1. 数据格式

1. 1 数据格式

数据格式（起始位，数据位，校验位，停止位）可以根据通讯的需要由软件设置，下面是设备支持的数据格式：

参数	描述
波特率	可选: 9600, 19200, 38400, 57600, 1152000
数据位	固定: 8 bits
起始位	固定: 1 Bits
停止位	固定: 1 bit.
校验位	可选: Odd, Even, None

下面是默认设置:

波特率	数据位	起始位	停止位	校验位
9600	8	1	1	None

1. 2 数据包格式

数据包格式，命令包是由主机发送到读写器，返回包是由读写器返回主机。

命令包格式 (主机到读写器)：

STX	STATION ID	DATA LENGTH	CMD	DATA [0..N]	BCC	ETX

(BCC) = STATION ID ⊕DATALENGTH⊕ CMD⊕ DATA [0] ⊕ … ⊕ DATA [n], where ⊕ is the “EOR”.

返回包格式 (读写器到主机)

STX	STATION ID	DATA LENGTH	STATUS	DATA[0..N]	BCC	ETX

(BCC) = STATION ID ⊕DATA LENGTH⊕ STATUS⊕ DATA [0] ⊕ … ⊕ DATA [n], where ⊕ is the “EOR”.

数据包中字节描述:

字段	长度	描述	备注
STX	1	0x02 – ‘起始字节’ – 标准控制字节. 表示一个数据包的开始	

STATION ID	1	设备地址，在多机通讯所必需，读写器在收到数据包后判断包内的地址与自身预设地址是否相符，相符才会响应。	地址 0x00 是一个在单机模式下使用的特殊地址。读写器会响应任何带 0 地址的数据包(不进行地址判断).
DATALENGTH	1	数据包中数据字节的长度.包括 CMD/STATUS 和 DATA field,但不包括 BCC. LENGTH= 字节数 (CMD/STATUS + DATA[0.. N])	
CMD	1	命令字:由一个命令字节组成.	可以参照命令表 该字节只在发送包中使用
STATUS	1	返回状态字节: 由读写器返回主机的状态	该字节只在返回包中使用
DATA [0-N]	0–255	这是一个长度与命令字有关的数据流。也有部分命令不需要附加数据.	
BCC	1	8bits 的校验字节.它包括除 STX, ETX 外所有字节的异或校验.	
ETX	1	0x03:'终止字节' – 标准控制字节，表示数据包的结束.	

COMMANDS（命令）

命　　令　　表		
命 令 字	名称	描述
ISO14443 TYPE A Commands (0x03~0x06)		
0x03	REQA	ISO14443 -A 寻卡请求
0x04	Anticoll A	防冲突
0x05	Select A	选定卡
0x06	Halt A	使卡进入 HAIT 状态
ISO14443B TYPEB Command ((0x09~0x0C)		
0x09	ReqB	ISO14443B 寻卡命令
0x0A	AnticollB	IS14443-B 防冲突命令
0x0B	Attrib_TypeB	ISO14443B ATTRIB 命令
0x0C	Rst_ TypeB	集成了寻卡和 ATTRIB 命令通过此命令直接对卡进行复位
0x0D	ISO14443_TypeB_Transfer_Command	ISO14443B 传送命令，可以通过此命令向卡发任意有效的命令，数据
Mifare Application Commands (0x20~0x2F)		
0x20	MF_ Read	集成寻卡，防冲突，选卡，验证密码，读卡等操作，一个命令完成读卡操作。
0x21	MF_ Write	集成寻卡，防冲突，选卡，验证密码，写卡等操作，一个命令完成写卡操作。
0x22	MF_ InitVal	集成寻卡，防冲突，选卡，验证密码等操作，一个命令完成块值初始化操作。

0x23	MF_Decrement	集成了寻卡，防冲突，选卡，验证密码，块值减操作，一个命令完成块减值操作。
0x24	MF_Increment	集成了寻卡，防冲突，选卡，验证密码，块值加等操作，一个命令完成块值加操作。
0x25	MF_GET_SNR	集成了寻卡，防冲突，选卡等操作，一个命令完成读取卡片序列号的操作
0x28	ISO14443_TypeA_Transfer_Command	ISO14443 TypeA 通用命令，可以根据 ISO14443 TypeA 向卡发任何数据
ISO15693 Commands (0x10~0x1D)		
0x10	ISO15693_Inventory	寻卡，防冲突
0x11	ISO15693_Read	读卡操作
0x12	ISO15693_Write	写卡操作
0x13	ISO15693_Lockblock	锁定卡扇区内容操作
0x14	ISO15693_StayQuiet	将卡至于静止状态
0x15	ISO15693_Select	选择卡
0x16	ISO15693_Resetready	使卡进入 READY 状态
0x17	ISO15693_Write_Afi	写 AFI
0x18	ISO15693_Lock_Afi	锁定 AFI
0x19	ISO15693_Write_Dsfid	写 DSFID
0x1A	ISO15693_Lock_Dsfid	锁定 DSFID
0x1B	ISO15693_Get_Information	获取卡信息
0x1C	ISO15693_Get_Multiple_Block_Security	获取块安全信息
0x1D	ISO15693_Transfer_Command	可以通过此命令向卡片发任何数据和命令

系统命令 (0x80~0xFF)		
0x80	SetAddress	设置读写器地址
0x81	SetBaudrate	设置通讯波特率
0x82	SetSerlNum	设置读写器的序列号
0x83	GetSerlNum	读取读写器的序列号
0x84	Write_UserInfo	设置用户数据信息
0x85	Read_UserInfo	读取用户数据信息
0x86	Get_VersionNum	用来读取模块的版本信息
0x87	Control_Led1	控制 led1 的工作方式（只有带有两个 LED 口的模块，支持此命令）
0x88	Control_Led2	控制 led2 的工作方式
0x89	Control_Buzzer	控制 buzzer 的工作方式

3 System Commands

3.1 SetAddress (0x80)

发送数据:

DATA[0]: 要设置的新地址 ,十六进制表示。

正确返回:

STATUS: 0x00 – OK

DATA[0] 设置的地址

错误返回:

STATUS: 0x01 –FAIL

DATA[0] 参考错误代码表

描述: 为读写器设置新的地址，读写器返回设置好的地址.

比如:

发送命令： **02 00 02 80 02 80 03**

回执数据： **02 00 02 00 02 00 03**

3.2 SetBaudrate (0x81)

发送数据:

DATA[0] 波特率

0x00 – 9600 bps

0x01 – 19200 bps

0x02 – 38400 bps

0x03 – 57600 bps

0x04 – 115200 bps

> 0x04—9600 bps

正确返回：

STATUS:　0x00 – OK

DATA[0] 设置的波特率代码.

错误返回：

STATUS:　0x01 –FAIL

DATA[0]　　参考错误代码表

描述 ：　　设置读写器与主机通讯的波特率. 这个波特率将被保存到 EEPROM 内并作为新的默认波特率.设置好新的波特率后，系统开始使用新的波特率，而不需要复位。

比如:

发送命令：　02　00　02 81 01 82 03

回执数据：　02　00　02 00 01 03 03　（设置波特率为19200,N,8,1）

3.3 SetSerNum (0x82)

发送数据:

DATA[0..7]: 8 个字节的读写器序列号

正确返回：

STATUS:　0x00 – OK

DATA[0] 0x80(表示操作成功)

错误返回：

STATUS:　0x01 –FAIL

DATA[0]　　参考错误代码表

描述:　　设置 8 个字节的序列号。

比如:

发送命令： **02 00 09 82 AA BB AA BB AA BB AA BB 8B 03**

回执数据： **0200 02 00 80 82 03**

3.4 GetserNum (0x83)

发送数据: N/A

正确返回:

STATUS: 0x00 – OK

DATA[0]: 读写器地址

DATA[1..8]: 8 个字节的读写器序列号

错误返回:

STATUS: 0x01 –FAIL

DATA[0] 参考错误代码表

描述: 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.

比如:

发送命令： **02 00 01 83 82 03**

回执数据：**02 00 0A 00 00 AA BB AA BB AA BB AA BB 0A 03**

其中橙色的“00”表示模块当前的地址，其后八个字节表示读卡器的序列号

3.5 Write_UserInfo (0x84)

发送数据:

DATA[0]: 对读写器进行写数据操作的区域号

0x00: 对读写器的 区域 0 进行写操作

0x01: 对读写器的区域 1 进行写操作

0x02: 对读写器的区域 2 进行写操作

0x03: 对读写器的区域 3 进行写操

DATA[1] 要写入的数据的长度，不能大于 120 字节(以 16 字节形式表示，比如要写 120 个字节，那么 DATA[1] = 0x78)

DATA[2..121] 要写入的数据信息.

正确返回：

STATUS: 0x00 – OK

DATA[0]: 0x80(表示操作成功)

错误返回：

STATUS: 0x01 –FAIL

DATA[0] 参考错误代码表

描述: 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.

比如:

发送命令：02 00 7B 84 01 78 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 86 03

回执数据：02 02 02 00 80 80 03

3.6 Read_UserInfo (0x85)

发送数据:

DATA[0]: 对读写器进行读数据操作的区域号

0x00: 对读写器的 区域 0 进行读操作

0x01: 对读写器的区域 1 进行读操作

0x02: 对读写器的区域 2 进行读操作

0x03: 对读写器的区域 3 进行读操作

DATA[1] 要读出的数据的长度，不能大于 120 字节(以 16 字节形式表示，比如要读 120 个字节，那么 DATA[1] = 0x78)

正确返回：

STATUS: 0x00 – OK

DATA{1..N} 读出的用户信息 < 120 byte

错误返回：

STATUS: 0x01 –FAIL

DATA[0] 参考错误代码表

描述: 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

比如:

发送命令： **02 00 03 85 01 78 FF 03**

回执数据： **02 00 79 00 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55**
AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55
AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55
AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55
AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55
AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55

AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 79 03

3.7 Get_VersionNum (0x86)

发送数据： 无

正确返回：

STATUS: 0x00 – OK

DATA[1..N] 版本号

错误返回：

STATUS: 0x01 –FAIL

DATA[0] 参考错误代码表

描述: 读取读写器的版本号

比如:

发送命令： 02 00 01 86 87 03

回执数据： 02 00 11 00 52 44 4D 35 30 30 5F 30 34 30 37 5F 31 30 30 30 7D 03

3.8 Control_Led1 (0x87)

发送数据：

DATA [0]: 在一次循环中灯亮的周期数（一个周期为 20ms,所以 DATA[0]最大为 50）

DATA [1]: LED 状态循环的次数(一个循环一秒）

正确返回：

STATUS: 0x00 – OK

DATA[0]: 0x80(表示操作成功)

错误返回：

STATUS: 0x01 –FAIL

DATA[0] 参考错误代码表

描述： 此命令用来控制 **LED1** 的运行状态（只有当模块或者读头，带有 **2** 个 **LED** 口时，才支持此命令，如果只带有一个 **LED** 口，只支持 **Control_Led2** 命令）

比如:

发送命令： 02 00 03 87 18 0A 96 03

回执数据： 02 00 02 00 80 82 03

其中 棕色的“18”，表示一次循环中灯亮的时间为 20ms * 24(0x18) = 480ms

绿色的“0A”，表示共运行 10 次这样的循环。

3.9 Control_Led2 (0x88)

发送数据：

DATA [0]: 在一次循环中灯亮的周期数(一个周期为 20ms ,所以 DATA[0] 最大为 50)

DATA [1]: LED 状态循环的次数(一个循环一秒)

正确返回：

STATUS: 0x00 – OK

DATA[0]: 0x80(表示操作成功)

错误返回：

STATUS: 0x01 –FAIL

DATA[0] 参考错误代码表

描述： 此命令用来控制 LED2 的运行状态

比如:

发送命令： 02 00 03 88 18 0A 99 03

回执数据： 02 00 02 00 80 82 03

其中 棕色的“18”，表示一次循环中灯亮的时间为 20ms * 24(0x18) = 480ms

绿色的“0A”，表示共运行 10 次这样的循环。

3.10 Control Buzzer (0x89)

发送数据：

DATA [0]: **在一次循环中，蜂鸣器鸣叫的周期数(一个周期为 20ms，所以 DATA[0]最大为 50)**

DATA [1]: **蜂鸣器状态循环的次数(一个循环一秒)**

正确返回：

STATUS: 0x00 – OK

DATA[0]: 0x80(表示操作成功)

错误返回：

STATUS: 0x01 –FAIL

DATA[0] 参考错误代码表

描述： 此命令用来控制蜂鸣器的运行状态

比如：

发送命令： 02 00 03 89 18 0A 98 03

回执数据： 02 00 02 00 80 82 03

其中棕色的“18”，表示一次循环中蜂鸣器**鸣叫**的时间为 20ms * 24(0x18) = 480ms

绿色的“0A”，表示共运行 10 次这样的循环。

4. ISO14443 Type-A Commands

4. 1 Type-A Commands

4.1.1 REQA (0x03)

发送数据：

DATA[0]: 寻卡模式

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）

正确返回：

STATUS: 0x00 – OK

DATA[0..1]: 2 字节的卡类型.0x0004 为 M1 卡

错误返回：

STATUS: 0x01 –FAIL

DATA[0] 参考错误代码表

描述: 发送 ISO14443 A 寻卡指令.

比如:

发送命令 ： 02 00 02 03 **26** 27 03

回执数据： 02 00 03 00 04 00 07 03

4.1.2 AnticollA (0x04)

发送数据: 无

正确返回：

STATUS: 0x00 – OK

DATA[0]: 单卡多卡标志.

0x00 – 检测到一张卡.

0x01 – 检测到多张卡.

DATA[1..4]: UID – 卡芯片号

错误返回：

STATUS: 0x01 –FAIL

DATA[0] 参考错误代码表

描述: 发送 ISO14443 A 防冲突指令.

比如:

发送命令 ： **02 00 01 04 05 03**（返回的数据由卡上的信息而定，不同卡的数据可能不同）

回执数据： **02 00 06 00 00 06 61 62 AE AD 03(放一张卡返回数据)**

回执数据： **02 00 06 00 01 86 69 F3 7F 64 03** （放多张卡返回数据，卡号为其中一张卡的卡号）

4.1.3 SelectA (0x05)

发送数据

DATA[0..3]: UID – 要选择的卡的卡芯片号

正确返回

STATUS: 0x00 – OK

DATA[0..3]: UID – 卡芯片号

错误返回：

STATUS: 0x01 –FAIL

DATA[0] 参考错误代码表

描述: 发送 ISO14443 A 选择卡指令.

比如:

发送命令 ：02 00 05 05 **86 69 F3 7F** 63 03（红色部分为卡号）

回执数据： 02 00 05 00 86 69 F3 7F 66 03

4 .1.4 HaltA (0x06)

发送数据： N/A-

正确返回：

STATUS: 0x00 – OK

DATA[0]: 0x80(表示操作成功)

错误返回：

STATUS: 0x01 –FAIL

DATA[0] 参考错误代码表

描述: 发送 ISO14443 A 将选择的卡置入 HALT 状态的指令.

比如：

发送命令： 02 00 01 06 07 03

回执数据： 02 00 02 00 80 82 03

4.2 Mifare Appilication Commands

4.2.1 MF_ Read (0x20)

发送数据：

DATA [0]: 读取模式控制

Bit0: Request Mode. 0=Request Idle, 1 = Request All

Bit1: Request Mode. 0=对 KEYA 进行校验，1 =对 KeyB 进行校验

DATA[1]: 要读的块数长度值，即读多少块。取值范围 01-04

DATA[2]: 要读的块的起点地址。Mifare s50 取值范围：十六进制 00-3F 即 0 块到 63 块。

DATA[3-8]: 6 个字节的密钥

正确返回：

STATUS: 0x00 – OK

DATA [0-3]: 4 字节卡序列号 (从低到高)

DATA [4..N] 从卡上返回的数据

错误返回：

STATUS: 0x01 –FAIL

DATA[0]

描述: 发送 ISO14443 A 读卡指令

比如：.

发送命令 ： **02 00 0A 20 01 01 10 ff ff ff ff ff ff 3A 03**（读块第 16 块的内容，返回的数据由卡上的信息而定，不同卡的数据可能不同）

其中红色的**“01”**表示用**“Request all”**形式寻卡，以 **KEYA** 进行密码校验。

其中绿色的**“01”**表示读一个块的内容。

其中紫色的**“10”**表示要读的块起点地址为 **16**（**0x10**）。

其中棕色的**“ff ff ff ff ff ff”**表示输入的 **6** 个字节的密钥。

回执数据： **02 00 15 00 06 61 62 AE FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF BE 03**

发送命令 ：**02 00 0A 20 01 04 10 ff ff ff ff ff ff 3F 03**（读取 **16~19** 块的内容，返回的数据由卡上的信息而定，不同卡的数据可能不同）

回执数据： **02 00 45 00 16 0F F4 7F**

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 FF 07 80 69 FF FF FF FF FF FF

C6 03

发送命令 ：**02 00 0A 20 01 04 3C ff ff ff ff ff ff 13 03**（读取 **60~63** 块的内容，返回的数据由卡上的信息而定，不同卡的数据可能不同）

回执数据：**02 00 45 00 16 0F F4 7F**

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 FF 07 80 BC FF FF FF FF FF FF

13 03

4.2.2 MF_ Write (0x21)

发送数据：

DATA [0]: 写操作模式控制

Bit0: Request Mode. 0=Request Idle, 1 = Request All

Bit1: Request Mode. 0=对 KEYA 进行校验，1 =对 KeyB 进行校验

DATA[1]: 要写的块数长度值，即读多少块。取值范围 01-04

DATA[2]: 要写的块的起点地址。取值范围：十六进制 00-3F 即 00 块到 63 块。

DATA[3-8]: 6 字节的密钥

DATA [9-N]: 要写入的数据.

正确返回：

STATUS: 0x00 – OK

DATA [0-3]: 卡的序列号 (LL LH HL HH)

错误返回：

STATUS: 0x01 –FAIL

DATA[0] 参考错误代码表

描述：　　ISO14443 写卡指令

比如：

发送命令 ：**02　00 1A 21 01 01 10 ff ff ff ff ff ff　FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 11 11 2B 03**（对 16 块进行写操作）

其中红色的**“01”**表示用**“Request all”**形式寻卡，以 **KEYA** 进行密码校验。

其中绿色的**“01”**表示读一个块的内容。

其中紫色的**“10”**表示要读的块起点地址为 **16**（**0x10**）。

其中棕色的**“ff ff ff ff ff ff”**表示输入的 **6** 个字节的密钥。

其中灰色的**“FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 11 11“**表示要写的 **16** 个字节的内容。

回执数据：02 00 05 00 CE 86 AE 67 84 03

4.2.3 MF_InitVal (0x22)

发送数据：

DATA [0]:　初始化模式控制

Bit0: Request Mode. 0=Request Idle, 1 = Request All

Bit1: Request Mode. 0=对 KEYA 进行校验, 1 =对 KeyB 进行校验

DATA [1]:　要初始化的扇区号 00-0F

Block0 –开放给用户使用

Block1 –数据存储

Block2 –数据备份

DATA[2-7]: 6 字节密钥

DATA [8-11]: 4 字节的要初始化的数据(数值格式: 由低到高)

正确返回：

STATUS: 0x00 – OK

DATA [0-3]: 卡的序列号 (LL LH HL HH)

错误返回：

STATUS: 0x01 –FAIL

DATA[0] 参考错误代码表

描述： ISO14443 扇区初始化命令

比如:

发送命令 ：**02 00 0D 22 01 04 ff ff ff ff ff ff 64 00 00 00 4E 03**（对扇区 **4** 进行初始化）

其中红色的**“01”**表示用**“Request all”**形式寻卡，以 **KEYA** 进行密码校验。

其中绿色的**“04”**表示要初始化的扇区号。

其中紫色的**“ff ff ff ff ff ff”**表示输入的 **6** 个字节的密钥。

其中棕色的**“64 00 00 00”**表示要初始化的值。

回执数据： **02 00 05 00 16 0F F4 7F 97 03**

4.2.4 MF_Decrement (0x23)

发送数据：

DATA [0]: 模式控制 ：

Bit0: Request Mode. 0=Request Idle, 1 = Request All

Bit1: Request Mode. 0=对 KEYA 进行校验，1 =对 KeyB 进行校验

DATA[1]: 保存数据的区号.

DATA[2-7]: 6 个字节的密钥

DATA[8-11]: **要减的值** (数据格式 ： 从低到高)

正确返回：

STATUS: 0x00 – OK

DATA[0-3]: 卡芯片号(数据格式 ： 从低到高)

DATA[4-7]: 减完后的值 (数据格式 ： 从低到高)

错误返回：

STATUS: 0x01 –FAIL

DATA[0] 参考错误代码表

描述： ISO14443 电子钱包减值命令

比如：

发送命令 ： **02 00 0d 23 01 04 ff ff ff ff ff ff 01 00 00 00 2A 03**

其中红色的“01”表示用“Request all”形式寻卡，以 KEYA 进行密码校验

其中绿色的“04”表示要操作的扇区号

其中紫色的“ff ff ff ff ff ff”表示输入的 6 个字节的密钥

其中棕色的“01 00 00 00”表示要减去的值

回执数据： **02 00 09 00 16 0F F4 7F 63 00 00 00 F8 03**

其中紫色的“16 0F F4 7F”表示返回的 4 个字节的卡号

其中棕色的“63 00 00 00”表示剩余的值

4.2.5 MF_Increment (0x24)

发送数据：

DATA [0]: 模式控制：

Bit0: Request Mode. 0=Request Idle, 1 = Request All

Bit1: Request Mode. 0=对 KEYA 进行校验，1 =对 KeyB 进行校验

DATA[1]: 保存数据的区号.

DATA[2-7]: 6 个字节的密钥

DATA [8-11]: **要增加的值**. (数据格式 ： 从低到高)

正确返回：

STATUS: 0x00 – OK

DATA[0-3]: 卡芯片号(数据格式 ： 从低到高)

DATA[4-7]: **增加后的值** (数据格式 ： 从低到高)

错误返回：

STATUS: 0x01 –FAIL

DATA[0] 错误代码

描述： ISO14443 电子钱包增值命令

比如：

发送命令 ： **02 00 0d 24 01 04 ff ff ff ff ff ff 01 00 00 00 2D 03**

其中红色的**“01”**表示用**“Request all”**形式寻卡，以 **KEYA** 进行密码校验

其中绿色的**“04”**表示要操作的扇区号

其中紫色的**“ff ff ff ff ff ff”**表示输入的 **6** 个字节的密钥

其中棕色的“01 00 00 00”表示要增加的值

回执数据： **02 00 09 00 16 0F F4 7F 63 00 00 00 F8 03**

其中紫色的“16 0F F4 7F”表示返回的 4 个字节的卡号

其中棕色的“63 00 00 00”表示要剩余的值

4.2.6 MF_GET_SNR (0x25)

发送数据：

DATA[0]: 寻卡模式

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）

DATA[1]: 00 不需要执行 halt 指令

01 读写器 执行 halt 指令

正确返回：

STATUS: 0x00 – OK

DATA[0]: 单卡多卡标志.

0x00 – 检测到一张卡.

0x01 – 检测到多张卡.

DATA[1-4]: 卡芯片号(LL LH HL HH)

错误返回：

STATUS: 0x01 –FAIL

DATA[0] 参考错误代码表

描述： 高级寻卡命令

比如：

发送命令 ： **02 00 03 25 26 00 00 03**

回执数据： **02 02 06 00 01 16 0F F4 7F 97 03**

其中橙色的“01”表示读卡区域内只有一张有效卡

其中紫色的“16 0F F4 7F”表示卡的 序列号

4.2.7 ISO14443_TypeA_Transfer_Command (0x28)

发送数据：

DATA[0]: CRC 校验模式模式（参考 ISO14443 协议）

0x00 —不需要进行 CRC 校验

0x01 —需要进行 CRC 校验

DATA[1]: 要对卡发送命令的长度

DATA [2…N]: 向卡发送的数据

正确返回：

STATUS: 0x00 – OK

DATA[0-N]: 从卡返回的数据

错误返回：

STATUS: 0x01 –FAIL

DATA[0] 参考错误代码表

比如：

发送命令： 02 00 04 28 00 01 26 0b 03（相当于发 requesta 命令）

回持数据： 02 00 03 00 04 00 07 03

5.ISO14443 Type-B Commands

5.1 ReqB (0x09)

发送数据： 无

正确返回：

STATUS: 0x00 – OK

DATA[0]: 卡片复位数据的长度

DATA [1..N] ATQB

错误返回：

STATUS: 0x01 –FAIL

DATA[0] 参考错误代码表

描述： 该命令执行 ISO14443B 中的 REQB 命令，获取卡片的 PUPI 代码

比如：

发送命令： 02 00 01 09 08 03

返回数据： 02 00 0E 00 0C 50 41 30 0A 10 41 F5 A3 44 00 71 85 9E 03

5.2 AnticollB(0x0A)

发送数据： 无

正确返回：

STATUS: 0x00 – OK

DATA[0]: 卡片复位数据的长度

DATA [1..N] 从卡片上返回的数据信息

错误返回：

STATUS: 0x01 –FAIL

DATA[0] 参考错误代码表

描述： 该命令执行 ISO14443B 中的 AnticollB 命令

比如： **发送命令**： 02 00 01 0A 0B 03

返回数据： 02 00 0B 00 0C 50 41 11 0A 02 41 F5 A3 44 5C 03

5.3 Attrib_TypeB (0x0B)

发送数据：

DATA[0~3]: 卡的四字节的序列号

正确返回：

STATUS: 0x00 - OK

Data Field

DATA[0] 0X80 （复位成功）

错误返回：

STATUS: 0x01 –FAIL

DATA[0] 参考错误代码表

描述：该命令执行 ISO14443B 中的 ATTRIB 命令，给已知 PUPI 的卡片分配一个识别号 CID

比如： **发送命令：** 02 00 05 0B 41 30 0A 10 65 03

数据回执： 02 00 02 00 80 82 03

5.4 Rst_ TypeB (0x0C)

发送数据： 无

正确返回：

STATUS: 0x00 - OK

Data Field

DATA[0.3] 卡的序列号

错误返回：

STATUS: 0x01 –FAIL

DATA[0] 参考错误代码表

描述： 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。

比如： 发送命令：02 00 01 0c 0d 03

返回数据：02 00 05 00 41 30 0A 10 6E 03

5.6 ISO14443_TypeB_Transfer_Command (0x0D)

发送数据：

DATA[1]: 要对卡发送命令的长度

DATA [2…N]: 向卡发送的数据

正确返回：

STATUS: 0x00 – OK

DATA[0-N]: 从卡返回的数据

错误返回：

STATUS: 0x01 –FAIL

DATA[0]　　参考错误代码表

比如：

发送命令： 02 00 0a 0d 08 00 00 05 00 84 00 00 08 86 03（取随机数）

回持数据： 02 00 0D 00 0A 00 69 60 B3 AE C8 2A 8A 7E 90 00 95 03

发送命令： 02 00 0c 0d 0a 00 00 07 00 a4 00 00 02 3f 00 95 03（选择主文件）

回持数据： 02 00 17 00 0B 00 6F 10 84 0E 31 50 41 59 2E 53 59 53 2E 44 44 46 30 31 90 00 1E 03

6 ISO15693 COMMANDS

6.1 SO15693_Inventory (0x10)

发送数据：

DATA [0]: Flags

DATA [1]: Afi

DATA [2]: Masklengh（这个数据的值一般为 0）

DATA [3..10]: Maskvalue（这个数据的值一般为 0）

正确返回：

STATUS: 0x00 - OK

Data[0] : 读卡区域内存在的卡的数量

Data[1..N] : 已经读到的卡的 UID 和其他信息，参见 ISO15693 协议。

错误返回：

STATUS: 0x01 –FAIL

DATA[0] 参考错误代码表

描述: 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和卡片的性能有关，一般能对 2~6 卡进行防冲突）

比如：

发送命令： 02 00 04 10 06 00 00 12 03

有一张卡返回： 02 00 0C 00 01 00 01 4A 80 E9 11 00 00 07 E0 D9 03

其中红色的 01 表示有一张卡被读到，蓝色的 00 01 分别为 FLAG 和 DSFID，紫色的 4A 80 E9 11 00 00 07 E0 为卡片的序列号。

有两张卡返回： 02 00 16 00 02 00 08 47 80 E9 11 00 00 07 E0 00 01 4A 80 E9 11 00 00 07 E0 10 03

有三张卡返回： 02 00 20 00 03 00 08 47 80 E9 11 00 00 07 E0 00 01 4A 80 E9 11 00 00 07 E0 00 00 3B 80 E9 11 00 00 07 E0 83 03

有四张卡返回： 02 00 2A 00 04 00 08 47 80 E9 11 00 00 07 E0 00 01 4A 80 E9 11 00 00 07 E0 00 00 3B 80 E9 11 00 00 07 E0 00 08 3E 80 E9 11 00 00 07 E0 27 03

6.2 ISO15693_Read (0x11)

发送数据：

DATA [0]: Flags

DATA [1] 要读的起始块

DATA [2] 要读块的数量

DATA[3..10] UID(如果你使用地址模式，需输入 8 位 UID)

正确返回：

STATUS: 0x00 - OK

Data Field

DATA [0] Flags

DATA [1..N] DATA

错误返回：

STATUS: 0x01 –FAIL

DATA[0] 参考错误代码表

描述： 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。

比如：

发送命令： 02 00 04 11 02 01 05 13 03（从卡的第 1 个块读到第 5 个块）

02 00 0C 11 22 01 05 3E 80 E9 11 00 00 07 E0 9A 03

正确返回： 02 00 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 16 03

当 Option_flag 标志位为 1，即发送中的命令参数 Flag 为 0X42

命令： 02 00 04 11 42 01 01 57 03（读第 1 个块的内容）

返回： 02 00 07 00 00 00 00 00 00 07 03 （其中红色的 0 表示这个块的安全状态，即没有锁定，如果锁定，值应该为 1））

6.3 ISO15693_Write (0x12)

发送数据：

DATA [0]: Flags

DATA [1] first block number

DATA [2] Number of blocks

地址模式：

DATA[3..10]　　UID(如果你使用地址模式，需输入 8 位 UID)

DATA[11..N]　需要写入的数据

非地址模式：

DATA[3..N]　　需要写入的数据

操作成功返回：

STATUS:　0x00 – OK

DATA[0]:　　0x80(表示操作成功)

操作错误返回：

STATUS:　　0x01 –FAIL

DATA[0]　　参考错误代码表

描述：　　对一个块进行写操作（每次只能写一个块）

比如：

命令：　　02 00 08 12 42 05 01 11 11 11 11 5c 03(对卡的第 5 块写入 11 11 11 11)

操作成功：　　02 00 02 00 80 82 03

6.4 ISO15693_Lock_Block (0x13)

发送数据：

DATA [0]:　Flags

DATA [1]:　　Block number

DATA[2..9]　　UID(如果你使用地址模式，需输入 8 位 UID)

操作成功返回：

STATUS: 0x00 – OK

DATA[0]: 0x80(表示操作成功)

操作失败返回：

STATUS: 0x01 –FAIL

DATA[0] 参考错误代码表

描述：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。

比如：

发送命令： 02 00 03 13 42 05 57 03

正确返回： 02 00 02 00 80 82 03

6.5 ISO15693_Stay_Quiet (0x14)

发送数据：

DATA [0]: Flags

DATA [1..8]: UID

操作成功返回：

STATUS: 0x00 – OK

DATA[0]: 0x80(表示操作成功)

操作失败返回：

STATUS: 0x01 –FAIL

DATA[0] 参考错误代码表

描述： 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。

比如：

发送命令： **02 00 0a 14 22 3E 80 E9 11 00 00 07 E0 9d 03**

正确返回： **02 00 02 00 80 82 03**

6.6 ISO15693_Select (0x15)

发送数据：

DATA [0]: Flag

DATA [1..8] UID

操作成功返回：

STATUS: 0x00 – OK

DATA[0]: 0x80(表示操作成功)

操作失败返回：

STATUS: 0x01 –FAIL

DATA[0 参考错误代码表

描述： 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。

比如：

发送命令： 02 00 0a 15 22 3E 80 E9 11 00 00 07 E0 9c 03

正确返回： 02 00 02 00 80 82 03

6.7 ISO15693_Reset_To_Ready (0x16)

发送数据：

DATA [0]: Flags

DATA [1..8] UID(如果你使用地址模式，需输入 8 位 UID)

操作成功返回：

STATUS: 0x00 – OK

DATA[0]: 0x80(表示操作成功)

操作失败返回：

STATUS: 0x01 –FAIL

DATA[0] 参考错误代码表

描述： 操作成功后，卡回到Ready状态。

比如：

发送命令： 02 00 0A 16 22 3E 80 E9 11 00 00 07 E0 9F 03 （地址模式）

发送命令： 02 00 02 16 02 16 03(非地址模式)

正确返回： 02 00 02 00 80 82 03

6.8 ISO15693_Write_AFI(0x17)

发送数据：

DATA [0]: Flags

DATA [1]: AFI

DATA[2..9] UID(如果你使用地址模式，需输入 8 位 UID)

操作成功返回：

STATUS: 0x00 – OK

DATA[0]: 0x80(表示操作成功)

操作失败返回：

STATUS: 0x01 –FAIL

DATA[0] 参考错误代码表

描述： **对卡的进行写AFI操作。**

比如：

发送命令： **02 00 03 17 42 06 50 03**

正确返回： **02 00 02 00 80 82 03**

6.9 ISO15693_Lock_AFI(0x18)

发送数据：

DATA [0]: Flags

DATA[1..8] UID(如果你使用地址模式，需输入 8 位 UID)

操作成功返回：

STATUS: 0x00 – OK

DATA[0]: 0x80(表示操作成功)

操作失败返回：

STATUS: 0x01 –FAIL

DATA[0] 参考错误代码表

描述： 用于锁定卡的AFI，锁定后AFI不可以更改

比如：

发送命令：　　**02 00 02 18 42 58 03**

正确返回：　　**02 00 02 00 80 82 03**

6.10 ISO15693_Write_DSFID(0x19)

发送数据：

DATA [0]:　　Flags

DATA [1]:　　DSFID

DATA[2..9]　　UID(如果你使用地址模式，需输入 8 位 UID)

操作成功返回：

STATUS:　0x00 – OK

DATA[0]:　　0x80(表示操作成功)

操作失败返回：

STATUS:　　0x01 –FAIL

DATA[0]　　参考错误代码表

描述：　　对卡的进行写DSFID操作

比如：

发送命令：　　**02 00 03 19 42 08 50 03**

正确返回：　　**02 00 02 00 80 82 03**

6.11 ISO15693_Lock_DSFID(0x1A)

发送数据：

DATA[0]:　　Flags

DATA[1..8]　　UID(如果你使用地址模式，需输入 8 位 UID)

操作成功返回：

STATUS:　0x00 – OK

DATA[0]:　　0x80(表示操作成功)

操作失败返回：

STATUS: 0x01 –FAIL

DATA[0] 参考错误代码表

描述: 用于锁定卡的DSFID，锁定后DSFID不可以更改

比如：

发送命令： **02 00 02 1a 42 5a 03**

正确返回： **02 00 02 00 80 82 03**

6.12 ISO15693_GET_System_Information (0x1B)

发送数据：

DATA [0]: Flags

DATA[1..8] UID(如果你使用地址模式，需输入 8 位 UID)

返回数据：

STATUS: 0x00 – OK

Data [0]: Flags

Data [1]: INFO Flags

Data [2..9]: UID

Data [10]: DSFID

Data [11]: AFI

Data [12..N]: Other fields

操作失败返回：

STATUS: 0x01 –FAIL

DATA[0] 参考错误代码表

描述: 用于获得卡的详细信息，具体内容请参考，ISO15693协议资料

比如：

发送命令： **02 00 02 1b 02 1b 03**

有卡 返回数据： **02 00 10 00 00 0F 4A 80 E9 11 00 00 07 E0 01 01 3F 03 88 7E 03**

6.13 ISO15693_Get_Multiple_Block_Security(0x1C)

发送数据：

DATA [0]: Flags

DATA[1] 第一个要读的块的地址

DATA[2] 要读块的个数

DATA[3..10] UID(如果你使用地址模式，需输入8位UID，否则不用输8位UID)

正确返回数据:

STATUS: 0x00 – OK

Data [0]: Flags

Data [1.N]: Block security status

操作失败返回：

STATUS: 0x01 –FAIL

DATA[0] 参考错误代码表

描述: 用于获取卡的各个块的安全状态位的数据

比如：

发送命令：　　02 00 04 1c 02 00 05 1f 03（读从0~4块的安全位）

正确返回：　　02 00 07 00 00 00 00 00 01 00 06 03

6.14 ISO15693_Transfer_Command (0x1D)

发送数据：

DATA[0]:　　要发送给卡的数据长度

DATA [1…N]　　要发送的数据（参照 ISO15693 协议）

正确返回数据:

STATUS:　0x00 – OK

Data [0…N]: 从卡返回的数据

操作失败返回：

STATUS:　　0x01 –FAIL

DATA[0]　　参考错误代码表

描述:　　一个通用命令，用户可以通过此命令，对卡进行各种操作。

比如：

发送命令：　　02 00 04 1D 02 02 2B 32 03（读取卡片信息）

正确返回：　　02 00 10 00 00 0F 72 9C 56 01 00 00 07 E0 08 00 3F 03 88 FD 03

7 错误/状态 代码(STATUS)

一般代码：

0x00:	表示命令执行成功
0x01:	表示命令操作失败（具体说明参见函数）
0x80:	表示参数设置成功
0x81:	表示参数设置失败
0x82:	表示通讯超时
0x83:	表示卡不存在
0x84	表示接收卡数据出错
0x87:	表示未知的错误
0x85:	表示输入参数或者输入命令格式错误
0x8f:	表示 输入的指令代码不存在

ISO14443 错误代码：

0x8A:	表示在块初始化中出现错误
0x8B:	表示在防冲突过程中得到错误的序列号
0x8C:	表示密码认证没通过

ISO15693 错误代码：

0x90	表示卡不支持这个命令
0x91	表示命令格式有错误
0x92	表示在命令的FLAG参数中，不支持OPTION 模式
0x93	表示要操作的BLOCK不存在

0x94	表示要操作的对象已经别锁定，不能进行修改
0x95	表示锁定操作不成功
0x96	表示写操作不成功