RFID READER

Compact Flash Reader

SL801

User Manual

Version 1.4
Nov 2011
StrongLink

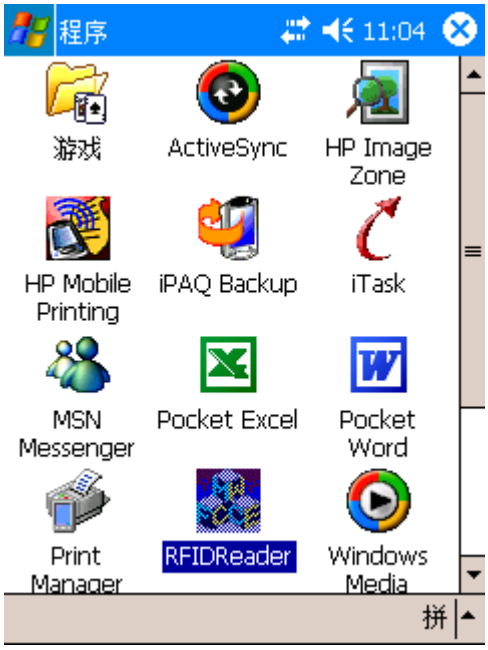
CONTENS

1. GENERAL INFORMATION

- Compact Flash Type II Interface
- DC3.0V - DC3.6V VDD Operating
- Pocket PC 2003/2005 Operating Systems Compatibility
- 13.56MHz RF Operating Frequency
- ISO14443A, ISO1443B & ISO15693 Protocols
- 120MA Working Current
- 5MA Idle Current
- Operating Temperature Range: -20°C ~ +50°C
- Storage Temperature Tange: -25°C ~ +60°C
- Dimension: 83 × 44 × 13 mm
- Weight: 20g

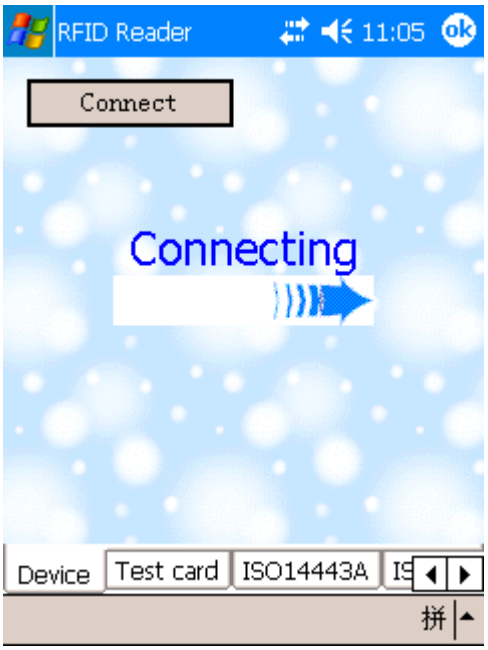
2. DEMO SOFTWARE

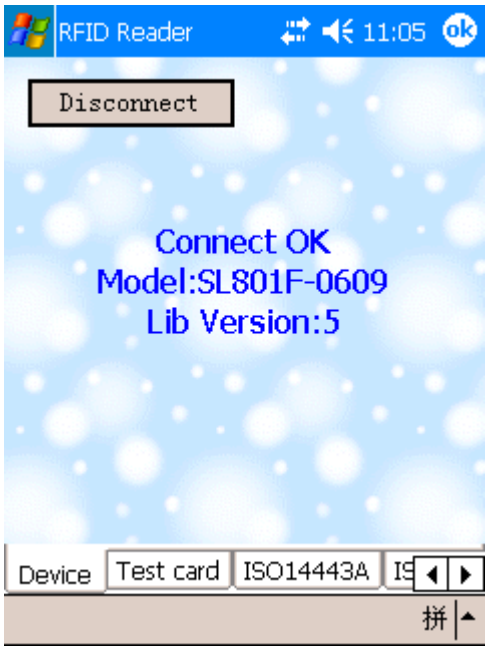
DEMO software is applicable for Pocket PC 2003/2005 operating system. SL801_Setup.exe will install the DEMO software and DLL onto PDA, and then become a logo as follow



2.1 ONLINE

Click [Connect] button on the windows of “Device”, the software will search for the port and show corresponding module information after successful connecting.





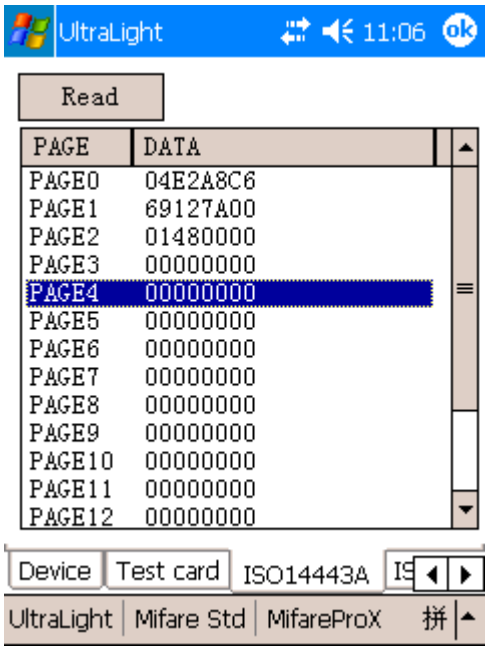
2.2 CARDS TESTING

Click [Start] button on the windows of “Test Card”, you can test cards types. At this time, when all the standard card come to the antenna area, the indicator light on the SL801 will shine, and show corresponding types of the card.

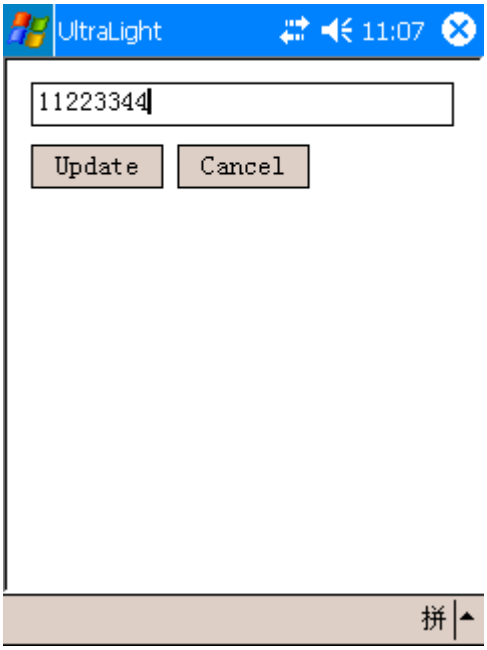
2.3 ISO14443A (Mifare class)

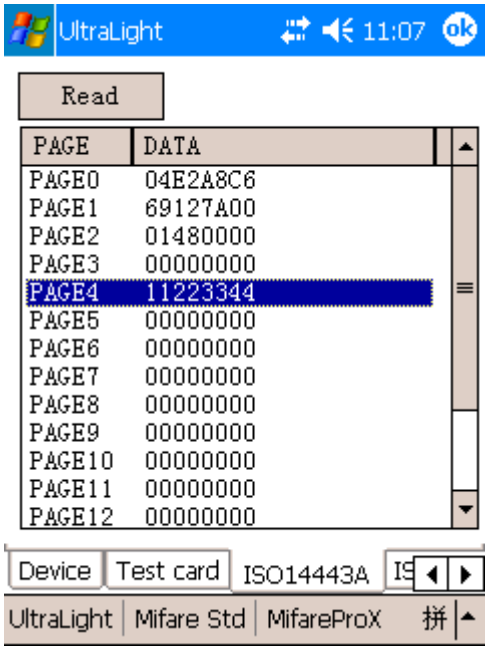
2.3.1 Mifare_Ultralight

Click the [Read] button to read data from UltraLight and display.



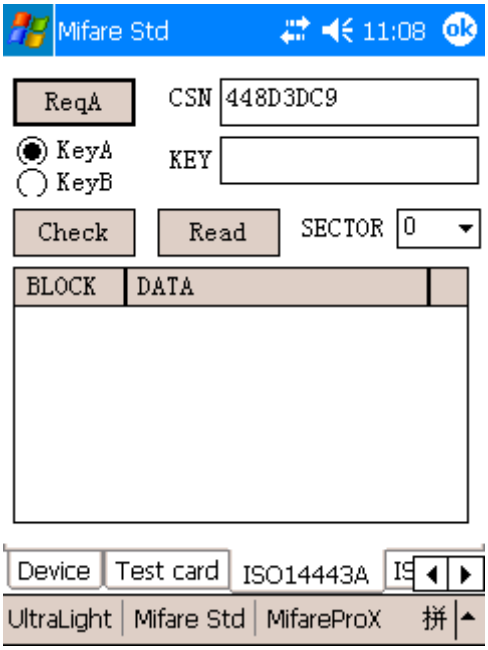
Double-clicking certain data module will pop-up a window. After input new data, click [Update] button to store the new data into corresponding address.



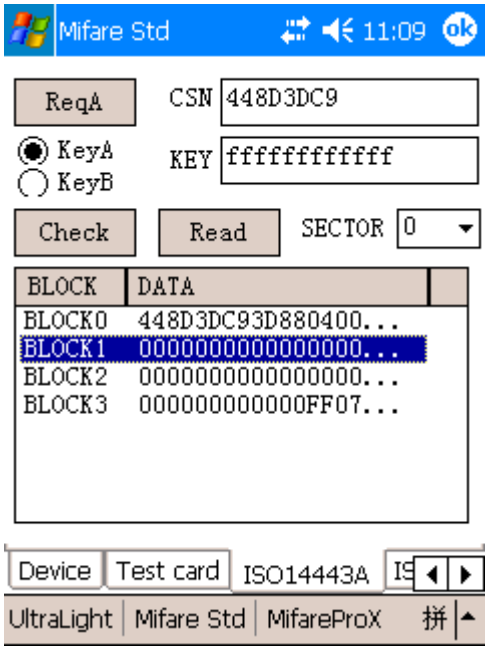


2.3.2 Mifare_Std

Click [ReqA] button to obtain the **C**hip unique **S**erial **N**umber.

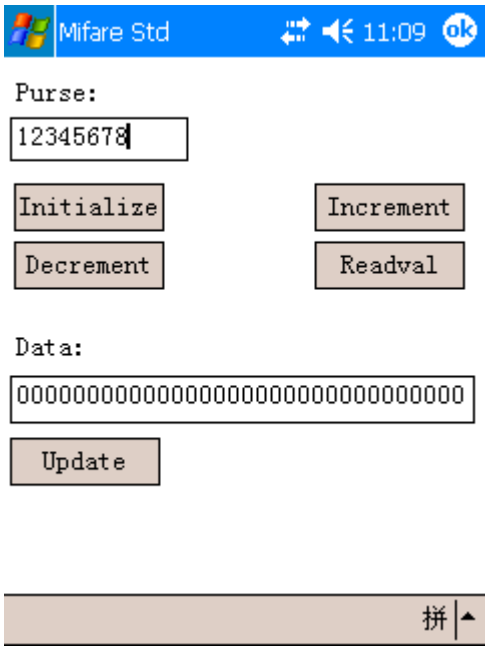


Input the correct 6 bytes password, click [Check] button to verify this password, then can read/write the tag.



Double-click certain data module will pop-up a window to rewrite data and operate the purse

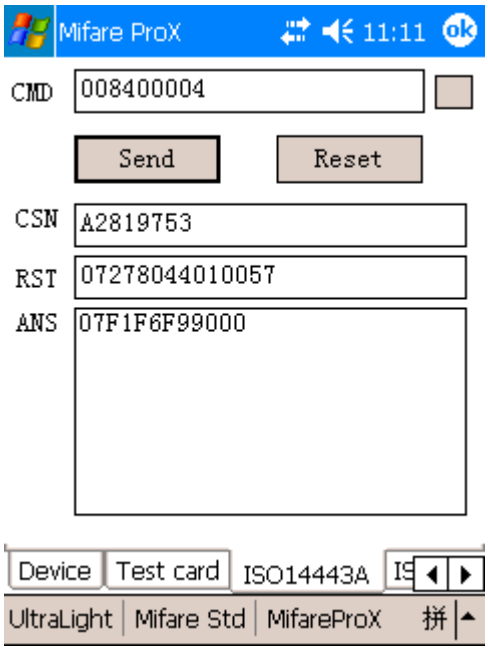
Notice: purse show at BCD, other data shows at HEX



2.3.3 Mifare_ProX

Click [Reset] button to obtain the serial number and the reposition information.

Input COS command to CMD, click [Send] button to obtain the response data from tag.

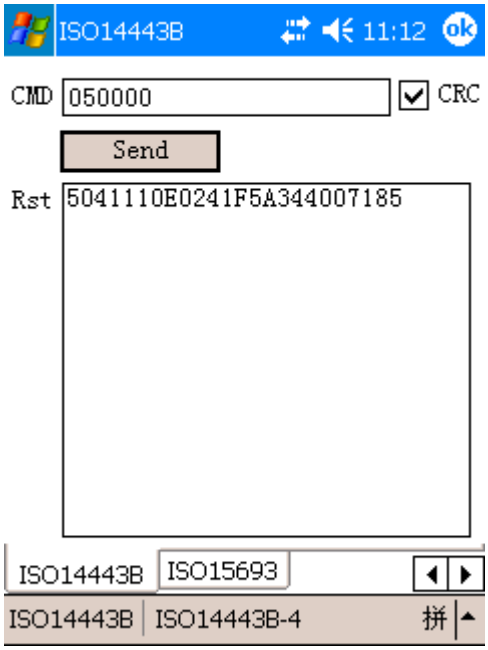


2.4 ISO14443B

There are two windows under the ISO14443B, which are ISO14443B and ISO14443B-4. The difference is on the window of ISO14443B, SL801 will transmit and receive all data, base function “int WINAPI rf_transceive”, while on the window of ISO14443B-4, SL801 auto manage ISO14443-4 **P**rotocol **C**ontrol **B**yte, the transferring base function “int WINAPI rf_cos_command”.

2.4.1 ISO14443B

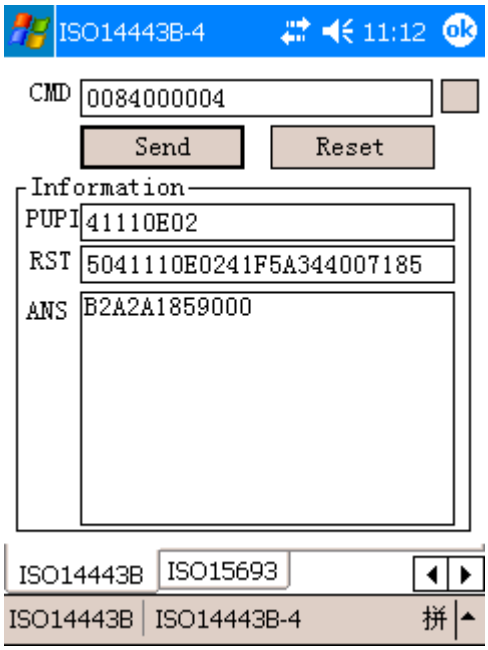
Example: Send REQB command to card on the window of the ISO14443B as follow



2.4.2 ISO14443B-4

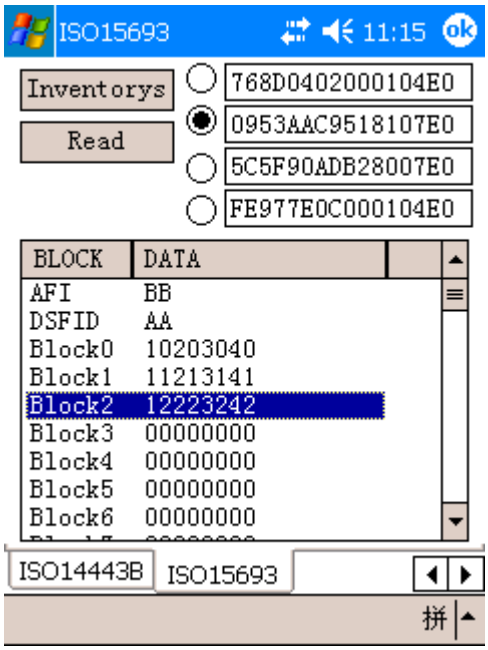
Click [Reset] button to obtain RATS information.

Input COS command to CMD, and click [Send] to obtain the response data from the card.

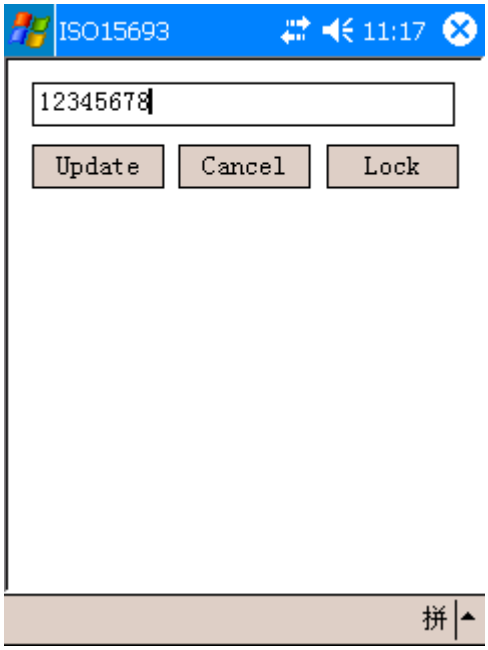


2.5 ISO15693

Click [Inventorys] button to obtain 4 sheets of UID of ISO15693 card at most, choose certain UID, click [Read] button can read the data from the card.



Double-click data module to rewrite and lock the data to corresponding block in the pop-up window.



3. DLL INFORMATION

3.1 SYSTEM FUNCTION

3.1.1 Get DLL Version

Prototype: int WINAPI lib_ver(unsigned int *pVer)
Parameter: pVer: [OUT] DLL version number
Return: return 0 on success

3.1.2 Connect

Prototype: int WINAPI rf_init_com (int port)
Parameter: port: [IN] serial port number
Return: return 0 on success

3.1.3 Disconnect

Prototype: int WINAPI rf_ClosePort(void)
Return: return 0 on success

3.1.4 Get Device Type

Prototype: int WINAPI rf_get_model (unsigned char *pVersion,
unsigned char *pLen)
Parameter: pVersion: [OUT] response information
pLen: [OUT] the length of response information
Return: return 0 on success

3.1.5 Manage Protocol

Prototype: int WINAPI rf_init_type(unsigned char type)
Parameter: type: [IN] woking mode of the reader
Return: return 0 on success
Explanation: type = 'A': set SL801 into ISO14443A mode
type = 'B': set SL801 into ISO14443B mode
type = '1': set SL801 into ISO15693 mode

3.1.6 Manage RF Transmittal

Prototype: int WINAPI rf_antenna_sta(unsigned char model)
Parameter: model: [IN] RF transmittal state
Return: return 0 on success
Explanation: model = 0: turn off RF transmittal
model = 1: turn on RF transmittal

3.1.7 Manage LED

Prototype: int WINAPI rf_light(unsigned char color)
Parameter: color: [IN] 0 = turn off, others = light on

3.2 DES FUNCTION

3.2.1 DES_Encrypt

Prototype: int WINAPI des_encrypt(unsigned char *pSzOut,
unsigned char *pSzIn,
unsigned int inlen,
unsigned char *pKey,
unsigned int keylen)

Parameter:			
	pSzOut:	[OUT]	ciphertext, bytes length equal to plaintext
	pSzIn:	[IN]	plaintext
	inlen:	[IN]	length of plaintext, integer times of 8 bytes
	pKey:	[IN]	encrypt key
	keylen:	[IN]	length of key, 8 bytes for single DES, 16 bytes for triple DES

Return: return 0 on success

3.2.2 DES_Decrypt

Prototype: int WINAPI des_decrypt(unsigned char *pSzOut,
unsigned char *pSzIn,
unsigned int inlen,
unsigned char *pKey,
unsigned int keylen)

Parameter:			
	pSzOut:	[OUT]	plaintext, bytes length equal to ciphertext
	pSzIn:	[IN]	ciphertext
	inlen:	[IN]	length of ciphertext, integer times of 8 bytes
	pKey:	[IN]	encrypt key
	keylen:	[IN]	length of key, 8 bytes for single DES, 16 bytes for triple DES

Return: return 0 on success

3.3 ISO14443A FUNCTION

3.3.1 REQA

Prototype: int WINAPI rf_request(unsigned char model, unsigned short *pTagType)
Parameter: model: [IN] Req mode
pTagType: [OUT] response data from tag, chip type code
Return: return 0 on success
Explanation: mode = 0x26: REQ_STD
mode = 0x52: REQ_ALL
pTagType: 0x4400 = Ultra_Light
0x0400 = Mifare_1K
0x0200 = Mifare_4K
0x4403 = Mifare_DESFire
0x0800 = Mifare_Pro
0x0403 = Mifare_ProX
To read the ISO14443A card in influence district, you should transfer the base function hereunder in turn

Mifare_Std:
1. int WINAPI rf_request
2. int WINAPI rf_anticoll
3. int WINAPI rf_select
Card will be on actived state thereafter

UltraLight:
1. int WINAPI rf_request
2. int WINAPI int rf_ul_select
This function aggregate anticollision and select command, Card will be on actived state thereafter

Mifare_DESFire:
1. int WINAPI rf_DESFire_rst
This function aggregate ReqA, anticollision, select command, Card will be on actived state thereafter

MifareProX:
1. int WINAPI rf_typea_rst
This function aggregate searching for card, anticollision ,activation and reposition order, Card will be on actived state thereafter

3.3.2 Select UltraLight

Prototype: int WINAPI int rf_ul_select (unsigned char *pSnr, unsigned char *pLen)
Parameter: pSnr: [OUT] response data from tag, unique serial number
pLen: [OUT] length of response data
Return: return 0 on success

3.3.3 Mifare_Std Anticollision

Prototype: int WINAPI rf_anticoll(unsigned char bcnt,
unsigned char *pSnr,
unsigned char *pLen)
Parameter: bcnt: [IN] must be 4
pSnr: [OUT] response data from tag, unique serial number
pLen: [OUT] length of response data
Return: return 0 on success

3.3.4 Mifare_Std Selectting

Prototype: int WINAPI rf_select(unsigned char *pSnr,
unsigned char snrLen,
unsigned char *pSize)
Parameter: pSnr: [IN] unique serial number of tag
snrLen: [IN] length of pSnr
pSize: [OUT] response data, capacity code
Return: return 0 on success

3.3.5 Mifare_Std Authentify

Prototype: int WINAPI rf_M1_authentication2(unsigned char model,
unsigned char block,
unsigned char *pKey)
Parameter: model: [IN] key validate mode
block: [IN] block absolute address
pKey: [IN] 6 bytes password
Return: return 0 on success
Explanation: model = 0x60: via KeyA
model = 0x61: vua KeyB

3.3.6 Mifare_Std (UltraLight) Read

Prototype: int WINAPI rf_M1_read (unsigned char block,
unsigned char *pData,
unsigned char *pLen)
Parameter: block: [IN] block absolute address
pData: [OUT] response data from tag
pLen: [OUT] length of response data
Return: return 0 if successful

Explanation: this function is applicable for Ultra_Light card. Every page of Ultra_Light card has 4bytes. Transfering this function every time, return data of 4 consecutive pages..

3.3.7 Mifare_Std Write

Prototype: int WINAPI rf_M1_write (unsigned char block, unsigned char *pData)
Parameter: block: [IN] block absolute address
pData: [IN] written data, 16 bytes
Return: return 0 on success

3.3.8 UltraLight Write

Prototype: int WINAPI int rf_ul_write (unsigned char page, unsigned char *pData)
Parameter: page: [IN] page address (0 – F)
pData: [IN] written data, 4 bytes
Return: return 0 on success

3.3.9 Mifare_Std Initialize Value

Prototype: int WINAPI rf_M1_initval(unsigned char block, long value)
Parameter: block: [IN] block absolute address
pValue: [IN] initialize purse at HEX format, low byte in former
Return: return 0 on success

3.3.10 Mifare_Std Read Value

Prototype: int WINAPI rf_M1_readval(unsigned char block, long *pValue)
Parameter: block: [IN] block absolute address
pValue: [OUT] return value at HEX format low byte in former
Return: return 0 on success

3.3.11 Mifare_Std Increment

Prototype: int WINAPI rf_M1_increment (unsigned char block, long value)
Parameter: block: [IN] block absolute address
value: [IN] increase value at HEX format, low byte in former
Return: return 0 on success

3.3.12 Mifare_Std Decrement

Prototype: int WINAPI rf_M1_decrement (unsigned char block, long value)
Parameter: block: [IN] block absolute address
value: [IN] deduct value at HEX format, low byte in former
Return: return 0 on success

3.3.13 Mifare_Std Restore

Prototype: int WINAPI rf_M1_restore (unsigned char block)
Parameter: block: [IN] block absolute address
Return: return 0 on success

3.3.14 Mifare_Std Transfer

Prototype: int WINAPI rf_M1_transfer (unsigned char block)
Parameter: block: [IN] block absolute address
Return: return 0 on success
Explanation: this function only use after increment, decrement and restore order

3.3.15 Mifare_Std Halt

Prototype: int WINAPI rf_halt( )
Return: return 0 on success

3.3.16 DESFire Reset

Prototype: int WINAPI rf_DESFire_rst(unsigned char model,
unsigned char *pData,
unsigned char *pMsgLg)
Parameter: model: [IN] ReqA mode
pDate: [OUT] response data from tag
pMsgLg:[OUT] length of response data
Return: return 0 on success
Explanation: mode = 0x26: REQ_STD
mode = 0x52: REQ_ALL
pDate = 7 bytes Chip Serial Number
+ n bytes RATS according to ISO14443-4 protocol

3.3.17 Mifare_ProX Reset

Prototype: int WINAPI rf_typea_rst(unsigned char model,
unsigned char *pData,
unsigned char *pMsgLg)
Parameter: model: [IN] ReqA mode
pDate: [OUT] response data from tag
pMsgLg:[OUT] length of response data
Return: return 0 on success
Explanation: mode = 0x26: REQ_STD
mode = 0x52: REQ_ALL
pDate = 7 bytes Chip Serial Number
+ n bytes ATS according to ISO14443-4 protocl

3.3.18 T=CL Protocol Transceive

Prototype: int WINAPI rf_cos_command(unsigned char *pCommand,
unsigned char cmdLen,
unsigned char *pData,
unsigned char *pMsgLg)

Parameter: pCommand: [IN] cos command
cmdLen: [IN] length of command
pDate: [OUT] response data from tag, including SW1 & SW2
pMsgLg: [OUT] length of response data

Return: return 0 on success

3.3.19 T=CL Protocol Deselect

Prototype: int WINAPI CL_Deselect( )

Return: return 0 on success

3.4 ISO14443B FUNCTION

3.4.1 ISO14443B Transceive

Prototype: int WINAPI rf_transceive(unsigned char crcon,
unsigned char *pTxData,
unsigned char sendLen,
unsigned char *pRxDate,
unsigned char *pMsgLg)

Parameter: crcon: [IN] not 0 means sending to crc, and calculated by reader,
Equal to 0 means not sending to CRC
pTxData: [IN] sending data
sendLen: [IN] length of sending data
pRxDate: [OUT] response data from tag
pMsgLg: [OUT] length of response data

Return: return 0 on success

3.4.2 REQB

Prototype: int WINAPI rf_atqb(unsigned char model,
unsigned char *pData,
unsigned char *pMsgLg)

Parameter: model: [IN] 0=REQB, 1=WUPB
pDate: [OUT] response data from tag,
according with ISO14443 protocol
pMsgLg: [OUT] length of response data

Return: return 0 on success

3.4.3 T=CL Protocol Transceive

Prototype: int WINAPI rf_cos_command(unsigned char *pCommand,
unsigned char cmdLen,
unsigned char *pData,
unsigned char *pMsgLg)

Parameter: pCommand: [IN] COS command
cmdLen: [IN] length of COS command
pDate: [OUT] response data from tag, including SW1 &SW2
pMsgLg: [OUT] length of response data

Return: return 0 on success

3.4.4 T=CL Protocol Deselect

Prototype: int WINAPI rf_cl_deselect(void)

Return: return 0 on success

3.4.5 AT88RF020 Authentify

Prototype: int WINAPI rf_at020_check (unsigned char *pKey)

Parameter: pKey: [IN] 8 bytes password

Return: return 0 on success

3.4.6 AT88RF020 Count

Prototype: int WINAPI rf_at020_count(unsigned char *pData)

Parameter: pDate: [IN] signature, 6 bytes

Return: return 0 on success

3.4.7 AT88RF020 Read

Prototype: int WINAPI rf_at020_read (unsigned char page,
unsigned char *pData,
unsigned char *pMsgLen)

Parameter: page: [IN] page address, 0 ~ 31
pDate: [OUT] response data from tag
pMsgLen: [OUT] length of response data

Return: return 0 on success

3.4.8 AT88RF020 Write

Prototype: int WINAPI rf_at020_write(unsigned char page,
unsigned char *pData)

Parameter: page: [IN] page address, 0 ~ 31
pDate: [IN] written data, 8 bytes

Return: return 0 on success

3.4.9 AT88RF020 Lock

Prototype: int WINAPI rf_at020_lock(unsigned char *pData)
Parameter: pDate: [IN] data, 4 bytes
Return: return 0 on success

3.4.10 AT88RF020 Deselect

Prototype: int WINAPI rf_at020_deselect(void)
Return: return 0 on success
Explanation: card will exit active state after receiving this order and never respond other command untill enter active state again.

3.4.11 ST Select

Prototype: int WINAPI rf_st_select(unsigned char *pChip_ID)
Parameter: pChip_ID: [IN] return card ID number, one byte
Return: return 0 on success

3.4.12 SR176 Read

Prototype: int WINAPI int rf_sr176_read block (unsigned char block,
unsigned char *pData,
unsigned char *pLen)
Parameter: block: [IN] block address
pData: [OUT] response data from tag
pLen: [OUT] length of response data
Return: return 0 on success

3.4.13 SR176 Write

Prototype: int WINAPI int rf_sr176_write block (unsigned char block,
unsigned char *pData)
Parameter: block: [IN] block address
pData: [IN] written data, 2 bytes
Return: return 0 on success

3.4.14 SR176 Lock

Prototype: int WINAPI int rf_sr176_protect block (unsigned char lockreg)
Parameter: lockreg: [IN] LOCKREG
Return: return 0 on success
Explanation: SR176 has 16 modules, every lockreg controls 2 modules

lockreg	BLOCK	bit_setting	
b7	14 & 15	0:Write Enable	1:Block set as ROM
b6	12 & 13	0:Write Enable	1:Block set as ROM
b5	10 & 11	0:Write Enable	1:Block set as ROM
b4	8 & 9	0:Write Enable	1:Block set as ROM
b3	6 & 7	0:Write Enable	1:Block set as ROM
b2	4 & 5	0:Write Enable	1:Block set as ROM
b1	2 & 3	0:Write Enable	1:Block set as ROM
b0	0 & 1	0:Write Enable	1:Block set as ROM

3.4.15 SRIX4K Get UID

Prototype: int WINAPI int rf_srix4k_getuid (unsigned char *pUid,
unsigned char *pLen)
Parameter: pUid: [OUT] response data from tag, UID
pLen: [OUT] length of UID
Return: return 0 on success

3.4.16 SRIX4K Read

Prototype: int WINAPI int rf_srix4k_readblock (unsigned char block,
unsigned char pData,
unsigned char *pLen)
Parameter: block: [IN] block address
pData: [OUT] response data from tag
pLen: [OUT] length of response data
Return: return 0 on success

3.4.17 SRIX4K Write

Prototype: int WINAPI int rf_srix4k_writeblock (unsigned char block,
unsigned char *pData)
Parameter: block: [IN] block address
pData: [IN] written data, 4 bytes
Return: return 0 on success

3.4.18 SRIX4K Lock

Prototype: int WINAPI int rf_srix4k_protectblock (unsigned char lockreg)
Parameter: lockreg: [IN] LOCKREG
Return: return 0 on success

Explanation: 7-15 blocks of SRIX4K card can be written-protect

lockreg	BLOCK	bit_setting	
b7	15	1:Write Enable	0:Block set as ROM
b6	14	1:Write Enable	0:Block set as ROM
b5	13	1:Write Enable	0:Block set as ROM
b4	12	1:Write Enable	0:Block set as ROM
b3	11	1:Write Enable	0:Block set as ROM
b2	10	1:Write Enable	0:Block set as ROM
b1	9	1:Write Enable	0:Block set as ROM
b0	7 & 8	1:Write Enable	0:Block set as ROM

3.4.19 ST Desactived

Prototype: int WINAPI rf_st_completion(void)
Return: return 0 on success

3.4.20 THR1064 Read

Prototype: int WINAPI rf_thr1064_read (unsigned char page,
unsigned char *pData,
unsigned char *pMsgLen)
Parameter: page: [IN] page address, 0 ~ 3
pDate: [OUT] response data from tag
pMsgLen: [OUT] length of response data
Return: return 0 on success

3.4.21 THR1064 Write

Prototype: int WINAPI rf_thr1064_write (unsigned char page,
unsigned char *pData,
unsigned char *pMsgLen);
Parameter: page: [IN] page address, 0 ~ 3
pDate: [IN] written data
pMsgLen: [IN] length of written data
Return: return 0 on success

3.4.22 THR1064 Authentify

Prototype: int WINAPI rf_thr1064_check (unsigned char *pKey)
Parameter: pKey: [IN] 8 bytes password
Return: return 0 on success

3.5 ISO15693 FUNCTION

3.5.1 ISO15693_Inventory

Prototype: int WINAPI ISO15693_Inventory (unsigned char *pData,
unsigned char *pLen)

Parameter: pData: [OUT] response data from tag, 1byteDSFID + 8bytesUID
pLen: [OUT] length of response data

Return: return 0 on success

3.5.2 ISO15693_Inventorys

Prototype: int WINAPI ISO15693_Inventorys (unsigned char *pData,
unsigned char *pLen)

Parameter: pData: [OUT] response data from tag, every 9 bytes is a team,
the structure of every team is: 1byte DSFID + 8bytesUID
pLen: [OUT] length of response data

Return: return 0 on success

3.5.3 ISO15693_Get_System_Information

Prototype: int WINAPI ISO15693_Get_System_Information (unsigned char model,
unsigned char *pUID,
unsigned char *pData,
unsigned char *pLen)

Parameter: model: [IN] bit0=Select_flag,bit1=Addres_flag,bit2=Option_flag
pUID: [IN] 8 bytes UID
pData: [OUT] response data from tag
pLen: [OUT] length of response data

Return: return 0 on success

Explanation: If set Select_flag, only the cards on Selected state respond this command
If set Address_flag, only the cards that the UID are congruous will respond this command
Clear Option_flag

3.5.4 ISO15693_Select

Prototype: int WINAPI ISO15693_Select(unsigned char *pUID)

Parameter: pUID: [IN] 8 bytes UID

Return: return 0 on success

3.5.5 ISO15693_Reset_To_Ready

Prototype: int WINAPI ISO15693_Reset_To_Ready(unsigned char model,
unsigned char *pUID)

Parameter: model: [IN] bit0=Select_flag,
bit1=Address_flag,
bit2=Option_flag
pUID: [IN] 8 bytes UID

Return: return 0 on success

Explanation: If set Select_flag, only the cards on Selected state respond this command
If set Address_flag, only the cards that the UID are congruous will respond this command
Clear Option_flag

3.5.6 ISO15693_Stay_Quiet

Prototype: int WINAPI ISO15693_Stay_Quiet(unsigned char *pUID)

Parameter: pUID: [IN] 8 bytes UID

Return: return 0 on success

3.5.7 ISO15693_Get_Block_Security

Prototype: int WINAPI ISO15693_Get_Block_Security (unsigned char model,
unsigned char *pUID,
unsigned char block,
unsigned char number,
unsigned char *pData,
unsigned char *pLen)

Parameter: model: [IN] bit0=Select_flag,
bit1=Address_flag,
bit2=Option_flag
pUID: [IN] 8 bytes UID
block: [IN] block address
number: [IN] the number of module to be read, < 0x40
pData: [OUT] response data from tag
pLen: [OUT] length of response data

Return: return 0 on success

Explanation: If set Select_flag, only the cards on Selected state respond this command
If set Address_flag, only the cards that the UID are congruous will respond this command
Clear Option_flag

3.5.8 ISO15693_Read

Prototype: int WINAPI ISO15693_Read(unsigned char model,
unsigned char *PUID,
unsigned char block,
unsigned char number,
unsigned char *pData,
unsigned char *pLen);

Parameter:			
	model:	[IN]	bit0=Select_flag, bit1=Address_flag, bit2=Option_flag
	pUID:	[IN]	UID 8 bytes
	block:	[IN]	block address
	number:	[IN]	the number of module to be read, max 16 blocks
	pData:	[OUT]	response data from tag
	pLen:	[OUT]	length of response data

Return: return 0 on success

Explanation: If set Select_flag, only the cards on Selected state respond this command
If set Address_flag, only the cards that the UID are congruous will respond this command
Clear Option_flag

3.5.9 ISO15693_Write

Prototype: int WINAPI ISO15693_Write (unsigned char model,
unsigned char *pUID,
unsigned char block,
unsigned char *pData);

Parameter:			
	model:	[IN]	bit0=Select_flag, bit1=Address_flag, bit2=Option_flag
	pUID:	[IN]	8 bytes UID
	block:	[IN]	block address
	pData:	[IN]	written data, 4bytes

Return: return 0 on success

Explanation: If set Select_flag, only the cards on Selected state respond this command
If set Address_flag, only the cards that the UID are congruous will respond this command
If write TI card, set Option_flag,
If write I.CODE SLI card, clear Option_flag

3.5.10 ISO15693_Lock_Block

Prototype: int WINAPI ISO15693_Lock_Block (unsigned char model,
unsigned char *pUID,
unsigned char block)

Parameter: model: [IN] bit0=Select_flag,
bit1=Address_flag,
bit2=Option_flag
pUID: [IN] 8 bytes UID
block: [IN] block address

Return: return 0 on success

Explanation: If set Select_flag, only the cards on Selected state respond this command
If set Address_flag, only the cards that the UID are congruous will respond this command
If write TI card, set Option_flag,
If write I.CODE SLI card, clear Option_flag

3.5.11 ISO15693_Write_AFI

Prototype: int WINAPI ISO15693_Write_AFI (unsigned char model,
unsigned char *pUID,
unsigned char AFI)

Parameter: model: [IN] bit0=Select_flag,
bit1=Address_flag,
bit2=Option_flag
pUID: [IN] 8 bytes UID
AFI: [IN] written AFI

Return: return 0 on success

Explanation: If set Select_flag, only the cards on Selected state respond this command
If set Address_flag, only the cards that the UID are congruous will respond this command
If write TI card, set Option_flag,
If write I.CODE SLI card, clear Option_flag

3.5.12 ISO15693_Lock_AFI

Prototype: int WINAPI ISO15693_Lock_AFI(unsigned char model,
unsigned char *pUID)

Parameter: model: [IN] bit0=Select_flag,
bit1=Address_flag,
bit2=Option_flag
pUID: [IN] 8 bytes UID

Return: return 0 on success

Explanation: If set Select_flag, only the cards on Selected state respond this command
If set Address_flag, only the cards that the UID are congruous will respond this command

If write TI card, set Option_flag,
If write I.CODE SLI card, clear Option_flag

3.5.13 ISO15693_Write_DSFID

Prototype: int WINAPI ISO15693_Write_DSFID (unsigned char model,
unsigned char *pUID,
unsigned char DSFID)

Parameter: model: [IN] bit0=Select_flag,
bit1=Address_flag,
bit2=Option_flag
pUID: [IN] 8 bytes UID
DSFID: [IN] written DSFID

Return: return 0 on success

Explanation: If set Select_flag, only the cards on Selected state respond this command
If set Address_flag, only the cards that the UID are congruous will respond this command
If write TI card, set Option_flag,
If write I.CODE SLI card, clear Option_flag

3.5.14 ISO15693_Lock_DSFID

Prototype: int WINAPI ISO15693_Lock_DSFID (unsigned char model,
unsigned char *pUID)

Parameter: model: [IN] bit0=Select_flag,
bit1=Address_flag,
bit2=Option_flag
pUID: [IN] 8 bytes UID

Return: return 0 on success

Explanation: If set Select_flag, only the cards on Selected state respond this command
If set Address_flag, only the cards that the UID are congruous will respond this command
If write TI card, set Option_flag,
If write I.CODE SLI card, clear Option_flag