S9 Contactless Smart Card Reader

Reference Manual

Contents

1. S9 Contactless IC Card Reader Introduction

1.1 Overview

S9 series Dual-Interface Smart Card Reader Writer can support both contact and contactless smart cards, with one slot for smart card of ISO7816 Size and Maximum 3 slots for SAM card of GSM11.11 size. With multi-slots structure, the reader can meet the higher level of security in smart card application. It connects to PC or related devices via RS232 Serial port or USB (PNP, plug and play) port which brings more convenience to customers whether in installation or operation of the reader. And there will be a CD for SDK (Software Development Kits) along with the reader which includes the USB Drivers and Examples for various development platforms, and customers can use the DEMO.exe program to do the testing for rfid card and reader.

S9 Reader is essential front-end processing equipment for IC card applications and system integration. Owning rich and useful interfaces and functions, the reader can be easily applied to many fields such as industry and commerce, telecommunications, postal services, taxation, banking, insurance, medical, meeting attendance, Internet cafe management, gas stations, parking lots and other smart card application programs for charges, stored value and query.

1.2 Features

Card type supported:
—contactless card: ISO14443 TypeA/B,ISO15693 standard
—contact card: ISO7816 logic encryption IC cards and the CPU card(T=0,1)

Security: can add 3 GSM11.11SAM card slots

Interface: RS232/USB (Plug and Play)

Antenna: Built-in antenna

Case Material: ABS plastic

Operating Frequency: 13.56MHz

Operating voltage: DC5V ± 10%

Communication speed between Card and equipment: 106Kbit

Operating temperature: -20 ℃ ~ 60 ℃

Standards followed: ISO14443 TypeA/B, ISO15693, ISO7816,GSM11.11, FCC, CE

Programming languages:
a variety of programming languages available, with luxuriant Examples (refer to "1.5 Software" for more information)

OS: Windows 98, Me, 2K, XP, 2003, Vista ,Unix and Linux

Memory of reader: the standard internal memory of reader is 2K bytes (can be expanded upon customers' requirements)

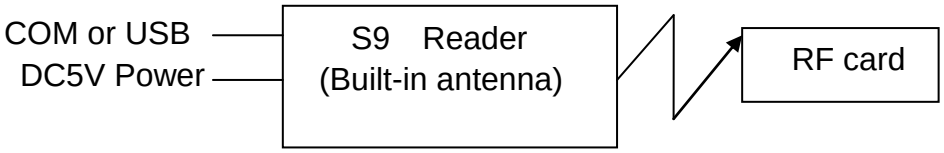
Status display: LED lights, indicating power or the state of communication, user controllable buzzer

Connection cable: length 1.5M, can choose serial K Line or usb cable

Dimensions(L*W*H): 123 * 95 * 27 mm

NW/GW: about 143g / 345g

1.3 Device Interface



RS232 Serial Interface or USB been used for communicating with PC.

1.4 Reader Packing List

Contains: Reader, Communication cable, CD for driver, and Warranty Card.

1.5 Software

Contains: Demo program, Lib of Function, Application Examples

a. Demo program

S9.exe

b. Lib of Fuction

Under Win32-dll directory

c. Application Examples

There are many examples supported for varivious of development platform,(such as VB,DELPHI,VC,C#,JAVA etc.) under the directory of EXAMPLES.

1.6 Typical Application

E-passports / Internet banking and online shopping / network access and access control systems / digital signature / customer bonus plan / stored value / Identification / e-ticket / parking lots/ Member payment / Time attendance / vending machine.

1.7 Reader Type Description

Naming rule **: S9-X X-X X-XX**

The first two(S9)：represent S9 series Reader，contain RS232 serial port and USB port,Support both contact and contactless smart card，without a display screen.

The third place： represent supportable card type , "A" represent to support S50、S70 and Ultralight card, "B" support 14443A card , "C" support 14443A 、14443B and 15693 card，“D” support 125K card compatible with ID card number，“E” support S50 card number，“F” support S50 card and 125K ID card number.

The forth：represent communication mode，“S” represent serial port communication，“U” represent USB communication，“E” represent virtual USB(install the USB driver) communication ，“P” represent PC/SC standard.

The fifth： represent the number of big card deck, “0” represent none，“1” represent to have a big card deck.

The sixth： represent the number of SIM card deck, “0” represent none，“1” represent to have a SIM card deck.

The last two：arabic number，starting numbering from 1 in the order.

1.8 Function Description

Function call should follow the following rules:

（**1**）Call function fw_init() first to initial Serial Interface or USB interface.

（**2**）Call function fw_load_key() to load key from card to Reader, a section once.

（**3**）Call funtion fw_card() (equivalent to continuous calls fw_request (), fw_anticoll (), fw_select () 3 functions) to get card serial number.

（**4**）Call function fw_authentication () to verify keys of Device and Card, once a section.

（**5**）You can read, write, initialize value, add value, and subtract value to the sections ONLY after verifying the password. Each time you do thesed operations of card, you should repeat the stpes (3), (4) .

（**6**）Afer operation with certain card,the function fw_halt() should be called to abort operation.

（**7**）Before exit program,the function fw_exit had better be called to close communication port.

（**8**）Please refer to the examples under directory Example\ to develop your program.

1.9 API Functions List

Common Functions

Function Name	Description
fw_init	Initialize the communication port
fw_exit	Close the port
fw_card	Find card
fw_card_hex	Find card
fw_card_str	Find card
fw_request	Find card request
fw_anticoll	anti-collision
fw_select	Select a card
fw_load_key	Load keys of sectors
fw_authentication	Authenticate sector key
fw_read	Read card (for S50 and S70 cards)
fw_write	Writer card (for S50 and S70 cards)
fw_halt	Stop the operation for card
fw_des	encrypt/decrypt
fw_changeb3	Change sector password (for S50 and S70 cards)
fw_initval	Initialize the value of block
fw_increment	Do Increment
fw_decrement	Do Decrement
fw_readval	Read value
fw_restore	Store data from the EEPROM to card's internal register
fw_transfer	Transfer the data from card to EEPROM
fw_config_card	set operation-card type
a_hex	convert hex string to the corresponding characters of ordinary ASC
hex_a	Convert ordinary ASC to the corresponding hex string

Device Functions:

Function name	Description
fw_beep	Make beep
fw_disp_mode	Set LED display mode
fw_gettime	Get system time from reader
fw_settime	Set the date/time in Reader
fw_getver	Get the version of Reader
fw_srd_eeprom	Read EEPROM
fw_swr_eeprom	Write EEPROM
fw_reset	Reset card
fw_ctl_mode	Set LED control mode
fw_LED_disp8	Set LED to display random 8 digits
fw_lcd_setbright	LCD light on & off
fw_lcd_dispstr	LCD display string
fw_lcd_dispclear	Clear LCD string

CPU（SAM）card-specific functions

Function Name	Description
fw_cpureset	Power-on reset
fw_setcpu	Set SAM card or CPU card slots for operation
fw_cpuapdu	Information transfer between CPU card and APDU
fw_setcpupara	Set parameters of CPU card

4442 card-specific functions

Function name	Description
fw_authentikey_4442	Authenticate key of 4442 card
fw_read_4442	Read data from 4442 card
fw_write_4442	Write data from 4442 card
fw_getProtectData_4442	Get protected data
fw_setProtectData_4442	Set protected data
fw_changkey_4442	Change key of 4442 card
fw_cntReadError_4442	Get the count of read-error

4428 Card-Specific Functions

Function Name	Description
fw_authentikey_4428	Authenticate key of 4428 card
fw_read_4428	Read data from 4428 card
fw_write_4428	Write data from 4428 card
fw_getProtectData_4428	Get protected data
fw_setProtectData_4428	Set protected data
fw_changkey_4428	Change key of 4428 card
fw_cntReadError_4428	Read counts of Error-Code

S70 Card-Specific Functions

Function Name	Description
fw_read_S70	Read S70 card
fw_write_S70	Write S70 card

Ultralight Card-Specific Functions

Function name	Description
fw_request_ultralt	Send request command to Ultralight card
fw_anticall_ultralt	Anti-collision for Ultralight card
fw_select_ultralt	Select Ultralight card
fw_read_ultralt	Read date from Ultralight card
fw_write_ultralt	Write data to Ultralight card
fw_halt_ultralt	Abort the operation with Ultralight card

Mifare Pro Card-Specific Functions

Function Name	Description
fw_reset_mifarepro	Reset Mifare Pro card
fw_apdu_mifarepro	Information transfer between Mifaere Pro Card and APDU

ICODE2 Card-Specific Functions

Function Name	Description
fw_inventory	Icode2 card request
fw_stay_quiet	Icode2 card stay in quiet state
fw_select_uid	Icode2 card stay in select state
fw_reset_to_ready	Icode2 card stay in ready state
fw_readblock	Read block data from Icode2 card
fw_writeblock	Write data to the block of Icode2 card
fw_lock_block	Lock a certain block
fw_write_afi	Write AFI
fw_lock_afi	Lock AFI
fw_write_dsfid	Write DSFID
fw_lock_dsfid	Lock DSFID
fw_get_systeminfo	Get card information
fw_get_securityinfo	Get card security state information

AT88RF020 Card-Specific Functions

Function Name	Description
fw_request_b	AT88RF020 card request
fw_attrib	Select card from several cards
fw_check_at	Verify a particular card

fw_read_at	Read AT88RF020 card
fw_write_at	Read AT88RF020 card
fw_changekey_at	Change the key of AT88RF020 card
fw_lock_at	Lock card
fw_halt_at	Stop the operation of AT88RF020 card
fw_count_at	Counting function

Contactless CPU (ISO1443) Card-Specific Functions

Function Name	Description
fw_pro_reset	Card reset
fw_pro_commandlink	Information exchange function

24C64 Card-Specific Functions

Function Name	Description
fw_read_24c64	Read data from 24c64 card
fw_write_24c64	Write data to 24c64 card
fw_check_24c64	Check if card exist in reader and check the card type

DESFIRE Card-Specific Functions

Function Name	Description
fw_anticoll2	The second anti-collision
fw_select2	The second finding card
fw_reset_desfire	Reset desifare card
fw_authen_desfire	Key authentication
fw_getver_desfire	Get version(Manufactuing info) of DESFIRE card
fw_getAIDs_desfire	Get application identifier
fw_selectApp_desfire	Select application

fw_getKeySetting_desfire	Get the master key settings
fw_getKeyver_desfire	Get the master key version
fw_createApp_desfire	Creat application
fw_delAID_desfire	Delete application
fw_changeKeySetting_desfire	Change master key setttings
fw_changeKey_desfire	Modify master key
fw_getFileIDs_desfire	Get file identifier
fw_getFileProper	Get file properties
fw_changeFileSetting	Change file settings
fw_createDataFile_desfire	Create a standard data file
fw_createValueFile_desfire	Create value file
w_createCsyRecord_desfire	Create cycle record file
fw_delFile_desfire	Delete file
fw_write_desfire	Write data file
fw_read_desfire	Read record file
fw_getvalue_desfire	Read value file
fw_credit_desfire	increment
fw_debit_desfire	decrement
fw_writeRecord_desfire	Write record
fw_readRecord_desfire	Read record
fw_clearRecord_desfire	Clear record
fw_commitTransfer_desfire	Commit data transmission
fw_abortTransfer_desfire	Abort data transmission
fw_formatPICC_desfire	Format card

125K (ID) Card-Specific Functions

Function Name	Description
fw_read_SerialNumberID	Read SerialNumber of ID card

1.10 Error codes and meanings

Error Ruturn Codes	Positive/Negative	Meanings
0x10(016)	-	Communication rror
0x11(017)	-	Timeout error
0x20(032)	-	Open port error
0x21(033)	-	Get port parameter error
0x22(034)	-	Set port parameter error
0x23(035)	-	Close port error
0x24(036)	-	Port is occupied
0x30(048)	-	Format error
0x31(049)	-	Data format error
0x32(050)	-	Data length error
0x40(064)	-	Read error
0x41(065)	-	Write error
0x42(066)	-	No receiving error
0x50(080)	-	Error that it is not enough for subtraction
0x51(081)	-	CPU data XOR error
0x52(082)	-	Address No error when 485 communicating
0x73(115)	-	Get the version number error
0xc2(194)	-	CPU card response error
0xd3(211)	-	CPU card response time-out
0xd6(214)	-	CPU card verification error
0xd7(215)	-	CPU card command returns error

0x01(001)	+	Not card or certificate error
0x02(002)	+	Data validation errors
0x03(003)	+	Value is null error
0x04(004)	+	Authentication Failed
0x05(005)	+	Parity Error
0x06(006)	+	The reader and card communication error
0x08(008)	+	Read card serial number error
0x09(009)	+	Password type error
0x0a(010)	+	Card has not been certified
0x0b(011)	+	Read bits operation error
0x0c(012)	+	Read bytes operation error
0x0f(015)	+	Write card failed
0x10(016)	+	Value-added operation failed
0x11(017)	+	Impaired operation failed
0x12(018)	+	Read card error
0x13(019)	+	Transfer Buffer Overflow
0x15(021)	+	Transmission frame error
0x17(023)	+	Unknown transmission needs
0x18(024)	+	Anti-collision error
0x19(025)	+	Sensor module reset error
0x1a(026)	+	Non-authentication Interface
0x1b(027)	+	Module communication timeout
0x3c(060)	+	Abnormal(Non-normal) operation
0x64(100)	+	Wrong data
0x7c(124)	+	Parameter error

1.11 Desfire return codes and meanings

Error Codes	Meanings
0x00	success
0x01	No card in operation area
0x02	CRC checksum error
0x03	Numerical overflow
0x05	Parity error
0x06	Communication error
0x08	Read the serial number error in anti-collision process
0x0B	The bits received from card error
0x0C	Backup File not change, do not need CommitTransaction or AbortTransaction
0x0E	Insufficient memory to complete the instruction
0x1C	The command codes are not supported
0x1E	CRC or MAC code not match, the filled bytes are i nvalid
0x40	The specified key number is invalid
0x7E	The length of command string is invalid
0x9D	The current configuration or the state does not allow the request
0x9E	Parameter value is invalid
0xA0	The request application identity does not exist
0xA1	Unrecoverable error in application, the application will be invalid
0xAE	The current authentication state does not allow the request command
0xAF	Expect the data frame re-sent

0xBE	Try to read/write data that beyond the file scope
0xC1	Unrecoverable error within the card, the card will be invalid
0xCD	Card is invalid for unrecoverable error
0xCE	The Maximum application number is 28, can not re-create the application.
0xDE	Can not create the file or application, the file number or application number has been existed.
0xEE	Because of the power-down, the internal backup, the reversal mechanism can not complete the writing operation.
0xF0	Specified File number does not exist
0xF1	Unrecoverable error in file, the file is invalid

2. API Function

2.1 Common Functions

int fw_init(int port,long baud);

Description

Initialize the communication port.

Parameters

port:COM Type.Serial port 1～20 when set value 0～19. Means USB port when set value 100(baud rate invalid in this case).

baud:Baud rate(value:9600～115200)

Return Value

>0 If successful,If unsuccessful,return <0

Example

```
int icdev,commdev;
icdev=fw_init(100,0);//initialize USB interface
commdev=fw_init(0,9600);// initial serial interface, Baud rate:9600
```

Remark

If there are more than one Reader connected to the computer,call this function can get their handle each.Next is an example:

```
int icdev1,icdev2,icdev3;/* presume there are three readers connected*/
icdev1=fw_init(100,0);/*get the first device handle*/
icdev2=fw_init(100,0);/*get the second device handle*/
icdev3=fw_init(100,0);/*get the third device handle*/
```

int fw_exit(int icdev);

Description

Close the communication port.

Parameters

icdev:Value of Device Handle.

Return Value

0 if successful; otherwise, Nonzero.

Example

```
fw_exit(icdev);
```

Remark

In WIN32 environment, Icdev is the handle of device; It must be released before next linking.

int fw_card(int icdev,unsigned char _Mode,unsigned long *_Snr);

Description

Find card, can return the card serial Number in working area. (Contain the next functions :fw_request,fw_anticoll,fw_select)

Parameters

icdev:Value of Device Handle.

_Mode: Model of find card.

Value:

0——IDLE mode,can operate one card once;

1——ALL mode,can operate serval card once;

_Snr:returned Card serial number.

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
unsigned long snr;
st=fw_card(icdev,0,&snr);
```

Remark

1. In IDLE mode, after read-write operations, we use function fw_halt to end the card operation, only when the card out and re-enter the operating area, the reader can operate it once again.
2. When calling this function, we should pay attention to incoming data from the type of last argument, it must be the address of an unsigned long integer variable (unsigned char long), or will be automatically converted into a signed one. Recommended use a function fw_card_hex to return hex card number or a function fw_card_str to return Decimal card number.

int fw_card_hex(int icdev,unsigned char _Mode,unsigned char * Snrbuf);

Description

Find card, Get the card serial Number in working area. (hex string)

Parameters

icdev:Value of Device Handle.

_Mode: Model of find card.

Value:

0——IDLE mode,can operate one card once;

1——ALL mode,can operate serval card once;

_ Snrbuf: the hex string card number returned(8 bytes)

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
unsigned char snr[9]={0};
st=fw_card_hex(icdev,1,snr);
```

int fw_card_str(int icdev,unsigned char _Mode,unsigned char* strSnr);

Description

Find card, Get the card serial Number in working area. (Decimal string)

Parameters

icdev:Value of Device Handle.

_Mode: Model of find card.

Value:

0——IDLE mode,can operate one card once;

1——ALL mode,can operate serval card once;

strSnr: the decimal string card number returned (10 digit sequence)

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
unsigned char snr[11]={0};
st=fw_card_str(icdev,1,snr);
```

int fw_request(int icdev,unsigned char _Mode,unsigned int *TagType);

Description

Find card request.

Parameters

icdev:Value of Device Handle.

_Mode:find card.Mode

Value:

0——IDLE mode,can operate one card once;

1——ALL mode,can operate serval card once;

Tagtype:returned value of Card Type;values mean following

eigenvalue (decimal)	Card type
4	MIFARE ONE (M1)

2	S70
8	MIFARE PRO
68	ULTRA LIGHT

Return Value
0 if successful; otherwise, Nonzero.

Example

```
int st;
unsigned int *tagtype;
st=fw_request(icdev,0,tagtype);
```

int fw_anticoll(int icdev,unsigned char _Bcnt,unsigned long *_Snr);

Description

Preventing Card conflict,return the card Serial number.

Parameters

icdev:Value of Device Handle.

_Bcnt: Shoud be set value 0

_Snr: [out] card serial number

Returned address of the card serial number

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
unsigned long snr;
st=fw_anticoll(icdev,0,&snr);
```

Remark

This function shoud be called immediately after the function fw_request unless we have know the card serial number.

int fw_select(int icdev,unsigned long _Snr,unsigned char *_Size);

int fw_load_key(int icdev,unsigned char _Mode,unsigned char _SecNr,unsigned char *_NKey);

Description

Load keys to RAM of Reader.

Parameters

icdev:Value of Device Handle.

_Mode:The model of key verify.

Value are as follows:

For each sector of M1 card, there are three sets of corresponding password (KEYSET0, KEYSET1, KEYSET2) in the reader, each password include A password (KEYA) and B password (KEYB), a total of six passwords, use 0~2,4 ~ 6 to represent the six Password:

0 - KEYSET0 of KEYA `

1 - KEYSET1 of KEYA

2 - KEYSET2 of KEYA

4 - KEYSET0 of KEYB
5 - KEYSET1 of KEYB
6 - KEYSET2 of KEYB
_SecNr:Section number (M1 Card:0～15; ML Card:0)
_Nkey:.Card key written to the reader

Return Value

0 if successful; otherwise, Nonzero.

Example

```
//key A and key B
unsigned char password[7]={0xa0,0xa1,0xa2,0xa3,0xa4,0xa5};
/*Load key of section 1 */
if((fw_load_key(icdev,0,1,password))!=0)
{
        printf("Load key error!");
        fw_exit(icdev);
}
```

int fw_authentication(int icdev,unsigned char _Mode,unsigned char _SecNr)

Description

Verify Key.

Parameters

icdev:Value of Device Handle.
_Mode:The mode of loading key.Value are as follows:
For each sector of M1 card, there are three sets of corresponding password (KEYSET0, KEYSET1, KEYSET2) in the reader, each password include A password (KEYA) and B password (KEYB), a total of six passwords, use 0~2,4 ~ 6 to represent the six Password:
0 - KEYSET0 of KEYA `
1 - KEYSET1 of KEYA
2 - KEYSET2 of KEYA
4 - KEYSET0 of KEYB
5 - KEYSET1 of KEYB
6 - KEYSET2 of KEYB

_SecNr:The section number to verify.

Return Value

0 if successful; otherwise, Nonzero.

Example

```
//Verify section 4 key with 0 model
if((fw_authentication(icdev,0,4))!=0)
{
        printf("Authentication error!");
}
```

int fw_authentication_pass(int icdev, unsigned char _Mode, unsigned

char Addr,unsigned char *passbuff)

Description

Verify Key function, when use this function, you can not implement function fw_load_key.

Parameters

icdev:Value of Device Handle.

_Mode:The mode of loading key.Value are as follows:

For each sector of M1 card, there are three sets of corresponding password (KEYSET0, KEYSET1, KEYSET2) in the reader, each password include A password (KEYA) and B password (KEYB), a total of six passwords, use 0~2,4 ~ 6 to represent the six Password:

0 - KEYSET0 of KEYA `
1 - KEYSET1 of KEYA
2 - KEYSET2 of KEYA
4 - KEYSET0 of KEYB
5 - KEYSET1 of KEYB
6 - KEYSET2 of KEYB

Addr:The section number to verify.

passbuff: The key to authentication(6 bytes)

Return Value

0 if successful; otherwise, Nonzero.

Example

```
//Verify section 4 key with 0 model
unsigned char password[7]={0xa0,0xa1,0xa2,0xa3,0xa4,0xa5};
if((fw_authentication_pass(icdev,0,4,password))!=0)
{
        printf("Authentication error!");
}
```

int fw_read(int icdev,unsigned char _Adr,unsigned char *_Data);

Description

Read content of card.

For M1 card :Read one block data(16 bytes) once.

For ML card:Read two pages with same property once (0 and 1,2 and 3 ,...), 8 bytes.

Parameters

icdev:Value of Device Handle.

_Adr:M1 Card——Address of block M1 (0～63) ,MS70(0-255);

_Data:[out] data of card.

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
unsigned char data[16];
```

```
st=fw_read(icdev,4,data);  //Read block 4 of M1 card
```

Related HEX function:

int fw_read_hex(int icdev,unsigned char _Adr,char *_Data)

Remark

the difference between HEX function and normal function is, the string of HEX function is in the form of hex, while the string of the corresponding normal function is in the form of ASC codes. For example, if the actual data of second block is: "1234567890abcedf", then called function fw_read_hex will return the string: "31323334353637383930616263646566." The same below.

int fw_write(int icdev,unsingned char _Adr,unsigned char *_Data);

Description

Write data to card.

For M1 card :Write one block data(16 bytes) once.

For ML card :Write one page data(4 bytes)once

Parameters

icdev:Value of Device Handle.

_Adr:M1 Card——Address of block M1 (0～63) ,MS70(0-255);

_Data:data for write

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
unsigned char *data=”1234567890123456”;
st=fw_write(icdev,4,data);//write block 4
```

Related HEX function:

int fw_write_hex(int icdev,unsingned char _Adr,unsigned char *_Data)

Remark

The second parameter of this function is the block number. For the M1 card, the 4th block of each sector is the password block; for S70 card, the 4th blocks of the first 32 sectors , and the 16 th block of the last eight sectors, is the password block; rewriting password block must be carefully, because the sector may damage after the rewrite operation.

int fw_halt(int icdev)

Description

Abort operation of card.

Parameters

icdev:Value of Device Handle.

Return Value

0 if successful; otherwise, Nonzero.

Example

```
st=fw_halt(icdev);
```

Remark

There is a parameter _Mode in function fw_card(),card been setted HALT state when it is setted value 0,the card should be take out the operating area and put in again.

int fw_des(unsigned char *key,unsigned char *sour,unsigned char *dest,__int16 m)

Description

Encrypt or Decrypt with DES algorithm

Parameters

key:secret key

sour:source of data for encrypt/decrypt

dest:out data after encrypt/decrypt

m:model of encrypt/decrypt,encrypt when m=1; decrypt when m=0

Return Value

0 if successful; otherwise, Nonzero.

int fw_changeb3(int icdev,unsigned char _SecNr,unsigned char *_KeyA, unsigned char *_CtrlW,unsigned char _Bk,unsigned char *_KeyB);

Description

Update data of block 3(update data of block 15 if card type is S70 and section number >31)

Parameters

icdev:Value of Device Handle.

_SecNr:Section number (M1:0～15,M1S70:0-39)

_KeyA:Key A

_CtrlW:Control Word of key

_Bk:Set value 0

_KeyB:Key B

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
unsigned char keya;
unsigned char keyb;
unsigned char ctrlword={0xff,0x07,0x80,0x69};
memset(keya,0xff,6);
memset(keyb,0xff,6);
st=fw_changeb3(icdev,1,keya, ctrlword,0,keyb);/*change the key of sector 1*/
```

int fw_initval(int icdev,unsigned char _Adr,unsigned long _Value);

Description

Initial value of block.

Parameters

icdev:Value of Device Handle.
_Adr:Address of block
_Value:value of block for seting

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
unsigned long value;
value=1000;                    /*set value  as 1000*/
st=fw_initval(icdev,1,value);        /*Initial value of block 1 as 1000*/
```

Remark

During value operation, you must first initialize the value function, and then make the others read, increment, decrement operation.

int fw_increment(int icdev,unsigned char _Adr,unsigned long _Value);

Description

Increment value of certain block.

Parameters

icdev:Value of Device Handle.
_Adr:Address of block
_Value:Value for increment

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
unsigned long value;
value=10;
st=fw_increment(icdev,1,value);  /*Increment 10 to the value of block 1*/
st=fw_transfer(icdev,1);
```

Remark

After call this function,the function fw_transfer should be called immediately, otherwise,the value will not be updated.

fw_readval(int icdev,unsigned char _Adr,unsigned long *_Value);

Description

Read value of certain block.

Parameters

icdev:Value of Device Handle.
_Adr:Address of block
_Value:[out]Value for read

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
unsigned long value;
st=fw_readval(icdev,1,&value);     / * Read out the value of block 1, and
```

```
assign to value * /
```

int fw_decrement(int icdev,unsigned char _Adr,unsigned long _Value);

Description

Decrement value of certain block.

Parameters

icdev:Value of Device Handle.

_Adr:Address of block

_Value:Value for decrement

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
unsigned long value;
value=10;
st=fw_decrement(icdev,1,value);    /*reduce value to the value of block 1 */
st=fw_transfer(icdev,1);
```

Remark

After call this function,the function fw_transfer should be called immediately , otherwise, the value will not be updated.

int fw_restore(int icdev,unsigned char _Adr);

Description

Return function, transfer the contents of EEPROM to the card's internal registers

Parameters

icdev:Value of Device Handle.

_Adr: Block address used to transfer.

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
st=fw_restore(icdev,1);
```

Remark

Use this function transfers the contents from one block of card to the internal register.and then use the function fw_transfer() transfers the contents from internal register to the other block of card. By this way,the contents would be transfers from one block to the other block and realize the value transfer between blocks.. The function can only be used for value block

int fw_transfer(int icdev,unsigned char _Adr);

Description

this function transfers the contents of the internal register to the transmitted a ddress.The sector must be authenticated for this operation .The transfer function can only be called directly after increment,decrement or restore.

Parameters

icdev:Value of Device Handle.

_Adr:the address on the card to which the contents of the internal register is transferred to

Return Value

0 if successful; otherwise, Nonzero.

Example

```
fw_restore(icdev,1);
fw_transfer(icdev,2);
this two lines will transfer the contents of block 1 to block 2.
```

Remark

reference the description of fw_ restore

__int16 fw_config_card(HANDLE icdev,unsigned char flags);

Description

Configure the card type.

Parameters

icdev:Value of Device Handle.

flags:Card type for operation (0x41=TYPEA,0x42=TYPEB,0x31=ISO15693)

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
st=fw_config_card(icdev,1);
```

__int16 a_hex(unsigned char *a,unsigned char *hex,__int16 len)

Description

String conversion function, hexadecimal characters convert into ordinary characters (long to short).

Parameters

a : Converted characters

hex: the characters to be converted

len:the length of character a

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
Unsigned char hexbuf[8]={ '3' ,' 1' ,' 6' ,' 1' ,' 4' ,' d' };
Unsigned char abuf[4];
st=a_hex(abuf,hexbuf,3);/* abuf =" 1aM" */
```

void hex_a(unsigned char *hex,unsigned char *a,__int16 len)

Description

String conversion function, ordinary characters convert into hexadecimal characters (short to long).

Parameters

hex : Converted characters

a : the characters to be converted

len:the length of character hex

Return Value

0 if successful; otherwise, Nonzero.

Example

```
unsigned char hexbuf[8]={0};
unsigned char abuf[4]= { '1' ,' a' ,' M' };;
hex_a(hexbuf, abuf ,6);/* abuf =" 31614d" */
```

2.2 Device Functions

int fw_beep(int icdev,unsigned int _Msec);

Description

Make beep

Parameters

icdev:Value of Device Handle.

unsigned int _Msec:Time of beep,(ms)

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
 st=fw_beep(icdev,10);              /*beep 100 ms*/
```

int fw_disp_mode(int icdev,unsigned char mode);

Description

Set the reader display mode, the settings will be saved after shutdown

Parameters

icdev:Value of Device Handle.

mode:Modelof display ,

0—— date, format is"year - month - day (yy-mm-dd)"

1—— time, format is "hours - minutes - seconds (hh-mm-ss)"

Return Value

0 if successful; otherwise, Nonzero.

Example

```
st=fw_disp_mode(icdev,0x01);//set time model
```

int fw_gettime(int icdev,unsigned char *time);

Description

Get date,week,time from Reader

Parameters

Icdev: Value of Device Handle.

Time: the returned data with length of 7 bytes. Format is "year, week, month, day, hour, minute, second."

Return Value

0 if successful; otherwise, nonzero.

Example

```
int st;
unsigned char datetime[7];
st=fw_gettime(icdev,datetime);
//datetime is“0x04,0x01,0x04,0x19,0x17,0x23,0x10”,
//means 17:23:10 of April 19th, 2004,Monday
```

int fw_getver(int icdev,unsinged char *buff);

Description

Get version of device

Parameters

icdev:Value of Device Handle.

buff:[out]buff for version number storage,the length is 3 bytes(including end character ’\0’)

Return Value

0 if successful; otherwise, Nonzero.

Example

```
unsigned char buff[3];
fw_getver(icdev,buff);
```

int fw_settime(int icdev,unsigned char *time);

Description

Set time of Reader

Parameters

icdev:Value of Device Handle.

time:the length is 7 bytes. Format is “year,week,month,day,hour ,minuter,second.”

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
unsigned char datetime[7]={0x04,0x01,0x04,0x19,0x16,0x35,0x10};
st=fw_settime(icdev,datetime);
```

int fw_srd_eeprom(int icdev,int offset,int length,unsigned char *rec_buffer);

Description

Get remark information of Reader

Parameters

icdev:Value of Device Handle.

offset:Offset address（0～1278）

length:Length of information to read（1～1279）

rec_buffer:[out] gotten Data

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
unsigned char buffer[100];
st=fw_srd_eeprom(icdev,0,100,buffer);
```

int fw_swr_eeprom(int icdev,int offset,int length,unsigned char* buffer);

Description

Write remark information to Reader

Parameters

icdev:Value of Device Handle.

offset:Offset address（0～1278）

length:Length of information to write（1～1279）

buffer:Information to write

Return Value

0 if successful; otherwise, Nonzero.

__int16 fw_reset(HANDLE icdev,unsigned __int16 _Msec)

Description

Reset the MCM(MIFARE read/write device core module)

Parameters

Icdev :Handle of Device

_Msec : Reset time, unit is Millionseconds (this value is 0 means off frequency, 1,2 means reset time is ... 1 ms, 2 ms ...

Return Value

<0 error. The absolute value is the error number

=0 sucessful.

Example

```
st=fw_reset(icdev,2)
```

int fw_ctl_mode(int icdev,unsigned char mode);

Description

Set Control mode of LED display

Parameters

icdev:Value of Device Handle.

mode:Display model

0——Control by computer

1——control by Reader

Return Value

0 if successful; otherwise, Nonzero.

Example

```
st=fw_ctl_mode(icdev,0x01);//Set to be controlled by Reader
```

int fw_LED_disp8(int icdev,unsigned char strlen,unsigned char* dispstr)

Description

Make the LED display random digits

Parameters

icdev:Value of Device Handle.
strlen:Numbers of digits for display (set as 8)
dispstr:Digits for display

Return Value

0 if successful; otherwise, Nonzero.

Example

```
//LED display “11112222”
unsigned char strbuf[8]={0x01,0x01,0x01,0x01,0x02,0x02,0x02,0x02};
st= fw_LED_disp8(icdev,8,strbuf);//
```

int fw_lcd_setbright(int icdev,unsigned char bright)

Description

Set LCD backlight on or off

Parameters

icdev:Value of Device Handle.
bright:sign of LCD light on or off .15—light on,0--off

Return Value

0 if successful; otherwise, Nonzero.

Example

```
st= fw_lcd_setbright(icdev,15);//light LCD
```

int fw_lcd_dispstr(int icdev,char *digit)

Description

Set LCD display string

Parameters

icdev:Value of Device Handle.
Digit :the string to display

Return Value

0 if successful; otherwise, Nonzero.

Example

```
char* sendchs=”abcdefgh”;
st=fw_lcd_dispstr(icdev,sendchs); //display    abcdefgh
```

int fw_lcd_dispclear(int icdev)

Description

Clear the display string of LCD

Parameters

icdev:Value of Device Handle.

Return Value

0 if successful; otherwise, Nonzero.

Example

```
st= fw_lcd_dispclear (icdev);
```

2.3 CPU(SAM) Functions

__int16 fw_cpureset (HANDLE ICDev, unsigned char *rlen, unsigned char

***rbuff)**

Description

Power-on reset function of CPU Card, it will automatically judge card protocol after reset.

Parameters

ICDev: Handle of Reader Device
Rlen: [out] Length of returned reset information
Rbuff: store returned reset information

Return Value

<0 error, its absolute value is error number
=0 successful

Example

```
unsigned char rlen;
unsigned char DataBuffer [100];
St=fw_cpureset (ICDev, &rlen, DataBuffer);
```

__int16 fw_setcpu (HANDLE ICDev, unsigned char SAMID)

Description

Set SAM Card deck for operation

Parameters

ICDev: Handle of Reader Device
SAMID: Sequence Number of Deck type, ox0c: Attached deck; 0x0d: SAM1; 0x0e:SAM2; 0x0f:SAM3;

Return Value

<0 error, its absolute value is error number
=0 successful

Example

```
St=fw_cpureset (ICDev, 0x0c); /* Set to attached deck */
```

__int16 fw_cpuapdu (HANDLE ICDev, unsigned char slen, unsigned char * sbuff, unsigned char *rlen, unsigned char * rbuff)

Description

Information transfer between CPU Card and APDU（Application Protocol Data Unit）, This function encapsulates the T = 0 and T = 1 operation

Parameters

ICDev: Handle of Reader Device
slen: Length of information for send
sbuff: Information buffer for send
rlen: Length of information received
rbuff: [out] buff for returned Information

Return Value

<0 error, its absolute value is error number
=0 successful

Example

```
    int st;
unsigned char slen,rlen,senddata[100], recdata[100];
```

```
slen=5;
senddata[0]=0x00;senddata[1]=0x84;senddata[2]=0x00;
senddata[3]=0x00;senddata[4]=0x04;
st= fw_cpuapdu ( icdev,slen,senddata,&rlen,recdata)
/*send the random number command to the card*/
```

__int16 fw_setcpupara (HANDLE ICDev, unsigned char cputype, unsigned char cpupro, unsigned char cpuetu)

Description

Set parameters of CPU card, default parameter cpupro=0(T=0 protocol) cpuetu=92(baud rate 9600) after power on.

Parameters

ICDev: Handle of Reader Device

cputype:Type of Deck.

ox0c: Main deck(ISO7816);

0x0d: SAM1;

0x0e:SAM2;

0x0f:SAM3.

Cpupro: Protocol of card. value 0: T=0 Protocol; value 1: T=1 protocol.

cpuetu: Time-delay data (Decimal) in card operation. For cards with different baud rates, the value of this parameter is different. Set 92 for 9600(baud rate), set 20 for 38400(baud rate)

Return Value

<0 error, its absolute value is error number

=0 successful

2.4 S70 Card-Specific Functions

int fw_read_S70 (int icdev, unsigned char _Adr, unsigned char *_Data);

Description

Read S70 card, only can read one block once, 16 bytes.

Parameters

icdev: Value of Device Handle.

_Adr: Block address (0-255);

_Data: Read data

Return Value

0 if successful; otherwise, nonzero.

Example

```
int st;
unsigned char data [16];
St=fw_read_S70 (icdev, 100, data); //read block 100 of S70 card
```

int fw_write_S70 (int icdev, unsigned char _Adr, unsigned char *_Data);

Description

Write S70 card, only can write one block once, 16 bytes.

Parameters

icdev: Value of Device Handle.

_Adr: Block address (0-255);

_Data: data to write

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
unsigned char *data=”1234567890abcdef”;
st=fw_write_S70 (icdev, 100, data); //write data to block 100
```

2.5 Ultralight Card-Specific Functions

int fw_request_ultralt (int icdev, unsigned char _Mode);

Description

Request of find card.

Parameters

icdev: Value of Device Handle.

_Mode: Mode of find card (0 or 1).

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
st= fw_request_ultralt (icdev, 0);
```

remark: Parameter _Mode can be set 0 or 1

int fw_anticall_ultralt (int icdev, unsigned long *_Snr);

Description

Prevent card conflict

Parameters

icdev: Value of Device Handle.

_ Snr: [out] Card serial number.

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
unsigned long nCard;
st= fw_anticall_ultralt (icdev, &nCard);
```

int fw_select_ultralt (int icdev, unsigned long _Snr);

Description

Select certain Card from several Ultralight cards.

Parameters

icdev: Value of Device Handle.

_ Snr: Card serial number.

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
st= fw_select_ultralt (icdev, nCard);
```

int fw_read_ultralt (int icdev, unsigned char iPage, unsigned char *redata);

Description

Read data from Ultralight card.

Parameters

icdev: Value of Device Handle.
iPage: Index number of page
redata: received Data

Return Value

0 if successful; otherwise, Nonzero.

Example

```
//read data of page 4
int st;
unsigned char ipage=4;
 unsigned char rebuffer [8] = {0};
 st= fw_read_ultralt (icdev,ipage,rebuffer);
```

int fw_write_ultralt(int icdev,unsigned char iPage,unsigned char *sdata);

Description

Write data to Ultralight card.

Parameters

icdev: Value of Device Handle.
iPage:Index number of page
sdata: Data for writing

Return Value

0 if successful; otherwise, Nonzero.

Example

```
//write data to page 4
 int st;
 unsigned char ipage=4;
  unsigned char sendbuffer[8]={0x44,0x44,0x44,0x44};

  st= fw_write_ultralt(icdev,ipage,sendbuffer);
```

int fw_halt_ultralt(int icdev);

Description

Abort operation with ultralight card.

Parameters

icdev: Value of Device Handle.

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
st= fw_halt_ultralt(icdev);
```

Remark

When you finish calling this functionl,you should call the below functions before read data next time: fw_request_ultralt first, fw_anticall_ultralt second and fw_select_ultralt finally.

2.6 Mifare Pro Card-Specific Functions

Int fw_reset_mifarepro(int icdev,unsigned char *rlen, unsigned char *rbuff);

Description

Reset Mifare Pro Card.

Parameters

icdev: Value of Device Handle.
rlen: Length of reset information
rbuff: buff for returned reset Information

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
int relen;
unsigned char rebuff[255]={0};
st= fw_reset_mifarepro(icdev,&relen,rebuff);
```

int fw_apdu_mifarepro(int icdev,unsigned char slen,unsigned char * sbuff, unsigned char *rlen,unsigned char * rbuff);

Description

Information transfer between Mifaere Pro Card and APDU（Application Protocol Unit）

Parameters

icdev: Value of Device Handle.
slen : Length of information for send
sbuff : buff for Information to send
rlen : Length of returned Information
rbuff : buff for returned Information

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
unsigned char slen,rlen,recdata[100];
slen=7;
unsigned char *senddata=”00A40000022F02”;
```

st= fw_apdu_mifarepro(icdev,slen,senddata,&rlen,recdata)

2.7 ICODE2 card-specific functions

__int16 fw_inventory(HANDLE icdev,unsigned char flags,unsigned char AFI,unsigned char masklen,unsigned char *rlen,unsigned char *rbuffer);

Description

ICODE2 Card request, return the card number (UID) and DSFID .

Parameters

icdev:Value of Device Handle.

flags: request flag; flags = 0x36: find a single card; flags = 0x16: find several cards;

AFI: Application ID

Masklen: Mask length

rlen: length returned

rbuffer: the returned content (DSFID (1 byte) + UID (8 bytes)), DSFID = rbuffer [0]

UID = rbuffer [1] ~ rbuffer [8]

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
st=fw_ inventory(icdev,0x36,AFI,0,&rlen,rbuffer); // to find a single card
st=fw_inventory(icdev,0x16,AFI,0,&rlen,rbuffer);// to find several cards;
```

__int16 fw_stay_quiet(HANDLE icdev,unsigned char flags,unsigned char *UID);

Description

Card into the quiet state, and will not return response information

Parameters

icdev: Value of Device Handle.

flags: request flags; can set 0x22;

UID: card unique identifier (card number)

Return Value

0 if successful; otherwise, Nonzero. Reference Table 1.11

Example

```
int st;
st=fw_stay_quiet(icdev,0x22,&UID[1]);
```

_int16 fw_select_uid(HANDLE icdev,unsigned char flags, unsigned char *UID);

Description

Get card into the selecting state

Parameters

icdev: Value of Device Handle.

flags: request flags; can set 0x22;
UID: card unique identifier (card number)

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
st= fw_select_uid(icdev,0x22,&UID[1]);
```

__int16 fw_reset_to_ready(HANDLE icdev,unsigned char flags, unsigned char *UID);

Description

Get card into the ready state

Parameters

icdev: Value of Device Handle.
flags: request flags; can set 0x22;
UID: card unique identifier (card number)

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
st=fw_reset_to_ready(icdev,0x22,&UID[1]);
```

__int16 fw_readblock(HANDLE icdev,unsigned char flags, unsigned char startblock,unsigned char blocknum,unsigned char *UID,unsigned char *rlen,unsigned char *rbuffer);

Description

Read block data from ICODE2 card.

Parameters

icdev: Value of Device Handle.
flags: request flags; can set 0x22;
startblock: starting block address (range 0-27)
blocknum: the number of blocks to read once(range 1-6)
UID: card unique identifier (card number)
rlen: length of returned bytes
rbuffer: returned block data

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
st=fw_readblock(icdev,0x22,1,2,&UID[1],&rlen,rbuffer); //read 2 blocks starting from block 1//
```

__int16 fw_writeblock(HANDLE icdev,unsigned char flags, unsigned charstartblock,unsigned char blocknum,unsigned char *UID,unsigned char wlen,unsigned char *rbuffer);

Description

Write data to the blocks of Icode2 card

Parameters

icdev: Value of Device Handle.
flags: request flags; can set 0x22;
startblock: starting block address (range 0-27)
blocknum: the number of blocks to write once (range 1-6)
UID: card unique identifier (number)
rlen: length of bytes to write
rbuffer: data to write

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
Unsigned char sbuffer[4]={0x00,0x00,0x00,0x00};
st=fw_writeblock(icdev,0x22,1,1,&UID[1],4,sbuffer); //write one block starting from block 1//
```

__int16 fw_lock_block(HANDLE icdev,unsigned char flags, unsigned char block,unsigned char *UID);

Description

Lock data of blocks

Parameters

icdev: Value of Device Handle.
flags: request flags; can set 0x22;
block: start block address (range 0-27)
UID: card unique identifier (card number)

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
st=fw_lock_block(icdev,0x22,1,&UID[1]); //lock block 1
```

__int16 fw_write_afi(HANDLE icdev,unsigned char flags,unsigned char AFI,unsigned char *UID);

Description

Write AFI

Parameters

icdev: Value of Device Handle.
flags: request flags; can set 0x22;
AFI: Application ID
UID: card unique identifier (card number)

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
st=fw_write_afi(icdev,0x22,0x00,&UID[1]);
```

__int16 fw_lock_afi(HANDLE icdev,unsigned char flags,unsigned char AFI,unsigned char *UID);

Description

Lock AFI

Parameters

icdev: Value of Device Handle.
flags: request flags; can set 0x22;
AFI: Application ID
UID: card unique identifier (number)

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
st=fw_lock_afi(icdev,0x22,0x00,&UID[1]);
```

__int16 fw_write_dsfid(HANDLE icdev,unsigned char flags,unsigned char DSFID,unsigned char *UID);

Description

Write DSFID

Parameters

icdev: Value of Device Handle.
flags: request flags; can set 0x22;
DSFID: data storage format ID
UID: card unique identifier (card number)

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
st=fw_write_dsfid(icdev,0x22,0x00,&UID[1]);
```

__int16 fw_lock_dsfid(HANDLE icdev,unsigned char flags,unsigned char DSFID,unsigned char *UID);

Description

Lock DSFID

Parameters

icdev: Value of Device Handle.
flags: request flags; can set 0x22;
DSFID: data storage format ID
UID: card unique identifier (number)

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
st=fw_lock_dsfid(icdev,0x22,0x00,&UID[1]);
```

__int16 fw_get_systeminfo(HANDLE icdev,unsigned char flags,unsigned char *UID,unsigned char *rlen,unsigned char *rbuffer);

Description

Read card information

Parameters

icdev: Value of Device Handle.
flags: request flags; can set 0x22;
UID: card unique identifier (card number)
rlen: length of returned bytes
rbuffer: Returned information

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
st=fw_get_systeminfo(icdev,0x22, &UID[1],&rlen,rbuffer);
```

__int16 fw_get_securityinfo(HANDLE icdev,unsigned char flags,unsigned char startblock,unsigned char blocknum, unsigned char *UID, unsigned char *rlen,unsigned char *rbuffer);

Description

Read the card security state information

Parameters

icdev: Value of Device Handle.
flags: request flags; can set 0x22;
startblock: starting block address
blocknum: number of block
UID: card unique identifier (card number)
rlen: length of returned bytes
rbuffer: Returned information

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
st=fw_get_securityinfo(icdev,0x22,10,10,&UID[1],&rlen,rbuffer)
```

2.8 AT88RF020 card-specific functions

__int16 fw_request_b(HANDLE icdev,unsigned char _Mode,unsigned char AFI,unsigned char N,unsigned char *ATQB);

Description

This function send card request command, you should call this function to select a new card.

Parameters

icdev: Value of Device Handle.
_Mode: Seek card mode

0: IDLE mode (only the cards in IDLE state can respond to this function)

1: ALL mode (cards in IDLE and HALT states, can respond to this function)

AFI: application identification number (0x00 or 0x01)

N: channel number (current effective value is 0, set to 0)

ATQB: Returned information

ATQB [0] APA, must be 0x50

ATQB [1] PUPI (1st byte), consistent with the card PUPI

/ / the first byte in PUPI

ATQB [2] PUPI (2nd byte) / / the second byte in PUPI

ATQB [3] PUPI (3rd byte) / / the third byte in PUPI

ATQB [4] PUPI (fourth byte) / / the fourth byte in PUPI

ATQB [5] APPLICATION DATA (1st byte), consistent with the card

(First byte)

ATQB [6] APPLICATION DATA (2nd byte) / / the second byte

ATQB [7] APPLICATION DATA (3rd byte) / / The third byte

ATQB [8] APPLICATION DATA (4th byte) / / the Fourth byte

ATQB [9] protocol info (1st byte), 0x00 / / protocol information (the first byte)

ATQB [10] protocol info (2nd byte), 0x00 / / protocol information (the second byte)

ATQB [11] protocol info (3rd byte), 0x41 / / protocol information (the third byte)

ATQB [12] Other

ATQB [13] Other

Return Value

0 if successful; otherwise, Nonzero.

Example

```
__int16 st;
unsigned char rData[15];
st= fw_request_b(icdev,0,0,0,&UID[1], rData);
```

__int16 fw_attrib(HANDLE icdev,unsigned char *PUPI, unsigned char CID);

Description

Select a card from the cards which have responded to REQB / WUPB command, and assigned an ID number to each card.

Parameters

icdev: Value of Device Handle.

PUPI: Pseudo-Unique PICC Identifier

CID: card ID number (0 ~ 15), this value is stored in the card for following operations.

Return Value

0 if successful; otherwise, Nonzero.

Example

```
__int16 st;
unsigned char rData[15];
st= fw_request_b(icdev,0,0,0,&UID[1], rData);
unsigned char PUPI[4];
for(int j=0;j<4;j++)
PUPI[j]= rData[1+j];
st=fw_attrib(icdev,PUPI,0);
```

Remark:

if several seleted cards are in active state, you can operate multi cards at the same time according to the CID (card ID number).

__int16 fw_check_at(HANDLE icdev,unsigned char cid,unsigned char *key);

Description

Check Password according to CID

Parameters

icdev: Value of Device Handle.
cid: card ID number, see parameter CID in “fw_attrib”
key: 8-byte password used to check

Return Value

0 if successful; otherwise, Nonzero.

Example

```
__int16 st;
unsigned char key[8]={0x1,0x2,0x3,0x4,0x5,0x6,0x7,0x8};
st=fw_check_at(icdev,0,key);
```

__int16 fw_read_at(HANDLE icdev,unsigned char Adr,unsigned char*key,unsigned char* rbuffer);

Description

Read card according to CID, one page each time.

Parameters

icdev: Value of Device Handle.
Adr: the page address to read (0 ~ 31)
key: 8-byte password used to check
rbuffer: returned 8 bytes data

Return Value

0 if successful; otherwise, Nonzero.

Example

```
//to read page 0
__int16 st;
unsigned char revbuf[16];
unsigned char key[8]={0x1,0x2,0x3,0x4,0x5,0x6,0x7,0x8};
st= fw_read_at(icdev,0,key,revbuf);
```

__int16 fw_write_at(HANDLE icdev,unsigned char Adr,unsigned char* sbuffer);

Description

Write card according to CID, one page each time.

Parameters

icdev: Value of Device Handle.

Adr: the page address to write (0 ~ 31)

rbuffer: 8 bytes data to write

Return Value

0 if successful; otherwise, Nonzero.

Example

```
//to write page 4
__int16 st;
unsigned char data[8]={0x1,0x2,0x3,0x4,0x5,0x6,0x7,0x8};
st= fw_write_at(icdev,4,data);
```

__int16 fw_changekey_at(HANDLE icdev,unsigned char* key);

Description

Change Password.

Parameters

icdev: Value of Device Handle.

key: The new 8-byte password

Return Value

0 if successful; otherwise, Nonzero.

Example

```
__int16 st;unsigned char key[8]={0};//change password to 00000000
st= fw_changekey_at(icdev,key);
```

__int16 fw_lock_at(HANDLE icdev,unsigned char Adr,unsigned char*sbuffer);

Description

Lock card. Can lock certain areas of card, the locked areas can only be read.

Parameters

icdev: Value of Device Handle.

Adr : block address No.

sbuffer: lock flag, the first byte means as follows:

0: Locked;

1: Unlocked

Return Value

0 if successful; otherwise, Nonzero.

Example

```
__int16 st;
unsigned char flag[8]={0};
st= fw_lock_at(icdev,4,flag); /to lock block 4
```

__int16 fw_halt_at(HANDLE icdev,unsigned char cid,unsigned char *key);

Description

Halt the operation of card, this function is invalid when the card in active state.

Parameters

icdev: Value of Device Handle.

cid : card ID number, see parameter CID in “fw_attrib”

key : 8 bytes password

Return Value

0 if successful; otherwise, Nonzero.

Example

```
__int16 st;
unsigned char key[8]={0};
st= fw_halt_at(icdev,0,key);
```

__int16 fw_count_at(HANDLE icdev,unsigned char cid,unsigned char* key);

Description

Count; each time when an order executed, the counter value of page 2 is increased by 1, the corresponding matching signature information will be written to the first 6 bytes in the 2nd page according to CID.

Parameters

icdev: Value of Device Handle.

cid : card ID number, see parameter CID in “fw_attrib”

key : 8 bytes password

Return Value

0 if successful; otherwise, Nonzero.

Example

```
__int16 st;
unsigned char key[8]={0};
st= fw_count_at(icdev,0,key);
```

2.9 Contactless CPU (ISO1443) card-specific functions

__int16 fw_pro_reset(int ICDev,unsigned char *rlen,unsigned char * rbuff);

Description

Card reset function

Parameters

icdev: Value of Device Handle.

rlen: length of returned reset information

rbuff: reset information

Return Value

0 if successful; otherwise, Nonzero.

Example

```
__int16 st;
```

unsigned char len;
Unsigned char revbuf[20]={0};
st= fw_pro_reset(icdev,&len,revbuf);
Remark: You should call function “fw_card” once first before this function is called.

__int16 fw_pro_commandlink(int ICDev,unsigned char slen,unsigned char * sbuff,unsigned char *rlen,unsigned char * rbuff,unsigned char tt,unsigned char FG);

Description

APDU data exchange function

Parameters

icdev: Value of Device Handle.
slen: the length of information to send
sbuff: command to send
rlen: length of returned information
rbuff: returned information
tt: delay time, unit: 10ms
FG: split length, it is recommended that this value be less than 64

Return Value

0 if successful; otherwise, Nonzero.

Example

__int16 st;
unsigned char srvBuffer[256]={0x80,0x84,0x00,0x00,0x10};
unsigned char revBuffer[256]={0};
unsigned char sendlen=5;
unsigned char ftt=9;//to delay 90ms
unsigned char fFG=60;//to send 60 bytes each time
unsigned char revlen;
st=fw_pro_commandlink(icdev,sendlen,srvBuffer,&revlen,revBuffer,ftt,fFG);

2.10 Desfire card-specific functions

int fw_anticoll2(int icdev,unsigned char _Bcnt,unsigned long *_Snr);

Description

The second anti-collision

Parameters

icdev: Value of Device Handle.
_Bcnt: Anti-conflict level, set 0 here
_Snr: Return 5 bytes of card information, the first byte is 0x88, the other four bytes are the high byte of the card number .

Return Value

0 if successful; otherwise, Nonzero.

Example

int st;

unsigned long snr2;
st= fw_anticoll2 (icdev,0,&snr2);
int fw_select2(int icdev,unsigned long _Snr);

int fw_select2(int icdev,unsigned long _Snr);

Description

The second time to select a card

Parameters

icdev: Value of Device Handle.

_Snr: card serial number gotten by the second Anti-collision

Return Value

0 if successful; otherwise, Nonzero.

Example

int st;
st= fw_select2(icdev,snr2);

int fw_reset_desfire(int icdev,unsigned char *rlen,unsigned char *rdata);

Description

Reset desifare card

Parameters

icdev: Value of Device Handle.

rlen: length of reset information

rdata: Return of the reset information

Return Value

0 if successful; otherwise, Nonzero.

Example

int st;
unsigned char revlen;
unsigned char revdata[50];
st= fw_reset_desfire (icdev,&revlen,revdata);

int fw_authen_desfire(int icdev,unsigned char keyNo, char* key,unsigned char* sessionKey);

Description

Key authentication

Parameters

icdev: Value of Device Handle.

keyNo: the key number to verify

key: 16 bytes key

sessionKey: the session key returned after successful key authentication

Return Value

0 if successful; otherwise, Nonzero. (Refer to 1.11 Tables for more information)

Example

//Verify key number 1

```
 int st;
 char
curkey[17]={0x11,0x22,0x33,0x44,0x55,0x66,0x77,0x88,0x11,0x22,0x33,0x4
4,
0x55,0x66,0x77,0x88};
 unsigned char sessionkey[50];
st= fw_authen_desfire(icdev,1,curkey,sessionkey);
```

int fw_getver_desfire(int icdev,unsigned char* rlen,unsigned char* version);

Description

Get version of DESFIRE

Parameters

icdev: Value of Device Handle.

rlen: length of returned data

version: returned card manufacturer data

Return Value

0 if successful; otherwise, Nonzero. Reference Table 1.11

Example

```
int st;
unsigned char revlen;
unsigned char data[50];
st= fw_getver_desfire (icdev,rlen,data);
```

int fw_getAIDs_desfire(int icdev,unsigned char* rlen,unsigned char* AIDS);

Description

Get application identifier

Parameters

icdev: Value of Device Handle.

rlen: length of returned data

AIDS: Return of the identification number of all applications

Return Value

0 if successful; otherwise, Nonzero. Reference Table 1.11

Example

```
int st;
unsigned char revlen;
unsigned char aids[50];
st= fw_getver_desfire (icdev,&rlen,aids);
```

int fw_selectApp_desfire(int icdev,unsigned char* AID);

Description

Select the current application

Parameters

icdev: Value of Device Handle.

AID : current application identifier to select

Return Value

0 if successful; otherwise, Nonzero. Reference Table 1.11

Example

```
int st;
unsigned char aid[4]={0x01,0x00,0x00};
st= fw_selectApp_desfire(icdev,aid);
```

int fw_getKeySetting_desfire(int icdev,unsigned char* rlen,unsigned char* setbuf);

Description

Get the master key settings

Parameters

icdev: Value of Device Handle.

Rlen: the length of returned data

Setbuf: set the master of key (card) application

Return Value

0 if successful; otherwise, Nonzero. Reference Table 1.11

Example

```
int st;
unsigned char revlen;
unsigned char set[4];
st= fw_getKeySetting_desfire (icdev,&revlen,set);
```

int fw_getKeyver_desfire(int icdev,unsigned char keyNo,unsigned char* keyVer);

Description

Get the version of master key

Parameters

icdev: Value of Device Handle.

keyNo: key numbers

keyVer: Key version

Return Value

0 if successful; otherwise, Nonzero. Reference Table 1.11

Example

```
int st;
unsigned char keyVersion[3];
st= fw_getKeyver_desfire(icdev,1, keyVersion);
```

int fw_createApp_desfire(int icdev,unsigned char*AID,unsigned char KeySetting,unsigned char NumOfKey);

Description

Creat application

Parameters

icdev: Value of Device Handle.

AID: key numbers

KeySetting: set application master key

The meaning of 8-bit Application master key can be decribed as below:

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
changeKey Access Rights Bit3	changeKey Access Rights Bit2	changeKey Access Rights Bit1	changeKey Access Rights Bit0	Configuration changeable	Free create/delete	Free directory list access	Allow change master key

NumOfKey: the number of keys

Return Value

0 if successful; otherwise, Nonzero. Reference Table 1.11

Example

```
int st;
unsigned char aid[4]={0x02,0x00,0x00};
unsigned char setting=0xef;
st= fw_createApp_desfire(icdev,aid,setting,0x0e);/* With 14 keys */
```

int fw_delAID_desfire(int icdev,unsigned char* AID);

Description

Delete application

Parameters

icdev: Value of Device Handle.

AID : Application ID

Return Value

0 if successful; otherwise, Nonzero. Reference Table 1.11

Example

```
int st;
unsigned char aid[3]={0x02,0x00,0x00};
st= fw_delAID_desfire(icdev,aid);
```

int fw_changeKeySetting_desfire(int icdev,unsigned char newSet,char* sessionKey);

Description

Change the master key settings

Parameters

icdev: Value of Device Handle.

newSet: new key settings

The meaning of 8-bit Application master key can be decribed as below:

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
changeKey Access Rights Bit3	changeKey Access Rights Bit2	changeKey Access Rights Bit1	changeKey Access Rights Bit0	Configuration changeable	Free create/delete	Free directory list access	Allow change master key

8-bit of card's master key, represent the following meanings:

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
RFU	RFU	RFU	RFU	Configuration changeable	Free create/delete Withiout PICC master Key	Free directory list access without PICC master key	Allow change master key

Each bit of Card-level key settings represents the meaning as follows:

Bit7-Bit4: reserved, must be set to 0.

Bit3: Code which determines whether it allows changing the master key settings:

0: configuration can not be changed no longer (frozen).

1: you can modify the configuration (the default) after authenticating the card's master key.

Bit2: Code which determines if it needs certification for card's master keys when creating /deleting application:

0: you can not create / delete applications until the master key of card has been authenticated successfully,.

1: the application is also permitted to creat without certificating card master key (the default settings).

Before deleting the application, you must authenticate the master key of application or the master key of card successfully.

Bit1: this byte dertermines whether it needs to certificate the card master key for access to the application directory:

- 0:fw_getAIDs_desfire and fw_getKeySetting_desfire: need to successfully authenticate the master key of card.
- 1: fw_getAIDs_desfire and fw_getKeySetting_desfire: do not need to authenticate the master key of card (Default setting).

Bit0: Code which determines whether the master key of card can be modified:

- 0: card master key can not be modified (frozen).
- 1: Card master key can be modified (the current card master key must be certified, default settings).

sessionKey: session key

Return Value

0 if successful; otherwise, Nonzero. Reference Table 1.11

Example

```
int st;
unsigned char set=0x0f;
st= fw_changeKeySetting_desfire (icdev,set,key);
```

int fw_changeKey_desfire(int icdev,unsigned char* sessionKey,unsigned char* curKey,unsigned char keyNo,unsigned char* newkey);

Description

Change the master key

Parameters

icdev: Value of Device Handle.
sessionKey: session key
curkey: the current key
keyNo: key number
newkey: new key

Return Value

0 if successful; otherwise, Nonzero. Reference Table 1.11

Example

```
int st;
unsigned char
currentkey[17]={0x11,0x22,0x33,0x44,0x55,0x66,0x77,0x88,0x11,0x22,0x33,
0x44,0x55,0x66,0x77,0x88};
unsigned char
newkey[17]={0x22,0x22,0x22,0x22,0x22,0x22,0x22,0x22,0x22,0x22,0x22,
0x22,0x22,0x22,0x22,0x22};
st= fw_changeKey_desfire(icdev,sessionkey,currentkey,1,newkey);
```

int fw_getFileIDs_desfire(int icdev,unsigned char* rlen,unsigned char* fileIDs);

Description

Get all file identification numbers of current application

Parameters

icdev: Value of Device Handle.
Rlen: the length of returned data
fileIDs: file identification number (each byte means a file identification number)

Return Value

0 if successful; otherwise, Nonzero. Reference Table 1.11

Example

```
int st;
unsigned char revlen;
unsigned char fileids[20];
st= fw_getFileIDs_desfire (icdev,&revlen,fileids);
```

int fw_getFileProper(int icdev,unsigned char fileNo,unsigned char* rlen,unsigned char * fileProper);

Description

Get File Settings

Parameters

icdev: Value of Device Handle.

fileNo: File ID

rlen: length of returned data

fileProper: file property settings

Return Value

0 if successful; otherwise, Nonzero. Reference Table 1.11

Example

```
int st;
unsigned char revlen;
unsigned char fileset[20];
st= fw_getFileProper (icdev,&revlen,fileset);
```

int fw_changeFileSetting(int icdev,unsigned char fileNo,unsigned char comSet,unsigned char* accessRight,char* sessionKey);

Description

Change the file settings

Parameters

icdev: Value of Device Handle.

fileNo: File ID

comSet: data transmission form:

0: transmission in the clear,

1: MAC code validation,

3: DES/3DES Encryption

accessRight: Access right

accessRight [0]:Low nibble has the right to modify access permissions

High nibble has the right to read / write the file

accessRight [1]:Low nibble has the access to write to the file

High nibble has the access to read the file

sessionKey: session key

Return Value

0 if successful; otherwise, Nonzero. Reference Table 1.11

Example

```
/ / Modify the setting of file with identification number of 0x01
int st;
unsigned char comSetting=0x01;// MAC code verification
unsigned char accessRights[3]={0x22,0x22};/* to Read, write, read / write, to modify the settings, all are required to verify the key No. 2 */
st= fw_changeFileSetting (icdev,0x01,comSetting,accessRights,sesskey);
/* sesskey key obtained by verifying the session key */
```

int fw_createDataFile_desfire(int icdev,unsigned char fileNo,unsigned char ComSet,unsigned char* AccessRight,unsigned char* FileSize);

Description

Create a standard data file

Parameters

icdev: Value of Device Handle.
fileNo: File ID
comSet: data transmission form:
0: transmission in the clear,
1: MAC code validation,
3: DES/3DES Encryption
accessRight: Access right
accessRight [0]:Low nibble has the right to modify access permissions
High nibble has the right to read / write the file
accessRight [1]:Low nibble has the access to write to the file
High nibble has the access to read the file
FileSize: File Size

Return Value

0 if successful; otherwise, Nonzero. Reference Table 1.11

Example

```
int st;
unsigned char comSetting=0x00;// transmission in the clear
unsigned char accessRights[3]={0x22,0x22};/* Read, write, read / write, modify the settings are all required to verify the key No. 2 */
unsigned char fsize[4]={0x20,0x00,0x00};// length of 32 bytes
st= fw_createDataFile_desfire(icdev,0x01,comSetting,accessRights,fsize);
```

int fw_createValueFile_desfire(int icdev,unsigned char fileNo,unsigned char ComSet,unsigned char* AccessRight,unsigned char* lowerLimit,unsigned char* upperLimit,unsigned char* value,unsigned char creditEnabled);

Description

Create value file

Parameters

icdev: Value of Device Handle.
fileNo: File ID
comSet: data transmission form:
0: transmission in the clear,
1: MAC code validation,
3: DES/3DES Encryption
accessRight: Access right
accessRight [0]:Low nibble has the right to modify access permissions
High nibble has the right to read / write the file
accessRight [1]:Low nibble has the access to write to the file
High nibble has the access to read the file
lowerLimit: Minimum value
upperLimit: Maximum value
value : the current value

creditEnabled: whether to support limited memory

Return Value

0 if successful; otherwise, Nonzero. Reference Table 1.11

Example

```
int st;
unsigned char comSetting=0x03;//DES Encrypted transmission
unsigned char accessRights[3]={0x22,0x22};/* Read, write, read / write, modify the settings are required to verify the key No. 2 */
unsigned char lower[4]={0x00,0x00,0x00,0x00};// Minimum is 0
unsigned char upper[4]={0xff,0xff,0xff,0x00};// Maximum is 0xffffff
unsigned char value[4]={0x32,0x00,0x00,0x00};// Current value is set to 50 (0x32)
unsigned char enable=0x01;// Support limited memory
st=fw_createValueFile_desfire(icdev,0x02,comSetting,accessRights,lower,upe
r,value,enabled);
```

int fw_ createCsyRecord_desfire(int icdev,unsigned char fileNo,unsigned charcomSet,unsignedchar*AccessRight,unsignedchar* RecordSize,unsigned char* MaxNum);

Description

Create cycle record file

Parameters

icdev: Value of Device Handle.

fileNo: File ID

comSet: data transmission form:

0: Express transmission,

1: MAC code validation,

3: DES/3DES Encryption

accessRight: Access right

accessRight [0] : Low nibble has the right to modify access permissions

High nibble has the right to read / write the file

accessRight [1] : Low nibble has the access to write to the file

High nibble has the access to read the file

RecordSize: the length of each record

MaxNum: the Max number of records

Return Value

0 if successful; otherwise, Nonzero. Reference Table 1.11

Example

```
int st;
unsigned char comSetting=0x03;//DES Encrypted transmission
unsigned char accessRights[3]={0x22,0x22};/* Read, write, read / write, modify the settings are required to verify the key No. 2 */
unsigned char recordLen[3]={0x20,0x00,0x00};//each record with 32 bytes
unsigned char number[3]={0x10,0x00,0x00};//the file has 16 records at most
```

st= fw_createCsyRecord_desfire
(icdev,0x03,comSetting,accessRights,recordLen,number);

int fw_delFile_desfire(int icdev,unsigned char fileNo);

Description

Delete File

Parameters

icdev: Value of Device Handle.
fileNo: File ID

Return Value

0 if successful; otherwise, Nonzero. Reference Table 1.11

Example

int st;
st= fw_delFile_desfire(icdev,0x03);

int fw_write_desfire(int icdev,unsigned char fileNo,unsigned int offset,unsigned int length,unsigned char* data,char* sessionKey);

Description

Write data file

Parameters

icdev: Value of Device Handle.
fileNo: File ID
offset: offset address
length: length of data to write
data: data to be written,
sessionKey: session key

Return Value

0 if successful; otherwise, Nonzero. Reference Table 1.11

Example

int st;
unsigned char data[10]={0x11,0x22,0x33,0x44,0x55,0x66,0x77,0x88};
st= fw_write_desfire (icdev,0x01,0,8,data,sesskey);/*sesskey means the session key obtained by verifying key */

int fw_read_desfire(int icdev,unsigned char fileNo,unsigned int offset,unsigned int length,unsigned char* revData,char*sessionKey);

Description

Read data file

Parameters

icdev: Value of Device Handle.
fileNo: File ID
offset: offset address
length: length of data to read
revData: the data to be read,
sessionKey: session key

Return Value

0 if successful; otherwise, Nonzero. Reference Table 1.11

Example

```
int st;
unsigned char data[10];
st= fw_read_desfire (icdev,0x01,0,8,data,sesskey);/ *sesskey means the session key obtained by verifying key */
```

int fw_getvalue_desfire(int icdev,unsigned char fileNo,unsigned int* value,char*sessionKey);

Description

Get the value of value file

Parameters

icdev: Value of Device Handle.
fileNo: File ID
value: the gotten value
sessionKey: session key

Return Value

0 if successful; otherwise, Nonzero. Reference Table 1.11

Example

```
int st;
unsigned int value;
st= fw_getvalue_desfire (icdev,0x02,&value,sesskey); / *sesskey means the session key obtained by verifying key */
```

int fw_credit_desfire(int icdev,unsigned char fileNo,unsigned int value,char*sessionKey);

Description

increment

Parameters

icdev: Value of Device Handle.
fileNo: File ID
value: the value to be increased
sessionKey: session key

Return Value

0 if successful; otherwise, Nonzero. Reference Table 1.11

Example

```
int st;
unsigned int value=100;/*set value=100*/
st= fw_credit_desfire (icdev,0x02,value,sesskey); / *sesskey means the session key obtained by verifying key */
```

Remark

After this function called successfully, you must also call the function fw_commitTransfer_desfire to make the operation into effect

int fw_debit_desfire(int icdev,unsigned char fileNo,unsigned int

value,char* sessionKey);

Description

decrement

Parameters

icdev: Value of Device Handle.
fileNo: File ID
value: the gotten value
sessionKey: session key

Return Value

0 if successful; otherwise, Nonzero. Reference Table 1.11

Example

```
int st;
unsigned int value=100;/*set the value to 100*/
st= fw_debit_desfire (icdev,0x02,value,sesskey);/ *sesskey means the session key obtained by verifying key */
```

Remark

After this function called successfully, you should also call the function fw_commitTransfer_desfire to make the operation into effect.

int fw_writeRecord_desfire(int icdev,unsigned char fileNo,unsigned int offset,unsigned int length,unsigned char* data,char* sessionKey);

Description

Wite one record to record file

Parameters

icdev: Value of Device Handle.
fileNo: File ID
offset: offset address
length: the length of written data
data : data to be written
sessionKey: session key

Return Value

0 if successful; otherwise, Nonzero. Reference Table 1.11

Example

```
int st;
unsigned char data[8]={0x11,0x22,0x33,0x44,0x55,0x66,0x77,0x88};
st= fw_writeRecord_desfire (icdev,0x03,0,8,data,sesskey); / *sesskey means the session key obtained by verifying key */
```

Remark

After this function called successfully, you must also call the function fw_commitTransfer_desfire to make the operation into effect

int fw_readRecord_desfire(int icdev,unsigned char fileNo,unsigned int offset,unsigned int length,unsigned char* revData,unsigned int* SgRecordlen,unsigned int* rlen,char* sessionKey);

Description

Read record file

Parameters

icdev: Value of Device Handle.
fileNo: File ID
offset: offset address, the beginning record number
length: the number of records to read from the offset address
revData: data read out
SgRecordlen: the length of individual record
Rlen: the total length of data read out
sessionKey: session key

Return Value

0 if successful; otherwise, Nonzero. Reference Table 1.11

Example

```
int st;
unsigned char data[1000];
unsigned int sglen;
unsigned int revlen;
st= fw_readRecord_desfire
(icdev,0x03,0,1,data,&sglen,&revlen,sesskey); / *sesskey means the session key obtained by verifying key */
```

Remark: If the length and offset are set to 0, all the records will be read out.

int fw_clearRecord_desfire(int icdev,unsigned char fileNo);

Description

clear the data of record file

Parameters

icdev: Value of Device Handle.
fileNo: File ID

Return Value

0 if successful; otherwise, Nonzero. Reference Table 1.11

Example

```
int st;
st= fw_clearRecord_desfire (icdev,0x03);
```

Remark: after this function called successfully, you must also call the function fw_commitTransfer_desfire to make the operation into effect

int fw_commitTransfer_desfire(int icdev);

Description

commit a data transmission

Parameters

icdev: Value of Device Handle.

Return Value

0 if successful; otherwise, Nonzero. Reference Table 1.11

Example

```
int st;
st= fw_commitTransfer_desfire (icdev);
```

Note: after the operations of Increase / impairment, writing / clearing the record are done, you must call this function to make the operations into effect.

int fw_abortTransfer_desfire(int icdev);

Description

Abort a data transmission

Parameters

icdev: Value of Device Handle.

Return Value

0 if successful; otherwise, Nonzero. Reference Table 1.11

Example

```
int st;
st= fw_abortTransfer_desfire (icdev);
```

int fw_formatPICC_desfire(int icdev);

Description

Format card

Parameters

icdev: Value of Device Handle.

Return Value

0 if successful; otherwise, Nonzero. Reference Table 1.11

Example

```
int st;
st= fw_formatPICC_desfire (icdev);
```

Remark

After Calling this function successfully, all data in the card will be cleared.

2.11 4442 card-specific function

int fw_read_4442(int icdev,unsigned char _Adr,unsigned char *_Data,int length);

Description

Read data from 4442 card.

Parameters

icdev:Value of Device Handle.
_Adr: Start address for reading(0~255)
_Data: Data returned.
length: Length of Data to read(0~255)

Return Value

0 if successful; otherwise, Nonzero.

Example

```
// Read 20 chars start from address:0
  int st;
  unsigned char rbuf[300]={0};
  st= fw_read_4442(icdev,0,rbuf,20);
```

Remarks: 1.Length should not exceed 256, otherwise the whole 256 bytes will be read.

int fw_write_4442(int icdev,unsigned char _Adr,unsigned char *_Data,int length);

Description

Write data to 4442 card.

Parameters

icdev:Value of Device Handle.
_Adr: Starting address for writing(0x30~0xff).
_Data: Data for writing.
length: Length of data to write.

Return Value

0 if successful; otherwise, Nonzero.

Example

```
// write 4 chars start from address:0x30
int st;
unsigned char sbuf[4]={0x01,0x02,0x03,0x04};
st= fw_ write _4442(icdev,0x30,sbuf,4);
```

Remark

1: the value of parameter _Adr should be set between 0x30 and 0xff;
2: the value of parameter “length” should be less than the actual length of data to be written;

int fw_getProtectData_4442(int icdev,unsigned char _Adr,unsigned char *_Data,int length);

Description

Read protected bits.

Parameters

icdev:Value of Device Handle.
_Adr: Starting address for Reading(must set 0)
_Data: returned Protect bits.
length: Length of data to read（must set 4）

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
unsigned char rbuf[4]={0};
st=fw_getProtectData_4442(icdev,0, rbuf,4);
```

Remark

4442 card has 32 bytes of protection data, the address is x00-0x20, 4-byte read out correspond to each of a corresponding bit, 0 for write protection, 1 for not write-protection;

int fw_setProtectData_4442(int icdev,unsigned char _Adr,unsigned char

***_Data,int length);**

Description

Write protected bits.

Parameters

icdev:Value of Device Handle.

_Adr: Starting address for writing(0~32).

_Data:Data for writing.

length: Length of data to write(0~32).

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
unsigned char rbuf[2]={0xa2,0x1e};
st=fw_setProtectData_4442(icdev,0, rbuf,2);
```

Remark

1: parameter _Adr should be set to 0~32;

2: Data for writing must be Consistent with the data stored in the card;

3: value of parameter “length” should not exceed 32;

int fw_authentikey_4442(int icdev,unsigned char _Adr,int rlen,unsigned char *key);

Description

Verify keys.

Parameters

icdev: Value of Device Handle.

_Adr: Starting address for data of verify(must set 0).

rlen: Length of data to verify(must set 3).

key: Key to verify(3 bytes).

Return Value

0 if successful; otherwise, Nonzero.

Example

```
unsigned char keybuffer[3]={0xff,0xff,0xff};
if(fw_authentikey_4442(icdev,0,3,keybuffer)!=0)
{
   printf(“Authentication error”);
}
```

Remark

Card will be locked if this function returns failure for **three** times continuously.

int fw_changkey_4442(int icdev,unsigned char _Adr,int rlen,unsigned char *key);

Description

Update Key of card.

Parameters

icdev: Value of Device Handle.

_Adr: Starting address of key-data (must set 0).

rlen:Length of key-data for updating(must set 3).

key: key-data for updating(3 bytes).

Return Value

0 if successful; otherwise, Nonzero.

Example

```
unsigned char keybuffer[3]={0x00,0x00,0x00};
if(fw_changkey_4442(icdev,0,3,keybuffer)!=0)
{
   printf("Change key error");
}
```

int fw_cntReadError_4442(int icdev,unsigned char *cntReadError);

Description

Read counts of Error-Code

Parameters

icdev: Value of Device Handle.

cntReadError: Times of Read-Error

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
unsigned char nerror;
st= fw_cntReadError_4442(icdev,&nerror);
```

2.12. 4428 card-specific functions

int fw_read_4428(int icdev,unsigned int _Adr,unsigned char *_Data,int length);

Description

Read data from 4428 card

Parameters

icdev: Value of Device Handle.

_Adr: The starting address to read

_Data: Returned data

length: length of returned data

Return Value

0 if successful; otherwise, Nonzero.

Example

```
//to read 20 bytes starting from address 0
int st;
unsigned char rbuf[300]={0};
st= fw_read_4428(icdev,0,rbuf,20);
```

remark :

1: the value of length should be less than 1024, otherwise reading card will fail.

2: the parameters _Adr should be less than 1023;

int fw_write_4428(int icdev,unsigned int _Adr,unsigned char *_Data,int length);

Description

Write data to 4428 card

Parameters

icdev: Value of Device Handle.
_Adr: The starting address of written data
_Data: data to be written
length: length of written data

Return Value

0 if successful; otherwise, Nonzero.

Example

```
//write 4 bytes starting from address 0x30
int st;
unsigned char sbuf[4]={0x01,0x02,0x03,0x04};
st= fw_write_4428(icdev,0x30,sbuf,4);
```

remark :

1:Parameter _Adr should value between 0x0 and 0x3ff;
2: the value of parameter length should be less than the actual length of data to be written;

int fw_getProtectData_4428(int icdev,unsigned int _Adr,unsigned char *_Data,int length);

Description

Read protected bit

Parameters

icdev: Value of Device Handle.
_Adr: The starting address of read data
_Data: returned protected bit
length: the length to read

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
unsigned char rbuf[4]={0};
st=fw_getProtectData_4428(icdev,0, rbuf,4);
```

int fw_setProtectData_4428(int icdev,unsigned int _Adr,unsigned char *_Data,int length);

Description

write protected bits

Parameters

icdev: Value of Device Handle.
_Adr: The starting address of written data
_Data: data to be written
length: the length of written data

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
unsigned char rbuf[2]={0xa2,0x1e};
st=fw_setProtectData_4428(icdev,0, rbuf,2);
```

Remark:

1: parameter _Adr should value between 0 and 1023;
2: written data should be consistent with the corresponding data which stored in card;
3: parameter length should not exceed 1024;

int fw_authentikey_4428(int icdev, unsigned char *key);

Description

Authenticate key of 4428 card

Parameters

icdev: Value of Device Handle.
Key : key to be verified(2 bytes)

Return Value

0 if successful; otherwise, Nonzero.

Example

```
unsigned char keybuffer[2]={0xff,0xff};
if(fw_authentikey_4428(icdev,keybuffer)!=0)
{
printf("Authentication error");
}
```

Remark:

Card will be locked if this function fails for 8 times continuously.

int fw_changkey_4428(int icdev, unsigned char *key);

Description

Change key

Parameters

icdev: Value of Device Handle.
Key : key to be changed(2 bytes)

Return Value

0 if successful; otherwise, Nonzero.

Example

```
unsigned char keybuffer[2]={0x00,0x00};
if(fw_changkey_4428(icdev,keybuffer)!=0)
{
printf("Change key error");
}
```

int fw_cntReadError_4428(int icdev,unsigned char *cntReadError);

Description

Get the count of read-error

Parameters

icdev: Value of Device Handle.

cntReadError : the count of read-error

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
unsigned char nerror;
st= fw_cntReadError_4428(icdev,&nerror);
```

2.13. 24C64 card-specific functions

int fw_read_24c64(int icdev,unsigned int offset,unsigned int length, unsigned char* rdata);

Description

Read data from 24C64 card

Parameters

icdev: Value of Device Handle.

offset: offset address of read data

length: the length of the data to be read

rdata: read out data

Return Value

0 if successful; otherwise, Nonzero.

Example

```
// Read 20 bytes Starting from address 0
int st;
unsigned char rbuf[300]={0};
st= fw_read_24c64(icdev,0,20,rbuf);
```

int fw_write_24c64(int icdev,unsigned int offset,unsigned int length,unsigned char* wdata);

Description

Write data to 24C64 card

Parameters

icdev: Value of Device Handle.

offset: offset address

length: the length of data to write

wdata: data to write

Return Value

0 if successful; otherwise, Nonzero.

Example

```
// write 2 bytes starting from the address 0
int st;
unsigned char wbuf[300]={0x11,0x22};
st= fw_write_24c64(icdev,0,2,wbuf);
```

int fw_check_24c64(int icdev);

Description

Check whether there is a card inserted, and whether it is 24C64 card

Parameters

icdev: Value of Device Handle.

Return Value

0 if successful—showing there is 24C64 card inserted; otherwise, Nonzero.

Example

```
int st;
st= fw_check_24c64(icdev);
```

2.14.125K(ID) Card-specific function

int fw_read_SerialNumberID(int icdev,unsigned int _Msec,unsigned char* snID);

Description

Get ID card number

Parameters

icdev: Value of Device Handle.

_Msec: the interval for the Reader to beep (unit: ms)

snID : returned 10-digit card number

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
unsigned char snr[11]={0};
st=fw_read_SerialNumberID(icdev,10,snr);
```

3. MIFARE ONE Card Structure

Features:

1K bytes of memory, Composed of 16 sectors, each sector has 4 blocks, and each block has 16 bytes.

User can control the writing restrictions for each block.

Each sector has an independent set of passwords and access control

Each card has a unique serial number with 32bits

With anti-collision mechanism, and supports multi-card operation

It comes with antenna, no power needed, and includes encryption control logic and communication logic circuit

Operating temperature: -20 ℃ ~ 50 ℃

Operating frequency: 13.56MHZ

Communication Rate: 106KBPS

Operating distance: 10mm or less (depending on the reader)

Data retention period is 10 years, can be rewritten 100,000 times, unlimited read times

Storage structure:

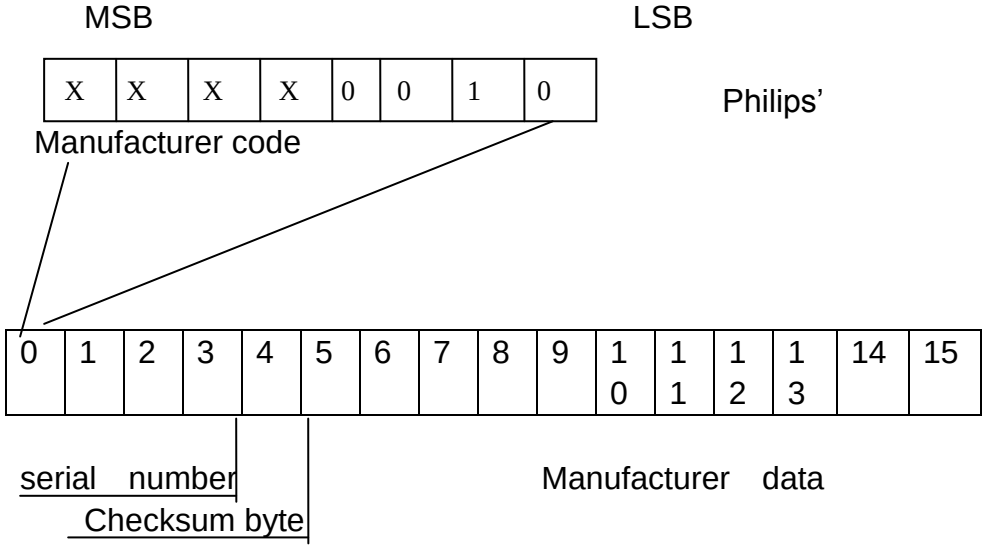
M1 card is divided into 16 sectors, each sector has 4 blocks (block 0～3), a total of 64 blocks, addressable by block number from 0 to 63. Block 0 of sector 0 (the absolute address 0 block) is used to store Manufacturer code, which has been Solidified, and can not be changed. The block 0, block 1, block 2 of other sectord is the data blocks, used for storing data; block 3 is the control block, stored passwords A, access control, password B, its structure as follows:

A0 A1 A2 A3 A4 A5	FF 07 80 69	B0 B1 B2 B3 B4 B5
Key A(6 bytes)	access control (4 bytes)	key B(6 bytes)

Sector	Block	1 2 3 4 5	6 7 8 9	10 11 12 13 14 15	Description	Number
0	0				Manualfacturer Block	0
	1				Data	1
	2				Data	2
	3	Key A	Access	Key B	Sector Trailer 0	3
1	0				Data	4
	1				Data	5
	2				Data	6
	3	Key A	Access	Key B	Sector Trailer 1	7

		⋮				
	0				Data	60
15	1				Data	61
	2				Data	62
	3	KeyA	Access	Key B	Sector Trailer 15	63

Manufacturer Block（IC card manufacturer code block）: the first block of t he first sector is used by the manufacturers, stored the IC card manufacturer code, the data of this block can not be changed after written



Data Block : the first three blocks of all sectors are used to store data (block 0 of sector 0 can only be read, block 1, block 2 can be used to store data)

Value Block: can be used as electronic purse (valid commands: read, write, increment, decrement, restore, transfe), the data of the value block only has 4 bytes

Sector Trailer(Block 3)（Control Block）: Each sector has a control block (block 3), including the key A (6 bytes) and key B (6 bytes) and a control bit (4 bytes)

Control Properties:

1. the key and access control of each sector are independent, it may set their own password and access control according to the actually need. In the access control ,Each block has three control bits corresponding, defined as follows:

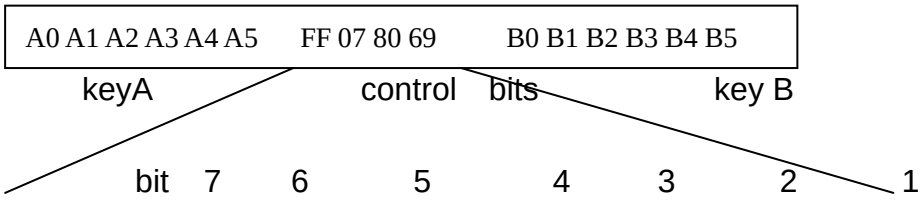
Block 0: C10 C20 C30

Block 1: C11 C21 C31

Block 2: C12 C22 C32

Block 3: C13 C23 C33

Three control bits exists in the access control byte by positive or negative form, determines access rights to the block (such as impairment operation must verify KEY A, for value-added operations must verify KEY B, etc). the position of three control bits in the access control byte as follows (byte 9 is spare byte, default is 0x69):



Byte 6	C23_b	C22_b	C21_b	C20_b	C13_b	C12_b	C11_b	C10_b
Byte 7	C13	C12	C11	C10	C33_b	C32_b	C31_b	C30_b
Byte 8	C33	C32	C31	C30	C23	C22	C21	C20
Byte 9								

（remark:_b means negation ,For example: if c11 is 1, c11_b is 0; c11 is 0, c11_b is 1 ）

1. **Control block**（block 3）the acess control of block 3 is different from the data block(blocks 0,1,2), its access control are as follows:

			key A		Control bit		Key B	
C13	C23	C33	Read	Write	Read	Write	Read	Write
0	0	0	Never	KeyA|B	KeyA|B	Never	KeyA|B	KeyA|B
0	1	0	Never	Never	KeyA|B	Never	KeyA|B	Never
1	0	0	Never	KeyB	KeyA|B	Never	Never	KeyB
1	1	0	Never	Never	KeyA|B	Never	Never	Never
0	0	1	Never	KeyA|B	KeyA|B	KeyA|B	KeyA|B	KeyA|B
0	1	1	Never	KeyB	KeyA|B	KeyB	Never	KeyB
1	0	1	Never	Never	KeyA|B	KeyB	Never	Never
1	1	1	Never	Never	KeyA|B	Never	Never	Never

（KeyA|B means key A or key B, Never means that can not be realized under any conditions)

For example: the block 3, access control bits is C13 C23 C33 = 100, means:

Password A: unreadable, verify KEYB correct, may write (change).

Access control: authentication KEYA or KEYB correctly, readable but can not write

Password B: unreadable, verify KEYB correctly, can write.）

2.The data block (block 0, block 1, block 2) the access control are as follows:

Control bit(X = 0,1,2)			Control condition(block 0,1,2)			
C1X	C2X	C3X	Read	Write	Increment	Decrement, transfer,Restore
0	0	0	KeyA|B	KeyA|B	KeyA|B	KeyA|B
0	1	0	KeyA|B	Never	Never	Never
1	0	0	KeyA|B	KeyB	Never	Never
1	1	0	KeyA|B	KeyB	KeyB	KeyA|B
0	0	1	KeyA|B	Never	Never	KeyA|B
0	1	1	KeyB	KeyB	Never	Never
1	0	1	KeyB	Never	Never	Never
1	1	1	Never	Never	Never	Never

（KeyA|B means key A or key B, Never means that can not be realized under any conditions)

For example:

When access control bits of the block 0 are C10 C20 C30 = 100, the key A or key B is correctly verified, readable; verify KEYB correctly, can write; can not do increment and decrement operation.