RFID READER

NFC Reader / Writer

SL600
User Manual

Version 1.1
Feb 2015
StrongLink

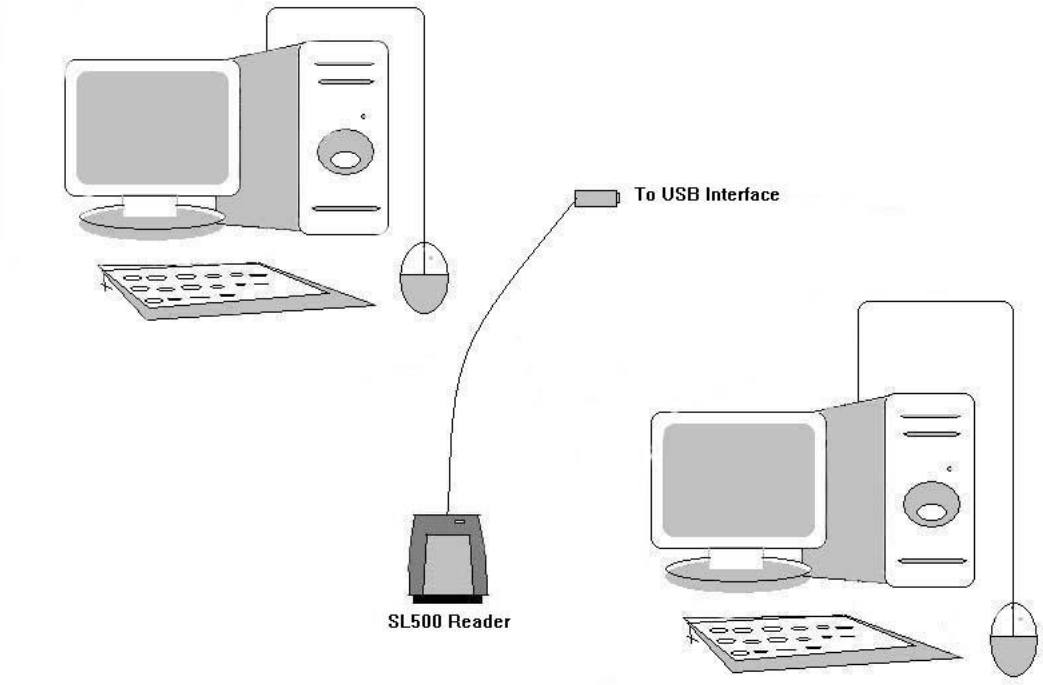
CONTENT

1. MAIN FEATURES

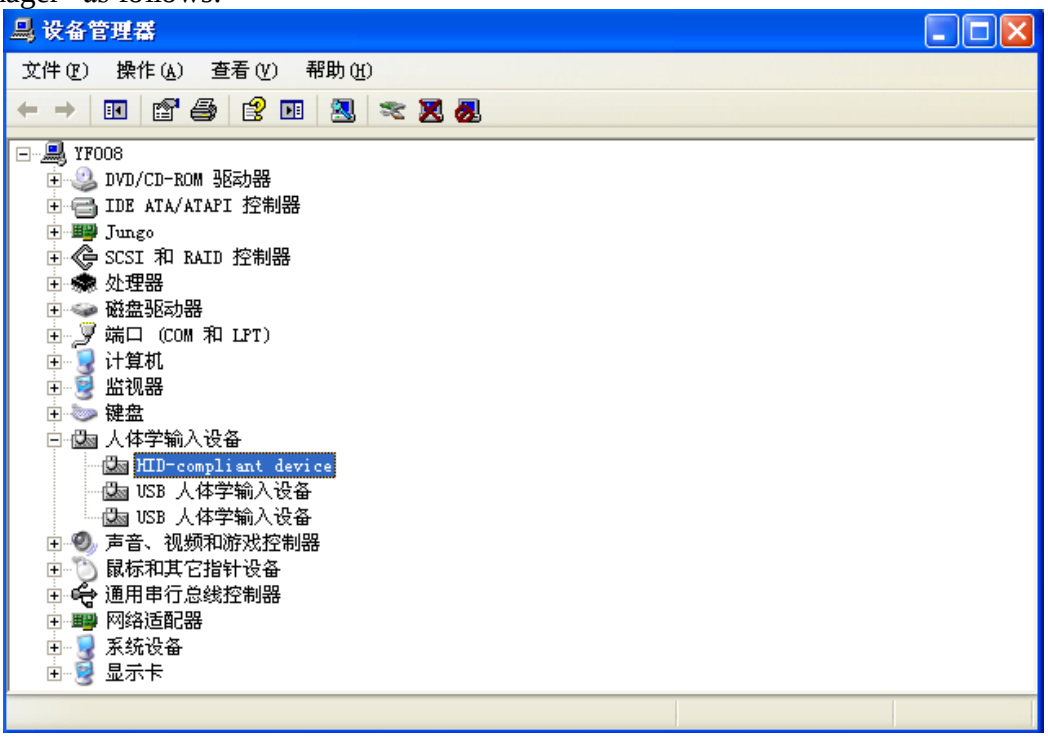
- Tags Supported: Mifare 1k, Mifare 4k, Mifare UltraLight, Mifare UltraLight C, DESFire and NTAG203
- NFCIP-1 Message Function
- USB HID Protocol Interface
- 13.56MHz RF Operating Frequency
- ISO14443, ISO18092 NFCIP-1 mode(text & url content)
- Windows 32bit/64bit Operation Systems Compatibility
- Offering DLL for Windows 32bit/64bit Operating Systems
- Work current less than 60mA
- Storage temperature: -40℃ ~ +75℃
- Operating temperature: -20℃ ~ +55℃
- Dimension: 110 ×81 ×26 mm
- Weight: 100g

2. CONNECTING TO PC

2.1 SL600- USB



Connect SL600 to the USB port of PC, then you can find the device in the “Device Manager” as follows:



3. COMMUNICATION PROTOCOL

3.1 Communication Setting

SL600 is kind of HID-compliant device, so there is no need to install the driver.
For HID-compliant device, user can send and receive data using “set report” and “get report” functions of USB HID protocol.
Of course, we also offer DLL for users to develop their own applications. Details see to chapter 4.

3.2 Communication Format

Host to Reader(Set Report in HID protocol):

Preamble	Len	DeviceID	Command	Data	Checksum

Preamble: 2 bytes equal to 0xAABB
Len: 2 bytes, indicating the number of bytes from DeviceID to Checksum
the left byte is used, and the right byte is also 0x00
DeviceID: 2 bytes (default: 0x00)
Command: 2 bytes, Command code
Data: Variable length depending on the command code
Checksum: 1 byte, XOR of all the bytes from DeviceID to Data

Reader to Host(Get Report in HID protocol):

Preamble	Len	DeviceID	Command	Status	Data	Checksum

Preamble: 2 bytes equal to 0xAABB
Len: 2 bytes, indicating the number of bytes from DeviceID to Checksum
the left byte is used, and the right byte is also 0x00
DeviceID: 2 bytes (default: 0x00)
Command: 2 bytes Command code
Status: 1 byte, 0 = success, not 0 = fail
Data: Response data, not necessary
Checksum: 1 byte, XOR of all the bytes from DeviceID to Data

4. DLL FUNCTION

Declarations of all the functions can also be found in the head file in the SDK with DLL.

4.1 SYSTEM FUNCTION

4.1.1 INT WINAPI LIB_VER

Function: Get DLL Version
Prototype: int WINAPI lib_ver (unsigned int *pVer)
Parameter: pVer: [OUT] DLL version
Return: return 0 if successful

4.1.2 INT WINAPI RF_GET_MODEL

Function: Get Device Type
Prototype: int WINAPI rf_get_model (unsigned short icdev,
unsigned char *pVersion,
unsigned char *pLen)
Parameter: icdev: [IN] Device ID
pVersion: [OUT] response information
pLen: [OUT] length of response information
Return 0 when success

4.1.3 INT WINAPI RF_INIT_TYPE

Function: Set Reader contactless working mode
Prototype: int WINAPI rf_init_type(unsigned short icdev, unsigned char type)
Parameter: icdev: [IN] Device ID
type: [IN] reader working mode
Return 0 when success
Explanation: this function is not effective to the readers only support single protocol.
type = 'A': set SL060 into ISO14443A mode
type = 'B': set ISO14443B mode
type = 'r': set AT88RF020 card mode
type = '1': set ISO15693 mode

4.1.4 INT WINAPI RF_ANTENNA_STA

Function: Manage RF status
Prototype: int WINAPI rf_antenna_sta (unsigned short icdev, unsigned char model)
Parameter: icdev: [IN] Device ID
model: [IN] transmittal state
Return 0 when success
Explanation: model = 0: turn off RF
model = 1: turn on RF

4.1.5 INT WINAPI RF_LIGHT

Function: Manage LED
Prototype: int WINAPI rf_light (unsigned short icdev, unsigned char color)
Parameter: icdev: [IN] Device ID
color: [IN] 0 = off
1 = on
Return 0 when success

4.1.6 HANDLE WINAPI RF_INIT_USB

Function: Init usb
Prototype: HANDLE WINAPI rf_init_usb(int HIDNum)

Parameter: HIDNum: [IN] the HID number of device
Return the handle of device when success

4.1.7 INT WINAPI RF_GET_DEVICE_NAME

Function: Get device's name
Prototype: int WINAPI rf_get_device_name(int HIDNum, char *buf, int sz)
Parameter: HIDNum: [IN] the HID number of device
sz: [IN] the length of name, maximum 255
buf: [OUT] the buf of device name

Return 0 when success

4.2 ENCRYPTION FUNCTION

4.2.1 INT WINAPI DES_ENCRYPT

Function: 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 0 when success

4.2.2 INT WINAPI DES_DECRYPT

Function: 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 0 when success

4.3 ISO14443A FUNCTION

4.3.1 Mifare Class

4.3.1.1 INT WINAPI RF_REQUEST

Function: ReqA
Prototype: int WINAPI rf_request (unsigned short icdev,
unsigned char model,
unsigned short *pTagType)
Parameter: icdev: [IN] Device ID
model: [IN] REQ MODE
pTagType: [OUT] response data, chip type code

Return 0 when success
Annotation: mode = 0x26: REQ_STD
mode = 0x52: REQ_ALL

4.3.1.2 INT WINAPI RF_ANTICOLL

Function: Mifare card Anti-collision
Prototype: int WINAPI rf_anticoll (unsigned short icdev,
unsigned char bcnt,
unsigned char *pSnr,
unsigned char *pLen)

Parameter:	icdev:	[IN]	Device ID
	bcnt:	[IN]	must be 4
	pSnr:	[OUT]	response data from card, unique serial number
	pLen:	[OUT]	length of response data

Return 0 when success

4.3.1.3 INT WINAPI RF_SELECT

Function: Select Mifare card
Prototype: int WINAPI rf_select (unsigned short icdev,
unsigned char *pSnr,
unsigned char snrLen,
unsigned char *pSize)

Parameter:	icdev:	[IN]	Device ID
	pSnr:	[IN]	card unique serial number
	snrLen:	[IN]	length of pSnr
	pSize:	[OUT]	response data from card, capacity code

Return 0 when success

4.3.1.4 INT WINAPI RF_M1_AUTHENTICATION2

Function: Mifare_Std Authentication
Prototype: int WINAPI rf_M1_authentication2 (unsigned short icdev,
unsigned char model,
unsigned char block,
unsigned char *pKey)
Parameter: icdev: [IN] Device ID
model: [IN] key validate mode
block: [IN] block absolute address
pKey: [IN] 6 bytes password
Return 0 when success
Annotation: model = 0x60: use KeyA
model = 0x61: use KeyB

4.3.1.5 INT WINAPI RF_M1_READ

Function: MifareOne Read
Prototype: int WINAPI rf_M1_read (unsigned short icdev,
unsigned char block,
unsigned char *pData,
unsigned char *pLen)
Parameter: icdev: [IN] Device ID
block: [IN] block absolute address
pData: [OUT] response data from card
pLen: [OUT] length of response data
Return 0 when success

4.3.1.6 INT WINAPI RF_M1_WRITE

Function: Mifare_Std Write
Prototype: int WINAPI rf_M1_write (unsigned short icdev,
unsigned char block,
unsigned char *pData)
Parameter: icdev: [IN] Device ID
block: [IN] block absolute address
pData: [IN] written data, 16 bytes
Return 0 when success

4.3.1.7 INT WINAPI RF_M1_INITVAL

Function: Mifare_Std card Initialize Value
Prototype: int WINAPI rf_M1_initval (unsigned short icdev,
unsigned char block,
long value)
Parameter: icdev: [IN] Device ID
block: [IN] block absolute address
pValue: [IN] initialize purse value at HEX format, low byte in former
Return 0 when success

4.3.1.8 INT WINAPI RF_M1_READVAL

Function: Mifare_Std Read Value
Prototype: int WINAPI rf_M1_readval (unsigned short icdev,
unsigned char block,
long *pValue)
Parameter: icdev: [IN] Device ID

block: [IN] block absolute address
pValue: [OUT] response value in HEX format, low byte in former
Return 0 when success

4.3.1.9 INT WINAPI RF_M1_INCREMENT

Function: Mifare purse increment
Prototype: int WINAPI rf_M1_increment (unsigned short icdev,
unsigned char block,
long value)
Parameter: icdev: [IN] Device ID
block: [IN] block absolute address
value: [IN] increase value at HEX format, low byte in former
Return 0 when success

4.3.1.10 INT WINAPI RF_M1_DECREMENT

Function: Mifare purse decrement
Prototype: int WINAPI rf_M1_decrement (unsigned short icdev,
unsigned char block,
long value)
Parameter: icdev: [IN] Device ID
block: [IN] block absolute address
value: [IN] decrease value at HEX format, low byte in former
Return 0 when success

4.3.1.11 INT WINAPI RF_M1_RESTORE

Function: Mifare_Std Restore
Prototype: int WINAPI rf_M1_restore (unsigned short icdev, unsigned char block)
Parameter: icdev: [IN] Device ID
block: [IN] block absolute address
Return 0 when success

4.3.1.12 INT WINAPI RF_M1_TRANSFER

Function: Mifare_Std Transfer
Prototype: int WINAPI rf_M1_transfer (unsigned short icdev, unsigned char block)
Parameter: icdev: [IN] Device ID
block: [IN] block absolute address
Return 0 when success
Annotation: this function only be transferred after increment, decrement and restore command

4.3.1.13 INT WINAPI RF_HALT

Function: Mifare Halt
Prototype: int WINAPI rf_halt (unsigned short icdev)
Parameter: icdev: [IN] Device ID
Return 0 when success
Annotation: card will exit active estate after received this command

4.3.2 Ultralight Class

4.3.2.1 INT WINAPI RF_UL_SELECT

Function: Select Ultralight card
Prototype: int WINAPI rf_ul_select(unsigned short icdev,
unsigned char *pSnr,
unsigned char *pLen)

Parameter: icdev: [IN] Device ID
Return 0 when success

4.3.2.2 INT WINAPI RF_UL_WRITE

Function: Write Ultralight card
Prototype: int WINAPI rf_ul_write(unsigned short icdev,
unsigned char page,
unsigned char *data)
Parameter: icdev: [IN] Device ID
page: [IN] page absolute address
data: [IN] written data, 4 bytes
Return 0 when success

4.3.2.3 INT WINAPI RF_ UC_AUTHENTICATION

Prototype: int WINAPI rf_UC_authentication(unsigned short icdev,
unsigned char *pKey)
Parameter: icdev: [IN] Device ID
pKey: [IN] key, 16 bytes
Return 0 when success

4.3.2.4 INT WINAPI RF_ UC_AUTHENTICATION

Prototype: int WINAPI rf_UC_changekey(unsigned short icdev,
unsigned char *pKey)
Parameter: icdev: [IN] Device ID
pKey: [IN] key, 16 bytes
Return 0 when success

4.3.3 DESFire Class

4.3.3.1 INT WINAPI RF_ DESFIRE_RST

Prototype: int WINAPI rf_DESFire_rst(unsigned short icdev,
unsigned char model,
unsigned char *pData,
unsigned char *pMsgLg)
Parameter: icdev: [IN] Device ID
model: [IN] the Model of searching card
pData: [IN] Response data
pMsglg: [IN] the Length of Response data
Return 0 when success

4.3.3.2 INT WINAPI RF_TYPE_RST

Function: Request ISO14443A-4 card and reset
Prototype: int WINAPI rf_typea_rst (unsigned short icdev,
unsigned char model,
unsigned char *pData,
unsigned char *pMsgLg)
Parameter: icdev: [IN] Device ID
model: [IN] request mode
pData: [OUT] response data from card
pMsgLg: [OUT] length of response data
Return 0 when success
Annotation: mode = 0x26: REQ_STD
mode = 0x52: REQ_ALL

pData: 4bytes CSN + RATS according to ISO14443A

4.3.3.3 INT WINAPI RF_ COS_COMMAND

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

Parameter:			
	icdev:	[IN]	Device ID
	pCommand:	[IN]	COS command
	cmdLen:	[IN]	length of COS command
	pData:	[OUT]	response data from card, including SW1& SW2
	pMsgLg:	[OUT]	length of response data

Return 0 when success

4.3.3.4 INT WINAPI RF_CL_DESELECT

Prototype: int WINAPI rf_cl_deselect (unsigned short icdev)

Parameter:			
	icdev:	[IN]	Device ID

Return 0 when success

4.4 NFC FUNCTION

4.4.1 INT WINAPI RF_NFC_SWITCH

Function: Enable NFC function

Prototype: int WINAPI rf_NFC_switch(unsigned short icdev,
unsigned char *pData)

Parameter:			
	icdev:	[IN]	Device ID
	pData:	[IN]	*pData=0:off *pData=1:on

Return 0 when success

4.4.2 INT WINAPI RF_NFC_NDEF

Function: send NFC NDEF message to NFC device

Prototype: int WINAPI rf_NDEF (unsigned short icdev,
unsigned char model,
unsigned char *pData)

Parameter:			
	icdev:	[IN]	Device ID
	model:	[IN]	NDEF MODE 0=text, 1=url
	pData:	[IN]	NDEF message data(max 96 bytes)

Return 0 when success