Communication Protocol
Of RFID Contactless Module

Revision V1.30

Table of Contents

1 Introduction

Purpose

This document defines a communication protocol, which will be as a generic protocol for products involving data communication with each other. Basically this generic protocol serves for communication between a HOST and one or more terminal devices.

Scope

Different aspects of the protocol will be described, which include the electrical interface, data format, and link layer. This generic protocol will be applied for

- Point to point – RS232
- Multi-drop (Point to multi-points) – RS422/RS485
- Two wires half-duplex mode and four wires full-duplex mode.

Glossary

UID – Unique Identification

LRC – Longitudinal Redundancy Check

CRC – Cyclic Redundancy Check

MAC – Message Authentication Code

ATR – Answer To Reset

Referenced Document

<Not available>

2 Physical Layer

Electrical Interface

Basically, this communication protocol does not need to be bound with any electrical interface characteristic. Typically the following types of physical link could be used:

- RS232 (Point to point only)
- CMOS-Logic Level (Point to point only)
- Half duplex, two wires RS485/RS422 (multi-drop mode supported)
- Full duplex, four wires RS485/RS422 (multi-drop mode supported)

Data Format

The data format (Start Bit, Data Bits, parity, Stop Bit) is software configurable, and can be set to match the special requirement of data transmission between two communication devices. The general data format is defined as:

Parameter	Description
Baud Rate	Selective: 9600, 19200, 38400, 57600, 1152000 (It can be changed by command Send from the Host)
Data Bits	Fixed: 8 bits
Start Bit	Fixed: 1 Bits
Stop Bit	Selective: 1 bit.
Parity	Selective: Odd, Even, None

The following is the default setting:

Baud Rate	Data Bits	Start Bit	Stop Bit	Parity
9600	8	1	1	None

3 Link Layer

The communication protocol is a packet-oriented protocol - all the data exchanged between two communication devices will be based on packet format. The protocol is designed for multi-drop mode and where point-to-point mode could be treated as a special case of multi-drop mode.

The data packet starts with the control character 'STX' and ends with 'ETX', which follows the 8-bit BCC checksum. Besides the checksum is used for error checking, character (byte) time-out and packet (command) time-out are used to re-synchronous the communication.

Packet Format

There are two types of data packets. Command Message is the packet Send from the Host to the reader device. The Reply Message is the packet Send from the reader to the Host.

Packet format for Command Message (Host to Reader)

STX	STATION ID	DATA LENGTH	CMD	DATA[0..N]	BCC	ETX

(BCC) = STATION ID ⊕DATALENGTH⊕ CMD⊕ DATA[0] ⊕ … ⊕ DATA[n], where ⊕ is the "EOR".

Packet format for Reply Message (Reader to Host)

STX	STATION ID	DATA LENGTH	STATUS	DATA[0..N]	BCC	ETX

(BCC) = STATION ID ⊕DATA LENGTH⊕ STATUS⊕ DATA[0] ⊕ … ⊕ DATA[n], where ⊕ is the "EOR".

The following table describes the packet fields:

Field	Length	Description	Remark
STX	1	0Xaa- "Start of Text' .It is the starting of a data packet.	
DADD	1	Device Address, which is used for multi-drop mode, only the reader (device) with matched pre-programmed device address will response the received command packet.	Address 0x00 is a special address for point-to-point mode communication. The reader responds to all the packets which has a "0" address. (No Address matching checking will be made.
DATA LENGTH	1	Length of the data bytes in the packet. LENGTH= Number_of_Bytes (TIME/STATUS + DATA[0..N])	The Data Length includes the TIME/STATUS and the DATA field, but not the BCC.
CMD	1	Command field: the command field consists of one command byte.	Refer the Command Table for listing of commands.
STATUS	1	Reply Status byte: The status replied from Reader to Host	This byte is only used for the Reply Packet.

DATA [0-N]	0 – 255	The Data Field is a stream of data with variable length, which depends on the Command word. There are also some COMMANDs have zero length of data field. If the Data Field of the Command/Reply Message has more then 80 bytes, the reader won't response and treats this command as an error and wait for another command.	
BCC	1	Eight-bit block check sum. The calculation of the check sum includes all the bytes within the package but excludes the STX, ETX.	
ETX	1	0xBB:'END of TEXT' –Which indicates the END of a packet.	

4 Command Set

The commands are grouped to different categories. They are System command, ISO14443A standard commands, ISO14443B standard commands ,MIFARE commands and ISO15693A standard commands.

ISO14443 TYPE A Commands (0x**03**~0x**06**)		
0x03	ReqA	ISO14443A Request Command
0x04	AnticollA	ISO14443A Anti-collision
0x05	SelectA	ISO14443A Select
0x06	HaltA	ISO14443A Halt
ISO14443-B Command (0x0**9**-0x0E)		
0x09	Request_B	ISO14443B REQB Command
0x0A	AnticollB	ISO14443B Anti-collision
0x0B	Attrib_B	ISO14443B ATTRIB Command
0x0C	Rst_ TypeB	Integrate the REQB and ATTRIB Command
0x0D	ISO14443_TypeB_Transfer_Command	ISO14443-4 transparent command Type B Card
Mifare **Application** Commands (0x**2**0~0x**2**F)		
0x20	MF_Read	The Read command integrates the low level commands (request, anti-collision, select, authentication, read) to achieve the reading operation with a one-step single command.
0x21	MF _Write	The Write command integrates the low level commands (request, anti-collision, select, authentication, write) to achieve the writing operation with a one-step single command.
0x22	MF _ InitVal	The Initialization command integrates the low level commands (request, anti-collision, select, authentication) to achieve the value block initialization with a one-step single command.
0x23	MF _Decrement	The Decrement command integrates the low level commands (request, anti-collision, select, authentication) to achieve the Decrement with a one-step single command.

0x24	MF _Increment	The Increment command integrates the low level commands (request, anti-collision, select, authentication) to achieve the Increment with a one-step single command.
0x25	MF _GET_SNR	The GetSnr command integrates the low level commands (request,anticoll,select) to achieve the select card with a one-step single command,and output the card's Snr
0x28	ISO14443_TypeA_Transfer_Command	Using this command you may transparent any command to The Card which these commands meet the ISO14443-TypeA protocol
System command (0x**8**0~0x**8**F)		
0x80	SetAddress	Program the Device Address to the reader (The range of address is 0~255)
0x81	SetBaudrate	Set the reader's communication baud rate(9600~115200)
0x82	SetSerlNum	Set the reader's Serial Number(The Seial Number is 8 byte)
0x83	GetSerlNum	Get the reader's Serial Number And Address
0x84	Write_UserInfo	Set the Usr Information
0x85	Read_UserInfo	Get the Usr Information
0X86	Get_VersionNum	Get the reader's firmware version number
0x87	Control_Led1	Turn On/Off the LED1(This Command is only supported by the module when The Module have two led,or The Module only support the "Control_Led2" command.)
0x88	Control_Led2	Turn On/Off the LED2
0x89	Control_Buzzer	Turn On/Off the Buzzer
ISO15693 Commands (0x10~0x1D)		
0x10	ISO15693_Inventory	ISO15693 Inventory Command
0x11	ISO15693_Read	ISO15693 Read Command
0x12	ISO15693_Write	ISO15693 Write Command
0x13	ISO15693_Lockblock	ISO15693 Lock_Block Command
0x14	ISO15693_StayQuiet	ISO15693 Stay_Quiet Command

0x15	ISO1569_Select	ISO15693_Select Command
0x16	ISO15693_Resetready	ISO15693_Reset_To_Ready Command
0x17	ISO15693_Write_Afi	ISO15693_Write_AFI Command
0x18	ISO15693_Lock_Afi	ISO15693_Lock_AFI Command
0x19	ISO15693_Write_Dsfid	ISO15693_Write_DSFID Command
0x1A	ISO15693_Lock_Dsfid	ISO15693_Lock_DSFID Command
0x1B	ISO15693_Get_Information	ISO15693_Get_System_Information Command
0x1C	ISO15693_Get_Multiple_Block_Security	ISO15693_Get_Multiple_Block_Security Command
0x1D	15693_Transfer_Command	Using this command may transparent any command to The Card which command meet the ISO15693 protocol

5 System Commands

5.1.1 SetAddress (0x80)

Data Field

DATA[0]: The new Address of the reader to be set

Response:

STATUS: 0x00 – OK

Data Field

DATA[0] The programmed device address.

Description

Program a device address to the reader and returns new device address.

EXAMPLE:

Send Data: AA 00 02 80 02 80 BB

Response Data: AA 00 02 00 02 00 BB

5.1.2 SetBaudrate (0x81)

Data Field

DATA[0] Communication speed

0x00 – 9600 bps
0x01 – 19200 bps
0x02 – 38400 bps
0x03 – 57600 bps
0x04 – 115200 bps

Response:

STATUS: 0x00 - OK

Data Field

DATA[0] Return the new communication speed Code.

0x00 – 9600 bps
0x01 – 19200 bps
0x02 – 38400 bps
0x03 – 57600 bps
0x04 – 115200 bps

Description

Set the reader's baud rate for host communication. The baud rate will be stored in the reader's EEPROM and used as the new default baud rate.The new baud rate can be used at once, which need not the reader reset.

EXAMPLE:

Send Data: **AA 00 02 81 01 82 BB**

Response Data: : AA 00 02 00 **01** **03** BB (19200,N,8,1)

5.1.3 SetSerNum (0x82)

Data Field

DATA[2~9]: 8 Byte reader's snr

Response:

STATUS: 0x00 – OK

Data Field

DATA[0]: 0x80

Description

Set the Serial Number from the reader.

EXAMPLE:

Send Data: AA 00 09 82 AA BB AA BB AA BB AA BB 89 BB

Response Data: AA 00 02 00 80 82 BB

5.1.4 GetSerNum (0x83)

Data Field N/A

Response:

STATUS: 0x00 – OK

Data Field

DATA[0]: Device Address

DATA[1..9]: 8 Byte reader's snr

Description

Get the Serial Number from the reader.

EXAMPLE:

Send Data: AA 00 01 83 82 BB

Response Data: **AA 00 0A 00** 00 **AA BB AA BB AA BB AA BB 0A BB**

THE "00**" is the address of the module, and the following 8 bytes is the snr of the module.**

5.1.5 SetUserInfo (0x84)

Data Field

DATA[0]: the number of the data area

0x00: Write the data to 0 data area

0x01: Write the data to 1 data area

0x02: Write the data to 2 data area

0x03: Write the data to 3 data area

DATA[1]: the length of the data(it will be written to the reader's data area),and the length must less than 120(0x78).

DATA[2..n] data

Response:

STATUS: 0x00 – OK

Data Field 0x80

Description

The Modlue provide the four data blocks to user. Each block have 120 bytes space .

EXAMPLE:

Send Data: **AA 00 7B 84 01 78** AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 **86 BB**

Response Data: AA 00 02 00 80 82 BB

5.1.6 GetUserInfo (0x85)

Data Field

DATA[0]: the number of the data area

0x00: Get the data from 0 data area

0x01: Get the data from 1 data area

0x02: Get the data from 2 data area

0x03: Get the data from 3 data area

DATA[1]: The length of data

Response:

STATUS: 0x00 – OK

Data Field

DATA{0..n} the returned data

Description

Get user information from the module.

EXAMPLE:

Send Data: AA 00 03 85 01 78 FF BB

Response Data: AA 00 79 00 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 79 BB

5.1.7 Get_VersionNum(0x86)

Data Field: N/A

Response

STATUS : 0x00 –OK

Data Field 6 or more bytes the module version nubmer

EXAMPLE:

Send Data: AA 00 01 86 87 BB

Response Date: AA 00 07 00 52 44 4D 38 31 30 65 BB

5.1.8 Control_Led1 (0x87)

Data Field

DATA[0]: Units of on time. Each unit is 20ms.So the data[0] is less than 50

DATA[1]: Number of cycles to turn on/off the LED. The cycle time is one second.

Response:

STATUS: 0x00 - OK

Data Field: N/A

DATA[0]: 0x80

Description:

Turn on/off the LEDs. This Command is only supported by the module when The Module have two led, or The Module only support the “Control_Led2” command

EXAMPLE:

Send Data： AA 00 03 87 18 0A 96 BB

Response Data： AA 00 02 00 80 82 BB

The“18” is the number of led1 on time. The on time equal to 480ms(20ms * 24)

The“0A” is the number of the cycles to turn on/off the led

5.1.9 Control_Led2(0x88)

Data Field

DATA[0]: Units of on time. Each unit is 20ms. So the data[0] is less than 50

DATA[1]: Number of cycles to turn on/of the LED. The cycle time is one second.

Response:

STATUS: 0x00 - OK

Data Field: N/A

DATA[0]: 0x80

Description:

Turn on/off the LEDs.

EXAMPLE:

Send Data： AA 00 03 88 18 0A 99 BB

Response Data： AA 00 02 00 80 82 BB

The“18”is the number of led2 on time. The on time equal to 480ms(20ms * 24)

The“0A” is the numbers of the cycles to turn on/off the led

5.1.10 SetBuzzer (0x89)

Data Field

DATA[0]: Units of on time. Each unit is 20ms. So the data[0] is less than 50

DATA[1]: Number of cycles to turn on/of the LED. The cycle time is one second.

Response:

STATUS: 0x00 - OK

Data Field: N/A

DATA[0]: 0x80

Description:

Turn on/off the Buzzer.

EXAMPLE:

Send Data: AA 00 03 89 18 0A 98 BB

Response Data: AA 00 02 00 80 82 BB

6 ISO14443 Type-A Commands

6.1.1 REQA (0x03)

Data Field

DATA[0]: Request mode

0x26 – Request Idle

0x52 – Request All (Wake up all)

Response:

STATUS: 0x00 - OK

DATA[0..1]: The two-bytes ATQ response from the card.

Description

Send the ISO14443 A REQUEST command to the card.

EXAMPLE:

Send Data： **AA 00 02 03 26 27 BB**

Response Data： AA 00 03 00 04 00 07 BB

6.1.2 AnticollA (0x04)

Data Field: N/A

Response:

STATUS: 0x00 - OK

Data Field

DATA[0]: Multi-card flag.

0x26 - One cared detected.

0x52 - Multiple cards detected.

DATA[1..4]: UID – the card serial number

Description:

Execute the ISO14443 Type A Anti-collision loop of cascadelevel1. The card's UID (serial number) of cascadelevel1 will be returned. If more then one cards are detected in the field, the Multi-Card Flag will be set.

EXAMPLE:

Send Data： AA 00 01 04 05 BB

Response Data： AA 00 06 00 00 06 61 62 AE AD BB

When there are two or more cards in the readable area:

Send Data： AA 00 01 04 05 BB

Response Data：AA 00 06 00 01 06 61 62 AE AC BB

Here the "01" means there are two or more cards in the readable area,and the following 4 bytes is the snr of the card.

6.1.3 SelectA(0x05)

Data Field

DATA[0..3]: UID – the UID of the card to be selected.

Response:

STATUS: 0x00 - OK

Data Field

DATA[0..3]: UID – the UID of the card to be selected.

Description:

ISO14443 A SELECT of Cascadelevel1 command.

EXAMPLE:

Send Data： AA 00 05 05 86 69 F3 7F 63 BB

Response Data：AA 00 05 00 86 69 F3 7F 66 BB

6.1.4 HaltA (0x06)

Data Field: N/A

Response:

STATUS : 0x00 - OK

Data Field

DATA[0] 0X80.

Description:

ISO14443 A Halt command.

EXAMPLE:

Send Data： AA 00 01 06 07 BB

Response Data：AA 00 02 00 80 82 BB

7 MIFARE Application Commands

7.1.1 MF_Read (0x20)

Data Field

DATA[0]: Mode Control

Bit0 : Request Mode. 0=Request Idle, 1 = Request All

Bit1 : Key Select. Select use KeyA or Key B for Authenticaiton

0=KeyA, 1=KeyB

DATA[1]: Number of blocks to be read (Max 4)

DATA[2]: The Start Address of blocks to be read(the range is 0~63).

DATA[3-8]: The six bytes block key

Response:

Data Field

STATUS: 0x00 – OK

DATA[0-3]: Card Serial Number (LL LH HL HH)

DATA[4..N] Data read from the card.

Description:

The Read Command integrates the low level commands (Request, Anti-Collision, Select, Authentication) and let the user to select the card and read data from the memory blocks by a single command.

EXAMPLE:

Send Data: AA 00 0A 20 01 01 10 ff ff ff ff ff ff 3A BB(Read the data from the 16th block to 19th block)

The"01"means that the request mode is "Request all" and use the keyA For authentication

The "01"means that only read one block contents

The "10"is the start address of the block

The"ff ff ff ff ff ff"is the key

Response Data: AA 00 45 00 16 0F F4 7F

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF

C6 BB

7.1.2 MF_Write (0x21)

Data Field

DATA[0]: Mode Control

Bit0 : Request Mode. 0=Request Idle, 1 = Request All

Bit1 : Key Select. Select use KeyA or Key B for Authenticaiton

0=KeyA, 1=KeyB

DATA[1]: Number of blocks to be write (Max 4)

DATA[2]: The Start Address of blocks to be write.(the value's range is 0~63)

DATA[3-8]: The six bytes block key

Response:

Data Field

STATUS: 0x00 – OK

DATA[0-3]: Card Serial Number (LL LH HL HH)

Description:

The Write Command integrates the low level commands (Request, Anti-Collision, Select, Authentication) and let the user to select the card and write data to the memory blocks by a single command.

EXAMPLE:

Send Data:

AA 00 1A 21 01 01 10 ff ff ff ff ff ff **FF FF FF FF FF FF FF FF FF FF FF FF FF FF 11 11 2B BB (write 16 bytes data to the 16th block of the card**)

The"01"means that the request mode is "Request all" and use the keyA For authentication

The "01"means that only read one block contents

The "10"is the start address of the block

The"ff ff ff ff ff ff"is the key

The "FF FF FF FF FF FF FF FF FF FF FF FF FF FF 11 11" is the datas that will be writen

Response Data: AA 00 05 00 CE 86 AE 67 84 BB

7.1.3 MF_InitVal (0x22)

Data Field

DATA[0]: Mode Control

Bit0 : Request Mode. 0=Request Idle, 1 = Request All

Bit1 : Request Mode. 0=KEYA 1 = KeyB

DATA[1]: The Sector used for Value storage.

Block0 –Opened for user use.

Block1 –Value Stored Block

Block2 –Value Backup Block.

DATA[2-7]: KEY (SIX BYTES)

DATA[8-11]: The initial value to be stored to the value block. (Value format : LL LH HL HH)

Response:

Data Field

STATUS: 0x00 – OK

DATA[0-3]: Card Serial Number (LL LH HL HH)

Description:

The High Level Value Initialization Command integrates the low level commands (Request, Anti-Collision, Select, Authentication,) and let the user to initialize a sector for value storage use.

EXAMPLE:

Send Data:

AA 00 0D 22 01 04 ff ff ff ff ff ff 64 00 00 00 **4E bb (Initval with the 4TH Sector)**

The"01"means that the request mode is "Request all" and use the keyA For authentication

The"04"is the numbers of the sector.

The "ff ff ff ff ff ff"is six bytes key.

The "64 00 00 00"is the value that will be initval

Response Data: AA 00 05 00 16 0F F4 7F 97 BB

7.1.4 MF_Decrement (0x23)

Data Field

DATA[0]: Mode Control

Bit0 : Request Mode. 0=Request Idle, 1 = Request All

Bit1 : Request Mode. 0=KEYA 1 = KeyB

DATA[1]: The Sector Number of the Value Sector.

DATA[2-7]: the six bytes block key

DATA[8-11]: The value to be decreased to the value block. (Value format : LL LH HL HH)

Response:

Data Field

STATUS: 0x00 – OK

DATA[0-3]: Card Serial Number (LL LH HL HH)

DATA[4-7]: Value after decreased (LL LH HL HH)

Description:

The High Level Value Decrement Command integrates the low level commands (Request, Anti-Collision, Select, Authentication, ..) and let the user to decrease the selected value.

EXAMPLE:

Send Data:

AA 00 0d 23 01 04 ff ff ff ff ff ff **01** 00 00 00 **2A BB** (**Decrement with the 4TH Sector**)

The"01"means that the request mode is "Request all" and use the keyA For authentication

The"04"is the numbers of the sector.

The "ff ff ff ff ff ff"is six bytes key.

The "01 00 00 00"is the value that will be decreased

Response Data: AA 00 09 00 16 0F F4 7F 63 00 00 00 F8 BB

The "16 0F F4 7F"is the card's snr

The "63 00 00 00"is Value after decreased

7.1.5 MF_Increment (0x24)

Data Field

DATA[0]: Mode Control

Bit0 : Request Mode. 0=Request Idle, 1 = Request All

Bit1 : Request Mode. 0=KEYA 1 = KeyB

DATA[1]: The Sector Number of the Value Sector.

DATA[2-7]: the six bytes block key

DATA[8-11]: The value to be increased to the value block. (Value format : LL LH HL HH)

Response:

Data Field

STATUS: 0x00 – OK

DATA[0-3]: Card Serial Number (LL LH HL HH)

DATA[4-7]: Value after Increased (LL LH HL HH)

Description:

The High Level Value increment Command integrates the low level commands (Request, Anti-Collision, Select, Authentication) and let the user to decrease the selected value.

EXAMPLE:

Send Data:

AA 00 0d 24 01 04 ff ff ff ff ff ff **01** 00 00 00 **2D BB** (**Increment with the 4TH Sector**)

The"01"means that the request mode is "Request all" and use the keyA For authentication

The"04"is the numbers of the sector.

The "ff ff ff ff ff ff"is six bytes key.

The "01 00 00 00**"is the value that will be increased**

Response Data: AA 00 09 00 16 0F F4 7F 63 00 00 00 F8 BB

The "16 0F F4 7F**"is the card's snr**

The "63 00 00 00**"is Value after increased**

7.1.6 MF_GET_SNR (0x25)

Data Field

DATA[0]: Request mode

0x26 – Request Idle

0x52 – Request All

DATA[1]: 00 do not need to execute the halt command

01 need to execute the halt command

Response:

Data Field

STATUS: 0x00 – OK

DATA[0]: FLAG

0x00 – Only one card is in the readable area

0x01 – At least two cards are in the readable area

DATA[1-4]: Card Serial Number

Description:

The High Level Command integrates the low level commands (Request, AntiColl1, Select) and get the SNR of selected card.

EXAMPLE:

Send Data: AA 00 03 25 26 00 00 BB

Response Data: AA 02 06 00 00 16 0F F4 7F 96 BB

7.1.7 ISO14443_TypeA_Transfer_Command(0X28)

Data Field

DATA[0]: CRC Flag

0x00: donot need to tranfer crc data to the card

0x01:

DATA[1]: The length of the data which send to the card

DATA[2…N]: DATA

Response:

Data Field

STATUS: 0x00 – OK

DATA[0~N]: The data that response by the card

Description:

This command is using for transparent any command to The Card which these commands meet the ISO14443-Typea protocol .

EXAMPLE:

Send Data: AA 00 08 28 00 05 00 84 00 00 08 a9 bb (ISO14443 APDU Command)

Response Data: AA 00 0B 00 69 60 B3 AE C8 2A 8A 7E 90 00 99 BB

8 ISO14443 Type-B Commands

8.1.1 ReqB (0x09)

Data Field

DATA[0]: The AFI (Application Family Identifier). Only cards with the matched AFI may answer to the REQB command. When AFI equals “00”, all the card shall process the REQB command

DARA[1] The PARAM of Request B command. This parameter defines the SLOT number (the probability of response). Please refer the ISO14443 –Part 3. (Chapter 7.74 – Coding of PARAM) for details.

Response:

STATUS: 0x00 - OK

DATA[0]: Length of the returned ATQB string. For a successful REQB command, normally a 14 bytes ATQB string will be returned.

DATA[1..N] The returned ATQB string from the card.

Example:

Send data: AA00 01 09 08 bb

Response: AA 00 0E 00 0C 50 41 30 0A 10 41 F5 A3 44 00 71 85 9E BB

8.1.2 Anticoll_B (0x0A)

Data Field: N/A

Response:

STATUS: 0x00 - OK

Data Field

DATA[0] Multi-card Flag

0x00 – Only one card detected within the field.

0x01- More then one card detected within the field.

Note : only the cards which are not in “HALT” state could be detected.

DATA[1..14] 14 bytes ATQB string

Description:

Run the anticollison loop and pick one TYPE B card. The ATQB string of the selected card will be returned. The multi-card flag will be set in case more than one card is found within the field

Example:

Send data: AA 00 01 0A 0B bb

Response: AA 00 02 00 80 82 BB

8.1.3 Attrib_B (0x0B)

Data Field

DATA[0..3]: UID – the UID (Card Serial Number) of the card to be select.

Response:

STATUS: 0x00 - OK

Data[0] 0x80

Description:

The simplified ISO14443 B ATTRIB command. The CID will be assigned to the selected card for further communication. Only the UID and CID are needed as parameters, the other parameters (such as param1 to 3 and the Higher Layer IINF) defined in the ISO14443-3 chapter 7.10.1 are ignored.

Example:

Send data: AA 00 05 0B 41 30 0A 10 65 bb

Response: AA 00 02 00 80 82 BB

8.1.4 Rst_ TypeB (0x0C)

Data Field: N/A

Response:

STATUS : 0x00 - OK

Data Field

DATA[0] The data length of reponsing from the card

DATA[2.5] The card's snr

Example:

Send data: AA 00 01 0c 0d bb

Response: AA 00 05 00 41 30 0A 10 6E BB

8.1.5 ISO14443_TypeB_Transfer_Command (0x0D)

Data Field：

DATA[0]: The length of the data which send to the card

DATA[1…N]: DATA

Response:

STATUS : 0x00 - OK

Data Field

DATA[0~N] the data response from the card

Example：

Send data：**AA 00 07 0d 05 00 84 00 00 08 83 BB**（**Get Random Data**）

Response：**AA 00 0D 00 0A 00 69 60 B3 AE C8 2A 8A 7E 90 00 95 BB**

9 ISO15693 COMMANDS

9.1.1 ISO15693_Inventory (0x10)

Data Field

DATA[0]: Flags

Bit0: Sub_carrier_flag
Bit1: Date_rate_flag
Bit2: Inventory_flag
Bit3: Protocol Extension_flag
Bit4: Afi_flag
Bit5: nb_slots_flag
Bit6: Option_flag
Bit7: RFU

DATA[1]: Afi

DATA[2]: Masklengh

DATA[3..10]: Maskvalue

Response:

STATUS: 0x00 - OK

Data[0] : The card's number that exist in the reading area

Data[1..n] : UID

Description:

Run the anticollison loop. through this command you can get the UID of all the VICC in the readable zone.(usually it may get 3 to 6 card's snr,it base on the strength of the RF power and the card)

Example:

Send Data: AA 00 04 10 06 00 00 12 bb

Response Data: AA 00 0B 00 01 00 01 4A 80 E9 11 00 00 07 3E BB

The "01" means that there is one card in the readable area, the" 00 01" are the FLAG and DSFID that response from the card, the "4A 80 E9 11 00 00 07 E0" is the snr of the card.

Two cards in the readable area:

Response Data: AA 00 16 00 02 00 08 47 80 E9 11 00 00 07 E0 00 01 4A 80 E9 11 00 00 07 E0 10 BB

Three cards in the readable area:

Response Data: AA 00 20 00 03 00 08 47 80 E9 11 00 00 07 E0 00 01 4A 80 E9 11 00 00 07 E0 00 00 3B 80 E9 11 00 00 07 E0 83 BB

Four cards in the readable area:

Response Data: AA 00 2A 00 04 00 08 47 80 E9 11 00 00 07 E0 00 01 4A 80 E9 11 00 00 07 E0 00 00 3B 80 E9 11 00 00 07 E0 00 08 3E 80 E9 11 00 00 07 E0 27 BB

No card in the readable area: AA 00 02 01 83 80 BB

9.1.2 ISO15693_Stay_Quiet (0x14)

DATA[0]: Flags

Bit0: Sub_carrier_flag
Bit1: Date_rate_flag
Bit2: Inventory_flag
Bit3: Protocol Extension_flag
Bit4: Select_flag
Bit5: Address_flag
Bit6: Option_flag
Bit7: RFU

DATA[1..8]: UID

Response:

STATUS: 0x00 – OK

Note : The Stay quiet command shall always be executed in Addressed mode (Select_flag is set to 0 and Adddress_flag is set to 1).

Example:

Send Data: AA 00 0A 14 02 3E 80 E9 11 00 00 07 E014 bb

Response Data: AA 00 02 01 80 83 BB

No card or occur some mistakes: AA 00 02 01 83 80 BB

9.1.3 ISO15693_Read (0x11)

Data Field

DATA[0]: Flags

Bit0: Sub_carrier_flag
Bit1: Date_rate_flag
Bit2: Inventory_flag
Bit3: Protocol Extension_flag
Bit4: Select_flag
Bit5: Address_flag
Bit6: Option_flag
Bit7: RFU

DATA[1] First block number

DATA[2] Number of blocks

DATA[3..10] UID(if you set the Address_flag to 1， you must input the UID)

Response:

STATUS : 0x00 - OK

Data Field

DATA[0] Flags

DATA[1..N] DATA

Description:

You can read one or many block data with this command.

NOTE: In this command , when the Option_flag set to 1, then every block response five byte data and the first data means the Block security status, following four byte are the data of the block, and it can read 51 blocks at best one time . And oppositely if the Option_flag is set to 0, every block only response four byte date,and it can read 63 blocks at best one time.

Example:

Send Data: AA 00 04 11 02 01 05 13 BB (read from the 1th block to the 5th block)

Reponse data: AA 00 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 16 BB

No card or have some mistake: AA 00 02 01 83 80 BB

9.1.4 ISO15693_Write (0x12)

DATA[0]: Flags

Bit0: Sub_carrier_flag
Bit1: Date_rate_flag
Bit2: Inventory_flag
Bit3: Protocol Extension_flag
Bit4: Select_flag
Bit5: Address_flag
Bit6: Option_flag
Bit7: RFU

DATA[1] First block number

DATA[2] Number of blocks

DATA[3..10] UID(if you set the Address_flag to 1，you must input the UID)

DATA[11..N] The data need be written

Response:

STATUS : 0x00 - OK

NOTE: At present, the mostly card cannot write multiple block at one time, such as the TI-tag and I-code2 tag. They only may write one block at one time.

Example:

Send Data: AA 00 08 12 42 05 01 11 11 11 11 5c bb

Response Data: AA 00 02 00 80 82 BB

No card or have some mistake: AA 00 02 01 83 80 BB

9.1.5 ISO15693_Lock_Block (0x13)

DATA[0]: Flags

Bit0: Sub_carrier_flag
Bit1: Date_rate_flag
Bit2: Inventory_flag
Bit3: Protocol Extension_flag
Bit4: Select_flag
Bit5: Address_flag
Bit6: Option_flag
Bit7: RFU

DATA[1]: Block number

DATA[2..9] UID(if you set the Address_flag to 1，you must input the UID)

Response:

STATUS : 0x00 - OK

Description: When receiving the Lock block command, the VICC shall lock permanently the requested block.

Example：

Send Data： AA 00 03 13 42 05 57 bb

Response Data： AA 00 02 00 80 82 BB

No card or have some mistake： AA 00 02 01 83 80 BB

9.1.6 ISO15693_Select (0x15)

DATA[0]: Flags

Bit0: Sub_carrier_flag
Bit1: Date_rate_flag
Bit2: Inventory_flag
Bit3: Protocol Extension_flag
Bit4: Select_flag
Bit5: Address_flag
Bit6: Option_flag
Bit7: RFU

DATA[1..8] UID

Response:

STATUS : 0x00 - OK

Description: if the UID is equal to its own UID, the VICC shall enter the selected state and shall send a response. if it is different, the VICC shall return to the Ready state and shall not send a response.
NOTE: The Select command shall always be executed in Addressed mode. (The Select_flag is set to 0. The Address_flag is set to 1.)

Example：

Send Data： AA 00 0a 15 22 3E 80 E9 11 00 00 07 E0 9c bb

Response Data： AA 00 02 00 80 82 BB

No card or have some mistake： AA 00 02 01 83 80 BB

9.1.7 ISO15693_Reset_To_Ready(0x16)

DATA[0]: Flags

Bit0: Sub_carrier_flag
Bit1: Date_rate_flag
Bit2: Inventory_flag
Bit3: Protocol Extension_flag
Bit4: Select_flag
Bit5: Address_flag
Bit6: Option_flag
Bit7: RFU

DATA[1..8] UID(if you set the Address_flag to 1，you must input the UID)

Response:

STATUS : 0x00 - OK

Description: When receiving a Reset to ready command, the VICC shall return to the Ready state.
NOTE: When you want to turn a vicc from Seleted state to ready state., you muse set the Select _flag to 1.

Example:

Send Data: AA 00 0A 16 02 3E 80 E9 11 00 00 07 E0 16 bb

Response Data: AA 00 02 00 80 82 BB

No card or have some mistake: AA 00 02 01 83 80 BB

9.1.8 ISO15693_Write_AFI(0x17)

DATA[0]: Flags

Bit0: Sub_carrier_flag
Bit1: Date_rate_flag
Bit2: Inventory_flag
Bit3: Protocol Extension_flag
Bit4: Select_flag
Bit5: Address_flag
Bit6: Option_flag
Bit7: RFU

DATA[1]: AFI

DATA[2..9] UID(if you set the Address_flag to 1，you must input the UID)

Response:

STATUS : 0x00 - OK

Description: When receiving the Write AFI request, the VICC shall write the AFI value into its memory.

Example:

Send Data: AA 00 03 17 42 06 50 bb

Response Data: AA 00 02 00 80 82 BB

No card or have some mistake: AA 00 02 01 83 80 BB

9.1.9 ISO15693_Lock_AFI(0x18)

DATA[0]: Flags

Bit0: Sub_carrier_flag
Bit1: Date_rate_flag
Bit2: Inventory_flag
Bit3: Protocol Extension_flag
Bit4: Select_flag
Bit5: Address_flag
Bit6: Option_flag
Bit7: RFU

DATA[1..8] UID(if you set the Address_flag to 1, you must input the UID)

Response:

STATUS : 0x00 - OK

Description: When receiving the Lock AFI request, the VICC shall lock the AFI value permanently into its memory.

Example:

Send Data: AA 00 02 18 42 58 bb

Response Data: AA 00 02 00 80 82 BB

No card or have some mistake: AA 00 02 01 83 80 BB

9.1.10 ISO15693_Write_DSFID(0x19)

DATA[0]: Flags

Bit0: Sub_carrier_flag
Bit1: Date_rate_flag
Bit2: Inventory_flag
Bit3: Protocol Extension_flag
Bit4: Select_flag
Bit5: Address_flag
Bit6: Option_flag
Bit7: RFU

DATA[1]: DSFID

DATA[2..9] UID(if you set the Address_flag to 1, you must input the UID)

Response:

STATUS : 0x00 - OK

Description: When receiving the Write DSFID request, the VICC shall write the DSFID value into its memory.

Example:

Send Data: AA 00 03 19 42 08 50 bb

Response Data: AA 00 02 00 80 82 BB

No card or have some mistake: AA 00 02 01 83 80 BB

9.1.11 ISO15693_Lock_DSFID(0x1A)

DATA[0]: Flags

Bit0: Sub_carrier_flag
Bit1: Date_rate_flag
Bit2: Inventory_flag
Bit3: Protocol Extension_flag
Bit4: Select_flag
Bit5: Address_flag
Bit6: Option_flag
Bit7: RFU

DATA[1..8] UID(if you set the Address_flag to 1, you must input the UID)

Response:

STATUS : 0x00 - OK

Description: When receiving the Lock DSFID request, the VICC shall lock the DSFID value permanently into its memory.

Example:

Send Data: AA 00 02 1a 42 5a bb

Response Data: AA 00 02 00 80 82 BB

No card or Have some mistake: AA 00 02 01 83 80 BB

9.1.12 ISO15693_GET_System_Information(0x1B)

DATA[0]: Flags

Bit0: Sub_carrier_flag
Bit1: Date_rate_flag
Bit2: Inventory_flag
Bit3: Protocol Extension_flag
Bit4: Select_flag
Bit5: Address_flag
Bit6: Option_flag
Bit7: RFU

DATA[1..8] UID(if you set the Address_flag to 1, you must input the UID)

Response:

STATUS : 0x00 – OK

Data[0] : Flags

Data[1] : INFO Flags

Data[2..9] : UID

Data[10] : DSFID

Data[11] : AFI

Data[12..N]: Other fields

Description: This command allows for retrieving the system information value from the VICC. You can consult the ISO15693 Protocol to find out the means of the parameter response from the VICC.

Example:

Send Data: AA 00 02 1b 02 1b bb

Response Data: AA 00 10 00 00 0F 4A 80 E9 11 00 00 07 E0 01 01 3F 03 88 7E BB

No card or Have some mistake: AA 00 02 01 83 80 BB

9.1.13 ISO15693_Get_Multiple_Block_Security(0x1C)

DATA[0]: Flags

Bit0: Sub_carrier_flag
Bit1: Date_rate_flag
Bit2: Inventory_flag
Bit3: Protocol Extension_flag
Bit4: Select_flag
Bit5: Address_flag
Bit6: Option_flag
Bit7: RFU

DATA[1] First block number

DATA[2] Number of blocks

DATA[3..10] UID(if you set the Address_flag to 1，you must input the UID)

Response:

STATUS : 0x00 – OK

Data[0] : Flags

Data[1..N] : Block security status

Description: When receiving the Get multiple block security status command, the VICC shall send back the block security status.

Example:

Send Data: aa 00 04 1c 02 00 05 1f bb

Response Data: AA 00 07 00 00 00 00 00 01 00 06 BB

No card or have some mistake: AA 00 02 01 83 80 BB

9.1.14 ISO15693_Transfer_Command (0x1D)

Data Field:

DATA[1]: The length of the data which send to the card

DATA[2…N]: DATA

Response:

Data Field

STATUS: 0x00 – OK

DATA[0~N]: The data that response by the card

Description

This command is using for transparent any command to The Card which these commands meet the ISO15693 protocol .

Example:

Send Data: AA 00 04 1D 02 02 2B 32 BB(Get The Card's Information)

Response Data: AA 00 10 00 00 0F 72 9C 56 01 00 00 07 E0 08 00 3F 03 88 FD BB

No card or have some mistake: AA 00 02 01 83 80 BB

10 Error/Status Code

System Error/Status Codes (0x00-0x0F)

0x00	Command OK.
0x01	Command FAILURE
0x80	SET OK.
0x81	SET FAILURE
0x82	Reader reply time out error
0x83	The card do not exist
0x84	The data response from the card is error
0x85:	The parameter of the command or the Format of the command Erro
0x87	Unknown Internal Error
0x8f	Reader received unknown command

ISO14443 Error Codes:

0x8A:	Some Erro appear in the card InitVal process
0x8B:	Get The Wrong Snr during anticollison loop
0x8C:	The authentication failure

ISO15693 Error Codes:

0x90	The Card do not support this command
0x91	The Foarmat Of The Command Erro
0x92	Do not support Option mode
0x93	The Block Do Not Exist
0x94	The Object have been locked
0x95	The lock Operation Do Not Success
0x96	The Operation Do Not Success