
S9 Series Contactless RFID Card Reader/ Writer



Contents

1. Introduction of S9 Series Contactless Card Reader	3
1.1 Overview.....	3
1.2 Specification.....	3
1.3 Device Interface.....	4
1.4 Packing List.....	4
1.5 Software.....	5
1.6 Typical Applications.....	5
1.7 Reader Type Description.....	5
1.8 Functions Description.....	6
1.9 API Functions List.....	6
1.10 Error codes and meanings.....	13
1.11 Desfire return codes and meanings.....	15
2. API Functions.....	17
2.1 Common Functions.....	17
2.2 Device Functions.....	28
2.3 CPU(SAM) Functions.....	31
2.4 S70 Card-Specific Functions.....	33
2.5 Ultralight Card-Specific Functions.....	34
2.6 Mifare Pro Card-Specific Functions.....	36
2.7 ICODE2 card-specific functions.....	36
2.8 AT88RF020 card-specific functions.....	41
2.9 Contactless CPU (ISO1443) card-specific functions.....	45
2.10. Desfire card-specific functions.....	46
2.11. 4442 card-specific function.....	60
2.12. 4428 card-specific functions.....	63
2.13. 24C64 card-specific functions.....	66
2.14. 125K(ID) Card-specific function.....	67
3. MIFARE ONE Card Structure.....	67

1. Introduction of S9 Series Contactless Card Reader

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 Standard size and Maximum 3 slots for SAM card of GSM11.11 size. Its multi-slots structure allows the reader to meet a higher demand of security in card applications. Users can connect the reader to the PC or other devices via RS232 Serial port or USB port(plug and play, no extra driver installation needed). And a CD for SDK (Software Development Kits) in the package includes examples for various development platforms, and users can use the DEMO.exe program to do the testing for the reader and the RFID cards.

S9 Reader is an essential front-end processing device for IC card applications and system integration. Its various and useful user interface functions enable the reader to easily read smart cards for functions such as charging fees, stored value card (SVC) and data query etc. Moreover, this reader is widely applied in a great number of occasions like industry and commerce, telecommunications, postal services, taxation, banking, insurance, medical, meeting attendance, Internet cafe management, gas stations, parking lots and so on.

1.2 Specification

Support Card Types:

- Contactless Cards: Cards in compliance with ISO14443 Type A/B, ISO15693 Mifare/DESFire/ICODE2/AT020/NXP etc.
- Contact Cards: ISO7816 logic encryption IC cards and the CPU card(T=0,1)

Security: can add 3 SAM card slots of Standard GSM11.11

Interface: USB (plug and play) / RS232

Antenna: Built-in antenna

Case Material: ABS plastic

Operating Frequency: 13.56MHz

Operating Voltage: DC5V \pm 10%

Data Transfer: 106Kbit/s

Operating Temperature: -20 °C ~ 60 °C

Protocols Compliance: ISO14443, ISO15693, ISO7816,
GSM11.11, FCC, CE

Programming Languages:

a variety of programming languages available with corresponding examples
(Please refer to "1.5 Software" for more information.)

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

Storage Capacity: the standard internal memory of the reader is 2K bytes
(Memory can be expanded upon users' needs)

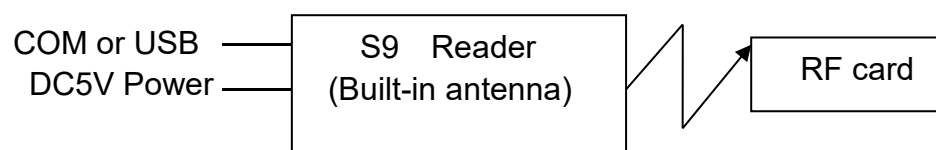
Display Mode: LED lights indicating the power mode or communication mode,
a user controllable beep

Connection Cable: 1.5M, USB cable (Serial K line, optional)

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

NW/GW: about 143g / 345g

1.3 Device Interface



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

1.4 Packing List

Including: a reader, a communication cable, a CD driver, and a warranty card.

1.5 Software

Including:

a. Demo program: S9.exe

b. Lib of Functions: Under Win32-dll directory

c. Application Examples:

There are many examples from various development platforms (such as VB, DELPHI, VC, C#, JAVA etc.) under the directory of EXAMPLES.

1.6 Typical Applications

E-passports / Internet banking and online shopping / network access and access control system / digital signature / customer bonus plan / stored value card / identification / e-ticketing / parking system/ loyalty program / time attendance management / vending machine.

1.7 Reader Type Description

Naming Rule: S9-X X-X X-XX

The first two (S9): illustrate S9 series reader with RS232 serial port and USB port, supporting both contact and contactless smart cards, without a display screen.

The third place: shows supportable card types , "A" for S50, S70 and Ultralight cards; "B" for 14443A card; "C" for 14443A, 14443B and 15693 cards; "D" for 125K card compatible with ID card number; "E" for S50 card; "F" for S50 card and 125K ID card.

The forth place: shows the communication mode. "S" for serial port, "U" for USB port, "E" for USB virtual COM port, "P" for PC/SC standards.

The fifth place: shows the number of big card slots. "0"- without slots,

"1"- with one slot.

The sixth place: shows the number of PSAM card slots, "0"-without slots,

"1"-with one slot, "2"- with two slots, "3"-with three slots.

The last two places: are Arabic numbers in sequential order starting from 01

1.8 Functions Description

Function calls should follow the following rules:

- 1) At the start of the program, call function `fw_init()` to initialize Serial Interface or USB interface.
- 2) Call function `fw_load_key()` to load key from the card to the reader, one sector at a time.
- 3) Call function `fw_card()` to get the serial number of the card. (The counterpart is to call functions of `fw_request()`, `fw_anticoll()`, `fw_select()` in a row.)
- 4) Call function `fw_authentication()` for key authentication of the device and the card, one sector at a time.
- 5) To read, write, initialize the value, do increment and decrement in the authenticated sector. Repeat steps (3) and (4) when executing these operations in any other sector.
- 6) When the operation is complete, call the function `fw_halt()` to halt the work of a certain card.
- 7) Before exit or exit-on-error, call the function `fw_exit()` to exit the communication port. Or errors will appear in step (1) next time.
- 8) To call any function, please refer to examples under directory Example\.

1.9 API Functions List

Common Functions

Function Name	Description
fw_init	To initialize the communication port
fw_exit	To exit the port
fw_card	To find card
fw_card_hex	To find card in hex format
fw_card_str	To find card in string format
fw_request	Card response to a request
fw_anticoll	Anti-collision
fw_select	To select a card
fw_load_key	To load keys to sectors from the card to the reader
fw_authentication	To authenticate keys of sectors
fw_read	To read data from card (For S50 and S70 cards)
fw_write	To write data to card (For S50 and S70 cards)
fw_halt	To halt the operation of the card
fw_des	To encrypt/decrypt
fw_changeb3	To change keys of sectors (For S50 and S70 cards)
fw_initval	To initialize the value of block
fw_increment	To do increment
fw_decrement	To do decrement
fw_readval	To read value
fw_restore	To store data from EEPROM to the internal register of the card
fw_transfer	To transfer the data from the card to EEPROM
fw_config_card	To set card types
a_hex	To convert hex string to the corresponding characters of ordinary ASC
hex_a	To convert ordinary ASC to the corresponding hex string

Device Functions:

Function name	Description
fw_beep	To beep
fw_disp_mode	To set LED display mode
fw_gettime	To get system time of the reader
fw_settime	To set the date/time in the reader
fw_getver	To get the hardware version of the reader
fw_srd_eeprom	To read EEPROM
fw_swr_eeprom	To write EEPROM
fw_reset	RF power-on reset
fw_ctl_mode	To set LED control mode
fw_LED_disp8	To set LED to display 8 digits randomly
fw_lcd_setbright	LCD light on & off
fw_lcd_dispstr	LCD display string
fw_lcd_dispclear	To clear LCD string

CPU (SAM) card-specific functions

Function Name	Description
fw_cpureset	Power-on reset of the CPU card
fw_setcpu	To set slots for the CPU (SAM) card
fw_cpuapdu	Data transfer between CPU card and APDU
fw_setcpupara	To set parameters of the CPU card

4442 card-specific functions

Function name	Description
fw_authentikey_4442	To authenticate the key of 4442 card
fw_read_4442	To read the 4442 card
fw_write_4442	To write to the 4442 card
fw_getProtectData_4442	To get protected data from the 4442 card
fw_setProtectData_4442	To set protected data in the 4442 card
fw_changkey_4442	To change the key of the 4442 card
fw_cntReadError_4442	To count read-errors of the 4442 card

4428 Card-Specific Functions

Function Name	Description
fw_authentikey_4428	To authenticate the key of the 4428 card
fw_read_4428	To read 4428 card
fw_write_4428	To write to the 4428 card
fw_getProtectData_4428	To get protected data from the 4428 card
fw_setProtectData_4428	To set protected data in the 4428 card
fw_changkey_4428	To change the key of the 4428 card
fw_cntReadError_4428	To count read-errors of the 4428 card

S70 Card-Specific Functions

Function Name	Description
fw_read_S70	To read the S70 card
fw_write_S70	To write in the S70 card

Ultralight Card-Specific Functions

Function name	Description
fw_request_ultralt	The response of the Ultralight Card to a request
fw_anticall_ultralt	Anti-collision of the Ultralight card
fw_select_ultralt	To select the Ultralight card
fw_read_ultralt	To read the Ultralight card
fw_write_ultralt	To write in the Ultralight card
fw_halt_ultralt	To halt the operation of the Ultralight card

Mifare Pro Card-Specific Functions

Function Name	Description
fw_reset_mifarepro	Power-on reset of the Mifare Pro card
fw_apdu_mifarepro	Data transfer between the Mifare Pro Card and APDU

ICODE2 Card-Specific Functions

Function Name	Description
fw_inventory	The response of the ICODE2 card to a request
fw_stay_quiet	To stay in quiet state of the ICODE2 card
fw_select_uid	To select UID state of the ICODE2 card
fw_reset_to_ready	Reset to ready state of the ICODE2 card
fw_readblock	To read block data of the ICODE2 card
fw_writeblock	To write block data in the ICODE2 card
fw_lock_block	To lock a certain block of the ICODE2 card
fw_write_afi	To write AFI of the ICODE2 card
fw_lock_afi	To lock AFI of the ICODE2 card
fw_write_dsfid	To write DSFID of the ICODE2 card
fw_lock_dsfid	To lock DSFID of the ICODE2 card
fw_get_systeminfo	To get system information of the ICODE2 card
fw_get_securityinfo	To get security information of the ICODE2 card

AT88RF020 Card-Specific Functions

Function Name	Description
fw_request_b	The response of the AT88RF020 card to a request
fw_attrib	To select among the cards that may have responded to a given REQb/WUPb command
fw_check_at	To check the password of the AT88RF020 card
fw_read_at	To read the AT88RF020 card
fw_write_at	To write in the AT88RF020 card
fw_changekey_at	To change the key of the AT88RF020 card
fw_lock_at	To lock the AT88RF020 card
fw_halt_at	To halt the operation of the AT88RF020 card
fw_count_at	To write Page 2 of the AT88RF020 card

Contactless CPU (ISO1443) Card-Specific Functions

Function Name	Description
fw_pro_reset	Power-on reset of the CPU card
fw_pro_commandlink	Data transfer between CPU card and commandlink

24C64 Card-Specific Functions

Function Name	Description
fw_read_24c64	To read the 24c64 card
fw_write_24c64	To write in the 24c64 card
fw_check_24c64	To check if the 24c64 card is inserted in the slot

DESFire Card-Specific Functions

Function Name	Description
fw_anticoll2	The second anti-collision
fw_select2	The second finding the DESFire card
fw_reset_desfire	Power-on reset of the DESFire card
fw_authen_desfire	To authenticate the key of the DESFire card
fw_getver_desfire	To return manufacturing related data of the DESFire card
fw_getAIDs_desfire	To get application identifier of the card
fw_selectApp_desfire	To select one application for further access
fw_getKeySetting_desfire	To get information on the DESFire card and application master key settings
fw_getKeyver_desfire	To get the master key version of the card
fw_createApp_desfire	To create new applications on the card
fw_delAID_desfire	To permanently deactivate applications on the card
fw_changeKeySetting_desfire	To change the master key settings

fw_changeKey_desfire	To change any key stored on the card
fw_getFileIDs_desfire	To get file identifier of all active files within the currently selected application
fw_getFileProper	To get information on the properties of a specific file
fw_changeFileSetting	To change the access parameters of an existing file
fw_createDataFile_desfire	To create a standard data file
fw_createBackupDataFile_desfire	To create files for the storage of plain unformatted user data
fw_createValueFile_desfire	To create files for the storage and manipulation of 32-bit signed integer values
w_createCsyRecord_desfire	To create cycle record
fw_delFile_desfire	To permanently deactivate a file from the card
fw_write_desfire	To write data to Standard Data files or Backup Data files
fw_read_desfire	To read data from Standard Data files or Backup Data files
fw_getvalue_desfire	To read the currently stored value from Value files
fw_credit_desfire	To increase a value stored in a Value file
fw_debit_desfire	To decrease a value stored in a Value file
fw_writeRecord_desfire	To write data to a record in a Cyclic or Linear Record file
fw_readRecord_desfire	To read out a set of complete records from a Cyclic or Linear Record file
fw_clearRecord_desfire	To reset a Cyclic or Linear Record file to empty state

fw_commitTransfer_desfire	To commit the changes made in the backed-up files
fw_abortTransfer_desfire	To invalidate all previous write accesses on Backup Data files, Value files and Record files within one application
fw_formatPICC_desfire	To format the DESFire card

125K (ID) Card-Specific Functions

Function Name	Description
fw_read_SerialNumberID	To read the serial number of the 125K card

1.10 Error codes and meanings

Error Return Codes	Positive/Negative	Meanings
0x10(016)	-	Communication error
0x11(017)	-	Timeout error
0x20(032)	-	Error while opening port
0x21(033)	-	Error while getting port parameters
0x22(034)	-	Error while setting port parameters
0x23(035)	-	Error while closing port
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 data received error

0x50(080)	-	Error while there are not enough loops for a decrement or a debit
0x51(081)	-	CPU data XOR error
0x52(082)	-	IP address error through RS-485 communication
0x73(115)	-	Error while getting the version number
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)	+	No card or certificate error
0x02(002)	+	Data validation errors
0x03(003)	+	Null value error
0x04(004)	+	Authentication failure error
0x05(005)	+	Parity error
0x06(006)	+	Communication error between the reader and the card
0x08(008)	+	Error on serial number while reading the card
0x09(009)	+	Error of wrong password type
0x0a(010)	+	Card without authentication
0x0b(011)	+	Incorrect number of bits while reading the card
0x0c(012)	+	Incorrect number of bytes while reading the card
0x0f(015)	+	Write card error
0x10(016)	+	Errors during increment or credit

0x11(017)	+	Errors during decrement or debit
0x12(018)	+	Read card error
0x13(019)	+	Data buffer overflow
0x15(021)	+	Transmission frame error
0x17(023)	+	Unknown transmission requirements
0x18(024)	+	Anti-collision error
0x19(025)	+	Sensor module reset error
0x1a(026)	+	Non-authentication interface
0x1b(027)	+	Module communication timeout
0x3c(060)	+	Abnormal 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 error
0x03	Numerical overflow
0x05	Parity error
0x06	Communication error
0x08	Error of reading the serial number in the process of anti-collision
0x0B	Incorrect number of bits receiving from the card
0x0C	Backup files remain unchanged; CommitTransaction or AbortTransaction not needed
0x0E	Insufficient memory to complete the instruction
0x1C	Command not supported
0x1E	CRC or MAC code does not match, invalid bytes sequence for

	encoding
0x40	The specified key 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	Requested application identifier not exist
0xA1	Unrecoverable error in application, the application will be invalid
0xAE	The requested command not allowed in the current state of authentication
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 an unrecoverable error
0xCE	Maximum number of applications be reached (28); No more applications created
0xDE	The file or application can not be created; the file number or application number has existed.
0xEE	Internal backup and the reversal mechanism can not complete the writing operation because of power-down.
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

To initialize the communication port

Parameters

port:COM Type

Set value 0~19: Serial port 1~20

Set value 100(baud rate invalid in this case): USB port

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 1, Baud rate:9600
```

Remark

If there are more than one readers (USB interface) connected to the PC, call this function to get their HDCs separately. Here is an example:

```
int icdev1,icdev2,icdev3;/* presume there are three readers connected*/
```

```
icdev1=fw_init(100,0);/*get HDC of the first device*/
```

```
icdev2=fw_init(100,0);/*get HDC of the second device*/
```

```
icdev3=fw_init(100,0);/*get HDC of the third device*/
```

int fw_exit(int icdev);

Description

To 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 HDC of the port; It must be released before next connection.

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

Description

To find card and return the serial number of a certain card in working area (This function contains these functions of fw_request,fw_anticoll,fw_select)

Parameters

icdev: Value of Device Handle

_Mode: Find card mode

Value:

0——IDLE mode, to operate one card at a time

1——ALL mode, to operate several cards at a time;

_Snr: returned the serial number of the card

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 procedures, function fw_halt is called to halt the card operation. Only after the card is taken out and brought back to the operation area, it can be operated by the reader again.
- 2) When this function is called, note that incoming data type of the last argument must be the address of an unsigned long integer variable (unsigned char long), otherwise it will be automatically converted into a signed one. The corresponding functions of fw_card_hex and fw_card_str are recommended to call in order to return hex card number or to return decimal card number.

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

Description

To find card and get the serial number of a certain card (hex string)

Parameters

icdev: Value of Device Handle

_Mode: Find card mode

Value:

0——IDLE mode, to operate one card at a time;

1——ALL mode, to operate several cards at a time;

_Snrbuf: Return a hex card number (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

To find card and get the serial number of a certain card (Decimal string)

Parameters

icdev:Value of Device Handle

_Mode: Find card mode

Value:

0——IDLE mode, to operate one card at a time;

1——ALL mode,to operate several cards at a time;

strSnr: Return a decimal card number (10-digit sequences)

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

Card response to a request

Parameters

icdev:Value of Device Handle

_Mode:Find card mode

Value:

0——IDLE mode,to operate one card at a time;

1——ALL mode,to operate several cards at a time;

Tagtype: Return values of the card type

The meaning of values is shown as follows:

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

Anti-collision, to return the serial number of the card

Parameters

icdev:Value of Device Handle

_Bcnt: Anti-collision level; The value should be 0.

_Snr:Return the serial number address of the card

Return Value

0 if successful; otherwise, Nonzero.

Example

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

Remark

This function fw_anticoll should be called immediately after the request command unless the card serial number is known.

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

Description

To select the card with a specified serial number from several cards

Parameters

icdev: Value of Device Handle
 _Snr: Card serial number
 _Size: [out] The size of card capacity

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st,type;
unsigned char size;
unsigned long snr;
fw_request(icdev,0,&type);
fw_anticoll(icdev,0,&snr);
st=fw_select(icdev,snr,&size);
```

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

Description

To load keys to RAM module

Parameters

icdev: Value of Device Handle
 _Mode: The mode of loading keys
 Value is set as follows:

For each sector of M1(S50) card, there are three sets of corresponding keys (KEYSET0, KEYSET1, KEYSET2). Each set contains Key A and Key B. In total, there are six sets of keys, number 0~2, 4 ~ 6 to represent each of them:

- 0 - KEYSET0 of KEY A
- 1 - KEYSET1 of KEY A
- 2 - KEYSET2 of KEY A
- 4 - KEYSET0 of KEY B
- 5 - KEYSET1 of KEY B
- 6 - KEYSET2 of KEY B

_SecNr: Sector number (0-15 for S50 Card, 0 for ML Card)

_Nkey: Card keys 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

Keys authentication

Parameters

icdev:Return device handle value of fw_init

_Mode:The mode of loading keys; Value set is as follows:

For each sector of S50 card, there are three sets of corresponding keys (KEYSET0, KEYSET1, KEYSET2). Each set includes KEYA and KEYB. In total, there are six sets of keys, number 0~2, 4 ~ 6 to represent each of them:

- 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 of keys to authenticate

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

To check key authentication

If this function is called, function fw_load_key isn't necessary.

Parameters

icdev:Value of Device Handle

_Mode:The mode of loading keys; value is as follows:

For each sector of S50 card, there are three sets of corresponding keys (KEYSET0, KEYSET1, KEYSET2). Each set includes KEYA and KEYB. In total, there are six sets of keys. No. 0~2,4 ~ 6 to represent each of them:

- 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 of keys to authenticate

passbuff: The keys of the sector(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

To read data of the card

For S50 (M1) card :Read one block of data(16 bytes) each time

For ML card: Read two pages of the same attribute each time

(0 and 1,2 and 3 ,...), 8 bytes.

Parameters

icdev:Value of Device Handle

_Adr:S50(M1) Card-- the block address (0-63), MS70(0-255);

_Data:[out] Data read out

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 that the string of HEX function is in the form of hex, while the string of the corresponding normal

one is in the form of ASC codes. For example, if the actual data of the second block is: "1234567890abcd", call function `fw_read_hex` and it will return the string: "31323334353637383930616263646566." The same below.

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

Description

To write data to card.

For S50 (M1) card: Write data of one block (16 bytes) at a time.

For ML card: Write data of one page (4 bytes) at a time.

Parameters

`icdev`:Value of Device Handle.

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

`_Data`:Data to be written

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,unsigned char _Adr,unsigned char *_Data)`

Remark

The second parameter of this function is the block number. For the S50 (M1) card, Block 4 of each sector is the password block; for S70 card, the password block is the 4th block of the first 32 sectors and the 16th block of the last eight sectors. Write the password block carefully because rewriting may damage the operation sectors.

`int fw_halt(int icdev)`

Description

To Halt operation of the card

Parameters

`icdev`:Value of Device Handle

Return Value

0 if successful; otherwise, Nonzero.

Example

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

Remark

Parameter `_Mode` is in the function `fw_card()`. If `_Mode=0`, `fw_halt()` is called after operations and then the card enters Halt state. The card should be taken out of and brought back again to operating area to find the card.

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

Description

To encrypt or decrypt with DES algorithm

Parameters

key:Secret keys

sour:Source of data for encrypt/decrypt

dest:Out data after encrypt/decrypt

m: Mode of encrypt/decrypt,encrypt if m=1; decrypt if 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

To update data of Block 3

(Update data of Block 15 if sector No. is >31 for S70 card)

Parameters

icdev:Value of Device Handle

_SecNr:Sector number(S50 card: 0-15, S70 card :0-39)

_KeyA:Key A

_CtrlW:Control Words of keys

_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);/*To change keys of  
sector 1*/
```

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

Description

To Initialize value of blocks.

Parameters

icdev:Value of Device Handle

_Adr:Address of blocks

_Value:Initialize value

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

In value operation, value initialization function is called before other functions like reading data reading, increment and decrement.

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

Description

Increment value of certain blocks

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 this function is called, the function fw_transfer should be called immediately. Otherwise,the value will not be updated successfully.

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

Description

To read value of certain blocks.

Parameters

icdev:Value of Device Handle

_Adr:Address of blocks

_Value:[out]Value read out

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 blocks

Parameters

icdev:Value of Device Handle

_Adr:Address of blocks

_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);    /*Value decrement of Block 1 */  
st=fw_transfer(icdev,1);
```

Remark

After this function is used,the function fw_transfer should be called immediately. Otherwise,the value will not be updated successfully.

int fw_restore(int icdev,unsigned char _Adr);

Description

Return function, to transfer data of EEPROM to internal registers of card

Parameters

icdev:Value of Device Handle

_Adr: Returned block address

Return Value

0 if successful; otherwise, Nonzero.

Example

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

Remark

This function is used to transfer data from one block of the card to the internal register. And then the function fw_transfer() is used to transfer the data from internal register to the other block of the card. The value is transferred between blocks in this way.The function can only be used for value blocks.

int fw_transfer(int icdev,unsigned char _Adr);

Description

This function is to transfer the content of the internal register to the transmitted address.The sector must be authenticated for this operation and the transfer function can only be called directly after increment, decrement or restore.

Parameters

icdev:Value of Device Handle

_Adr:The address of blocks 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);
```

The content of Block 1 will be transferred to Block 2 by these commands.

Remark

Please refer to the description of fw_restore

__int16 fw_config_card(HANDLE icdev,unsigned char flags);

Description

To configure the card type

Parameters

icdev: Value of Device Handle

flags: Card type setting 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 to be converted into ordinary characters (From long characters to short ones).

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 to be converted into hexadecimal characters (From short characters to long ones).

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

To beep

Parameters

icdev:Value of Device Handle

unsigned int _Msec:Time to 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

To set display mode for the reader, settings will be saved at shutdown

Parameters

icdev:Value of Device Handle

mode:Model of 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);//To set time mode
```

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

Description

To get time,week,date from the reader

Parameters

Icdev: Value of Device Handle

Time: The length of returned data is 7 bytes.

Format is "year, days of week, month, date, 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

To get version of the 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

To set time of the reader

Parameters

icdev: Value of Device Handle

time: The length is 7 bytes.

Format is "year, days of the week, month, date, hour, minute, 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

To get remark information of the 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

To write remark information to the 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

Power-on reset of the card

Parameters

icdev :Handle of Device

_Msec : Reset time, the unit is milliseconds (ms)

If value is 0, frequency is off; If value is 1,2 ... , reset time is 1 ms, 2 ms ...

Return Value

<0 error. The absolute value is the error number

=0 successful

Example

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

int fw_ctl_mode(int icdev,unsigned char mode);

Description

To set Control mode of LED display

Parameters

icdev:Value of Device Handle

mode:Display model

0——Controlled by the PC

1——Controlled by the reader

Return Value

0 if successful; otherwise, Nonzero.

Example

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

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

Description

LED display random digits

Parameters

icdev:Value of Device Handle

strlen:Numbers of digits to display (set to 8)

dispstr:Digits to 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

To set LCD backlight on or off

Parameters

icdev: Value of Device Handle

bright: The sign of LCD light on or off.

15—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**

To set strings to display on LCD

Parameters

icdev: Value of Device Handle

Digit : Strings 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 of the CPU Card,
will automatically judge the card protocol after being 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
0x0c: 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 as 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 to send
sbuff: Sender buffer
rlen: Length of information received
rbuff: [out] Receiver buffer

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 cputetu)

Description

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

Parameters

ICDev: Handle of Reader Device

cputype: Type of Deck.

0x0c: Main deck(ISO7816);

0x0d: SAM1;

0x0e: SAM2;

0x0f: SAM3.

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

cputu: 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

To read S70 card, a block of data (16 bytes) can be read each time

Parameters

icdev: Value of Device Handle

_Adr: The block address (0-255)

_Data: Data to be read

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

To write S70 card, a block of data (16 bytes) can be written each time

Parameters

icdev: Value of Device Handle

_Adr: The block address(0-255)

_Data: Data to be written

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 to find the Ultralight card

Parameters

icdev: Value of Device Handle

_Mode: Mode of finding the Ultralight card (0 or 1)

Return Value

0 if successful; otherwise, Nonzero.

Example

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

Remark: The value of parameter _Mode can be set to 0 or 1

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

Description

Anti-collision

Parameters

icdev: Value of Device Handle

_Snr: [out] The serial number of the Ultralight card

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

To select a certain card from several Ultralight cards

Parameters

icdev: Value of Device Handle

_Snr: The serial number of the card

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

To read data from the Ultralight card

Parameters

icdev: Value of Device Handle
iPage: Index number of pages
redata: Data read out

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

To write data to Ultralight card

Parameters

icdev: Value of Device Handle
iPage:Index number of pages
sdata: Data to be written

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

To halt operations of the Ultralight card

Parameters

icdev: Value of Device Handle

Return Value

0 if successful; otherwise, Nonzero.

Example

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

Remark

After this function is executed, the below functions are called in a row if the data is read again: fw_request_ultralt , fw_anticall_ultralt and fw_select_ultralt.

2.6 Mifare Pro Card-Specific Functions

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

Description

Power-on reset of the Mifare Pro Card

Parameters

icdev: Value of Device Handle

rlen: Length of reset data

rbuff: Buffer for returned reset data

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

Data transfer between Mifare Pro Card and Application Protocol Unit

Parameters

icdev: Value of Device Handle

slen : Length of information sent

sbuff : Sender buffer

rlen : Length of returned data

rbuff : Receiver buffer

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

The request of ICODE2 Card

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 family identifier

Masklen: Mask length

rlen: Length returned

rbuffer: 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 multiple cards;
```

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

Description

Card to enter Quiet state, and will not return any response data.

Parameters

icdev: Value of Device Handle

flags: Request flags;Value can be set 0x22;

UID: Unique identifier of the card (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

Card to enter Select state

Parameters

icdev: Value of Device Handle

flags: Request flags;Value can be set 0x22;

UID: Unique identifier of the card (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

Card to enter Ready state

Parameters

icdev: Value of Device Handle

flags: Request flags; Value can be set 0x22;

UID: Unique identifier of the card (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

To read block data from ICODE2 card

Parameters

icdev: Value of Device Handle

flags: Request flags; Value can be set 0x22;

startblock: The address of the starting block (range 0-27)

blocknum: The number of blocks to read at a time(range 1-6)

UID: Unique identifier of the card (card number)

rlen: Length of returned bytes

rbuffer: Returned data of blocks

Return Value

0 if successful; otherwise, Nonzero.

Example

```
int st;
```

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

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

Description

To write data to blocks to the Icode2 card

Parameters

icdev: Value of Device Handle

flags: Request flags; Value can be set 0x22;

startblock: The address of the starting block (range 0-27)

blocknum: The number of blocks to write at a time (range 1-6)

UID: Unique identifier of the card (card number)

rlen: Length of bytes to write

rbuffer: Data written in blocks

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); //To write one block of  
data starting from Block 1//
```

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

Description

To lock data of blocks

Parameters

icdev: Value of Device Handle
flags: Request flags; Value can be set 0x22;
block: The address of the starting block (range 0-27)
UID: Unique identifier of the card (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

To write AFI

Parameters

icdev: Value of Device Handle.
flags: Request flags;Value can be set 0x22;
AFI: Application family identifier
UID: Unique identifier of the card (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

To lock AFI

Parameters

icdev: Value of Device Handle
flags:Request flags;Value can be set 0x22;
AFI: Application family identifier

UID:Unique identifier of the card (card 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

To write DSFID

Parameters

icdev: Value of Device Handle.

flags: Request flags;Value can be set 0x22;

DSFID: Data storage format ID

UID:Unique identifier of the card (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

To lock DSFID

Parameters

icdev: Value of Device Handle.

flags:Request flags;Value can be set 0x22;

DSFID: Data storage format ID

UID: Unique identifier of the card (card 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

To read card information

Parameters

icdev: Value of Device Handle

flags: Request flags;Value can be set 0x22;

UID: Unique identifier of the card (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

To read security state information of the card

Parameters

icdev: Value of Device Handle
flags: Request flags;Value can be set 0x22;
startblock: The address of the starting block
blocknum: The number of blocks
UID: Unique identifier of the card (the 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 is to send the command of finding card and it is called to select a new card.

Parameters

icdev: Value of Device Handle
_Mode: Find 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, the value 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_attrb(HANDLE icdev,unsigned char *PUPI, unsigned char
CID);**

Description

To select a card from those which may have responded to a given REQb / WUPb command, and then assign an ID number to each selected 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 further access.

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_attrb(icdev,PUPI,0);

```

Remark:

If several selected 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_attrb"

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

To read the card according to its 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-byte 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

To write data 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

To lock the card; Certain areas of card can be locked and the locked areas can be read only.

Parameters

icdev: Value of Device Handle
Adr : Block address No.
sbuffer: Lock flag ; The meaning of the first byte is 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 if the card is in Active state.

Parameters

icdev: Value of Device Handle
cid : Card ID number, see parameter CID in “fw_attrb”

key : 8-byte 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 and the corresponding signature information will be written to the first 6 bytes in the 2nd page according to CID of the card.

Parameters

icdev: Value of Device Handle
cid : Card ID number, see parameter CID in “fw_attrb”
key : 8-byte 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

Power-on reset of the card

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: Function “fw_card” must be called before this one 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: Length of information to send

sbuff: Command to send

rlen: Length of returned information

rbuff: Returned information

tt: Delay time (Unit: 10ms)

FG: Length of block segmentation

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-collision level set the value 0

_Snr: Returned 5-byte card information

The first is 0x88 and the other four bytes are high bytes 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 from 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

Power-on reset of the DESFire card

Parameters

icdev: Value of Device Handle

rlen: Length of reset information

rdata: Returned 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 authenticate

key: 16-byte key

sessionKey: The session key returned after a successful key authentication.

Return Value

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

Example

//Verify Key No. 1

```
int st;  
char  
curkey[17]={0x11,0x22,0x33,0x44,0x55,0x66,0x77,0x88,0x11,0x22,0x33,0x44,  
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

To return manufacturing data of the DESFire card

Parameters

icdev: Value of Device Handle

rlen: Length of returned data

version: Returned data of the card manufacturer

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

To get application identifier

Parameters

icdev: Value of Device Handle

rlen: Length of returned data

AIDs: Returned 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

To 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

To get the master key settings

Parameters

icdev: Value of Device Handle

Rlen: Length of returned data

Setbuf: Set the master key of the application (or the card)

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

To get the version of master key

Parameters

icdev: Value of Device Handle

keyNo: Key number

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

To create applications

Parameters

icdev: Value of Device Handle

AID: Key numbers

KeySetting: Set application master key

The meaning of 8-bit application master key can be described 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

To delete applications

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

To 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 described as below:

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

8-bit master key of the card represents the following meanings:

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

Each bit of card key settings represents the meaning as follows:

Bit7-Bit4: reserved, be set to 0.

Bit3: The code determines if it is allowed to change the master key settings:

0: Configuration can not be changed any more (frozen).

1: The configuration can be modified if the card's master key is authenticated.(the default setting)

Bit2: The code determines if the authentication for the card's master keys is needed when creating /deleting applications:

0: Applications can not be created / depleted until the master key of card has been authenticated successfully.

1: Applications are also allowed to be created without the authentication of the master key. (default setting)

Before deleting any applications, master key of applications or of the card must be authenticated successfully.

Bit1: This code determines whether it needs to certificate the card master key for access to the application directory:

0: If fw_getAIDs_desfire and fw_getKeySetting_desfire are called, a successful authentication of the master key is needed.

1: The master key authentication is not needed (default setting) if fw_getAIDs_desfire and fw_getKeySetting_desfire are called.

Bit0: This code determines if the master key of the card can be modified:

0: The master key can not be modified (frozen).

1: The master key can be modified (The current master key must be certified, which is a default setting).

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

To change the master key

Parameters

icdev: Value of Device Handle

sessionKey: Session key

curkey:The current key

keyNo: Key number
newkey: The 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

To get file identification numbers of current applications

Parameters

icdev: Value of Device Handle
Rlen: Length of returned data
fileIDs: File identification number (Each byte is 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

To get file settings

Parameters

icdev: Value of Device Handle
fileNo: File ID
rlen: Length of returned data
fileProper: Property settings of files

Return Value

0 if successful; otherwise, Nonzero. Reference Table 1.11

Example

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

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

Description

To change file settings

Parameters

icdev: Value of Device Handle

fileNo: File ID

comSet: Methods of data transmission:

0: Cleartext transmission

1: MAC code validation

3: DES/3DES Encryption

accessRight: Access right

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

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, to write, to read/write  
and to modify settings, Key No. 2 is required to be authenticated.*/
```

```
st= fw_changeFileSetting (icdev,0x01,comSetting,accessRights,sesskey);
```

```
/* sesskey is the session key after the key authentication */
```

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

Description

To create a standard data file

Parameters

icdev: Value of Device Handle

fileNo: File ID

comSet: Methods of data transmission:

0: Cleartext transmission

1: MAC code validation

3: DES/3DES Encryption

accessRight: Access right

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

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;// Cleartext transmission
unsigned char accessRights[3]={0x22,0x22};/*To read, to write, to read/write
and to modify settings, Key No. 2 is required to be authenticated. */
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

To create value files

Parameters

icdev: Value of Device Handle

fileNo: File ID

comSet: Methods of data transmission:

0: Cleartext transmission

1: MAC code validation

3: DES/3DES Encryption

accessRight: Access right

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

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};/* To read, to write, to read/write
and to modify settings, Key No. 2 is required to be authenticated.*/
unsigned char lower[4]={0x00,0x00,0x00,0x00};// Minimum value is 0
unsigned char upper[4]={0xff,0xff,0xff,0x00};// Maximum value is 0xffffffff
unsigned char value[4]={0x32,0x00,0x00,0x00};// The current value is set to 50
(0x32)
unsigned char enable=0x01;// Support limited memory
st=fw_createValueFile_desfire
(icdev,0x02,comSetting,accessRights,lower,upper,value,enabled);
```

```
int fw_createCsyRecord_desfire(int icdev,unsigned char fileNo,unsigned char comSet,unsigned char* AccessRight,unsigned char* RecordSize,unsigned char* MaxNum);
```

Description

To create cycle record files

Parameters

icdev: Value of Device Handle

fileNo: File ID

comSet: Methods of data transmission:

0: Cleartext transmission

1: MAC code validation

3: DES/3DES Encryption

accessRight: Access right

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

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};/*To read, to write, to read/write
and to modify settings, Key No. 2 is required to be authenticated. */
unsigned char recordLen[3]={0x20,0x00,0x00};//Each record is 32 bytes.
unsigned char number[3]={0x10,0x00,0x00};//The file has 16 records at the
most
st= fw_createCsyRecord_desfire
(icdev,0x03,comSetting,accessRights,recordLen,number);
```

```
int fw_delFile_desfire(int icdev,unsigned char fileNo);
```

Description

To delete files

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

To write data files

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 is the session  
key after the key authentication*/
```

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

Description

To read data file

Parameters

icdev: Value of Device Handle
fileNo: File ID
offset: Offset address
length: Length of data to read
revData: 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 is the session  
key after the key authentication */
```

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

Description

To get the value of value files

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 is the  
session key after the key authentication */
```

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

Description

Value of credit file

Parameters

icdev: Value of Device Handle
fileNo: File ID
value: The gotten credit value
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 is the session key  
after the key authentication */
```

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

Value of debit files

Parameters

icdev: Value of Device Handle
fileNo: File ID
value: The gotten debit 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 is the session key  
after the key authentication */
```

Remark

After this function is called successfully, fw_commitTransfer_desfire must

be called to put 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

To write one record to record files

Parameters

icdev: Value of Device Handle

fileNo: File ID

offset: Offset address

length: 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 is the
session key after the key authentication */
```

Remark

After this function is called successfully, fw_commitTransfer_desfire must be called to put 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

To read record file

Parameters

icdev: Value of Device Handle

fileNo: File ID

offset: Offset address, the initial record number

length: Number of records read from the offset address

revData: Data read

SgRecordlen: Length of an individual record

Rlen: Total length of data read

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 is the session key  
after the key authentication */
```

Remark:

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

int fw_clearRecord_desfire(int icdev,unsigned char fileNo);

Description

To clear data of record files

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 is called successfully, fw_commitTransfer_desfire must be called to put the operation into effect.

int fw_commitTransfer_desfire(int icdev);

Description

To 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 credit, debit, writing and clearing data of the record, this function must be called to put the operation into effect.

int fw_abortTransfer_desfire(int icdev);

Description

To 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 the 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

If this function was called 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

To read data from the 4442 card

Parameters

icdev:Value of Device Handle

_Adr: Starting address to read data(0~255)

_Data: Data returned

length: Length of Data read out(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.
2. The value of parameter _Adr should not exceed 255.

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

Description

To write data to 4442 card

Parameters

icdev: Value of Device Handle

_Adr: Starting address to write data(0x30~0xff)

_Data: Data to be written

length: Length of data written

Return Value

0 if successful; otherwise, Nonzero.

Example

```
// write 4 chars starting from the 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 not exceed the actual length of data to be written.

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

Description

To read protected bits

Parameters

icdev: Value of Device Handle
_Adr: Starting address to read data (must set 0)
_Data: Returned protect bits
length: Length of data read out (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-byte data protection whose address is x00-0x20. The 4-byte data read out corresponds to each equivalent bit. 0 for write-protected, 1 for not write-protected;

int fw_setProtectData_4442(int icdev,unsigned char _Adr,unsigned char *_Data,int length);

Description

To write protected bits

Parameters

icdev: Value of Device Handle
_Adr: Starting address to write data (0~32)
_Data: Data to be written
length: Length of data written(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 from 0 to 32.
- 2: Data written 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

To authenticate keys

Parameters

icdev: Value of Device Handle

_Adr: Starting address to authenticate keys (must set 0)

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

key: Keys to be authenticated(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

The card will be locked after three wrong attempts.

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

Description

To change keys of the 4442 card

Parameters

icdev: Value of Device Handle

_Adr: Starting address to change keys (must set 0)

rlen:Length of keys to be changed(must set 3)

key: The key-data (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

Counts of reading errors

Parameters

icdev: Value of Device Handle
cntReadError: Counts of read-errors

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

To read data from the 4428 card

Parameters

icdev: Value of Device Handle
_Adr: The starting address to read data
_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 not exceed 1024, otherwise reading will fail.
- 2) The parameters _Adr should not exceed 1023.

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

Description

To write data to 4428 card

Parameters

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

Return Value

0 if successful; otherwise, Nonzero.

Example

```
//write 4-byte data starting from the address 0x30
```

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

Remark :

- 1) The value of parameter _Adr should be between 0x0 and 0x3ff.
- 2) The value of parameter length should not exceed the actual length of data to be written.

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

Description

To read protected bits

Parameters

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

To write protected bits

Parameters

icdev: Value of Device Handle
 _Adr: The starting address to write 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) The value of parameter _Adr should be between 0 and 1023.
- 2) The written data should be consistent with the corresponding data stored in card.
- 3) The value of parameter length should not exceed 1024.

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

Description

To authenticate keys of 4428 card

Parameters

icdev: Value of Device Handle.

Key : Keys to be authenticated(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:

The card will be locked after 8 wrong attempts.

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

Description

To change keys of the 4428 card

Parameters

icdev: Value of Device Handle

Key : Keys 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

Counts of reading errors

Parameters

icdev: Value of Device Handle

cntReadError : Counts 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

To read data from the 24C64 card

Parameters

icdev: Value of Device Handle

offset: The offset address

length: The length of data to be read

rdata: Data read out

Return Value

0 if successful; otherwise, Nonzero.

Example

```
// Read 20 -byte data starting from the 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

To write data to the 24C64 card

Parameters

icdev: Value of Device Handle

offset: The offset address

length: The length of data written

wdata: Data to be written

Return Value

0 if successful; otherwise, Nonzero.

Example

```
// Write 2 -byte data 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

To check whether a card inserted is a 24C64 card

Parameters

icdev: Value of Device Handle

Return Value

0 if successful—showing a 24C64 card is 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

To get the card number of the 125K card

Parameters

icdev: Value of Device Handle

_Msec: The beep interval of the reader (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:

- 1) 1K bytes of EEPROM, organized in 16 sectors with 4 blocks of 16 bytes each (one block consists of 16 byte)
- 2) User definable access conditions for each memory block
- 3) Individual set of two keys per sector (per application) to support multi-application with key hierarchy
- 4) Unique serial number(32bits) for each card
- 5) With anti-collision mechanism, multi-card operation accessible
- 6) Contactless transmission of data and supply energy (no battery needed), with antenna, and built-in encryption control logic and communication logic circuit
- 7) Operating temperature: -20 °C- 50 °C
- 8) Operating frequency: 13.56 MHz

9) Fast data transfer: 106 kbit/s

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

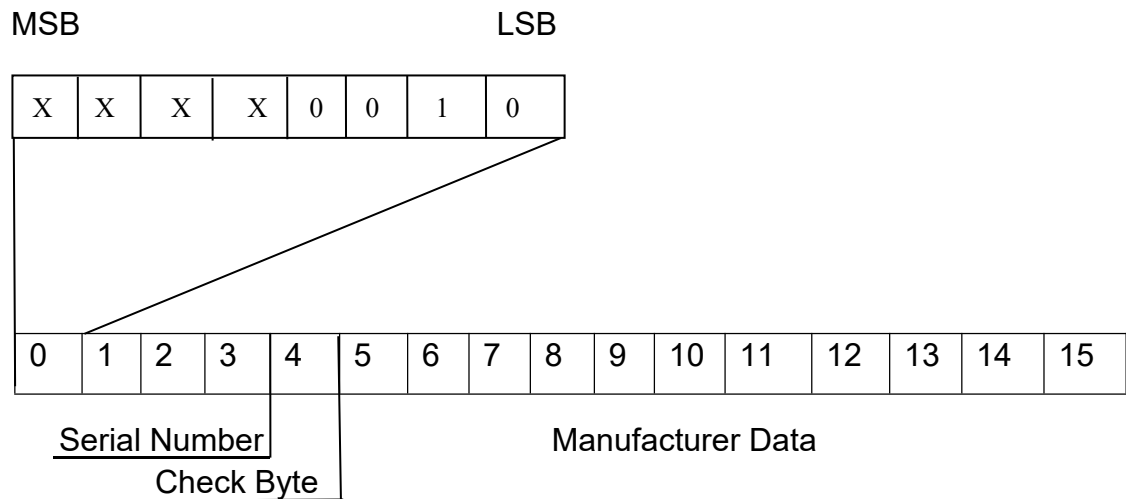
11) Data retention of 10 years, write endurance 100,000 cycles, unlimited read cycles

Storage structure

M1 card is organized in 16 sectors with 4 blocks (Blocks 0-3) of 16 bytes each, a total of 64 blocks (block number from 0 to 63). Block 0 of sector 0 (the absolute address 0 block) is used to store the IC manufacturer code, which is write protected after having been programmed by the manufacturer at production. Blocks 0 to 2 of other sectors are used for storing data while block 3 is the sector trailer for storing Key A, access bits and Key B. Its structure is as follows:

<u>A0 A1 A2 A3 A4 A5</u>						<u>FF 07 80 69</u>				<u>B0 B1 B2 B3 B4 B5</u>								
Key A(6 bytes)						access bits (4 bytes)				key B(6 bytes)								
Sector	Block	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Description	No.
0	0															Manufacturer Block	0	
	1															Data	1	
	2															Data	2	
	3	Key A					Access Bits			Key B						Sector Trailer 0	3	
1	0															Data	4	
	1															Data	5	
	2															Data	6	
	3	Key A					Access Bits			Key B						Sector Trailer 1	7	
	⋮																	
15	0															Data	60	
	1															Data	61	
	2															Data	62	
	3	Key A					Access Bits			Key B						Sector Trailer 15	63	

Manufacturer Block: The first block of the first sector is used by the IC card manufacturers. It contains the IC manufacturer data which is write protected after having been programmed by the card manufacture at production.



Data Blocks: All sectors contain 3 blocks for storing data (Sector 0 contains only two data blocks and the read-only manufacturer block).

Value Blocks: allow to perform electronic purse functions (valid commands: read, write, increment, decrement, restore, transfer). A value block can only be generated through a write operation in the value block format. Each block signifies a signed 4-byte value.

Sector Trailer (Block 3): Each sector has a sector trailer containing the secret Keys A and B (6 bytes each) and access bits (4 bytes).

Access Conditions

In each sector, secret keys and access bits are independent, and their own passwords and access control may be set based on actual needs. The access conditions for every data block and sector trailer are defined by 3 bits, which are stored non-inverted and inverted in the sector trailer of the specified sector.

Note: With each memory access the internal logic verifies the format of the access conditions. If it detects a format violation the whole sector is irreversible blocked.

Access Bits	Valid Commands	
C1 ₃ C2 ₃ C3 ₃	read, write	→
C1 ₂ C2 ₂ C3 ₂	read, write, increment, decrement, transfer, restore	→
C1 ₁ C2 ₁ C3 ₁	read, write, increment, decrement, transfer, restore	→
C1 ₀ C2 ₀ C3 ₀	read, write, increment, decrement, transfer, restore	→

Block	Description
3	sector trailer
2	data block
1	data block
0	data block

These 3 access bits control the rights of memory access using the secret keys A and B.(e.g. Decrement can be carried out after authentication of Key A, while for increment after authentication of Key B). The layout of the access bits in the access control byte is as follows (Byte 9 is a spare byte, default value is 0x69):

A0 A1 A2 A3 A4 A5								FF 07 80 69				B0 B1 B2 B3 B4 B5							
Key A								Access Bits				Key B							
								Bit 7 6 5 4 3 2 1 0											
Byte 6	C2 _{3_b}	C2 _{2_b}	C2 _{1_b}	C2 _{0_b}	C1 _{3_b}	C1 _{2_b}	C1 _{1_b}	C1 _{0_b}											
Byte 7	C1 ₃	C1 ₂	C1 ₁	C1 ₀	C3 _{3_b}	C3 _{2_b}	C3 _{1_b}	C3 _{0_b}											
Byte 8	C3 ₃	C3 ₂	C3 ₁	C3 ₀	C2 ₃	C2 ₂	C2 ₁	C2 ₀											
Byte 9																			

(Remark: **_b** means inverted value ,for example: if c11 is 1, c11_b is 0; c11 is 0, c11_b is 1)

1) **Sector trailer** (Block 3)

The access control of this block is different from the data blocks (Blocks 0,1,2). Its access control are as follows:

			Key A		Access bits		Key B	
C1 ₃	C2 ₃	C3 ₃	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

(Key A|B refers to key A or key B, Never means that data is not accessible in any circumstances.)

For example: Block 3, access bits is C1₃ C2₃ C3₃ = 100, it means:

Key A: not-readable, may be written after authentication of Key B
(transport configuration).

Access control: readable but not writable after authentication of Key A or Key B

Key B: unreadable, writable after authentication of Key B)

2) **Data blocks**

The access bits of data blocks (block 0, block 1, block 2) are specified as follows:

Access bits (X=0 or 1)			Access condition for (block 0,1,2)			
C1 _x	C2 _x	C3 _x	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

(Key A|B refers to key A or key B, Never means that data is not accessible in any circumstances.)

For example:

When access control bits of the block 0 are C1₀ C2₀ C3₀ = 100, Key A or Key B is readable after authentication; Key B is writable after authentication ; increment and decrement are not allowed.