MIFARE & ISO14443A & ISO14443B & ISO15693 CONTACTLESS, ISO7816 CONTACT IC CARD READ/WIRTE MODULE

# JMY680D IC Card Reader

## User's manual

(Revision 3.50)

**Jinmuyu Electronics Co. LTD**

**2012/6/28**

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

# Contents

# 1 Product introduction

JMY680D is a RFID read/write module with UART, IIC, RS232C or USB port. JMY680D has various functions and supports multi ISO/IEC standard of contactless card. The RF protocol is complex, but 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 modules build in SAM slot. It could operate contact smart card according to ISO7816.

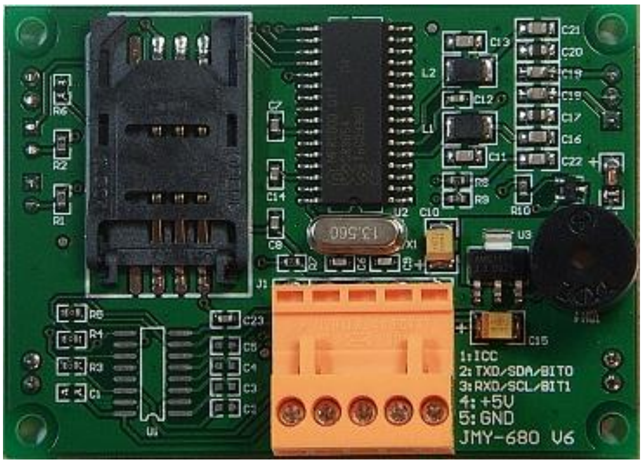
The module has a length of 506 bytes command buffer could send APDU over 256 bytes to T=CL smart cards and SAM cards. The modules support FSDI=8 of ISO14443-4. The module and antenna is integrated. The impedance between RF circuit and antenna was tuned by impedance analyzer, and then the module has excellent performance and stability. There is ferrite plate between main PCB and antenna, so such design applies to some metallic-around systems.

# 2 Characteristics

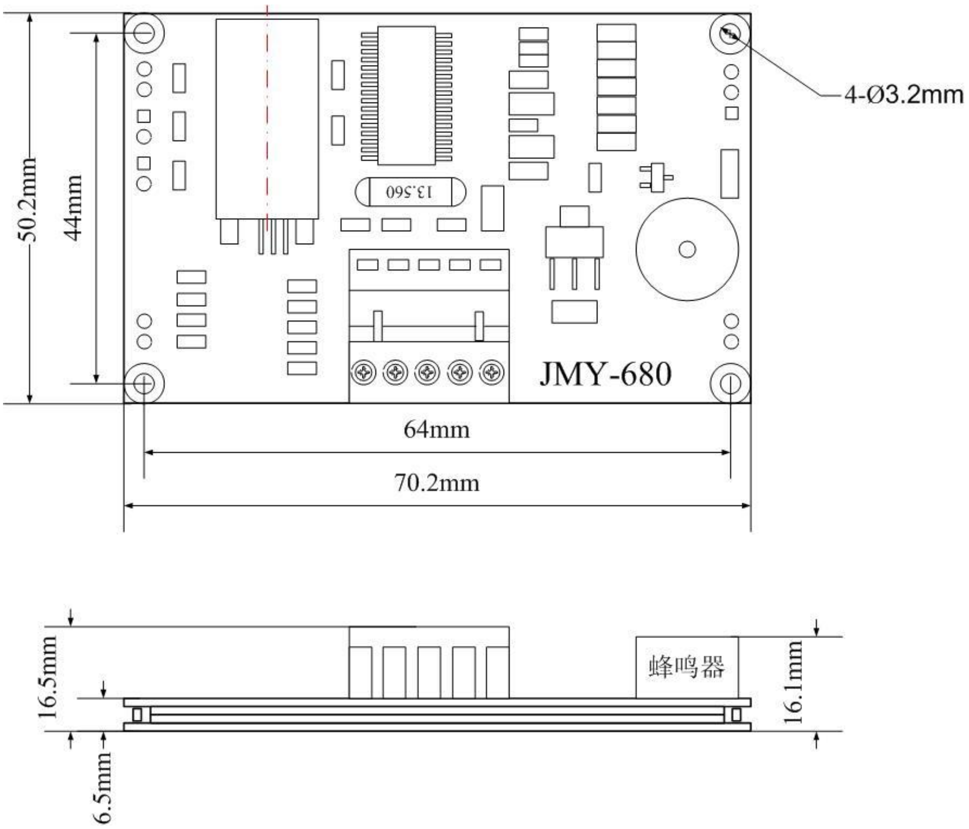
- PCD model: NXP MF RC500
- Working frequency: 13.56MHz
- Supported standard: ISO14443A, ISO7816
- Card supported: Mifare 1K/4K, FM11RF08, Ultra Light, DesFire, Mifare ProX, T=CL CPU cards(ISO14443A only) and ISO7816 SAM cards (both T=0 & T=1)
- Anti collision ability: Full function anti collision; be able to process multi-cards; be able to set operate single card only
- Auto detecting card: Supported, default OFF. The default state could be set.
- SAM slot: 1 slot
- SAM baud rate: 9600bps/38400bps
- ISO7816 PPS set: Supported
- EEPROM: 512 Bytes
- Power supply: DC 5V (±0.5V)
- Interface: IIC/UART/RS232C/USB (select when place order)
- Communication rate: IIC: 400Kbps  
  UART/RS232C/USB: 19.2Kbps/115.2Kbps
- Max. command length: 511 Bytes
- Interface level: UART/IIC: 3.3V (TTL level; 5V tolerance)
- Power consumption: 150mA
- Operating distance: 80mm (depending on card and antenna design)
- Dimension: 70mm*50mm*16.5mm
- Weight: About 120g
- 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 number | Function | Type | Description |
|---|---|---|---|
| 1 | ICC | Output | Card in/out indication 0: Card IN; 1: Card OUT |
| 2 | TXD/SDA | Input/output | RS232C TXD / UART TXD / IIC SDA |
| 3 | RXD/SCL | Input | RS232C RXD / UART RXD / IIC SCL |
| 4 | VCC | Power | VCC |
| 5 | GND | Power | GND |

## 3.4 Model available

| | |
|---|---|
| JMY680DI | IIC interface |
| JMY680DT | UART interface, TTL level |
| JMY680DS | RS232C (UART interface, RS232 level) |
| JMY680DU | USB to UART Bridge (Mini USB port with 5 pins) |

# 3.5 Model naming rule

## 3.5.1 Model format

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| JMY | 680 | X | X |

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

## 3.5.2 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 ISO14443A/B and Mifare Class with 511 bytes communication buffer
F: PCD is RC632, support ISO15693, ISO14443A, ISO14443B and Mifare Class with 511 bytes communication buffer

## 3.5.3 Communication port

I: IIC
T: UART
S: RS232C
U: USB

# 4 Communication Protocols

## 4.1 Overview

There are IIC and UART two types of hardware interface between the module and host. We recommend using IIC interface whose communication data rate is up to 400Kbps. 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 may not modify the sample code except pin definition for actually use.

Whatever types 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

- Host send:

| Length | C.A. | Command | Data | Checksum |
|---|---|---|---|---|

- Length: 2 bytes, number of bytes from length byte to the last byte of Data, MSB first
- C.A.(communication address): the address of UART multi-device communication, default address: 1; broadcast address: 0
- Command: 1 byte, the command of this instruction
- Data: length depends on the command type, length from 0 to 506 bytes
- Checksum: 1 byte, Exclusive OR (XOR) results from length byte to the last byte of data

### 4.2.3 Data return format

- Success:

| Length | C.A. | Command | Data | Checksum |
|---|---|---|---|---|

- Failure:

| Length | C.A. | Invert Command | Checksum |
|---|---|---|---|

# 4.3 IIC protocol

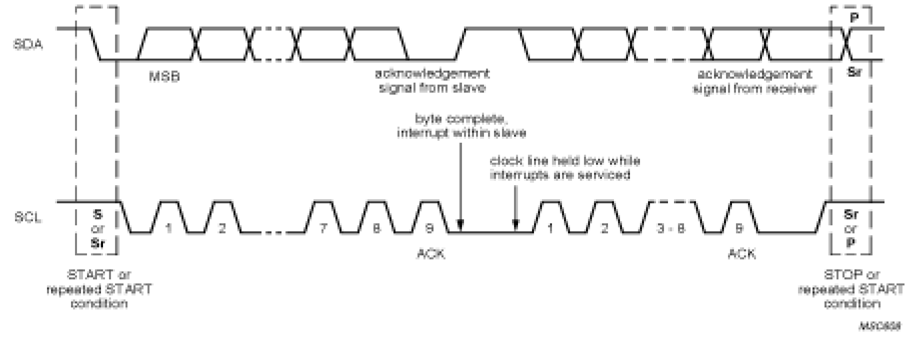
## 4.3.1 Module IIC address and multi device communications

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.


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

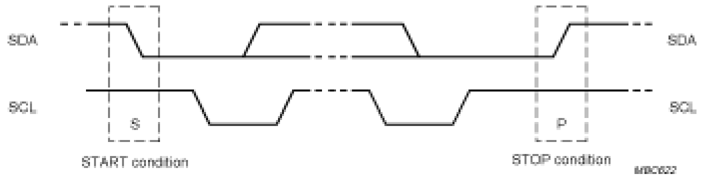
Data transfer on the $I^2C$-bus.

### 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.



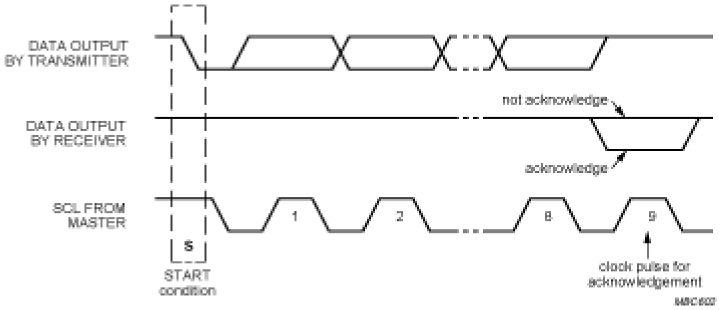
START and STOP conditions.

### 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 I$^2$C-bus.

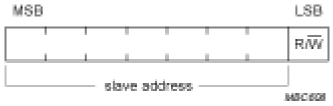
### 4.3.2.6 Device addressing

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

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

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

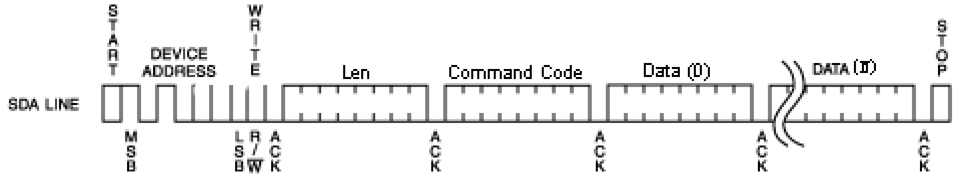
The eighth bit of the device address is the read/write operation select bit. A read operation is initiated if this bit is high and a write operation is initiated if this bit is low.



The first byte after the START procedure.

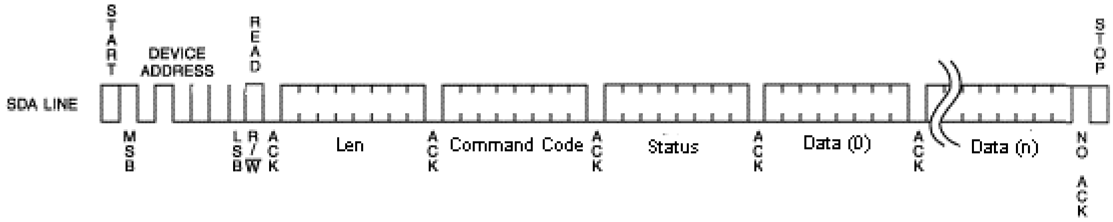
### 4.3.2.7 Write data operation

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



#### 4.3.2.8 Read data 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 4.2.2.7 and 4.2.2.8.

### 4.3.4 Data send format

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

- Length: 2 bytes, number of bytes from length to the last byte of Data, MSB first
- Command: 1 byte, the command of this instruction
- RFU: 1 byte, the address of UART multi-device communication; when IIC, then write 0
- Data: Data length depending on the command type, length from 0 to 506 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 | RFU | Command | Data | Checksum |
|---|---|---|---|---|

- Failure:

| Length | RFU | 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: 000C00210001FFFFFFFFFFFF2C

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: 0x000C210001FFFFFFFFFFFF
   4. Send command checksum: 0x2C
   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 2 bytes and send ACK, if the data is 0x0014, the meaning is there are 0x0014 bytes useful bytes in this package.
   2. Get the else 18 bytes data (0x0014-2=0x0012) 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 fourth 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 the product information |
| 0x11 | Module working mode set |
| 0x12 | Sets module idle |
| 0x13 | Set LED |
| 0x14 | Set the buzzer |
| 0x15 | EEPROM read |
| 0x16 | EEPROM write |
| 0x17 | Set UART communication baud rate |
| 0x18 | Set UART multi-device communication address |
| 0x19 | Set IIC address |
| 0x1A | Set multi-card operation |
| 0x1C | Set automatic detecting card interval time |
| 0x1D | Set the default automatic detect card state default |
| 0x1E | Set automatic detect card and output card UID default |
| 0x20 | ISO14443A Request cards |
| 0x21 | Mifare 1K/4K data block 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 (RATS) |
| 0x31 | Send APDU to ISO14443-4 card |
| 0x41 | Ultra Light card read |
| 0x42 | Ultra Light card write |
| 0x50 | SAM slot reset baud rate set |
| 0x51 | SAM reset |
| 0x52 | Set SAM baud rate after reset (through PPS) |
| 0x53 | Send APDU to SAM |

# 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:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0x10 | Checksum |

**Module returns success:**

| | | | | |
|---|---|---|---|---|
| 0x0021 | C.A. | 0x10 | Info. | Checksum |

**Information:** 29 bytes, 8 bytes product name(0x4A 4D 59 36 38 30 44 20), 4 bytes firmware version(0x35 2E 33 33), 8 bytes firmware date(0x32 30 31 32 30 35 32 39), 1 byte UART baud rate code(0x00), 1byte UART multi-device communication address(0x01), 1 byte IIC address(0xA0), 1 byte multi-card operation enable state(0x01), 1 byte ISO15693 automatic detecting card AFI(0x00), 2 bytes RFU, 1 byte automatic detecting card interval(0x14) (multiple of 10mS), 1 byte default automatic detecting card status when power on(0x00), 1 byte default automatic output SNR set when power on(0x00)

Data in the brackets above taken from JMY680D default product information, as follow:

Send: 0x00 04 00 10 14

Return: 0x00 21 01 10 4A 4D 59 36 38 30 44 20 35 2E 33 33 32 30 31 32 30 35 32 39 00 01 A0 01 00 00 14 00 00 94

**Module returns failure:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0xEF | Checksum |

## 5.2.2 Module working mode set

**Function:** Set the antenna RF output ON/OFF; set the automatic detecting card ON/OFF; set automatic detect card and output card UID ON/OFF. Antenna RF output is default ON, and automatic detecting card is OFF, automatic detect card and output card UID if 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:**

| | | | | |
|---|---|---|---|---|
| 0x0005 | C.A. | 0x11 | Mode | Checksum |

Mode: 1 byte

| | | |
|---|---|---|
| Antenna status: | BIT0= 0: OFF; | BIT0= 1: ON |
| Auto request: | BIT1= 0: OFF; | BIT1= 1: ON |
| Auto request and output UID: | BIT2= 0: OFF; | BIT2= 1: ON |

**Module returns success:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0x11 | Checksum |

**Module returns failure:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 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:**

| | | | | |
|---|---|---|---|---|
| 0x0005 | C.A. | 0x12 | Random data | Checksum |

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

**Module returns success:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0x12 | Checksum |

**Module returns failure:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0xED | Checksum |

### 5.2.4 Set LED

**Function:** Set the LED ON or OFF.

**Host sends:**

| | | | | |
|---|---|---|---|---|
| 0x0005 | C.A. | 0x13 | State | Checksum |

State: 1 byte, 0: OFF, 1: ON, other value: RFU

**Module returns success:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0x13 | Checksum |

**Module returns failure:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0xEC | Checksum |

### 5.2.5 Set buzzer

**Function:** Set buzzer to beep.

**Host sends:**

| | | | | |
|---|---|---|---|---|
| 0x0005 | C.A. | 0x14 | Time | Checksum |

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

**Module returns success:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0x14 | Checksum |

**Module returns failure:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0xEB | Checksum |

### 5.2.6 EEPROM read

**Function:** Read data in EEPROM of the module.

**Host sends:**

| | | | | | |
|---|---|---|---|---|---|
| 0x0007 | C.A. | 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:**

| | | | | |
|---|---|---|---|---|
| - | C.A. | 0x15 | Data | Checksum |

Note: the byte length is “-”, means the byte length depends on the card feedback information.

(The same to below)

Data: data read

**Module returns failure:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0xEA | Checksum |

### 5.2.7 EEPROM write

**Function:** Write data into EEPROM of the module

**Host sends:**

| - | C.A. | 0x16 | Address | Bytes | Data | Checksum |
|---|---|---|---|---|---|---|

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

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

Data: “Bytes” data to write

**Module returns success:**

| 0x0004 | C.A. | 0x16 | Checksum |
|---|---|---|---|

**Module returns failure:**

| 0x0004 | C.A. | 0xE9 | Checksum |
|---|---|---|---|

### 5.2.8 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:**

| 0x0005 | C.A. | 0x17 | Baud rate | Checksum |
|---|---|---|---|---|

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

**Module returns success:**

| 0x0004 | C.A. | 0x17 | Checksum |
|---|---|---|---|

**Module returns failure:**

| 0x0004 | C.A. | 0xE8 | Checksum |
|---|---|---|---|

### 5.2.9 Set UART multi-device communication address

**Function:** Set UART multi-device communication address of the module. After module receive the command, it will save the new setting first, and then send the execute result to the host. At last it will validate the new setting. UART multi-device communication address is default 1. Settings will SAVE in the module; it will not be lost after power OFF.

**Host sends:**

| | | | | |
|---|---|---|---|---|
| 0x0005 | C.A. | 0x18 | Address | Checksum |

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

**Module returns success:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0x18 | Checksum |

**Module returns failure:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0xE7 | Checksum |

### 5.2.10 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. The module default address is 0xA0.

**Host sends:**

| | | | | |
|---|---|---|---|---|
| 0x0005 | C.A. | 0x19 | Address | Checksum |

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

**Module returns success:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0x19 | Checksum |

**Module returns failure:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0xE6 | Checksum |

### 5.2.11 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:**

| | | | | |
|---|---|---|---|---|
| 0x0005 | C.A. | 0x1A | Multi-card enable | Checksum |

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

**Module returns success:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0x1A | Checksum |

**Module returns failure:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0xE5 | Checksum |

### 5.2.12 Set automatic detect card interval time

**Function:** Set interval time between two detect card operation

**Host sends:**

| | | | | |
|---|---|---|---|---|
| 0x0005 | C.A. | 0x1C | Interval Time | Checksum |

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

**Module returns success:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0x1C | Checksum |

**Module returns failure:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0xE3 | Checksum |

### 5.2.13 Set the default automatic detect card state default

**Function:** Set the default automatic detects card state when boot device. For temporary open or close automatically detect card, please use the 0x11 command.

**Host sends:**

| | | | | |
|---|---|---|---|---|
| 0x0005 | C.A. | 0x1D | State | Checksum |

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

**Module returns success:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0x1D | Checksum |

**Module returns failure:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0xE2 | Checksum |

### 5.2.14 Set automatic detect card and output card UID default

**Function:** Set automatic detect card and output the card serial number when boot device. Under this mode, the card serial number can be output from serial port when swiping the card. The RF protocol is following ISO14443A and ISO15693. The output format is the same to 0x20 commands or 0x5C commands returned format. This command cannot be operated under IIC mode. While this command is on, then the read/write card cannot be operated because of the card entering into halt state once while the card is detected. If need to read/write card, automatic output the card serial number must be shut temporarily via 0x11 command and then go on with the read/write card operations.

**Host sends:**

| | | | | |
|---|---|---|---|---|
| 0x0005 | C.A. | 0x1E | State | Checksum |

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

**Module returns success:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0x1E | Checksum |

**Module returns failure:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0xE1 | Checksum |

### 5.2.15 ISO14443A request cards

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

**Host sends:**

| | | | | |
|---|---|---|---|---|
| 0x0005 | C.A. | 0x20 | Mode | Checksum |

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

**Module returns success:**

| | | | | |
|---|---|---|---|---|
| - | C.A. | 0x20 | Data | Checksum |

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

**Module returns failure:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0xDF | Checksum |

### 5.2.16 Mifare 1K/4K data block read

**Function:** Read Mifare 1K/4K data block

**Host sends:**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0x000C | C.A. | 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

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

BIT7=1: The block has been certified and passed. This operation no need certification (the operation and automatically detect the card cannot be used at the same time)

(IMPORTANT: please read Chapter 5.3 about Key identification)

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

Key: 6 bytes, the key of the card

**Module returns success:**

| | | | | |
|---|---|---|---|---|
| 0x0014 | C.A. | 0x21 | Data | Checksum |

Data: 16 bytes card data

**Module returns failure:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0xDE | Checksum |

### 5.2.17 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 reading will fail.

**Host sends:**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0x0000D | C.A. | 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 blocks to read(depend on structure of card,1-4 for S50)

Key: 6 bytes, the key of the card

**Module returns success:**

| | | | | |
|---|---|---|---|---|
| - | C.A. | 0x2A | Data | Checksum |

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

**Module returns failure:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0xD5 | Checksum |

### 5.2.18 Mifare 1K/4K data block write

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

**Host sends:**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0x001C | C.A. | 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:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0x22 | Checksum |

**Module returns failure:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0xDD | Checksum |

### 5.2.19 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:**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| - | C.A. | 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 blocks to write

Key: 6 bytes, the key of the card

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

**Module returns success:**

|  |  |  |  |
|---|---|---|---|
| 0x0004 | C.A. | 0x2B | Checksum |

**Module returns failure:**

|  |  |  |  |
|---|---|---|---|
| 0x0004 | C.A. | 0xD4 | Checksum |

### 5.2.20 Mifare 1K/4K purse block initialize

**Function:** Initialize a block of Mifare 1K/4K as 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:**

|  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|
| 0x0010 | C.A. | 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:**

|  |  |  |  |
|---|---|---|---|
| 0x0004 | C.A. | 0x23 | Checksum |

**Module returns failure:**

|  |  |  |  |
|---|---|---|---|
| 0x0004 | C.A. | 0xDC | Checksum |

### 5.2.21 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:**

|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
| 0x000C | C.A. | 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:**

| | | | | |
|---|---|---|---|---|
| 0x0008 | C.A. | 0x24 | Data | Checksum |

Data: 4 bytes value data, LSB first

**Module returns failure:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0xDB | Checksum |

### 5.2.22 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:**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0x0010 | C.A. | 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:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0x25 | Checksum |

**Module returns failure:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0xDA | Checksum |

### 5.2.23 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:**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0x0010 | C.A. | 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, decrement value, LSB first

**Module returns success:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0x26 | Checksum |

**Module returns failure:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0xD9 | Checksum |

### 5.2.24 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:**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0x000D | C.A. | 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:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0x27 | Checksum |

**Module returns failure:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0xD8 | Checksum |

### 5.2.25 ISO14443A card halt

**Function:** Set the current operating ISO14443A card in halt state.

**Host sends:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0x28 | Checksum |

**Module returns success:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0x28 | Checksum |

**Module returns failure:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0xD7 | Checksum |

### 5.2.26 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. While 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:**

| | | | | | |
|---|---|---|---|---|---|
| 0x000B | C.A. | 0x2D | Key Index | Key | Checksum |

Key Index: 1 byte, store the Key Index (0 --0x1F) in the module

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

**Module returns success:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0x2D | Checksum |

**Module returns failure:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0xD2 | Checksum |

### 5.2.27 ISO14443-4 TYPE-A card reset (RATS)

**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 the automatic detecting card function is on, after a successful implementation of the RATS command, the automatic detect card function will be forced to shut.

**Host sends:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0x30 | Checksum |

**Module returns success:**

| | | | | |
|---|---|---|---|---|
| - | C.A. | 0x30 | Info | Checksum |

Info: card reset information, length depends on card

**Module returns failure:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0xCF | Checksum |

### 5.2.28 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 detect card. That's because the ISO14443-4 card's state will be lost in automatic detecting card.

**Host sends:**

| | | | | |
|---|---|---|---|---|
| - | C.A. | 0x31 | APDU | Checksum |

APDU: APDU to send

**Module returns success:**

| | | | | |
|---|---|---|---|---|
| - | C.A. | 0x31 | Response | Checksum |

Response: card response, length depends on the detailed command

**Module returns failure:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0xCE | Checksum |

### 5.2.29 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 (0x0F), then these 4 blocks data are the 15th, 0th, 1st and 2nd block.

**Host sends:**

| | | | | |
|---|---|---|---|---|
| 0x0005 | C.A. | 0x41 | Read start block | Checksum |

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

**Module returns success:**

| | | | | |
|---|---|---|---|---|
| 0x0014 | C.A. | 0x41 | Data | Checksum |

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

**Module returns failure:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0xBE | Checksum |

### 5.2.30 Ultra Light card write

**Function:** Write data to Ultra Light card. Each for one block data

**Host sends:**

| | | | | | |
|---|---|---|---|---|---|
| 0x0009 | C.A. | 0x42 | Block | Data | Checksum |

Block: 1 byte, block number to write

Data: 4 bytes data to write

**Module returns success:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0x42 | Checksum |

**Module returns failure:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0xBD | Checksum |

### 5.2.31 SAM slot default baud rate set

**Function:** Before SAM card reset, to set default baud rate of the SAM slot. This baud rate will be used by the reader to reset the SAM. In ISO7816, the default baud rate for the card is 9600bps.

**Host sends:**

| | | | | |
|---|---|---|---|---|
| 0x0005 | C.A. | 0x50 | Baud rate | Checksum |

Baud rate: 1 byte, baud rate code of SAM, 0: 9600bps (default); 2: 38400bps; other value: RFU

**Module returns success:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0x50 | Checksum |

**Module returns failure:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0xAF | Checksum |

### 5.2.32 SAM reset

**Function:** Reset the SAM in the slot, get ATQ and set the relevant communication parameter.

**Host sends:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0x51 | Checksum |

**Module returns success:**

| | | | | |
|---|---|---|---|---|
| - | C.A. | 0x51 | Info | Checksum |

Info: reset info of SAM card, length depends on card

**Module returns failure:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0xAE | Checksum |

### 5.2.33 Set SAM baud rate after reset (through PPS)

**Function:** Some SAM support PPS instruction and then user could modify the communication baud rate.

**Host sends:**

| | | | | |
|---|---|---|---|---|
| 0x0005 | C.A. | 0x52 | Baud rate | Checksum |

Baud rate: 1 byte, baud rate code of SAM, 0: 9600bps; 2: 38400bps; other value: RFU

**Module returns success:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0x52 | Checksum |

**Module returns failure:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0xAD | Checksum |

### 5.2.34 Send APDU to SAM

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

**Host sends:**

| | | | | |
|---|---|---|---|---|
| - | C.A. | 0x53 | APDU | Checksum |

APDU: APDU need to send

**Module returns success:**

| | | | | |
|---|---|---|---|---|
| - | C.A. | 0x53 | Response | Checksum |

Response: response of SAM, length depends on detailed commands

**Module returns failure:**

| | | | |
|---|---|---|---|
| 0x0004 | C.A. | 0xAC | 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 |
| | | | | | | | |

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 be certified via the above key

BIT7=1: The block has been authenticated. Current operation does not need authentication (this operation and automatically detect card cannot 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) by using command 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. When power on, the default state is set via 0x1D command. This setting will keep on next power up. After power on, the automatic detect card function can be started or shut via 0x11 commands. The module after re-power will return to the set default state.

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

The CPU card can be detected when the automatic detect card function is working. If to operate CPU card, first to send RATS command (0x30). After the module has received correct RATS command, and then the automatic detect card function will be shut. In using, please note this.

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 detect card function is on.

## 5.5 Example of commands

### 5.5.1 About UART communication protocol

**For example:**

Read block 1: 000C00210001AABBCCDDEEFF3D
000C: package length; from 000C to FF are total 0x000C bytes
00: UART multi-device communication address
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
3D: 00 ^ 0C ^ 00 ^ 21 ^ 00 ^ 01 ^ AA ^ BB ^ CC ^ DD ^ EE ^ FF = 3D, in sample program, the function will calculate it.

### 5.5.2 UART commands sample

| | |
|---|---|
| Read block 1 | 000C00210001FFFFFFFFFFFF2C |
| Read block 255 (S70) | 000C002100FFFFFFFFFFFFFFD2 |
| Write block 1 | 001C00220001FFFFFFFFFFFF1234567890ABCDEF1234567890ABCDEF3F |
| Request card (WUPA) | 000500200025 |
| Halt card | 0004001216 |

### 5.5.3 IIC commands sample

| | |
|---|---|
| Read block 1 | 000C00210001FFFFFFFFFFFF2C |
| Read block 255 (S70) | 000C002100FFFFFFFFFFFFFFD2 |
| Write block 1 | 001C00220001FFFFFFFFFFFF1234567890ABCDEF1234567890ABCDEF3F |
| Request card (WUPA) | 000500200025 |
| Halt card | 0004001216 |

## 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.