

USB Reader 开发文档

1、版本说明(版本 V1.00)

* All rights reserved, Specifications subject to change without notice.

2、返回值说明：

2. 1 函数返回值代码表：

0x00	命令执行成功
0x01	命令操作失败（失败原因请参考 2.2 底层单片机上传代码表）
0x02	地址校验错误
0x04	读写器返回超时
0x05	数据包流水号不正确：
0x07	接收异常
0x0A	参数值超出范围

2. 2 底层单片机上传代码表：

0x80:	参数设置成功
0x81:	参数设置失败
0x82:	通讯超时
0x83:	卡不存在
0x84:	接收卡数据出错
0x87:	未知的错误
0x85:	输入参数或者输入命令格式错误
0x8f:	输入的指令代码不存在（）
0x8A:	在对于卡块初始化命令中出现错误（仅用于 14443 命令）
0x8B:	在防冲突过程中得到错误的序列号（仅用于 14443 命令）
0x8C	密码认证没通过（仅用于14443命令）
0x90	卡不支持这个命令（仅用于15693命令）
0x91	命令格式有错误（仅用于15693命令）
0x92	在命令的FLAG参数中，不支持OPTION 模式（仅用于15693命令）
0x93	要操作的BLOCK不存在（仅用于15693命令）
0x94	要操作的对象已经别锁定，不能进行修改（仅用于15693命令）
0x95	锁定操作不成功（仅用于15693命令）
0x96	写操作不成功（仅用于 15693 命令）

3 System Commands

5、int SetSerNum(unsigned char *newValue,
unsigned char *buffer);

函数功能：设置 8 个字节的產品序列号。

输入参数:	描述
*newValue	8 个字节的读写器序列号
*buffer	用来返回接收到的数据的指针

输出参数:

*buffer 返回 STATUS 后的状态，如果设置成功，则*buffer=0x 80
如果设置失败，则*buffer 为底层（读卡器单片机）上传的错误代码（参照底层协议代码表 2.2）

返回值:
0x00, 设置成功
0x01, 设置失败

6、int GetSerNum(unsigned char *buffer);

函数功能:

读取由厂家预设的 1 个字节的读卡器地址和 8 个字节序列号.

输入参数:
*buffer 传入一个指针，用来返回接收到的数据

输出参数:

*buffer	buffer[0] 读写器地址
	buffer[1...8] 8 个字节的读写器序列号

返回值:
0x00, 操作成功
0x01, 操作失败

6、 int WriteUserInfo(int num_blk,
int num_length,
char *user_info);

函数功能:

读卡器提供 4 个块（每个块不能大于 120 个字节），共 480 个字节空间的用户数据区。用户可以根据需要，储存相应的用户信息到读写器中。

输入参数:

num_blk 区域号

num_length 数据长度

*user_info 用户数据

返回值:

0x00, 操作成功

0x01, 操作失败

```
7、int ReadUserInfo(int num_blk,  
                     int num_length,  
                     char *user_info);
```

函数功能:

读取读卡器中提供 4 个块（每个块不能大于 120 个字节）的数据。

输入参数:

int num_blk 区域号

int num_length 读取的数据长度

*user_info 待读入的用户数据

输出参数:

*user_info 如果：操作失败，则 user_info[0] 为 错误代码
 如果：操作成功，则 user_info[0..N] 为读取的用户信息数据

返回值:

0x00, 操作成功

0x01, 操作失败

```
8、int GetVersionNum(char *VersionNum);
```

函数功能：

读取读写器的版本号。

输入参数：

*VersionNum 待读入的版本号

输出参数：

*VersionNum 如果：操作失败，则 VersionNum [0] 为 错误代码
 如果：操作成功，则 VersionNum [0..N] 为读入的版本号

返回值：

0x00，操作成功
0x01，操作失败

```
9、int ControlLED( unsigned char freq,  
                   unsigned char duration,  
                   unsigned char *buffer);
```

函数功能： 设置灯的工作状态，包括，灯亮的周期以及循环的次数

输入参数： freq 周期数
duration 次数
*buffer 待返回的参数

输出参数：

*buffer 如果：操作失败，则 buffer [0] 为 错误代码（参考 2.2）
 如果：操作成功，则 buffer [0] 为成功标志，即为 0x80

返回值：

0x00，操作成功，
0x01，操作失败

```
10、int ControlBuzzer(unsigned char freq,  
                      unsigned char duration,  
                      unsigned char *buffer);
```

函数功能： 设置**蜂鸣器**的工作状态，包括，**蜂鸣器**的工作周期以及循环的次数

输入参数：

freq	周期数
duration	次数
*buffer	待返回的参数

输出参数：

*buffer	如果 ： 操作失败，	则 buffer [0] 为 错误代码（参考 2.2）
	如果 ： 操作成功，	则 buffer [0] 为成功标志 ， 即为 0x80

返回值：

0x00，操作成功
0x01，操作失败

4、ISO14443 Type-A Commands

4. 1 Type-A Commands

4.1.1 int MF_Request(
 unsigned char inf_mode,
 unsigned char *buffer);

函数功能： 发送 ISO14443 A 寻卡指令.

输入参数：

inf_mode	寻卡模式
0x01 –Idle	模式（一次只对一张卡操作）
0x00 –All	模式（一次可对多张卡操作）

*buffer	待返回的参数
---------	--------

输出参数：

*buffer	如果 ： 操作失败，	则 buffer [0] 为 错误代码
	如果 ： 操作成功，	则 buffer [0..1]，返回 2 个字节的数据串

返回值:

0x00, 操作成功

0x01, 操作失败

4.1.2 int MF_Anticoll(

```
    unsigned char    *snr,  
    unsigned char    &status);
```

函数功能:

检测卡片数量, 单卡或多卡, 并返回 4 个字节的卡号。(如果有多张卡, 就返回其中一张卡的卡号)

输入参数:

*snr	传送一个指针, 返回 4 个字节的卡号
&Status	传送一个指针, 返回卡片的数量

输出参数:

如果操作成功

Status	检测到的卡片的数量 (0x00 表示检测到单卡, 0x01 表示检测到多卡)
--------	--

*snr	4 个字节的卡号 (snr[0..3])
------	----------------------

如果操作失败

*snr	为 错误代码 (具体参考 2.2)
------	-------------------

返回值:

0x00, 操作成功,

0x01, 操作失败

4.1.3 int MF_Select(

```
    unsigned char    *snr);
```

函数功能:

选择卡, 使卡进入被选择的状态...

输入参数:

*snr	传送一个指针, 传入 4 个字节卡号, 并且返回 4 个字节的卡号
------	-----------------------------------

输出参数:

Status	检测到的卡片的数量
--------	-----------

*snr	4 个字节的卡号 (snr[0..3])
------	----------------------

返回值:

0x00, 操作成功,

0x01, 操作失败

4.1.4 int MF_Halt();

函数功能:

选择卡, 使卡进入被中断的状态...

输入参数:

无

返回值:

0x00, 操作成功,

0x01, 操作失败

4.2 Mifare Appilication Commands

4.2.1 int MF_Read(unsigned char mode,
unsigned char blk_add,
unsigned char num_blk,
unsigned char *snr,
unsigned char *buffer);

函数功能:

在指定位置读取指定长度的数据

输入参数:

mode, 读取模式

(Request Idle + Key A mode=00, Request Idle + Key B mode= 02,
Request All + Key A mode=01, Request All + Key B mode=03)
(以上数字均为十六进制数字)

blk_add, 读取块地址

num_blk, 读取块数目

*snr, 一个指针, 传递的是六个字节的密钥

*buffer 等待接受输出的指针变量

输出参数:

如果操作成功

*snr, 4个字节的卡号

*buffer, 读取到的数据(具体数量为: num_blk*16)

如果操作失败

buffer[0] 错误代码（具体参考 2.2）

返回值:

0x00, 操作成功,

0x01, 操作失败

```
4.2.2 int MF_Write(unsigned char mode,
                    unsigned char blk_add,
                    unsigned char num_blk,
                    unsigned char *snr,
                    unsigned char *buffer);
```

函数功能:

在指定位置写入数据

输入参数:

mode, 要写的模式
(Request Idle + Key A mode=00 , Request Idle + Key B mode= 02,
Request All + Key A mode=01 , Request All + Key B mode=03)
blk_add, 要写块地址
num_blk, 要写块数目
*snr, 待写入的数据
*buffer, 传入的指针符号..用来传出数据

输出参数:

如果操作成功

snr[0..3], 4 个字节的卡号

如果操作失败

buffer[0] 错误代码（具体参考 2.2）

返回值:

0x00, 操作成功,

0x01, 操作失败

```
4.2.3 int MF_InitVal(
                    unsigned char mode,
                    unsigned char SectNum,
                    unsigned char *snr,
                    unsigned char *value);
```

函数功能:

初始化卡

输入参数:

mode, 初始化模式
(Request Idle + Key A mode=00 , Request Idle + Key B mode= 02,
 Request All + Key A mode=01 , Request All + Key B mode=03)
SectNum, 要初始化的扇区号 00-0F
*snr, 6 字节密钥 (以指针的形式传入)
*value 4 字节要初始化的数据

输出参数:

如果操作成功:

snr[0..3], 4 个字节的卡号

如果操作失败:

snr[0], 错误代码 (具体参考 2.2)

返回值:

0x00, 操作成功,

0x01, 操作失败

4.2.4 int MF_Dec(

 unsigned char mode,
 unsigned char SectNum,
 unsigned char *snr,
 int *value);

函数功能:

对卡的指定扇区进行减值操作。

输入参数:

mode, 模式控制
(Request Idle + Key A mode=00 , Request Idle + Key B mode= 02,
 Request All + Key A mode=01 , Request All + Key B mode=03)
SectNum, 要写值的扇区号 00-0F
*snr, 6 字节密钥 (以指针的形式传入)
value 要减的值, 4 个字节长度

输出参数:

如果操作成功

snr[0..3], 4 个字节的卡号

value[0..3] 4 个字节操作后的数据串

如果操作失败

snr[0] 错误代码 (具体参考 2.2)

返回值:

0x00, 操作成功,
0x01, 操作失败

4.2.5 int MF_Inc (

```
        unsigned char    mode,  
        unsigned char    SectNum,  
        unsigned char    *snr,  
        int              *value);
```

函数功能：对卡的指定扇区进行加值操作。

输入参数：

mode, 模式控制
(Request Idle + Key A mode=00 , Request Idle + Key B mode= 02,
 Request All + Key A mode=01 , Request All + Key B mode=03)
SectNum, 要加值的扇区号 00-0F
*snr, 6 字节密钥（以指针的形式传入）
value 要加的值，4 个字节长度

输出参数：

如果操作成功
 snr[0..3], 4 个字节的卡号
 value[0..3] 4 个字节操作后的数据串
如果操作失败
 snr[0] 错误代码（具体参考 2.2）

返回值：

0x00, 操作成功,
0x01, 操作失败

4.2.6 int MF_Getsnr (

```
        unsigned char    mode,  
        unsigned char    halt,  
        unsigned char    *snr  
        unsigned char    *value);
```

函数功能：返回 1 个字节的单卡或多卡标识，4 个字节的卡号。

输入参数：

mode, 模式控制 （模式控制 26 or 52 ）

0x26 –Idle 模式（一次只对一张卡操作）

0x52 –All 模式（一次可对多张卡操作）

halt, 是否需要 halt 卡（halt 选择 00 or 01）

00 不需要执行 halt 指令

01 读写器 执行 halt 指令

snr, 返回的 1 个字节的单卡或多卡标识(如果读卡不成功，返回错误码)

value 返回的 4 个字节的卡号

输出参数：

如果操作成功

snr[0], 1 个字节的单卡或多卡标识

value[0..3] 返回的 4 个字节的卡号

如果操作失败

snr[0] 错误代码（具体参考 2.2）

返回值：

0x00, 操作成功，

0x01, 操作失败

4.2.7 int MF_Restore(

unsigned char mode,
int cardlength,
unsigned char *carddata);

函数功能：

按照选择的模式，进行数据的发送

输入参数：

mode, 模式控制 0x00 一不需要进行 CRC 校验
0x01 一需要进行 CRC 校验

cardlength, 卡数据长度

*carddata, 发送时（卡数据） 接收时（返回数据）

输出参数：

如果操作成功

carddata[0..N], 接收返回数据

如果操作失败

carddata[0] 错误代码（具体参考 2.2）

返回值：

0x00, 操作成功，

0x01, 操作失败

5. ISO14443 Type-B Commands

5.1 int RequestType_B(
 unsigned char *buffer)

函数功能：该命令执行 ISO14443B 中的 REQB 命令，获取卡片的 PUPI 代码

输入参数：

*buffer, 卡片复位后的数据串 (ATQB)

输出参数：

如果操作成功：

*buffer, 卡片复位后的数据串 (ATQB)

 buffer[0] 卡片复位数据的长度

 buffer[1..N] 操作后的数据串 (ATQB)

如果操作失败：

buffer[0] 错误代码（具体参考 2.2）

返回值：

0x00，操作成功，

0x01，操作失败

5.2 int AntiType_B(
 unsigned char *buffer);

函数功能：该命令执行 ISO14443B 中的 AnticollB 命令

输入参数：

*buffer, 卡片返回的数据串 (ATQB)

输出参数：

如果操作成功：

 buffer[0..N], 卡片返回的数据串 (ATQB)

如果操作失败：
buffer[0] 错误代码（具体参考 2.2）

返回值：
0x00，操作成功，
0x01，操作失败

5.3 int SelectType_B (
unsigned char *SerialNum);

函数功能：该命令执行 ISO14443B 中的 ATTRIB 命令，给已知 PUP1 的卡片分配一个识别号 CID

输入参数：
*SerialNum, 卡的序列号

返回值：
0x00，操作成功，
0x01，操作失败

5.4 int Request_AB(
unsigned char * buffer);

函数功能：
该命令执行几集成了 ISO14443B 中的，REQUEST 和 ATTRIB 命令，通过一个命令使卡复位。

输入参数：
* buffer, 返回操作后的卡的序列号 4 个字节

输出参数：
如果操作成功：
buffer[0..3], 返回操作后的卡的序列号 4 个字节
如果操作失败：
buffer[0], 错误代码（具体参考 2.2）

返回值：
0x00，操作成功，

0x01, 操作失败

5.5 int API_ISO14443TypeBTransCOSCcmd(

unsigned char *cmd,
int cmdSize,
unsigned char *buffer);

函数功能:

ISO14443 传送命令, 可以通过此命令向卡发任意有效的命令, 数据

输入参数:

*cmd, 待发送的数据
cmdSize, 数据长度
* buffer, 回收的数据

输出参数:

* buffer, 回收的数据
如果: 操作成功, 则 buffer[0..N] 为从卡返回的数据
操作失败, 则 buffer[0] 为错误代码

返回值:

0x00, 操作成功,
0x01, 操作失败

6 ISO15693 COMMANDS

6.1 ISO15693_Inventory(unsigned char *Cardnumber,
unsigned char *pBuffer);

函数功能:

此命令通过防冲突用于得到读卡区域内所有卡片的序列号 (能得到的卡片数量与模块天线的输出

功率有关, 一般能对 2~6 卡进行防冲突)

输入参数:

*Cardnumber, 返回的卡的数量 (一个字节)
*pBuffer 返回的数据 (包括 FLAG 和 DSFID 和 8*n 个字节的卡号)

输出参数:

如果: 操作成功

*Cardnumber 返回的卡的数量 (一个字节)
*pBuffer 返回的数据 (包括 FLAG 和 DSFID 和 8*n 个字节的卡号)

如果: 操作失败

*Cardnumber 为错误代码

返回值:

0x00, 操作成功,

0x01, 操作失败

```
6.2            int    ISO15693_Read (
                                     unsigned char    flags,
                                     unsigned char    blk_add,
                                     unsigned char    num_blk,
                                     unsigned char    *uid,
                                     unsigned char    *buffer);
```

函数功能:

用来读取 1 个或多个扇区的值, 如果要读每个块的安全位, 将 FLAGS 中 Option_flag 置为 1, 即 FLAG = 0X42, 每个扇区将返回 5 个字节, 包括 1 个表示安全状态字节和 4 个字节的块内容, 这时候每次最多能读 12 个块。如果 FLAG = 02, 将只返回 4 字节的块内容, 这时候每次最多能读 63 个块。

输入参数:

flags 0x02 不带 uid
 0x22 带 uid
 0x42 不带 uid 但是要读安全位
blk_add, 要读的起始块号
num_blk, 块的数量
*uid UID 信息
*buffer 返回值

输出参数:

如果: 返回操作成功

buffer[0] 返回的 flag buffer[1..N] Data
操作失败,
buffer[0]为错误代码

返回值:

0x00, 操作成功,


```

    unsigned char    frags,
    unsigned char    blk_add,
    unsigned char    num_blk,
    unsigned char    *uid,
    unsigned char    *data);

```

flags	0x02 不帶 uid
	0x22 帶 uid
	0x42 不帶 uid 但是要读安全位
blk_add,	要写的起始块号
num_blk,	写的块的数量
*uid	UID 信息
*data	写入的数据

如果：操作失败，则 data[0]为错误代码

0x00, 操作成功,
0x01, 操作失败

```

    unsigned char    flags,
    unsigned char    num_blk,
    unsigned char    *uid
    unsigned char    *buffer);

```

函数功能：用于锁定块内容。注意：此过程不可逆（不能解锁）块锁定后内容不能在修改。

输入参数：

flags	0x02 不带 uid
	0x42 不带 uid 但是要读安全位
	0x22 带 uid
num_blk,	要锁的块号
*uid	UID 信息
*buffer	返回值

输出参数：

如果：操作成功，则 buffer[0] 值为 0x80
如果：操作失败，则 buffer[0]为错误代码

返回值：

0x00，操作成功，
0x01，操作失败

6.5 int ISO15693_StayQuiet(

unsigned char	flags,
unsigned char	*uid,
unsigned char	*buffer);

函数功能：

此命令用于将卡置于静止的状态，必须用地址模式，如果发送的数据与被操作的卡的序列号相同，

操作成功后，卡将进入静止状态，否则状态不变。

输入参数：

flags	标识字节 1 个字节
*uid	UID 信息
*buffer	返回值

输出参数：

如果：操作成功，则 buffer[0] 返回的 0x80，表示操作成功
如果：操作失败，则 buffer[0]为错误代码（具体参考 2.2）

返回值:

0x00, 操作成功

0x01, 操作失败

6.6 int ISO15693_Select(

unsigned char flags,
unsigned char *uid,
unsigned char *buffer);

函数功能:

此命令必须用地址模式，如果发送的数据与被操作的卡的序列号相同，操作成功后，卡将进入被

选择状态，否则状态不变。

输入参数:

flags 标识字节 1 个字节

*uid UID 信息

*buffer 返回值

输出参数:

如果：操作成功，则 buffer[0]为的 0x80，表示操作成功

如果：操作失败，则 buffer[0]为错误代码

返回值:

0x00, 操作成功

0x01, 操作失败

6.7 int ISO15693_ResetToReady(

unsigned char flags,
unsigned char *uid,
unsigned char *buffer);

函数功能:

操作成功后，卡回到 Ready 状态。

输入参数:

flags	标识字节 1 个字节
	0x02 不带 uid
	0x42 不带 uid 但是要读安全位
	0x22 带 uid
*uid	UID 信息
*buffer	返回值

输出参数:

如果: 操作成功, 则 buffer[0] 的值为 0x80, 表示操作成功
如果: 操作失败, 则 buffer[0] 为错误代码 (具体参考 2.2)

返回值:

0x00, 操作成功
0x01, 操作失败

6.8 int ISO15693_WriteAFI(

unsigned char	flags,
unsigned char	afi,
unsigned char	*uid,
unsigned char	*buffer);

函数功能:

对卡进行写 AFI 操作。

输入参数:

flags	标识字节 1 个字节
	0x02 不带 uid
	0x42 不带 uid 但是要读安全位
	0x22 带 uid

afi 待写的 AFI

*uid	UID 信息
*buffer	返回值

输出参数:

如果: 操作成功, 则 buffer[0] 值为 0x80, 表示操作成功
如果: 操作失败, 则 buffer[0] 为错误代码 (具体参考 2.2)

返回值:

0x00, 操作成功

0x01, 操作失败

```
6.9 int ISO15693_LockAFI(unsigned char    flags,  
                        unsigned char    *uid,  
                        unsigned char    *buffer );
```

函数功能:

用于锁定卡的 AFI , 锁定后 AFI 不可以更改

输入参数:

flags	标识字节 1 个字节
0x02	不带 uid
0x42	不带 uid 但是要读安全位
0x22	带 uid

*uid	UID 信息
------	--------

*buffer	返回值
---------	-----

输出参数:

如果: 操作成功, 则 buffer[0] 返回的 0x80, 表示操作成功

如果: 操作失败, 则 buffer[0] 返回 为错误代码

返回值:

0x00, 操作成功

0x01, 操作失败

```
6.10 int ISO15693_WriteDSFID(  
                        unsigned char    flags,  
                        unsigned char    DSFID,  
                        unsigned char    *uid,  
                        unsigned char    *buffer );
```

函数功能:

对卡的进行写 DSFID 操作

输入参数:

flags	标识字节 1 个字节
0x02	不带 uid
0x42	不带 uid 但是要读安全位
0x22	带 uid

DSFID	要写的 DSFID 字节, 长度为 1 个字节
-------	-------------------------

*uid	UID 信息
------	--------

*buffer	返回值
---------	-----

输出参数:

如果: 操作成功, 则 buffer[0] 值为 0x80, 表示操作成功

如果: 操作失败, 则 buffer[0] 值为错误代码 (具体参考 2.2)

返回值:

0x00, 操作成功

0x01, 操作失败

```
6.11 int ISO15693_LockDSFID(  
                                unsigned char flags,  
                                unsigned char *uid,  
                                unsigned char *buffer );
```

函数功能:

用于锁定卡的 DSFID, 锁定后 DSFID 不可以更改

输入参数:

flags	标识字节 1 个字节
0x02	不带 uid
0x42	不带 uid 但是要读安全位
0x22	带 uid

*uid	UID 信息
------	--------

*buffer	返回值
---------	-----

输出参数:

如果: 操作成功, 则 buffer[0] 返回的 0x80, 表示操作成功

如果: 操作失败, 则 buffer[0] 返回 为错误代码

返回值:

0x01, 操作失败

```
6.13 int ISO15693_GetMulSecurity(
                                unsigned char flag,
```

```

unsigned char    blkAddr,
unsigned char    blkNum,
const unsigned char *uid,
unsigned char    *pBuffer);

```

函数功能： 用于获取卡的各个块的安全状态位的数据

输入参数：

flag	0x02 不带 uid
	0x22 带 uid
	0x42 不带 uid 但是要读安全位
blkAddr,	要读的起始块号
blkNum,	读的块的数量
*uid	UID 信息
*pBuffer	返回值

输出参数：

如果：操作成功

*pBuffer	返回的数据
pBuffer [0]	返回的 flags
pBuffer [1..N]	Block security status (块的安全状态)

如果：操作失败，则 pBuffer[0]为错误代码（具体参考 2.2）

返回值：

0x00,	操作成功
0x01,	操作失败

6.14 int ISO15693_TransCOSCcmd(

```

unsigned char    *cmd,
int              cmdSize,
unsigned char    *buffer);

```

函数功能： 一个通用命令，用户可以通过此命令，对卡进行各种操作。

输入参数：

*cmd,	需要发送的数据
cmdSize,	数据长度
*buffer	返回值

输出参数:

如果: 操作成功

`*buffer` 返回的数据

`buffer [0..N]` 从卡返回的数据

如果: 操作失败, 则 `buffer[0]`为错误代码

返回值:

`0x00`, 操作成功

`0x01`, 操作失败