

1. Common functions

(initialcom)(/*[in]*/int port,/*[in]*/long baud,/*[out,retval]*/int *hdev)

Function

Initial the communication port.

ID: FUNCIDS._fid_initialcom

Parameters

port: Serial port 1~20 when value 0~19. USB when value 100

baud: Baud rate (value: 9600~115200)

Return Value

Return 0 if success otherwise error code

Valid parameter returned, is handle value of reader (>0), negative value if failed.

Example

```
obj.initialcom (100,9600);//initial USB
obj.onResult(function(rData){
    if(rData.FunctionID == FUNCIDS._fid_initialcom)
    {
        var rel = rData.RePara_Int;
        if(0 == rel){ var hdev = parseInt(rData.RePara_Str);}
    }
});
```

(exit)(/*[in]*/short icdev,/*[out,retval]*/short *resul)

Function

Close the communication port.

ID: FUNCIDS._fid_exit

Parameters

icdev: the handle value of reader

Return Value

0 if success; Otherwise, NonZero

Example

```
obj.exit(icdev);
```

Remark

icdev must be released before next linking.

(hexTochar)(/*[in]*/BSTR hexstr,/*[in]*/short chlen,/*[out,retval]*/BSTR *charstr)

Function

Transform HEX string to corresponding ASC bytes array.

ID: FUNCIDS._fid_strToHex

Parameters

hexstr: the string of HEX
 chlen: the length of chars
 charstr: the bytes array

Return Value

The bytes array.

Example

```
var str="30313233";
var hex="";
obj.hexTochar(str,4);
obj.onResult(function(rData){
    if(rData.FunctionID == FUNCIDS. _fid_strToHex)
    {
        var rel = rData.RePara_Int;
        if(0 == rel){
            hex = rData.RePara_Str;
            if(hex!="0123")
                alert("error");
        }
    }
});
```

(charToHex)(/*[in]*/BSTR charstr,/*[in]*/short hexlen,/*[out,retval]*/BSTR *hexstr)

Function

Transform ASC bytes array to corresponding HEX string.

ID: FUNCIDS. _fid_hexToStr

Parameters

charstr: the bytes array.
 hexlen: the length of HEX string.
 hexstr: the string of HEX.

Return Value

The string of HEX.

Example

```
var chstr="abcd";
var str="";
obj.charToHex(chstr,8);
obj.onResult(function(rData){
    if(rData.FunctionID == FUNCIDS. _fid_hexToStr)
    {
        str = rData.RePara_Str;
        if(str!="61626364")
            alert("error");
    }
});
```

```
    }
});
```

(GetDevSN)(/*[in]*/ short icdev, /*[out,retval]*/ BSTR* devSN);

Function

Get the device's serial number
ID: FUNCIDS._fid_GetDevSN

Parameters

icdev: the handle value of reader

Return Value

Device serial number as string

Example

```
obj.GetDevSN(icdev);
obj.onResult(function(rData){
    if(rData.FunctionID == FUNCIDS._fid_GetDevSN)
    {
        var rel = rData.RePara_Int;
        if(0 == rel){
            var devSN= rData.RePara_Str;
        }
    }
});
```

2. Device Functions

(beep)(/*[in]*/short icdev,/*[in]*/short isec,/*[out,retval]*/short *resul)

Function

Make reader to beep.
ID: FUNCIDS._fid_beep

Parameters

icdev: the handle value of reader
isec: time of beep.

Return Value

0 if success; Otherwise, Nonzero.

Example

```
obj.beep(icdev,10);           //beep 10ms
```

(srd_eeprom)(/*[in]*/short icdev, /*[in]*/ int offset, /*[in]*/ int length, /*[out,retval]*/BSTR* pData);

Function

Read the EEPROM data of reader
ID: FUNCIDS._fid_srd_eeprom

Parameters

icdev: the handle value of reader

offset: the start address to read

length: the length to read

Return Value

The data of EEPROM as string

Example

```
var data;
var offset = 0;
var len = 5;
obj.srd_eeprom(icdev, offset, len);
obj.onResult(function(rData){
    if(rData.FunctionID == FUNCIDS._fid_srd_eeprom)
    {
        var rel = rData.RePara_Int;
        if(0 == rel){
            var data = rData.RePara_Str;
        }
    }
});
```

(swr_eeprom)(/*[in]*/short icdev, /*[in]*/ int offset, /*[in]*/ int length,
/*[in]*/BSTR strData, /*[out,retval]*/short* result);

Function

Write data to EEPROM of reader

ID: FUNCIDS._fid_swr_eeprom

Parameters

icdev: the handle value of reader

offset: the start address to read

length: the length to read

strData: the data ready write to EEPROM as string

Return Value

0 if success; Otherwise, Nonzero

Example

```
var data= "ATest";
var offset = 0;
var len = 5;
var st;
obj.swr_eeprom(icdev, offset, len, data);
obj.onResult(function(rData){
    if(rData.FunctionID == FUNCIDS._fid_swr_eeprom)
    {
        var rel = rData.RePara_Int;
        if(0 == rel){
            alert("write eeprom ok");
        }
    }
});
```

```
    }
  }
});
```

(lcd_setBright)(/*[in]*/short icdev, /*[in]*/ short fBright, /*[out,retval]*/short* result);

Function

Set the bright state of LCD

ID: FUNCIDS. _fid_setBright

Parameters

icdev: the handle value of reader

fBright: values 0 to make LCD back light off, otherwise 1 to make on

Return Value

0 if success; Otherwise, Nonzero.

Example

```
obj.setBright(icdev, 0);           //make back light off
```

(lcd_dispstr_ex) (/*[in]*/short icdev, /*[in]*/ BSTR str, /*[in]*/ int line,/*[in]*/ int offset, /*[in]*/ int str_len, /*[in]*/ int flag,/*[out,retval]*/ short *result);

Function

Display string on the LCD

ID: FUNCIDS. _fid_dispstr_ex

Parameters

icdev: the handle value of reader

str: the string for display

line: line number the string show on

offset: start address in the line

str_len: the size of string,when flag values 0, size equals the char number of string, when flag values 1, size equals twice of char number

flag: 0 if show as ANSI string, or 1 for show as GB2312 string

Return Value

0 if success; Otherwise, Nonzero.

Example

```
var st;
var str = "123456";
obj lcd_dispstr_ex(icdev,str, 1, 0, str.length, 0);
```

(lcd_dispclear) (/*[in]*/ short icdev, /*[out, retval]*/ short* result);

Function

Clear string on the LCD

ID: FUNCIDS. _fid_dispClear

Parameters

icdev: the handle value of reader

Return Value

0 if success; Otherwise, Nonzero.

Example

```
obj.lcd_dispclear(icdev);
```

(passInReady) (/*[in]*/short icdev,/*[in]*/short waitTime,/*[out,retval]*/short* result);

Function

Set the device to state of ready for inputing

ID: FUNCIDS. _fid_passInReady

Parameters

icdev: the handle value of reader

waitTime: the time for inputing as second,device input will canceled if out of the time

Return Value

0 if success; Otherwise, Nonzero.

Example

```
obj.passInReady(icdev, 10);
```

(getPassPressed) (/*[in]*/short icdev, /*[out, retval]*/ BSTR* strPass);

Function

Check the value user typed in

ID: FUNCIDS. _fid_getPassPressed

Parameters

icdev: the handle value of reader

Return Value

String user typed in

Example

```
var data;
obj.getPassPressed(icdev);
obj.onResult(function(rData){
    if(rData.FunctionID == FUNCIDS. _fid_getPassPressed)
    {
        var rel = rData.RePara_Int;
        if(0 == rel){
            data = rData.RePara_Str;
        }
    }
});
```

(getPassConfirmed) (/*[in]*/short icdev, /*[out,retval]*/ short* result);

Function

Get the input password after typed Enter

ID: FUNCIDS. _fid_getPassConfirmed

Parameters

icdev: the handle value of reader

Return Value

00h if user typed the Enter button
 A1h input overflow, 25 chars max
 A2h user typed the Cancel button
 A3h the device is not in inputing state
 A4h in the state of inputing
 A5h input time out

Example

```
var st;
obj.getPassConfirmed(icdev);
obj.onResult(function(rData){
    if(rData.FunctionID == FUNCIDS. _fid_getPassConfirmed)
    {
        st = rData.RePara_Int;
        switch(st)
        {
            case 0:alert("You typed the enter button");break;
            case 0xA1:alert("Password too long");break;
            case 0xA2:alert("You cancel the input");break;
            case 0xA3:alert("Keyboard is not in input mode");break;
            case 0xA4:alert("Input running");break;
            case 0xA5:alert("Input timeout");break;
        }
    }
});
```

(passInEnd) (/*[in]*/short hdev, /*[out,retval]*/ short* result);

Function

End the input, type in will invalid after this called
 ID: FUNCIDS. _fid_passInEnd

Parameters

icdev: the handle value of reader

Return Value

0 if success; Otherwise, Nonzero.

Example

```
obj.passInEnd(icdev);
```

3. Class Mifare Card Functions

(reset)(/*[in]*/short icdev, /*[in]*/short Msec, /*[out,retval]*/short *result)

Function

Reset the MCM (MIFARE read/write device core mode).

ID: FUNCIDS. _fid_resetRF

Parameters

icdev: the handle value of reader

Msec: the time to reset. Unit: 10ms

Return Value

0 if success; Otherwise, Nonzero.

Example

```
obj.reset(icdev,10);//RF reset100ms
```

(findcard)(/*[in]*/int hdev,/*[in]*/short mode,/*[out,retval]*/long *ncard)

Function

Find card,can return the card serial number in working area. (Contain the next functions: request, anticoll, select).

ID: FUNCIDS. _fid_findCard

Parameters

hdev: the handle value of reader

mode: the mode of find card (refer to appendix)

ncard: card serial number returned

Return Value

If success, returned the card serial number.

Example

```
var ncard=0;
obj.findcard(icdev,0);//mode 0 authentication
obj.onResult(function(rData){
    if(rData.FunctionID == FUNCIDS. _fid_findCard)
    {
        var rel = rData.RePara_Int;
        if(0 == rel){
            ncard = rData.RePara_Str;
        }
    }
});
```

Remark

1. In IDLE mode, after read-write operation, we use function halt() to end the card operation, only when the card out and re-enter the operating area, the reader can operate it once again.
2. Related HEX function is findcardHex, recommend use findcardHex.

(findcardHex)(/*[in]*/int hdev,/*[in]*/short mode,/*[out,retval]*/BSTR *strcard)

Function

Find card,can return the card serial number in working area. (HEX string type).

ID: FUNCIDS. _fid_findCardHex

Parameters

hdev: the handle value of reader
 mode: the mode of find card (refer to appendix)
 strcard: card serial number returned

Return Value

If success, return the card serial number.

Example

```
var ncard=0;
obj.findcardHex(icdev,0);//mode 0 authentication
obj.onResult(function(rData){
    if(rData.FunctionID == FUNCIDS._fid_findCardHex)
    {
        var rel = rData.RePara_Int;
        if(0 == rel){
            ncard = rData.RePara_Str;
        }
    }
});
```

Remark

In IDLE mode, after read-write operation, we use function halt() to end the card operation, only when the card out and re-enter the operating area, the reader can operate it once again.

(loadkey)(/*[in]*/short hdev,/*[in]*/short mode,/*[in]*/ short sNr,/*[in]*/BSTR key,/*[out,retval]*/short *resul)

Function

Load keys to RAM of Reader.
 ID: FUNCIDS._fid_loadKey

Parameters

hdev: the handle value of reader
 mode: the mode of key verify.
 Value as following:

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

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

sNr: the sector number.

key: sector key write to the reader

Return Value

0 if success; Otherwise, Nonzero

Example

```
/* load sector 0 key with set 0 and keyA*/
obj.loadkey(icdev,0,0,"FFFFFFFFFFFF");
```

(authentication)(/*[in]*/ short hdev,/*[in]*/short mode,/*[in]*/short snr,/*[out,retval]*/short *resul);

Function

Verify key.

ID: FUNCIDS._fid_authenClass

Parameters

hdev: the handle value of reader

mode: the mode of key verify.

Value as following:

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

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

snr: the sector number.

key: sector key write to the reader

Return Value

0 if success; Otherwise, NonZero

Example

```
//authentication sector 4 with mode 0
var st;
obj.authentication(icdev,0,4);
obj.onResult(function(rData){
    if(rData.FunctionID == FUNCIDS._fid_authenClass)
    {
        st = rData.RePara_Int;
        if(st != 0){
            alert("Authentication error!");
        }
    }
});
```

(read)(/*[in]*/short hdev,/*[in]*/short blk,/*[out,retval]*/BSTR *blkdata);

Function

Read content of card, for M1 card ,read one block data (16 bytes) once.

ID: FUNCIDS._fid_readAsStr

Parameters

hdev: the handle value of reader

blk: Address of block (M1- 0~63, MS70 – 0~255)

blkdata: [out] data of card.

Return Value

The data of block.

Example

```
var data="";
var st;
obj.read(icdev,4); //read data of block 4
obj.onResult(function(rData){
    if(rData.FunctionID == FUNCIDS._fid_readAsStr)
    {
        st = rData.RePara_Int;
        if(st == 0){
            data = rData.RePara_Str;
        }
    }
});
```

(write)(/*[in]*/short hdev,/*[in]*/short blk,/*[in]*/BSTR wtdata,/*[out,retval]*/short *resul)

Function

Write data to card (HEX string). One block once (16 bytes).

ID: FUNCIDS._fid_writeAsStr

Parameters

hdev: the handle value of reader

blk: Address of block (M1- 1~63, MS70 – 1~255)

wtdata: Data write to the card.

Return Value

0 if success; Otherwise, Nonzero

Example

```
var st;
obj.write(icdev,2,"11223344556677889900AABBCCDDEEFF");//write block 2
```

(directRead)(/*[in]*/short hdev,/*[in]*/short blk,/*[out,retval]*/BSTR *blkdata)

Function

Read data from card directly.

ID: FUNCIDS. _fid_readAsHex

Parameters

hdev: the handle value of reader

blk: Address of block (M1- 0~63, MS70 – 0~255)

blkdata: Data read from card block.

Return Value

Data of card.

Example

```
var data="";
var st;
obj.directRead(icdev,4); //read block 4
obj.onResult(function(rData){
    if(rData.FunctionID == FUNCIDS._fid_readAsStr)
    {
        st = rData.RePara_Int;
        if(st == 0){
            data = rData.RePara_Str;
        }
    }
});
```

**(directWrite)(/*[in]*/short hdev,/*[in]*/short blk,/*[in]*/BSTR
wtdata,/*[out,retval]*/short *resul)**

Function

Write data to card directly (as ASC bytes).

ID: FUNCIDS. _fid_writeAsHex

Parameters

hdev: the handle value of reader

blk: Address of block (M1- 1~63, MS70 – 1~255)

wtdata: Data write to card block.

Return Value

0 if success; Otherwise, Nonzero.

Example

```
var st;
obj.directWrite(icdev,5,"1376666888"); /*write block 5*/
```

(halt)(/*[in]*/short hdev, /*[out,retval]*/short *resul)

Function

Abort operation of card.

ID: FUNCIDS. _fid_halt

Parameters

hdev: the handle value of reader

Return Value

0 if success; Otherwise, Nonzero.

Example

```
var st;
obj.halt(icdev);
```

Remark

When using findcard(), there is a parameter mode, card been set to HALT status when it is set value as 0, the card should be take out off operating area and put it again.

(changkey)(/*[in]*/short hdev,/*[in]*/ short secNr,/*[in]*/ BSTR keya,/*[in]*/ BSTR ctrlword,/*[in]*/short bk,/*[in]*/ BSTR keyb,/*[out,retval]*/

Function

Update key of special sector.
ID: FUNCIDS._fid_changeKey

Parameters

hdev: the handle value of reader.
secNr: sector number.
keya: keyA to update.
ctrlword: control word to update.
bk: reserved, set to 0.
keyb: keyB to update.

Return Value

0 if success; Otherwise, Nonzero.

Example

```
//update key of sector 3
var st;
obj.changkey(icdev,3,"FFFFFFFFFFFF","FF078069",0,"FFFFFFFFFFFF");
```

Remark

The parameters keya, ctrlword, and keyb, type as HEX string.

(initialval)(/*[in]*/short hdev, /*[in]*/short blk, /*[in]*/long value, /*[out,retval]*/short *resul)

Function

Initial the value of special block.
ID: FUNCIDS._fid_initVal

Parameters

hdev: the handle value of reader
blk: address of block
value: the value to initial block

Return Value

0 if success; Otherwise, Nonzero.

Example

```
var st;
var value=1000;
obj.initialval(icdev,1,value);           //initial the block 1 to 1000
```

Remark

Should call initial value before increment decrement value.

(increment)(/*[in]*/short hdev,/*[in]*/ short blk, /*[in]*/long value,/*[out,retval]*/short *resul);

Function

Increment value of special block.

ID: FUNCIDS._fid_increment

Parameters

hdev: the handle value of reader

blk: address of block

value: the value to increment.

Return Value

0 if success; Otherwise, Nonzero.

Example

```
var st;
var value=10;
obj.increment(icdev,1,value); //increment 10 to block 1
```

(readval)(/*[in]*/short hdev,/*[in]*/ short blk, /*[out,retval]*/long *value)

Function

Read value of special block.

ID: FUNCIDS._fid_readVal

Parameters

hdev: the handle value of reader

blk: address of block

value: the value to read.

Return Value

The value of special block.

Example

```
var st;
var value;
value=obj.readval(icdev,1);           //read value of block 1
obj.onResult(function(rData){
    if(rData.FunctionID == FUNCIDS._fid_readVal)
    {
        st = rData.RePara_Int;
        if(st == 0){
            value = rData.RePara_Str;
        }
    }
})
```

```
});
```

```
(decrement)(/*[in]*/short hdev,/*[in]*/ short blk,/*[in]*/ long  
value,/*[out,retval]*/ short *resul)
```

Function

Decrement value of special block.

ID: FUNCIDS._fid_decrement

Parameters

hdev: the handle value of reader

blk: address of block

value: the value to decrement.

Return Value

0 if success; Otherwise, Nonzero.

Example

```
var st;  
var value=10;  
obj.decrement(icdev,1,value);    //decrement the value of block 1
```

Remark

The parameters keya, ctrlword, and keyb, type as HEX string.

```
(transfer)(/*[in]*/short hdev,/*[in]*/short blk,/*[out,retval]*/short *resul)
```

Function

Transfer data in register to EEPROM.

ID: FUNCIDS._fid_transfer

Parameters

hdev: the handle value of reader

blk: address of block

Return Value

0 if success; Otherwise, Nonzero.

Example

```
var st;  
obj.transfer(icdev,2);
```

Remark

This function should be called just after increment or decrement.

4. Contactless-CPU functions

```
(fcpureset)(/*[in]*/short icdev,/*[out]*/short *rlen,/*[out,retval]*/BSTR* rbuf)
```

Function

Power on and reset of CPU card.

ID: FUNCIDS._fid_fcPuReset

Parameters

icdev: the handle value of reader
 rlen: the length of returned information.
 rbuf: the information returned.

Return Value

Information returned.

Example

```
var data="" ";
Var rlen;
obj.fcpureset(icdev,rlen);
obj.onResult(function(rData){
    if(rData.FunctionID == FUNCIDS._fid_fcpuReset)
    {
        st = rData.RePara_Int;
        if(st == 0){
            data = rData.RePara_Str;
        }
    }
});
```

Remark

This function should be called just after increment or decrement.

(fcpuCommandlink)(/*[in]*/short icdev,/*[in]*/short slen,/*[in]*/BSTR sdata,/*[in]*/short tt,/*[in]*/short FG,/*[out,retval]*/BSTR * rdata)

Function

APDU data exchange function.

ID: FUNCIDS. _fid_fcpuAPDU

Parameters

icdev: the handle value of reader
 slen: the length of information to send.
 sdata: command message to send.
 tt: delay time. unit: 10 ms
 FG: split length, recommend small than 64.
 rbuf: the information returned.

Return Value

Information returned.

Example

```
var cmd= "00A404000752736120417070";//12 bytes (HEX string)
Var data="" ";
obj.fcpuCommandlink(icdev,12,cmd,7,60);//delay 70ms,60 bytes once send.
obj.onResult(function(rData){
    if(rData.FunctionID == _fid_fcpuAPDU)
    {
        st = rData.RePara_Int;
```



```
        if(st == 0){  
            data = rData.RePara_Str;  
        }  
    });
```

Appendix

Mode of find card

There are three cases: IDLE mode, ALL mode, and special card mode.

0— means IDLE mode, one card once operate.

1— means ALL mode, multi card once operate.