JMY505H User's Manual

(Revision 3.42)

Jinmuyu Electronics Co. LTD

2011/6/28

Please read this manual carefully before using. If any problem, please mail to: Jinmuyu@vip.sina.com

Contents

1 Product introduction

JMY505H is RFID read/write module with an UART and IIC serial port. JMY505H has various functions and supports multi ISO/IEC standards of contactless IC card. The designer combined some frequent used command of RF card and then user could operate the cards with full function by sending simple command to the module.

The impedance between RF module and antenna was tuned by impedance analyzer. And then the module has excellent performance and stability.

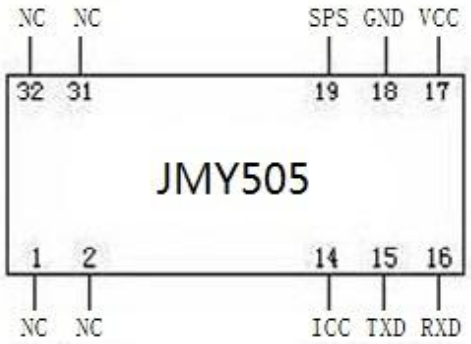
The module and antenna is split design. They are connecting by a 50ohm coaxial wire whose length is up to 1000cm in good situation. But the best length of the coaxial line is 60cm.

2 Characteristics

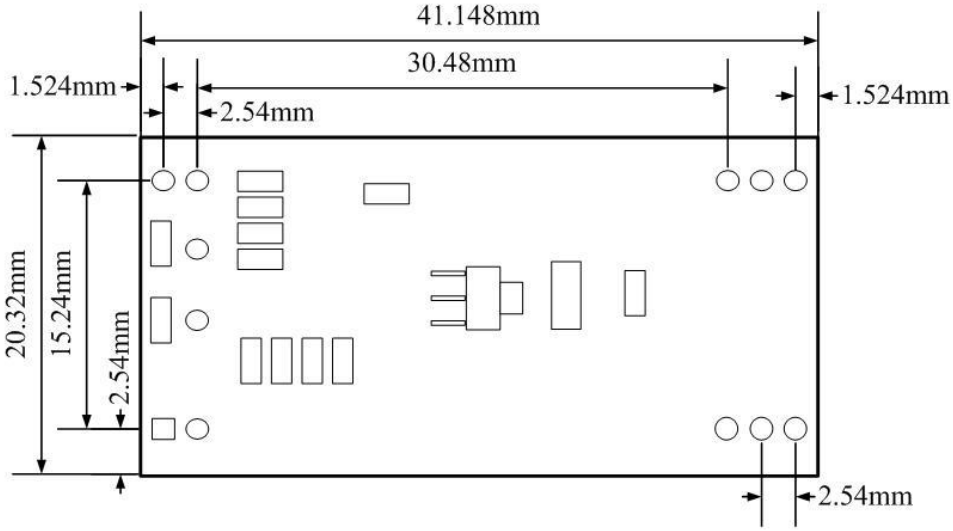
- PCD model: NXP CL RC632
- Working frequency: 13.56MHz
- Supported standard: ISO14443A, ISO14443B, ISO15693
- Antenna Connection: length 60cm, max 1000cm, customization is available
- Card supported: Mifare 1K/4K, FM11RF08, Ultra Light, DesFire, Mifare ProX, SR176, SRI512, SRI1K, SRI2K, SRI4K, SRIX4K, T=CL smart cards(both ISO14443A & ISO14443B) , TI TagIt, I.Code SLI, ST LRI and other tags according to ISO15693
- Anti collision ability: Full function anti collision; be able to process multi-cards; be able to set operate single card only
- Auto detecting card: Supported, default OFF
- EEPROM: 512 Bytes
- Power supply: DC 5V (±0.5V)
- Interface: IIC & UART (selected by SPS pin, recommend to use IIC)
- Communication rate: IIC: 400Kbps
  UART: 19.2Kbps/115.2Kbps
- Max. command length: 254 Bytes
- Interface level: 3.3V (TTL level; 5V tolerance)
- Power consumption: 70mA
- Operating distance: 100mm (depending on card and antenna design)
- Dimension: 21mm * 42mm
- Package: DIP32
- Weight: About 15g
- ISP: Supported
- Operating temperature: -25 to +85℃
- Storage temperature: -40 to +125 ℃
- RoHS: Compliant

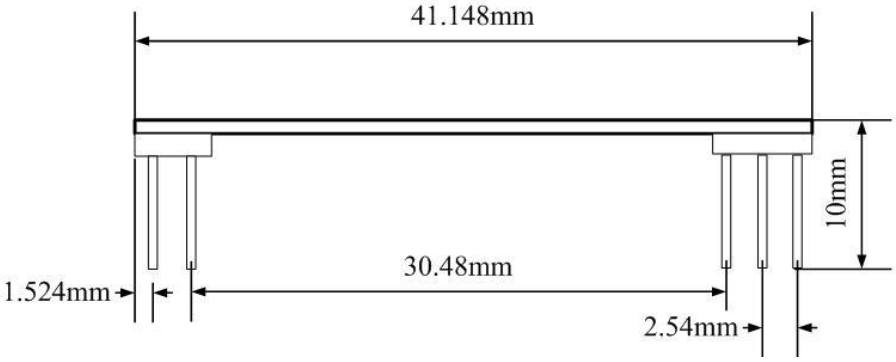
3 Physical parameter and pin outs

3.1 Photo



3.2 Dimension





3.3 Pin configurations and pin outs

PIN	Function	Type	Description
1	NC		NC
2	NC		NC
13	RE	Output	RE/DE 485 directional control output
14	ICC	Output	Card in/out indication 0: card in; 1: card out
15	TXD/SDA	Input/output	UART TXD/IIC SDA
16	RXD/SCL	Input	UART RXD/IIC SCL
17	VCC	Power	VCC
18	GND	Power	GND
19	SPS	Input	Serial port selector 0: IIC 1: UART
31	NC		NC
32	NC		NC

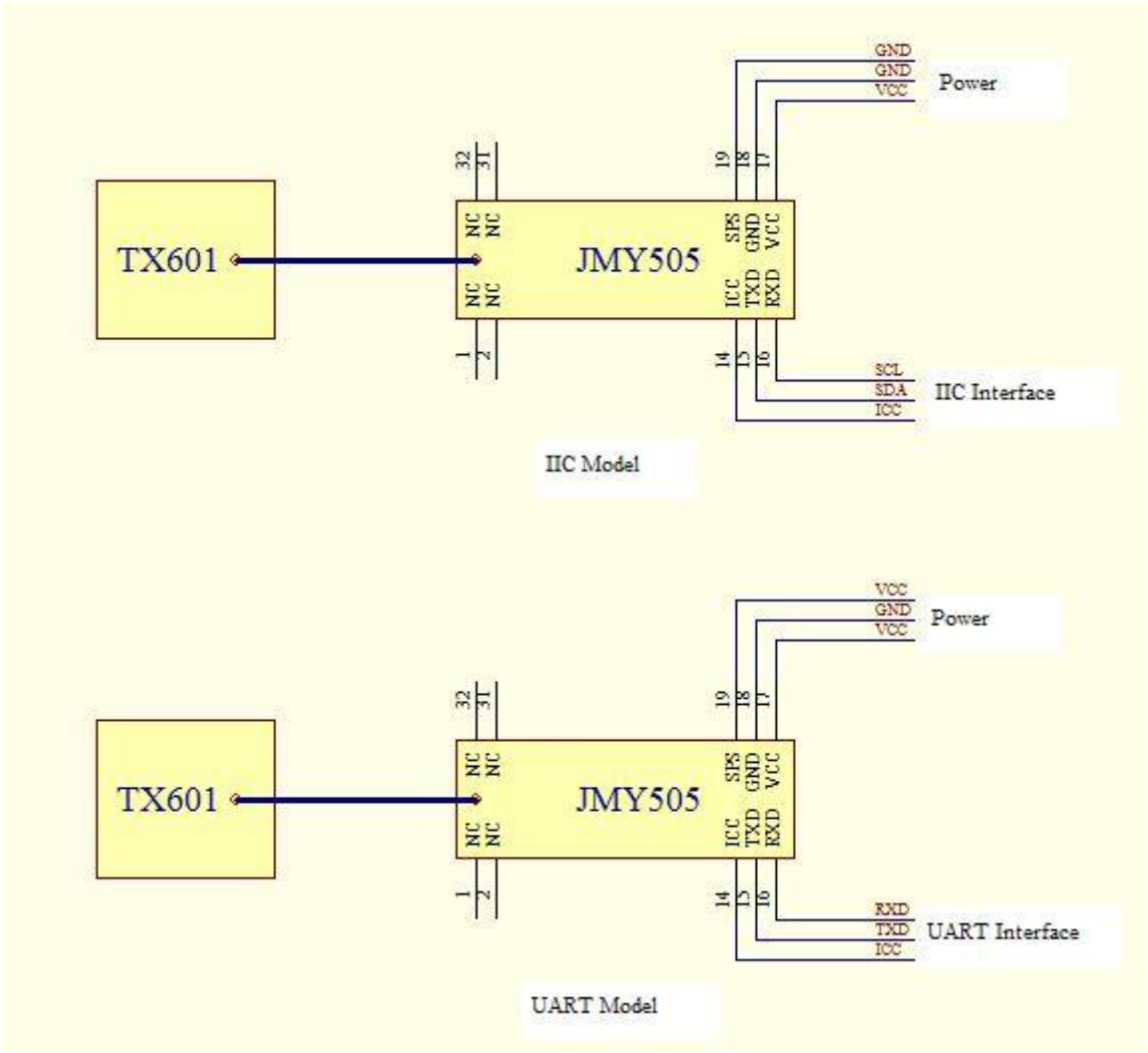
3.4 Antennas

Normally, as the size of TX600 series antenna may not meet the actual demands, the antenna needs to be customized, especially in some compact systems. The following information for customization is needed: 1. Dimension of the antenna PCB; 2. the position and direction of the antenna outlet and the connector; 3. the description of the antenna periphery. Jinmuyu will design the most proper antenna according to the user's exact requirements.

We provide many models of antenna. Please visit our website to get more information. There are some recommended models in the table:

Antenna model	Size of antenna	Card operating distance
TX600	70mm * 70mm	90mm
TX601	50mm * 50mm	70mm
TX602	30mm * 30mm	60mm

3.5 Connection schematics



3.6 JMY500 testing board

JMY500 testing board is a tool designed for testing of JMY50x series module, it could test the module completely with several steps. JMY500 operate the module via MCS51 MCU and it could to change the communication port (IIC or UART) of module. According to our source program (include IIC and UART), user is able to finish the program of application system.

JMY500 is also communicating with PC through RS232 port. Then user programs the testing software and completes the test to the module.

3.7 Model rule

3.7.1 Model format

1	2	3
JMY	505	X

1: company code; 2: product series code; 3: card operating type

3.7.2 Card operating type

M: PCD is RC500, support Mifare Class
A: PCD is RC500, support ISO14443A and Mifare Class
C: PCD is RC531, support ISO14443A, ISO14443B and Mifare Class
G: PCD is RC400, support ISO15693
H: PCD is RC632, support ISO15693, ISO14443A, ISO14443B and Mifare Class
D: PCD is RC500, support ISO14443A and Mifare Class with 511 bytes communication buffer
E: PCD is RC531, support ISO14443AB and Mifare Class with 511 bytes communication buffer

4 Communication protocols

4.1 Overview

The module has IIC and UART interfaces. We recommend using IIC interface whose communication rate is up to 400Kbps (normal rate is 100Kbps). But the baud rate of UART is 19.2Kbps and 115.2Kbps. We supply sample source code in C and ASM of MCS51 of the interface program both in IIC and UART. IIC mode is very convenient, user no need to modify the sample code except pin definition in actual using.

Whatever what type of interface user chooses. Please read this chapter before programming and refer to the sample program. There are detailed comments in the sample source code.

4.2 UART protocol

4.2.1 Parameters

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
- Data bits: 8 bits
- Stop bits: 1 bit
- Parity check: None
- Flow control: None

4.2.2 Data send format

Header	Length	Command	Data	Checksum

- Header: 2 bytes, they are 0xAA 0xBB
- Length: 1 byte, number of bytes from Command length byte to the last byte of Data.
- Command: 1 byte, the command of this instruction
- Data: length depends on the command type, length from 0 to 251 bytes
- Checksum: 1 byte, Exclusive OR (XOR) results from length byte to the last byte of data
- If there is 0xAA in data package, and then MUST insert 0x00 follow to distinguish with header. But Length byte in the package does NOT increase

4.2.3 Data return format

- Success:

Header	Length	Command	Data	Checksum

- Failure:

Header	Length	Invert Command	Checksum

4.3 IIC protocol

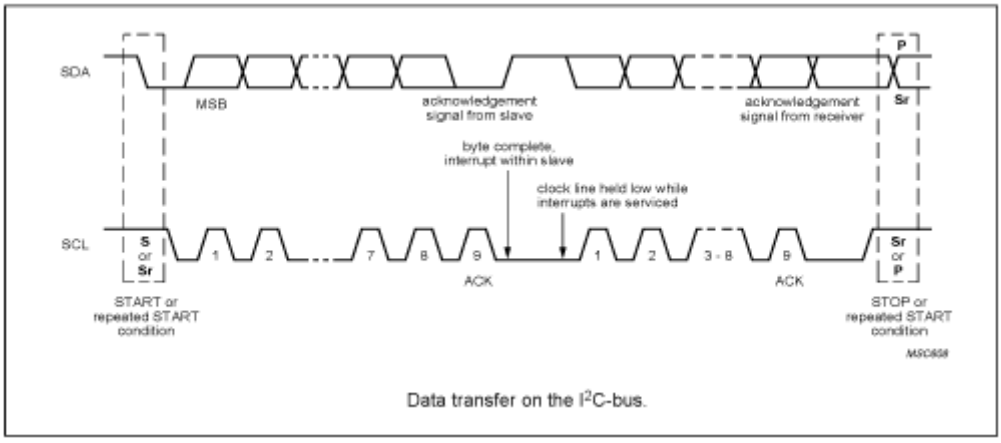
4.3.1 Module IIC address and multi device communications

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

4.3.2 IIC device operation

4.3.2.1 Clock and data transaction

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



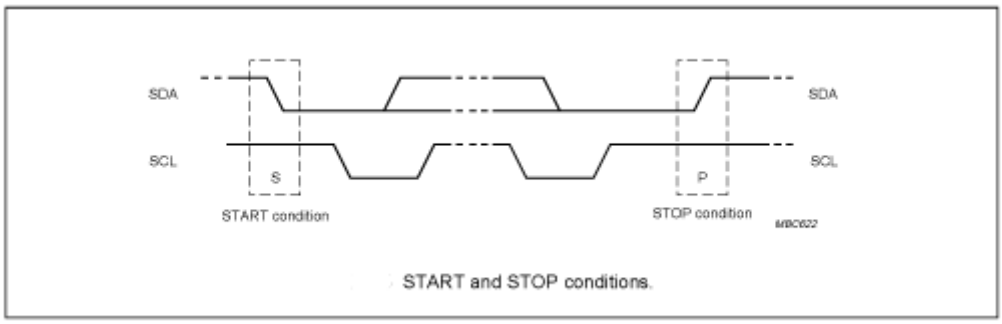
Data transfer on the I²C-bus.

4.3.2.2 Start condition

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

4.3.2.3 Stop condition

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



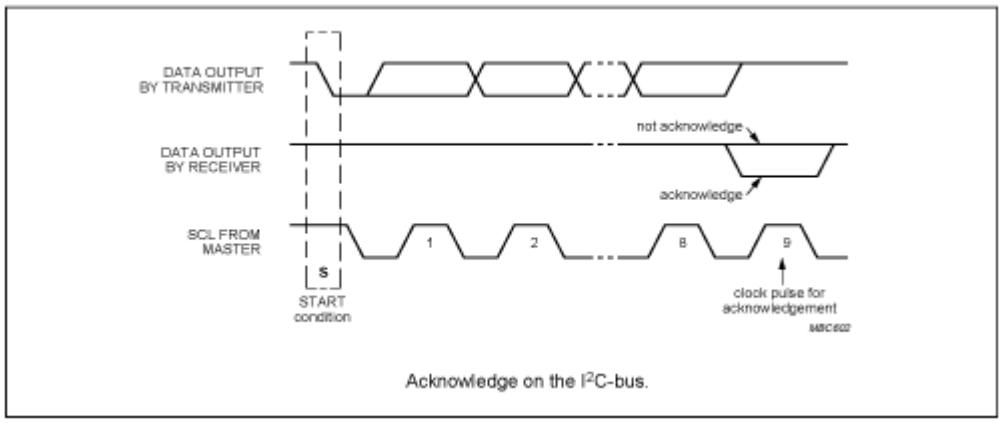
START and STOP conditions.

4.3.2.4 Acknowledge (ACK)

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

4.3.2.5 Bus state

When the module has received command, and then doesn't acknowledge IIC bus until ends with the card communication.



Acknowledge on the I²C-bus.

4.3.2.6 Device addressing

The module requires a 7-bit device address following a start condition to enable the chip for a read or write operation.

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

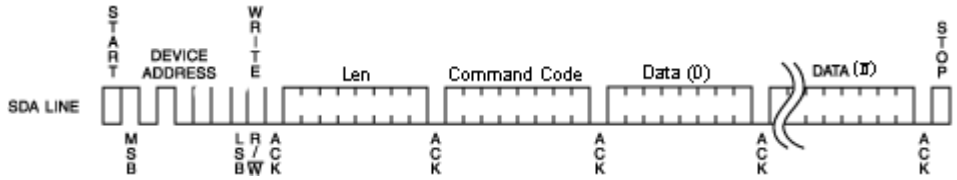
The first 7 bits of the module address are 1010000 (0xA0 in hex)

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

The first byte after the START procedure.

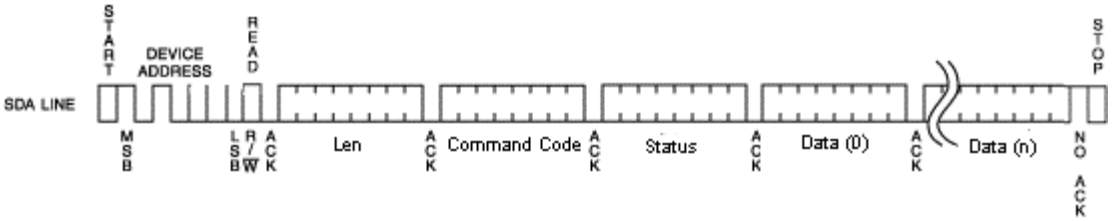
4.3.2.7 Write operation

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



4.3.2.8 Read operation

The host device gets result via read operation.



4.3.3 Data transaction

The module is a slave device of the IIC bus, then the host need to write the command package to module. The module will execute the command. Then the host needs to poll the status of the module while it is working by sending out the command of "read" continuously. If the module answered to a read operation, then the last command execution were finished. At this time the host could read the result and/or data from the module. The read and write operation see chapter 4.3.2.7 and 4.3.2.8.

4.3.4 Data send format

Length	Command	Data	Checksum

- Length: 1 byte, number of bytes from length to the last byte of Data
- Command: 1 byte, the command of this instruction
- Data: Data length depending on the command type, length from 0 to 251 bytes
- Checksum: 1 byte, Exclusive OR (XOR) results from length byte to the last byte of data

4.3.5 Data return format

- Success:

Length	Command	Data	Checksum

- Failure:

Length	Invert Command	Checksum

4.3.6 Description of IIC command transaction

E.g.: to read the block 1 of Mifare card, the steps:
Send command: 0A210001FFFFFFFFFFFF2A
There are steps here:

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

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

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

D. Verify the checksum. if ok then the communication is ok

E. Verify the received data from second byte; this byte is the status of the command just executed. If equal to the command (0x21) then the command execute successful. Then the 16 bytes data started from third byte are correct.

5 Description of commands

5.1 List of commands

Command code	Command function
0x10	Read product information
0x11	Module working mode set
0x12	Sets module idle
0x15	EEPROM read
0x16	EEPROM write
0x17	Set UART communication baud rate
0x19	Set IIC address
0x1A	Set multi-card operation
0x1B	Set ISO15693 automatic detecting card AFI and AFI enable
0x1C	Set automatic detecting card interval time
0x20	ISO14443A Request cards
0x21	Mifare 1K/4K data block read
0x29	Mifare 1K/4K sector (4 blocks) read
0x2A	Mifare 1K/4K multi blocks read
0x22	Mifare 1K/4K data block write
0x2B	Mifare 1K/4K multi blocks write
0x23	Mifare 1K/4K purse block initialize
0x24	Mifare 1K/4K purse read
0x25	Mifare 1K/4K purse increment
0x26	Mifare 1K/4K purse decrement
0x27	Mifare 1K/4K purse copy
0x28	ISO14443A card halt
0x2D	Download Mifare 1K/4K card key to module
0x30	ISO14443-4 TYPE-A card reset
0x31	Send APDU to ISO14443-4 card
0x41	Ultra Light card read
0x42	Ultra Light card write
0x70	Set module card operating protocol
0x60	ISO14443-4 TYPE B card request
0x62	ISO14443-4 TYPE B card halt
0x63	SR serial cards 1 slot initiate card
0x64	SRI serial cards 16 slots initiate card
0x65	SR serial cards select
0x66	SRI serial cards return to inventory
0x67	SR serial cards completion
0x68	SR176 card read
0x69	SR176 card write

0x6A	SR176 data block lock
0x6B	SRI serial cards read
0x6C	SRI serial cards write
0x6D	SRI serial cards lock block
0x6E	SRI serial cards read UID
0x6F	SRIX serial cards authentication
0x5C	ISO15693 inventory
0x5D	ISO15693 stay quiet
0x5E	ISO15693 get system information
0x5F	ISO15693 reset to ready
0x54	ISO15693 read blocks
0x55	ISO15693 write blocks
0x56	ISO15693 block lock
0x57	ISO15693 AFI write
0x58	ISO15693 AFI lock
0x59	ISO15693 DSFID write
0x5A	ISO15693 DSFID lock
0x5B	ISO15693 get blocks security

5.2 Explanation of commands

5.2.1 Read product information

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

Host sends:

0x02	0x10	Checksum

Module returns success:

0x1D	0x10	Information	Checksum

Information: 27 bytes, 8 bytes product name, 4 bytes firmware version, 8 bytes firmware date, 1 byte UART baud rate code, 1byte RFU, 1 byte IIC address, 1 byte multi-card operation enable state, 1 byte ISO15693 automatic detecting card's AFI, 1 byte ISO15693 automatic detecting card's AFI enable state, 1 byte automatic detecting card interval (multiple of 10ms).

Module returns failure:

0x02	0xEF	Checksum

5.2.2 Module working mode set

Function: set the antenna RF output ON/OFF; set the automatic detecting card ON/OFF. Antenna RF output is default ON, and automatic detecting card is OFF. The module will NOT SAVE the setting, and all settings will LOSE on next power up. The multi-card operation will be prohibited while users turn ON the automatic detecting card. If there is more than one card in the RF electric field then the operation will fail.

Host sends:

0x03	0x11	Mode	Checksum

Mode: 1 byte

Antenna status: BIT0 = 0: OFF; BIT0 = 1: ON

Auto request: BIT1 = 0: OFF; BIT1 = 1: ON

Module returns success:

0x02	0x11	Checksum

Module returns failure:

0x02	0xEE	Checksum

5.2.3 Set module idle

Function: set the module idle. In idle mode, the module of RF output turn to OFF, PCD power down, and CPU in idle mode, so the power consumption reduces to about 100uA. Sending the next command to module will wake up the module, and then the RF output ON and automatic detecting card restore default settings. The module will enter into idle mode after the answer procedure is finished. In IIC mode, host need to read the answer and then the module will goes into idle mode.

Host sends:

0x03	0x12	Random data	Checksum

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

Module returns success:

0x02	0x12	Checksum

Module returns failure:

0x02	0xED	Checksum

5.2.4 EEPROM read

Function: read data in EEPROM of the module.

Host sends:

0x05	0x15	Address	Bytes	Checksum

Address: 2 bytes, read start address, address from 0x0000 to 0x01FF, MSB first

Bytes: 1 byte, number of bytes to read, max. 64 bytes

Module returns success:

-	0x15	Data	Checksum

Remark: the byte length is "-", means the byte length depends on the card feedback information. (The same to below)

Data: data read

Module returns failure:

0x02	0xEA	Checksum

5.2.5 EEPROM write

Function: write data into EEPROM of the module

Host sends:

-	0x16	Address	Bytes	Data	Checksum

Address: 2 bytes, read start address, address from 0x0000 to 0x01FF, MSB first

Bytes: 1 byte, number of bytes to read, max. 64 bytes

Data: "Bytes" data to write

Module returns success:

0x02	0x16	Checksum

Module returns failure:

0x02	0xE9	Checksum

5.2.6 Set UART communication baud rate

Function: set UART communication baud rate of the module. After module receive the command, it will first save the new setting, and then send the execute result according to the host. At last it will validate the new setting. UART communication baud rate is default 19200bps. Settings will SAVE in the module; it will not be lost after power OFF.

Host sends:

0x03	0x17	Baud rate	Checksum

Baud rate: 1 byte, baud rate code; 0: 19200bps; 1: 115200bps; other values: RFU

Module returns success:

0x02	0x17	Checksum

Module returns failure:

0x02	0xE8	Checksum

5.2.7 Set IIC communication address

Function: set IIC communication address of the module. After module receive the command, it will first save the new address, and then send the executed result to the host. At last it will validate the new settings. The IIC address of the module is 1 byte HEX data. Lsb is 0; the address of module must be the even number, and the invalid address will NOT be

accepted. Settings will save in the module, and it will be not lost after power OFF.

Host sends:

0x03	0x19	Address	Checksum

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

Module returns success:

0x02	0x19	Checksum

Module returns failure:

0x02	0xE6	Checksum

5.2.8 Set multi-card operation

Function: set multi-card operation. If users need select on card from multi-card, then need to use the multi-card operation. If users set the automatic detecting card, the multi-card operation will be prohibited. If there is more than one card in the RF effective field then the operation will fail. Settings will save in the module; it will be not lost after power OFF. Multi-card operation default enables. This function is suitable for ISO14443A & ISO15693.

Host sends:

0x03	0x1A	Multi-card enable	Checksum

Multi-card enable: 1 byte, 0: disable multi-card; 1: enable multi-card; other values: RFU

Module returns success:

0x02	0x1A	Checksum

Module returns failure:

0x02	0xE5	Checksum

5.2.9 Set ISO15693 automatic detecting card AFI and AFI enable

Function: set AFI and AFI enables of automatic detecting card 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 be not lost after power OFF. AFI is default 0, AFI function is disable.

Host sends:

0x04	0x1B	AFI	AFI enable	Checksum

AFI: 1 byte, AFI, data values: 0~0xFF

AFI enable: 1 byte, 0: unable; 1: enable; other data value: RFU

Module returns success:

0x02	0x1B	Checksum

Module returns failure:

0x02	0xE4	Checksum

5.2.10 Set automatic detecting card interval time

Function: set interval time between two automatic detecting card

Host sends:

0x03	0x1C	Interval Time	Checksum

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

Module returns success:

0x02	0x1C	Checksum

Module returns failure:

0x02	0xE3	Checksum

5.2.11 ISO14443A request cards

Function: ISO14443A request cards, cards include Mifare and other ISO14443A cards. In the return results, user can ascertain the length of serial number via the return data package length, and also judge the card type by ATQA, and judge whether the card supports ISO14443-4 by SAK. If automatic detecting card function was turned on, then this command is read the result of automatic detecting card.

Host sends:

0x03	0x20	Mode	Checksum

Mode: 1 byte, 0: WUPA (request all); 1: REQA (Request not halted only); other value: RFU

Module returns success:

-	0x20	Data	Checksum

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

Module returns failure:

0x02	0xDF	Checksum

5.2.12 Mifare 1K/4K data block read

Function: read Mifare 1K/4K data block

Host sends:

0x0A	0x21	Key ID	Block	Key	Checksum

Key ID: 1 byte, Key identification

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

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

BIT6:BIT5:BIT4:BIT3:BIT2: if using the downloaded, then name the key number here.

(IMPORTANT: please read Chapter 5.3 about Key identification)

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

Module returns success:

0x12	0x21	Data	Checksum

Data: 16 bytes card data

Module returns failure:

0x02	0xDE	Checksum

5.2.13 Mifare 1K/4K sector (4 blocks) read

Function: read Mifare 1K/4K sector (4 blocks). For S50 and sector number less than 32 of S70, this command is called read sector, it will read the sector trailer. For sector 32 to 39 of S70, this command is called “read 4 blocks”. Because the sectors are include 16 blocks, and then module will read 4 blocks. If you need to read the 16 blocks in these sectors, you need do this command 4 times to fill the requirements. The “Sector” in package is: read start block number shift right 2 bits.

Host sends:

0x0A	0x29	Key ID	Sector	Key	Checksum

Key ID: 1 byte, Key identification

Sector: 1 byte, Sector number to read, 0 to 0x0F for S50; 0 to 0x3F for S70

Key: 6 bytes, the key of the card

Module returns success:

0x42	0x29	Data	Checksum

Data: 64 bytes card data

Module returns failure:

0x02	0xD6	Checksum

5.2.14 Mifare 1K/4K multi blocks read

Function: read multi data blocks in the same sector. The function is supported only in the same sector. If cross sectors, then read will fail.

Host sends:

0x0A	0x2A	Key ID	Start Block	Blocks	Key	Checksum

Key ID: 1 byte, key identification

Start Block: 1 byte, start block to read

Blocks: 1byte, number of block to read

Key: 6 bytes, the key of the card

Module returns success:

-	0x2A	Data	Checksum

Data: (blocks)*(16 bytes card data)

Module returns failure:

0x02	0xD5	Checksum

5.2.15 Mifare 1K/4K data block write

Function: write the data to a block of Mifare 1K/4K.

Host sends:

0x1A	0x22	Key ID	Block	Key	Data	Checksum

Key ID: 1 byte, Key identification

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

Key: 6 bytes, the key of the card

Data: 16 bytes data to write

Module returns success:

0x02	0x22	Checksum

Module returns failure:

0x02	0xDD	Checksum

5.2.16 Mifare 1K/4K multi blocks write

Function: write multi data blocks. The function is supported only in the same sector. If cross sector, it will fail when write the first block and prompt the error in the returned result.

Host sends:

0x0A	0x2B	Key ID	Start Block	Blocks	Key	Data	Checksum

Key ID: 1 byte, key identification

Start Block: 1 byte, the start block to write

Blocks: 1 byte, number of block to write

Key: 6 bytes, the key of the card

Data: (blocks)*(16 bytes data to write)

Module returns success:

0x42	0x2B	Checksum

Module returns failure:

0x02	0xD4	Checksum

5.2.17 Mifare 1K/4K purse block initialize

Function: initialize a block of Mifare 1K/4K to a purse. The format of purse uses Mifare 1K/4K's default. The key of the card could not use as a purse.

Host sends:

0x0E	0x23	Key ID	Block	Key	Value	Checksum

Key ID: 1 byte, Key identification

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

Key: 6 bytes, the key of the card

Value: 4 bytes, initialized value, LSB first

Module returns success:

0x02	0x23	Checksum

Module returns failure:

0x02	0xDC	Checksum

5.2.18 Mifare 1K/4K purse read

Function: read a purse of Mifare 1K/4K. The format of the purse uses Mifare 1K/4K's default. Module will read the data in the block and check if it is a purse format. If yes, return 4 bytes value data, if no, return failure.

Host sends:

0x0A	0x24	Key ID	Block	Key	Checksum

Key ID: 1 byte, Key identification

Block: 1 byte, block number of the value to read, 0 to 0x3F for S50; 0 to 0xFF for S70

Key: 6 bytes, the key of the card

Module returns success:

0x06	0x24	Data	Checksum

Data: 4 bytes value data, LSB first

Module returns failure:

0x02	0xDB	Checksum

5.2.19 Mifare 1K/4K purse increment

Function: purse increment of Mifare 1K/4K. The format of the purse uses Mifare1K/4K's default. Purse increment means the increment on the basis of the original number.

Host sends:

0x0E	0x25	Key ID	Block	Key	Value	Checksum

Key ID: 1 byte, Key identification

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

Key: 6 bytes, the key of the card

Value: 4 bytes, increment value, LSB first

Module returns success:

0x02	0x25	Checksum

Module returns failure:

0x02	0xDA	Checksum

5.2.20 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:

0x0E	0x26	Key ID	Block	Key	Value	Checksum

Key ID: 1 byte, Key identification

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

Key: 6 bytes, the key of the card

Value: 4 bytes, increment value, LSB first

Module returns success:

0x02	0x26	Checksum

Module returns failure:

0x02	0xD9	Checksum

5.2.21 Mifare 1K/4K purse copy

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:

0x0B	0x27	Key ID	Source	Target	Key	Checksum

Key ID: 1 byte, Key identification

Source: 1 byte, block number to copy, 0 to 0x3F for S50; 0 to 0xFF for S70

Target: 1 byte, copy the purse to this block (source and target need in same sector)

Key: 6 bytes, the key of the card

Module returns success:

0x02	0x27	Checksum

Module returns failure:

0x02	0xD8	Checksum

5.2.22 ISO14443A card halt

Function: set the current operating ISO14443A card in halt state.

Host sends:

0x02	0x28	Checksum

Module returns success:

0x02	0x28	Checksum

Module returns failure:

0x02	0xD7	Checksum

5.2.23 Download Mifare 1K/4K card key to module

Function: download the Mifare 1K/4K card key to module. There are 32 key memory spaces in the module that can storage 32 different keys. When using the downloaded key on the module, this key wouldn't appear on the pin-outs of the PCD. So it could provide more security.

Host sends:

0x09	0x2D	Key Index	Key	Checksum

Key Index: 1 byte, store the Key Index in the module

Key: 6 bytes, the key of the card to store in module

Module returns success:

0x02	0x2D	Checksum

Module returns failure:

0x02	0xD2	Checksum

5.2.24 ISO14443-4 TYPE-A card reset

Function: reset an ISO14443-4 TYPE-A card. Before executing this command, it needs to request card and verifies the card support ISO14443-4 in the SAK of card. If operate ISO14443-4 card, then need to turn OFF the automatic detecting card. That's because the ISO14443-4 card state will be lost in the automatic detecting card.

Host sends:

0x02	0x30	Checksum

Module returns success:

-	0x30	Info	Checksum

Info: card reset information, length depends on card

Module returns failure:

0x02	0xCF	Checksum

5.2.25 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, then need to turn OFF the automatic detecting card. That's because the ISO14443-4 card's state will be lost in automatic detecting card.

Host sends:

-	0x31	APDU	Checksum

APDU: APDU to send

Module returns success:

-	0x31	Response	Checksum

Response: card answers, length depends on the detailed command

Module returns failure:

0x02	0xCE	Checksum

5.2.26 Ultra Light card read

Function: read the data from Ultra Light card. A read command will read 4 blocks data from the card. If read start block is the last block, then these 4 blocks data are the 15th, 0th, 1st and 2nd block.

Host sends:

0x05	0x41	Read start block	Checksum

Read start block: 1 byte, start block number to read

Module returns success:

0x12	0x41	Data	Checksum

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

Module returns failure:

0x02	0xBE	Checksum

5.2.27 Ultra Light card write

Function: write data to Ultra Light card.

Host sends:

0x05	0x42	Block	Data	Checksum

Block: 1 byte, block number to write

Data: 4 bytes data to write

Module returns success:

0x12	0x42	Checksum

Module returns failure:

0x02	0xBD	Checksum

5.2.28 Set module card operating protocol

Function: set module card operating protocol, default is ISO14443A. The setting will not be saved and will return to the default state at next power up**.**

Host sends:

0x03	0x70	Model	Checksum

Model: 1 byte, 0: ISO14443A; 1: ISO14443B; 2: ISO15693; other value: RFU

Module returns success:

0x12	0x70	Checksum

Module returns failure:

0x02	0x8F	Checksum

5.2.29 ISO14443-4 TYPE B card request

Function: ISO14443-4 TYPE B card request and set the communication parameters. The reader will skip the remains operation after one card answer successful.

Host sends:

0x05	0x60	Model	AFI	SLOT	Checksum

Model: 1 byte, 0: WUPB (Wakeup B); 1: REQB (Request B); other values: RFU

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

SLOT: 1 byte, slot numbers for request; uses 1, 2, 4, 8, 16, all other value are RFU

Module returns success:

0x0E	0x60	Info	Checksum

Info: 12 bytes, card reset information

Module returns failure:

0x02	0x9F	Checksum

5.2.30 ISO14443-4 TYPE B card halt

Function: set the current ISO14443B card halt.

Host sends:

0x03	0x62	PUPI	Checksum

PUPI: 4 bytes, PUPI of the card to halt

Module returns success:

0x02	0x62	Checksum

Module returns failure:

0x02	0x9D	Checksum

5.2.31 SR serial cards 1 slot initiate card

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

Host sends:

0x06	0x63	Checksum

Module returns success:

0x03	0x63	Card ID	Checksum

Card ID: 1 byte, card ID

Module returns failure:

0x02	0x9C	Checksum

5.2.32 SRI serial cards 16 slots initiate card

Function: SRI serial cards 16 slots initiate card.

Host sends:

0x02	0x64	Checksum

Module returns success:

0x22	0x64	Status	Card ID	Checksum

Status: 16 bytes, the initiate result of 16 channels, 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 status of current channel is successful

Module returns failure:

0x02	0x9B	Checksum

5.2.33 SR serial cards select

Function: select a SR card as the CURRENT CARD. You could operate the card after select.

Host sends:

0x03	0x65	Card ID	Checksum

Card ID: 1 byte; Card ID to select

Module returns success:

0x03	0x65	Card ID	Checksum

Card ID: 1 byte; the selected card ID

Module returns failure:

0x02	0x9A	Checksum

5.2.34 SRI serial cards return to inventory

Function: set a selected SRI card return to inventory state.

Host sends:

0x02	0x66	Checksum

Module returns success:

0x02	0x66	Checksum

Module returns failure:

0x02	0x99	Checksum

5.2.35 SR serial cards completion

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

Host sends:

0x02	0x67	Checksum

Module returns success:

0x02	0x67	Checksum

Module returns failure:

0x02	0x98	Checksum

5.2.36 SR176 card read

Function: read SR176 card data block.

Host sends:

0x06	0x68	Block	Checksum

Block: 1 byte, data block number to read

Module returns success:

0x04	0x68	Data	Checksum

Data: 2 bytes, data read

Module returns failure:

0x02	0x97	Checksum

5.2.37 SR176 card write

Function: write the data block of SR176 card. After write, module will read the data and compare. If not equal, then return failure.

Host sends:

0x05	0x69	Block	Data	Checksum

Block: 1 byte, data block number to read

Data: 2 bytes, data to write

Module returns success:

0x02	0x69	Checksum

Module returns failure:

0x02	0x96	Checksum

5.2.38 SR176 data block lock

Function: write the data of SR176 lock register of the card. The module will check the lock result after write.

Host sends:

0x03	0x6A	Lock value	Checksum

Locked value: 1 byte; the lock register values to write

Module returns success:

0x02	0x6A	Checksum

Module returns failure:

0x02	0x95	Checksum

5.2.39 SRI serial cards read

Function: read data block of SRI serial card

Host sends:

0x03	0x6B	Block	Checksum

Block: 1 byte; data block number to read

Module returns success:

0x06	0x6B	Data	Checksum

Data: 4 bytes, data read

Module returns failure:

0x02	0x94	Checksum

5.2.40 SRI serial cards write

Function: write data block of SRI serial card. After write, module will read data and compare. If not equal, then return failure.

Host sends:

0x07	0x6C	Block	Data	Checksum

Block: 1 byte; data block number to read

Data: 4 bytes; data to write

Module returns success:

0x02	0x6C	Checksum

Module returns failure:

0x02	0x93	Checksum

5.2.41 SRI serial cards data block lock

Function: write the data of SRI card to lock the register. It will check the locked result after write.

Host sends:

0x03	0x6D	Locked value	Checksum

Locked value: 1 byte; the locked register values to write

Module returns success:

0x02	0x6D	Checksum

Module returns failure:

0x02	0x92	Checksum

5.2.42 SRI serial cards read UID

Function: SRI serial cards read UID

Host sends:

0x02	0x6E	Checksum

Module returns success:

0x0A	0x6E	UID	Checksum

UID: 8 bytes, UID of CURRENT CARD

Module returns failure:

0x02	0x91	Checksum

5.2.43 SRIX serial cards authentication

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

Host sends:

0x06	0x6F	Data	Checksum

Data: 6 bytes, data input

Module returns success:

0x05	0x6F	Result	Checksum

Result: 3 bytes, result return

Module returns failure:

0x02	0x90	Checksum

5.2.44 ISO15693 inventory

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

Host sends:

0x03	0x5C	AFI	Checksum

AFI: 1byte AFI, inventory card equal to AFI only

If not use AFI, then host sends:

0x02	0x5C	Checksum

Module returns success:

0x0B	0x5C	DSFID	UID	Checksum

DSFID: 1 byte, DSFID of CURRENT TAG

UID: 8 bytes, UID of CURRENT TAG

Module returns failure:

0x02	0xA3	Checksum

5.2.45 ISO15693 stay quiet

Function: set the CURRENT TAG stay quiet

Host sends:

0x02	0x5D	Checksum

Module returns success:

0x02	0x5D	Checksum

Module returns failure:

0x02	0xA2	Checksum

5.2.46 ISO15693 get system information

Function: get the system information of CURRENT TAG

Host sends:

0x02	0x5E	Checksum

Module returns success:

-	0x5E	Data	Checksum

Data: tag information, the length is a variable, depends on the manufacturer of the tag

Module returns failure:

0x02	0xA1	Checksum

5.2.47 ISO15693 reset to ready

Function: set a stay quiet TAG reset to ready

Host sends:

0x0A	0x5F	Data	Checksum

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

Module returns success:

0x02	0x5F	Checksum

Module returns failure:

0x02	0xA0	Checksum

5.2.48 ISO15693 read blocks

Function: read data blocks of CURRENT TAG

Host sends:

0x04	0x54	Start	Blocks	Checksum

Start: 1 byte, read start block

Blocks: 1 byte, number of blocks to read, max. 62 blocks in one command

Module returns success:

-	0x54	Data	Checksum

Data: Blocks * 4 bytes,

Module returns failure:

0x02	0xAB	Checksum

5.2.49 ISO15693 write blocks

Function: write data blocks of CURRENT TAG

Host sends:

-	0x55	Start	Blocks	Data	Checksum

Start: 1 byte, write start block

Blocks: 1 byte, number of blocks needs to write, max. 62 blocks

Data: Blocks * 4 bytes, data to write to tag

Module returns success:

0x02	0x55	Checksum

Module returns failure:

0x02	0xAA	Checksum

5.2.50 ISO15693 block lock

Function: lock a block of CURRENT TAG

Host sends:

0x03	0x56	Block	Checksum

Block: 1 byte, block number to lock

Module returns success:

0x02	0x56	Checksum

Module returns failure:

0x02	0xA9	Checksum

5.2.51 ISO15693 AFI write

Function: write AFI to CURRENT TAG

Host sends:

0x03	0x57	AFI	Checksum

AFI: 1 byte, AFI value to write to tag

Module returns success:

0x02	0x57	Checksum

Module returns failure:

0x02	0xA8	Checksum

5.2.52 ISO15693 AFI lock

Function: lock AFI of CURRENT TAG

Host sends:

0x02	0x58	Checksum

Module returns success:

0x02	0x58	Checksum

Module returns failure:

0x02	0xA7	Checksum

5.2.53 ISO15693 DSFID write

Function: write DSFID of CURRENT TAG

Host sends:

0x03	0x59	DSFID	Checksum

DSFID: 1 byte, DSFID value to write to tag

Module returns success:

0x02	0x59	Checksum

Module returns failure:

0x02	0xA6	Checksum

5.2.54 ISO15693 DSFID lock

Function: lock DSFID of CURRENT TAG

Host sends:

0x02	0x5A	Checksum

Module returns success:

0x02	0x5A	Checksum

Module returns failure:

0x02	0xA5	Checksum

5.2.55 ISO15693 get blocks security

Function: get blocks security of CURRENT TAG

Host sends:

0x04	0x5B	Start	Blocks	Checksum

Start: 1 byte, start block

Blocks: 1 byte, number of blocks

Module returns success:

-	0x5B	Data	Checksum

Data: bytes equal to the sent blocks in the command, the locked info of data block

Module returns failure:

0x02	0xA4	Checksum

5.3 About KEY Identification

There is a byte of KEY identification in command of Mifare 1K/4K read/write. This byte will identify the way to get the card key.

Key Identification							
BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
0							

BIT0 0: KEY A; authenticate Key A of the card.
1: KEY B; authenticate Key B of the card.

BIT1 0: Using the following Key in command.
1: Using the downloaded Key by command 0x2D.

BIT6: BIT5: BIT4: BIT3: BIT2: Index of the Key already downloaded (0 to 31).

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) by using command 0x2D 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 Identification is 0x30, 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 unused in this command.

E.g.: key Identification is 0x33; binary system is 00110011, here:

BIT0 = 1; authenticate Key B of the card

BIT1 = 1; using the downloaded Key in the module

BIT6:BIT5:BIT4:BIT3:BIT2:01100, then use the already downloaded key 01100, and hexadecimal is 0x0C, decimal is 12.

5.4 About automatic detecting card

The automatic detecting card function supports ISO14443A and ISO15963. It is default OFF. User could set the automatic detecting card on by send command 0x11. This setting will lose on next power up.

If the card operating protocol is set to ISO15963, then automatic detecting card detect the ISO15963 cards. And if card operating protocol is set to ISO14443A, then detect the Mifare class and ISO14443A.

Automatic detecting card supports full function of ISO15963.

Automatic detecting card supports full function of Mifare 1K/4K and Ultra Light.

Automatic detecting card can find ISO14443A smart cards. If user needs to send APDU to the card, then must turn automatic detecting card OFF for correct operation.

Automatic detecting card supports only one card operation. If there is more than one card in the RF effective field then the operation may fail. Then the multi-card operation will automatically turn OFF while the automatic detecting card function turned on.

5.5 Example of commands

5.5.1 About UART communication protocol

For example:

Read block 1: AABB 0A210001AA00BBCCDDEEFF2A

AABB: Header of UART protocol, IIC protocol no this part

0A: package length; from 0A to FF are total 0x0A bytes, the 00 in red is a protocol byte, see chapter 4.2.2

21: instruction of read

00: Authenticate KEY A, using the key in package. The key is “AABBCCDDEEFF”

01: block number to read

AABBCCDDEEFF: key of the sector of the card

00: protocol byte, used to distinguish header. See chapter 3.1.2

2A: 0A ^ 21 ^ 00 ^ 01 ^ AA ^ BB ^ CC ^ DD ^ EE ^ FF = 2A, in sample program, the function will calculate it, see chapter 4.5

5.5.2 UART commands sample

Read block 1	AABB 0A210001FFFFFFFFFFFF2A
Read block 255 (S70)	AABB 0A2100FFFFFFFFFFFFFFD4
Write block 1	AABB 1A220001FFFFFFFFFFFF1234567890ABCDEF1234567890ABCDEF39
Request card (WUPA)	AABB 03200023

Halt card	AABB 021210
ISO15693 inventory	AABB035C005F
ISO15693 read blocks	AABB0454000858
ISO15693 write blocks	AABB0C55080211223344AA00BBCCDD17
ISO15693 get system information	AABB025E5C

5.5.3 IIC commands sample

Read block 1	0A210001FFFFFFFFFFFF2A
Read block 255 (S70)	0A2100FFFFFFFFFFFFFFD4
Write block 1	1A220001FFFFFFFFFFFF1234567890ABCDEF1234567890ABCDEF39
Request card (WUPA)	03200023
Halt card	021210
ISO15693 inventory	035C005F
ISO15693 read blocks	0454000858
ISO15693 write blocks	0C55080211223344AABBCCDD17
ISO15693 get system information	025E5C

5.6 Interface program source code

We have interface program source code to help users. They are KELL project in C51 or ASM51 format. Please mail to jinmuyu@vip.sina.com to obtain the program.