



East Wind Technologies, Inc.

---

# EWTJ680E Module

---

## **General Technical Manual**

(Revision 1.01)

**East Wind Technologies, Inc.**

**January 30, 2024**



## · Revise history

No.	Date	Description	Rev.	Page
1	2023/11/08	Original version	1.00	
2	2023/11/21	Add supported card types	1.01	7

# 1 Introduction

## 1.1 Features

- NFC Function
- Support ISO 14443A/B, ISO 15693, FeliCa Lite-S (RCS966) and more ISO STD
- Support ISO-DEP(ISO 14443-4)
- Frequency: 13.56 MHz ( $\pm 20$ PPM)
- Input voltage: 4V – 6V
- Average current: 150mA( $\pm 10$ mA)
- Detect distance: 5.5cm( $\pm 0.5$ cm, ISO 14443A card (MF1 IC S50))
- Communication interface: RS-232
- Baud Rate: 19,200bps
- Main board size: 70mm x 40mm x 5.8mm
- Storage temperature:  $-40^{\circ}\text{C} \sim 125^{\circ}\text{C}$
- Operating temperature:  $-40^{\circ}\text{C} \sim 105^{\circ}\text{C}$
- Operating humidity : 90%
- With LED light
- With metal cover



## **Warning:**

### **Federal Communication Commission Interference Statement**

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) This device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

FCC Caution: Any changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate this equipment.

### **Radiation Exposure Statement:**

The product comply with the FCC portable RF exposure limit set forth for an uncontrolled environment and are safe for intended operation as described in this manual. The further RF exposure reduction can be achieved if the product can be kept as far as possible from the user body or set the device to lower output power if such function is available.



---

**KDB 996369 D03 OEM Manual rule sections:**

**2.2 List of applicable FCC rules**

This module has been tested for compliance to FCC Part 15.225, 15.215

**2.3 Summarize the specific operational use conditions**

Any other usage conditions such as co-location with other transmitter(s) will need a separate reassessment through a class II permissive change application or new certification.

**2.4 Limited module procedures**

Not applicable.

**2.5 Trace antenna designs**

Not applicable.

**2.6 RF exposure considerations**

The product comply with the FCC portable RF exposure limit set forth for an uncontrolled environment and are safe for intended operation as described in this manual. The further RF exposure reduction can be achieved if the product can be kept as far as possible from the user body or set the device to lower output power if such function is available.

**2.7 Antennas**

The following antennas have been certified for use with this module; antennas of the same type with equal or lower gain may also be used with this module.

Antenna Type	Integrated
Antenna connector	N/A

**2.8 Label and compliance information**

The final end product must be labeled in a visible area with the following: “Contains FCC ID: 2AGMLEWTJ680E”. The grantee's FCC ID can be used only when all FCC compliance requirements are met.

The OEM integrator has to be aware not to provide information to the end user regarding how to install or remove this RF module in the user’s manual of the end product which integrates this module.

The end product user manual shall include all required regulatory information/warning as shown in this manual.

**2.9 Information on test modes and additional testing requirements**

Any co-located or simultaneous transmission with other transmitter(s) will require a separate class II permissive change re-evaluation or new certification.

**2.10 Additional testing, Part 15 Subpart B disclaimer**

This transmitter module is tested as a subsystem and its certification does not cover the FCC Part 15 Subpart B (unintentional radiator) rule requirement applicable to the final host. The



final host will still need to be reassessed for compliance to this portion of rule requirements if applicable.

OEM/Host manufacturers are ultimately responsible for the compliance of the Host and Module. The final product must be reassessed against all the essential requirements of the FCC rule such as FCC Part 15 Subpart B before it can be placed on the US market. This includes reassessing the transmitter module for compliance with the Radio and EMF essential requirements of the FCC rules. This module must not be incorporated into any other device or system without retesting for compliance as multi-radio and combined equipment.

As long as all conditions above are met, further transmitter test will not be required. However, the OEM integrator is still responsible for testing their end-product for any additional compliance requirements required with this module installed.

### **2.11 Note EMI Considerations**

Please follow the guidance provided for host manufacturers in KDB publications 996369 D02 and D04.

### **2.12 How to make changes**

Only Grantees are permitted to make permissive changes. Please contact us should the host integrator expect the module to be used differently than as granted:

Ben Chen / Vice President  
TEL: +886-2-7712-8686#17  
FAX: +886-2-7712-8688  
Email: [benchen@ewt.com.tw](mailto:benchen@ewt.com.tw)

**IMPORTANT NOTE:** In the event that these conditions cannot be met (for example certain laptop configurations or co-location with another transmitter), then the FCC authorization is no longer considered valid and the FCC ID cannot be used on the final product. In these circumstances, the OEM integrator will be responsible for re-evaluating the end product (including the transmitter) and obtaining a separate FCC authorization.



---

**Industry Canada statement:**

This device complies with ISED's licence-exempt RSSs. Operation is subject to the following two conditions: (1) This device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

Le présent appareil est conforme aux CNR d'ISED applicables aux appareils radio exempts de licence. L'exploitation est autorisée aux deux conditions suivantes : (1) le dispositif ne doit pas produire de brouillage préjudiciable, et (2) ce dispositif doit accepter tout brouillage reçu, y compris un brouillage susceptible de provoquer un fonctionnement indésirable.

**Radiation Exposure Statement:**

The product comply with the Canada portable RF exposure limit set forth for an uncontrolled environment and are safe for intended operation as described in this manual. The further RF exposure reduction can be achieved if the product can be kept as far as possible from the user body or set the device to lower output power if such function is available.

**Déclaration d'exposition aux radiations:**

Le produit est conforme aux limites d'exposition pour les appareils portables RF pour les Etats-Unis et le Canada établies pour un environnement non contrôlé.

Le produit est sûr pour un fonctionnement tel que décrit dans ce manuel. La réduction aux expositions RF peut être augmentée si l'appareil peut être conservé aussi loin que possible du corps de l'utilisateur ou que le dispositif est réglé sur la puissance de sortie la plus faible si une telle fonction est disponible.

**This device is intended only for OEM integrators under the following conditions: (For module device use)**

1) The transmitter module may not be co-located with any other transmitter or antenna.

As long as 1 condition above is met, further transmitter test will not be required. However, the OEM integrator is still responsible for testing their end-product for any additional compliance requirements required with this module installed.

**Cet appareil est conçu uniquement pour les intégrateurs OEM dans les conditions suivantes: (Pour utilisation de dispositif module)**

1) Le module émetteur peut ne pas être coïmplanté avec un autre émetteur ou antenne.

Tant que les 1 condition ci-dessus sont remplies, des essais supplémentaires sur l'émetteur ne seront pas nécessaires. Toutefois, l'intégrateur OEM est toujours responsable des essais sur son produit final pour toutes exigences de conformité supplémentaires requis pour ce module installé.

**IMPORTANT NOTE:**

In the event that these conditions can not be met (for example certain laptop configurations or co-location with another transmitter), then the Canada authorization is no longer considered valid and the IC ID can not be used on the final product. In these circumstances, the OEM integrator will be responsible for re-evaluating the end product (including the transmitter) and obtaining a separate Canada authorization.

**NOTE IMPORTANTE:**

Dans le cas où ces conditions ne peuvent être satisfaites (par exemple pour certaines configurations d'ordinateur portable ou de certaines co-localisation avec un autre émetteur), l'autorisation du Canada n'est plus considéré comme valide et l'ID IC ne peut pas être utilisé sur le produit final. Dans ces circonstances, l'intégrateur OEM sera chargé de réévaluer le produit final (y compris l'émetteur) et l'obtention d'une autorisation distincte au Canada.

**End Product Labeling**

The product can be kept as far as possible from the user body or set the device to lower output power if such function is available. The final end product must be labeled in a visible area with the following: "Contains IC:24646-EWTJ680E".

**Plaque signalétique du produit final**

L'appareil peut être conservé aussi loin que possible du corps de l'utilisateur ou que le dispositif est réglé sur la puissance de sortie la plus faible si une telle fonction est disponible. Le produit final doit être étiqueté dans un endroit visible avec l'inscription suivante: "Contient des IC: 24646-EWTJ680E".

**Manual Information To the End User**

The OEM integrator has to be aware not to provide information to the end user regarding how to install or remove this RF module in the user's manual of the end product which integrates this module.


The end user manual shall include all required regulatory information/warning as show in this manual.

**Manuel d'information à l'utilisateur final**

L'intégrateur OEM doit être conscient de ne pas fournir des informations à l'utilisateur final quant à la façon d'installer ou de supprimer ce module RF dans le manuel de l'utilisateur du produit final qui intègre ce module. Le manuel de l'utilisateur final doit inclure toutes les informations réglementaires requises et avertissements comme indiqué dans ce manuel.

取得審驗證明之低功率射頻器材，非經核准，公司、商號或使用者均不得擅自變更頻率、加大功率或變更原設計之特性及功能。

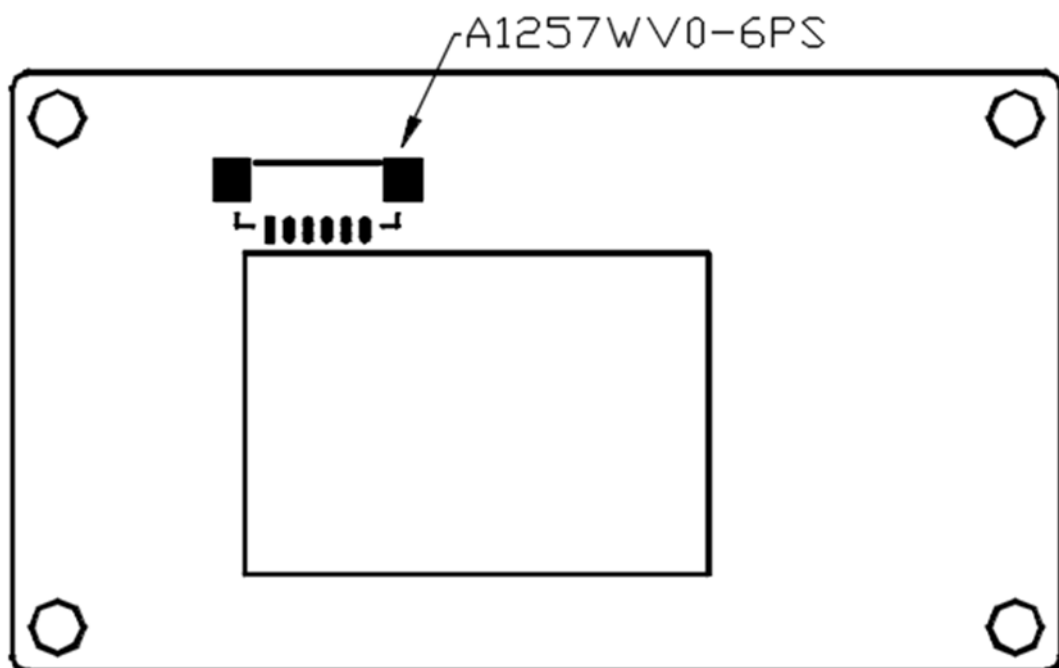
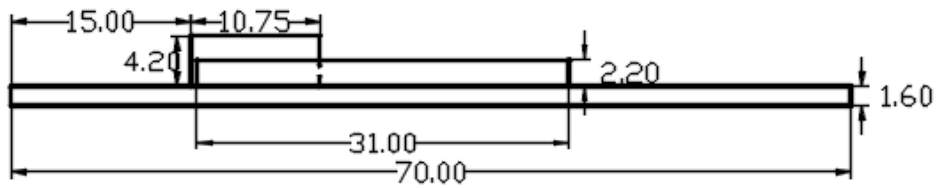
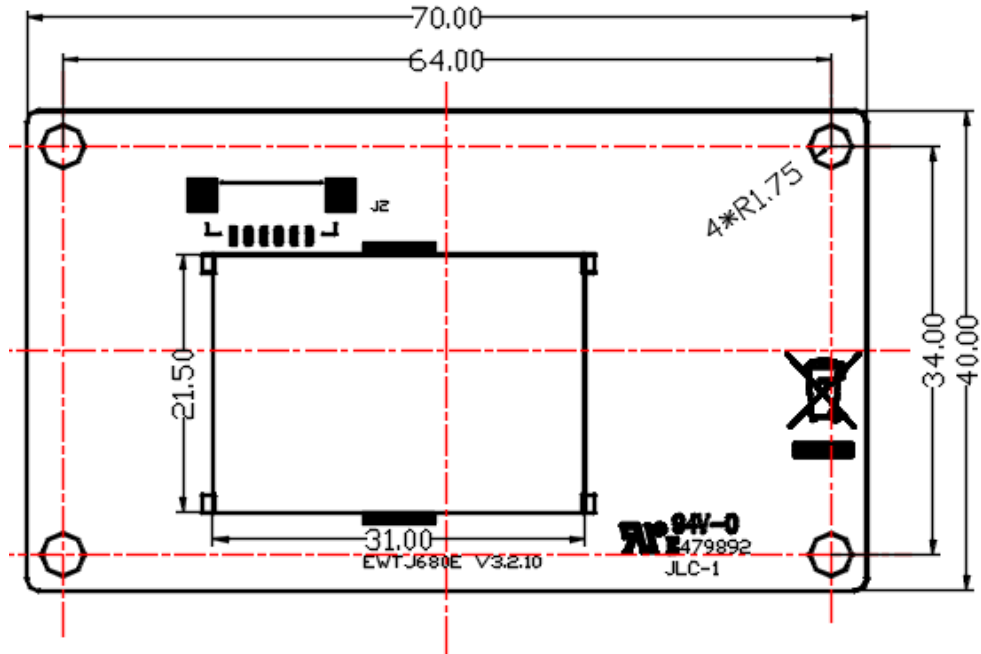
低功率射頻器材之使用不得影響飛航安全及干擾合法通信；經發現有干擾現象時，應立即停用，並改善至無干擾時方得繼續使用。前述合法通信，指依電信管理法規定作業之無線電通信。低功率射頻器材須忍受合法通信或工業、科學及醫療用電波輻射性電機設備之干擾。

系統廠商應於平台上標示「本產品內含射頻模組：XXXyyyLPDzzzz-x」字樣。



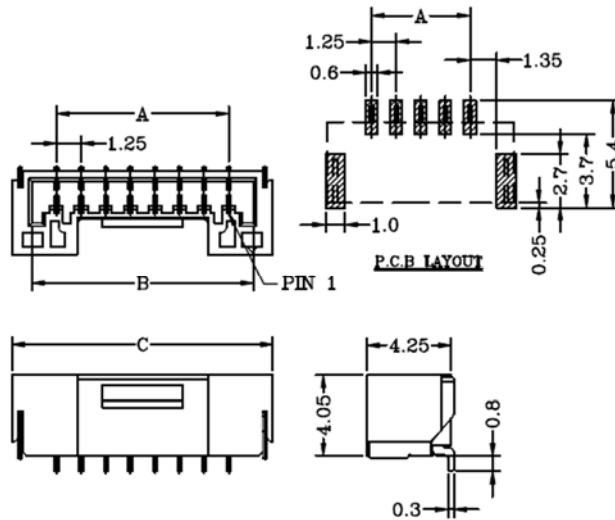
## 1.2 Dimension

Main board:



Connector 排針型號:A257WV0-6PS

- \* Material : Insulator: Nylon 6T UL 94V-0 (Milky Yellow)  
Contact Post: 0.20mm<sup>t</sup> Phosphor Bronze, Matte Tin plated  
Solder Tab: 0.20mm<sup>t</sup> Brass, Matte Tin Plated
- \* Mates with JWT A1257 series socket
- \* JWT P/N : A1257WR0-NPS-\*\*



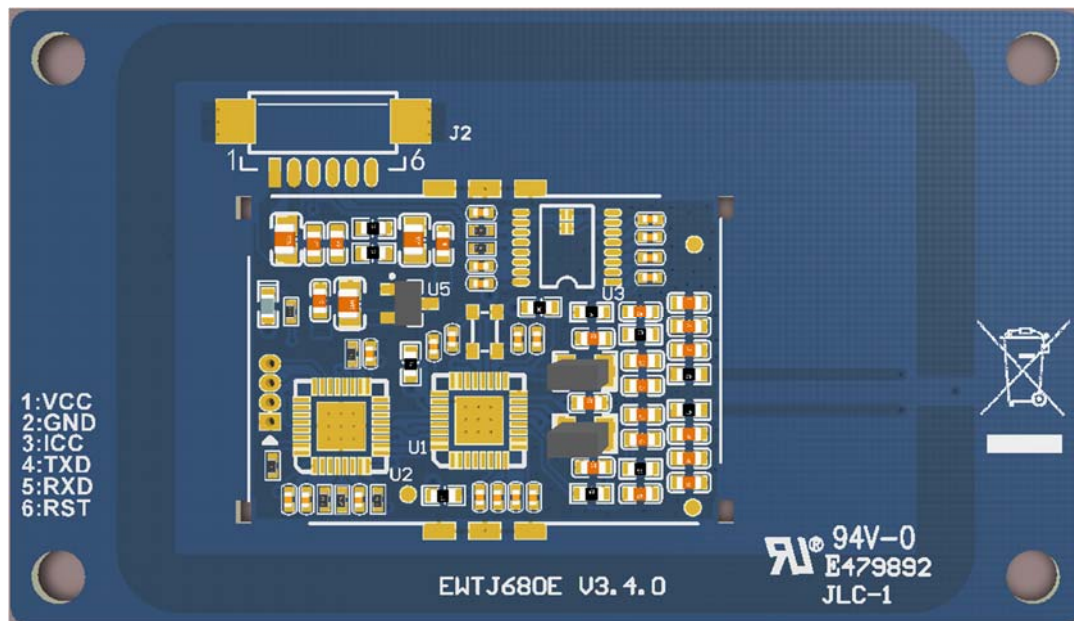
Dimensional & Ordering Information:

Circuits	Dimensions			Qty/Reel
	A	B	C	
2	1.25	3.75	5.75	1000
3	2.50	5.00	7.00	1000
4	3.75	6.25	8.25	1000
5	5.00	7.50	9.50	1000
6	6.25	8.75	10.75	1000
7	7.50	10.00	12.00	1000
8	8.75	11.25	13.25	1000
9	10.00	12.50	14.50	1000
10	11.25	13.75	15.75	1000
11	12.50	15.00	17.00	1000
12	13.75	16.25	18.25	1000
13	15.00	17.50	19.50	1000
14	16.25	18.75	20.75	1000
15	17.50	20.00	22.00	1000

### 1.3 Pin Configurations

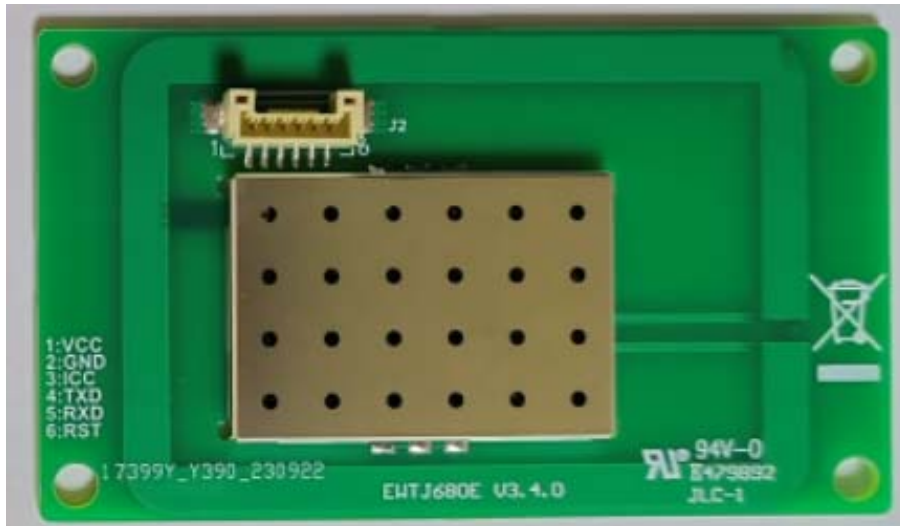
PIN	Function	Type	Description
1	VCC	Power	VCC
2	GND	Power	GND
3	ICC	Output	Indication of whether there is a card in the antenna area: 0: Card present; 1: No card
4	TXD	Input/ Output	RS232C TXD / UART TXD
5	RXD	Input	RS232C RXD / UART RXD
6	RST	Input	hardware reset (active low, time:150us(Typ))

視圖(View)

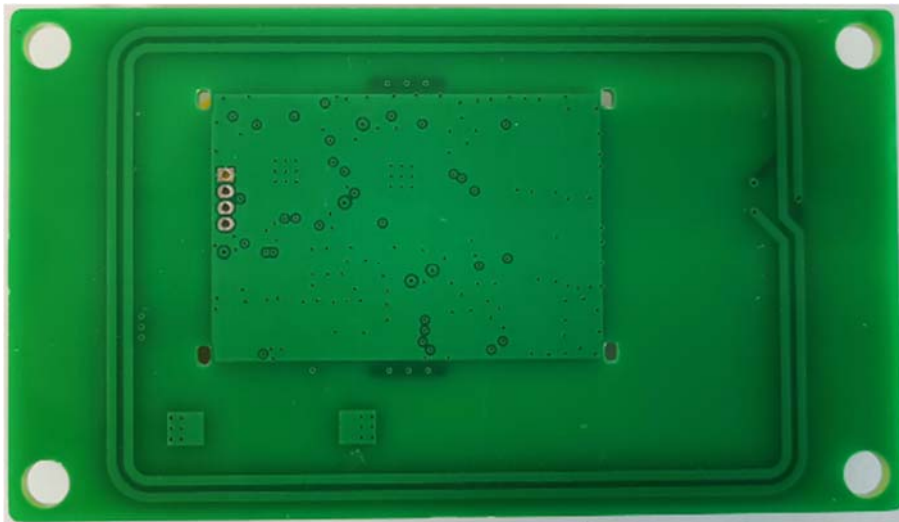


## 1.4 Physical photos

**Module front:**



**Module back:**



## 1.5 Supported card types

Card name	EWTJ680E
MIFARE 1K	●
MIFARE 4K	●
MIFARE Ultra Light	●
MIFARE Ultra Light C	●
MIFARE Mini	●
MIFARE DESfire	●
MIFARE Plus	●
T=CL TYPE A	●
SR176	●
SRI512	●
SRI1K	●
SRI2K	●
SRI4K	●
SRIX4K	●
T=CL TYPE B	●
Sony Felica RCS962	●
Sony Felica RCS965	●
Sony Felica RCS966	●
Suica	●
ICOCA	●
I.CODE SLI	●
I.CODE SLI-S	●
On Chip Data FLASH	512 bytes

## 1.6 Basic Mode

EWTJ680E is a slave device; the "ask&answer" is the basic working mode. That means EWTJ680E received a command from master machine, and then to execute the command and answer to master machine. This is a command cycle. A new command will not 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, then to send the next command.

## 1.7 Automatically Detecting Card

EWTJ680E supports automatic detecting cards that are based on ISO14443A and ISO15693. When the automatic detecting cardsfunction is open, then EWTJ680E continuously send searching card commands. Once the card is entering into the effective electrical field, the ICC pin will show low level. In this situation, you could directly to do operation to the card. If the automatic detecting function is close, when you want to do operation to the card, you need send searching card command firstly.

The default automatic detecting cardsfunction could be set via 0x1D command. This setting is saved in FLASH. The setting will be effect on next power on. While module is working, the automaticdetecting cardsfunction could be temporarily open or closed via 0x11 command. This setting will not be saved. So after power on again, it will be back to the default setting.

If the moduleoperate card mode (via 0x70 command)is set to ISO15693, then EWTJ680E just only auto detect ISO15693 cards. If the moduleoperate card mode (via 0x70 command) is set to ISO14443A, then EWTJ680E just only detectISO14443A cards.

Automatic detecting cards functionsupport ISO15693 cards.

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

ISO14443A T=CL card could be detected when the automatic detecting cards function is open. If you want to operate the card, you need 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.

User could set automatic detecting card and output the card UID. Under this mode, the card serial Data Packet Format

## 1.8 Communication Protocol

### 1.8.1 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, pleaserefers 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.

## 1.8.2 Data returned format

- Success:

Length	Command	Data	Checksum
--------	---------	------	----------

- Failure:

Length	Invert Command	Checksum
--------	----------------	----------

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

## 1.9 Data Returned Time

Slaves begin to execute the command once received host's commands. The executive time is normally less than 100ms, depend on the command type. Some command executive time maybe need longer. 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.

## 2 Communication Interface

### 2.1 RS232C interface

#### 2.1.1 Physical Interface

RS232C is the UART interface which belongs to RS232C-level. The communication between RS232C and PC is via TXD, RXD and GND pins. The host TXD pin connects to the device RXD pin, meanwhile the host RXD pin connects to the device 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: 1bit

Data bits: 8 bits

Stop bits: 1 bit

Parity check: None

Flow control: None

#### 2.1.2 Communication Process

Host send command to the slave, and the slave to execute once received the host's command, then to send the result to the host. This is a command cycle.



## 3 Application-layer Protocol

### 3.1 Overview

This chapter will introduce the communication protocol application level commands and data structures in detail, the application level protocol only introduce the commands and the data.

We illustrate each command in the following format:

Frame	Command	Data	Checksum
-------	---------	------	----------

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

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, "mode" is 0x00, so, the command is:

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.

## 3.2 System commands

### 3.2.1 Module reset to factory default

Function: Reset all configuration of the module to factory default setting. The new setting will effect after re-power on.

Host send:

Frame	0x0F	52 45 53 45 54	Checksum
-------	------	----------------	----------

Success:

Frame	0x0F	Checksum
-------	------	----------

Failure:

Frame	0xF0	Checksum
-------	------	----------

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

return: 0x02 0F 0D

### 3.2.2 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
-------	------	----------

### 3.2.3 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 equal to the set AFI. Settings will save in the module; it will not 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
-------	------	----------

### 3.2.4 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
-------	------	----------

### 3.2.5 Set the Default of Automatic Detecting Card

Function: Set the default state of automatic detecting card when power 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
-------	------	----------

### 3.2.6 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.

The default value is set to 0x00 (the strongest).

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
-------	------	----------

### 3.3 ISO14443A/B CPU Card Commands

#### 3.3.1 ISO14443 TYPE ARequest

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 judge whether the card supports ISO14443-4 by SAK. If automatic detect card function was opened, then this command is only to read the result of automatic detect 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
-------	------	----------

#### 3.3.2 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 verifies 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 detect card function will be forced OFF.

Host sends:

Frame	0x30	Checksum
-------	------	----------

Success:

Frame	0x30	ATS	Checksum
-------	------	-----	----------

ATS: ATS, length depends on card.

Failure:

Frame	0xCF	Checksum
-------	------	----------

#### 3.3.3 ISO14443-4 TYPE B Request

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

Host sends:

Frame	0x60	Mode	AFI	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.

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 (3bytes), 1 byte answer to Attribute.

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

Failure:

Frame	0x9F	Checksum
-------	------	----------

### 3.3.4 Request Card according to EMV and PBOC

Function: Card Request according to EMV and PBOC standards and 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 requested 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, 1byte UIDLength; 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
-------	------	----------

### 3.3.5 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 operate ISO14443-4 card, the automatic detectfunction 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
-------	------	----------

### 3.3.6 ISO14443-4 TYPE B Card Halt

Function: To let the current ISO14443B card enters into halt status. Not all of the cards support this command, most 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
-------	------	----------

## 3.4 MIFARE 1K/4K/miniCard Commands

### 3.4.1 MIFARERequest

MIFAREseriescards request, please refer to [ISO14443 TYPE A Request](#).

### 3.4.2 MIFARE 1K/4KData Block Read

Function: Read MIFARE1K/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 informationplease refer 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
-------	------	----------

### 3.4.3 MIFARE 1K/4KMulti-Blocks Read

Function: Read multi data blocks in the same sector. The 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: 1byte, 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
-------	------	----------

### 3.4.4 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
-------	------	----------

### 3.4.5 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
-------	------	----------

### 3.4.6 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 reference 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
-------	------	----------

### 3.4.7 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 response 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
-------	------	----------

### 3.4.8 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
-------	------	----------

### 3.4.9 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
-------	------	----------

### 3.4.10 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
-------	------	----------

### 3.4.11 ISO14443A Card Halt

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

Host sends:



Success:	Frame	0x28	Checksum
----------	-------	------	----------

Failure:	Frame	0x28	Checksum
----------	-------	------	----------

	Frame	0xD7	Checksum
--	-------	------	----------

### 3.4.12 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 more safety. Because EEPROM be written times are limited, therefore, 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
-------	------	----------

### 3.4.13 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 6bytes 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.

## 3.5 MIFARE Ultralight/Ultralight C/Ultralight EV1Card Commands

### 3.5.1 MIFARE Ultralight/Ultralight C/Ultralight EV1 Request

For MIFAREUltraLight/UltraLight C card request, please refer to [ISO14443 TYPE A Request](#).

### 3.5.2 MIFARE Ultralight/Ultralight C/Ultralight EV1 Card Read

Function: Read the data from MIFAREUltraLight/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
-------	------	-------------	----------

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

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
-------	------	----------

### 3.5.3 MIFARE Ultralight/Ultralight C/Ultralight EV1 Card Write

Function: Write data to MIFAREUltraLight/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
-------	------	----------

### 3.5.4 MIFAREUltraLightCKey Authentication

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

Host sends:

Frame	0x43	Key	Checksum
-------	------	-----	----------

Key: 16bytes UltraLight C key

Success:

Frame	0x43	Checksum
-------	------	----------

Failure:

Frame	0xBC	Checksum
-------	------	----------

### 3.5.5 MIFAREUltraLightCEk (RndB)Read

Function: To read encrypted RndB this is generated by Ultralight C card. Command 0x44 and 0x45 are the separation commands to authenticate Ultralight C. Because the microcontroller 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
-------	------	----------

### 3.5.6 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.

### 3.5.7 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
-------	------	----------

As follows:

### 3.5.8 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 \*4bytes card data.

Failure:

Frame	0xB8	Checksum
-------	------	----------

### 3.5.9 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: 3bytes Ultralight EV1 counter data.



Failure:

Frame	0xB7	Checksum
-------	------	----------

### 3.5.10 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: 3bytes, Ultralight EV1 INCR\_CNT data.

Success:

Frame	0x49	Checksum
-------	------	----------

Failure:

Frame	0xB6	Checksum
-------	------	----------

### 3.5.11 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
-------	------	----------

### 3.5.12 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: 32bytes signature data.

Failure:

Frame	0xB4	Checksum
-------	------	----------

### 3.5.13 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
-------	------	----------

### 3.5.14 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
-------	------	----------

## 3.6 MIFAREPlusCard Commands

EWTJ680E reader module support MIFAREPlus card operation. NXP MIFAREPlus cards are used to instead MIFARE 1 card. We provide application commands 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 MIFAREPlus R & D works. But for advanced user, also could use APDU to implement the card.

### 3.6.1 MIFAREPlusPrepare Commands

#### 3.6.1.1 MIFAREPlus Request

For MIFAREPlus card request, please refer to [ISO14443 TYPE A Request](#).

#### 3.6.1.2 MIFAREPlusRATS

Please refer to [ISO14443-4 TYPE-A card reset \(RATS\)](#).

#### 3.6.1.3 MIFAREPlus Request and RATS

Please refer to [Card Request according to EMV and PBOC](#).

### 3.6.2 MIFAREPlus Initialization Commands

#### 3.6.2.1 MIFAREPlus Write Perso

Function: Initialization of the AES key and all other blocks. About these blocks address and the default value please reference the MIFAREPlus datasheet or contact us.

Host sends:

Frame	0x33	Address	Data	Checksum
-------	------	---------	------	----------

Address: 2 bytes block address, MSB first.

Data: 16bytes.

Success:

Frame	0x33	Status	Checksum
-------	------	--------	----------

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

Please reference: [MIFAREPlus Returned Status Code](#).

Failure:

Frame	0xCC	Checksum
-------	------	----------

#### 3.6.2.2 MIFAREPlus Commit Perso

Function: Level 0 command, to switch Level 0 to Level 1 or Level 3. 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 MIFAREPlus Write Persocommand to write all AES key and the initial value of all the blocks, then make the changed data effect.

Host sends:

Frame	0x34	Checksum
-------	------	----------

Success:

Frame	0x34	Status	Checksum
-------	------	--------	----------

Status: statuscode returned from the card.

Please reference: [MIFARE Plus Returned Status Code](#).

Failure:

Frame	0xCB	Checksum
-------	------	----------

### 3.6.2.3 MIFAREPlus 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, levelto be switched, 2: Level 2; 3: Level 3.

Key: 16bytes.

Success:

Frame	0x35	Status	Checksum
-------	------	--------	----------

Status: status code returned from the card.

Please reference: [MIFARE Plus Returned Status Code](#).

Failure:

Frame	0xCA	Checksum
-------	------	----------

## 3.6.3 MIFAREPlus Application Layer Commands

### 3.6.3.1 MIFAREPlusData 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: 16bytes.

Success:

Frame	0x36	Status	Checksum
-------	------	--------	----------

Status: status code returned from the card.

Please reference: [MIFARE Plus Returned Status Code](#).

Failure:

Frame	0xC9	Checksum
-------	------	----------

**3.6.3.2 MIFAREPlusData 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\*16bytes

Failure:

Frame	0xC8	Checksum
-------	------	----------

**3.6.3.3 MIFAREPlusData 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 written

Data: block\*16bytes data to be written

Success:

Frame	0x38	Status	Checksum
-------	------	--------	----------

Status: status code returned from the card.

Please reference: [MIFARE Plus Returned Status Code](#).

Failure:

Frame	0xC7	Checksum
-------	------	----------

**3.6.3.4 MIFAREPlus Purse Create**

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

Host sends:

Frame	0x39	Block	Value	Checksum
-------	------	-------	-------	----------

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

Value: 4bytes(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
-------	------	----------

### 3.6.3.5 MIFAREPlusPurse 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
-------	------	----------

### 3.6.3.6 MIFAREPlusPurse Increment

Function: Level 3 command, purse increment of MIFAREPlus.

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
-------	------	----------

### 3.6.3.7 MIFAREPlusPurse Decrement

Function: Level 3 command, purse decrement of MIFAREPlus.

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
-------	------	----------

### 3.6.3.8 MIFAREPlusPurse Copy

Function: Level 3 command, copy the MIFAREPlus 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
-------	------	----------

### 3.6.3.9 MIFAREPlusFirst Authenticate

Function: Level 1/3 Command. In Level 3, this command is used for authentication 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
-------	------	----------

### 3.6.3.10 MIFAREPlusFollowing Authenticate

Function: Level 1/3 Command. In Level 3, this command is used for authentication for none data block before reading and writing. It is used for authentication 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
-------	------	----------

### 3.6.4 MIFAREPlusReturned Status Code

There is a status code after MIFAREPluscardresponse;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	MACError.
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.



## 3.7 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 EWTJ680E 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.

### 3.7.1 DESFirePrepare Commands

#### 3.7.1.1 DESFire Request

DESFire card request, Please reference [ISO14443 TYPE A Request](#).

#### 3.7.1.2 DESFire RATS

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

#### 3.7.1.3 DESFire Request and RATS

This command support DESFire Request and RATS. Please reference: [Card Request according to EMV and PBOC](#).

#### 3.7.1.4 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 sent 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 helpful for users, if need 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 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 senssionkey 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
-------	------	----------

### 3.7.1.5 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
-------	------	----------

### 3.7.1.6 DESFire Authenticate second step get ekNo (RndA')

Function: Random number RndA generated by the host. Host encrypts the assembled "RndA and RndB", and then send to card and get ekNo (RndA') from card, decrypt it to get RndA', reassemble 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.

SenssionKey: 16 bytes, Combination of RndA and RndB:

SenssionKey = RndA[0..3] + RndB[0..3] + RndA[4..7] + RndB[4..7]

The senssionkey 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
-------	------	----------

### 3.7.1.7 DESFire Select Application

Function: Select the specified card application. The following operation will effect 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
-------	------	----------

## 3.7.2 DESFire Initialization Commands

### 3.7.2.1 DESFireFormat 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
-------	------	----------

### 3.7.2.2 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
-------	------	----------

### 3.7.2.3 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
-------	------	----------

### 3.7.2.4 DESFire Get Key Settings

Function: Get the master key/application master key configuration 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: 1 byte, Max. Key numbers of current application

Failure:

Frame	0x6D	Checksum
-------	------	----------

### 3.7.2.5 DESFire Change Key

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

Host sends:

Frame	0x93	KeyID	ekKey	Checksum
-------	------	-------	-------	----------

KeyID: 1 byte.

ekKey: 24 bytes (Refer to the datasheet for encryption calculations, or use the tools we provide and references 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
-------	------	----------

### 3.7.2.6 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
-------	------	----------

### 3.7.2.7 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
-------	------	----------

### 3.7.2.8 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: 28bytes card manufacturer and production information.

Failure:

Frame	0x65	Checksum
-------	------	----------

### 3.7.2.9 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
-------	------	----------

### 3.7.2.10 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 Identifier length is 1 byte, the total length is file number\* 1 bytes.

Failure:

Frame	0x64	Checksum
-------	------	----------

### 3.7.2.11 DESFire Get FileSettings

Function: Getspecified 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 bytelimited credit enable.
- Record file: 1 byte file type+ 1 byte comm. setting + 2 bytes access right+3 bytes record size+3 bytes Max record+ 3bytes current number of records.

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

Failure:

Frame	0x63	Checksum
-------	------	----------

### 3.7.2.12 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: 2bytes (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
-------	------	----------

### 3.7.2.13 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: 1byte,0: Plaintext;1: MAC code checksum; 3: DES/3DES encryption.

AccessRight: 2bytes (LSB in first).

Size: 3bytes (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
-------	------	----------

### 3.7.2.14 DESFire Create Backup Data File

Function: Create Data File in current application, support backupmechanism (mirror).Then the fileactualsize 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: 2bytes (LSB in first).

Size: 3bytes (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
-------	------	----------

**3.7.2.15 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: 1 byte, 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
-------	------	----------

**3.7.2.16 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	MaxRecords	Checksum
-------	------	-----	------------	--------------	-------------	------------	----------

FID: 1 byte.

Comm.Sett: 1 byte, 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
-------	------	----------

**3.7.2.17 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: 2bytes (LSB in first).

Record Size: 3bytes (LSB in first), bytes of single record.

Max Records: 3bytes (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
-------	------	----------

### 3.7.2.18 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
-------	------	----------

## 3.7.3 DESFire Application Layer Commands

### 3.7.3.1 DESFire Read Data

Function: Reads specified DataFile(Standard Data File or Backup Data File) in current application.

Host sends:

Frame	0xA4	FID	Offset	Length	Checksum
-------	------	-----	--------	--------	----------

FID: 1 byte.

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

Length: 3bytes (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
-------	------	----------

### 3.7.3.2 DESFire Write Data

Function: Write specified Data File (Standard Data File or Backup Fata File) in current application. For BackupData File, Commit isneeded to take effectafter write, refer to:[DESFire Commit Transaction](#) please.

Host sends:

Frame	0xA5	FID	Offset	Length	Data	Checksum
-------	------	-----	--------	--------	------	----------

FID: 1 byte.

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

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

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
-------	------	----------

### 3.7.3.3 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 encryption.

Plaintext: 4bytesvalue (LSB in first).

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

Failure:

Frame	0x59	Checksum
-------	------	----------

### 3.7.3.4 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 encryption.

Plaintext: 4bytesvalue (LSB in first).



Encryption: 8 bytes encrypted data, After decryption: 4bytes 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
-------	------	----------

### 3.7.3.5 DESFire Debit

Function: Decrease 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	0xA8	FID	Data	Checksum
-------	------	-----	------	----------

FID: 1 byte.

Data: There are two lengths, depending on whether encryption.

Plaintext: 4bytes value (LSB in first).

Encryption: 8 bytes encrypted data, After decryption: 4bytes 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
-------	------	----------

### 3.7.3.6 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 encryption.

Plaintext: 4bytes value (LSB in first).

Encryption: 8 bytes encrypted data, After decryption: 4bytes 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
-------	------	----------

### 3.7.3.7 DESFire Write Record

Function: Write data to specified Data File in current application. The data file could be Linear Record or CyclicRecord 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: 3bytes (LSB in first), offset in the record.

Length: 3bytes (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
-------	------	----------

### 3.7.3.8 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: 3bytes (LSB in first), offset of the record.

Length: 3bytes (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
-------	------	----------

### 3.7.3.9 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
-------	------	----------

### 3.7.3.10 DESFire Commit Transaction

Function: Submit all WRITE operation of Backup Data file, Value file and Record file incurrent 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
-------	------	----------

### 3.7.3.11 DESFire Abort Transaction

Function:Abortall WRITE operation of Backup Data file, Value file and Record file incurrent 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
-------	------	----------

### 3.7.4 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 match data 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	APPLICATION_NOT_FOUND	Requested AID not present on PICC
0xA1	APPL_INTEGRITY_ERROR	Unrecoverable error within application, application will be disabled *
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 CreateApplication 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.

## 3.8 SR176 Card Commands

### 3.8.1 SRSerial 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. 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
-------	------	----------

### 3.8.2 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
-------	------	----------

### 3.8.3 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
-------	------	----------

### 3.8.4 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
-------	------	----------

### 3.8.5 SR176 Card Write

Function: Write into the data block of SR176 card. After wrote, module will read the data to compare. If not equal, then 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
-------	------	----------

### 3.8.6 SR176 BlockLock

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

Host sends:

Frame	0x6A	LOCK_REG	Checksum
-------	------	----------	----------

LOCK\_REG: 1 byte, the lock register values to be written.

Success:

Frame	0x6A	Checksum
-------	------	----------

Failure:

Frame	0x95	Checksum
-------	------	----------



## 3.9 SRI512/1K/2K/4KCard Commands

### 3.9.1 SRI Serial Cards 1 Slot Initiate Card

Please reference: [SR serial cards 1 slot initiate card.](#)

### 3.9.2 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
-------	------	----------

### 3.9.3 SR Serial Cards Select

Please reference: [SR serial cards select.](#)

### 3.9.4 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
-------	------	----------

### 3.9.5 SR Serial Cards Completion

Please reference: [SR serial cards completion.](#)

### 3.9.6 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
-------	------	----------

### 3.9.7 SRI Serial Cards Write

Function: Write data block of SRI serial card. After write, module will read the data to compare. If not equal, then 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 write to the card.

Success:

Frame	0x6C	Checksum
-------	------	----------

Failure:

Frame	0x93	Checksum
-------	------	----------

### 3.9.8 SRI Serial Cards Block Lock

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

Host sends:

Frame	0x6D	LOCK_REG	Checksum
-------	------	----------	----------

LOCK\_REG: 1 byte, the lock register values to be written.

Success:

Frame	0x6D	Checksum
-------	------	----------

Failure:

Frame	0x92	Checksum
-------	------	----------

### 3.9.9 SRISerial 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
-------	------	----------

### 3.9.10 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
-------	------	----------

## 3.10 ISO15693 Operation Commands

### 3.10.1 ISO15693 Inventory

Function: Find a card in RF effective field. If success, to set the tag as CURRENT TAG.

If automatic detecting card function was turned on, then this command is to take the result of automatic detecting card, not 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
-------	------	----------

### 3.10.2 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 could also read and write operations.

Host sends:

Frame	0x5D	Checksum
-------	------	----------

Success:

Frame	0x5D	Checksum
-------	------	----------

Failure:

Frame	0xA2	Checksum
-------	------	----------

### 3.10.3 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
-------	------	----------

### 3.10.4 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
-------	------	----------

### 3.10.5 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 (dependon the cards).

Failure:

Frame	0xAB	Checksum
-------	------	----------

### 3.10.6 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
-------	------	----------

### 3.10.7 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
-------	------	----------

### 3.10.8 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
-------	------	----------

### 3.10.9 ISO15693 Lock AFI

Function: Lock AFI of CURRENT TAG.

Host sends:

Frame	0x58	Checksum
-------	------	----------

Success:

Frame	0x58	Checksum
-------	------	----------

Failure:

Frame	0xA7	Checksum
-------	------	----------

### 3.10.10 ISO15693 Write DSFID

Function: Write DSFID of CURRENT TAG.

Host sends:



Frame	0x59	DSFID	Checksum
-------	------	-------	----------

DSFID: 1 byte, DSFIDvalue to be written.

Success:

Frame	0x59	Checksum
-------	------	----------

Failure:

Frame	0xA6	Checksum
-------	------	----------

### 3.10.11 ISO15693 Lock DSFID

Function: Lock DSFID of CURRENT TAG.

Host sends:

Frame	0x5A	Checksum
-------	------	----------

Success:

Frame	0x5A	Checksum
-------	------	----------

Failure:

Frame	0xA5	Checksum
-------	------	----------

### 3.10.12 ISO15693 Get Blocks Security

Function: Get blocks securityinformation 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 gottensecurity.

Success:

Frame	0x5B	Data	Checksum
-------	------	------	----------

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

Failure:

Frame	0xA4	Checksum
-------	------	----------

## 3.11 Felica Card Commands

### 3.11.1 Send APDU to Felica card

Function: Use this command to send / receive APDU to / from Felica card.

Host sends:

Frame	0x2F	APDU command	Checksum
-------	------	--------------	----------

APDU: Please refer to Felica user manual for detailed APDU command.

Success:

Frame	0x2F	Response	Checksum
-------	------	----------	----------

Response: Felica card answers. Response length depends on each command.

Failure:

Frame	0xD0	Checksum
-------	------	----------

### 3.11.2 Polling

Function: Use this command to acquire and identify a card. Acquisition of Manufacture ID (IDm) and Manufacture Parameter (PMm) is possible with this command.

Host sends:

Frame	0x2F	APDU command	Checksum
-------	------	--------------	----------

APDU: 0x06 00 FF FF 01 00

Success:

Frame	0x2F	Response	Checksum
-------	------	----------	----------

Response: Felica card answers.

Failure:

Frame	0xD0	Checksum
-------	------	----------

Example:

Send: 0x08 2F 06 00 FF FF 01 00 20

Response: 0x16 2F 14 01 01 2E 3D 23 BA 07 5C 45 00 F1 00 00 01 43 00 88 B4 36

This section describes Manufacture ID (IDm) and Manufacture Parameter (PMm). IDm and PMm can be acquired as the response data to the Polling command.

- Manufacture ID (IDm)

Using Manufacture ID (IDm), the Reader/Writer identifies a card to be the counterpart of communication. If more than one System exists on a card, IDm is set to each such System.

IDm consists of Manufacturer Code and Card Identification Number.

The upper 4 bits of the 1 Byte of data located at the top of Manufacturer Code indicate System Number in the card. System Number is automatically incremented by one in the order of separation of System. The upper 4 bits of IDm of System 0 is 0000b, so (for example), the upper 4 bits of IDm of System 1 becomes 0001b.



- Manufacture Parameter (PMm)

PMm consists of IC Code (2 Bytes) and the maximum response time parameters (6 Bytes).

- oIC Code (2 Bytes)

This is the information to identify a product. IC code consists of two components, i.e., ROM Type and IC Type.

- Maximum response time parameters (6 Bytes)

The timeout time is determined, based on the period of time necessary to process the commands.

Therefore (and because this period of time depends on the status of the card, and on the type and content of each command), the Reader/Writer shall dynamically determine the timeout time. In FeliCa technology, the maximum response time is determined by using the lower 6 Bytes of PMm. The card provides this parameter to the Reader/Writer, enabling the Reader/Writer to dynamically determine the timeout time.

### 3.11.3 Read Without Encryption

**Function:** Use this command to read Block Data.

Host sends:

Frame	0x2F	APDU command	Checksum
-------	------	--------------	----------

APDU: 0x10 06 01 2E 3D 23 BA 07 5C 45 01 09 00 01 80 00

**Success:**

Frame	0x2F	Response	Checksum
-------	------	----------	----------

Response: Felica card answers.

Failure:

Frame	0xD0	Checksum
-------	------	----------

Example:

Send: 0x12 2F 10 06 01 2E 3D 23 BA 07 5C 45 01 09 00 01 80 00 37

Response: 0x1F 2F 1D 0701 2E 3D 23 BA 07 5C 45 00 00 01 00 00 00 00 00 00 00 00 00 00 00  
00 00 00 36

### 3.11.4 Write Without Encryption

**Function:** Use this command to write Block Data.

Host sends:

Frame	0x2F	APDU command	Checksum
-------	------	--------------	----------

APDU: 0x20 08 01 2E 3D 23 BA 07 5C 45 01 09 00 01 80 00 11 22 33 44 55 66 77 88 11 22 33 44 55 66 77 88

Success:

Frame	0x2F	Response	Checksum
-------	------	----------	----------

Response: Felica card answers.



Failure:

Frame	0xD0	Checksum
-------	------	----------

Example:

Send: 0x22 2F 20 0801 2E 3D 23 BA 07 5C 45 01 09 00 01 80 0011 22 33 44 55 66 77 88 11 22 33 44 55 66 77 88

Response: 0x0E2F 0C0901 2E 3D 23 BA 07 5C 45 00 00 19