关键词

- TX800B 读卡模块
- Mifare One S50, S70
- CPU 卡 FM1208
- 扇区 密钥
- 一卡多用 一卡通
- 非接触式 智能卡

摘 要

TX800 CPU 卡模块是基于 13.56MHz 频率的非接触 Mifare、CPU 卡读写模块; 该模块支持 ISO14443 typeA、ISO14443-4、ISO7816-1/2/3/4 协议，具备 MIFARE 标准的加密算法，以及 CPU 卡操作所需的 DES 加密算法。模块支持 Mifare One（ S50、Mifare1 S70、Mifare Light、Mifare UltraLight 等）和非接触 CPU 卡(复旦微的 FM1208、握奇、英飞凌)等卡片的读写

TX800B 系列 CPU 卡读写模块具有易用、高可靠、多种接口、体积小等特点，可帮助用户方便、快捷地将当今最流行的非接触式 IC 卡技术融入到系统中，提高产品的档次。

本文详细介绍了 TX800B 模块的基于 51 单片机的 C51 库函数的使用方法。

# 目　录

# 1. 适用范围

本文对 TX800 CPU 卡读写模块在 51 单片机下的 C51 库函数做了非常详细的讲解，使用 51 系列单片机与 TX800 进行通讯的用户参考本文提供的各函数，可以非常容易的编写操作 CPU 卡的应用程序。使用其它处理器的 TX800 用户，也可以参考本文所描述提供的通讯格式和函数，将本文提供的函数很容易的移植到其它处理器上。

对于 C51 函数的调用，用户可不必关心数据块格式，只要理解函数的功能，输入、输出参数即可。当用户自己编写函数（包括非 C51 下的函数）时，就需了解数据块格式，必须按照数据块的格式来编写函数。

# 2. TX800 简介

TX800 是一个简单的串行读写模块，不带后缀“＋”时，接口为 CMOS 电平，用于与常用的微处理器（如单片机、ARM）的 UART 接口；当采用带后缀“＋”时，TX800＋模块内部带有 RS232 电平转换电路，能够直接与 PC 机的串口连接通信。

用户通过主机（包括单片机、ARM、DSP 以及 PC 机等）向 TX800 模块发送命令来对 TX800 进行读写控制。本应用指南将重点描述 TX800 与主机之间串行通信协议和命令。

## 2.1 引脚描述

### 2.1.1 天线一体化模块引脚描述

表 1 外接天线接口 J1

| 接　口 | 管　脚 | 符　号 | IO 类型 | 功能描述 |
|---|---|---|---|---|
| J1(1) | J1-1 | TX1 | 输出/O | 天线发送端 1 |
| | J1-2 | GND | 地/Power | 地 |
| | J1-3 | TX2 | 输出/O | 天线发送端 2 |
| | J1-4 | GND | 地/Power | 地 |

(1) J1 为模块与天线的接口,对于天线一体化（带后缀 T）的模块，如果用户使用模块上天线，可不用 J1。

表 2 与用户 MCU 接口 J2

| 接 口 | 管脚 | 符 号 | IO 类型 | 功能描述 | 上电后状态 |
|---|---|---|---|---|---|
| J2(1) | J2-1 | CTRL | 开漏/OC | 控制信号输出 | 1 |
| | J2-2 | BZ | 开漏/OC | 外部蜂鸣器驱动电路控制信号，需要串电阻 | 1 |
| | J2-3 | INT_OUT | 开漏/OC | 自动寻卡中断输出，低电平有效 | 1 |
| | J2-4 | VCC | 电源/Power | 电源正极, +3.6V～+12V | |
| | J2-5 | IDLE | 输入/I | 上电如果一直悬空模块处于工作状态。上电后如果将 IDLE 置低然后再置高，模块会进入空闲模式，功耗 4uA，空闲时不读卡；置低后模块进入工作模式。不用功耗控制时悬空即可。 | 1 |
| | J2-6 | GND | 地/Power | 地 | |
| | J2-7 | RXD | 输入/I | UART 接收端 | 1 |
| | J2-8 | TXD | 输出/O | UART 发送端 | 1 |

(1) 可将自动寻卡配置为寻到卡后自动回发，然后用 UART 接收中断，就可以不使用 INT_OUT 管脚。

表 3 用户接口 J3

| 接 口 | 管 脚 | 符号 | IO 类型 | 功能描述 |
|---|---|---|---|---|
| J3 | J3-0 | +3.3V | 地/Power | 3.3V 电源输出，最大提供 70mA 电流 |
| | J3-1、2、3 | NC | - | 预留未来使用 |
| | J3-4 | GetSnr_En | 输入/I | 只读卡号使能管脚，和 GND 短接时使能，悬空时为 1 |
| | J3-5 | GND | 地/Power | 地 |

如果 J3-4(GetSnr_En)接地，模块变为只读卡号，在此模式下，模块不能接收外部命令。

### 2.1.2 天线分体模块引脚描述

表 4 分体式 TX800 引脚描述

| 管脚 | 符 号 | IO 类型 | 功能描述 | 上电后状态 |
|---|---|---|---|---|
| 1 | TX1 | 输出/O | 天线发送端1 | |
| 2 | TX2 | 输出/O | 天线发送端2 | |
| 10 | GND | 地/Power | 地 | |
| 11 | GetSnr_en | 输入 | 只读卡号使能管脚，和 GND 短接时使能 | 1 |
| 12 | CTRL | 输出/O | 控制信号输出 | 1 |
| 13 | BZ | 输出/O | 外部蜂鸣器驱动电路控制信号，需要串电阻 | 1 |
| 14 | NC | | 空闲脚 | |
| 15 | TXD | 输出/O | UART发送端 | 1 |
| 16 | VCC | 电源/Power | 电源正极, +3. 5V～+7V, 请外接100uF电解电容 | |
| 17 | GND | 地/Power | 地 | |
| 18 | RXD | 输入/I | UART接收端 | 1 |
| 19 | INT_OUT | 输出/O | 自动寻卡中断输出，低电平有效 | 1 |
| 20 | IDLE | 输入/I | 上电如果一直悬空模块处于工作状态。上电后如果将 IDLE 置高然后再置低，模块会进入空闲模式，功耗 3uA，空闲时不读卡；置高后模块进入工作模式。不用功耗控制时上电后一直悬空即可，或者接高电平。 | |
| 21~22 | NC | | 空闲脚 | |

如果 11 脚(GetSnr_En)接地，模块变为只读卡号，在此模式下，模块不能接收外部命令。

## 2.2 典型电路

TX800 模块可以与任何带有 UART 口的 MCU 接口，图 1 所示为 TX800 与 MCS51 单片机 UART 的典型接口。

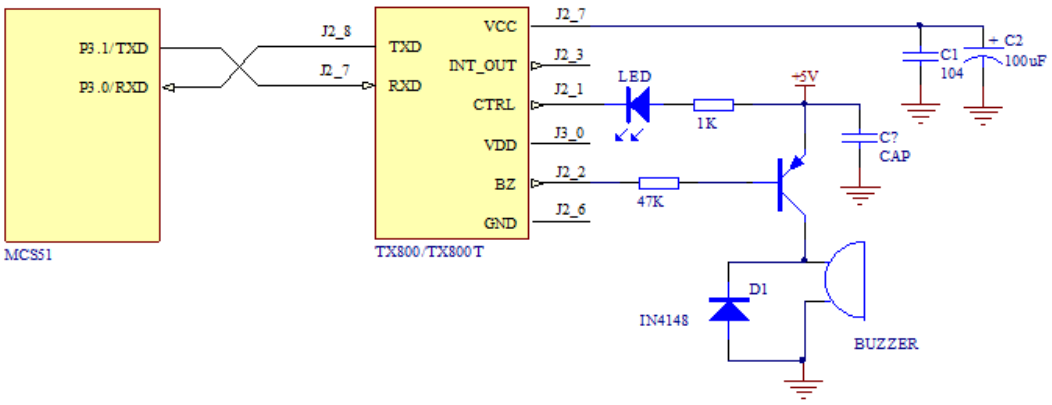



图 1 TX800/TX800T 与 MCS51 单片机接口图

当采用带 RS232 电平转换的 TX800T+模块时，如 图 2 所示，读写模块可以通过串口电缆直接与计算机相连，通过计算机来控制 TX800T+。

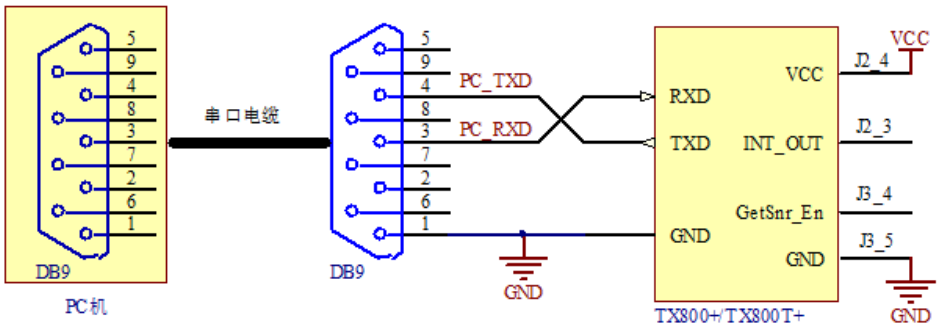



图 2 TX800+/TX800T+与 PC 机串口接口图

## 2.3 技术参数

表 5 TX800 模块技术参数表

| 产品型号 | TX800 系列 |
|---|---|
| 功　耗 | 50 毫安/直流 5V； |
| 工作频率 | 13.56 兆赫兹 |
| 读卡距离 | 20～100 毫米（FM1208 卡） |
| 接口方式 | UART、RS232 |
| 数据传输速率 | UART/RS232：9600～115200bit/s |
| 支持卡类型 | mifare1 S50、mifare1 S70、FM1208 等 |
| 尺　寸 | 分体式（不带后缀 T）：18.5mm×37mm×5mm |
| | 天线一体化（带后缀 T）：58mm×34.5mm×2.5mm |

## 2.4 极限参数

每个管脚的对地电压 ……-0.5～+5.5V
天线一体化模块 Vcc 对地的电压……-0.3～+12V
天线分体模块 Vcc 对地的电压……-0.3～+7V
每个管脚的最大 $I_{OL}$ ……20mA
湿度（相对湿度）……5～95%

**超出“绝对最大额定值”列出的值的条件下工作会造成器件的永久损坏。以上列出的是器件正常工作的额定值，并未涉及器件在这些条件或超出这些条件下的功能操作。器件不能长时间工作在绝对最大额定值条件下，否则会影响其可靠性。**

## 2.5 直流特性

VCC=+5.0V，器件都工作在建议的温度范围-30～85℃条件下，除非特别说明。

表 6　TX800 模块的直流特性

| 符　号 | 参　数 | 测试条件 | 最小值 | 典型值(1) | 最大值 | 单位 |
|---|---|---|---|---|---|---|
| VCC | TX800T工作电压 | | +3.6 | +5.0 | +9 | V |
| | TX800工作电压 | | +3.5 | +5.0 | +7 | V |
| VDD | 输出电压 | | | 3.3V | | V |
| $I_{VDD}$ max | VDD最大输出电流 | | | | 70 | mA |
| $T_{OK}$ | 上电后稳定工作时间 | | 5 | | | ms |

| | | | | | | |
|---|---|---|---|---|---|---|
| $I_{CC}$ | 电流消耗 | 读卡芯片配置成功 | | 50 | | mA |
| $V_{IL}$ | 输入低电平 | | - | 0.7 | 1.3 | V |
| $V_{IH}$ | 输入高电平 | | 2 | 3.3 | - | V |
| $V_{OL}$ | 输出低电平 | $I_{OL}$=20mA | | 0.6 | 1.0 | V |
| | | $I_{OL}$=3.2mA | | 0.2 | 0.3 | V |
| $V_{OH}$ | 输出高电平 | $I_{OH}$=-20uA | 3 | 3.1 | | V |
| $I_{IL}$ | 逻辑低电平输入电流 | Vpin=0.4V | | | -80 | μA |
| $I_{TL}$ | 逻辑1到0变化电流 | Vpin=2V | -30 | | -450 | μA |
| $I_{OL}$ | 低电平时的灌电流 | | | | -20 | mA |
| $I_{OH}$ | 高电平时的拉电流 | | | | 20 | μA |
| $C_{IO}$ | 管脚输入电容 | | | | 15 | pF |
| $T_{OP}$ | 工作温度(I) | | -30 | | +85 | ℃ |
| $T_{STR}$ | 存储温度 | | -55 | | +125 | ℃ |

(1) 典型值是难以保证的，这个值是在常温条件下测试得到。

## 2.6 封装及机械尺寸

TX800 模块根据是否采用天线一体化有两种封装型号：TX800 和 TX800T。天线分体式 TX800 封装尺寸如图 3 所示，天线分体式 TX800 封装尺寸如图 4 所示。

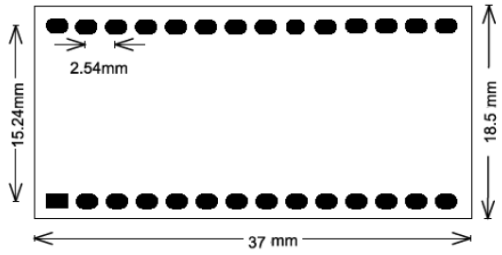



图 3 TX800 封装

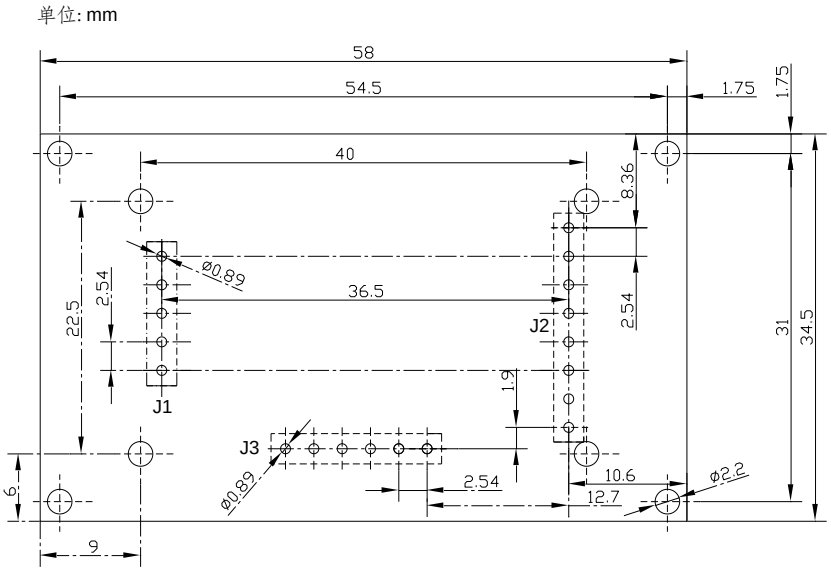



图 4 TX800T 封装

# 3. 只读卡号的最简单应用

如果是只要读取 mifare 的卡号，那只需要关注本小节，本小节以后的内容都可以忽略。

TX800B 能够配置为上电主动只读卡号模式，上电后无需向模块发送任何命令，只要有卡靠近模块就能主动通过 uart 发送卡号。主动只读卡号模式操作按下面步骤实现。

## 3.1 硬件配置

1.将 J3_4 对管脚旁的焊接点短接，以使能主动只读卡号功能。

2.按照下图所示连接 TX800 和 PC 机的串口或其它主机串口。根据硬件不同可选带 RS232 转换和不带 RS232 转换。

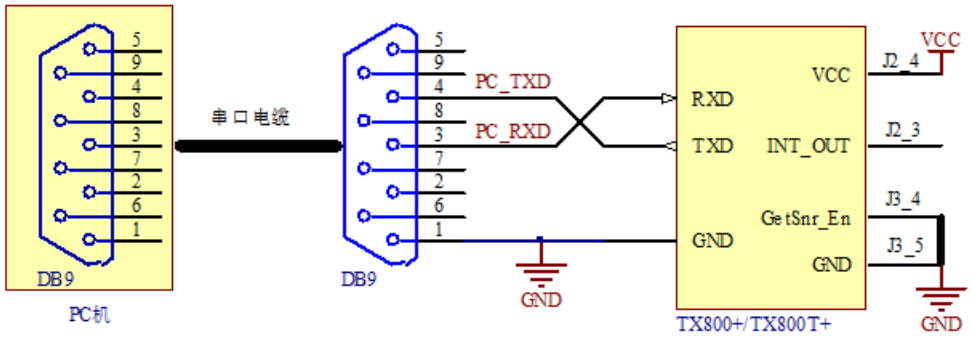



图 5 主动只读卡号硬件接线图

3.将您接收所用串口配置为：9600 波特率，1 个起始位，8 个数据位、无奇偶校验位、1 个停止位。

## 3.2 检测到卡片指示

当有 mifare 卡接近 TX800T 时，有卡的指示管脚 J2_3（INT_OUT）变为为低电平，在 J2_3 变为低电平 5ms 后，TX800BT 将卡号发出。如果用户使用串口接收中断方式进行卡号的接收，就可以不使用指示管脚 J2_3（INT_OUT）。使用中断接收要注意关中断时间不能超过 20ms。

一次刷卡，只发送一次卡号，如果卡片一直不拿开则不重复发生卡号，但有卡指示管脚仍然有效，即一直为低。

## 3.3 返回卡号说明

上电主动只读卡号模式下能读取所有 mifare 家族的卡号，包括 FM1208、Mifare1 S50、Mifare1 S70、Mifare Light、Mifare UltraLight、Mifare Pro 等。

对于 4 字节卡号的 mifare 卡（如 FM1208、mifare1 S50），返回的格式为 4 字节卡号+4 字节卡号的异或取反校验。卡号低字节在前。

如果是 7 字节卡号的 mifare 卡（如 Mifare UltraLight），返回的格式为 7 字节卡号+7 字节卡号的异或取反校验。

# 4. TX800 读卡模块数据传输协议

## 4.1 串口协议

主机与 TX800 串行接口通信过程中一帧的数据格式采用 1 个起始位，8 个数据位、无奇偶校验位、1 个停止位。

用户可以通过串口调试助手来调试开发 TX800，具体见 6 使用串口调试助手开发调试 TX800T 模块。

## 4.2 控制字符定义

下表列出了 TX800 与主机串行通信过程中用到的控制字符定义。

表 7 TX800 串行通信控制字符表

| 描述 | 定义 | 值 |
|---|---|---|
| 开始符 | STX | 0x20 |
| 终止符 | ETX | 0x03 |

## 4.3 协议描述

### 4.3.1 数据通信帧描述

数据通信以一帧为单位进行，格式如下：

表 8 数据通信帧结构

| 起始符<br>STX | 包号<br>SEQNR | 命令/状态<br>CMD/Status | 数据长度<br>Length | 数据<br>DATA | 校验和<br>BCC | 帧结束符<br>ETX |
|---|---|---|---|---|---|---|
| 1byte | 1byte | 1byte | 1byte | N bytes | 1byte | 1byte |

数据帧中各字段说明如表 9 所示：

表 9 数据帧各字段说明

| 字段 | 长度 | 说明 | 补充 |
|---|---|---|---|
| STX | 1 | STX=0x20，数据帧的起始符，每一帧数据都是以 STX 开始 | |
| SEQNR | 1 | 该数据帧包序号，从 0 到 255 循环。可以用来作为通信间的错误检查，从机（模块）接收到主机发来的信息，在应答信息中发出一个同样的 SEQ 信息，主机可以通过此信息检查是否发生的“包丢失”的错误。第一个包的 SEQ 可为任意值。 | 该字段主机发送和接收的应该相同 |
| Cmd<br>/Status | 1 | 主机——从机：命令 Command<br>从机——主机：状态 Status | |
| Length | 1 | 该帧所带数据信息长度<br>若模块返回状态不为 0（OK），则 Length!=0。 | |
| DATA | Length | 数据信息,长度等于 Length | |
| BCC | 1 | 校验和。从包号（SEQNR）开始到数据（DATA）的最后一字节异或取反。 | |
| ETX | 1 | ETX=0x03，是一个帧的结束标志 | |

为了下文的描述，把 SEQ+CMD/Status+LEN+DATA+BCC 同称为数据块 DATA Block。

表 10 数据块 DATA BLOCK

| DATA BLOCK | 包号<br>SEQNR | 命令/状态<br>CMD/Status | 数据长度<br>Length | 数据<br>DATA | 校验和<br>BCC |
|---|---|---|---|---|---|
| | 1byte | 1byte | 1byte | N bytes | 1byte |

数据帧接收规则：

- 一帧的结束一定是 ETX，但接收到 0x03 则不一定是帧结束；
- 帧长必须不小于 6 字节，最大不能超过 255 字节，且帧长必须等于数据长度加 6；
- BCC 计算必须正确。

主机发送数据必须符合以上规则，否则从机不会执行任何命令，也不会有任何错误响应。同样主机接收从机的数据也必须符合以上规则，如果不符合，主机必须丢弃这帧数据。

### 4.3.2 主机发送命令至模块

主机发送格式如表 11 所示。

表 11 主机发送给 TX800 遵循的格式

| 主机 | 数据传送方向 | TX800 | 说明 |
|---|---|---|---|
| STX+DATA BLOCK+ETX | → | | 一次将一帧数据全部发送完。 |

用户在给 TX800 模块发送命令时，连续的发送 STX (0x20)+数据块+0x03(结束符)。通过判断 TX800 返回数据的正确性来判断 TX800 是否正确执行了本条命令。

### 4.3.3 从机返回数据给主机

TX800 在接收到主机发送的数据后，根据数据块的内容执行相应命令，并将执行命令后的状态或者数据（以下将状态和数据都统称为数据）返回给主机。

TX800 发送格式如表 12 所示：

表 12 TX800 发送格式

| TX800 | 数据传送方向 | 主机 | 说明 |
|---|---|---|---|
| STX+DATA BLOCK+ETX | → | | 一次将一帧数据全部发送完。 |

TX800 返回数据给主机时，也采用 STX+ DATA BLOCK+ETX 的数据格式。

# 5. TX800T 的 C51 函数

TX800T 提供了 C51 函数库，使用 C51 单片机的客户可直接调用这些函数库中的函数，其它系列的微控制器可参考 C51 函数库来进行移植，移植过程只要进行少量的修改。

## 5.1 函数列表

TX800 模块具备 CPU 卡的读写功能同时兼容 TX522 模块的功能，为了便于使用及查找特将其函数功能分类如表 13、14、15 所示。通用函数包括 M1 卡或 CPU 卡共同需要使用的函数及模块的相关配置函数。

CPU 卡寻卡、反碰撞、选卡、跟 M1 卡一样；M1 卡选卡成功后，可以用 M1 卡专用命令对卡操作，但 CPU 卡还需要进一步请求协议参数 ATS 激活 CPU 卡（就是所谓的 CPU 卡转入命令），激活成功后 CPU 卡才能转入卡内部 COS 操作。

表 13 通用函数列表

| 函数名称 | 命令值 | 输入参数(发送) | 输出参数 | 功能描述 |
|---|---|---|---|---|
| TX_Config | 0x52 | -- | -- | 复位且配置模块 |
| TX_Close | 0x3F | -- | -- | 关闭模块 |
| TX_ConfigAnt | 0x53 | AntMode | -- | 配置模块的天线驱动模式 |
| TX_SetBaud | 0x54 | Cnt | -- | 波特率设置 |
| TX_GetInfo | 0x4F | -- | *Info | 读取模块信息 |
| TX_GetCardSnr | 0x10 | ReqCode,*TagType, *Sak,*SnrLen, *Snr | *TagType, *Sak,*SnrLen, *Snr | 激活卡片并获取卡号卡片类型 |
| TX_Halt | 0x45 | -- | -- | 将卡置于挂起模式 |
| TX_Reset | 0x4E | Msec | -- | 关闭天线输出数ms，使卡复位 |
| TX_SetControlBit | 0x50 | -- | -- | 将控制位置为高电平 |
| TX_ClrControlBit | 0x51 | -- | -- | 将控制位置为低电平 |
| TX_Buzzer | 0x60 | Freuence,Opentm, Closetm,Repcnt | -- | 输出驱动蜂鸣器信号，控制动作时间、间隙时间和重复次数 |

表 14　CPU 卡专用库函数列表

| 函数名称 | 命令值 | 输入参数(发送) | 输出参数 | 功能描述 |
|---|---|---|---|---|
| TX_CPULoadKey | 0x20 | KeyNo,KeyID, *NewKey | -- | 改变存贮在模块内密匙区中的密钥 |
| TX_CPUEnter | 0x21 | *ATS | *ATS | CPU卡转入（请求ATS参数转入CPU卡内文件目录，默认选择MF根目录） |
| TX_CPUActiveEnter | 0x22 | ReqCode | *Snr<br>*ATS | 复合指令（一步完成激活卡片，请求ATS参数；CPU卡内文件目录，默认选择MF根目录） |
| TX_CPUSelectADF | 0x23 | *ADF（2字节应用目录名）<br>*Data | *Data | 通过应用目录名选择应用目录 |
| TX_CPUAuth | 0x24 | *ADF ,KeyNo , KeyID | -- | 使用模块内部密钥区中的密码对指定的卡的应用目录进行验证 |
| TX_CPUReadBinary | 0x26 | *FID,Offset,*Data, NBytes | *Data | 读二进制文件 |
| TX_CPUReadBinaryAuth | 0x27 | *ADF,KeyNo,KeyId,*FID, Offset,*Data, NBytes | *Data | 读二进制文件 |

| | | | | |
|---|---|---|---|---|
| TX_CPUWriteBinary | 0x28 | *FID,Offset, *Data, NBytes | -- | 写二进制文件 |
| TX_CPUWriteBinaryAuth | 0x29 | *ADF,KeyNo,KeyId,*FID, Offset,*Data, | -- | 写二进制文件 |
| TX_CPUFormat | 0x2a | *MFMasterKey（主密钥） | -- | CPU卡格式化、擦除MF下所有文件信息成为空白卡 |
| TX_CPUInitialize | 0x2b | *MFMasterKey（主密钥） AppNo (应用个数) | -- | CPU卡初始化；将卡片生成预先定义的文件结构 |
| TX_CPUChangeKey | 0x2c | * ADF, KID *NewKey | -- | 增加/修改密钥 |
| TX_CPUDeselect | 0x2d | -- | -- | CPU卡协议停活 |
| TX_CosCommand | 0x2F | *CosCommand,TxNBytes, *Data, *RxNBytes | *Data *RxNBytes | CPU卡正常复位后，可以采用COS命令代替对上面各项操作 |

以上函数大多数命令是经过封装复合命令,使用者不需要了解 CPU 卡 COS 结构；如果特殊应用可以采用 COS 命令单步操作实现。

表 15 M1 卡专用库函数列表

| | | | | |
|---|---|---|---|---|
| TX_M1AuthKey | 0x73 | KeyAB,Sector,*Key | -- | 直接密码验证 |
| TX_M1Write | 0x47 | Block ,*Data | -- | 向卡中指定块写入一16字节的数据块 |
| TX_M1Read | 0x46 | Block | *Data | 从卡中指定块中读出一个16字节的块 |
| TX_M1Value | 0x70 | ValueMode,Block,*Value, Trans_Block | -- | 包含加、减、恢复函数，并可进行不同块之间的自动传送 |
| TX_UlWrite | 0x76 | Block,*Data | -- | 向UltraLight卡中相应块写入4字节数据 |
| TX_M1LoadKey | 0x4C | KeyAB,Sector,*Key | -- | 改变存贮在模块内密匙区中的密钥 |
| TX_M1Auth | 0x72 | KeyAB,Sector,KeySector | -- | 使用模块内部密钥区中的密码对指定的卡的扇区Sector进行验证 |
| TX_M1WriteAuth | 0x11 | KeyAB, KeySector , Block, idata *Data | -- | 带验证的写操作 |
| TX_M1ReadAuth | 0x12 | KeyAB, KeySector , Block | *Data | 带验证的读操作 |

## 5.2 函数返回状态值列表

所有函数的返回值都是该函数执行后的状态结果，具体返回值请查看各函数说明。如果各函数有返回数据，则都是以指针的形式返回，同时返回状态标识如下。

表 16 TX800 库函数的用到的状态值列表

| 名称 | 值（hex） | 描述 |
|---|---|---|
| OK, COMM_OK | 00 | 函数调用成功 |
| NO_TAG_ERR | 01 | 在有效区域内没有卡 |
| CRC_ERR | 02 | 从卡中接收到了错误的 CRC 校验和 |
| EMPTY | 03 | 值溢出 |
| AUTH_ERR | 04 | 不能验证 |
| PARITY_ERR | 05 | 从卡中接收到了错误的奇偶校验位 |
| CODE_ERR | 06 | 通信错误 |
| SENDR_ERR | 08 | 在防冲突时读到了错误的串行码 |
| KEY_ERR | 09 | 证实密码错 |

| | | |
|---|---|---|
| NOT_AUTH_ERR | 0A | 卡没有验证 |
| BIT_COUNT_ERR | 0B | 从卡中接收到了错误数量的位 |
| BYTE_COUNT_ERR | 0C | 从卡中接收了错误数量的字节 |
| TRANS_ERR | 0E | 调用 Transfer 函数出错 |
| WRITE_ERR | 0F | 调用 Write 函数出错 |
| INCR_ERR | 10 | 调用 Increment 函数出错 |
| DECR_ERR | 11 | 调用 Decrment 函数出错 |
| READ_ERR | 12 | 调用 Read 函数出错 |
| COLL_ERR | 18 | 冲突错 |
| ACCESS_TIMEOUT | 1B | 访问超时 |
| QUIT | 1F | 上一次了送命令时被打断 |
| CHK_WR_OK | 00 | Check Write 正确 |
| CHK_WR_FAILED | 01 | Check Write 出错 |
| CHK_WR_COMP_ERR | 02 | Check Write:写出错（比较出错） |
| COMM_ERR | FF | 串行通信错误 |
| MI_WRONG_VALUE | 7B | 值块格式错误 |
| STATUS_COS_ERROR | 28 | COS 操作时执行出错（具体状态内容见表 16） |

表 17　TX800 库函数 COS 操作时执行出错状态值列表

| SW1 SW2　(hex) | 意　义 |
|---|---|
| 6281 | 回送的数据可能出错 |
| 6283 | 选择文件无效，文件或密钥效验出错 |
| 63CX | X 表示还可以验证的次数（当变为 0 时，被锁死） |
| 6400 | 状态标志未改变 |
| 6581 | 写 EEPROM 不成功 |
| 6700 | 错误的长度 |
| 6900 | CAL 与线路保护要求不匹配 |
| 6901 | 无效的状态 |
| 6981 | 命令与文件结构不相符 |
| 6982 | 不满足安全状态 |
| 6983 | 密钥被锁死 |
| 6985 | 使用条件不足 |
| 6987 | 无安全报文 |
| 6988 | 安全报文数据项不正确 |
| 6A80 | 数据域参数出错 |
| 6A81 | 功能不支持或卡中无 MF 或卡片已锁定 |
| 6A82 | 文件未找到 |
| 6A83 | 记录未找到 |
| 6A84 | 文件无足够的空间 |
| 6A86 | 参数 P1 P2 错误 |
| 6A88 | 密钥未找到 |
| 6B00 | 在达到 Le/Lc 字节之前文件结束，偏移量出错 |

| | |
|---|---|
| 6Cxx | Le 错误 |
| 6E00 | 无效的 CLA |
| 6F00 | 数据无效 |
| 9302 | MAC 错误 |
| 9303 | 应用已被锁定 |
| 9401 | 金额不足 |
| 9403 | 密钥未找到 |
| 9406 | 所需的 MAC 不可以 |

注意：
当 SW1 的高半字节为‘9’，且低半字节不为‘0’时，其含义依赖于相关应用。
当 SW1 的高半字节为‘6’，且低半字节不为‘0’时，其含义与应用无关。
只有错误标志 STATUS_COS_ERROR “28”出现时，SW1 SW2 状态信息才会出现并指示当前 CPU 卡所处 COS 的状态。可以通过表 18 了解状态的具体含义。

## 5.3 描述

### 5.3.1 配置—TX_Config

函数原型：uchar TX_Config(void);
输入参数：无
输出参数：无
函数返回：TX800 执行命令后的状态。
功能描述: 对模块进行初始化，初始化成功后，模块上的指示灯将点亮，此时天线发射载波信号,任何进入天线感应区的卡可得电进入 IDLE 状态,可使用任一函数对卡进行操作。此时读卡芯片完全被激活，所消耗的电流最大。
建议用户每次读卡操作前都调用该函数。
注意：对模块的配置需要大约 2.5ms 的时间，因此调用该函数后需要等待 2.5ms 以上才能进行其他操作。
数据块格式描述：
主机→TX800 命令模式）：
SEQNR： 0 （可自定义）
COMMAND： 0x52
LENGTH： 0
DATA[…]： 无
例如：数据帧

| STX | SEQNR | CMD | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x52 | 0x00 | none | 0xAD | 0x03 |

TX800→主机（响应模式）：
SEQNR： 0
STATUS： 状态字
LENGTH： 0
DATA[…]： 无
例如：数据帧

| STX | SEQNR | Status | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x00 | 0x00 | none | 0xFF | 0x03 |

### 5.3.2 关闭－TX_Close

函数原型：uchar TX_Close(void);
输入参数：无
输出参数：无
函数返回：TX800B 执行命令后的状态。
功能描述: 此函数将关闭 TX800B 模块，指示灯熄灭，天线不发送载波信号，模块消耗的电流最小，在此状态下，模块不能使用。若要重新使用模块，需要调用 TX_Config()函数对 TX800B 重新进行配置。
数据块格式描述：
主机→TX800 命令模式）：
SEQNR： 0 （可自定义）
COMMAND： 0x3F
LENGTH： 0
DATA[…]： 无
例如：数据帧

| STX | SEQNR | CMD | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x3F | 0x00 | none | 0xC0 | 0x03 |

TX800→主机（响应模式）：
SEQNR： 0
STATUS： 状态字
LENGTH： 0
DATA[…]： 无
例如：数据帧

| STX | SEQNR | Status | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x00 | 0x00 | none | 0xFF | 0x03 |

### 5.3.3 配置—TX_ConfigAnt

函数原型：uchar TX_ConfigAnt (uchar idata AntMode);
输入参数：AntMode:天线驱动模式，0x00= TX1 和 TX2 关闭；0x01=TX1 驱动；0x02 = TX2 驱动；0x03 = TX1 和 TX2 同时驱动
输出参数：无
函数返回：TX800 执行命令后的状态。
功能描述: 对模块天线驱动模式进行配置。模块默认为 TX1 和 TX2 同时驱动。只要要进行双天线操作时，才会使用到该函数。
分体式模块不支持该函数。
数据块格式描述：
主机→TX800 命令模式）：
SEQNR： 0 （可自定义）
COMMAND： 0x53

LENGTH： 1
DATA[…]： TxMode
例如：配置为 TX1 驱动的数据帧

| STX | SEQNR | CMD | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x53 | 0x01 | 0x01 | 0xAC | 0x03 |

TX800→主机（响应模式）：
SEQNR： 0
STATUS： 状态字
LENGTH： 0
DATA[…]： 无
例如：数据帧

| STX | SEQNR | Status | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x00 | 0x00 | none | 0xFF | 0x03 |

### 5.3.4 配置—TX_SetBaud

函数原型：uchar TX_SetBaud (uchar idata Cnt);
输入参数：Cnt 序号‘0’—4800；‘1’—9600；‘2’—19200；‘3’—38400；‘4’—57600；‘5’—115200；
输出参数：无
函数返回：TX800 执行命令后的状态。
功能描述：对模块模块串口通信的波特率进行配置。 设置成功后需复位后才新的波特率。注意：模块的默认波特率为 9600，如果设置波特率后必须记得更新后的波特率。
数据块格式描述：
主机→TX800 命令模式）：
SEQNR： 0 （可自定义）
COMMAND： 0x54
LENGTH： 1
DATA[0]： Cnt
例如：将波特率配置为 115200 的数据帧

| STX | SEQNR | CMD | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x54 | 0x01 | 0x01 | 0x AB | 0x03 |

TX800→主机（响应模式）：
SEQNR： 0
STATUS： 状态字
LENGTH： 0
DATA[…]： 无
例如：数据帧

| STX | SEQNR | Status | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x00 | 0x00 | none | 0xFF | 0x03 |

### 5.3.5 获取信息—TX_GetInfo

函数原型：uchar TX_GetInfo (uchar idata *Info);

输入参数：无

输出参数：模块的信息*Info，Info 为保存信息空间的首地址。Info[0]～Info[4]为模块类型标识，依次为‘T’，‘X’，‘8’，‘0’，‘0’，Info[5]～Info[8]为模块的唯一序列号，Info[9]为固件版本号，高四位为版本号的整数，取值从 1 到 15，低四位为版本号的小数，取值从 0 到 9。

函数返回：TX800B 执行命令后的状态。

功能描述: 获取 TX800B 模块的信息。

数据块格式描述：

主机→TX800 命令模式）：

SEQNR： 0 （可自定义）

COMMAND： 0x4F

LENGTH： 0

DATA[…]： 无

例如：数据帧

| STX | SEQNR | CMD | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x4F | 0x00 | none | 0xB0 | 0x03 |

TX800→主机（响应模式）：

SEQNR： 0

STATUS： 状态字

LENGTH：10

DATA[0]：产品类型标识 0，‘T’

DATA[1]：产品类型标识 1，‘X’

DATA[2]：产品类型标识 2，‘8’

DATA[3]：产品类型标识 3，‘0’

DATA[4]： 产品类型标识 4，‘0’

DATA[5]：读卡芯片序列号 0

DATA[6]：读卡芯片序列号 1

DATA[7]：读卡芯片序列号 2

DATA[8]：读卡芯片序列号 3

DATA[9]：产品软件版本号

例如：数据帧

| STX | SEQNR | Status | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x00 | 0x0a | 11 字节产品型号及版本 | 0xXX | 0x03 |

### 5.3.6 激活卡片并获取卡号—TX_GetCardSnr

函数原型：uchar TX-GetCardSnr(uchar ReqCode, uchar idata *TagType, uchar idata *Sak, uchar idata *SnrLen, uchar idata *Snr);

输入参数： ReqCode：请求模式 ReqCode 取值为 1 或 0

ReqCode＝0（IDLE），请求天线范围内 IDLE 状态的卡（HALT 状态的除外）

ReqCode＝1（ALL），请求天线范围内的所有卡。

输出参数：

(1) *TagType：请求应答：2 个字节的卡片类型，对于 mifare 1 卡，返回类型为 0x0004，则*TagType ＝ 0x04 ，*(TagType+1)＝0x00。其意义见表 19。

表 19 *TagType 的含义

| b15 | b14 | b13 | b12 | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 保留 | | | | | | | | 序列号大小<br>00:4bytes<br>01:7bytes<br>10:10bytes | | 保留 | 任何位为 1 时，则为比特帧防碰撞方式 | | | | |

各种常用卡返回的数据如下表 20 所示：

表 20 常用卡的*TagType 值

| 卡类型 | *TagType | 卡类型 | *TagType |
|---|---|---|---|
| Mifare1 S50 | 0x0004 | SHC1101 | 0x0004 |
| Mifare1 S70 | 0x0002 | SHC1102 | 0x3300 |
| Mifare Light | 0x0010 | 11RF32 | 0x0004 |
| Mifare UltraLight | 0x0044 | | |

(2) *Sak：最后一级选择应答的应答，其意义如表 21 所示。

表 21 *Sak 含义

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| RFU | | | RFU | | 判断是否选择完成 | RFU | |

bit2 位用来判断是否还有序列号未读出，bit2＝0 时，表示所有序列号输入完毕，选择成功，bit2＝1 时表示序列号没有全部输入，选择没有完全成功，还要进行下一级的防碰撞和选择操作。

当*Sak＝xxxxx1xx 时，序列号没有完成，还要进行下一级的防碰撞和选择操作。

当*Sak＝xx1xx0xx 时，选择成功，该卡符合 ISO/IEC 14443-4 标准。

当*Sak＝xx0xx0xx 时，选择成功，该卡不符合 ISO/IEC 14443-4 标准。

各种常用卡返回的数据如下所表 22 所示：

表 22 常用卡的*Sak 的值

| 卡类型 | *Sak | 卡类型 | *Sak |
|---|---|---|---|
| Mifare1 S50 | 0x08 | Mifare0 UltraLight | 0x04 |
| Mifare1 S70 | 0x18 | SHC1101 | 0x22 |
| Mifare1 Light | 0x01 | 11RF32 | 0x08 |
| MF1208 | 0x28 | ----- | ----- |

(3) 函数返回： TX800 执行命令后的状态，可能的状态值如下：OK, QUIT, NO_TAG_ERR,

(3) *SnrLen：返回卡片序列号的长度。

(4) *Snr：返回卡片的序列号。

函数返回：执行命令后的状态。

功能描述：该命令为后面将要介绍的请求、防碰撞和选择三条命令的组合。成功执行该命令后即可进行验证及后续操作。

数据块格式描述：

主机→TX800 命令模式）： 读取卡号 26 寻卡 ： 2000100100EE03

读取卡号 52 寻卡 ： 2000100101EF03

SEQNR： 0 （可自定义）
COMMAND：0x10
LENGTH： 1
DATA[0]： ReqCode( 00:26 寻卡；01：52 寻卡 )
例如：以 IDLE 方式激活卡的数据帧

| STX | SEQNR | CMD | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x10 | 0x01 | 0x00 | 0xee | 0x03 |

TX800→主机（响应模式）：
SEQNR： 0
STATUS： 状态字
LENGTH： 4 字节+序列号的长度，Mifare1 S50、S70、Light 卡：8 字节，
Mifare0 UltraLight 和 Mifare3 Desfire 卡：11 字节
DATA[0..1]： *TagType：请求应答，2 个字节的卡片类型
DATA[2]： *Sak：最后一级选择应答的应答
DATA[3]： *SnrLen：返回卡片序列号的长度
DATA[4..4+ SnrLen]： *Snr：返回卡片的序列号
例如：一张序列号为 0x007e0a42 的 Mifare1 S50 卡返回的数据

| STX | SEQNR | Status | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x00 | 0x08 | 0x04 0x00 0x08 0x04 0x42 0x0A 0x7E 0x00 | 0xXX | 0x03 |

例如：一张序列号为 0x000000007e0a42 的 Mifare UltraLight 卡返回的数据

| STX | SEQNR | Status | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x00 | 0x0b | 0x44 0x00 0x00 0x07 0x42 0x0A 0x7E 0x00 0x00 0x00 0x00 | 0xXX | 0x03 |

### 5.3.7 将卡置于挂起模式—TX_Halt

函数原型：TX_Halt (void);
输入参数：无
输出参数：无
函数返回：执行命令后的状态。

功能描述：函数返回：TX800 执行命令后的状态。
功能描述：将天线区所选择卡置为挂起状态。如果要进行重新选择，则应用 ALL 模式调用 TX_GetCardSnr 命令。如果要进行重新选择,也可以将卡离开天线操作区再进入，或执行复位函数 TX_Reset()。
可以配合使用 TX_GetCardSnr()和 TX_Halt()函数，进行一次性扣费，如卡进入感应区后只扣一次（一元钱），离开后，下次进入再扣一次，若卡在感应区内停留时间较长，也不会扣多一次。
数据块格式描述：
主机→TX800 命令模式）：20004500BA03
SEQNR： 0 （可自定义）
COMMAND：0x45
LENGTH： 0x00

例如：

| STX | SEQNR | CMD | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x45 | 0x00 | ------ | 0xBA | 0x03 |

TX800→主机（响应模式）：

SEQNR： 0

STATUS： 状态字

LENGTH： 0

例如：

| STX | SEQNR | Status | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x00 | 0x0 | ----- | 0xFF | 0x03 |

### 5.3.8 复位—TX_Reset

函数原型：uchar TX_Reset(uchar Msec);

输入参数：Msec：取值 0~255，模块上射频电路关闭时间（以 ms 毫秒为单位），Msec＝0 时，一直关闭。

输出参数： 无

函数返回：TX800 执行命令后的状态。

功能描述:该函数使模块上的射频电路关闭,关闭的时间由参数 Msec 指定,若 Msec=0,射频电路将一直处于关闭状态，一直到下一个 TX_GetCardSnr 命令到来。关闭射频电路能使天线内的所有卡复位。

数据块格式描述：

主机→TX800 命令模式）： 20004E010ABA03

SEQNR： 0 （可自定义）

COMMAND：0x4E

LENGTH： 1

DATA[0]： 10Msec

例如：将天线信号关闭 10ms 的数据帧

| STX | SEQNR | CMD | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x4e | 0x01 | 0x0A | 0xbA | 0x03 |

TX800→主机（响应模式）：

SEQNR： 0

STATUS： 状态字

LENGTH： 0

DATA[0]： 无

例如：数据帧

| STX | SEQNR | Status | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x00 | 0x00 | none | 0xFF | 0x03 |

### 5.3.9 置位控制位—Set_Control_Bit

函数原型：uchar TX_Set_Control_Bit(void);

输入参数： 无
输出参数： 无
函数返回：TX522B 执行命令后的状态，可能的状态值如下：OK，QUIT，COMM_ERR 中的某一个。
功能描述：此函数设置 TX522B 模块的控制位 Ctrl（J2_1 脚）为高电平。
数据块格式描述：
主机→TX522 命令模式）：
SEQNR： 0 （可自定义）
COMMAND：0x50
LENGTH： 0
DATA[0]： 无

TX522→主机（响应模式）：
SEQNR： 0
STATUS： OK，QUIT，COMM_ERR 中的某一个
LENGTH： 0
DATA[0]： 无

### 5.3.10 清除控制位—Clr_Control_Bit

函数原型：TX_Clr_Control_Bit(void);
输入参数： 无
输出参数： 无
函数返回：TX522B 执行命令后的状态，可能的状态值如下：OK，QUIT，COMM_ERR 中的某一个。
功能描述：此函数设置 TX522B 模块的控制位 Ctrl（J2_1 脚）为低电平。
数据块格式描述：
主机→TX522 命令模式）：
SEQNR： 0 （可自定义）
COMMAND：0x51
LENGTH： 0
DATA[0]： 无

TX522→主机（响应模式）：
SEQNR： 0
STATUS： OK，QUIT，COMM_ERR 中的某一个
LENGTH： 0
DATA[0]： 无

### 5.3.11 输出蜂鸣器信号—TX_Buzzer

函数原型：TX_Buzzer(uchar Frquence, uchar Opentm, uchar Closetm, uchar Repcnt);
输入参数：Frequence: 输出方波频率，取值（0～255），对应频率（0.73～4K），0 为直流，198 对应 2K。
Opentm: 方波输出持续时间，取值（0～255），10ms 的分辨率
Closetm: 间隙时间，取值（0～255），10ms 的分辨率

Repcnt: 重复次数

输出参数： 无

函数返回：TX800B 执行命令后的状态，可能的状态值如下：OK， COMM_ERR 中的某一个。

功能描述：此函数输出一方波用于驱动交流蜂鸣器或低电平驱动直流蜂鸣器，驱动的频率、持续时间、间隙时间和重复次数可设定。下面是频率对照表。

表 23 设定值与输出频率对照表

| Frequence | 输出频率（KHz） | Frequence | 输出频率（KHz） |
|---|---|---|---|
| 0 | 输出低电平（直流） | 140 | 1.333 |
| 1 | 0.735 | 160 | 1.515 |
| 20 | 0.787 | 180 | 1.74 |
| 40 | 0.847 | 198 | 2.00 |
| 60 | 0.913 | 200 | 2.04 |
| 80 | 0.990 | 220 | 2.50 |
| 100 | 1.081 | 240 | 3.17 |
| 120 | 1.198 | 255 | 4.00 |

数据块格式描述：

主机→TX800 命令模式）：

SEQNR： 0 （可自定义）

COMMAND：0x60

LENGTH： 400000

DATA[0]： Frequence

DATA[1]: Opentm

DATA[2]: Closetm

DATA[3]: Repcnt

TX522→主机（响应模式）：

SEQNR： 0

STATUS： OK， COMM_ERR 中的某一个

LENGTH： 0

### 5.3.12 改变存贮在模块内密匙区中的密钥—TX_CPULoadKey

函数原型：TX_CPULoadKey (uchar KeyNo,uchar KeyID,uchar idata *NewKey);

输入参数：

（1）KeyNo 密钥区号

（2） KeyID 密钥标识

（3）*NewKey 新密钥 16 字节

输出参数：无

函数返回：执行命令后的状态。

功能描述：该命令可以实现将 CPU 卡相关目录的密钥保存在模块中。需要注意的是密钥只是保存在读卡模块中与卡片无关。

特别要强调的是将模块作为发卡器使用的时候，调用 CPU 初始化函数时，是默认将模

块密码区中保存的密码依次对应的作为 CPU 卡文件系统的初始化密码，每个密码区共有 3 个密码,密钥标识分别为 00、01、02。初始化时默认对应关系：00 区为管理员密码区，01 区为第一个应用的密码区, 02 区为第二个应用的密码区,依此规律往下递增。因此在调用 CPU 初始化函数时化前需要预先在安全的地方将相关的密钥利用此函数装载到模块内部。

实际应用时的模块中保存的密钥可以跟卡片的应用文件密钥无任何内在关联，可以使用任何一个密钥区的任何一个密钥对卡片的任意一个密钥做认证；但是为了便于管理建议每个应用最好按次序跟模块保存的密钥对应。（事实上每个密钥区保存 3 个密钥，具体其中的每个密钥用于那个应用可以由使用者定义）

下表列出了 3 个应用的密钥对应关系，用户可以参照下面的关系装载更多的应用密钥。目前模块最多支持 15 个应用，如果应用数目大于 15 个客户可以根据需要定制。

表 24 应用的密钥说明

| KeyNo（密钥区号) | KeyID（密钥标识） | 对应卡的文件目录 | 备注 |
|---|---|---|---|
| 00 | 00 | 3f00（根目录） | 卡片管理（用于初始化卡） |
| | 01 | | 卡片管理（用于初始化卡） |
| | 02 | | 卡片管理（用于初始化卡） |
| 01 | 00 | 1001（应用目录 1） | 应用目录 1（管理密钥） |
| | 01 | | 应用目录 1（ 读密钥 ） |
| | 00 | | 应用目录 1（ 读密钥 ） |
| 02 | 00 | 1002（应用目录 2） | 应用目录 2（管理密钥） |
| | 01 | | 应用目录 2（ 读密钥 ） |
| | 02 | | 应用目录 2（ 读密钥 ） |
| 03 | 00 | 1002（应用目录 3） | 应用目录 3（管理密钥） |
| | 01 | | 应用目录 3（ 读密钥 ） |
| | 02 | | 应用目录 3（ 读密钥 ） |

注：没有 TX_CPULoadKey 前 模块内的密钥为 CA AA AF 4D EA F1 DB AE CA AA AF 4D EA F1 DB AE

数据块格式描述：

主机→TX800 命令模式)：2000201200000011223344556677889 9AABBCCDDEEFFCD 03

SEQNR： 0 （可自定义）

COMMAND：0x20

LENGTH： 0x12 （18）

DATA[0]： 00 应用序号 (0 号应用)

DATA[1]： 00 密钥标识 ID (0 号密码)

DATA[2..18]： 新密钥 00112233445566778899AABBCCDDEEFF

例如：可以假设模块密钥区 00 号应用的密钥 00 号密钥是卡片的主控密钥

| STX | SEQNR | CMD | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x20 | 0x12 | 00112233445566778899AABBCCDDEEFF | 0xCD | 0x03 |

TX800→主机（响应模式）：

SEQNR： 0

STATUS： 状态字

LENGTH： 0

例如：

| STX | SEQNR | Status | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x00 | 0 | ----- | 0xFF | 0x03 |

失败时返回其它不确定值。

### 5.3.13 CPU 卡转入—TX_CPUEnter

函数原型：uchar TX_CPUEnter (uchar idata *ATS, uchar idata *Snr);

输入参数： 无

输出参数：

(1) *ATS 返回卡片 ATS 参数；

(2) *Snr 卡片序列号。

函数返回：执行命令后的状态。

功能描述：该命令只有在 TX_GetCardSnr 执行成功后才能使用；实现 CPU 卡转入功能，命令成功后才能正常响应 CPU 卡 COS 命令（CPU 卡转入成功后默认选择 MF（3F00）根目录）。

值得注意的是同一张卡一旦转入成功后，一值停留在 CPU 卡协议激活状态而不能再响应该命令，如果再次发送这条命令时返回无卡错误。此时需调用 TX_ CPUDeselect 函数使当前激活的卡返回到 HALT 状态或者调用 TX_Reset 关闭天线使卡复位才能重新响应其它的非 CPU 卡命令。

数据块格式描述：

主机→TX800 命令模式）：20002100de03

SEQNR： 0 （可自定义）

COMMAND：0x21

LENGTH： 0

DATA[0]： 无

例如：以 IDLE 方式激活卡的数据帧

| STX | SEQNR | CMD | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x21 | 0x00 | --- | 0xde | 0x03 |

TX800→主机（响应模式）：

SEQNR： 0

STATUS： 状态字

LENGTH： 16 字节

DATA[0]： TL:长度字节

DATA[1]： T0：格式字节

DATA[2]： TA1：(TA、TB、 TC 存在指示)

DATA[3]： TB1 : DR / DS

DATA[4]： TC1： SEI / SFGI

DATA[5]： T1： COS 版本号

DATA[6]： T2： COS 厂商代码（复旦微电子）

DATA[7]： T3：保留字节
DATA[8..]：T4-T11：卡序列号

例如：一张序列号为 0x2197EE92 的卡返回的数据

| STX | SEQNR | Status | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x00 | 0x10 | 0x10 0x78 0x80 0x90 0x00 0x00 0x00 x00 x00 0x00 21 97 EE 92 | 0xEF | 0x03 |

### 5.3.14 CPU 卡复合指令复位—TX_CPUActiveEnter

函数原型： uchar TX_CPUActiveEnter(uchar ReqCode , uchar idata *ATS, uchar idata *Snr);

输入参数： ReqCode：请求模式 ReqCode 取值为 1 或 0
ReqCode＝0（IDLE），请求天线范围内 IDLE 状态的卡（HALT 状态的除外）
ReqCode＝1（ALL），请求天线范围内的所有卡。

输出参数：

(1) *ATS 返回卡片 ATS 参数；

(2) *Snr 卡片序列号。

函数返回：执行命令后的状态。

功能描述：该命令是 TX_GetCardSnr 命令与 TX_CPUEnter 命令的组合；完成一步实现 CPU 卡激活功能，转入 CPU 卡内部 EEPROM 操作；命令成功激活后才能响应 CPU 卡 COS 命令（激活成功后默认选择 MF 根目录）。TX_CPUActiveEnter 函数调用成功后卡已经转入协议激活状态，此时卡只能接受 CPU 卡命令或者 COS 命令，其它命令卡片返回没有找到卡错误，同一张卡一旦转入 CPU 卡成功，一停留在 CPU 卡 COS 协议激活的状态；如果再发送其它非 COS 文件支持的命令均返回无卡错误。解决方法是发送 CPU 协议激活指令 TX_ CPUDeselect，或者将卡拿开一段时间，再将卡放入感应区。

数据块格式描述：

主机→TX800 命令模式）：2000220101DD03

SEQNR： 0 （可自定义）
COMMAND：0x22
LENGTH： 0
DATA[0]： 无

例如：以 IDLE 方式激活卡的数据帧

| STX | SEQNR | CMD | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x22 | 0x00 | 0x00 / 0x01 | 0XDC/0xDD | 0x03 |

TX800→主机（响应模式）：
SEQNR： 0
STATUS： 状态字
LENGTH： 16 字节
DATA[0]： TL:长度字节
DATA[1]： T0：格式字节
DATA[2]： TA1：(TA、TB、 TC 存在指示)
DATA[3]： TB1 : DR / DS

DATA[4]： TC1: SEI / SFGI
DATA[5]： T1: COS 版本号
DATA[6]： T2: COS 厂商代码（复旦微电子）
DATA[7]： T3：保留字节
DATA[8..]：T4-T11：卡序列号
例如：一张序列号为 0x2197EE 的卡返回的数据

| STX | SEQNR | Status | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x00 | 0x10 | 10788090000000000000002197EE92 | 0xEF | 0x03 |

### 5.3.15 选择文件—TX_CPUSelectADF

函数原型：TX_CPUSelectADF (uchar idata *ADF ,uchar idata *Data);

输入参数：*ADF 应用目录名
输出参数：*Data 应用目录文件信息
函数返回：执行命令后的状态。
功能描述：该命令实现选择 CPU 卡中的文件或者目录，执行成功后被选择的目录为当前操作目录。
数据块格式描述：

主机→TX800 命令模式）：200023023F00E103

SEQNR： 0 （可自定义）
COMMAND：0x23
LENGTH： 0x02 （2 字节）
DATA[0…1]：0x3F00 应用目录名,这里是选择 MF 根目录；所以 ADF 为 3F00; 假设选择 1 个应用目录则应用目录名为 1001, 那么 DATA[0…1]为 1001.
例如：

| STX | SEQNR | CMD | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x23 | 0x02 | 0x3f 0x00 | 0XE1 | 0x03 |

TX800→主机（响应模式）：
SEQNR： 0
STATUS： 状态字
LENGTH： 长度字节
DATA[0]： *Data 返回应用目录文件信息
例如：

| STX | SEQNR | Status | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x00 | 0x-- | ----- | 0xff | 0x03 |

失败时返回 2 字节的 COS 命令状态码，可以查询 COS 表了解其内容。

### 5.3.16 使用模块内部密认证—TX_CPUAuth

函数原型：TX_CPUAuth(uchar idata*ADF, uchar KeyNo, uchar Keyid);

输入参数：
（1）* ADF 应用目录名 （应用目录（1001 ~100f）注：ADF 为主目录时（3f00））
（2） KeyNo 密钥区号( 0 ~ 15)

（3） KeyID 密钥标识 ID （主控密钥（00）、读密钥（01）、写密钥（02））

输出参数：无

函数返回：执行命令后的状态。

功能描述：该命令可以实现使用模块内部密钥区中的密钥对卡片的指定应用目录进行验证。模块中保存的密钥跟卡片的应用文件密钥没有强制的对应关系，可以使用任何一个密钥区对卡片的任意一个密钥做认证；但是为了便于管理建议每个应用最好按次序跟模块保存的密钥对应。（每个应用序号区可以保存 3 个密钥，具体其中的每个密钥用于那个应用由使用者定义）

数据块格式描述：

主机→TX800 命令模式）：200024043F000001E103

SEQNR： 0 （可自定义）

COMMAND：0x24

LENGTH： 0x04

DATA[0]： 3f ADF_H

DATA[1]： 00 ADF_L

DATA[2]： 00 应用序号(0 号应用)(管理员 MF 根目录 root)

DATA[3]： 01 密钥标识 ID(1 号密码)

例如：可以假设模块密钥区 00 号应用的密钥 00 号密钥是卡片的主控密钥

| STX | SEQNR | CMD | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x24 | 0x04 | 3F000000 | 0x E1 | 0x03 |

TX800→主机（响应模式）：

SEQNR： 0

STATUS： 状态字

LENGTH： 0

例如：

| STX | SEQNR | Status | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x00 | 0x00 | ----- | 0xFF | 0x03 |

失败时返回其它不确定值。如果返回 2 字节的 COS 命令状态码，可以查询 COS 表了解其内容。

### 5.3.17 读二进制文件—TX_CPUReadBinary

函数原型：uchar TX_CPUReadBinary (uchar idata*FID, uchar Offset,
uchar NBytes,uchar idata*Data);

输入参数：

（1）*FID 文件 ID

（2）Offset 偏移地址

（3）NBytes 需要读回数据的长度

（4）*Data 读回数据存放缓冲区

输出参数：

*Data 读出数据缓冲区

函数返回：执行命令后的状态。

功能描述：该命令实现在当前目录下从指定文件中读取需要读取的二进制数据，其中FID 为文件名，用于确需要操作的文件。

数据块格式描述：

主机→TX800 命令模式）：2000260400150409C503

SEQNR： 0 （可自定义）

COMMAND：0x26

LENGTH： 0x04

DATA[0…1]：0x0015 二进制文件 ID 号 ,这里选择基本二进制文件，如果选择用户二进制文件则为 0016。

DATA[2]： 0x04 偏移地址为 4 ； 表示从从第 4 字节开始读。

DATA[3]： 0x09 需要读回的字节数。

DATA[4…15]： 返回数据。

例如：

| STX | SEQNR | CMD | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x26 | 0x04 | 0015 04 09 | 0XC5 | 0x03 |

TX800→主机（响应模式）：

SEQNR： 0

STATUS： 状态字

LENGTH： 0x09 （9 字节）

例如：

| STX | SEQNR | Status | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x00 | 0x09 | 112233445566778899 | 0xE7 | 0x03 |

失败时如果返回 2 字节的 COS 命令状态码，可以查询 COS 状态表了解其内容。

### 5.3.18 带认证读二进制文件—TX_CPUReadBinaryAuth

函数原型：uchar TX_CPUReadBinaryAuth(uchar idata*ADF, uchar KeyNo, uchar Keyid,
uchar idata*FID, uchar Offset, uchar Nbytes,
uchar idata*Data);

输入参数：

(1) *ADF 应用目录(2 字节) （1001~100f）

(2) KeyNo 密码区号（1 字节）（00~0F）

(3) Keyid 密码标识(1 字节) (00~02)

(4) *FID 二进制文件名（2 字节）

(5) Offset 偏移地址(1 字节) （0~0F）

(6) NBytes 需要读回数据的长度(1 字节) （0~0F）为 0 时默认 16 字节全部读出

(7) *Data 读回数据存放缓冲区

输出参数：

*Data 读出数据缓冲区

函数返回：执行命令后的状态。

功能描述：该改函数是 CPUSelectFile、 CPUAuth、 CPUReadBinary 三个函数的组合；实现从 CPU 卡中指定应用目录中指定文件读取需要读取的二进制数据。

数据块格式描述：

主机→TX800 命令模式）：20 00 27 08 1001 01 02 0016 00 00D403
SEQNR： 0 （可自定义）
COMMAND：0x27
LENGTH： 0x08
DATA[0..1]：0x1001 应用目录号 ADF ，选择 1001 应用目录
DATA[2]： 0x01 密钥区号，表示区号 1
DATA[3]： 0x02 密钥标识号，02 号密钥
DATA[4.. 5]：0x0016 需要读取的文件名
DATA[6]： 0x00 偏移地址
DATA[7]： 0x00 读取字节数，以为为 0，所以全部读出 16 字节
例如：

| STX | SEQNR | CMD | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x27 | 0x08 | 1001 01 02 0016 00 00 | 0x D4 | 0x03 |

TX800→主机（响应模式）：
SEQNR： 0
STATUS： 状态字
LENGTH： 0x10（16 字节）
例如：

| STX | SEQNR | Status | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x00 | 0x10 | 00 00 00 00 00 00 00 00<br>00 00 00 00 00 00 00 00 | 0xEF | 0x03 |

失败时返回 2 字节的 COS 命令状态码，可以查询 COS 表了解其内容。

### 5.3.19 写二进制文件—TX_CPUWriteBinary

函数原型：uchar TX_CPUWriteBinary ( uchar idata*FID, uchar Offset,uchar NBytes,
uchar idata *Data )

输入参数：
（1）*FID 文件 ID
（2）Offset 偏移地址
（4）NBytes 需要写入的字节数
（3）*Data 写入数据缓冲区
输出参数：无
函数返回：执行命令后的状态。
功能描述：该命令实现在当前目录下写入数据到指定制文件中，其中 FID 为文件名，用于确需要操作的文件。
数据块格式描述：
主机→TX800 命令模式）：2000280F0015040B11223344556677889 9AABBC203
SEQNR： 0 （可自定义）
COMMAND：0x28
LENGTH： 0x0F （15）
DATA[0…1]：0x0015 二进制文件 ID 号 ，这里选择基本二进制文件，如果选择用户二进制文件则 为 0016。

DATA[2]： 0x04 偏移地址为 4 ；表示从从第 4 字节开始写入。

DATA[3]： 0x0B 需要写入字节数为 11；

DATA[4…15]：112233445566778899AABB 写入数据。

注意：字节数+偏移地址小于等于文件的空间尺寸； 文件的空间尺寸由初始化时决定，这里为 16 字节。

例如：

| STX | SEQNR | CMD | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x28 | 0x0f | 0015 04 0B 1122 33445566778899AABB | 0XC2 | 0x03 |

TX800→主机（响应模式）：

SEQNR： 0

STATUS： 状态字

LENGTH： 0 字节

例如：

| STX | SEQNR | Status | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x00 | 0x00 | ----- | 0xff | 0x03 |

失败时返回 0x40 同时跟随有 2 字节的 COS 命令状态码，可以查询 COS 表了解其内容。

### 5.3.20 带认证写二进制文件—TX_CPUWriteBinaryAuth

函数原型：uchar TX_CPUWriteBinaryAuth( uchar idata*ADF, uchar KeyNo, uchar Keyid，uchar idata *FID，uchar Offset，uchar NBytes，uchar idata*Data )

输入参数：

(1) *ADF 应用目录(2 字节)（1001~100f）

(2) KeyNo 密码区号（1 字节）（00~0F）

(3) Keyid 密码标识(1 字节) (00~02 )

(4) *FID 二进制文件名（2 字节）

(5) Offset 偏移地址(1 字节)

(6) NBytes 需要写入的数据长度(1 字节)

(7) *Data 写入数据缓冲区

输出参数：无

函数返回：执行命令后的状态。

功能描述：该改函数是 CPUSelectFile、 CPUAuth、 TX_CPUWriteBinary 三个函数的组合；实现写入数据到指定目录的指定文件中。

数据块格式描述：

主机→TX800 命令模式）：

2000 29 13 1001 01 02 0016 04 0B112233445566778899AABBCE03

SEQNR： 0 （可自定义）

COMMAND：0x29

LENGTH： 0x0F （15）

DATA[0…1]：0x1001 应用目录序号。

DATA[2]： 0x01 密码区号。

DATA[3]： 0x02 密码标识。

DATA[4…5]：0x0015　二进制文件 ID 号 ，这里选择基本二进制文件，如果选择用户二进制文件则 为 0016。

DATA[6]：　0x04　偏移地址为 4 ；表示从从第 4 字节开始写入。

DATA[7]：　0x0B　需要写入字节数为 11；

DATA[8…19]：112233445566778899AABB　写入数据。

注意：字节数+偏移地址小于等于文件的空间尺寸；　文件的空间尺寸由初始化时决定，这里为 16 字节。

例如：

| STX | SEQNR | CMD | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x29 | 0x0f | 0015040B11223344<br>5566778899AABB | 0xCE | 0x03 |

TX800→主机（响应模式）：

SEQNR：　0

STATUS：　状态字

LENGTH：　0 字节

例如：

| STX | SEQNR | Status | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x00 | 0x00 | ----- | 0xff | 0x03 |

失败时返回 0x40 同时跟随有 2 字节的 COS 命令状态码，可以查询 COS 表了解其内容。

### 5.3.21 格式化 CPU 卡—TX_CPUFormat

函数原型：uchar TX_CPUFormat (uchar idata *MFMasterKey);

输入参数：　*MFMasterKey (16 字节的管理员主密钥)

输出参数：　无

函数返回：执行命令后的状态。

功能描述：一步删除 CPU 卡 MF 下所有信息,该命令实现选卡认证删除功能，成功后将删除 MF 文件夹下管理员密钥，所有文件信息恢复成出厂状态的卡；另外会建立默认目录 3F00;可以在该目录下建立任何文件不受权限限制。

数据块格式描述：

主机→TX800 命令模式）：20002A10ffffffffffffffffffffffffffffffffC503

SEQNR：　0 （可自定义）

COMMAND：0x2A

LENGTH：　0x10 （16 字节）

DATA[0…15]：ffffffffffffffffffffffffffffffff　管理员密钥

例如：

| STX | SEQNR | CMD | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x2A | 0x10 | ffffffffffffffffffffffffffffffff | 0XC5 | 0x03 |

TX800→主机（响应模式）：

SEQNR：　0

STATUS：　状态字

LENGTH：　0 字节

例如：

| STX | SEQNR | Status | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x00 | 0x00 | ----- | 0xff | 0x03 |

失败时返回 2 字节的 COS 命令 SW 状态码，可以查询 COS 表了解其内容。

### 5.3.22 CPU 卡初始化－TX_CPUInitialize

函数原型：uchar TX_CPUInitialize (uchar idata*MFMasterKey，uchar APPNO );

输入参数： *MFMasterKey (16 字节的管理员主密钥)；APPNO 建立应用的个数。

输出参数：无

函数返回：执行命令后的状态。

功能描述：该命令实现 CPU 卡预初始化，建立所需要的应用目录个数，成功后卡片具有特定的文件系统结构。此命令按照模块预先规定好文件结构格式来建立文件系统的，用户不能自行更改。

文件内容为根目录 MF（3 F 0 0），MF 下第一个应用目录为 1001；第二个应用目录为 1002；根据建立应用的个数往下递增。

在每个应用目录中都建立了个 4 个二进文件提供给用户使用；特别值得注意的是：在 4 个二进制文件中，文件标识为 0015 的文件是基本二进制文件，对它进行读操作时不需要任何权限；也就是说不需要对目录做任何密钥认证也可以对其读操作，在满足写权限时可以对其改写操作；此文件主要用于对卡片内应用目录的识别，避免盲目的对目录认证，使该目录进入锁死状态。

3 个用户文件 0016 、0017、0018 均受读写权限的控制，只有在满足读写权限时才能对其进行读写操作。

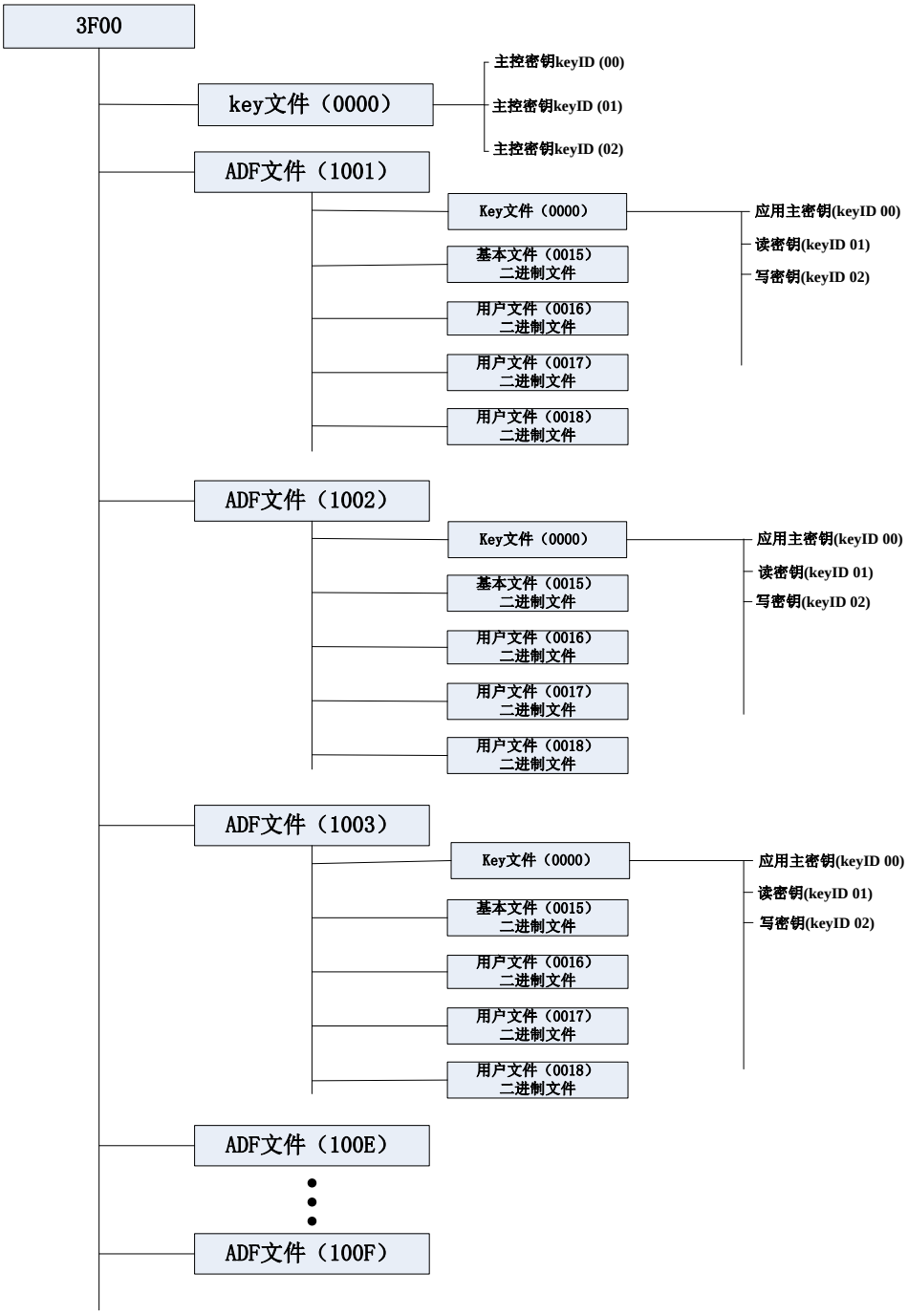



图 6　CPU 卡初始化文件结构

以上所示是调用 CPU 卡初始函数实现的文件系统结构；该文件结构固定不能改变；客户如果需要实现自己想要的文件系统结构，需要提前预定；也可以参考卡片资料根据自己需要发送 COS 命令建立相关的文件结构。

数据块格式描述：

主机→TX800 命令模式）：20002B1103ffffffffffffffffffffffffffffffffC103

SEQNR：　　0 （可自定义）

COMMAND：0x2B

LENGTH：　0x11 （16 字节）

DATA[0]： 0x03 应用个数

DATA[1…15]：ffffffffffffffffffffffffffffffff 管理员密钥

例如：

| STX | SEQNR | CMD | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x2B | 0x11 | 03ffffffffffffffffffffffffffffffff | 0XC1 | 0x03 |

TX800→主机（响应模式）：

SEQNR： 0

STATUS： 状态字

LENGTH： 0 字节

例如：

| STX | SEQNR | Status | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x00 | 0x00 | ----- | 0xff | 0x03 |

失败时返回 2 字节的 COS 命令状态码，可以查询 COS 表了解其内容。

### 5.3.23 增加/修改密钥—TX_CPUChangeKey

函数原型：uchar TX_CPUChangeKey (uchar idata*ADF, uchar KID，uchar idata*NewKey);

输入参数：

(1) *ADF；应用目录序号 2 字节

（2）KID；密钥标识号 1 字节

（3）*NewKey ；新密钥 16 字节

输出参数：无

函数返回：执行命令后的状态。

功能描述：该命令用于在修改 ADF 所指定应用的密钥；操作成功后卡片处在 ADF 指定的目录，权限为特权。所以通常只是在发卡器上使用；作为读卡器使用时最好不开放这个函数。

数据块格式描述：

主机→TX800 命令模式）:20002c 131001 01555555555555555555555555555555DA03

SEQNR： 0 （可自定义）

COMMAND：0x2C

LENGTH： 0x13

DATA[0...1]： 1001 应用文件 ID ；

DATA[2]：01 密钥标识

DATA[3…19]: 55555555555555555555555555555555 新密钥 16 字节

例如：

| STX | SEQNR | CMD | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x2C | 0x13 | 131001015555555555555555555555555555555555 | 0xDA | 0x03 |

TX800→主机（响应模式）：

SEQNR： 0

STATUS： 状态字

LENGTH： 0 字节

例如：

| STX | SEQNR | Status | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x00 | 0x00 | ----- | 0xff | 0x03 |

失败时返回 0x40 同时跟随有 2 字节的 COS 命令状态码，可以查询 COS 表了解其内容。

### 5.3.24 CPU 卡协议停活—TX_ CPUDeselect

函数原型：uchar TX_CPUDeselect(void);

输入参数： 无

输出参数： 无

函数返回：执行命令后的状态。

功能描述：该函数可以使被激活的 CPU 卡，返回到 Halt 状态；读卡模块跟 CPU 卡交易完成之后，CPU 卡应被置为 Halt 状态；但如果 CPU 卡成功获取 ATS 协议参数之后就不能再使用 TX_Halt 命令让卡返回到 Halt 状态，只有采用 TX_CPUDeselect 命令才能使卡返回到 Halt 状态； 此命令主要用于多张卡一起进入读卡模块电场范围的情况，可以让交易过的卡返回到 Halt 状态，然选择另外一张卡操作。

数据块格式描述：

主机→TX800 命令模式）：

SEQNR： 0 （可自定义）

COMMAND：0x2D

LENGTH： 0x00

例如：

| STX | SEQNR | CMD | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x2D | 0x00 | ___ | 0Xd2 | 0x03 |

TX800→主机（响应模式）：

SEQNR： 0

STATUS： OK, QUIT, NO_TAG_ERR, CRC_ERR, PARITY_ERR, BIT_COUNT_ERR, COMM_ERR 中的某一个

例如：

| STX | SEQNR | Status | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x00 | 0x00 | 0x00 | 0xFF | 0x03 |

### 5.3.25 COS 命令—TX_CosCommand

函数原型：uchar TX_CosCommand (uchar idata *CosCommand， uchar TxNBytes, uchar idata *Data, uchar RxNBytes );

输入参数：

（1）* CosCommand CPU 卡 cos 命令

（2） TxNBytes 发送命令长度

（3）*Data 返回数据地址

（4）*RxNBytes 返回数长度

输出参数：

（1）*Data 返回数据地址

（2）*RxNBytes　　返回数长度

函数返回：执行命令后的状态。

功能描述：该命令可以实现让模块响应 CPU 卡的 COS 命令对 CPU 卡做相关的操作，此命令需在 CPU 正常复位后，才能正常操作，否则返回状态不确定。

数据块格式描述：

主机→TX800 命令模式）：20002F05 0084 0000 085903

SEQNR：　0 （可自定义）

COMMAND：0x2F

LENGTH：　TxNBytes=0x05， 发送的 CPU 卡 cos 命令为 5 字节数据

DATA[0…5]： CPU 卡 cos 命令：Data“0084 0000 08”获取 8 字节的随机数

例如:从 CPU 卡获取 8 字节的随机数

| STX | SEQNR | CMD | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x2F | 0x05 | 0x0084000008 | 0x59 | 0x03 |

TX800→主机（响应模式）：

SEQNR：　0

STATUS：　状态字

LENGTH：　8 返回数长度(RxNBytes=8)

DATA[0…8]：返回 8 字节的随机数“78 86 79 52 77 55 32”

例如：

| STX | SEQNR | Status | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x00 | 0x08 | 0x78 86 79 52 77 55 32 | 0xFF | 0x03 |

失败时返回其它不确定值。如果返回 2 字节的 COS 命令状态码，可以查询 COS 表了解其内容。

### 5.3.26 直接密码证实—TX_M1AuthKey

函数原型：uchar TX_ M1AuthKey(uchar KeyAB, uchar Sector, uchar idata *Key);

输入参数：KeyAB：密钥类型（1 字节）。可取值为 KeyAB＝0x00（KEYA），利用密钥 A 进行验证，或 KeyAB＝0x04（KEYB），利用密钥 B 进行验证。

Sector：所要验证的卡扇区号（也即将要访问的卡的扇区号），取值范围 0～39，能用于 S70 卡。

Key：用于证实的密码首地址，,应在外部定义一个共 6 个字节的数组用于存放密码

输出参数： 无

函数返回：TX800B 执行命令后的状态，可能的状态值如下：OK，QUIT，NO_TAG_ERR，AUTH_ERR, PARITY_ERR，BIT_COUNT_ERR，COMM_ERR。

功能描述： TX_Auth2 函数必须要依赖于 TX_M1LoadKey ()函数的曾经的成功执行，在进行证实之前，一定要确认正确的密钥已经存于模块的密钥区内。而 TX_AuthKey 函数则将密钥存放于 Key 指针所指向的 6 个字节存储区内，TX_AuthKey 函数执行时不对密钥区进行操作，因此也称为直接密码证实。若卡中的密钥与所传输的密码相匹配。则证实成功，函数将返回 OK。

直接密码证实一般用于对每一张卡来说密钥都不同的应用，如在使用安全模块（PSAM 卡）的消费应用中，消费机首先将卡的序列号读出，然后与 PSAM 卡中消费主密钥一起生

成导出密钥，然后直接用导出密钥与卡的一个应用扇区相互证实。

数据块格式描述：
主机→TX800 命令模式）：
SEQNR： 0 （可自定义）
COMMAND：0x73
LENGTH： 8
DATA[0]： KeyAB
DATA[1]： Sector
DATA[2]： Key[0]
…
DATA[7]： Key[5]
例如：用密码 0xff 0xff 0xff 0xff 0xff 0xff 证实扇区 0 的数据帧

| STX | SEQNR | CMD | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x73 | 0x08 | 0x00 0x00 0xff 0xff 0xff 0xff 0xff 0xff | 0x84 | 0x03 |

TX800→主机（响应模式）：
SEQNR： 0
STATUS： OK， QUIT， NO_TAG_ERR， PARITY_ERR， BIT_COUNT_ERR，COMM_ERR 中的某一个
LENGTH： 0
DATA[0]： 无
例如：数据帧

| STX | SEQNR | Status | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x00 | 0x00 | none | 0xFF | 0x03 |

### 5.3.27 写块—TX_M1Write

函数原型：uchar TX_ M1Write(uchar Block,uchar idata *Data)
输入参数：Block：卡块号（1 字节）： S50：1～63；S70：1～255
*Data：16 字节数据指针，Data 为写入的 16 字节数据的首地址。
输出参数： 无
函数返回:TX800B 执行命令后的状态,可能的状态值如下:OK,QUIT,NO_TAG_ERR，CRC_ERR，NOT_AUTH_ERR，PARITY_ERR，BIT_COUNT_ERR，COMM_ERR 中的某一个。
功能描述：对卡内某一块进行验证成功后，即可对同一扇区的各个块进行写操作（只要访问条件允许），其中包括位于扇区尾的密码块，这是更改密码的唯一方法。
Mifare 卡中一个块的数据是 16 字节，因此读写一次均是 16 个字节。所读块号必须与之前所验证的块号在同一个扇区内，mifare1 卡从块号 0 开始按顺序每 4 个块 1 个扇区。
数据块格式描述：
主机→TX800 命令模式）：
SEQNR： 0 （可自定义）
COMMAND：0x47
LENGTH： 17

DATA[0]： Block
DATA[1]： 所要写的第一个字节
：
DATA[16]： 所要写的最后一个字节
例如：往块 2 写入数据的数据帧

| STX | SEQNR | CMD | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x47 | 0x11 | 0x02 16 字节数据 | 0xXX | 0x03 |

TX800→主机（响应模式）：
SEQNR： 0
STATUS： OK， QUIT， NO_TAG_ERR， NOT_AUTH_ERR， WRITE_ERR，BIT_COUNT_ERR，COMM_ERR 中的某一个
LENGTH： 0
DATA[0]： 无
例如：数据帧

| STX | SEQNR | Status | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x00 | 0x00 | none | 0xFF | 0x03 |

补充说明：
注意：在将数据写到卡片上的某一扇区时，一定要小心。因为有些 block 中存储了密码数据以及存储允许使能数据。特别是每一个扇区的 Block3 中存放了该扇区的存取条件，包含有 KEYA，KEYB 及该扇区的控制字。Mifare 1 卡片出厂时的 Block3 有缺省值，为：“a0a1a2a3a4a5ff078069b0b1b2b3b4b5”,共 16 个 Bytes。
涉及 Mifare 1 卡片的存储结构等信息，请参考 Mifare 1 卡片数据手册。
程序员在使用 Mifare 1 卡片做应用时，一定要清清楚楚记住每一个扇区的 Block3 的数据，这样也就记住了扇区的密码和存取控制字。否则，扇区的存储空间将不执行 Read/Write 等操作而失效。
任何人试图用任何方式来读写不知密码的卡片或某一扇区都是徒劳无益的。
卡片应放在安全的地方，即不要放在离模块天线较近的地方。因为当模块对其它卡片执行某些指令时，有可能无意间对这一卡片进行了读/写等操作，从而操作卡片的失效。

### 5.3.28 读块—TX_M1Read

函数原型：uchar TX_ M1Read(uchar Block,uchar idata *Data)
输入参数：Block：卡块号（1 字节）： S50：0～63；S70：0～255
输出参数：*Data：Data 为读回 16 字节数据的首地址。
函数返回：TX800B 执行命令后的状态，可能的状态值如下：OK，QUIT，NO_TAG_ERR，CRC_ERR，NOT_AUTH_ERR，PARITY_ERR，BIT_COUNT_ERR，COMM_ERR 中的某一个。
功能描述：在验证成功后，使用该函数读 Mifare 卡中相应块的数据。Mifare 卡中一个块的数据是 16 字节，因此读写一次均是 16 个字节。
所读块号必须与之前所验证的块号在同一个扇区内，mifare1 卡从块号 0 开始按顺序每 4 个块 1 个扇区。密码数据不能被读取。

数据块格式描述：

主机→TX800 命令模式）：

SEQNR： 0 （可自定义）

COMMAND：0x46

LENGTH： 1

DATA[0]： Block

例如：读取块 2 的数据的数据帧

| STX | SEQNR | CMD | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x46 | 0x01 | 0x02 | 0xba | 0x03 |

TX800→主机（响应模式）：

SEQNR： 0

STATUS： OK，QUIT，NO_TAG_ERR，CRC_ERR，NOT_AUTH_ERR，PARITY_ERR，BIT_COUNT_ERR，COMM_ERR 中的某一个

LENGTH： 16

DATA[0]： 所访问块的第一个字节

：

DATA[15]： 所访问块的最后一个字节

例如：数据帧

| STX | SEQNR | Status | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x00 | 0x10 | 16 字节数据 | 0xXX | 0x03 |

### 5.3.29 带内部自动传送的值操作—TX_M1Value

函数原型：uchar TX_M1Value(uchar ValueMode, uchar Block, uchar idata *Value, uchar Trans_Block);

输入参数：ValueMode：0xC0—减； 0xC1—加； 0xC2—恢复

Block：卡内块地址，对该块进行值操作，取值范围：S50：1～63；S70：1～255

*Value：4 字节数据指针，用来存储减少值或增加值，当进行恢复操作时，该值为空值。Value 是减少值或增加值存放的首地址，存放时，低地址存放高字节。

Trans_Block：传输块地址，取值范围：S50：1～63；S70：1～255。

输出参数： 无

函数返回：TX800B 执行命令后的状态，可能的状态值如下：OK, QUIT, NO_TAG_ERR, BIT_COUNT_ERR, TRANS_ERR, CODE_ERR, COMM_RERR 中的某一个。

功能描述：此函数对卡内的某一块进行加、减或数据备份，该块必须为值块格式，并支持自动传送。该函数其实是 TX_Increment、TX_Decrement 或 TX_Restore 函数与 TX_Transfer 函数的组合。因此可以用该函数替换上述函数。若卡块号与传输块号相同，则将操作后的结果写入原来的块内；若卡块号与传输块号不相同，则将操作后的结果写入传输块内，结果传输块内的数据被覆盖，原块内的值不变。当模式为“恢复”时，“值”无意义。

数据块格式描述：

主机→TX800 命令模式）：

SEQNR： 0 （可自定义）

COMMAND：0x70

LENGTH： 7
DATA[0]： ValueMode
DATA[1]： Block
DATA[2]： Value(LL)
DATA[3]： Value(LH)
DATA[4]： Value(HL)
DATA[5]： Value(HH)
DATA[6]： Trans_Block
例如：将块 1 的值减 3，然后传送到块 2 的数据帧

| STX | SEQNR | CMD | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x70 | 0x07 | 0xc0 0x01 0x03 0x00 0x00 0x00 0x02 | 0x48 | 0x03 |

TX800→主机（响应模式）：
SEQNR： 0
STATUS： OK, QUIT, NO_TAG_ERR, BIT_COUNT_ERR, TRANS_ERR, CODE_ERR, COMM_RERR 中的某一个
LENGTH： 0
DATA[0]： 无
例如：数据帧

| STX | SEQNR | Status | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x00 | 0x00 | none | 0xFF | 0x03 |

### 5.3.30 写 UltraLight—TX_UlWrite

函数原型：uchar TX_UlWrite(uchar Block, uchar *Data);
输入参数：Block：卡块号（1 字节）： 0～15
*Data： 4 字节数据指针，Data 为写入的 4 字节数据的首地址。
输出参数：无
函数返回：执行后可能返回：OK，QUIT，NO_TAG_ERR，CRC_ERR，NOT_AUTH_ERR，PARITY_ERR ， BIT_COUNT_ERR ， COMM_ERR ， CHK_WR_FAILED ，CHK_WR_COMP_ERR 中的某一个。
功能描述：对 UltraLight 卡写入一个 4 字节的数据。此命令只对 UltraLight 有效。对 UltraLight 进行读操作与 mifare1 一样。该函数在写入数据后会立即进行读操作并进行数据写入正确性判断。
数据块格式描述：
主机→TX522 命令模式）：
SEQNR： 0 （可自定义）
COMMAND：0x47
LENGTH： 5
DATA[0]： Block
DATA[1]： 所要写的第一个字节
：
DATA[4]： 所要写的最后一个字节
例如：往 Ultralight 卡块 1 写入数据的数据帧

| STX | SEQNR | CMD | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x47 | 0x05 | 0x01 4 字节数据 | 0xXX | 0x03 |

TX522→主机（响应模式）：

SEQNR： 0

STATUS： OK， QUIT， NO_TAG_ERR， NOT_AUTH_ERR， WRITE_ERR，BIT_COUNT_ERR，COMM_ERR 中的某一个

LENGTH： 0

DATA[0]： 无

例如：数据帧

| STX | SEQNR | Status | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x00 | 0x00 | none | 0xFF | 0x03 |

### 5.3.31 装载密钥—TX_M1LoadKey

函数原型：uchar TX_M1LoadKey(uchar KeyAB,uchar KeySector,uchar idata *Key)

输入参数：KeyAB：密钥类型（1 字节）。可取值为 KeyAB＝0x00（KEYA）—密钥 A，或 KeyAB＝0x04（KEYB）—密钥 B。

KeySector：模块内的密钥区号（1 字节）：取值范围 0～15

*Key：需要装载到模块内密钥区的密钥（6 字节）

输出参数：无

函数返回： TX800B 执行命令后的状态，可能的状态值如下：OK，QUIT，AUTH_ERR，COMM_ERR。

功能描述：此函数的作用是将指定的密码（*Key）装载到模块内 E2PROM 指定的密钥区（KeySector），并非改变 Mifare1 卡内扇区的密码。本函数只对模块进行操作，模块与卡之间没有数据传输。装置的密钥掉电不丢失，因此只用装载一次就可以。考虑到系统安全性，装载密钥过程可单独进行，用户程序中可不出现该命令。

模块内有 16 个密码区（区号 0——15），称它为密钥区号 KeySector。每个区分密钥 A（0x60）和密钥 B（0x61）两个，总共 32 个密码。装载成功后，可用该密钥对 Mifare1 卡进行验证。

在 M1 卡中也有 16 个存储区，称它为扇区号 Sector。若要改变 Mifare1 卡内的密钥，可在用原密码验证通过后，直接用写块数据 TX_Write()函数，将密码块改写。Mifare 卡出厂后的初始密钥为 6 个 FFH，A 和 B 密钥都一样。

数据块格式描述：

主机→TX800 命令模式）：

SEQNR： 0 （可自定义）

COMMAND：0x4C

LENGTH： 8

DATA[0]： KeyAB

DATA[1]： KeySector

DATA[2]： Key[0]

…

DATA[7]： Key[5]

例如：往密匙 0 区装载密匙 A：0xff 0xff 0xff 0xff 0xff 0xff 的数据帧

| STX | SEQNR | CMD | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x4c | 0x08 | 0x00 0x00 0xff 0xff 0xff 0xff 0xff 0xff | 0xbb | 0x03 |

TX800→主机（响应模式）：

SEQNR： 0

STATUS： OK，QUIT，AUTH_ERR，COMM_ERR 中的某一个

LENGTH： 0

DATA[0]： 无

例如：数据帧

| STX | SEQNR | Status | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x00 | 0x00 | none | 0xFF | 0x03 |

### 5.3.32 证实 2—TX_M1Auth

函数原型：uchar TX_ M1Auth(uchar KeyAB, uchar Sector, uchar KeySector);

输入参数：KeyAB：密钥类型（1 字节）。可取值为 KeyAB＝0x00（KEYA），利用密钥 A 进行验证，或 KeyAB＝0x04（KEYB），利用密钥 B 进行验证。

Sector：所要验证的卡扇区号（也即将要访问的卡的扇区号），取值范围 0～39，能用于 S70 卡。

KeySector：模块内的密钥区号（1 字节）：取值范围 0～15。

输出参数： 无

函数返回：TX800B 执行命令后的状态，可能的状态值如下：OK，QUIT，NO_TAG_ERR，PARITY_ERR，BIT_COUNT_ERR，COMM_ERR。

功能描述：使用模块内部密钥区中 KeySector 中的密码对指定的卡的扇区 Sector 进行验证，若卡 Sector 区中的密码与存储在模块内 KeySector 中的密码相同，则验证成功，返回 OK。

该函数也依赖与 TX_M1LoadKey 函数曾经成功执行过，因为模块内部密码区（KeySector）中的密码要由 TX_M1LoadKey 函数事先装载。该函数适用于对于所有卡来说密码相同的应用，密钥的装载可以在一个安全的场合一次性装入。

数据块格式描述：

主机→TX800 命令模式）：

SEQNR： 0 （可自定义）

COMMAND：0x72

LENGTH： 3

DATA[0]： KeyAB

DATA[1]： Sector

DATA[2]： KeySector

例如：用密匙 0 区的密匙 A 证实卡的扇区 0 的数据帧

| STX | SEQNR | CMD | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x72 | 0x03 | 0x00 0x00 0x00 | 0x8e | 0x03 |

TX800→主机（响应模式）：

SEQNR： 0

STATUS： OK， QUIT， NO_TAG_ERR， PARITY_ERR， BIT_COUNT_ERR，COMM_ERR 中的某一个

LENGTH： 0

DATA[0]： 无

例如：数据帧

| STX | SEQNR | Status | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x00 | 0x00 | none | 0xFF | 0x03 |

### 5.3.33 带验证的写—TX_M1WriteAuth

函数原型：uchar TX_ M1WriteAuth(uchar KeyAB, uchar KeySector ,uchar Block, uchar idata *Data)

输入参数：

(1) KEYAB--密钥 AB（1 字节）： 0x00——密钥 A
0x04——密钥 B

(2) KeySector：模块内的密钥区号（1 字节）：取值范围 0～15。

(3) Block--卡块号（1 字节）： S50：1~63
S70：1~255

(4) *Data：16 字节数据指针，Data 为写入的 16 字节数据的首地址。

输出参数： 无

函数返回：执行后可能返回：OK，QUIT，NO_TAG_ERR，CRC_ERR，NOT_AUTH_ERR，PARITY_ERR，BIT_COUNT_ERR,COMM_ERR，CHK_WR_FAILED，CHK_WR_COMP_ERR 中的某一个。

功能描述：该函数在 GetCardSnr 后执行，先对卡内某一块进行验证，成功后对指定块进行写操作（只要访问条件允许），其中包括位于扇区尾的密码块，这是更改密码的唯一方法。该函数在写入数据后会立即进行读操作并进行数据写入正确性判断。

数据块格式描述：

主机→TX800 命令模式）：

SEQNR： 0 （可自定义）

COMMAND：0x11

LENGTH： 19

DATA[0]： KEYAB

DATA[1]： KeySector

DATA[2]： Block

DATA[3]： 所要写的第一个字节

：

DATA[18]： 所要写的最后一个字节

例如：使用模块内部密匙 0 区的密匙 A 进行验证，往块 2 写入数据的数据帧

| STX | SEQNR | CMD | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x11 | 0x13 | 0x00 0x00 0x02 16 字节数据 | 0xXX | 0x03 |

TX800→主机（响应模式）：

SEQNR： 0

STATUS： OK， QUIT， NO_TAG_ERR， NOT_AUTH_ERR， WRITE_ERR，BIT_COUNT_ERR，COMM_ERR 中的某一个

LENGTH： 0

DATA[0]： 无

例如：数据帧

| STX | SEQNR | Status | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x00 | 0x00 | none | 0xFF | 0x03 |

补充说明：

注意：在将数据写到卡片上的某一扇区时，一定要小心。因为有些 block 中存储了密码数据以及存储允许使能数据。特别是每一个扇区的 Block3 中存放了该扇区的存取条件，包含有 KEYA，KEYB 及该扇区的控制字。Mifare 1 卡片出厂时的 Block3 有缺省值，为：“a0a1a2a3a4a5ff078069b0b1b2b3b4b5”,共 16 个 Bytes。

涉及 Mifare 1 卡片的存储结构等信息，请参考 Mifare 1 卡片数据手册。

程序员在使用 Mifare 1 卡片做应用时，一定要清清楚楚记住每一个扇区的 Block3 的数据，这样也就记住了扇区的密码和存取控制字。否则，扇区的存储空间将不执行 Read/Write 等操作而失效。

任何试图用任何方式来读写不知密码的卡片或某一扇区都是徒劳无益的。

卡片应放在安全的地方，即不要放在离模块天线较近的地方。因为当模块对其它卡片执行某些指令时，有可能无意间对这一卡片进行了读/写等操作，从而操作卡片的失效。

### 5.3.34 带验证的读—TX_M1ReadAuth

函数原型：uchar TX_M1ReadAuth(uchar KeyAB, uchar KeySector ,uchar Block,uchar idata *Data)

输入参数：

(1) KEYAB--密钥 AB（1 字节）： 0x00——密钥 A
0x04——密钥 B

(2) KeySector：模块内的密钥区号（1 字节）：取值范围 0～15。

(3) Block--卡块号（1 字节）： S50：1~63
S70：1~255

输出参数：*Data：Data 为读回 16 字节数据的首地址。

函数返回：执行后可能返回：OK，QUIT，NO_TAG_ERR，CRC_ERR，NOT_AUTH_ERR，PARITY_ERR，BIT_COUNT_ERR，COMM_ERR 中的某一个。

功能描述：该函数在 GetCardSnr 后执行，先对卡内某一块进行验证，成功后读 Mifare 卡中相应块的数据。密码数据不能被读取。

数据块格式描述：

主机→TX800 命令模式）：

SEQNR： 0 （可自定义）

COMMAND：0x12

LENGTH： 3

DATA[0]： KEYAB

DATA[1]： KeySector

DATA[2]： Block

例如：使用模块内部密匙 0 区的密匙 A 进行验证，读取块 2 的数据的数据帧

| STX | SEQNR | CMD | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x12 | 0x03 | 0x00，0x00，0x02 | 0xEC | 0x03 |

TX800→主机（响应模式）：

SEQNR： 0

STATUS： OK，QUIT，NO_TAG_ERR，CRC_ERR，NOT_AUTH_ERR，PARITY_ERR，BIT_COUNT_ERR，COMM_ERR 中的某一个

LENGTH： 16

DATA[0]： 所访问块的第一个字节

：

DATA[15]： 所访问块的最后一个字节

例如：数据帧

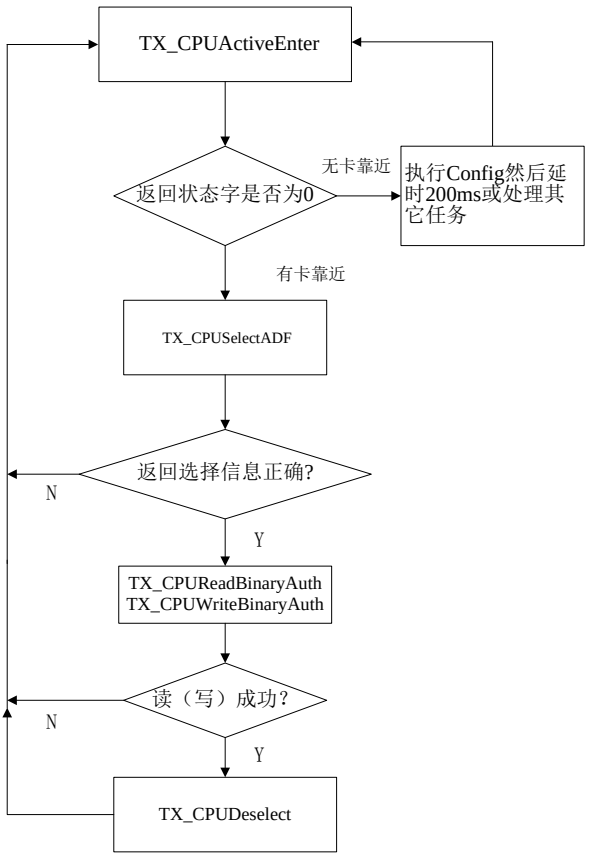
| STX | SEQNR | Status | Length | DATA | BCC | ETX |
|---|---|---|---|---|---|---|
| 0x20 | 0x00 | 0x00 | 0x10 | 16 字节数据 | 0xXX | 0x03 |

## 5.4 函数调用描述

### 5.4.1 CPU 卡操作流程

对于没有经过初始化的卡片，应采用配套的专用发卡器的对其初始化操作；对于没有发卡器的用户也可以调用模块提供的 CPU 卡初始化命令来完成卡片的初始化工作。为了保证模块内部保存的密钥是跟使用者需要设定的密钥一致，首先得确保 TX_CPULoadKey 函数正确执行。考虑到安全性，要保证密钥在产品使用过程中不在通信过程中传输，可以将 TX_CPULoadKey 在用户程序以外提前进行，可以使用我们公司提供的密钥装载工具在 PC 机上完成。FM1028 卡默认密钥为 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF，默认密钥也要执行装载操作。

对于已经初始化好的 cpu 卡片操作，可以采用如下方法进行读写。

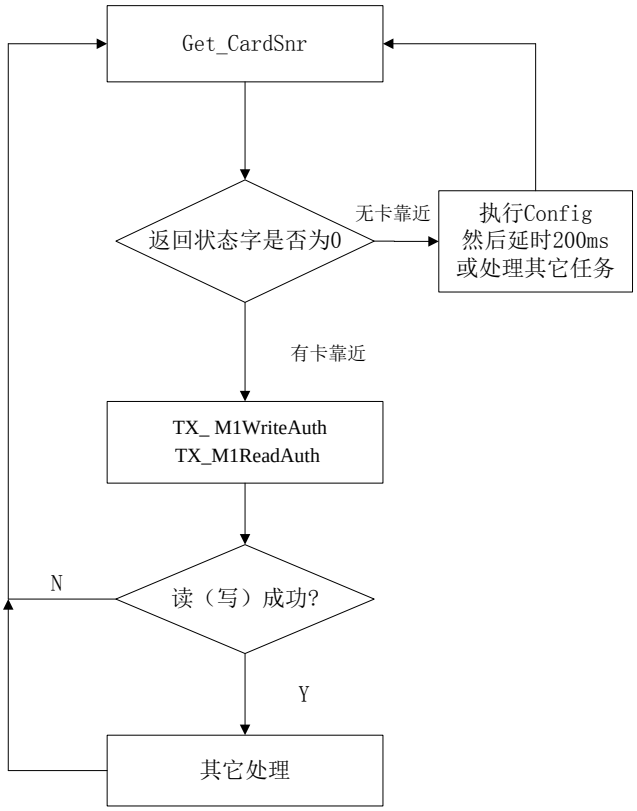



### 5.4.2 M1 操作流程

要确保 TX_M1LoadKey 函数正确执行。考虑到安全性，要保证密钥在产品使用过程中不在通信过程中传输，可以将 LoadKey 在用户程序以外提前进行，可以使用我们公司提供的密钥装载工具在 PC 机上完成。Mifare 卡默认密钥为 0xffffffffffff，默认密钥也要执行装载操作。

关于对 Mifare 卡片操作，有两种方法。

方法一，用户主程序定时（比方间隔 250ms）循环发送 GetCardSnr，依据返回值判断是否有卡片靠近。如果有卡靠近再做相应处理。




## 5.5 典型应用流程图

### 5.5.1 CPU 卡应用流程

**（一）、CPU 卡发卡操作**

CPU 卡有别于 M1 卡，没有固定的卡文件结构，出厂时是只具有根目录的空白卡，没有任何可以读写文件，必须发卡后才能读进行写操作。在此可以采用两种对 CPU 卡法卡的方法。简单的应用建议采用方法一进行发卡 ，使用者就不需要去了解 CPU 卡的 COS 命令；对于特殊应用场合，使用者想建立自己的应用文件结构则必须采用 COS 命令发卡，即发卡操作方法二。

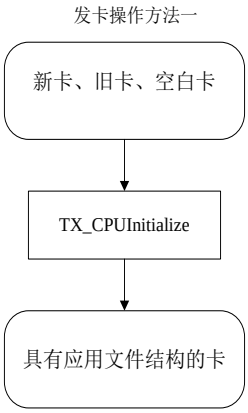



图 7

发卡操作方法二

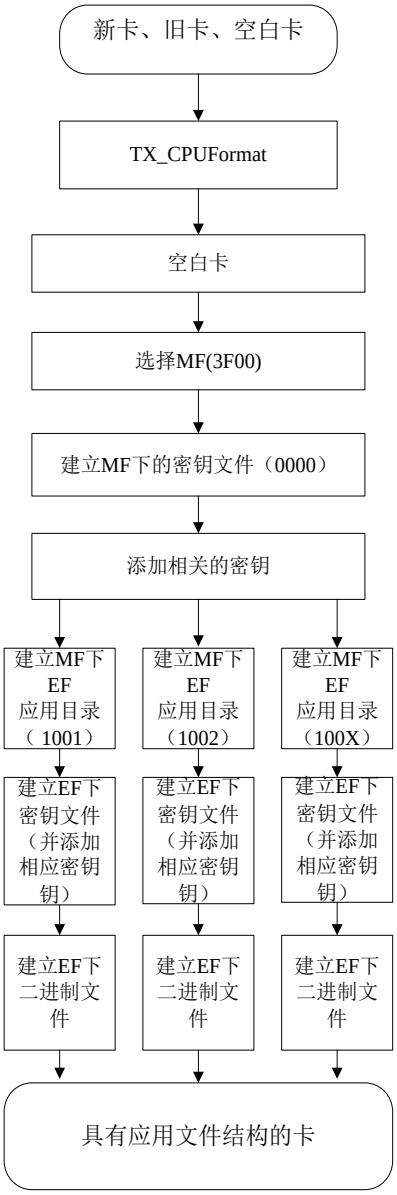



图 8

**（二）、CPU 卡读写操作**

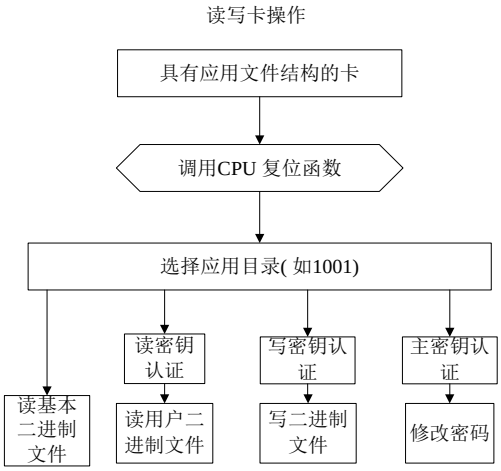



图 9

### 5.5.2 M1 卡应用流程

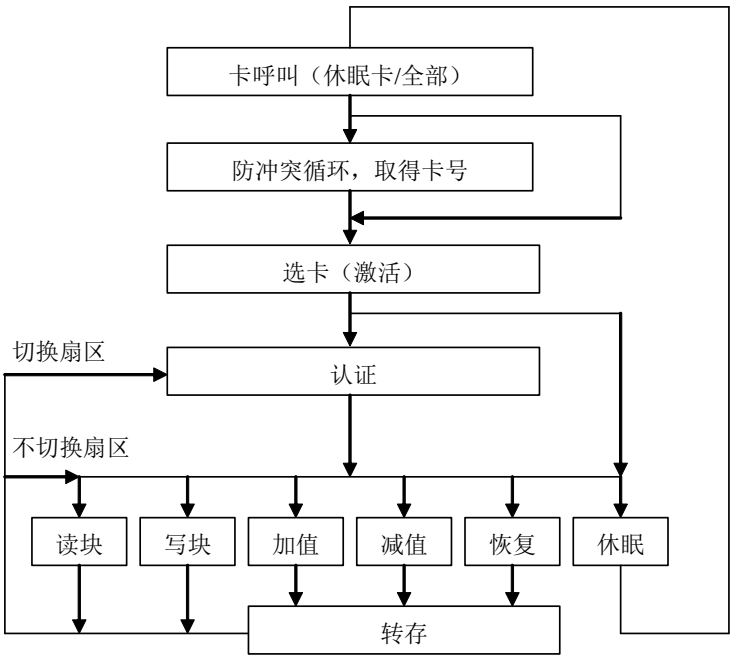



图 10

# 6. 使用串口调试助手开发调试 TX800T 模块

## 6.1 软硬件准备

1. TX800T 外接 RS232 转换
2. 串口电缆
3. 5V 电源
4. 5 根连线，用于连接 TX800T 和串口电缆、电源
5. Mifare 卡一张、FM1208 卡一张
6. 串口调试软件
7. 串口监视软件（选）
8. PC 机

## 6.2 电路连接

请按照下图所示连接 TX800 和 PC 机的串口。

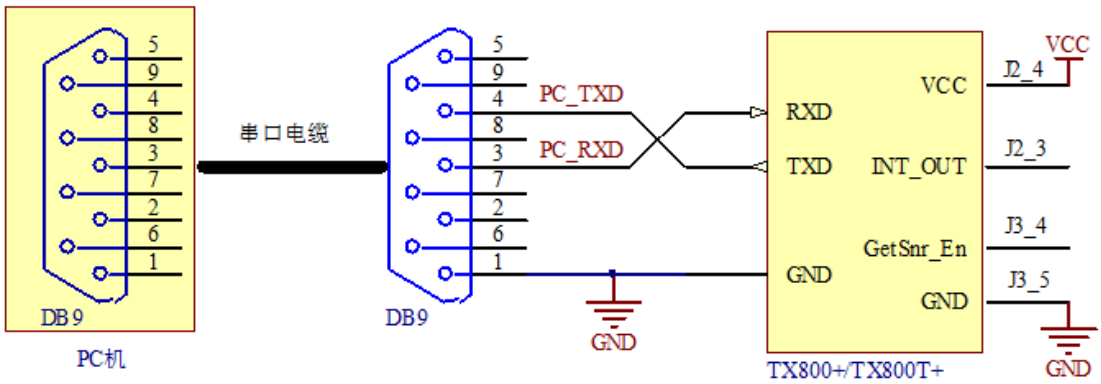



图 11 硬件接线图

## 6.3 设置串口调试软件

为了方便调试我们提供一款跟 TX800 配套的串口调试工具（TX 串口调试小助手）使用者可以参照按照下图所示设置串口调试软件。

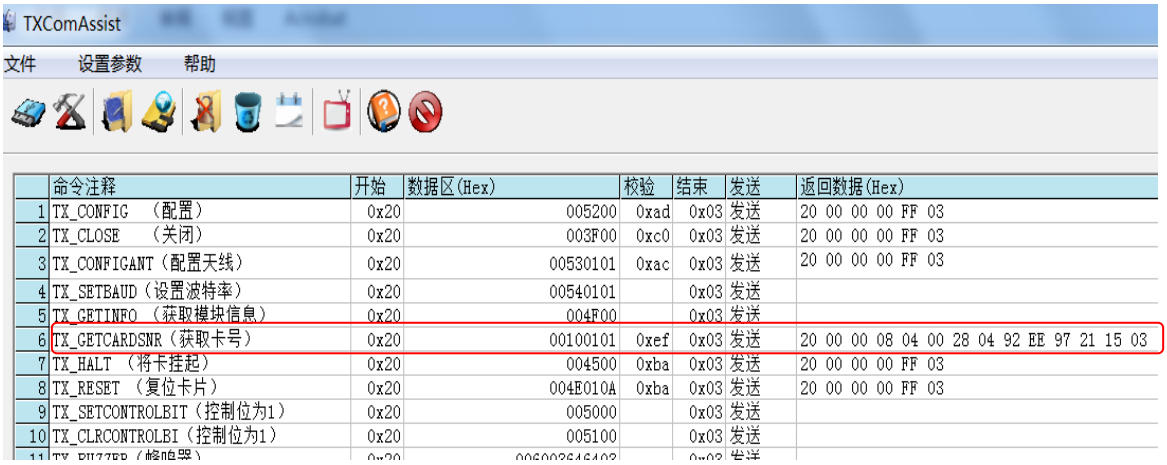

图 12 串口调试软件的设置

## 6.4 向 TX800T+发送命令

发送命令的格式为：起始符(STX: 0x20) + 包序号(一般设 00 即可) + 命令字 + 数据长度 + 数据 + BCC 校验 + 结束符(EXT: 03)。详细说明见前文。

《TX800B 命令集_V1.00.txt》列出了一些常用的命令，一些可以直接拷贝到串口调试软件的发送窗口中直接进行发送。有一些需要用户进行一定的修改，例如使用的卡片不同，发送选择命令（select）时应填入相应的卡号。注意如果数据有些不同，则 BCC 校验也不同。

举例：TX_GetCardSnr 激活并获取卡号，发送命令如下图所示。




| | 命令注释 | 开始 | 数据区(Hex) | 校验 | 结束 | 发送 | 返回数据(Hex) |
|---|---|---|---|---|---|---|---|
| 1 | TX_CONFIG （配置） | 0x20 | 005200 | 0xad | 0x03 | 发送 | 20 00 00 00 FF 03 |
| 2 | TX_CLOSE （关闭） | 0x20 | 003F00 | 0xc0 | 0x03 | 发送 | 20 00 00 00 FF 03 |
| 3 | TX_CONFIGANT（配置天线） | 0x20 | 00530101 | 0xac | 0x03 | 发送 | 20 00 00 00 FF 03 |
| 4 | TX_SETBAUD（设置波特率） | 0x20 | 00540101 | | 0x03 | 发送 | |
| 5 | TX_GETINFO （获取模块信息） | 0x20 | 004F00 | | 0x03 | 发送 | |
| 6 | TX_GETCARDSNR（获取卡号） | 0x20 | 00100101 | 0xef | 0x03 | 发送 | 20 00 00 08 04 00 28 04 92 EE 97 21 15 03 |
| 7 | TX_HALT （将卡挂起） | 0x20 | 004500 | 0xba | 0x03 | 发送 | 20 00 00 00 FF 03 |
| 8 | TX_RESET （复位卡片） | 0x20 | 004E010A | 0xba | 0x03 | 发送 | 20 00 00 00 FF 03 |
| 9 | TX_SETCONTROLBIT（控制位为1） | 0x20 | 005000 | | 0x03 | 发送 | |
| 10 | TX_CLRCONTROLBI（控制位为1） | 0x20 | 005100 | | 0x03 | 发送 | |

图 13 串口调试软件的设置

将 FM1028 卡靠近模块，发送完 2000100101 命令后，串口接收区就会显示接收到的数据。图中接收区中的数据解释如下：

20 00 00 08 04 00 28 04 92 EE 97 21 15 03 是自动寻卡后的发回的数据。20 是起始符，第一个 00 是包号，最后一个字节是结束符，第二个 00 是状态表示 OK， 08 是本数据块长度，04 00 是请求的应答即卡片类型，表示兼容 S50 卡， 28 是选择的应答，表示选择卡成功， 04 是卡号长度， 92 EE 97 21 是 4 字节卡号，15 是 BCC 校验。最后一个字节 03 是本帧数据的结束符。

# 7. 免责声明

- **开发预备知识**

TX®系列产品将提供尽可能全面的开发模板、驱动程序及其应用说明文档以方便用户使用，但也需要用户熟悉自己设计产品所采用的硬件平台及相关 C 语言的知识。

- **EMI 与 EMC**

TX®系列模块机械结构决定了其 EMI 性能必然与一体化电路设计有所差异。TX®系列模块的 EMI 能满足绝大部分应用场合，用户如有特殊要求，必须事先与我们协商。

TX®系列模块的 EMC 性能与用户底板的设计密切相关，尤其是电源电路、I/O 隔离、复位电路，用户在设计底板时必须充分考虑以上因素。我们将努力完善 TX®系列模块的电磁兼容特性，但不对用户最终应用产品 EMC 性能提供任何保证。

- **修改文档的权利**

本公司保留任何时候在不事先声明的情况下对 TX®系列产品相关文档的修改权力。

- **ESD 静电放电保护**


TX®系列产品部分元器件内置 ESD 保护电路，但在使用环境恶劣的场合，依然建议用户在设计底板时提供 ESD 保护措施，特别是电源与 I/O 设计，以保证产品的稳定运行。安装 TX®系列产品，为确保安全请先将积累在身体上的静电释放，例如佩戴可靠接地的静电环，触摸接入大地的自来水管等。

# 8. 修订历史

| 版本 | 日期 | 原因 |
| --- | --- | --- |
| V1.00 | 2012/05/03 | 创建文档。 |