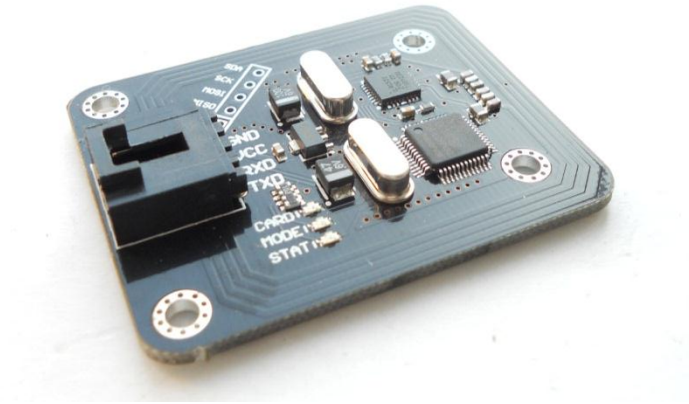
SSRFIDV1.0 Manual



Instruction

This RFID reader module is based on MFRC522, supporting the ISO14443 standard. It supports ISO 14443A/MIFARE mode and MIFARE Classic (e.g. MIFARE Standard) products.

This module has UART interface. Users don't have to consider the complex control algorithms within the RF unit, 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_Pro, 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*
- *Typical operating distance in Reader/Writer mode for communication to a ISO/IEC14443A / MIFARE is 40~50mm*
- *With the chip ISO14443, it supports MIFARE standard encryption algorithm.*
- *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.*
- *ROSH: lead-free*

Parameter

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

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 descripted 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 ^ 0C ^ 00 ^ 00 ^ 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.*

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

Instruction	*Description*	*Format*	*Parameter*
0x01	*Read the card type*	*Command:* ***AB 02 01*** *Respond:* Success: ***AB 04 01 [Card_Type] (Checksum)*** Fail: ***AB 02 FE (FC)***	***Card_Type:*** *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 00*** //Card type is Mifare_One (S50)

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.

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 06 02 DE CE C9 61*** // Card NO. is DE CE C9 61

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 of the card: 0x03***

Instruction	*Description*	*Format*	*Parameter*
0x03	*Reads the data in the blocks (0-63) of the card*	*Command:* ***AB 0A 03 [Block Number] [Key type] [Key] (Checksum)*** *Respond:* Success: ***AB [Length] 03 [Data] (Checksum)***	***Block Number****: 0~63 (S50) and 0~255(S70)* ***Key type****: 0x00 //*A type *0x01 //*B type

	Fail: ***AB 02 FC***	***Key***: *authorization key, 6 bytes* ***Data***: *the data in that block, 16 bytes.*

Example

Send: ***AB 0A 03 02 00 FF FF FF FF FF FF***

Return: ***AB 12 03 [Block data]***

Tips: For a new card, the Key is 0xFFFFFF**. Not every block of the card can be read. Please refer to the Mifare's datasheet.**

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

Instruction	*Description*	*Format*	*Parameter*
0x04	*Write to the blocks (0-63) of the card*	*Command:* ***AB 1A 04 [Block Number] [Key type] [Key] [Data] (Checksum)*** *Respond:* Success: ***AB 02 04 (06)*** Fail: ***AB 02 FB (F9)***	***Block Number***: *0~63 (S50) and 0~255(S70)* ***Key type***: *0x00* //A type *0x01* //B type ***Key***: *authorization key, 6 bytes*

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 00 ff***

Return: ***AB 02 04***

Tips:

- **Not every block can be written. Please refer to the Mifare's datasheet.**
- **Data filed should be 16 bytes. If the Data is less than 16 bytes, checksum or even part of next command will be written into blocks as data.**

5. ***Initialize the wallet: 0x05***

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

Example

Send: ***AB 0E 05 02 00 ff 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***

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

Example

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

Return: **AB 02 06**

7. **Deduct from wallet: 0x07**

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

Example

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

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

Example

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

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

9. **Read EEPROM: 0x09**

Instruction	Description	Format	Parameter
0x09	Read data from specified address in EEPROM	Command: **AB 05 09 [Address] [Data_Length] (Checksum)** Respond: Success: **AB [Data_Length+2] 09 [Data] (Checksum)** Fail: **AB 02 F6 (F4)**	**Address**: 2 bytes, High byte First **Data_Length**: the byte number to read **Data**: 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 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*
0x0A	*Write data to EEPROM*	*Command:* ***AB [Data_Length+5] 0A [Mode] [Address] [Data] (Checksum)*** *Respond:* Success: ***AB 02 0A (08)*** Fail: ***AB 02 F5 (F7)***	***Data_Length****: the byte number to write* ***Mode:*** • *0x00* //normal writing • *0x01* //compulsive writing ***Address****: 2 bytes, High byte First* ***Data****: 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

Instruction	*Description*	*Format*	*Parameter*
0x0B	*Erase data in specified sector of EEPROM*	*Command:* ***AB 03 0B [Sector_Number] (Checksum)*** *Respond:* Success: ***AB 03 0B 02 (09)*** Fail: ***AB 02 F4 (F6)***	***Sector_number****: 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.**

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

12. *Check status of EEPROM: 0x0C*

Instruction	*Description*	*Format*	*Parameter*
0x0C	*Check if the specified address in EEPROM of is already written*	*Command:* ***AB 05 0C [Address] [Data_Length] (Checksum)*** *Respond:* Unwritten: ***AB 02 0C (0E)*** Written: ***AB 02 F3 (F1)***	***Address**: 2 bytes, High byte First* ***Data_Length**: 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*

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

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

Example

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

Return: ***AB 02 0E***

Baud Rate

Number (HEX)	*Baud Rate (bps)*
0x01	*2400*
0x02	*4800*
0x03	*9600*
0x04	*14400*
0x05	*19200*
0x06	*28800*
0x07	*38400*
0x08	*57600*
0x09	*115200*

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

Instruction	*Description*	*Format*	*Parameter*
0x0F	*Restore the default configuration:* • *No checksum* • *9600bps*	*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***

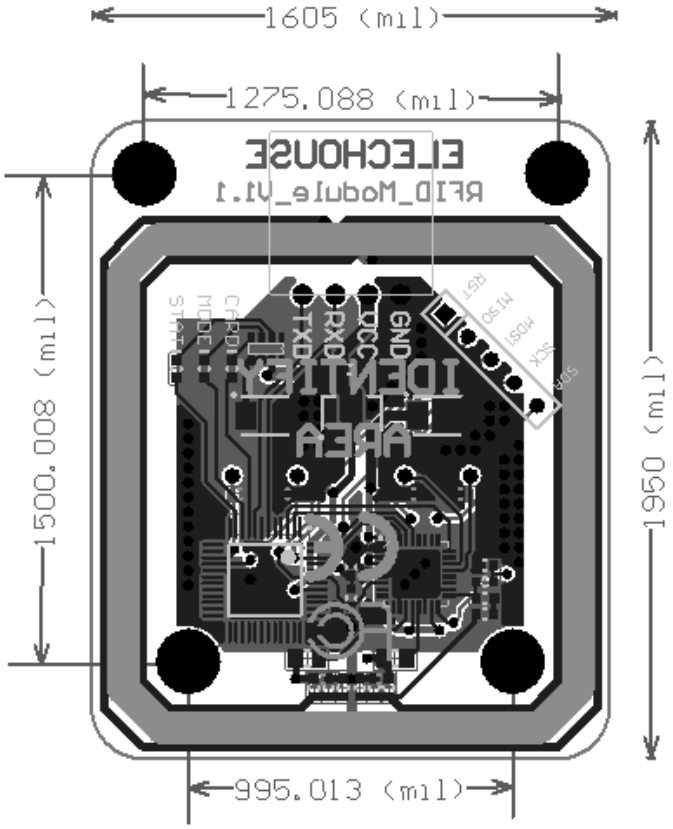
Some commands such as ***0x01*** *will occupy this module until next command is received. This command will release the module and let it in standby mode, working like reset function but data will be not lost.*

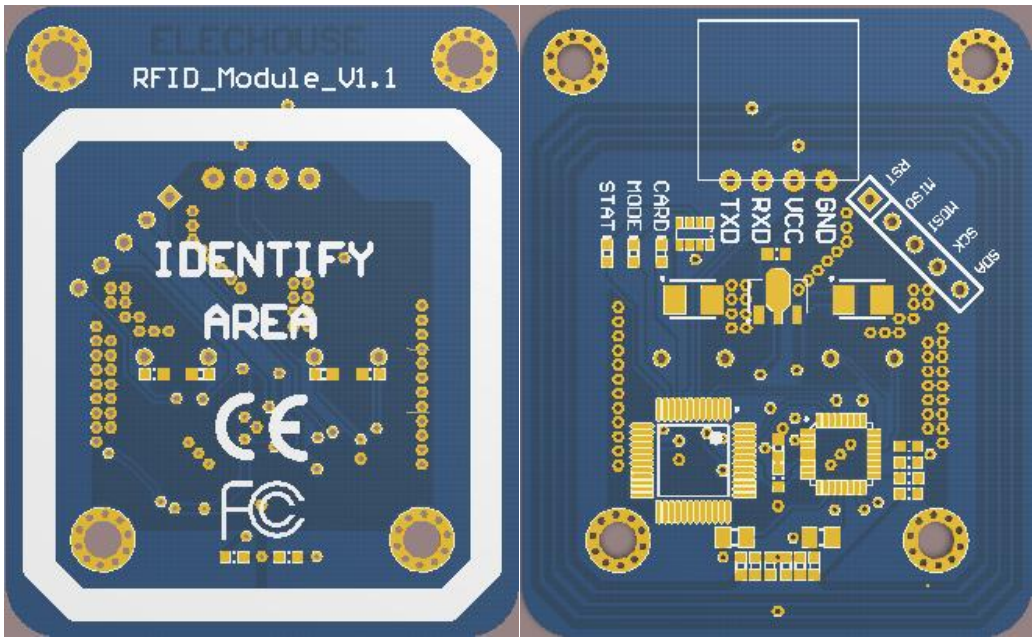
Example:

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
unsigned char i;
for (i = 0; i <Cmd [1] -1; i + +)
{
xorRes = xorRes ^ Cmd [i +2]; / / XOR on from the Length field to the last byte of data
}
Cmd [Cmd [1] +1] = xorRes;
}


/ *
Function: verify the checksum of basic commands
Parameters: the basic commonds 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*
- *MFRC522 Datasheet*

Disclaimer and Revisions

The information in this document may change without notice. Please visit www.elechouse.com for new information.
Revision History

Rev.	Date	Author	Description
A	Nov. 22nd, 2011	Wilson Shen	Initial version