IT WORKS
Everything Works

IT WORKS

RFID-M1

API

Programmer Guide

Revision 2.00

http://www.itworks.co.th email: biosupport@itworksolutions.com

Table of Contents

1 API Functions

1.1 Configuration, I/O commands

1.1.1 Int GetVersion API(char *VersionAPI)

Parameter	Description
VersionAPI	Character pointer to C string which return the Version Number of the API
Return	0x00 – Successful

Description

Get the Version Nmber of the API

1.1.2 int ActiveLED (int DeviceAddress, unsigned char NumLED, unsigned char ontime, ussigned char cycle)

Parameter	Description
DeviceAddress	Device Address of the reader
NumLED	The LED to be selected
	0x01 - Red LED
	0x02- Green LED
	0x03 – Both Red & Green LED
on-time	Units of the LED turn-on time (duty cycle). Each unit is 100ms
cycle	Number of cycles that the LED will be turned on and off.
Return	0x00 – Successful
	(Refer the API return code for other values)

Description

The LED is blinking for the number of cycles. Each cycle is one second. The turn-on time of the LED in each cycle is set by the “on-time”.

Example

The Green LED will be toggled (turn on and off) for three times. Each time the LED will be turned on for 400ms.

```
DeviceAddress=0x00;
unsigned char NumLED  = 0x02; //Select Green LED
unsigned char ontime  = 4;  //LED on time =400ms
unsigned char cycle   = 3;  //on-off  3 times (3 second)
if(!ToggleLED(DeviceAddress, NumLED, ontime, cycle))
{ //successful  }
```

1.1.3 int SetLED (int DeviceAddress, unsigned char LEDState)

Parameter	Description
DeviceAddress	Device Address of the reader
LEDState	Turn on/off the LEDs.
	Bit0 - Red LED. 0= LED on, 1= LED off.

	Bit1 - Green LED. 0= LED on, 1= LED off.
	Bit2-7 - unused
Return	0x00 – Successful
	(Refer the API return code for other values)

Description

Turn on/off the selected LEDs

1.1.4 int ActiveBuzzer (int DeviceAddress, unsigned char mode, unsigned char *pattern)

Parameter	Description
DeviceAddress	Device Address of the reader
mode	buzzer mode
	0 – Turn off the buzzer
	1 – Turn on the buzzer.
	4 – Play a sound pattern. The pattern is a sequence of on-off-on-off sound controlled by the parameter array "pattern".
pattern	Pointer to a parameter array which controls the sound pattern.
	pattern[0]: Units of first on time. Each unit is 100ms.
	pattern[1]: Units of first off time.
	pattern[2]: Units of second on time.
	pattern[3]: Units of second off time.
	pattern[4]: Cycle
Return	0x00 – Successful
	(Refer the API return code for other values)

Description

Control the buzzer. You can control the buzzer to play a sound pattern by using mode 4.

Example

A sound pattern (on 0.5sec – off 0.3sec – on 1sec –off 0.6sec) will be played for 7 times.

```
DeviceAddress=0x00;
pattern[0]      = 5;
pattern[1]      = 3;
pattern[2]      = 10;
pattern[3]      = 6;
pattern[4]      = 7;
if (!BuzzerControl(DeviceAddress, 0x04, pattern))
{ //successful  }
else
{ //Not successful }
```

1.1.5 int RF_Field (int DeviceAddress, unsigned char time)

Parameter	Description
DeviceAddress	Device Address of the reader
time	Units of time to turn off the RF filed. Each unit is 100us.
Return	0x00 – Successful (Refer the API return code for other values)

Description

Turn off the RF field for the units of time after that the RF will be on again. The value 0x00 will turn off the field forever until a new RF_Field( ) command is issued.

1.2 ISO14443-A Command

1.2.1 int MF_Request (int DeviceAddress, unsigned char mode, unsigned char *ATQ)

Parameter	Description
DeviceAddress	Device Address of the reader
Mode	0x00 – request all (wake up all)
	0x01 – request idle
ATQ	two bytes of ATQ returned from the contactless card.
Return	0x00 – Successful
	(Refer the API return code for other values)

Description

Send the ISO14443 A REQUEST command to the card. The two-byte ATQ string will be returned.

Example

A REQUEST-ALL command will be sent to the card. The ATQ (0x0004) of the card will be returned if the request command is successful.

```
DeviceAddress=0x00;
unsigned char ATQ[2];
if (!Request(DeviceAddress, 0x00, ATQ))
{ //successful ATQ[0]= 0x04; ATQ[1] = 0x00  }
else
{ //Not successful }
```

NOTE: The ATQ for the MIFARE®1 card is 0x0004, other value may be return from cards other then MIFARE®1.

1.2.2 int MF_Anticoll (int DeviceAddress, unsigned char *UID, unsigned char &Collision)

Parameter	Description
DeviceAddress	Device Address of the reader
UID	Pointer to four bytes buffer for the UID (card serial number) returned by the anticollision loop.
Collision	Collision Flag.
	0x00 – No collision detected.
	0x01 – Collision detected. (More then one waked-up card detected in the field
Return	0x00 – Successful
	0x46 – Successful
	(Refer the API return code for other values)

Description

Enable the ISO14443A anti-collision loop of cascade level1, the card's UID of cascade level1 will be returned. The Collision flag indicates that a collision is happened. (There are more than one card in the Halt mode within the field)

int MF_Anticoll2 (int DeviceAddress, unsigned char *UID, unsigned char &Collision)

Parameter	Description
DeviceAddress	Device Address of the reader
UID	Pointer to four bytes buffer for the UID (card serial number) returned by the anticollision loop.
Collision	Collision Flag.
	0x00 – No collision detected.
	0x01 – Collision detected. (More then one waked-up card detected in the field
Return	0x00 – Successful
	(Refer the API return code for other values)

Description

Enable the ISO14443A anti-collision loop of cascade level2, the card's UID of cascade level2 will be returned. The Collision flag indicates that a collision is happened. (There are more than one card in the Halt mode within the field)

int MF_Anticoll3 (int DeviceAddress, unsigned char *UID, unsigned char &Collision)

Parameter	Description
DeviceAddress	Device Address of the reader
UID	Pointer to four bytes buffer for the UID (card serial number) returned by the anticollision loop.
Collision	Collision Flag.
	0x00 – No collision detected.
	0x01 – Collision detected. (More then one waked-up card detected in the field
Return	0x00 – Successful
	(Refer the API return code for other values)

Description

Enable the ISO14443A anti-collision loop of cascade level3, the card's UID of cascade level3 will be returned. The Collision flag indicates that a collision is happened. (There are more than one card in the Halt mode within the field)

1.2.3 int MF_Select(int Device Address, unsigned char *UID)

Parameter	Description
DeviceAddress	Device Address of the reader
UID	Pointer to a four-byte buffer storing the UID (card serial number) of the card to be selected.
Return	0x00 – Successful
	0x46 – Successful, need next anticollision
	(Refer the API return code for other values)

Description

ISO14443A SELECT command of Cascadelevel1. The requested card with the specified UID of Cascadelevel1 will be (open) for further card commands.

int MF_Select2(int Device Address, unsigned char *UID)

Parameter	Description
DeviceAddress	Device Address of the reader
UID	Pointer to a four-byte buffer storing the UID (card serial number) of the card to be selected.
Return	0x00 – Successful 0x46 – Successful, need next anticollision (Refer the API return code for other values)

Description

ISO14443A SELECT command of Cascadelevel2. The requested card with the specified UID of Cascadelevel2 will be (open) for further card commands.

int MF_Select3(int Device Address, unsigned char *UID)

Parameter	Description
DeviceAddress	Device Address of the reader
UID	Pointer to a four-byte buffer storing the UID (card serial number) of the card to be selected.
Return	0x00 – Successful (Refer the API return code for other values)

Description

ISO14443A SELECT command of Cascadelevel3. The requested card with the specified UID of Cascadelevel3 will be (open) for further card commands.

Example

Select the card. The serial number of the card is 0xF05320D1.

```
DeviceAddress=0x00;
unsigned char UID[4];
UID[0] = 0xD1;
UID[1] = 0x20;
UID[2] = 0x53;
UID[3] = 0xF0;
if (!Select(DeviceAddress, UID))
{ //successful  }
else
{ //Not successful }
```

1.2.4 int MF_Halt(int Device Address, unsigned char mode)

Parameter	Description
DeviceAddress	Device Address of the reader
Mode	Enable CRC/MAC as checksum

	0x00 – Standard ISO14443-A Halt command. Use the 16 bit CRC as checksum.
	0x01 – Special mode for the SLE55Rxx. The four-byte MAC is used to replace the CRC checksum. This special mode is used to halt a SLE55Rxx card which is under protected mode.
Return	0x00 – Successful
	(Refer the API return code for other values)

Description

ISO14443A Halt command. The MAC mode is a special mode for the SLE55Rxx card. The SLE55Rxx card enters protected mode after a successful AUTHENTICATION and uses the four-byte MAC to replace the 16-bit CRC checksum. Under the protected mode, you need to use the MAC mode to halt the SLE55Rxx cards. For normal MIFARE® card mode 0x00 should be used for the normal CRC checksum.

1.2.5 int SLE_Generic(int DeviceAddress,unsigned char CRC_Flag,unsigned char &length,unsigned char *buffer)

Parameter	**Description**
DeviceAddress	Device Address of the reader
CRC_Flag	Enable Flag.
	0x00 – No CRC checksum will be calculated and appended.
	0x01 – The CRC checksum will be calculated and appended.
Length	the byte number of buffer
Buffer	APDU to/form card
Return	0x00 – Successful
	(Refer the API return code for other values)

Description

ISO14443A general command. Used to Access ISO14443A CPU Card, e.g. Pro(X), DesFire.

1.3 Mifare® function

1.3.1 int MF_Auth(int DeviceAddress, unsigned char KeyAB, unsigned char *snr, unsigned char add_blk)

Parameter	Description
DeviceAddress	Device Address of the reader
KeyAB	Key A or B selection
	0x60: Use KEYA for authentication
	0x61: Use KEYB for authentication
snr	Pointer to a four-bytes buffer which stores the UID(card serial number) of the card to be authenticated.
add_blk	The address of the block (block number : 00..63) to be authenticated.
Return	0x00 – Successful
	(Refer the API return code for other values)

Description

This command is used to authenticate the selected Mifare® card. Further read/write and value related operations are allowed only after the successful authentication.

1.3.2 int MF_Read (int DeviceAddress,unsigned char add_blk, unsigned char num_blk, unsigned char *buffer);

Parameter	Description
DeviceAddress	Device Address of the reader
Add_blk	Start address of memory blocks to read
num_blk	Number(1 –4) of block to read.
Buffer	Pointer to the buffer which returns the data read from the card. The length of the buffer equals to num_blk *16 bytes.
Return	0x00 – Successful
	(Refer the API return code for other values)

Description

Read multiple (up to four) blocks from the Mifare® card.

Note: The blocks to be read must be in the same sector.

1.3.3 int MF_Write(int DeviceAddress,unsigned char add_blk, unsigned char num_blk, unsigned char *buffer)

Parameter	Description
DeviceAddress	Device Address of the reader
Add_blk	Start address of memory blocks to write
num_blk	Number(1 –4) of block to write.
Buffer	Pointer to the buffer which stores the data to be written to the card. The length of the buffer equals to num_blk *16 bytes.
Return	0x00 – Successful

(Refer the API return code for other values)

Description

Write multiple (up to four) blocks data to the Mifare® card.

Note: The blocks to be written must be in the same sector. Writing data to the sector trailer (block address = N*4+3, where N is the sector number) should be handled carefully; otherwise you may corrupt the KEY area and lock the sector.

1.3.4 int MF_Transfer(int DeviceAddress, unsigned char add_blk)

Parameter	Description
DeviceAddress	Device Address of the reader
Add_blk	Block address of the value block.
Return	0x00 – Successful
	(Refer the API return code for other values)

Description

Transfer a value amount from the Mifare® Reader Chip's internal value buffer register to the selected value block.

1.3.5 int MF_Incremnet(int DeviceAddress, unsigned char add_blk, int value)

Parameter	Description
DeviceAddress	Device Address of the reader
Add_blk	Block address of the value block.
Value	the value to be increased
Return	0x00 – Successful
	(Refer the API return code for other values)

Description

Increase the value of a MIFARE® Value Block. The value block must be pre-initialized according to the MIFARE® Value Block Format.

Note: The VALUE is in plain format, the user is no need to take care the MIFARE® value block format.

Note: The result of the MIFARE®'s Decrement or Increment operation is stored within the on-chip buffer register. The value in the selected value block will not be updated until the Transfer command is done.

1.3.6 int MF_Decrement(int DeviceAddress, unsigned char add_blk, int value)

Parameter	Description
DeviceAddress	Device Address of the reader
Add_blk	Block address of the value block.
Value	the value to be decreased
Return	0x00 – Successful
	(Refer the API return code for other values)

Description

Decrease the value of a MIFARE® Value Block. The value block must be initialized according to the MIFARE® Value Block Format.

Note: The VALUE is in plain format, the user is no need to take care the MIFARE® value block format.

1.3.7 int MF_Restore(int DeviceAddress, unsigned char add_blk)

Parameter	Description
DeviceAddress	Device Address of the reader
Add_blk	Block address of the value block.
Return	0x00 – Successful
	(Refer the API return code for other values)

Description

Restore the content of the selected Value Block to the MIFARE® Reader Chip's internal value buffer register.

1.3.8 int MF_InitValue(int DeviceAddress, unsigned char add_blk, int value)

Parameter	Description
DeviceAddress	Device Address of the reader
Add_blk	Block address of the value block.
Value	the value to be written. (Initialize the value block)
Return	0x00 – Successful
	(Refer the API return code for other values)

Description

Write (initialize) a value to a MIAFARE value block. The block to be written will be automatically formatted with the MIFARE® Value Block Format.

1.3.9 int MF_ReadValue(int DeviceAddress, unsigned char add_blk, int *value)

Parameter	Description
DeviceAddress	Device Address of the reader.
Add_blk	Block address of the value block.
Value	the value read back from the value block.
Return	0x00 – Successful
	(Refer the API return code for other values)

Description

Read back the value amount of a MIFARE® Value Block.

1.3.10 int MF_LoadKey(int DeviceAddress, unsigned char *Key)

Parameter	Description
DeviceAddress	Device Address of the reader
Key	Pointer to a 6-bytes buffer storing the key string.
Return	0x00 – Successful
	(Refer the API return code for other values)

Description

Directly load the key to the Master Key Buffer. A Master Key must be loaded to the Master Key Buffer (by MF_LoadKey ( ) or MF_LoadKeyFromEeprom( )) before executing the Authentication command.

1.3.11 int MF_StoreKeyToEE(int DeviceAddress, unsigned char KeyAB, unsigned char Sector, unsigned char *Key)

Parameter	Description
DeviceAddress	Device Address of the reader
KeyAB	Select KEYA or KEYB.
	0x60: the Key will be stored as KEYA.
	0x61: the Key will be stored as KEYB.
Sector	The sector number {0x00-0x0F}: where the key to be stored..
Key	Pointer to a 6-bytes buffer storing the uncoded key.string. (i.e. A0A1A2A3A4A5)
Return	0x00 – Successful
	(Refer the API return code for other values)

Description

Stores a key to the reader's EEPROM.

1.3.12 int MF_LoadKeyFromEF(int DeviceAddress, unsigned char KeyAB, unsigned char Sector)

Parameter	Description
DeviceAddress	Device Address of the reader
KeyAB	Select KEYA or KEYB.
	0x60: KEYA will be loaded from the EEProm.
	0x61: KEYB will be loaded from the EEProm.
Sector	The sector number {0x00-0x0F} of the key to be loaded..
Return	0x00 – Successful
	(Refer the API return code for other values)

Description

Load key to the Master Key Buffer from the MFRC500 chip's internal EEPROM. This function has the same function as the MF_LoadKey( ), but the key is loaded from the internal EEPROM instead providing the 6-byte uncoded key string

1.4 Mifare® High Level Functions

The High Level command integrated the low level commands – Request, Anti-Collision – Select – LoadKey – Authentication – Read/Write/Increment/Decrement to a single one-step operation.

1.4.1 int MF_HLRead (int DeviceAddress, unsigned char mode, unsigned char add_blk, unsigned char num_blk, unsigned char *snr, unsigned char *buffer);

Parameter	Description
DeviceAddress	Device Address of the reader
mode	Operating Mode
	Bit0 – All – Select Request all or Request IDLE mode (0/1= IDLE/ALL)
	Bit1 – SNR – Enable the Card Serial Number comparison features. The further operating will be processed only if the detected card number matches the card serial number sent by the Host.
	Bit2 – KeyB – Authenticate with KeyA or Key B. 0/1= KeyA/KeyB
Add_blk	Start address of memory blocks to read
num_blk	Number(1 –4) of block to read
snr	The serial number, which will be ignored if the SNR bit is not set, of the card to be selected.
	The serial number of the card selected will be return by the snr after the function is called successfully.
buffer	Pointer to the buffer which returns the data read from the card. The length of the buffer equals to num_blk *16 bytes.
Return	0x00 – Successful
	(Refer the API return code for other values)

Description

Read multiple (up to four) blocks from the Mifare® card.

Note : The blocks to be read must be in the same sector.

1.4.2 int MF_HLWrite (int DeviceAddress, unsigned char mode, unsigned char add_blk, unsigned char num_blk, unsigned char *snr, unsigned char *buffer);

Parameter	Description
mode	Operating Mode
	Bit0 – All – Select Request all or Request IDLE mode (0/1= IDLE/ALL)
	Bit1 – SNR – Enable the Card Serial Number comparison features. The further operating will be processed only if the detected card number matches the card serial number sent by the Host.
	Bit2 – KeyB – Authenticate with KeyA or Key B. 0/1= KeyA/KeyB
Add_blk	Starting block address of memory blocks to be written
num_blk	Number(1 –4) of block to write
snr	The serial number, which will be ignored if the SNR bit is not set, of the card to be selected.
	The serial number of the card selected will be return by the snr after the function is called successfully.
buffer	Pointer to the buffer which stores the data to be written to the card. The length of the buffer equals to num_blk *16 bytes.
Return	0x00 – Successful

	(Refer the API return code for other values)

Description

Write multiple (up to four) blocks data to the Mifare® card.

Note: The blocks to be written must be in the same sector. Writing data to the sector trailer (block address = N*4+3, where N is the sector number) should be handled carefully, otherwise you may corrupt the KEY area and lock the sector.

1.4.3 int MF_HLInitVal (int DeviceAddress, unsigned char mode, unsigned char sect_num, unsigned char *snr, int value);[1]

Parameter	Description
mode	Operating Mode
	Bit0 – All – Select Request all or Request IDLE mode (0/1= IDLE/ALL)
	Bit1 – SNR – Enable the Card Serial Number comparison features. The further operating will be processed only if the detected card number matches the card serial number sent by the Host.
	Bit2 – KeyB – Authenticate with KeyA or Key B. 0/1= KeyA/KeyB
sec_num	The number of the sector to be formatted as value block.
snr	The serial number, which will be ignored if the SNR bit is not set, of the card to be selected.
	The serial number of the card selected will be return by the snr after the function is called successfully.
value	The initial value to be written to the value blocks.
Return	0x00 – Successful
	(Refer the API return code for other values)

Description

Initials the value block (Block 1) and the back-up block (Block 2) to the MIFARE® VALUE BLOCK format with the initial value.

1.4.4 int MF_HLInc (int DeviceAddress, unsigned char mode, unsigned char sect_num, unsigned char *snr, int *value)

Parameter	Description
mode	Operating Mode
	Bit0 – All – Select Request all or Request IDLE mode (0/1= IDLE/ALL)
	Bit1 – SNR – Enable the Card Serial Number comparison features. The further operating will be processed only if the detected card number matches the card serial number sent by the Host.
	Bit2 – KeyB – Authenticate with KeyA or Key B. 0/1= KeyA/KeyB
sec_num	The number of the sector to be formatted as value block.
snr	The serial number, which will be ignored if the SNR bit is not set, of the card to be selected.

[1] For the functions MF_HLInitVal, MF_HLInc and MF_HLDec (High Level Value related commands), the second block (Block 1) is used as the Value Block of the selected sector. The third block (Block 2) is used as the backup of the value block and the first block (Block0) is free for use.

	The serial number of the card selected will be return by the snr after the function is called successfully.
value	The value to be increment.
	The content of the value block will be returned after the increment is done.
Return	0x00 – Successful
	(Refer the API return code for other values)

Description

Increase the value of a MIFARE® Value Block and also copy the value block to the backup block.

1.4.5 int MF_HLDec (int DeviceAddress, unsigned char mode, unsigned char sect_num, unsigned char *snr, int *value)

Parameter	Description
Mode	Operating Mode
	Bit0 – All – Select Request all or Request IDLE mode (0/1= IDLE/ALL)
	Bit1 – SNR – Enable the Card Serial Number comparison feature. The further operating will be processed only if the detected card number matches the card serial number sent by the Host.
	Bit2 – KeyB – Authenticate with KeyA or Key B. 0/1= KeyA/KeyB
sec_num	The number of the sector to be formatted as value block.
snr	The serial number, which will be ignored if the SNR bit is not set, of the card to be selected.
	The serial number of the card selected will be return by the snr after the function is called successfully.
Value	The value to be increment.
	The content of the value block will be returned after the decrement is done.
Return	0x00 – Successful
	(Refer the API return code for other values)

Description

Decrease the value of a MIFARE® Value Block and also copy the value block to the backup block.

1.4.6 int MF_StoreKeyToEE(int DeviceAddress, unsigned char KeyAB, unsigned char Sector, unsigned char *Key)

Parameter	Description
DeviceAddress	Device Address of the reader
KeyAB	Select KEYA or KEYB.
	0x60: the Key will be stored as KEYA.
	0x61: the Key will be stored as KEYB.
Sector	The sector number {0x00-0x0F}: where the key to be stored..
Key	Pointer to a 6-bytes buffer storing the uncoded key.string. (i.e. A0A1A2A3A4A5)
Return	0x00 – Successful
	(Refer the API return code for other values)

Description

Stores a key to the reader's EEPROM.

1.4.7 int MF_HLRequest (int DeviceAddress, unsigned char mode, int &length, unsigned char *UID)

Parameter	Description
DeviceAddress	Device Address of the reader
Mode	Operating Mode
	Bit0 – All – Select Request all or Request IDLE mode (0/1= IDLE/ALL)
	Bit1 – SNR – Enable the Card Serial Number comparison features. The further operating will be processed only if the detected card number matches the card serial number sent by the Host.
Length	number(4 -10 byte) of UID.
UID	Pointer to a 4-byte (or 7-byte, 10-byte) buffer storing the UID (card serial number) of the card to be selected.
Return	0x00 – Successful
	(Refer the API return code for other values)

Description

The MF_HLRequest command integrated the low level commands – Request, Anti-Collision – Select(AntiColl2-Seledt2, AntiColl3-Select) to a single one-step operation.

1.5 Misc. Commands

1.5.1 int SetFirmwareBaudrate(int DeviceAddress, unsigned char Baudrate)

Parameter	Description
DeviceAddress	Device Address of the reader
Baudrate	The baudrate of the Reader Module
	0x01 - 9600bps
	0x02 - 19200bps
	0x03 - 38400bps
	0x04 - 57600bps
	0x05 - 115200bps
Return	0x00 – Successful
	(Refer the API return code for other values)

Description

Change the communication baud rate of the reader.

Note: the SetRDRBaudrate() modify the default baud rate stored in the reader's internal EEPROM. The new setting will not take effect until the next reset of the reader.

Note: Need to change the baud rate of the Host controller correspondly.

1.5.2 int SetDeviceAddress (int DeviceAddress, unsigned char &newAddress unsigned char mode, char *SerialNum)

Parameter	Description
DeviceAddress	Device Address of the reader
newAddress	The new device address to be programmed to the reader
mode	0x00 - Disable Checking of the Serial Number
	0x01 - Enable Checking of the Serial Number
SerialNum	Pointer to the string storing the Serial Number
Return	0x00 – Successful
	(Refer the API return code for other values)

Description

The SetDeviceAddress() function programs a device address to the reader. If the Enable Serial Number Checking Flag is set, the correct Reader Serial Number must be submitted to program the device address.

Example

To program a new device address (0x04) to a reader whose current address is 0x01 and the serial number is "12345678"

```
DeviceAddress=0x01;
unsigned char newAddress=0x04;
unsigned char mode=0x01;
if(SetDeviceAddress(DeviceAddress,newAddress,mode,"12345678") )
```

```
{ //successful   }
else
{ //Not successful }
```

1.5.3 int GetVersionlNum (int DeviceAddress, char *VersionNUM)

Parameter	Description
DeviceAddress	Device Address of the reader
VersionNum	Pointer to the string of the Version Number. The version number is the firmware version of the reader device.
Return	0x00 – Successful (Refer the API return code for other values)

Description

Get the Reader's Firmware version number.

1.5.4 int GetSerialNum (int DeviceAddress, int &CurrentAddress, char *SerialNUM)

Parameter	Description
DeviceAddress	Device Address of the reader
CurrentAddress	Device address returned
SerialNum	Pointer to the string storing the Serial Number returned.
Return	0x00 – Successful (Refer the API return code for other values)

Description

Get the Device Address and the Serial Number from the reader.

Example

To program a new device address (0x04) to a reader whose current address is 0x01 and the serial number is "12345678"

```
DeviceAddress=0x00;
int CurrentAddress;
char SerialNum[8];
if(!GetSerialNum(DeviceAddress,CurrentAddress,SerialNum))
{ //successful
    CurrentAddress = 1,
    SerialNum =”12345678”
}
else
{ //Not successful }
```

1.5.5 int GetUserInfo (int DeviceAddress, char *UserInfo)

Parameter	Description
DeviceAddress	Device Address of the reader

UserInfo	Pointer to the 32-bytes user information.
Return	0x00 – Successful
	(Refer the API return code for other values)

Description

Get the 32-bytes programmable user information.

1.5.6 int SetUserInfo (int DeviceAddress, char *UserInfo)

Parameter	**Description**
DeviceAddress	Device Address of the reader
UserInfo	Pointer to the 32-bytes user information to be programmed.
Return	0x00 – Successful
	(Refer the API return code for other values)

Description

Program the 32-bytes user information to the reader.

1.5.7 int SetWiegandStatus(int DeviceAddress,unsigned char status)

Parameter	**Description**	
DeviceAddress	Device Address of the reader	
Status	bit0	1 ----- Buzzer and LED controlled by external I/O
		0 ----- Buzzer and LED not controlled by external I/O
	bit1	1 ----- Prompt from Buzzer and LED after successful read
		0 ------ No prompt from Buzzer and LED after successful read
	bit4	
		1 ------ Enable Wiegand mode
		0 ------- Disable Wiegand mode
Return	0x00 – Successful	
	(Refer the API return code for other values)	

Description

Set Wiegand Status of the reader, setting will be saved after power out.

1.5.8 int ActiveWiegandMode(int DeviceAddress,unsigned char status)

Parameter	**Description**	
DeviceAddress	Device Address of the reader	
Status	bit0	1 ----- Buzzer and LED controlled by external I/O

0 ----- Buzzer and LED not controlled by external I/O

bit1 1 ----- Prompt from Buzzer and LED after successful read

0 ------ No prompt from Buzzer and LED after successful read

bit4 1 ------ Enable Wiegand mode

0 ------- Disable Wiegand mode

Return	0x00 – Successful (Refer the API return code for other values)

Description

Set Wiegand Status of the reader, setting will not be saved after power out.

1.5.9 int ReadChar(unsigned char *byte)

Parameter	**Description**
Data Field	N/A-
Return	0x00 – Successful (Refer the API return code for other values)
byte[0]	The length of the data returned from the rs232
byte[1~n]	Data returned from the rs232.

Description

Get data string returned from the reader.

1.6 Huahong SHC1102 command set

1.6.1 int SHC1102_Auth(int DeviceAddress,unsigned char *key);

Parameter	**Description**
DeviceAddress	Device Address of the reader
Key	4 Byte key authentication
Return	0x00 – Successful (Refer the API return code for other values)

Description

Huahong SHC1102 card key authentication

1.6.2 int SHC1102_Read(int DeviceAddress,unsigned char block, unsigned char *dat)

Parameter	**Description**
DeviceAddress	Device Address of the reader
Block	Block address (0~15)
Data	4 Byte returned data
Return	0x00 – Successful

(Refer the API return code for other values)

Description

Read data from Huahong SHC1102 card

1.6.3 int SHC1102_Write(int DeviceAddress,unsigned char block, unsigned char *dat)

Parameter	Description
DeviceAddress	Device Address of the reader
Block	Block address (0~15)
Data	4 Byte date writing
Return	0x00 – Successful
	(Refer the API return code for other values)

Description

Write data to Huahong SHC1102 card

1.6.4 int SHC1102_Halt(int DeviceAddress)

Parameter	Description
DeviceAddress	Device Address of the reader
Return	0x00 – Successful
	(Refer the API return code for other values)

Description

Halt command of Huahong SHC1102 card

2 Additional Information

2.1 Error/Status Code

System Error/Status Codes (0x00-0x0F)

OK	0x00	Command OK. (success)
PARA_ERR	0x01	Parameter value out of range error
TMO_ERR	0x04	Reader reply time out error
SEQ_ERR	0x05	Communication Sequence Number out of order
CMD_ERR	0x06	Reader received unknown command
CHKSUM_ERR	0x07	Communication Check Sum Error
INTR_ERR	0x08	Unknown Internal Error

Card Error/Status Codes (0x10-0x1F)

NOTAG_ERR	0x11	No card detected
CRC_ERR	0x12	Wrong CRC received from card
PARITY_ERR	0x13	Wrong Parity Received from card
BITCNT_ERR	0x14	Wrong number of bits received from the card
BYTECNT_ERR	0x15	Wrong number of bytes received from the card
CRD_ERR	0x16	Any other error happened when communicate with card

MIFARE® Error/Status Codes (0x20-0x2F)

MF_AUTHERR	0x20	No Authentication Possible
MF_SERNRERR	0x21	Wrong Serial Number read during Anti-collision.
MF_NOAUTHERR	0x22	Card is not authenticated
MF_VALFMT	0x23	Not value block format
MF_VAL	0x24	Any problem with the VALUE related function

Type-B Card Error/Status Codes (0x30-0x3F)

<To be defined>

SAM Error/Status Codes (0x40-0x4F)

<To be defined>