RFID READER

13.56MHz Reader/Writer

E1356D

Communication Protocol

CONTROLLED

Version 3.5
Jul 2011

CONTENT

1. MAIN FEATURES

The reader is slave device, not active send data until received command form host

2. COMMUNICATION PROTOCOL

2.1 Host Command Format

Host to Reader

Preamble	Len	DeviceID	Command code	Data	Checksum

Preamble: 2 bytes, 0xAABB
Len: 2 bytes, indicating the number of bytes from the DeviceID to Checksum
In this reader, the first byte is effective, the second byte keep 0
DeviceID: 2 bytes
Command code: 2 bytes
Data: variable length depends on the Command code
Verification: 1 byte, XOR of all the bytes from DeviceID to Data

Attention
If there is any byte equaling to AA occurs between Len and Checksum, one byte 00 will be added after this byte to differentiate preamble. However, the Len keeps unchanged.
Example, writing data (0x00112233445566778899AABBCCDDEEFF) to block 1

Host ⇨ E1356D: AABB1600000009020100112233445566778899AA**00**BBCCDDEEFF0A

2.2 Reader response format

E1356D to Host

Preamble	Len	DeviceID	Command code	Status	Data	Checksum

Preamble: 2 bytes, 0xAABB
Len: 2 bytes, indicating the number of bytes from the DeviceID to Checksum
In this reader, the first byte is effective, the second byte keep 0
DeviceID: 2 bytes
Command code: 2 bytes
Status: 1 byte , 00 = success, Not 0 = fail
Data Range: Response date, maybe blank
Verification: 1 byte, each byte is XOR from Device ID to the last byte of the Command Sending data

Attention
If there is any byte equaling to AA occurs between Len and Checksum, one byte 00 will be added after this byte to differentiate preamble. However, the Len keeps unchanged.
Example, reading data (0x00112233445566778899AABBCCDDEEFF) from block 1

E1356D ⇨ Host: AABB1600000009020000112233445566778899AA**00**BBCCDDEEFF0B

3. COMMAND DETAILS

E1356D does answer each valid command.

"Response date: None" in the following description means the "Data" range of response stream is blank.

3.1 System Function

3.1.1 rf_init_com [1]

Function: Set baud rate
Command code: 0x0101
Sending data: 00 = 4800
01 = 9600
02 = 14400
03 = 19200
04 = 28800
05 = 38400
06 = 57600
07 = 115200
Remark: Default baud rate is 19200 bps after power on
Response date: None
Example:
Host ⇨ E1356D: AABB0600000001010303
E1356D ⇨ Host: AABB0600111201010003

3.1.2 rf_get_model [1]

Function: Read the reader model, refer **Table .1**
Command code: 0x0401
Sending data: None
Response date: Reader model
Example:
Host ⇨ E1356D: AABB05000000040105
E1356D ⇨ Host: AABB11001112040100534C3530304C2D3036303843

3.3.3 rf_init_device_number [1]

Function: Set Device ID
Command code: 0x0201
Sending data: 2 bytes device ID
Remark: E1356D only response to the command that Device ID is in accord with itself, and broadcast command that DeviceID equals to 0x0000
Response date: None

3.3.4 rf_get_device_number[1]

Function: Read DeviceID

Command code: 0x0301
Sending data: None
Response date: 2 bytes DeviceID

3.3.5 rf_beep[1]

Function: Set buzz time
Command code: 0x0601
Sending data: 1 byte beep time, unit 10ms
Response date: None

3.3.6 rf_light[1]

Function: Set LED color
Command code: 0x0701
Sending data: 00 = off
01 = red
02 = green
03 = yellow
Response date: None

3.3.7 rf_init_type[1]

Function: Set E1356D working mode
Command code: 0x0801
Sending data: 1 byte
'A' = ISO14443A mode
'B' = ISO1443B mode
'1' = ISO5693 mode
Response date: None

3.3.8 rf_antenna_sta[1]

Function: Manage RF transmit
Command code: 0x0C01
Sending data: 1 byte
00 = off
Not 0 = on
Response date: None

3.2 ISO14443A - Mifare Function

3.2.1 rf_request[2]

Function: ReqA
Command code: 0x0102
Sending data: 1 byte

0x26 = REQ_STD
0x52 = REQ_ALL
Response date: 2 bytes card type code

3.2.2 rf_anticoll[2]

Function: Anticollision
Command code: 0x0202
Sending data: None
Response date: 4 bytes card serial number

3.2.3 rf_select[2]

Function: Select card
Command code: 0x0302
Sending data: 4 bytes card serial number
Response date: 1 byte card capacity code

3.2.4 rf_halt[2]

Function: Halt
Command code: 0x0402
Sending data: None
Response date: None

3.2.5 rf_M1_authentication2[2]

Function: Authenticate Mifare card key
Command code: 0x0702
Sending data: 1 byte code authenticate mode + 1 byte absolute block number + 6 bytes key
Authenticate mode
0x60 = KeyA
0x61 = KeyB
Response date: None

3.2.6 rf_M1_read[2]

Function: Read block
Command code: 0x0802
Sending data: 1 byte absolute block address
Response date: 16 bytes date

3.2.7 rf_M1_write[2]

Function: Write block
Command code: 0x0902
Sending data: 1 byte absolute block address + 16 bytes written date
Response date: None

3.2.8 rf_M1_initval[2]

Function: Initialize electronic purse
Command code: 0x0A02
Sending data: 1 byte absolute block address + 4 bytes initial value (low bytes in the former)
Response date: None

3.2.9 rf_M1_readval[2]

Function: Read purse value
Command code: 0x0B02
Sending data: 1 byte absolute block address
Response date: 4 bytes value (low bytes in the former)

3.2.10 rf_M1_decrement[2]

Function: Do decrement
Command code: 0x0C02
Sending data: 1 byte absolute block address + 4 bytes decrement value
Response date: None

3.2.11 rf_M1_increment[2]

Function: Do increment
Command code: 0x0D02
Sending data: 1 byte absolute block address + 4 bytes increase value
Response date: None

3.2.12 rf_M1_restore[2]

Function: Transfer a block date into card buffer
Command code: 0x0E02
Sending data: 1 byte absolute block address
Response date: None

3.2.13 rf_M1_transfer[2]

Function: Write the date in the card buffer into a block of card
Command code: 0x0F02
Sending data: 1 byte absolute block address
Response date: None

3.2.14 rf_ul_select[2]

Function: Ultralight card Anticoll and Select
Command code: 0x1202
Sending data: None
Response date: 7 bytes ultralight UID

3.2.15 rf_ul_write[2]

Function: Write a page of data into ultralight card
Command code: 0x1302
Sending data: 1 byte page address + 4 bytes written date
Response date: None

3.2.16 rf_UC_authentication[2]

Function: Authenticate password of Ultralight_C
Setp1
Command code: 0x4002
Sending data: None
Response date: ek(RndB)
Setp2
Command code: 0x4102
Sending data: ek(RndB')
Response date: None
Remark: This command is only one with two steps

3.2.17 rf_UC_changekey[2]

Function: Change password of Ultralight_C
Command code: 0x4202
Sending data: 16 byte new password
Response date: None

3.2.18 rf_typea_rst[3]

Function: Request MifareProX and reset
Command code: 0x1002
Sending data: 0x26 = REQ_STD
0x52 = REQ_ALL
Response date: 4 bytes CSN + ATS information

3.2.19 rf_DESFire_rst[3]

Function: Request DESFire and reset
Command code: 0x3002
Sending data: 0x26 = REQ_STD
0x52 = REQ_ALL
Response date: 7 bytes CSN + ATS information

3.2.20 rf_cos_command[3]

Function: Exchange data between PICC and PCD according with T = CL protocol
Command code: 0x1102
Sending data: COS command
Response date:

3.2.21 rf_Shc1102_Auth[3]

Function: SHC1102 card check password
Command code: 0x2002
Sending data: 4 bytes password
Response date: None

3.2.22 rf_Shc1102_Read[3]

Function: Read data block of SHC1102 card
Command code: 0x2102
Sending data: 1 bytes block address
Response date: 4 bytes data

3.2.23 rf_Shc1102_Write[3]

Function: Write data block of SHC1102 card
Command code: 0x2202
Sending data: 1 bytes block address +4 bytes written
Response date: None

3.3 ISO14443B Function

3.3.1 rf_atqb[4]

Function: RTQB and Attrib
Command code: 0x0103
Sending data: 1 byte RTQB mode code
0 = REQB
1=WUPB
Response date: ATQB Response

3.3.2 rf_at020_check[4]

Function: rf_at020_check
Command code: 0x0104
Sending data: 8 bytes password
Response date: None

3.3.3 rf_at020_read[4]

Function: Read a page of data from AT88RF020
Command code: 0x0204
Sending data: 1 byte page address
Response date: 8 bytes read data

3.3.4 rf_at020_write[4]

Function: Write a page of data into AT88RF020
Command code: 0x0304
Sending data: 1 byte page address + 8 bytes written data
Response date: None

3.3.5 rf_at020_lock[4]

Function: AT88RF020 lock operation
Command code: 0x0404
Sending data: 4 bytes
Response date: None

3.3.6 rf_at020_count[4]

Function: AT88RF020 take count
Command code: 0x0504
Sending data: 6 bytes signature
Response date: None

3.3.7 rf_at020_count[4]

Function: AT88RF020 take count
Command code: 0x0504
Sending data: 6 bytes signature
Response date: None

3.3.8 rf_st_select[4]

Function: Req ST card (SR176/SRIX4K)
Command code: 0x0105
Sending data: None
Response date: 1 byte chip ID number

3.3.9 rf_st_completion[4]

Function: Set ST card into DESACTIVED status
Command code: 0x0205
Sending data: None
Response date: None

3.3.10 rf_sr176_readblock[4]

Function: Read one block of data from SR176
Command code: 0x0305
Sending data: 1 byte block address
Response date: 1 byte data

3.3.11 rf_sr176_writeblock[4]

Function: Write one block of data to SR176
Command code: 0x0405
Sending data: 1 byte block address + 1 byte written data
Response date: None

3.3.12 rf_sr176_protectblock[4]

Function: Lock SR176
Command code: 0x0505
Sending data: 1 byte lockreg
Response date: None

3.3.13 rf_srix4k_readblock[4]

Function: Read one block of data from SRIX4K
Command code: 0x0605
Sending data: 1 byte block address
Response date: 4 bytes data

3.3.14 rf_srix4k_writeblock[4]

Function: Write 1 block data to SRIX4K
Command code: 0x0705
Sending data: 1 byte block address
Response date: 4 bytes written data

3.3.15 rf_srix4k_getuid[4]

Function: Get SRIX4K UID
Command code: 0x0905
Sending data: 8 byte UID
Response date: None

3.3.16 rf_cos_command[4]

Function: Exchange data between PICC and PCD according with T = CL protocol
Command code: 0x1102
Sending data: COS command
Response date:

3.4 ISO15693 Function

3.4.1. ISO15693_Inventorys (Multi Card)[5]

Command code: 0x0010
Sending data: None
Response date: 9 * n bytes, 9 bytes in a frame

The structure of every stream is: 1byte DSFID + 8byte UID

3.4.2. ISO15693_Inventory (Single card) [5]

Command code: 0x0110
Sending data: None
Response date: 9 bytes, 1 byte DSFID + 8 bytes UID

3.4.3. ISO15693_Stay_Quiet[5]

Command code: 0x0210
Sending data: 8 bytes UID
Response date: None

3.4.4. ISO15693_Select[5]

Command code: 0x0310
Sending data: 8 bytes UID
Response date: None

3.4.5. ISO15693_Reset_To_Ready[5]

Command code: 0x0410
Sending data: 1 byte model + 8 bytes UID
Response date: None
Remark

Model Byte							
bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
					option flag	address flag	select flag

3.3.6. ISO15693_Read[5]

Command code: 0x0510
Sending data: 1 byte model + 8 bytes UID + 1 byte initial block number + 1 byte block number
Response date: Read date

3.4.7. ISO15693_Write[5]

Command code: 0x0610
Sending data: 1 byte model + 8 bytes UID + 1 byte block number + 4 bytes written date
Response date: None

3.4.8. ISO15693_Lock_Block[5]

Command code: 0x0710
Sending data: 1 byte model + 8 bytes UID + 1 bytes block number
Response date: None

3.4.9. ISO15693_Write_AFI[5]

Command code: 0x0810
Sending data: 1 byte model + 8 bytes UID + 1 byte written date
Response date: None

3.4.10. ISO15693_Lock_AFI[5]

Command code: 0x0910
Sending data: 1 byte model + 8 bytes UID
Response date: None

3.4.11. ISO15693_Write_DSFID[5]

Command code: 0x0A10
Sending data: 1 byte model + 8 bytes UID + 1 byte written date
Response date: None

3.4.12. ISO15693_Lock_DSFID[5]

Command code: 0x0B02
Sending data: 1 byte model + 8 bytes UID
Response date: None

3.4.13. ISO15693_Get_System_Information[5]

Command code: 0x0C10
Sending data: 1 byte model + 8 bytes UID
Response date: 8 bytes UID + 1 byte DSFID + 1 byte AFI

3.4.14. ISO15693_Get_Block_Security[5]

Command code: 0x0D10
Sending data: 1 byte model + 8 bytes UID + 1 byte initial block number + 1 byte block number
Response date: n bytes lock state, one byte to one block correspondence

3.4.15. Srf55vp_Read (Infineon tag special) [5]

Command code: 0x1010
Sending data: 8 bytes UID + 1 byte page number
Response date: 8 bytes read date

3.4.16. SRF55V_Inventorys (Infineon tag special) [5]

Command code: 0x1410
Sending data: 1 byte AFI
Response date: 9 * n bytes, 9 bytes in a frame
The structure of every stream is: 1byte DSFID + 8byte UID

3.5 Pass through Function

3.5.1. rf_transceive[5]

Function: Send data to Tag and receive response data
Command code: 0106
Sending data: Sending data sent to tag, without CRC bytes
CRC bytes is auto managed by reader
Response date:

Table .1

Model	E1356DL	E1356DA	E1356DD	E1356DF
Protocol	ISO14443A	ISO14443A	ISO15693	ISO14443A ISO14443B ISO15693
Command	[1] [2]	[1] [2] [3]	[1] [5]	[1] [2] [3] [4] [5]