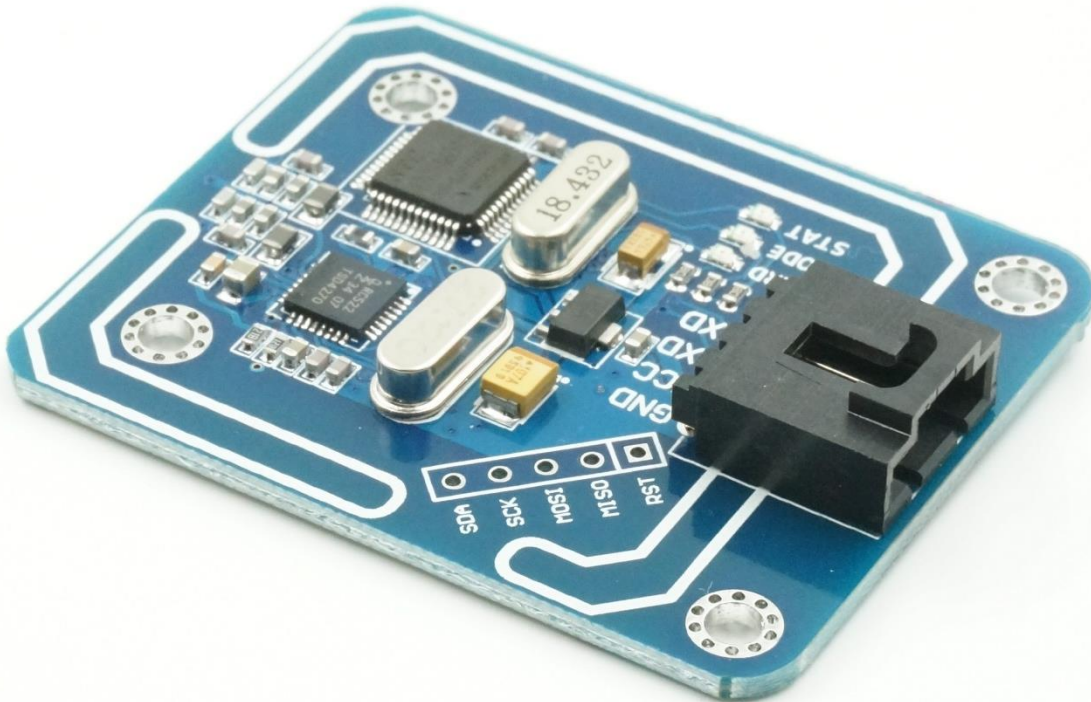


# SSRFIDV3.0 Manual



## ***Instruction***

*As NXP has discontinued 4 byte UID tags, we need to upgrade our module. This RFID reader module is based on MFRC522, supporting the ISO14443 standard.*

*This module has UART interface. Users don't have to consider the complex control algorithms. Simply sending commands through the UART interface can do all the operation. This module also provides compact commands, which is useful in the application of access control, attendance and other identification system.*

*This module supports cards including Mifare One S50, S70, Mifare UltraLight, Mifare Ultralight C, Mifare DESFire etc.*

*With built-in 8K of EEPROM in this module, users can easily read and write data by sending commands.*

## ***Feature***

- *Control by Serial UART interface*
- *Excellent reading/writing distance*
- *Configuration data is preserved in EEPROM against power cut off*
- *Built-in 8K EEPROM, easy to access it by sending commands.*
- *Easy to use, by sending simple commands users can control it to read or write cards.*
- *In compact command, command is as short as one byte.*
- *Excellent EMC performance.*

## Parameter

- Power Supply: 4.5V~5.5V, typically 5V.
- Interface: UART (5V TTL) and SPI (3.3V TTL)
- Size: 40mm x 50mm
- Performance:

TAGS	READING	WRITING	MAX DISTANCE	NOTE
MIFARE 1 S50/S70	✓	✓	55mm	White PVC card
MIFARE ULTRALIGHT	✓	✓	66mm	White PVC card
MIFARE ULTRALIGHT C	✓	✓	28mm	White PVC card, better Min. 8mm
MIFARE DESFIRE EV1	✓	✗	38mm	White PVC card, only read tag ID

The SPI interface is directly connected with MFRC522's SPI. You can access MFRC522 via this interface. If you do so, you need to refer to [MFRC522 Datasheet](#). Command described below is not supported by this interface.

## LED

There are 3 LEDs on the board. We marked near each LED.

- **STATE LED:** Show status. While this module is powered up, STATE LED is on. If this module executes command successfully, STATE LED flashes once. Otherwise, it flashes 4 times.
- **CARD LED:** While this module detects cards, this LED is on. While the card leaves the detection area, this LED is off.
- **MODE LED:** In Basic Command mode, MODE LED is off. In Compact Command mode, MODE LED is on.

## Command Description

There are two kinds of commands: Basic Command and Compact Command. Basic Command consists of 3 or more bytes. Compact Command consists of only 1 byte. (All Command Data in hexadecimal format)

For any wrong command sending to this module, 0xFF will be returned. Basic Command should be sent to this module within 5 seconds. Otherwise the module will return 0xEE. If any other data is added behind a right command and sent to this module, those data will be ignored. Take the command **AB 02 01** for example, if **AB 02 01 AA** is sent to this module, the **AA** will be ignored and the command **AB 02 01** will be executed.

## UART Configuration

Baud rate could be 2400bps ~ 115200bps

Default setting:

- Baud Rate: 9600bps
- Parity bit: None
- Start bit: 1
- Data bit: 8
- Stop bit: 1

## Compact Command

### Command Format

No.	Command	Explain
1	0x01	Automatically search cards
2	0x02	Automatically read the card serial number.
3	0x03	Card serial number will be automatically stored in the EEPROM
4	0x04	Automatically determine whether the card is in authorization list
5	0x05	Automatically find and remove the card in authorization list

### Respond Data

Success: **Related Command** or **Data**

- **Related Command:** the command calling this respond
- **Data:** data for the command, depending on the command

Fail: Value of NOT operation over the related command code

### Compact Command Description

#### 1. Search card: 0x01

Description: look for cards. Return **0x01** while it detects a card in its reading area. Return **0xFE** while a card leaves the reading area.

Command format: **01**

#### 2. Read the card serial number: 0x02

Description: Be ready to read the card. Return the card serial number if a card is detected.

Command format: **02**

#### 3. Record the card's serial number into an authorization list: 0x03

Description: Whenever a card enters the reading area, it records the card's serial number into the EEPROM. Maximum 256 cards' number can be recorded.

This module will check the free space of EEPROM from 0x0000. If there are 4 continuous addresses available, it will save the card number there. Each number will be only saved once. If you use this command, we do not recommend you do writing operation to the EEPROM. Otherwise, the data of authorization list might be lost.

Command format: **03**

Return:

- Success: **03**
  - Fail: **FC**
- #### 4. Check if a card is in authorization list: 0x04

Description: Whenever the card enters the reading area, check if the card is in the authorization list.

Command format: **04**

Return:

- In the list: **04**
  - Not in the list: **FB**
5. **Remove a card's serial number from authorization list**

Description: Whenever the card enters the reading area, remove the card from authorization list in EEPROM.

Command format: **05**

Return:

- Success: **05**
- Fail: **FA**

## Basic Command Description

### Command format

**Header + Length + Instruction + Data + (Checksum)**

1. **Header: 0xAB**
2. **Length:** 1 byte, the byte number from **Length** field (included) to the last byte of **Data** field.
3. **Instruction:** 1 byte, operation instruction, more detail on latter pages.
4. **Data:** Depending on the command, some commands contain none data.
5. **Checksum:** 1 byte, optional, can be configured by command. It is the value of XOR operation over all the bytes from the **Length** byte to the last byte of **Data**. By default, this byte is not included in basic demand. However, to improve working stability of this module in certain environment, checksum can be added in Basic Command. If the checksum is active in your command, you have to calculate it. We supply example code of adding and verifying checksum. For example, this command **AB 07 0C 00 00 04 0F** has a checksum of **0F**. We got the checksum in the following way:

$$0F = 07 \wedge 0C \wedge 00 \wedge 00 \wedge 04$$

If you need more information about XOR operation, [you can click here](#).

**Tips: before sending basic commands, you should make sure the checksum setting first. By default the basic command has no checksum. More information will be explained later.**

**Instruction code:**

No.	Instruction	Explanation
1	0x01	Read the card type
2	0x02	Search cards, and read the card's serial number
3	0x03	Read data in the card
4	0x04	Write data to the card
5	0x05	Initialize the wallet
6	0x06	Recharge the wallet
7	0x07	Deduct from the wallet
8	0x08	Read wallet
9	0x09	Read EEPROM
10	0x0a	Write EEPROM
11	0x0b	Erase EEPROM
12	0x0c	Check if the EEPROM is being written

13	0x0d	Add or remove checksum of Basic Command
14	0x0e	Configure the baud rate
15	0x0f	Return to default configuration
16	0x10	Return to standby state

Tips: No.5 ~ NO.8 are means e-wallet functions.

### Respond Data

Success: **Header + Length + Instruction + Data + (Checksum)**

- **Header:** 0xAB
- **Length:** 1 byte, all the bytes from **Length** filed to the last byte of **Data** field
- **Instruction:** 1 byte, the Instruction calling this respond
- **Data:** Depending on the command, can be empty
- **Checksum:** 1 byte, optional, value of XOR operation over all the bytes from the **Length** byte to the last byte of **Data**.

Fail: **Header + Length + NOT\_of\_Instruction + (Checksum)**

- **Header:** 0xAB
- **Length:** 1 byte, all the bytes from **Length** filed to the last byte of **Data** field, usually it is 0x03
- **NOT\_of\_Instruction:** 1 byte, the value of NOT operation over the related Instruction code.
- **Checksum:** 1 byte, optional, value of XOR operation over all the bytes from the **Length** byte to the last byte of **Data**.

### Command Description

**Note:** In description of some basic commands, we supply examples. Examples are all without checksum. [Testing Tool](#)

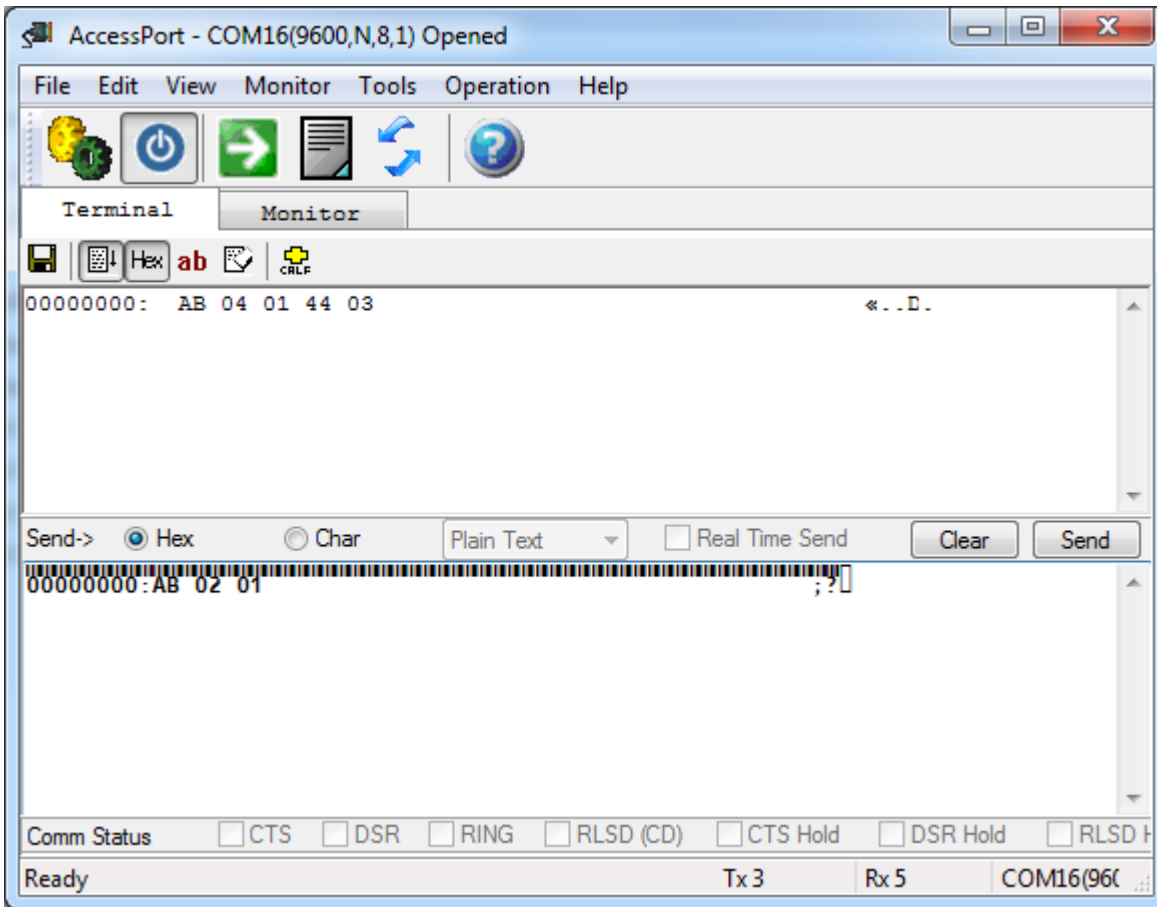
#### 1. Read the card type: 0x01

Instruction	Description	Format	Parameter
0x01	Read the card type	Command: <b>AB 02 01</b> Respond: Success: <b>AB 04 01 [Card_Type] (Checksum)</b> Fail: <b>AB 02 FE (FC)</b>	<b>Card_Type:</b> 0x4400 //Mifare_UltraLight 0x0400 //Mifare_One (S50) 0x0200 //Mifare_One (S70) 0x0800 //Mifare_Pro (X) 0x4403 // Mifare_DESFire

#### Example

Send: **AB 02 01**

Return: **AB 04 01 04 03** //Card type is Mifare\_DESFire



**Tips:** while sending this command, the card has to be in reading area. This means, this operation runs successful only while this module “knows” a card near it. Although the reader could not read ID of Mifare Pro, it could read the card type.

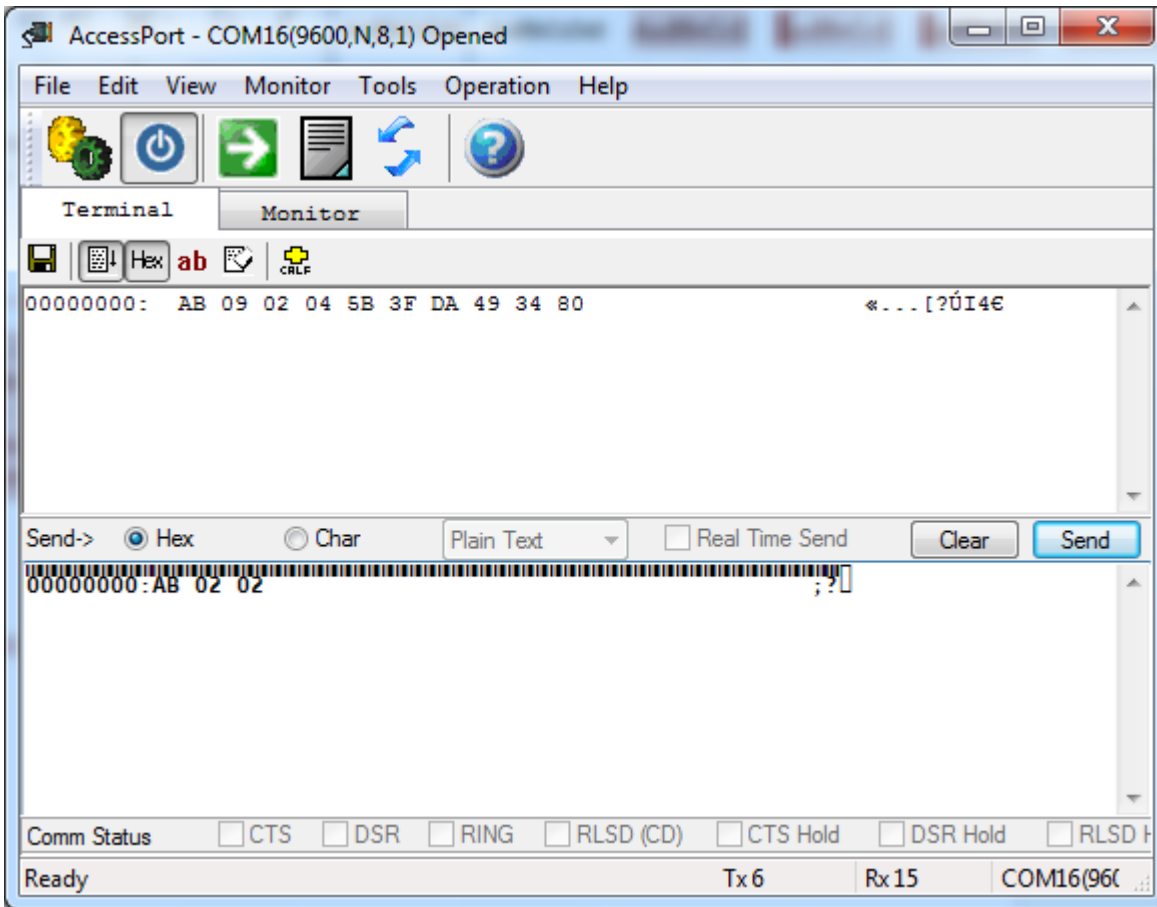
**2. Read the card serial number: 0x02**

Instruction	Description	Format	Parameter
0x02	Read the card serial number	Command: AB 02 02 (00) Respond: Success: AB 06 02 [Serial Number] (Checksum) Fail: AB 02 FD (FF)	Serial Number: 4-byte serial number of the card

**Example**

Send: AB 02 02

Return: AB 09 02 04 5B 3F DA 49 34 80 // Card NO. is 04 5B 3F DA 49 34 80



**Tips: while sending this command, the card has to be in reading area. This means, this operation runs successful only while this module “knows” a card near it.**

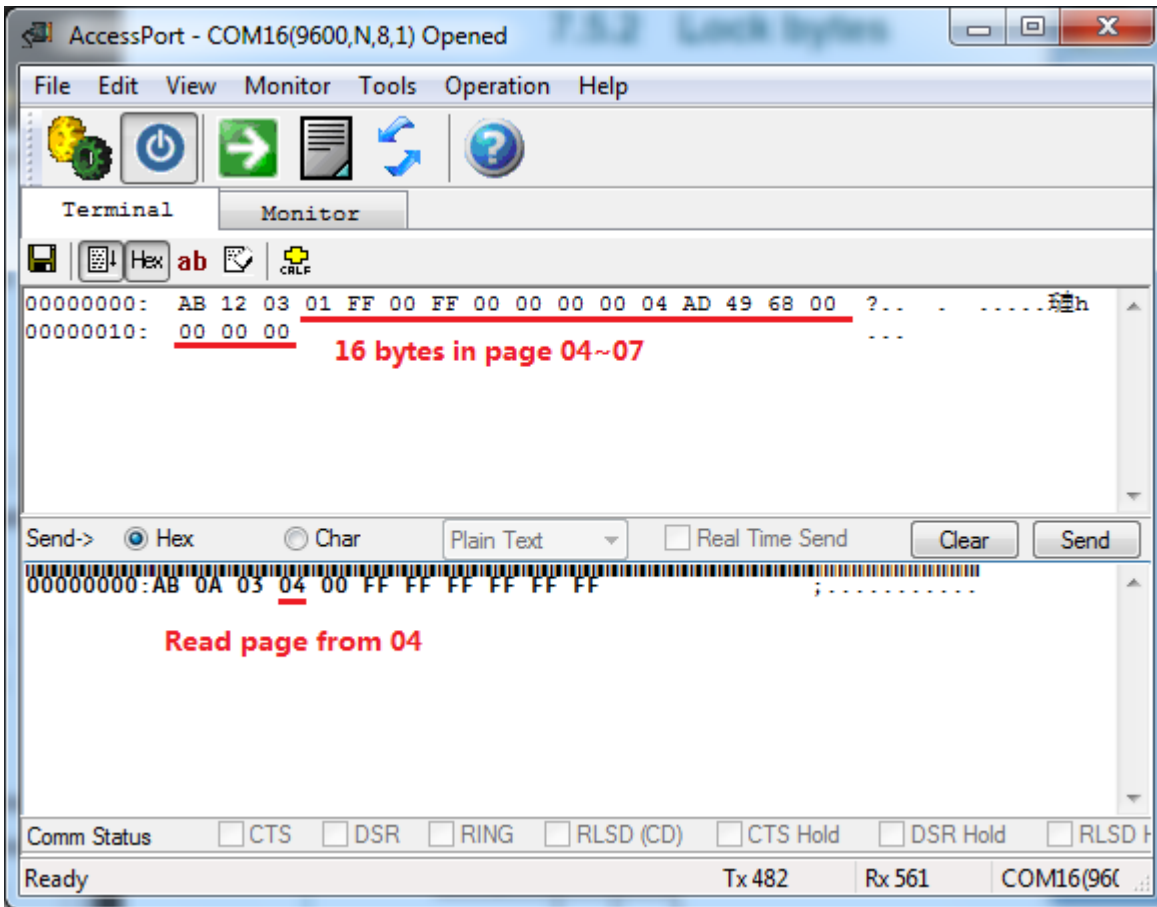
### 3. Read data in a block/page of the card: 0x03

Instruction	Description	Format	Parameter
0x03	Reads the data in the blocks/pages	Command: <b>AB 0A 03 [Block/page Number] [Key type] [Key] (Checksum)</b> Respond: Success: <b>AB [Length] 03 [Data] (Checksum)</b> Fail: <b>AB 02 FC</b>	<b>Block/page Number:</b> Mifare 1 S50: 0~63, 16 bytes per block Mifare 1 S70: 0~255, 16 bytes per block Mifare Ultralight: 4-15, 4 bytes per page Mifare Ultralight C: 4-39, 4 bytes per page. <b>Key type:</b> 0x00 //A type 0x01 //B type <b>Key:</b> authorization key, 6 bytes <b>Data:</b> the data in that block/page, 16 bytes

#### Example

Send: **AB 0A 03 04 00 FF FF FF FF FF FF**

Return: **AB 12 03 [Block/page Data]**



**Tips:**

- For a new tag, the Key is 0xFFFFFFFF. Not every block/page of the card can be read. Please refer to the Mifare's datasheet.
- For Mifare Ultralight (C) tags, each block/page contains 4 bytes. Each time the command could read 16 bytes (4 blocks/pages) once.
- For Mifare Ultralight (C) tags, block/page 0~3 could be read. But they are not user pages. Refer to tag datasheet for more information.
- For Mifare Ultralight tag, ignore key type and key data. Fill them with any data. For Mifare Ultralight C tag, ignore key type data. Fill it with any data.



4. Write data to a block/page of the card: 0x04

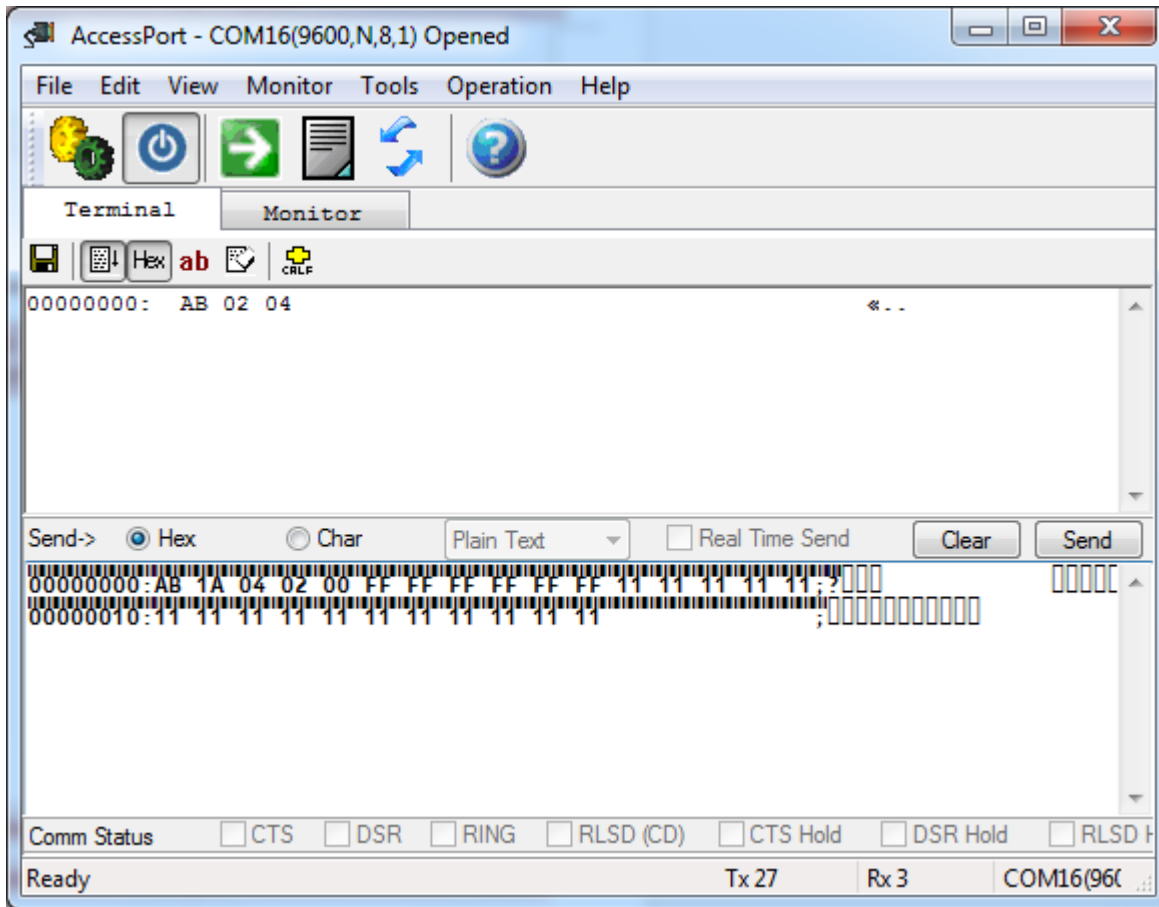
Instruction	Description	Format	Parameter
0x04	Write to the blocks/pages	Command: AB [Length] 04 [Block/page Number] [Key type] [Key] [Data] (Checksum) Respond: Success: AB 02 04 (06) Fail: AB 02 FB (F9)	<b>Length:</b> 1A/0E <b>Block/page Number:</b> Mifare 1 S50: 0~63, 16 bytes per block Mifare 1 S70: 0~255, 16 bytes per block Mifare Ultralight: 4-15, 4 bytes per page Mifare Ultralight C: 4-39, 4 bytes per page. <b>Key type:</b> 0x00 //A type 0x01 //B type <b>Key:</b> authorization key, 6 bytes <b>Data:</b> the data in that block/page, 4 bytes (S50/S70) or 16 bytes (Ultralight or Ultralight C)

Example

Send: AB 1A 04 02 00 ff ff ff ff ff ff 00 ff 00 ff 00 ff 00 ff 00 ff 00 ff 00 ff

Return: AB 02 04

For S50/S70 tags, write 16 bytes data once:



For Ultralight/Ultralight C tags, write 4 bytes data once.

There are 2 ways to write the data:

AccessPort - COM16(9600,N,8,1) Opened

File Edit View Monitor Tools Operation Help

Terminal Monitor

00000000: AB 02 04 <<..

00000000: AB 1A 04 04 00 FF FF FF FF FF FF 11 11 11 11 11 ;?[] [][]

00000010: 11 11 11 11 11 11 11 11 11 11 11 ;:[] [] [] [] [] [] [] []

Send-> Hex Char Plain Text Real Time Send Clear Send

The 4 bytes above read line would be valid

The 12 bytes above yellow line would be ignored

Comm Status CTS DSR RING RLSD (CD) CTS Hold DSR Hold RLSD H

Ready Tx 27 Rx 3 COM16(9600,N,8,1)

AccessPort - COM16(9600,N,8,1) Opened

File Edit View Monitor Tools Operation Help

Terminal Monitor

00000000: AB 0E 04 04 00 FF FF FF FF FF FF 33 33 33 33 ;?[] [][]

00000010: 3333

Send-> Hex Char Plain Text Real Time Send Clear Send

Comm Status CTS DSR RING RLSD (CD) CTS Hold DSR Hold RLSD H

Ready Tx 97 Rx 13 COM16(9600,N,8,1)

**Tips:**

- Be careful while writing to the card. User page/block is safe to write. But for some page/block, it is very dangerous to write. Please refer to tag's datasheet for more information.

**5. Initialize the wallet: 0x05 (for S50/S70 only)**

Command	Description	Format	Parameter
<b>0x05</b>	Initialize wallet, set a specified number (money amount) in the specified block/page	Command: <b>AB 0E 05 [Block/page Number] [Key type] [Key] [Value] (Checksum)</b> Respond: Success: <b>AB 02 05 (07)</b> Fail: <b>AB 02 FA (F8)</b>	<b>Block/page Number:</b> 0~63 (S50) and 0~255(S70) <b>Key type:</b> 0x00 //A type 0x01 //B type <b>Key:</b> authorization key, 6 bytes <b>Value:</b> money amount, 4 bytes, Low Byte first, High byte last

**Example**

Send: **AB 0E 05 02 00 ff ff ff ff ff 00 ff 00 ff** //initial amount is 0xff00ff00

Return: **AB 02 05**

**Tips: Usually we take the value as a 4-byte unsigned int. If you take this value as signed 4-byte int, please remember it is always the complement code.**

**6. Recharge wallet: 0x06 (for S50/S70 only)**

Instruction	Description	Format	Parameter
<b>0x06</b>	increase value in the specified block/page	Command: <b>AB 0F 06 [Block/page Number] [Key type] [Key] [Value] (Checksum)</b> Respond: Success: <b>AB 02 06 (04)</b> Fail: <b>AB 02 F9 (FB)</b>	<b>Block/page Number:</b> 0~63 (S50) and 0~255(S70) <b>Key type:</b> 0x00 //A type 0x01 //B type <b>Key:</b> authorization key, 6 bytes <b>Value:</b> money amount, 4 bytes, Low Byte first, High byte last

**Example**

Send: **AB 0E 06 02 00 ff ff ff ff ff 00 00 00 01**

Return: **AB 02 06**

**7. Deduct from wallet: 0x07(for S50/S70 only)**

Instruction	Description	Format	Parameter
<b>0x07</b>	Reduce value in the specified block/page	Command: <b>AB 0E 07 [Block/page Number] [Key type] [Key] [Value] (Checksum)</b> Respond: Success: <b>AB 02 07 (05)</b> Fail: <b>AB 02 F8 (FA)</b>	<b>Block/page Number:</b> 0~63 (S50) and 0~255(S70) <b>Key type:</b> 0x00 //A type 0x01 //B type <b>Key:</b> authorization key, 6 bytes <b>Value:</b> money amount, 4 bytes, Low Byte first, High byte last

**Example**

Send: **AB 0E 07 02 00 ff ff ff ff ff 00 00 00 01**

Return: **AB 02 07**

**Tips: Always read the wallet to check the balance before you do the deduction.**

### 8. Read wallet: 0x08 (for S50/S70 only)

Instruction	Description	Format	Parameter
<b>0x08</b>	Read value in the specified block/page	Command: <b>AB 0A 08 [Block/page Number] [Key type] [Key] (Checksum)</b> Respond: Success: <b>AB 06 08 [Value] (Checksum)</b> Fail: <b>AB 02 F7 (F5)</b>	<b>Block/page Number:</b> 0~63 (S50) and 0~255(S70) <b>Key type:</b> 0x00 //A type 0x01 //B type <b>Key:</b> authorization key, 6 bytes <b>Value:</b> money amount, 4 bytes, Low Byte first,High byte last

#### Example

Send: **AB 0A 08 02 00 ff ff ff ff ff**

Return: **AB 06 08 [value] (Checksum)**

### 9. Read EEPROM: 0x09

Instruction	Description	Format	Parameter
<b>0x09</b>	Read data from specified address in EEPROM	Command: <b>AB 05 09 [Address] [Data_Length] (Checksum)</b> Respond: Success: <b>AB [Data_Length+2] 09 [Data] (Checksum)</b> Fail: <b>AB 02 F6 (F4)</b>	<b>Address:</b> 2 bytes, High byte First <b>Data_Length:</b> the byte number to read <b>Data:</b> 4 bytes, the reply data in that address

#### Example

Send: **AB 05 09 00 00 04 //4 bytes data**

Return: **AB 06 09 [Data] (4 bytes) (Checksum)**

**Tips: The EEPROM is the data area of the microcontroller, which is 8K. For the Data\_Length field is 1 byte. So this command can read max 255 bytes data once.**

### 10. Write to EEPROM: 0x0A

Instruction	Description	Format	Parameter
<b>0x0A</b>	Write data to EEPROM	Command: <b>AB [Data_Length+5] 0A [Mode] [Address] [Data] (Checksum)</b> Respond: Success: <b>AB 02 0A (08)</b> Fail: <b>AB 02 F5 (F7)</b>	<b>Data_Length:</b> the byte number to write <b>Mode:</b> <ul style="list-style-type: none"> <li>0x00 //normal writing</li> <li>0x01 //compulsive writing</li> </ul> <b>Address:</b> 2 bytes, High byte First <b>Data:</b> the data to write

You can write to EEPROM in two ways: **Normal Writing** or **Compulsive Writing**. By **Normal Writing**, writing is refused if the addresses already have data in it (any data but 0xFF). You have to erase the sector first if the addresses are already written. By **Compulsive Writing**, data can be written to the addresses no matter if the addresses already have data.

#### Example

Send: **AB 09 0A 00 00 01 02 03 04 07**

Return: **AB 02 0A**

**Warning:**

- Be careful while using compulsive writing. All the data (except the config data) is unprotected, which means you can change the data in all addresses with this command. We recommend that check the status of that address before writing to it.
- The addresses 0x0200 and 0x0201 save the config data. Those 2 addresses are protected. Writing to them will fail.
- Data can't be written into 2 Sectors by one command. You should make sure the addresses in one command are all in the same Sector. If in 2 Sectors, writing will fail. And no data was written in.

**11. Erase EEPROM: 0x0B**

<i>Instruction</i>	<i>Description</i>	<i>Format</i>	<i>Parameter</i>
<b>0x0B</b>	Erase data in specified sector of EEPROM	Command: <b>AB 03 0B [Sector_Number] (Checksum)</b> Respond: Success: <b>AB 03 0B 02 (09)</b> Fail: <b>AB 02 F4 (F6)</b>	<b>Sector_number</b> : The sector number of EEPROM

**Example**

Send: **AB 03 0B 02**

Return: **AB 02 0B**

**Tips:**

- The addresses 0x0200 and 0x0201 in Sector 2 save the config data. Erasing Sector 2 will not delete data in those two addresses.
- This module has 16 sectors with each sector 512 bytes.

<i>Sector NO.</i>	<i>Address Range</i>
<b>1</b>	0x0000 ~ 0x01FF
<b>2</b>	0x0200 ~ 0x03FF
<b>3</b>	0x0400 ~ 0x05FF
<b>4</b>	0x0600 ~ 0x07FF
<b>5</b>	0x0800 ~ 0x09FF
<b>6</b>	0x0A00 ~ 0x0BFF
<b>7</b>	0x0C00 ~ 0x0DFF
<b>8</b>	0x0E00 ~ 0x0FFF
<b>9</b>	0x1000 ~ 0x11FF
<b>10</b>	0x1200 ~ 0x13FF
<b>11</b>	0x1400 ~ 0x15FF
<b>12</b>	0x1600 ~ 0x17FF
<b>13</b>	0x1800 ~ 0x19FF
<b>14</b>	0x1A00 ~ 0x1BFF
<b>15</b>	0x1C00 ~ 0x1DFF
<b>16</b>	0x1E00 ~ 0x1FFF

**12. Check status of EEPROM: 0x0C**

<b>Instruction</b>	<b>Description</b>	<b>Format</b>	<b>Parameter</b>
<b>0x0C</b>	Check if the specified address in EEPROM of is already written	Command: <b>AB 05 0C [Address] [Data_Length] (Checksum)</b> Respond: Unwritten: <b>AB 02 0C (0E)</b> Written: <b>AB 02 F3 (F1)</b>	<b>Address:</b> 2 bytes, High byte First <b>Data_Length:</b> The address number to be checked.

**Example**

Send: **AB 05 0C 00 00 04** //check addresses 0x0000~0x0003

Return: **AB 02 0C**

**Tips: if the data in the addresses are 0xFF, those addresses are regarded as unwritten.**

**13. Set the checksum in Basic Command: 0x0D**

<b>Instruction</b>	<b>Description</b>	<b>Format</b>	<b>Parameter</b>
<b>0x0D</b>	Add or remove checksum of Basic Command	Command: <b>AB 03 0D [Value]</b> Respond: Success: <b>AB 02 0D</b> Fail: <b>AB 02 F2</b>	<b>Value:</b> <b>0x00</b> // No checksum <b>0x01</b> // With checksum

**Example**

Send: **AB 03 0D 00**

Return: **AB 02 0D**

**Tips: this command has no checksum in any time.**

**14. Set the baud rate: 0x0E**

<b>Instruction</b>	<b>Description</b>	<b>Format</b>	<b>Parameter</b>
<b>0x0E</b>	Set the baud rate	Command: <b>AB 03 0E [Number] (Checksum)</b> Respond: Success: <b>AB 02 0E (0C)</b> Fail: <b>AB 02 F1 (F3)</b>	<b>Number:</b> See the table below

**Example**

Send: **AB 03 0E 05** // set the baud rate of 19200

Return: **AB 02 0E**

**Baud Rate**

<b>Number (HEX)</b>	<b>Baud Rate (bps)</b>
<b>0x01</b>	2400
<b>0x02</b>	4800
<b>0x03</b>	9600
<b>0x04</b>	14400
<b>0x05</b>	19200
<b>0x06</b>	28800

0x07	38400
0x08	57600
0x09	115200

**15. Restore the default configuration: 0x0F**

Instruction	Description	Format	Parameter
0x0F	Restore the default configuration: <ul style="list-style-type: none"> <li>No checksum</li> <li>9600bps</li> </ul>	Command: AB 02 0F (Checksum) Respond: Success: AB 02 0F (0D) Fail: AB 02 F0 (F2)	NC

**Example**

Send: **AB 02 0F**

Return: **AB 02 0F**

**16. Set the module in standby mode**

Instruction	Description	Format	Parameter
0x10	Exit from executing any command and wait for new command.	Command: AB 02 10 (Checksum) Respond: Success: AB 02 10 (12) Fail: AB 02 EF (ED)	NC

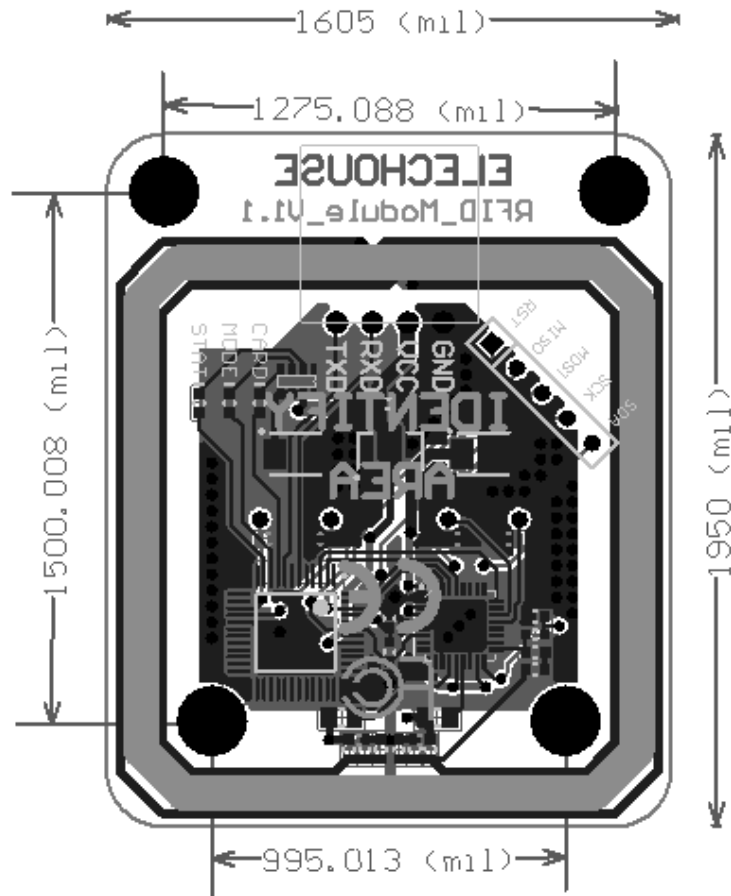
Some commands such as **0x01** will occupy this module until next command is received. This command will release the module and let it in standby mode, working like reset function but data will be not lost.

**Example:**

Send: **AB 02 10**

Return: **AB 02 10**

**Size and Drawing**



### Code Example of Checksum

Here we supply code example of adding checksum and verifying command by checksum.

```
/*  
Function: add checksum for basic commands  
Parameters: the basic commands without checksum  
*/  
  
void AddChkCode (unsigned char * Cmd)  
{  
    unsigned char xorRes = Cmd [1]; // the result of XOR  

```



Parameters: the basic commands with checksum

Returns: check correct return 1. Parity error, it returns 0.

```

*/
unsigned char ChkCmd (unsigned char * Cmd)
{
uchar i;
uchar xorRes = Cmd [1];
for (i = 0; i <Cmd [1] -1; i ++ )
xorRes = xorRes ^ Cmd [i +2];
if (xorRes == Cmd [Cmd [1] +1])
return 1;
else
return 0;
}

```

### Example

```

void main ()
{
unsigned char cmd1 [4] = {0xAB, 0x02, 0x01}; // store the basic command 1, Card type, no checksum
AddChkCode (cmd1); // add basic instruction a check code
ChkCmd (recCmd); // check the received command school
}

```

### Reference information

To understand how to write to Mifare cards, you may need more information about the structure of S50 and S70. And if you use the SPI interface, you may need MFRC522 datasheet.

- ❖ [Mifare S50](#)
- ❖ [Mifare S70](#)
- ❖ [Ultralight C](#)
- ❖ [Ultralight](#)
- ❖ [MFRC522 Datasheet](#)

### Disclaimer and Revisions

The information in this document may change without notice. Please visit [www.elechouse.com](http://www.elechouse.com) for new information.

Revision History

Rev.	Date	Author	Description
A	Nov. 22nd, 2011	Wilson Shen	Initial version
B	Jan. 11th, 2015	Wilson Shen	Add supporting to Ultralight or Ultralight C