# JMY603C User's Manual

## (Revision 3.42)

**Jinmuyu Electronics Co. LTD**

**2011/6/27**

# Contents

# 1 Product introduction

JMY603C is RFID reader/writer module with an UART port. JMY603C has various functions and supports multi ISO/IEC standard of contactless card. The RF protocol is complex. The designer combined some frequently used command of RF card and then user could operate the cards with full function by sending simple command to the module.

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

# 2 Characteristics

- PCD model:               NXP MF RC531
- Working frequency:       13.56MHz
- Supported standard:      ISO14443A, ISO14443B
- Card supported:          Mifare 1K/4K, FM11RF08, Ultra Light, DesFire, Mifare ProX, SR176, SRI512, SRI1K, SRI2K, SRI4K, SRIX4K, T=CL smart cards (both ISO14443A & ISO14443B)
- 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:               RS232C
- Communication rate:      19.2Kbps/115.2Kbps
- Max. command length:     254 Bytes
- Power consumption:       80mA
- Operating distance:      100mm (depending on card and antenna design)
- Dimension:               85mm*59mm*5.6mm
- Weight:                  About 30g
- ISP:                     Supported
- Operating temperature:   -25 to +85℃
- Storage temperature:     -40 to +125℃
- RoHS:                    Compliant

# 3  Physical parameter and pin outs

## 3.1  Photo
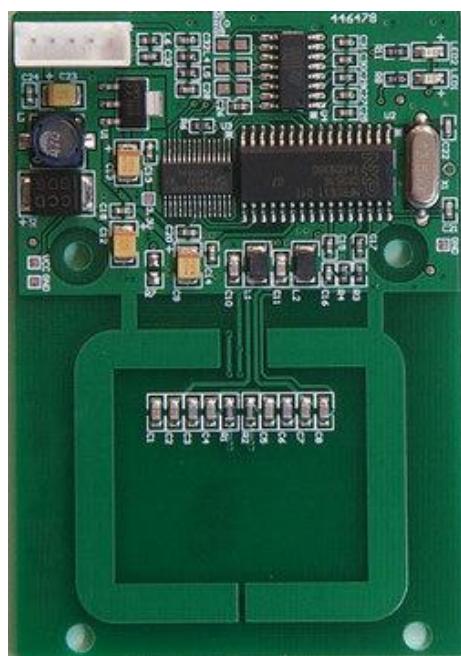


## 3.2  Dimension
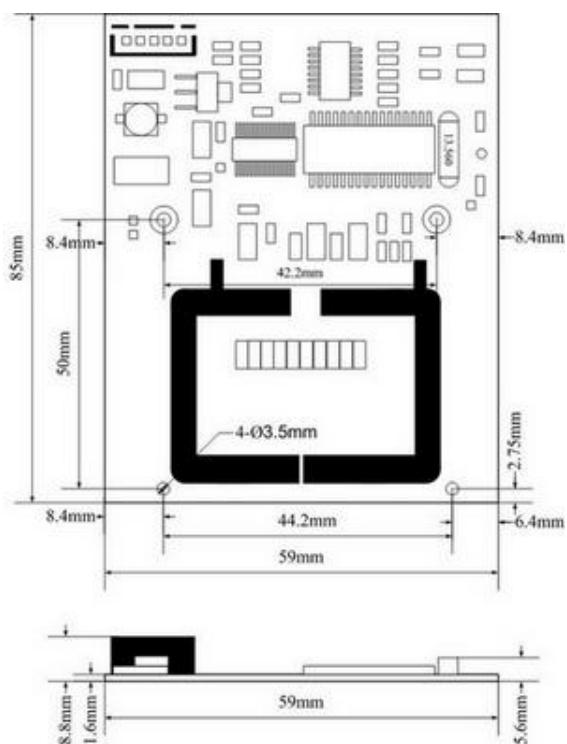
## 3.3  Pin configurations and Pin outs

| PIN | Function | Type | Description |
|---|---|---|---|
| 1 | ICC | Output | Card in/out: 0-card in; 1-card out |
| 2 | RXD | Input | RXD |
| 3 | TXD | Output | TXD |
| 4 | VCC | Power | VCC |
| 5 | GND | Power | GND |

## 3.4  Model available

● JMY603CS          RS232 (UART interface, 232 level)

## 3.5  Model rule

### 3.5.1  Model format

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| JMY | 603 | X | X |

1: company code; 2: product series code; 3: card operating type; 4: communication port type

### 3.5.2  Card operating type

A: PCD is RC500, support ISO14443A and Mifare Class
C: PCD is RC531, support ISO14443A, ISO14443B and Mifare Class

### 3.5.3  Communication port

S: RS232C

# 4 Communication Protocols

## 4.1 Overview

There is COMS UART port on the module. 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

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

## 4.3 Data send format

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

- 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

## 4.4 Data return format

- Success:

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

- Failure:

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

# 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 |
| 0x1A | Set multi-card operation |
| 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 contactless 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 |

## 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, 2 bytes RFU, 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 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 only.

**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 |
|------|------|----------|

## 5.2.8 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 |
|------|------|----------|

## 5.2.9 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 |
|------|------|----------|

## 5.2.10  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 key downloaded by

command 0x2D

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

   (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 |
|------|------|----------|

## 5.2.11  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 |
|------|------|----------|

## 5.2.12 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 |
|------|------|----------|

## 5.2.13 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 |
|------|------|----------|

## 5.2.14  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 while writing the first block in the next sector and then 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 |
|------|------|----------|

## 5.2.15  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 |
|------|------|----------|

## 5.2.16  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 |
|------|------|----------|

## 5.2.17  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 |
|------|------|----------|

## 5.2.18  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 |
|------|------|----------|

## 5.2.19  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 |
|------|------|----------|

## 5.2.20  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 |
|------|------|----------|

## 5.2.21  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 |
|------|------|----------|

## 5.2.22  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 |
|------|------|----------|

## 5.2.23  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 |
|------|------|----------|

## 5.2.24  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 |
|------|------|----------|

## 5.2.25  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 |
|------|------|----------|

## 5.2.26  Set module contactless 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 |
|------|------|----------|

## 5.2.27  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 |

## 5.2.28  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 |

## 5.2.29  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 |
|------|------|----------|

## 5.2.30  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 |
|------|------|----------|

## 5.2.31  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 |
|------|------|----------|

## 5.2.32  SRI serial cards return to inventory

**Function:** set a selected SRI card returning to inventory state.

**Host sends:**

| 0x02 | 0x66 | Checksum |
|------|------|----------|

**Module returns success:**

| 0x02 | 0x66 | Checksum |
|------|------|----------|

**Module returns failure:**

| 0x02 | 0x99 | Checksum |
|------|------|----------|

## 5.2.33  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 |
|------|------|----------|

## 5.2.34  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 |
|------|------|----------|

## 5.2.35  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 |
|---|---|---|

## 5.2.36  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 |
|---|---|---|

## 5.2.37  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 |
|---|---|---|

## 5.2.38  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 |
|------|------|----------|

## 5.2.39  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 |
|------|------|----------|

## 5.2.40  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 |
|------|------|----------|

## 5.2.41  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 |
|------|------|----------|

# 5.3  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 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 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 the automatic detecting card

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

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: 0A210001AABBCCDDEEFF2A

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

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

### 5.5.2 UART commands sample

| | |
|---|---|
| Read block 1 | 0A210001FFFFFFFFFFFF2A |
| Read block 255 (S70) | 0A2100FFFFFFFFFFFFFFFFD4 |
| Write block 1 | 1A220001FFFFFFFFFFFF1234567890ABCDEF1234567890ABCDEF39 |
| Request card (WUPA) | AABB03200023 |
| Halt card | AABB021210 |

## 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.