 C0

Return: 0x00 0D 01 C4 00 11 22 33 44 55 66 77 88 40

5.15 NFC Tag Functions

5.15.1 Summary

Some module with NFC target function could be set to NFC Tag mode. It could be operated by NFC Tag readers after set. The commands for target are command to operate NFC Tag FLASH space over communication port. The commands for initiator are command to operate NFC Tag FLASH space over RF interface.

The data storage space is 128 bytes. Organization is 4bytes/page * 32 pages. The page 0 and page 1 are read only. First byte and second byte in page 2 are read only too.

5.15.2 Initiator Commands

5.15.2.1 Read Data from NFC Tag

Function: initiator (reader) read data from NFC Tag over RF interface. The NFC Tag responds to the READ command by sending 16 bytes starting from the page address defined in the command (e.g. if address is ‘03h‘ pages 03h, 04h, 05h, 06h are returned). If address is ‘1Fh’, the contents of pages 1Fh, 00h, 01h and 02h are returned).

Host sends:

Frame	0xC8	PAGE	Checksum

PAGE: 1 byte start page number, 0x00 to 0x1F

Success:

Frame	0xC8	DATA	Checksum

DATA: 16 bytes data from NFC Tag

Failure:

Frame	0x37	Checksum

Example:

Send: 0x00 05 00 C8 07 CA

Return: 0x00 14 01 C8 07 07 07 07 08 08 08 08 FF FF FF FF FF FF FF FF DD

5.15.2.2 Write Data to NFC Tag

Function: A WRITE command is performed page-wise, programming 4 bytes in a page.

Host sends:

Frame	0xC9	PAGE	DATA	Checksum

PAGE: 1 byte page number

DATA: 4 bytes data to write

Success:

Frame	0xC9	Checksum

DATA: returned message from target

Failure:

Frame	0x36	Checksum

Example:

Send: 0x00 09 00 C9 07 07 07 07 07 C7

Return: 0x00 04 01 C9 CC

5.15.3 Target Commands

5.15.3.1 Read Data from NFC Tag

Function: read data from NFC Tag over communication port.

Host sends:

Frame	0xCA	PAGE	NUMBER	Checksum

PAGE: 1 byte start page

NUMBER: number of pages to read

Success:

Frame	0xCA	DATA	Checksum

DATA: retruned data from NFC Tag

Failure:

Frame	0x35	Checksum

Example:

Send: 0x00 06 00 CA 07 01 CA

Return: 0x00 08 01 CA 07 07 07 07 C3

5.15.3.2 Write Data to NFC Tag

Function: write data to NFC Tag over communication port.

Host sends:

Frame	0xCB	PAGE	NUMBER	DATA	Checksum

PAGE: 1 byte start page

NUMBER: number of pages to write

DATA: data to write

Success:

Frame	0xCB	Checksum

DATA: returned message from target

Failure:

Frame	0x34	Checksum

Example:

Send: 0x00 0E 00 CB 07 02 07 07 07 07 08 08 08 08 C0

Return: 0x00 04 01 CB CE

5.15.3.3 Write UID of NFC Tag

Function: Write UID of NFC Tag over communication port.

Host sends:

Frame	0xCC	UID	Checksum

UID: 7 bytes UID to write

Success:

Frame	0xCC	Checksum

Failure:

Frame	0x33	Checksum

Example:

Send: 0x00 0B 00 CC 01 02 03 04 05 06 07 C7

Return: 0x00 04 01 CC C9

5.16 FeliCa Command

5.16.1 Transaction of FeliCa command

Function: Transmit and command and get the result from FeliCa card.

Host sends:

Frame	0x2F	DATA	Checksum

DATA: FeliCa command and parameters.

Success:

Frame	0x2F	RETURN	Checksum

RETURN: data return from FeliCa card.

Failure:

Frame	0xD0	Checksum

Example:

Send: 0x00 0A 00 2F 06 00 FF FF 01 00 22

Return: 0x00 18 01 2F 14 01 01 2E 30 C3 76 94 48 1F 03 35 0B 82 82 44 83 FF 12 FC A1

5.17 ISO18000-3M3

5.17.1 Returned Status Code

Retrned status code in reply of module

Error-Coder Support	Error Code (binary)	Error-Code Name
Error specific	0000 0000	Other error
	0000 0011	Memory overrun
	0000 0100	Memory locked
	0000 1011	Insufficient power
Non specific	0000 1111	Non specific error
Reader operation Error	1111 1111	No tag in field

5.17.2 ActivateCard

Function: Get the Handle of single tag

Host sends:

Frame	0x85	0x00	Checksum

Success:

Frame	0x85	0x00	Handle	Checksum

Handle: handle of the tag, it will use in other command

Failure:

Frame	0x7A	0x00	Error Code	Checksum

Error Code: Returned status code from tag

Example:

Send: 00 05 00 85 00 80

Return: 00 07 01 85 00 CF 16 5A

5.17.3 ActivateCards

Function: Get A Handle of multi-tags

Host sends:

Frame	0x85	0x01	Q	Checksum

Q: 1 byte, number of time slots ($2^{Q,}$ the max. time slots are 2^{Q}), value range from 0x00 to 0x05 (Max. 32 tags for common products), large Q means more time needs.

Success:

Frame	0x85	0x01	Handle	Checksum

Failure:

Frame	0x7A	0x01	ErrorCode	Checksum

Error Code: Returned status code from tag

Example:

Send: 00 06 00 85 01 01 83

Return: 00 07 01 85 01 C9 1C 57

5.17.4 Read

Function: Read data of the tag

Host sends:

Frame	0x85	0x02	ReadInfo	Checksum

ReadInfo: 6 to 9 bytes parameters, depending on byte3 length

ReadInfo	Byte1	MemBank	Memory bank select 00h: Reserved 01h: UII 02h: TID 03h: User
	Byte2	WordPtrlength	Length of wordpointer 00h: 8 bit (byte3,WordPtr is 1 byte) 01h: 16 bit (byte3,WordPtr are 2 bytes) 02h: 24 bit (byte3,WordPtr are 3 bytes) 03h: 32 bit (byte3,WordPtr are 4 bytes)
	Byte3(Note1)	WordPtr	Starting address pointer (MSB first)
	Byte4	WordCount	The number of 16-bit words to be read
	Byte5_6	handle	2 bytes handle

Note1: the bytes of this parameter are depending on WordPtrLength. The WrodPtr will be 2 bytes if WordPtrLength equal to 0x02.

Success:

Frame	0x85	0x02	Data	Checksum

DATA: n words (16 bits) + handle, data from tag

Failure:

Frame	0x7A	0x02	Error Code	Checksum

Error Code: Returned status code from tag

Example:

Send: 00 0B 00 85 02 00 00 00 04 A2 D4 FE

Return: 00 0F 01 85 02 00 00 00 00 00 00 00 00 A2 D4 FF

5.17.5 Write

Function: write a Word(16bits) to tag

Host sends:

Frame	0x85	0x03	WriteInfo	Checksum

WriteInfo: 7 to 10 bytes parameter and data

WriteInfo	Byte1	MemBank	Memory bank select 00h: Reserved 01h: UII 02h: TID 03h: User
	Byte2	WordPtrlength	Length of wordpointer 00h: 8 bit (byte3,WordPtr is 1 byte) 01h: 16 bit (byte3,WordPtr are 2 bytes) 02h: 24 bit (byte3,WordPtr are 3 bytes) 03h: 32 bit (byte3,WordPtr are 4 bytes)
	Byte3(Note1)	WordPtr	Starting address pointer (MSB first)
	Byte4_5	Data	A 16-bit word to be written
	Byte6_7	handle	2 bytes handle

Success:

Frame	0x85	0x03	Handles	Checksum

Failure:

Frame	0x7A	0x03	Error Code	Checksum

Error Code: Returned status code from tag

Example:

Send: 00 0C 00 85 03 00 00 00 11 11 A2 D4 FC

Return: 00 07 01 85 03 A2 D4 F6

Send: 00 0B 00 85 02 00 00 00 04 A2 D4 FE

Return: 00 0F 01 85 02 11 11 00 00 00 00 00 00 A2 D4 FF

5.17.6 KILL or Recommissioning

Function: Kill or recommissioning the tag. Ref. to Information technology — Radio frequency identification for item management —Part 3: Parameters for air interface communications at 13.56 MHz

Host sends:

Frame	0x85	0x04	KillInfo	Checksum

KillInfo: 7 byes parameter

KillInfo	Byte1_4	Kill password	The kill password is a 32-bit value stored in Reserved memory 00h to 1Fh, MSB first。
	Byte5	Recom	Recommissioning bits。
	Byte6_7	handle	2 bytes handle

Success:

Frame	0x85	0x04	Handles	Checksum

Failure:

Frame	0x7A	0x04	Error Code	Checksum

Error Code: Returned status code from tag

Example:

Send: 00 0C 00 85 04 11 11 00 00 04 12 9E 05

Return: 00 07 01 85 04 12 9E 0B

5.17.7 Lock

Function: Lock the tag by lock info.

Host sends:

Frame	0x85	0x05	LockInfo	Checksum

LockInfo: 6 bytes parameter

WriteInfo	Byte1_2	Mask	Kill pwd	Bit15	Skip/write
				Bit14	Skip/write
			Access pwd	Bit13	Skip/write
				Bit12	Skip/write
			UII memory	Bit11	Skip/write
				Bit10	Skip/write
			TID memory	Bit9	Skip/write
				Bit8	Skip/write
			User memory	Bit7	Skip/write
				Bit6	Skip/write
			RFU	Bit5_0	RFU
	Byte3_4	Action	Kill pwd	Bit15	Pwd read/write
				Bit14	Perma lock
			Access pwd	Bit13	Pwd read/write

				Bit12	Perma lock
			UII memory	Bit11	Pwd write
				Bit10	Perma lock
			TID memory	Bit9	Pwd write
				Bit8	Perma lock
			User memory	Bit7	Pwd write
				Bit6	Perma lock
			RFU	Bit5_0	RFU
	Byte5_6	handle	2 bytes handle		

Pwd write	**Perma lock**	**Description**
0	0	Associated memory bank is writeable from either the open or secured states
0	1	Associated memory bank is permanently writeable from either the open or secured states and may never be locked.
1	0	Associated memory bank is writeable from the secured state but not from the open state.
1	1	Associated memory bank is not writeable from any state.
Pwd read/write	**Perma lock**	Description
0	0	Associated password location is readable and writeable from either the open or secured states.
0	1	Associated password location is permanently readable and writeable from open or secured or secured states and may never be locked.
1	0	Associated password location is readable and writeable from the secured state but not from the open state.
1	1	Associated password location is not readable or writeable from any state.

Success:

Frame	0x85	0x05	Handles	Checksum

Failure:

Frame	0x7A	0x05	Error Code	Checksum

Error Code: Returned status code from tag

Example:

Send: 00 0B 00 85 05 30 00 20 00 12 9E 17

Return: 00 07 01 85 05 12 9E 0A

5.17.8 Access

Function: get the access condition

Host sends:

Frame	0x85	0x06	AccessInfo	Checksum

KillInfo: 6 bytes parameter

KillInfo	Byte1_4	Access password	The access password is a 32-bit value stored in Reserved memory 20h to 3Fh, MSB first。
	Byte5_6	handle	2 bytes handle

Success:

Frame	0x85	0x06	Handles	Checksum

Failure:

Frame	0x 7A	0x06	Error Code	Checksum

Error Code: Returned status code from tag

Example:

Send: 00 0B 00 85 06 00 00 00 00 12 9E 04

Return: 00 07 01 85 06 12 9E 09

5.17.9 BlockWrite

Function: Block Write

Host sends:

Frame	0x85	0x07	WriteInfo	Checksum

WriteInfo: n bytes parameter and data

WriteInfo	Byte1	MemBank	Memory bank 00h: Reserved 01h: UII 02h: TID 03h: User
	Byte2	WordPtrlength	Length of wordpointer 00h: 8 bit (byte3,WordPtr is 1 byte) 01h: 16 bit (byte3,WordPtr are 2 bytes) 02h: 24 bit (byte3,WordPtr are 3 bytes) 03h: 32 bit (byte3,WordPtr are 4 bytes)
	Byte3(Note1)	WordPtr	Starting address pointer
	Byte4	WordCount	Number of words(16-bit) to write

			(For i.Code ILT-m, WordCount = 01h to 02h. If WordCount = 00h, the tag shall ignore the BlockWrite. If WordCount = 01h, the tag shall write a single data word. The max. write length is 2 words)
	Byte5_n	Data	Data to be written
	Byten+1_n+2	handle	2 bytes handle

Success:

Frame	0x85	0x07	Handles	Checksum

Failure:

Frame	0x 7A	0x07	Error Code	Checksum

Error Code: Returned status code from tag

Example:

Send: 00 0F 00 85 07 00 00 02 02 11 22 33 44 12 9E 45

Return: 00 07 01 85 07 12 9E 08

Send: 00 0B 00 85 02 00 00 00 04 12 9E 04

Return: 00 0F 01 85 02 11 11 00 00 11 22 33 44 12 9E 41

5.17.10 BlockErase

Function: Block Erase

Host sends:

Frame	0x85	0x08	EraseInfo	Checksum

EraseInfo: parameter and data

EraseInfo	Byte1	MemBank	Memory bank 00h: Reserved 01h: UII 02h: TID 03h: User
	Byte2	WordPtrlength	Length of wordpointer 00h: 8 bit (byte3,WordPtr is 1 byte) 01h: 16 bit (byte3,WordPtr are 2 bytes) 02h: 24 bit (byte3,WordPtr are 3 bytes) 03h: 32 bit (byte3,WordPtr are 4 bytes)
	Byte3	WordPtr	Starting address pointer
	Byte4	WordCount	Number of words(16-bit) to erase (For i.Code ILT-m, WordCount = 01h to

			02h. If WordCount = 00h, the tag shall ignore the BlockWrite. If WordCount = 01h, the tag shall write a single data word. The max. write length is 2 words)
	Byte5_6	handle	2 bytes handle

Success:

Frame	0x85	0x08	Handles	Checksum

Failure:

Frame	0x 7A	0x08	Error Code	Checksum

Error Code: Returned status code from tag

5.17.11 Inventory

Function: Inventory

Host sends:

Frame	0x85	0x0A	Q	Checksum

Q: 1 byte, number of time slots ($2^{Q,}$ the max. time slots are 2^Q), value range from 0x00 to 0x05 (Max. 32 tags for common products), large Q means more time needs.

Success:

Frame	0x85	0x0A	AcknowLedged	Checksum

AcknowLedged: 1byte tag number n + n bytes tag Info. Length + tag Info. See below.

Failure:

Frame	0x 7A	0x0A	Error Code	Checksum

Error Code: Returned status code from tag

Example:

Send: 00 06 00 85 0A 04 8D

Return: 00 3A 01 85 0A 04 0C 0C 0C 0C 00 00 00 00 00 00 40 00 FA B1 F5 23 00 00 00 00 00 00 40 00 FA B1 4C AB 00 00 00 00 00 00 40 00 FA B1 39 46 00 00 00 00 00 00 40 00 FA B1 58 92 34

There are 04 tags found. The Infomation length of the 4 tags are 0C 0C 0C 0C. The Infomation of the 4 tags are 00 00 00 00 00 00 40 00 FA B1 F5 23 00 00 00 00 00 00 40 00 FA B1 4C AB 00 00 00 00 00 00 40 00 FA B1 39 46 00 00 00 00 00 00 40 00 FA B1 58 92

5.17.12 ReqRnCard

Function: get the handle of the tag, uses with Inventory

Host sends:

Frame	0x85	0x0B	Info	Checksum

Info: 1byte info length + n bytes info.

Success:

Frame	0x85	0x0B	Handles	Checksum

Failure:

Frame	0x 7A	0x0B	Error Code	Checksum

Error Code: Returned status code from tag

Example:

Send: 00 12 00 85 0B 0C 00 00 00 00 00 00 40 00 FA B1 F5 23 4D

Return: 00 06 01 85 8A 32 3A

Send: 00 0B 00 85 02 00 00 00 04 8A 32 30

Return: 00 0E 01 85 33 33 33 33 00 00 00 00 8A 32 32

------ End of file ------