Smart card Protocol
Document Version : 1.9.5
Date: 22-Mar-16
Last modified: Sindhu Sawant

Smart card Protocol - V1.9.5

Basic Protocol information

- Baud Rate : 38400
- Data bits : 8
- Parity : None
- Stop bits : 1
- Flow control : None

Command Send (Send from PC to Reader)

<Start of command><Slave no><Command><Data><End of command>

Start of command is → $
End of command is → '\n'

Slave no is one char which represents ID of reader it starts from ASCII char '1' to '8'
Command is single char which represents the command
Data is represented in BCD and is always in ASCII format
i.e. to send hex byte (or decimal 255) it sends two chars FF.

Return command (Return from Reader)

<Start of return><>Return code><Data><end if command>

Start of return → #
End of command → '\n'

If there is error in that case return code is attached with x and two char return code as follows

x<return code hexa decimal >
x → char indicates that error is there
return code is two char code (which can be converted in to byte)

#xFE
indicates error and return error code is FE (-2)

Commands

Commonly used symbols in following protocol

Keys:

k = Key type (This is single char indicates key type)

k = '0' → Key A

k = '1' → Key B

There are two types of keys used in smart card authentication.

Request Type:

r = Card Request type

r = '0' Request only idle cards

r = '1' indicates request all cards

Sector:

ss = Sector (two chars)

This is sector no of key which is to be used for authentication this is uses as hexadecimal for sector 5 you have to send 05 for sector 15 you have to send 0F

Sector is key location in reader IC.

Block:

bb = Block (two chars)

This is one byte hexadecimal. This is block no in card which is to be read/ Write.

for block 10 you have to send "0A" and for block no 16 you have to send "10"

Card No:

cccccccc = 4-Byte Card no (Serial no of card)

Data:

dddd.. = Data and data is send as hexadecimal i.e to send data '0' you have send same in "30"

(data byte 0 to 15 is send in sequence 1st 0th byte and last is 15th)

b Baud rate setting Command

$1bXX↵

Where XX – 2 = 9600
3 = 19200
4 = 38400

Start-up string indicates Baud rate→ #00V42P013204
Last two digit is baud rate.

Return:

#<return code>

Example:

$1b02↵

#00

V Version command

User to read version no of Reader.

$1V↵

Return:

#00<Major ver no><minor version no><version char>

Example:

$1V↵

#0011A

S Request card :

This command is used to check if any card is present in location of reader

$1Sr

Return is

#<return code -2><Card Type low -2><card type high -2>

Example:
$1S0
#000200

T → anti-collide card

This command is used to get serial no of requested card.

$1T00

00 = default value.
Return:

#<return code -2><card no -8>

card no = 4 bytes of card no (hexadecimal)

Example:

$1T00↵

#007E1D1E46

I → Select Card

This command is used to select card for operations

$1I<card no –8>

Return :
#<return code><Card Type -2>

$1I7E1D1E46↵

#0018

U → Authenticate card

Authenticate command is used to authenticate card with reader keys and you can use any one key A/B
$1Ukssbb

k = key
ss = sector
bb = block

Return :
#<return code>

Example:
$1U00000

#00

R → Read card

$1Rbb

#<return code><ddddddddddddddddddddddddddddddddd>

dd.. = Data (data byte 0 to 15 is send in sequence 1st 0th byte and last is 15th)

Example:

$1R00↵

#00461E1D7E3B980200648E261849303503

W→ Write card

$1Wbbdddddddddddddddddddddddddddddddd

---

bb = Card Block
dd.. = data to write

Return:
#<Return code>

Example:

$1W010000BEBF30000830000000000004C0101↵

#00

O → Increment Data

Q → Decrement Data

This command is used to increment/Decrement the card block value by given number. Both increment and decrement data commands is same except operation which is increment / decrement

Data format for increment decrement block
01 is stored in data location as example
Note: as per protocol we send 00 byte 1st and then 15th byte.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FF	00	FF	00	00	00	00	01	FF	FF	FF	FE	00	00	00	01
Add		inv add		data				inverted data				data			

$1Obbttvvvvvvvv
$1Qbbttvvvvvvvv

where :
tt : Same as block value bb

---

vvvvvvvv : is value by which data is to be incremented / decremented this is in two's compliment format.

Return:

#<Return code>CCCCCCCC

Example:
$1OO202OOOOOOO3

This will increment block 2 by value 3 and store same in block 2

J →Write Key to EEPROM in READER IC

This command is used to write keys in reader eeprom.

$1JSSkddddddddddddd

---

ddd... is Key 12 char (6 bytes)
ss = Sector no in reader IC
k = Key type

#<return code>

$1J000FFFFFFFFFFFF↵

#00

Group command

Group commands are combination of different commands like request+anticollide+select+auth

Z → Request anti-collide Select Auth card

$1Zrkssbb

---

r = Request type
k = Key type
ss = Sector
bb = block to authenticate

Return:

#<return code>cccccccc

X → Auth Read

To authenticate and read the card

$1Xrkssbb

Return:

#<return code>cccccccddddddddddddddddddddddddddddddd

Example:

$1X000102↵

#000000BEBF30000830000000000004C0101

Y → auth Write

$1Yrkssbbddddddddddddddddddddddddddddddd

Return :

#<return code>cccccccc

Example:

$1Y0001021212121212121212121212121212↵

#007E1D1E46

N → Auth 512 block Read

$1Nrkssbb

bb: start Block number

Return:

#00003633 1D7E6698 0200648E 261849303503
#00010011 2233445566778899AABBCCDDEEFF
|
|
#001F000000000000FF078069FFFFFFFFFFFF

N → Auth Read number of block

$1Nrkssbbcc

bb: start Block number
cc : No. of block to read

Command: $1N00000005

Return:

#000400000000000000000000000000000000
#000300000000000FF078069FFFFFFFFFFFF
#000200000000000000000000000000000000
#000100000000000000000000000000000000
#00005B2C2C3368980200648ED41361304108

K → Auth Increment Data

L → Auth Decrement Data

This command is used to increment/Decrement the card block value by given number. Both increment and decrement data commands is same except increment / decrement

(Note: look in to increment/ decrement command O/Q for data format)

$1Krkssbbttvvvvvvvv
$1Lrkssbbttvvvvvvvv

where :
tt : same as block number bb

vvvvvvvv: is value by which data is to be incremented / decremented this is in two's compliment format.

Return:

#<Return code>CCCCCCCC

Example:
$1L00000202000000003

This will decrement block 2 by value 3 and store same in block 2

P → Poll card

Poll command is used to poll card and remain in poll condition unless card shown to the reader. once card is received poll command return the card information which is requested by command. you can stop poll command by proper poll data command. you should not run any other command when pool command is in process. (only poll stop command can be issued to stop poll command). Once card is detected poll command comes out of poll command mode.

$1P<Poll Type -1>rkssbb

Poll Type --> '0' = Stop Poll
'1' = Just Poll for card presence
'2' = Poll and auth card
'3' = Poll auth and read sector

Return:

#00

Once card is detected it will return as per Poll type

For poll type '1'
#<Return Code>P1

For '2'
#<return code>P2cccccccc

For '3'

#<return code>P3ccccccccddddddddddddddddddddddddddddddd

Example:

$1P3000001

#00
after some time when card is displayed to reader

#00P3*7E1D1E46*0000BEBF3000083000000000004C0102

For '5'

<[return code]P5cccccccc
<[return code][data location][data ……]
|
|
Total 16 row of data (16*16 = 512 data)

Example:

Command send to read 512 bytes from 0th block

```
$1P5000000

#00
After card is detected

#00P57E1D3336
#0000036331D7E66980200648E261849303503
#0001001122334455667788 99AABBCCDDEEFF
|
|
#001F000000000000FF078069FFFFFFFFFFFF
```

G → Buzzer control for commands (added in V1.9.1)

This will give single beep for any command operation. This will enable Buzzer (Beep) for following commands,

1) Poll
2) Read
3) Write
4) ReqAntiSelAuth
5) Increment/Decrement

$1G00<Control Byte>↵

"Control Byte" can be signified using following table,

Command Type					Control Byte	Description
Inc/Dec	ReqAntiSelAuth	Write	Read	Poll		
0	**0**	**0**	**0**	**0**	**0x00**	**Disable Beep for all commands**
0	**0**	**0**	**0**	**1**	**0x01**	**Enable beep for Poll command only**
0	**0**	**0**	**1**	**0**	**0x02**	**Enable Beep for Read command only**
0	0	0	1	1	0x03	
0	**0**	**1**	**0**	**0**	**0x04**	**Enable Beep for Write command only**
0	0	1	0	1	0x05	
0	0	1	1	0	0x06	
0	0	1	1	1	0x07	
0	**1**	**0**	**0**	**0**	**0x08**	**Enable Beep for Req-Anti-Select-Auth command only**
0	1	0	0	1	0x09	
0	1	0	1	0	0x0A	
0	1	0	1	1	0x0B	
0	1	1	0	0	0x0C	
0	1	1	0	1	0x0D	
0	1	1	1	0	0x0E	
0	1	1	1	1	0x0F	
1	**0**	**0**	**0**	**0**	**0x10**	**Enable Beep for Increment, decrement command only**
1	0	0	0	1	0x11	
1	0	0	1	0	0x12	
1	0	0	1	1	0x13	
1	0	1	0	0	0x14	
1	0	1	0	1	0x15	

1	0	1	1	0	0x16	
1	0	1	1	1	0x17	
1	1	0	0	0	0x18	
1	1	0	0	1	0x19	
1	1	0	1	0	0x1A	
1	1	0	1	1	0x1B	
1	1	1	0	0	0x1C	
1	1	1	0	1	0x1D	
1	1	1	1	0	0x1E	
1	**1**	**1**	**1**	**1**	**0x1F**	**Enable Beep for all commands**

Whereas,
0 = Disable Beep
1 = Enable Beep

Return:

#00

Example1: Beep enable for Read command

Command:

$1G0002↵

Response:
#00

After reading data using **Read/AuthRead** command buzzer gives a single beep.

Example2: Beep enable for ReqAntiSelAuth and Read command

Command:

$1G000A↵

Response:
#00

After sending **ReqAntiSelAuth** or **Read/AuthRead** command buzzer gives single beep.

M → Buzzer Beep Sound control (added in V1.9.1)

This command is used to make buzzer ON for different beeps. The reader will support following beeps,

A] Fixed Beep

1) Single Beep
2) Double Beep
3) Error Beep

B] Programmable timer Beep

$1M<Beep Type><Beep Count>↵

A] Fixed Beep

For Fixed beep, Beep Count should be 0x00.

Beep type:

0x01 = Single Beep
0x02 = Double Beep
0x03 = Error Beep

Example1: Make Single Beep ON

Command:

$1M0100↵

Response:

#00

After Sending this command buzzer gives single beep.

Example2: Make Double Beep ON

Command:

$1M0200↵

Response:

#00

After Sending this command buzzer gives double beep.

B] Programmable Timer Beep

For programmable beep, Beep type should be 0x00.

Beep Timer:

Beep timer = Beep Count * 100msec

Whereas,
Beep Count = 0x01 to 0xFF
(Note: This timer is programmable for 100msec to 25500msecs)

For E.g. if we want buzzer ON for 5sec (5000msec) then we have to set Beep Count to 50 i.e. 0x32.

Example1: Make Buzzer ON for 5 secs

Command:

$1M0032↵

Response:
#00

After Sending this command buzzer remains ON for 5secs.

Example2: Make Buzzer ON for 1 sec

Command:

$1M000A↵

Response:
#00

After Sending this command buzzer remains ON for 1 sec.

NOTE:
Buzzer timer will reset if other command is send in which buzzer is enabled.

For e.g. if we have enabled beep for Read command and made buzzer ON for 10 sec, but after 3sec if we send Read command then buzzer will give beep and Buzzer goes OFF. It will not remain ON for next 7 secs.

o → RED LED

This command is used to on red LED with time based beeps.

Command: $1o0100bbtt

where:

00: red LED
bb: beep type
0x01 = Single beep
0x02 = Double beep
0x03 = Error beep
tt: Beep timer

Command: $1o01000005
Return: #000100

---

o → GREEN LED

This command is used to on red LED with time based beeps.

Command: $1o0101bbtt

where:

01: Green LED
bb: beep type
0x01 = Single beep
0x02 = Double beep
0x03 = Error beep
tt: Beep timer

Command: $1o01000005
Return: #000000

Error codes of Smart card reader

// Error Codes
// Each function returns a status value, which corresponds to the mifare error codes.

ERROR DESCRIPTION	ERROR CODE	Hex Codes
#define READER_ERR_BASE_START	0	0x00
#define MI_OK	0	0x00
#define MI_CHK_OK	0	0x00
#define MI_CRC_ZERO	0	0x00
// ICODE1 Error Codes		
#define I1_OK	0	0x00
#define I1_NO_ERR	0	0x00
#define MI_CRC_NOTZERO	1	0x01
#define MI_NOTAGERR	-1	0xFF
#define MI_CHK_FAILED	-1	0xFF
#define MI_CRCERR	-2	0xFE
#define MI_CHK_COMPERR	-2	0xFE
#define MI_EMPTY	-3	0xFD
#define MI_AUTHERR	-4	0xFC
#define MI_PARITYERR	-5	0xFB
#define MI_CODEERR	-6	0xFA
#define MI_SERNRERR	-8	0xF8
#define MI_KEYERR	-9	0xF7
#define MI_NOTAUTHERR	-10	0xF6
#define MI_BITCOUNTERR	-11	0xF5
#define MI_BYTECOUNTERR	-12	0xF4
#define MI_IDLE	-13	0xF3
#define MI_TRANSERR	-14	0xF2
#define MI_WRITEERR	-15	0xF1
#define MI_INCRERR	-16	0xF0
#define MI_DECRERR	-17	0xEF
#define MI_READERR	-18	0xEE
#define MI_OVFLERR	-19	0xED
#define MI_POLLING	-20	0xEC
#define MI_FRAMINGERR	-21	0xEB
#define MI_ACCESSERR	-22	0xEA
#define MI_UNKNOWN_COMMAND	-23	0xE9
#define MI_COLLERR	-24	0xE8
#define MI_RESETERR	-25	0xE7
#define MI_INITERR	-25	0xE7
#define MI_INTERFACEERR	-26	0xE6
#define MI_ACCESSTIMEOUT	-27	0xE5
#define MI_NOBITWISEANTICOLL	-28	0xE4
#define MI_QUIT	-30	0xE2
#define MI_CODINGERR	-31	0xE1
#define MI_RECBUF_OVERFLOW	-50	0xCE

#define MI_SENDBYTENR	-51	0xCD
#define MI_CASCLEVEX	-52	0xCC
#define MI_SENDBUF_OVERFLOW	-53	0xCB
#define MI_BAUDRATE_NOT_SUPPORTED	-54	0xCA
#define MI_SAME_BAUDRATE_REQUIRED	-55	0xC9
#define MI_WRONG_PARAMETER_VALUE	-60	0xC4
// ICODE1 Error Codes		
#define I1_WRONGPARAM	-98	0x9E
#define I1_WRONGPARAM	-61	0xC3
#define I1_NYIMPLEMENTED	-62	0xC2
#define I1_TSREADY	-63	0xC1
#define I1_TIMEOUT	-70	0xBA
#define I1_NOWRITE	-71	0xB9
#define I1_NOHALT	-72	0xB8
#define I1_MISS_ANTICOLL	-73	0xB7
#define I1_COMM_ABORT	-82	0xAE
#define MI_BREAK	-99	0x9D
#define MI_NY_IMPLEMENTED	-100	0x9C
#define MI_NO_MFRC	-101	0x9B
#define MI_MFRC_NOTAUTH	-102	0x9A
#define MI_WRONG_DES_MODE	-103	0x99
#define MI_HOST_AUTH_FAILED	-104	0x98
#define MI_WRONG_LOAD_MODE	-106	0x96
#define MI_WRONG_DESKEY	-107	0x95
#define MI_MKLOAD_FAILED	-108	0x94
#define MI_FIFOERR	-109	0x93
#define MI_WRONG_ADDR	-110	0x92
#define MI_DESKEYLOAD_FAILED	-111	0x91
#define MI_WRONG_SEL_CNT	-114	0x8E
#define MI_WRONG_TEST_MODE	-117	0x8B
#define MI_TEST_FAILED	-118	0x8A
#define MI_TOC_ERROR	-119	0x89
#define MI_COMM_ABORT	-120	0x88
#define MI_INVALID_BASE	-121	0x87
#define MI_MFRC_RESET	-122	0x86
#define MI_WRONG_VALUE	-123	0x85
#define MI_VALERR	-124	0x84