



---

# 友我科技高频 RFID 读写器 Linux 接口

---

用户手册 V1.0



2019-11

北京友我科技有限公司

## 目录

|        |                                  |    |
|--------|----------------------------------|----|
| 1      | 概述.....                          | 5  |
| 2      | 支持卡片类型.....                      | 5  |
| 2.1    | ISO14443A .....                  | 5  |
| 2.2    | ISO14443B.....                   | 5  |
| 2.3    | ISO15693 .....                   | 6  |
| 2.4    | NFC Type1~Type4 .....            | 6  |
| 2.5    | ISO7816 .....                    | 6  |
| 3      | 二次开发指南.....                      | 7  |
| 3.1    | 动态库及读写器相关函数.....                 | 7  |
| 3.1.1  | 读取库函数内部版本号 .....                 | 7  |
| 3.1.2  | DES 加解密函数 .....                  | 7  |
| 3.1.3  | 3DES 加解密函数 .....                 | 7  |
| 3.1.4  | 带向量的 3DES 加解密函数 .....            | 8  |
| 3.1.5  | 初始化串口.....                       | 8  |
| 3.1.6  | 释放串口.....                        | 9  |
| 3.1.7  | USB 无驱读写器，初始化 USB .....          | 9  |
| 3.1.8  | USB 无驱读写器，释放 USB .....           | 9  |
| 3.1.9  | 设置设备标识.....                      | 9  |
| 3.1.10 | 查询设备标识.....                      | 10 |
| 3.1.11 | 读取读卡器内部版本号 .....                 | 10 |
| 3.1.12 | 查询读写器产品序列号 .....                 | 10 |
| 3.1.13 | 蜂鸣器控制函数.....                     | 11 |
| 3.1.14 | LED 指示灯控制 .....                  | 11 |
| 3.1.15 | 设置天线的状态.....                     | 12 |
| 3.1.16 | 设置寻卡模式.....                      | 12 |
| 3.1.17 | 设置寻卡模式（仅 YW-607，YW-627 系列） ..... | 13 |
| 3.2    | ISO7816 标准接触卡 .....              | 15 |
| 3.2.1  | SAM 卡复位 .....                    | 15 |
| 3.2.2  | SAM 卡执行 COS 命令 .....             | 16 |
| 3.2.3  | SAM 卡 PPS 波特率设置.....             | 16 |
| 3.3    | ISO14443 TypeA 相关函数.....         | 18 |
| 3.3.1  | 寻卡.....                          | 19 |
| 3.3.2  | 防碰撞并且选定一张卡.....                  | 19 |
| 3.3.3  | 卡片休眠.....                        | 20 |
| 3.3.4  | 下载密钥到读卡器.....                    | 20 |
| 3.3.5  | 用下载的密钥授权卡片 .....                 | 21 |
| 3.3.6  | 用当前密钥授权卡片 .....                  | 21 |
| 3.3.7  | 读取一块数据.....                      | 22 |
| 3.3.8  | 写入一块数据.....                      | 22 |
| 3.3.9  | 将某一块初始化为钱包.....                  | 23 |
| 3.3.10 | 读取钱包值.....                       | 23 |
| 3.3.11 | 钱包扣款.....                        | 24 |
| 3.3.12 | 钱包充值.....                        | 24 |

|        |                                                 |    |
|--------|-------------------------------------------------|----|
| 3.3.13 | 钱包 Restore 命令 .....                             | 25 |
| 3.3.14 | 钱包 Transfer 命令 .....                            | 25 |
| 3.3.15 | 读取多块数据 .....                                    | 25 |
| 3.3.16 | 写多块数据 .....                                     | 26 |
| 3.4    | ISO14443 TYPEA UltraLight 卡操作函数 .....           | 27 |
| 3.4.1  | UltraLight 卡读块数据 .....                          | 27 |
| 3.4.2  | UltraLight 卡写块数据 .....                          | 28 |
| 3.5    | ISO14443-4 Type A CPU 卡操作函数 .....               | 29 |
| 3.5.1  | ISO14443-4 TypeA CPU 卡复位 .....                  | 29 |
| 3.5.2  | ISO14443-4 TypeA CPU 卡执行 COS 命令 .....           | 30 |
| 3.6    | Mifare Plus 卡操作函数 .....                         | 31 |
| 3.6.1  | Mifare Plus 卡 Level 0 级写数据 .....                | 34 |
| 3.6.2  | Mifare Plus 卡从 Level 0 级向 Level 1 或 3 级切换 ..... | 35 |
| 3.6.3  | Mifare Plus 卡从 Level1 或 Level2 向高级切换 .....      | 35 |
| 3.6.4  | Mifare Plus 卡 Level 3 级授权 .....                 | 36 |
| 3.6.5  | Mifare Plus 卡 Level 3 级读块数据 .....               | 36 |
| 3.6.6  | Mifare Plus 卡 Level 3 级写块数据 .....               | 37 |
| 3.6.7  | Mifare Plus 卡 Level 3 级将某一块初始化为钱包 .....         | 37 |
| 3.6.8  | Mifare Plus 卡 Level 3 级读取钱包值 .....              | 38 |
| 3.6.9  | Mifare Plus 卡 Level 3 级钱包充值 .....               | 38 |
| 3.6.10 | Mifare Plus 卡 Level 3 级钱包扣款 .....               | 38 |
| 3.6.11 | Mifare Plus 卡 Level 3 级备份钱包 .....               | 39 |
| 3.6.12 | Mifare Plus 卡 Level 3 级通用读写第一次授权 .....          | 39 |
| 3.6.13 | Mifare Plus 卡 Level 3 级通用读写第二次授权 .....          | 40 |
| 3.6.14 | Mifare Plus 卡 Level 3 级通用读块 .....               | 40 |
| 3.6.15 | Mifare Plus 卡 Level 3 级通用写块 .....               | 41 |
| 3.7    | ISO14443 TypeB .....                            | 42 |
| 3.7.1  | ISO14443 Type B 卡复位 .....                       | 45 |
| 3.7.2  | ISO14443 Type B 卡复位（扩展函数） .....                 | 45 |
| 3.7.3  | ISO14443 Type B 卡 COS 命令 .....                  | 46 |
| 3.7.4  | ISO14443 Type B 卡 Halt .....                    | 46 |
| 3.7.5  | 获取中国居民身份证卡号 .....                               | 47 |
| 3.7.6  | AT88RF020/080 密钥认证 .....                        | 47 |
| 3.7.7  | AT88RF020/080 读数据块 .....                        | 47 |
| 3.7.8  | AT88RF020/080 写数据块 .....                        | 48 |
| 3.7.9  | AT88RF020/080 锁数据块 .....                        | 48 |
| 3.7.10 | AT88RF020/080 读取计数 .....                        | 49 |
| 3.7.11 | AT88RF020/080 Deslect 当前卡片 .....                | 49 |
| 3.7.12 | AT88RF020/080 读取多块 .....                        | 49 |
| 3.7.13 | AT88RF020/080 写多块 .....                         | 50 |
| 3.7.14 | 激活 S T 系列卡 .....                                | 50 |
| 3.7.15 | S T 卡休眠 .....                                   | 51 |
| 3.7.16 | SR176 卡获取 UID .....                             | 51 |
| 3.7.17 | SR176 卡读数据块 .....                               | 51 |

|        |                              |    |
|--------|------------------------------|----|
| 3.7.18 | SR176 卡写数据块.....             | 52 |
| 3.7.19 | SR176 卡锁数据块.....             | 52 |
| 3.7.20 | SR176 卡获取锁块状态.....           | 53 |
| 3.7.21 | SR176 卡读取多个数据块.....          | 53 |
| 3.7.22 | SR176 卡写多个数据块.....           | 54 |
| 3.7.23 | SRI/SLI 卡获取 UID .....        | 54 |
| 3.7.24 | SRI/SLI 卡读数据块.....           | 55 |
| 3.7.25 | SRI/SLI 卡写数据块.....           | 55 |
| 3.7.26 | SRI/SLI 卡锁数据块.....           | 55 |
| 3.7.27 | SRI/SLI 卡获取锁块状态.....         | 56 |
| 3.7.28 | SRI/SLI 卡读多块 .....           | 56 |
| 3.7.29 | SRI/SLI 卡写多块 .....           | 57 |
| 3.7.30 | SRIX4K 卡密钥认证.....            | 57 |
| 3.8    | ISO15693 标签 .....            | 58 |
| 3.8.1  | ISO15693 标签 Inventory .....  | 59 |
| 3.8.2  | ISO15693 标签 Stay Quiet ..... | 60 |
| 3.8.3  | ISO15693 标签选卡.....           | 60 |
| 3.8.4  | ISO15693 标签复位.....           | 61 |
| 3.8.5  | ISO15693 标签读块.....           | 62 |
| 3.8.6  | ISO15693 标签写块.....           | 63 |
| 3.8.7  | ISO15693 标签锁块.....           | 64 |
| 3.8.8  | ISO15693 标签写 AFI .....       | 65 |
| 3.8.9  | ISO15693 标签锁 AFI .....       | 65 |
| 3.8.10 | ISO15693 标签写 DSFID.....      | 66 |
| 3.8.11 | ISO15693 标签锁 DSFID.....      | 67 |
| 3.8.12 | 获取 ISO15693 标签系统信息.....      | 68 |
| 3.8.13 | 获取 ISO15693 标签块安全信息.....     | 69 |
| 3.8.14 | ISO15693 标签防冲突寻多张卡.....      | 70 |
| 3.9    | NFC-Type1 TOPAZ 标签 .....     | 70 |
| 3.9.1  | TOPAZ 卡片获取卡号 .....           | 71 |
| 3.9.2  | TOPAZ 卡片读所有数据 .....          | 72 |
| 3.9.3  | TOPAZ 卡片读单个字节 .....          | 72 |
| 3.9.4  | TOPAZ 卡片先擦除再写单个字节.....       | 73 |
| 3.9.5  | TOPAZ 卡片不带擦除写单个字节.....       | 73 |
| 3.9.6  | TOPAZ 卡片读 Segment .....      | 74 |
| 3.9.7  | TOPAZ 卡片读块 .....             | 74 |
| 3.9.8  | TOPAZ 卡片先擦除再写块 .....         | 75 |
| 3.9.9  | TOPAZ 卡片不带擦除写块 .....         | 75 |
| 3.10   | NFC-Type2 NTAG 系列标签.....     | 76 |
| 3.10.1 | NTAG 卡片获取版本信息 .....          | 77 |
| 3.10.2 | NTAG 卡片读数据 .....             | 78 |
| 3.10.3 | NTAG 卡片快速读数据 .....           | 78 |
| 3.10.4 | NTAG 卡片写数据 .....             | 79 |
| 3.10.5 | NTAG 卡片兼容写数据 .....           | 79 |

---

|        |                          |    |
|--------|--------------------------|----|
| 3.10.6 | NTAG 卡片读计数器 .....        | 80 |
| 3.10.7 | NTAG 卡片密码授权 .....        | 80 |
| 3.10.8 | NTAG 卡片读签名 .....         | 81 |
| 3.11   | NFC-Type3 Felica 卡 ..... | 82 |
| 3.11.1 | Felica 卡片寻卡 .....        | 82 |
| 3.11.2 | Felica 卡片不加密读操作 .....    | 83 |
| 3.11.3 | Felica 卡片不加密写操作 .....    | 84 |
| 4      | 订购方式.....                | 85 |
| 附录 A   | .....                    | 86 |

# 1 概述

友我科技高频 RFID 读写器 Linux 开发接口, 库文件名称: libyw60x.so, 支持的读写器有: SDT-HA, YW-605 系列, YW-607 系列, YW-608, YW-610, YW-627, YW-1607 等等.

支持的接口: 串口, 485 和 USB 免驱动接口.

## 2 支持卡片类型

### 2.1 ISO14443A

- ◆ Mifare One S50/70
- ◆ Mifare One Mini
- ◆ Mifare Ultra Light
- ◆ Mifare Desfire
- ◆ Mifare Plus(支持安全层级: Level0~level3)
- ◆ Jcop 41 only the (Mifare 1K & 4K compatible)
- ◆ Jewel
- ◆ ISO14443-4 (T=CL) TYPE A dual interface CPU Card

### 2.2 ISO14443B

- ◆ AT88RF020
- ◆ AT88RF080
- ◆ SR176
- ◆ SRI512/1k/2k/4k
- ◆ SLIX4K
- ◆ ISO14443-4 (T=CL) TYPE B dual interface CPU Card

## 2.3 ISO15693

- ◆ I.CODE SLI
- ◆ Tag-it HF-I
- ◆ SRF55VxxP/ SRF55VxxS
- ◆ LRI12/64/128/2k
- ◆ EM4135
- ◆ 其它兼容的 ISO15693 标签

## 2.4 NFC Type1~Type4

- ◆ Type1 :ISO14443 TypeA, Topaz96/512
- ◆ Type2 :ISO14443 TypeA, Mifare ultralight, NTAG203/210/212/213/216
- ◆ Type3 :Sony Felica
- ◆ Type 4 :ISO14443 TYPEA/B, Mifare desfire 等

## 2.5 ISO7816

- ◆ ISO7816 T=0/T=1 接触非接触卡, 支持最大速度 230400bps 和最大 APDU FSD = 256 字节. 支持A类卡和B类卡操作.

## 3 二次开发指南

库函数，C语言版.

### 3.1 动态库及读写器相关函数

#### 3.1.1 读取库函数内部版本号

函数原形: `int YW_GetDLLVersion(void);`

参数列表: 无

返回值: 大于0为版本号, <=0为错误

#### 3.1.2 DES 加解密函数

函数原形: `int DES(unsigned char cModel, unsigned char *pkey, unsigned char *in, unsigned char *out);`

参数列表:

| 参数     | 类型              | 方向  | 含义                         |
|--------|-----------------|-----|----------------------------|
| cModel | unsigned char   | IN  | 加解密模式选择:<br>0->加密<br>1->解密 |
| pkey   | unsigned char * | IN  | 加解密密钥, 8个字节                |
| in     | unsigned char * | IN  | 原始数据, 8个字节                 |
| out    | unsigned char * | OUT | 加解密后的数据, 8个字节              |

返回值: 1成功, <=0失败

#### 3.1.3 3DES 加解密函数

函数原形: `int DES3(unsigned char cModel, unsigned char *pKey, unsigned char *In, unsigned char *Out);`

参数列表:

| 参数     | 类型             | 方向  | 含义                               |
|--------|----------------|-----|----------------------------------|
| cModel | unsigned char  | IN  | 加解密模式选择：<br>0x00->加密<br>0x01->解密 |
| pkey   | unsigned char* | IN  | 加解密密钥，16个字节                      |
| in     | unsigned char* | IN  | 原始数据，8个字节                        |
| out    | unsigned char* | OUT | 加解密后的数据，8个字节                     |

**返回值：**1成功，<=0失败

### 3.1.4 带向量的 3DES 加解密函数

**函数原形：**`int DES3_CBC(unsigned char cModel, unsigned char *pKey, unsigned char *In, unsigned char *Out, unsigned char *pIV);`

**参数列表：**

| 参数     | 类型             | 方向  | 含义                               |
|--------|----------------|-----|----------------------------------|
| cModel | unsigned char  | IN  | 加解密模式选择：<br>0x00->加密<br>0x01->解密 |
| pkey   | unsigned char* | IN  | 加解密密钥，16个字节                      |
| in     | unsigned char* | IN  | 原始数据，8个字节                        |
| out    | unsigned char* | out | 加解密后的数据，8个字节                     |
| pIV    | unsigned char* | IN  | 加解密向量，8个字节                       |

**返回值：**1成功，<=0失败

### 3.1.5 初始化串口

**函数原形：**`int stdcall YW_ComInitial(int PortIndex, int Bound);`

**参数列表：**

| 参数 | 类型 | 方向 | 含义 |
|----|----|----|----|
|----|----|----|----|

|           |     |    |                                      |
|-----------|-----|----|--------------------------------------|
| PortIndex | int | IN | 串口号，与<br>/dev/ttySn ,n是多少，<br>这个就是多少 |
| Bound     | int | IN | 通信波特率，<br>2400—115200，默认为<br>19200   |

返回值：1成功，0失败

### 3.1.6 释放串口

函数原形：int stdcall YW\_ComFree(void);

参数列表：无

返回值：1成功，0失败

### 3.1.7 USB 无驱读写器，初始化 USB

函数原形：int YW\_USBHIDInitial(void);

参数列表：无

返回值：1成功，<=0失败

### 3.1.8 USB 无驱读写器，释放 USB

函数原形：int YW\_USBHIDFree(void);

参数列表：无

返回值：1成功，<=0失败

### 3.1.9 设置设备标识

函数原形：int YW\_SetReaderID(int OldID, int NewID);

参数列表：

| 参数    | 类型  | 方向  | 含义                          |
|-------|-----|-----|-----------------------------|
| OldID | int | IN  | 旧的设备标示ID，范围0x0000-0xFFFF    |
| NewID | int | OUT | 修改成新的设备标示ID，范围0x0000-0xFFFF |

返回值：1成功，<=0失败

### 3.1.10 查询设备标识

函数原形：int YW\_GetReaderID(int ReaderID);

参数列表：

| 参数       | 类型  | 方向 | 含义                                           |
|----------|-----|----|----------------------------------------------|
| ReaderID | int | IN | 所要获取的设备标示ID，范围0x0000-0xFFFF，如果未知，则ReaderID=0 |

返回值：>=0成功，并且为所获取的设备标示，<0失败

### 3.1.11 读取读卡器内部版本号

函数原形：int YW\_GetReaderVersion(int ReaderID);

参数列表：

| 参数       | 类型  | 方向 | 含义                                           |
|----------|-----|----|----------------------------------------------|
| ReaderID | int | IN | 所要获取的设备标示ID，范围0x0000-0xFFFF，如果未知，则ReaderID=0 |

返回值：大于0为版本号，<=0为错误

### 3.1.12 查询读写器产品序列号

函数原形：int YW\_GetReaderSerial(int ReaderID, char \*ReaderSerial);

参数列表：

| 参数           | 类型     | 方向  | 含义                                           |
|--------------|--------|-----|----------------------------------------------|
| ReaderID     | int    | IN  | 所要获取的设备标示ID，范围0x0000-0xFFFF，如果未知，则ReaderID=0 |
| ReaderSerial | Char * | OUT | 读取的产品序列号，长度为8个字节                             |

**返回值：**大于0为成功，≤0为失败

### 3.1.13 蜂鸣器控制函数

**函数原形：**`int YW_Buzzer(int ReaderID, int Time_ON, int Time_OFF, int Cycle);`

**参数列表：**

| 参数       | 类型  | 方向 | 含义                                           |
|----------|-----|----|----------------------------------------------|
| ReaderID | int | IN | 所要获取的设备标示ID，范围0x0000-0xFFFF，如果未知，则ReaderID=0 |
| Time_ON  | int | IN | 蜂鸣器鸣叫时间，单位：100毫秒                             |
| Time_OFF | int | IN | 蜂鸣器静音时间，单位：100毫秒                             |
| Cycle    | int | IN | 把Time_ON和Time_OFF作为一个周期，则此参数为执行此周期的次数。       |

**返回值：**大于0为命令发送成功，≤0为命令发送失败

### 3.1.14 LED 指示灯控制

**函数原形：**`int YW_Led(int ReaderID, int LEDIndex, int Time_ON, int Time_OFF, int Cycle, int LedIndexOn);`

**参数列表：**

| 参数       | 类型  | 方向 | 含义                                           |
|----------|-----|----|----------------------------------------------|
| ReaderID | int | IN | 所要获取的设备标示ID，范围0x0000-0xFFFF，如果未知，则ReaderID=0 |
| LEDIndex | int | IN | LED灯序号<br>0x01：红灯<br>0x02：绿灯                 |
| Time_ON  | int | IN | LED灯亮时间，单位：100毫秒                             |
| Time_OFF | int | IN | LED灯灭时间，单位：100毫秒                             |
| Cycle    | int | IN | 把Time_ON和Time_OFF作为一个周期，则此参数                 |

|            |     |    |                                          |
|------------|-----|----|------------------------------------------|
|            |     |    | 为执行此周期的次数。                               |
| LedIndexOn | int | IN | 最后要亮的灯：<br>0x00：全灭<br>0x01：红灯<br>0x02：绿灯 |

**返回值：**大于0为命令发送成功，≤0为命令发送失败

### 3.1.15 设置天线的状态

**函数原形：** `int YW_AntennaStatus(int ReaderID, bool Status);`

**参数列表：**

| 参数       | 类型   | 方向 | 含义                                           |
|----------|------|----|----------------------------------------------|
| ReaderID | int  | IN | 所要获取的设备标示ID，范围0x0000-0xFFFF，如果未知，则ReaderID=0 |
| Status   | bool | IN | 天线状态：<br>0x01→开天线<br>0x00→关天线                |

**返回值：**大于0为命令发送成功，≤0为命令发送失败

### 3.1.16 设置寻卡模式

**函数原形：** `int YW_SearchCardMode(int ReaderID, int SearchMode);`

**参数列表：**

| 参数         | 类型   | 方向 | 含义                                                                                                                                         |
|------------|------|----|--------------------------------------------------------------------------------------------------------------------------------------------|
| ReaderID   | int  | IN | 所要获取的设备标示ID，范围0x0000-0xFFFF，如果未知，则ReaderID=0                                                                                               |
| SearchMode | char | IN | 卡类型：<br><b>0x41 ( 'A' )→ISO14443 TypeA</b><br>包括如Mifare Classic卡，如S50、S70、Mifare mini、Mifare ultralight、MifarePlus、Mifare Desfire、Jcop 41、 |

|  |  |  |                                                                                                                                                                                                                                                                                                                                     |
|--|--|--|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  |  |  | <p>NTAG203/210/212/213/216、Jewel及其ISO14443-4 TYPEA CPU卡等。</p> <p><b>0x42( 'B' )-&gt;ISO14443 TypeB</b></p> <p>包括如AT88RF020、AT88RF080 、SR176、SRI512/1k/2k/4k、SLIX4K 、ISO14443-4 (T=CL) TYPE B CPU卡</p> <p><b>0x31( '1' )-&gt;ISO15693</b></p> <p>包括如 I.CODE SLI 、Tag-it HF-I 、EM4135、SRF55VxxP、SRF55VxxS、LRI12/64/128/2k、等其它兼容卡片</p> |
|--|--|--|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**返回值：**大于0为命令发送成功，<=0为命令发送失败

### 3.1.17 设置寻卡模式（仅 YW-607，YW-627 系列）

**函数原形：** `int YW_SearchCardModeEx(int ReaderID, int SearchMode, int Param);`

**参数列表：**

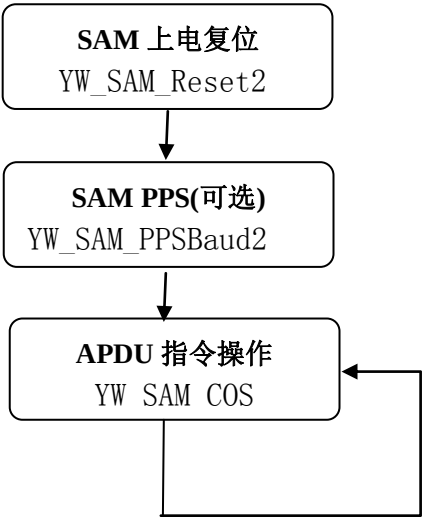
| 参数         | 类型   | 方向 | 含义                                                                                                                                                                                                                                                                                                                                                                                                 |
|------------|------|----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ReaderID   | int  | IN | 所要获取的设备标示ID，范围0x0000-0xFFFF，如果未知，则ReaderID=0                                                                                                                                                                                                                                                                                                                                                       |
| SearchMode | char | IN | <p>卡类型：</p> <p><b>0x41( 'A' )-&gt;ISO14443 TypeA</b></p> <p>包括如Mifare Classic卡，如S50、S70、Mifare mini、 Mifare ultralight、MifarePlus、Mifare Desfire、Jcop 41、NTAG203/210/212/213/216、Jewel及其ISO14443-4 TYPEA CPU卡等。</p> <p><b>0x42( 'B' )-&gt;ISO14443 TypeB</b></p> <p>包括如AT88RF020、AT88RF080 、SR176、SRI512/1k/2k/4k、SLIX4K 、ISO14443-4 (T=CL) TYPE B CPU卡</p> <p><b>0x31( '1' )-&gt;ISO15693</b></p> |

|       |     |    |                                                                                                                                                                                 |
|-------|-----|----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|       |     |    | 包括如 I.CODE SLI 、 Tag-it HF-I 、 EM4135、 SRF55VxxP、<br>SRF55VxxS、 LRI12/64/128/2k、 等其它兼容卡片<br><br><b><i>0X54( 'T' )-&gt;Topaz</i></b><br><br><b><i>0X46( 'F' )-&gt;Felica</i></b> |
| Param | int | IN | 此参数与SearchMode参数相关，具体设置如下：<br>当SearchMode = 0x46 Felica<br>0x01 ->424k bps Felica<br>0x00 ->212k bps Felica<br>其他SearchMode该参数暂未定义                                              |

**返回值：**大于0为命令发送成功，<=0为命令发送失败

### 3.2 ISO7816 标准接触卡

YW-607HC读卡器包含2个符合ISO7816(T=0/T=1)标准的SAM卡槽和1个符合ISO7816(T=0/T=1)标准的大卡卡槽。



(图-0 ISO7816 卡片操作流程图)

#### 3.2.1 SAM 卡复位

**函数原形:** `int YW_SAM_ResetEx(int ReaderID, int SAMIndex, unsigned char SAMVoltage, int *rtLen, unsigned char *pData);`

**参数列表:**

| 参数         | 类型            | 方向  | 含义                                                       |
|------------|---------------|-----|----------------------------------------------------------|
| ReaderID   | int           | IN  | 所要获取的设备标示ID，范围0x0000-0xFFFF，如果未知，则ReaderID=0             |
| SAMIndex   | int           | IN  | ISO7816 卡序号<br>0x01-> SAM1<br>0x02-> SAM2<br>0x03> IC卡插槽 |
| SAMVoltage | unsigned char | IN  | SAM卡供电电压, 0为5V，1为3.3V                                    |
| rtLen      | int *         | OUT | SAM卡复位返回的数据pData的长度                                      |

|       |                 |     |             |
|-------|-----------------|-----|-------------|
| pData | unsigned char * | OUT | SAM卡复位返回的数据 |
|-------|-----------------|-----|-------------|

**返回值：**大于0为成功，≤0为失败

### 3.2.2 SAM 卡执行 COS 命令

**函数原形：**`int YW_SAM_COS(int ReaderID, int SAMIndex, int LenCOS, unsigned char *Com_COS, int *rtLen, unsigned char *pData);`

**参数列表：**

| 参数       | 类型              | 方向  | 含义                                                     |
|----------|-----------------|-----|--------------------------------------------------------|
| ReaderID | int             | IN  | 所要获取的设备标示ID，范围0x0000-0xFFFF，如果未知，则ReaderID=0           |
| SAMIndex | int             | IN  | IS07816 卡序号<br>0x01→ SAM1<br>0x02→ SAM2<br>0x03→ IC卡插槽 |
| LenCOS   | int             | IN  | 向SAM卡要发送的COS命令的长度                                      |
| Com_COS  | unsigned char * | IN  | 向SAM卡要发送的COS命令                                         |
| rtLen    | unsigned char * | OUT | SAM执行COS命令后返回的数据的长度                                    |
| pData    | unsigned char * | OUT | SAM执行COS命令后返回的数据                                       |

**返回值：**大于0为成功，≤0为失败

### 3.2.3 SAM 卡 PPS 波特率设置

**函数原形：**`int YW_SAM_PPSBaudEx(int ReaderID, int SAMIndex, int BaudIndex, int Protocol);`

**参数列表：**

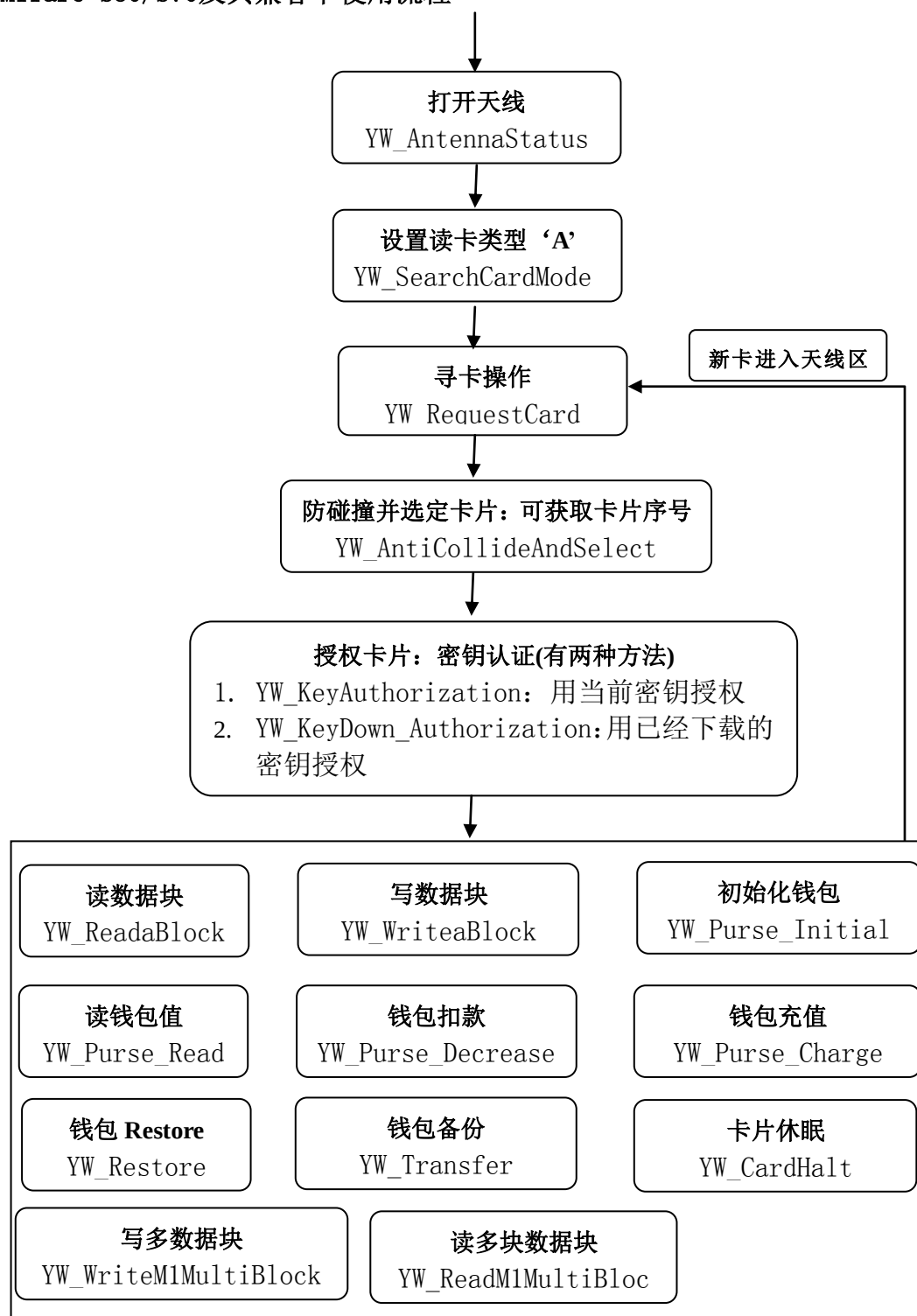
| 参数 | 类型 | 方向 | 含义 |
|----|----|----|----|
|----|----|----|----|

|           |     |    |                                                                                                                                                             |
|-----------|-----|----|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ReaderID  | int | IN | 所要获取的设备标示ID，范围0x0000-0xFFFF，<br>如果未知，则ReaderID=0                                                                                                            |
| SAMIndex  | int | IN | ISO7816 卡序号<br><br>0x01-> SAM1<br><br>0x02-> SAM2<br><br>0x03-> IC卡插槽                                                                                       |
| BaudIndex | int | IN | ISO7816 卡速率：<br><br>0x00->9600、0x01->19200、0x02->38400、<br>0x03->55800、0x04->57600、0x05->115200、<br>0x06->223200、0x07->230400、0x08->446400、<br>0x09->460800 |
| Protocol  | int | IN | 重新定义协议类型：(必须卡片支持此协议)<br><br>0x00->T0<br><br>0x01->T1<br><br>其它->保持原有协议不变                                                                                    |

**返回值：**大于0为成功，<=0为失败

### 3.3 ISO14443 TypeA 相关函数

Mifare S50/S70及其兼容卡使用流程



(图-1 ISO14443 TYPEA Mifare S50/S70操作流程图)

注: 1. YW\_RequestCard 和 YW\_AntiCollideAndSelect 两步骤可以用 YW\_RequestAntiandSelect 代替。

### 3.3.1 寻卡

**函数原形：** `int YW_RequestCard(int ReaderID, char RequestMode, unsigned short *CardType);`

**参数列表：**

| 参数          | 类型               | 方向  | 含义                                                                                                                                                                                                                                                                                                      |
|-------------|------------------|-----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ReaderID    | int              | IN  | 所要获取的设备标示ID，范围<br>0x0000-0xFFFF，如果未知，则ReaderID=0                                                                                                                                                                                                                                                        |
| RequestMode | char             | IN  | 寻卡的模式：<br><br>0x52->所有卡<br><br>0x26->未休眠卡                                                                                                                                                                                                                                                               |
| CardType    | unsigned short * | OUT | 返回卡的类型:ATQA(2字节)<br>0x4400-> Ultralight/UltraLight C /MifarePlus(7Byte UID)<br>0x0400-> Mifare Mini/Mifare 1K (S50) /MifarePlus(4Byte UID)<br>0x0200-> Mifare_4K(S70)/ MifarePlus(4Byte UID)<br>0x0800-> Mifare_Pro<br>0x0403-> Mifare_ProX<br>0x4403->Mifare_DESFire<br>0x4200-> MifarePlus(7Byte UID) |

**返回值：** 大于0为命令发送成功，<=0为命令发送失败

### 3.3.2 防碰撞并且选定一张卡

**函数原形：** `int YW_AntiCollideAndSelect(int ReaderID, unsigned char MultiCardMode, unsigned char *CardMem, unsigned char *SNLen, unsigned char *SN);`

**参数列表：**

| 参数       | 类型  | 方向 | 含义             |
|----------|-----|----|----------------|
| ReaderID | int | IN | 所要获取的设备标示ID，范围 |

|               |                 |     |                                             |
|---------------|-----------------|-----|---------------------------------------------|
|               |                 |     | 0x0000-0xFFFF, 如果未知, 则ReaderID=0            |
| MultiCardMode | unsigned char   | IN  | 对多张卡的处理方式:<br>0x00->多张卡返回错误<br>0x01->返回一张卡号 |
| CardMem       | unsigned char * | OUT | 卡片容量代码:SAK(1字节)                             |
| SNLen         | unsigned char*  | OUT | 输出卡号的长度(4/7/10)                             |
| SN            | unsigned char * | OUT | 输出卡的序列号(4/7/10字节)                           |

**返回值:** 大于0为命令发送成功, <=0为命令发送失败

### 3.3.3 卡片休眠

**函数原形:** `int YW_CardHalt(int ReaderID);`

**参数列表:**

| 参数       | 类型  | 方向 | 含义                                              |
|----------|-----|----|-------------------------------------------------|
| ReaderID | int | IN | 所要获取的设备标示ID, 范围0x0000-0xFFFF, 如果未知, 则ReaderID=0 |

**返回值:** 大于0为命令发送成功, <=0为命令发送失败

### 3.3.4 下载密钥到读卡器

**函数原形:** `int YW_DownloadKey(int ReaderID, int KeyIndex, unsigned char * Key);`

**参数列表:**

| 参数       | 类型  | 方向 | 含义                                              |
|----------|-----|----|-------------------------------------------------|
| ReaderID | int | IN | 所要获取的设备标示ID, 范围0x0000-0xFFFF, 如果未知, 则ReaderID=0 |

|          |                 |    |                                                                 |
|----------|-----------------|----|-----------------------------------------------------------------|
| KeyIndex | int             | IN | 待写入密钥存储的序号(0~31), 共可存储32个密钥, 该密钥是先写入读卡器保存, 到卡片验证密钥的时候可以指定序号去验证。 |
| Key      | unsigned char * | IN | 密钥, 每个密钥6个字节                                                    |

**返回值:** 大于0为命令发送成功, 小于0为命令发送失败

### 3.3.5 用下载的密钥授权卡片

**函数原形:** `int YW_KeyDown_Authorization (int ReaderID, char KeyMode , int BlockAddr, int KeyIndex);`

**参数列表:**

| 参数        | 类型   | 方向 | 含义                                              |
|-----------|------|----|-------------------------------------------------|
| ReaderID  | Int  | IN | 所要获取的设备标示ID, 范围0x0000-0xFFFF, 如果未知, 则ReaderID=0 |
| KeyMode   | char | IN | 验证密钥类别:<br>0x60->A密钥<br>0x61->B密钥               |
| BlockAddr | int  | IN | 要授权卡片的绝对块号地址                                    |
| KeyIndex  | int  | IN | 授权密钥的序号(0~31)<br>通过YW_DownloadKey下载保存的密钥序号。     |

**返回值:** 大于0为命令发送成功, 小于0为命令发送失败

### 3.3.6 用当前密钥授权卡片

**函数原形:** `int YW_KeyAuthorization (int ReaderID, char KeyMode, int BlockAddr, unsigned char *Key);`

**参数列表:**

| 参数       | 类型  | 方向 | 含义              |
|----------|-----|----|-----------------|
| ReaderID | int | IN | 所要获取的设备标示ID, 范围 |

|           |                 |    |                                   |
|-----------|-----------------|----|-----------------------------------|
|           |                 |    | 0x0000-0xFFFF, 如果未知, 则 ReaderID=0 |
| KeyMode   | char            | IN | 验证密钥类别:<br>0x60→A密钥<br>0x61→B密钥   |
| BlockAddr | int             | IN | 要授权卡片的绝对块号地址                      |
| Key       | unsigned char * | IN | 当前密钥字节 (共6个字节)                    |

**返回值:** 大于0为命令发送成功, 小于0为命令发送失败

### 3.3.7 读取一块数据

**函数原形:** `int YW_ReadaBlock (int ReaderID, int BlockAddr, int LenData, unsigned char *Data);`

**参数列表:**

| 参数        | 类型              | 方向  | 含义                                                   |
|-----------|-----------------|-----|------------------------------------------------------|
| ReaderID  | int             | IN  | 所要获取的设备标示ID, 范围 0x0000-0xFFFF, 如果未知, 则 ReaderID=0    |
| BlockAddr | int             | IN  | 绝对地址块号                                               |
| LenData   | int             | IN  | 要读出的数据的字节数, Mifare S50/S70为16字节<br>一般固定为: 0x10(16字节) |
| Data      | unsigned char * | OUT | 输出读到的块的数据                                            |

**返回值:** 大于0为命令发送成功, 小于0为命令发送失败

### 3.3.8 写入一块数据

**函数原形:** `int YW_WriteaBlock (int ReaderID, int BlockAddr, int LenData, unsigned char *Data);`

**参数列表:**

| 参数        | 类型              | 方向 | 含义                                                   |
|-----------|-----------------|----|------------------------------------------------------|
| ReaderID  | int             | IN | 所要获取的设备标示ID，范围 0x0000-0xFFFF，如果未知，则 ReaderID=0       |
| BlockAddr | int             | IN | 绝对块号地址                                               |
| LenData   | int             | IN | 要写入的数据的字节数，Mifare S50/S70固定为16字节<br>一般固定为：0x10(16字节) |
| Data      | unsigned char * | IN | 要写入的块的数据                                             |

**返回值：**大于0为命令发送成功，小于0为命令发送失败

### 3.3.9 将某一块初始化为钱包

**函数原形：** `int YW_Purse_Initial (int ReaderID, int BlockAddr, int IniMoney);`

**参数列表：**

| 参数        | 类型  | 方向 | 含义                                            |
|-----------|-----|----|-----------------------------------------------|
| ReaderID  | int | IN | 所要获取的设备标示ID，范围 0x0000-0xFFFF，如果未知，则ReaderID=0 |
| BlockAddr | int | IN | 绝对块号地址                                        |
| IniMoney  | int | IN | 初始化钱包时的初始值                                    |

**返回值：**大于0为命令发送成功，小于0为命令发送失败

### 3.3.10 读取钱包值

**函数原形：** `int YW_Purse_Read (int ReaderID, int BlockAddr, int *Money);`

**参数列表：**

| 参数       | 类型  | 方向 | 含义             |
|----------|-----|----|----------------|
| ReaderID | int | IN | 所要获取的设备标示ID，范围 |

|           |       |     |                                  |
|-----------|-------|-----|----------------------------------|
|           |       |     | 0x0000-0xFFFF, 如果未知, 则ReaderID=0 |
| BlockAddr | int   | IN  | 绝对块号地址                           |
| Money     | int * | OUT | 钱包的当前值                           |

**返回值:** 大于0为命令发送成功, 小于0为命令发送失败

### 3.3.11 钱包扣款

**函数原形:** `int YW_Purse_Decrease (int ReaderID, int BlockAddr, int Decrement);`

**参数列表:**

| 参数        | 类型  | 方向 | 含义                                                  |
|-----------|-----|----|-----------------------------------------------------|
| ReaderID  | int | IN | 所要获取的设备标示ID, 范围<br>0x0000-0xFFFF, 如果未知, 则ReaderID=0 |
| BlockAddr | int | IN | 绝对块号地址                                              |
| Decrement | int | IN | 扣款值                                                 |

**返回值:** 大于0为命令发送成功, 小于0为命令发送失败

### 3.3.12 钱包充值

**函数原形:** `int YW_Purse_Charge (int ReaderID, int BlockAddr, int Charge);`

**参数列表:**

| 参数        | 类型  | 方向 | 含义                                                 |
|-----------|-----|----|----------------------------------------------------|
| ReaderID  | int | IN | 所要获取的设备标示ID, 范围0x0000-0xFFFF,<br>如果未知, 则ReaderID=0 |
| BlockAddr | int | IN | 绝对块号地址                                             |
| Charge    | int | IN | 钱包中要充值的值                                           |

**返回值:** 大于0为命令发送成功, 小于0为命令发送失败

### 3.3.13 钱包 Restore 命令

函数原形: `int YW_Restore (int ReaderID, int BlockAddr);`

参数列表:

| 参数        | 类型  | 方向 | 含义                                              |
|-----------|-----|----|-------------------------------------------------|
| ReaderID  | int | IN | 所要获取的设备标示ID, 范围0x0000-0xFFFF, 如果未知, 则ReaderID=0 |
| BlockAddr | int | IN | 待备份钱包的绝对块号地址                                    |

返回值: 大于0为命令发送成功, 小于0为命令发送失败

**注意:** Restore指令是将钱包值暂存于卡片Data缓冲区, 已备Transfer指令备份到不同的钱包块中, 这两个指令是结合起来使用, 使用方法: 1. 先使用Restore将待备份的钱包值写入卡片Data缓冲区; 2. 使用Transfer将Data缓冲区中的钱包值备份到指定钱包的块中。

### 3.3.14 钱包 Transfer 命令

函数原形: `int YW_Transfer (int ReaderID, int BlockAddr);`

参数列表:

| 参数        | 类型  | 方向 | 含义                                              |
|-----------|-----|----|-------------------------------------------------|
| ReaderID  | int | IN | 所要获取的设备标示ID, 范围0x0000-0xFFFF, 如果未知, 则ReaderID=0 |
| BlockAddr | int | IN | 备份到钱包的绝对块号地址                                    |

返回值: 大于0为命令发送成功, 小于0为命令发送失败

### 3.3.15 读取多块数据

函数原形: `int YW_ReadM1MultiBlock(int ReaderID, int StartBlock, int BlockNums, int *LenData, char *pData);`

参数列表:

| 参数 | 类型 | 方向 | 含义 |
|----|----|----|----|
|----|----|----|----|

|            |                 |     |                                              |
|------------|-----------------|-----|----------------------------------------------|
| ReaderID   | int             | IN  | 所要获取的设备标示ID，范围0x0000-0xFFFF，如果未知，则ReaderID=0 |
| StartBlock | int             | IN  | 绝对地址开始块号                                     |
| BlockNums  | int             | IN  | 要读块的块数                                       |
| LenData    | int*            | OUT | 实际读出的数据的字节数                                  |
| Data       | unsigned char * | OUT | 实际读到的块的数据                                    |

**返回值：**大于0为命令发送成功，小于0为命令发送失败

### 3.3.16 写多块数据

**函数原形：**`int YW_WriteM1MultiBlock(int ReaderID, int StartBlock, int BlockNums, int LenData, char *pData);`

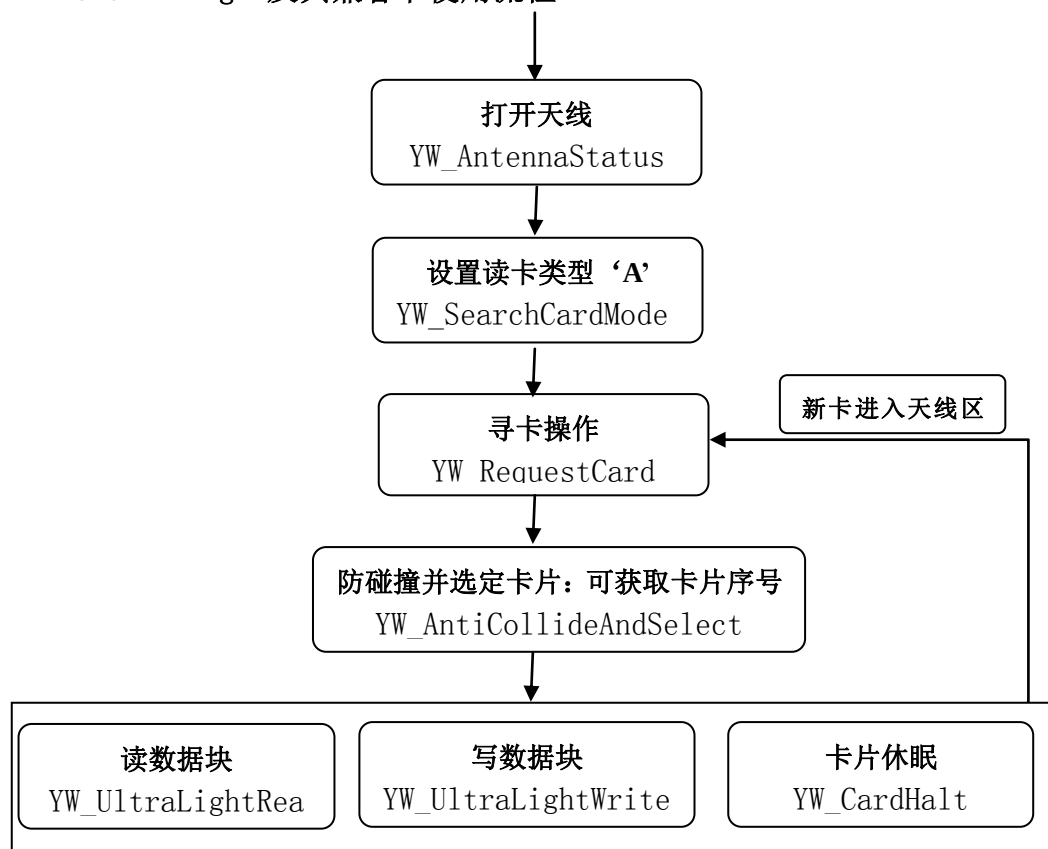
**参数列表：**

| 参数         | 类型              | 方向 | 含义                                           |
|------------|-----------------|----|----------------------------------------------|
| ReaderID   | int             | IN | 所要获取的设备标示ID，范围0x0000-0xFFFF，如果未知，则ReaderID=0 |
| StartBlock | int             | IN | 写入块数据的绝对地址开始块号                               |
| BlockNums  | int             | IN | 写入块的数量                                       |
| LenData    | int             | IN | 要写入的数据的字节数，Mifare One为16* BlockNums个字节       |
| Data       | unsigned char * | IN | 写入的块的数据                                      |

**返回值：**大于0为命令发送成功，小于0为命令发送失败

## 3.4 ISO14443 TYPEA UltraLight 卡操作函数

Mifare Ultralight及其兼容卡使用流程



(图-2 ISO14443 TYPEA Ultralight卡操作流程)

注：1. YW\_RequestCard和YW\_AntiCollideAndSelect 两步骤可以用

YW\_RequestAntiandSelect 代替。

2. YW\_UltraLightRead 和 YW\_ReadaBlock功能一致。

### 3.4.1 UltraLight 卡读块数据

函数原形: `int YW_UltraLightRead(int ReaderID, int BlockID, unsigned char *pData);`

参数列表:

| 参数       | 类型  | 方向 | 含义                                               |
|----------|-----|----|--------------------------------------------------|
| ReaderID | int | IN | 所要获取的设备标示ID, 范围 0x0000-0xFFFF, 如果未知, 则ReaderID=0 |

|         |                    |     |                                                        |
|---------|--------------------|-----|--------------------------------------------------------|
| BlockID | int                | IN  | 起始绝对地址块号                                               |
| pData   | unsigned<br>char * | OUT | 输出读到的块的数据, 16字节, Ultralight<br>一次读出从起始块的4块数据, 所以4*4=16 |

**返回值:** 大于0为命令发送成功, 小于0为命令发送失败

### 3.4.2 UltraLight 卡写块数据

**函数原形:** `int YW_UltraLightWrite(int ReaderID, int BlockID, unsigned char *pData);`

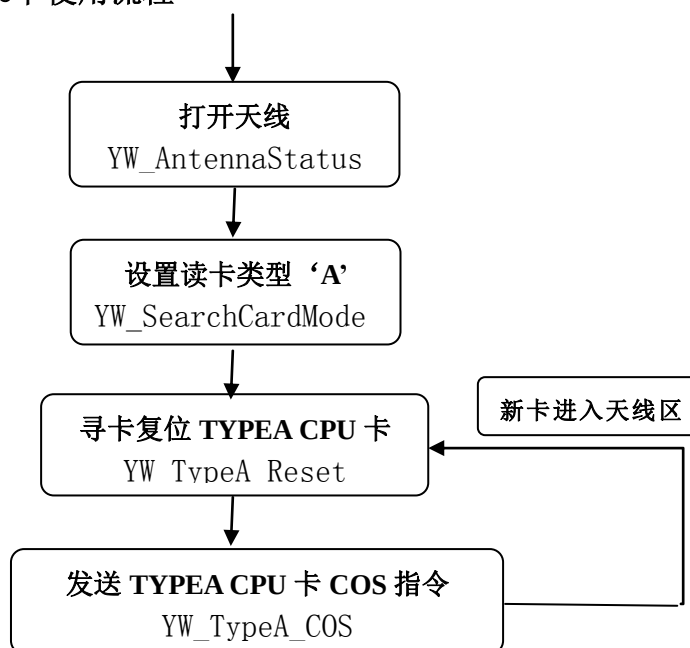
**参数列表:**

| 参数       | 类型              | 方向 | 含义                                                      |
|----------|-----------------|----|---------------------------------------------------------|
| ReaderID | int             | IN | 所要获取的设备标示ID, 范围<br>0x0000-0xFFFF, 如果未知, 则<br>ReaderID=0 |
| BlockID  | int             | IN | 绝对地址块号                                                  |
| pData    | unsigned char * | IN | 要写入的块的数据, 4字节                                           |

**返回值:** 大于0为命令发送成功, 小于0为命令发送失败

## 3.5 ISO14443-4 Type A CPU 卡操作函数

ISO14443-4 TypeA CPU卡使用流程



(图-3 ISO14443-4 TYPEA CPU 操作流程图)

注：1. YW\_TypeA\_COS 其中包括 A 卡的寻卡，选卡操作，同时多出复位动作。

### 3.5.1 ISO14443-4 TypeA CPU 卡复位

**函数原形：** `int YW_TypeA_Reset(int ReaderID, unsigned char Mode, unsigned char MultiMode, int *rtLen, unsigned char *pData);`

**参数列表：**

| 参数        | 类型            | 方向 | 含义                                             |
|-----------|---------------|----|------------------------------------------------|
| ReaderID  | int           | IN | 所要获取的设备标示ID，范围 0x0000-0xFFFF，如果未知，则 ReaderID=0 |
| Mode      | unsigned char | IN | 寻卡的模式：<br>0x52->所有的卡<br>0x26->未休眠的卡            |
| MultiMode | unsigned char | IN | 对多张卡的处理方式：                                     |

|       |                 |     |                                  |
|-------|-----------------|-----|----------------------------------|
|       |                 |     | 0x00->多张卡返回错误<br>0x01->返回一张卡复位信息 |
| rtLen | int *           | OUT | 返回复位信息的长度                        |
| pData | unsigned char * | OUT | 返回复位信息内容                         |

**返回值：**大于0为命令发送成功，小于0为命令发送失败

### 3.5.2 ISO14443-4 TypeA CPU 卡执行 COS 命令

**函数原形：**`int YW_TypeA_COS(int ReaderID, int LenCOS, unsigned char *Com_COS, int *rtLen, unsigned char *pData);`

**参数列表：**

| 参数       | 类型              | 方向  | 含义                                             |
|----------|-----------------|-----|------------------------------------------------|
| ReaderID | int             | IN  | 所要获取的设备标示ID，范围 0x0000-0xFFFF，如果未知，则 ReaderID=0 |
| LenCOS   | unsigned char*  | IN  | 输入的 C O S 命令的长度                                |
| Com_COS  | unsigned char*  | IN  | C O S 命令                                       |
| rtLen    | int *           | OUT | 返回执行命令结果的长度                                    |
| pData    | unsigned char * | OUT | 返回执行命令结果                                       |

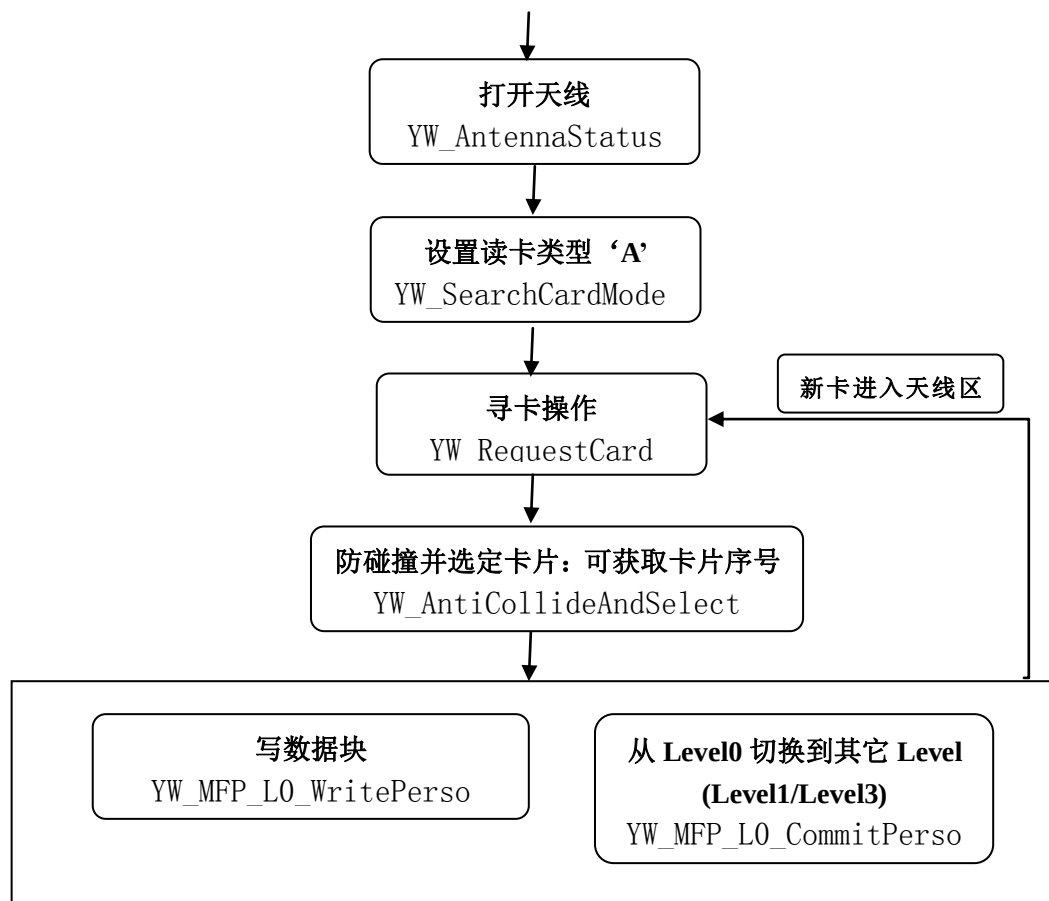
**返回值：**大于0为命令发送成功，小于0为命令发送失败

## 3.6 Mifare Plus 卡操作函数

MifarePlus 卡片结构见附录，MifarePlus 分为 4 个安全级别(Level0~Level3)，不同的安全级别对寻卡操作不同，有的只需要寻卡片序号，有的需要寻卡后对卡片进行复位操作。其中 MifarePlus Level1 兼容原来的 mifareone，所有操作同 Mifareone。

### Level0 操作如下：

Mifare Plus Level0 是出厂层级，在该层级主要完成一些密钥和数据初始化，然后切换到其它层级，切换到的 level 依据卡不同而不同，常见是切换到 level1。在该层级必须初始化的参数是：0x9000~0x9003。也可以写入 AES 的初始密钥，具体见 MifarePlus 数据手册。Level0 操作流程如下：



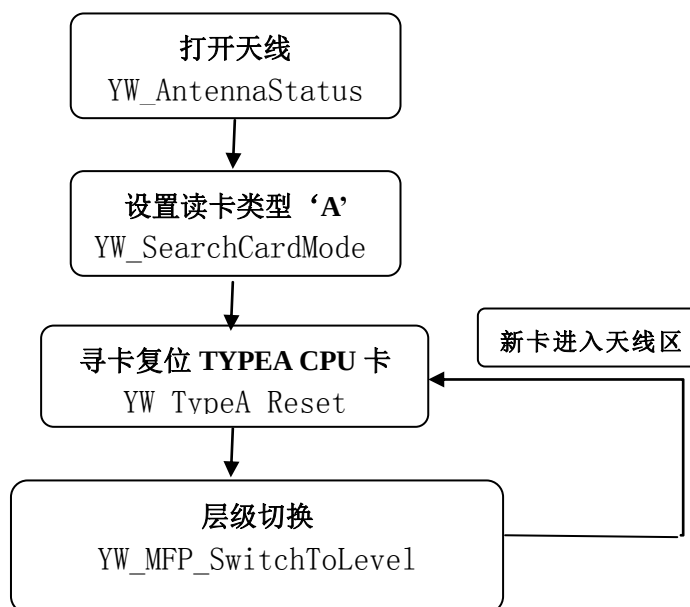
(图-4 MifarePlus Level0 卡操作流程图)

### Level1 块操作如下：

Mifare Plus level1所有的块操作兼容mifare S50/S70, 操作流程见 图-1

### Level1 Switch 操作:

Level1 层级切换，可以切换到 Level2 或 Level3，只能切换到高层级，不能向低的层级切换。



(图-5 MifarePlus Level1 层级切换操作流程)

注意：从 Level1 切换到其它 Level，假如想从 Level1 切换到 Level2，那么 Switch Key 就用 Switch Key2。切换到 Level3，那么 Switch Key 就用 Switch Key2

### Level2 操作:

Level2 是该读卡器仅提供过度层级，只支持切换到 Level3 操作，操作流程见图-5，切换 key 用 Switch Key3。

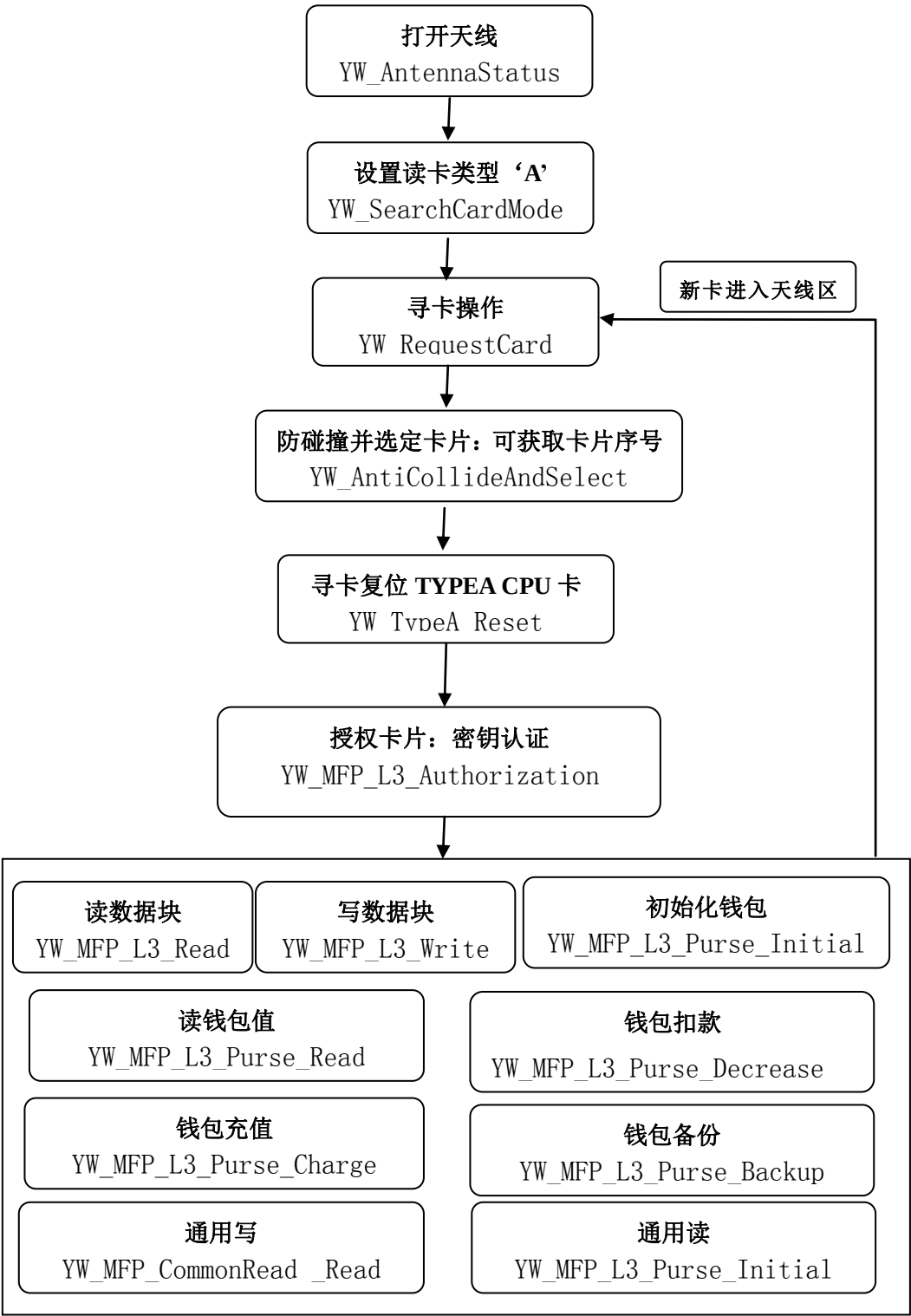
假如想从 Level2 切换到 Level3，那么 Switch Key 就用 Switch Key3。

### Level3 操作:

Level3 操作分两种模式：类似 Mifare S50/S70 模式，块地址用 1 个字节（因为高字节是 00）；通用模式，该模式所有地址都是 2 个字节，完全按 MifarePlus 数据手册定义操作。对于高位地址不是 0x00 的块，如 0x9000 块，则可以通过通用模式指令读写，但是相应的授权需根据 MifarePlus 手册定义的选择用

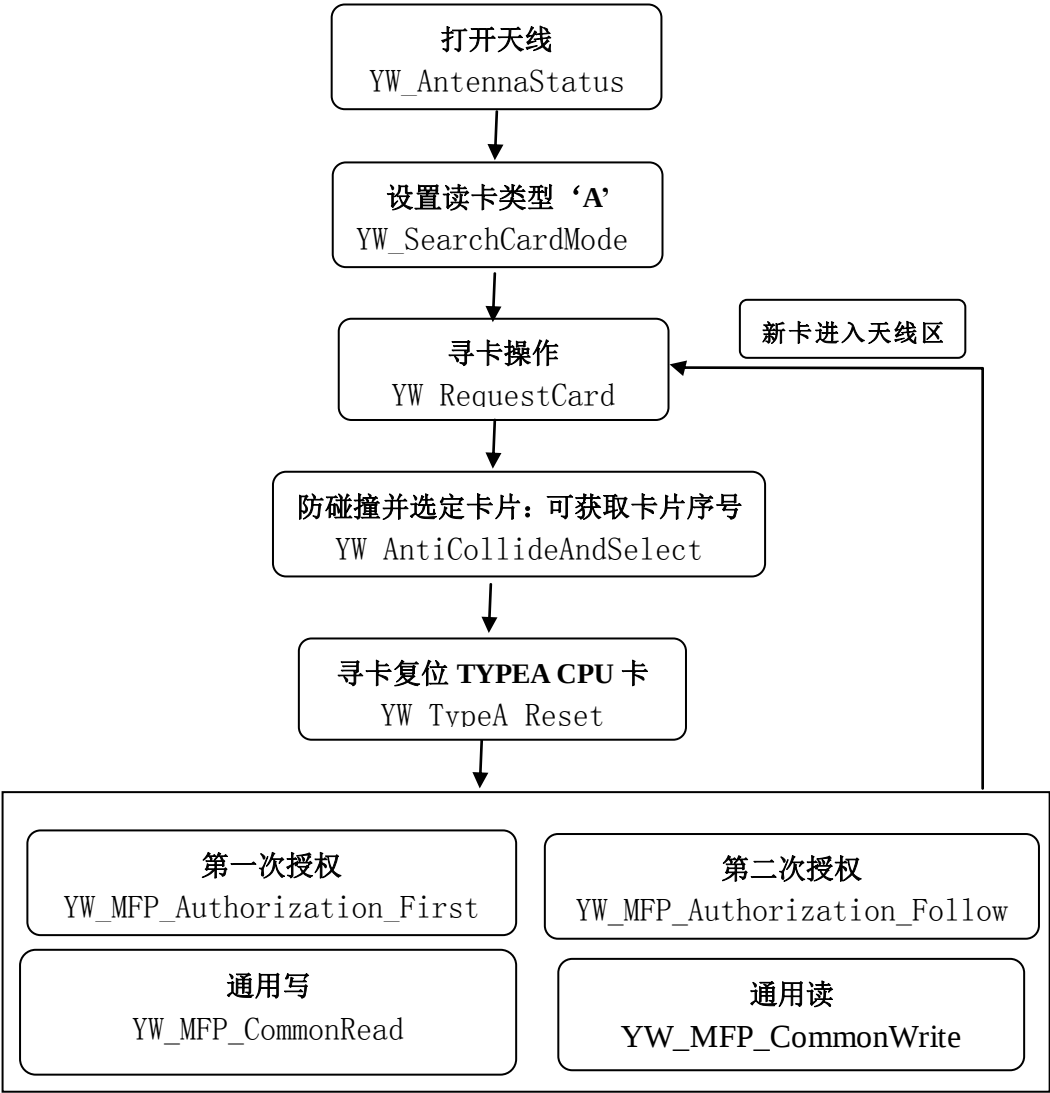
YW\_MFP\_Authorization\_First 或者 YW\_MFP\_Authorization\_Follow 进行授权。

类似 *Mifare S50/S70* 模式:



(图-6 Mifare Plus Level3 模式1操作流程图)

Level3 通用模式:



(图-7 Mifare Plus Level3 模式2操作流程图)

3.6.1 Mifare Plus 卡 Level 0 级写数据

函数原形: `int YW_MFP_L0_WritePerso(int ReaderID, int Address, unsigned char *pData);`

参数列表:

| 参数       | 类型  | 方向 | 含义                                                |
|----------|-----|----|---------------------------------------------------|
| ReaderID | int | IN | 所要获取的设备标示ID, 范围 0x0000-0xFFFF, 如果未知, 则 ReaderID=0 |

|         |                |    |               |
|---------|----------------|----|---------------|
| Address | int            | IN | 要写入数据的地址      |
| pData   | unsigned char* | IN | 要写入的数据， 16 字节 |

**返回值：**大于0为命令发送成功，小于0为命令发送失败

### 3.6.2 Mifare Plus 卡从 Level 0 级向 Level 1 或 3 级切换

**函数原形：**int YW\_MFP\_L0\_CommitPerso(int ReaderID);

**参数列表：**

| 参数       | 类型  | 方向 | 含义                                           |
|----------|-----|----|----------------------------------------------|
| ReaderID | int | IN | 所要获取的设备标示ID，范围0x0000-0xFFFF，如果未知，则ReaderID=0 |

**返回值：**大于0为命令发送成功，小于0为命令发送失败

注意：切换到的层级依据不同的卡而不同

### 3.6.3 Mifare Plus 卡从 Level1 或 Level2 向高级切换

**函数原形：**int YW\_MFP\_SwitchToLevel(int ReaderID, int DesLevel, unsigned char \*SwitchKey);

**参数列表：**

| 参数        | 类型             | 方向 | 含义                                                                                     |
|-----------|----------------|----|----------------------------------------------------------------------------------------|
| ReaderID  | int            | IN | 所要获取的设备标示ID，范围0x0000-0xFFFF，如果未知，则ReaderID=0                                           |
| DesLevel  | unsigned char  | IN | 要切换到的层级，最高3级<br>0x02-> 从当前层级切换到Level2<br>0x03-> 从当前层级切换到Level3<br>注意：只能从低层级切换到高层级，不可逆。 |
| SwitchKey | unsigned char* | IN | 切换AES密钥，16字节<br>若切换到Level2，用Switch Key2                                                |

|  |  |  |                         |
|--|--|--|-------------------------|
|  |  |  | 若切换到Level3，用Switch Key3 |
|--|--|--|-------------------------|

**返回值：**大于0为命令发送成功，小于0为命令发送失败

### 3.6.4 Mifare Plus 卡 Level 3 级授权

**函数原形：**`int YW_MFP_L3_Authorization(int ReaderID, int KeyMode, int BlockID, unsigned char *Key);`

**参数列表：**

| 参数       | 类型             | 方向 | 含义                                                    |
|----------|----------------|----|-------------------------------------------------------|
| ReaderID | int            | IN | 所要获取的设备标示ID，范围0x0000~0xFFFF，如果未知，则ReaderID=0          |
| KeyMode  | unsigned char  | IN | 授权密钥类别：<br>0x60→A密钥<br>0x61→B密钥<br>Level3 A/B密钥定义见附录A |
| BlockID  | int            | IN | 块号(范围0x00~0xFF)                                       |
| Key      | unsigned char* | IN | 密钥，16字节                                               |

**返回值：**大于0为命令发送成功，小于0为命令发送失败

### 3.6.5 Mifare Plus 卡 Level 3 级读块数据

**函数原形：**`int YW_MFP_L3_Read(int ReaderID, int StartBlock, int BlockNums, int *DataLen, unsigned char *pData);`

**参数列表：**

| 参数         | 类型  | 方向 | 含义                                           |
|------------|-----|----|----------------------------------------------|
| ReaderID   | int | IN | 所要获取的设备标示ID，范围0x0000~0xFFFF，如果未知，则ReaderID=0 |
| StartBlock | int | IN | 开始块号(范围0x00~0xFF)                            |

|           |                |     |               |
|-----------|----------------|-----|---------------|
| BlockNums | int            | IN  | 块数量(建议在一个扇区内) |
| DataLen   | int*           | OUT | 读到的数据长度       |
| pData     | unsigned char* | OUT | 读到的数据         |

**返回值：**大于0为命令发送成功，小于0为命令发送失败

### 3.6.6 Mifare Plus 卡 Level 3 级写块数据

**函数原形：**int YW\_MFP\_L3\_Write(int ReaderID, int StartBlock, int BlockNums, unsigned char \*pData);

**参数列表：**

| 参数         | 类型             | 方向 | 含义                                           |
|------------|----------------|----|----------------------------------------------|
| ReaderID   | int            | IN | 所要获取的设备标示ID，范围0x0000~0xFFFF，如果未知，则ReaderID=0 |
| StartBlock | int            | IN | 开始块号(范围0x00~0xFF)                            |
| BlockNums  | int            | IN | 块数量(建议在一个扇区内)                                |
| pData      | unsigned char* | IN | 要写入的数据，长度必须是 16 * BlockNums                  |

**返回值：**大于0为命令发送成功，小于0为命令发送失败

### 3.6.7 Mifare Plus 卡 Level 3 级将某一块初始化为钱包

**函数原形：**int YW\_MFP\_L3\_Purse\_Initial(int ReaderID, int BlockID, int InitialValue);

**参数列表：**

| 参数           | 类型  | 方向 | 含义                                           |
|--------------|-----|----|----------------------------------------------|
| ReaderID     | int | IN | 所要获取的设备标示ID，范围0x0000~0xFFFF，如果未知，则ReaderID=0 |
| BlockID      | int | IN | 绝对块号地址(范围0x00~0xFF)                          |
| InitialValue | int | IN | 初始化钱包时的初始值                                   |

**返回值：**大于0为命令发送成功，小于0为命令发送失败

### 3.6.8 Mifare Plus 卡 Level 3 级读取钱包值

**函数原形：**`int YW_MFP_L3_Purse_Read(int ReaderID, int BlockID, int *Value);`

**参数列表：**

| 参数       | 类型    | 方向  | 含义                                           |
|----------|-------|-----|----------------------------------------------|
| ReaderID | int   | IN  | 所要获取的设备标示ID，范围0x0000-0xFFFF，如果未知，则ReaderID=0 |
| BlockID  | int   | IN  | 绝对块号地址(范围0x00~0xFF)                          |
| Value    | int * | OUT | 钱包的当前值                                       |

**返回值：**大于0为命令发送成功，小于0为命令发送失败

### 3.6.9 Mifare Plus 卡 Level 3 级钱包充值

**函数原形：**`int YW_MFP_L3_Purse_Charge(int ReaderID, int BlockID, int Value);`

**参数列表：**

| 参数       | 类型  | 方向 | 含义                                           |
|----------|-----|----|----------------------------------------------|
| ReaderID | int | IN | 所要获取的设备标示ID，范围0x0000-0xFFFF，如果未知，则ReaderID=0 |
| BlockID  | int | IN | 绝对块号地址(范围0x00~0xFF)                          |
| Value    | int | IN | 充值金额(范围 大于0)                                 |

**返回值：**大于0为命令发送成功，小于0为命令发送失败

### 3.6.10 Mifare Plus 卡 Level 3 级钱包扣款

**函数原形：**`int YW_MFP_L3_Purse_Decrease(int ReaderID, int BlockID, int Value);`

**参数列表：**

| 参数       | 类型  | 方向 | 含义                                              |
|----------|-----|----|-------------------------------------------------|
| ReaderID | int | IN | 所要获取的设备标示ID, 范围0x0000-0xFFFF, 如果未知, 则ReaderID=0 |
| BlockID  | int | IN | 绝对块号地址 (范围0x00~0xFF)                            |
| Value    | int | IN | 扣款金额 (范围 大于0)                                   |

**返回值:** 大于0为命令发送成功, 小于0为命令发送失败

### 3.6.11 Mifare Plus 卡 Level 3 级备份钱包

**函数原形:** `int YW_MFP_L3_Purse_Backup(int ReaderID, int BlockID, int DesBlockID);`

**参数列表:**

| 参数         | 类型  | 方向 | 含义                                              |
|------------|-----|----|-------------------------------------------------|
| ReaderID   | int | IN | 所要获取的设备标示ID, 范围0x0000-0xFFFF, 如果未知, 则ReaderID=0 |
| BlockID    | int | IN | 要备份的钱包块号 (范围0x00~0xFF)                          |
| DesBlockID | int | IN | 备份到的目标块号 (范围0x00~0xFF)                          |

**返回值:** 大于0为命令发送成功, 小于0为命令发送失败

### 3.6.12 Mifare Plus 卡 Level 3 级通用读写第一次授权

**函数原形:** `int YW_MFP_Authorization_First(int ReaderID, int AESKeyAddr, unsigned char *AESKey);`

**参数列表:**

| 参数         | 类型       | 方向 | 含义                                                   |
|------------|----------|----|------------------------------------------------------|
| ReaderID   | int      | IN | 所要获取的设备标示ID, 范围0x0000-0xFFFF, 如果未知, 则ReaderID=0      |
| AESKeyAddr | int      | IN | AES 密钥地址 (范围0x0000~0xffff)<br>密钥对应保护的区块见MifarePlus手册 |
| AESKey     | unsigned | IN | AES密钥, 16字节                                          |

|  |        |  |  |
|--|--------|--|--|
|  | char * |  |  |
|--|--------|--|--|

**返回值：**大于0为命令发送成功，小于0为命令发送失败

### 3.6.13 Mifare Plus 卡 Level 3 级通用读写第二次授权

**函数原形：**int YW\_MFP\_Authorization\_Follow(int ReaderID, int AESKeyAddr, unsigned char \*AESKey);

**参数列表：**

| 参数         | 类型              | 方向 | 含义                                                  |
|------------|-----------------|----|-----------------------------------------------------|
| ReaderID   | int             | IN | 所要获取的设备标示ID，范围0x0000~0xFFFF，如果未知，则ReaderID=0        |
| AESKeyAddr | int             | IN | AES 密钥地址(范围0x0000~0xffff)<br>密钥对应保护的区块见MifarePlus手册 |
| AESKey     | unsigned char * | IN | AES密钥, 16字节                                         |

**返回值：**大于0为命令发送成功，小于0为命令发送失败

### 3.6.14 Mifare Plus 卡 Level 3 级通用读块

**函数原形：**int YW\_MFP\_CommonRead(int ReaderID, int BlockID, int BlockNums, int \*DataLen, unsigned char \*pData);

**参数列表：**

| 参数        | 类型              | 方向  | 含义                                           |
|-----------|-----------------|-----|----------------------------------------------|
| ReaderID  | int             | IN  | 所要获取的设备标示ID，范围0x0000~0xFFFF，如果未知，则ReaderID=0 |
| BlockID   | int             | IN  | 通用块地址(范围0x0000~0xffff)                       |
| BlockNums | int             | IN  | 块数量(建议在一个扇区内)                                |
| DataLen   | int*            | OUT | 返回的数据长度                                      |
| pData     | unsigned char * | OUT | 返回的数据(每个块数据长度16字                             |

|  |  |  |                       |
|--|--|--|-----------------------|
|  |  |  | 节, 总字节= BlockNums*16) |
|--|--|--|-----------------------|

返回值: 大于0为命令发送成功, 小于0为命令发送失败

### 3.6.15 Mifare Plus 卡 Level 3 级通用写块

函数原形: `int YW_MFP_CommonWrite(int ReaderID, int BlockID, int BlockNums, unsigned char *pData);`

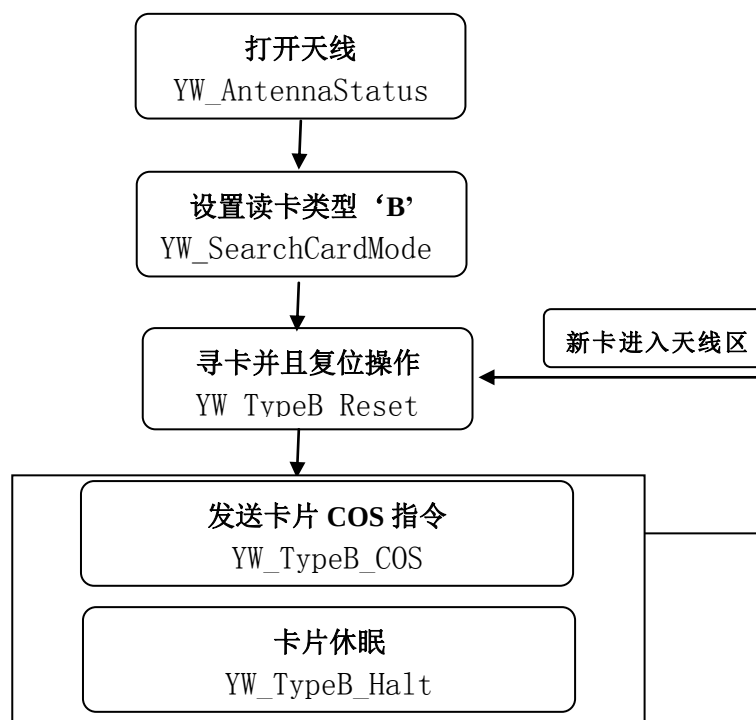
参数列表:

| 参数        | 类型              | 方向 | 含义                                              |
|-----------|-----------------|----|-------------------------------------------------|
| ReaderID  | int             | IN | 所要获取的设备标示ID, 范围0x0000~0xFFFF, 如果未知, 则ReaderID=0 |
| BlockID   | int             | IN | 通用块地址(范围0x0000~0xffff)                          |
| BlockNums | int             | IN | 块数量(建议在一个扇区内)                                   |
| pData     | unsigned char * | IN | 要写入的数据, 长度必须为16*BlockNums                       |

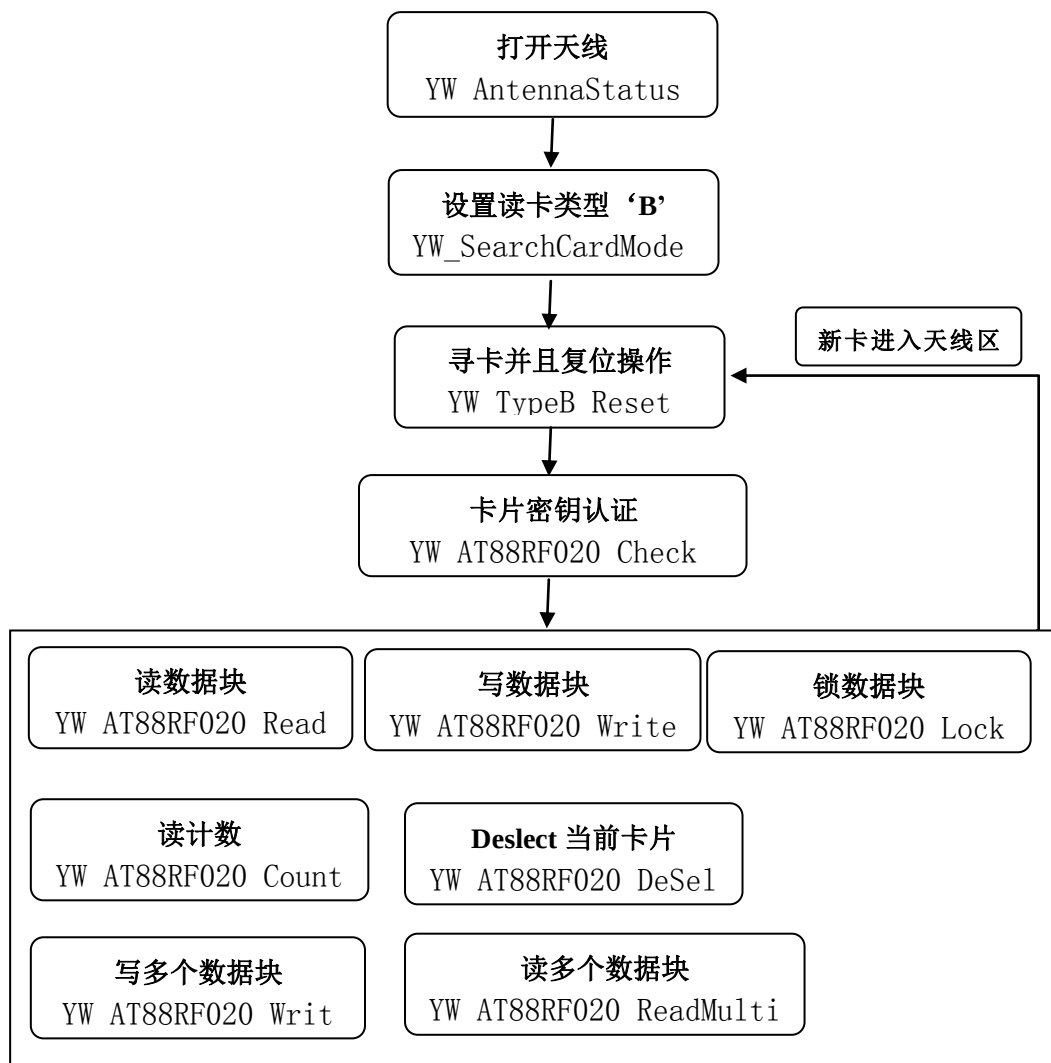
返回值: 大于0为命令发送成功, 小于0为命令发送失败

## 3.7 ISO14443 TypeB

ISO14443-4 TypeB CPU卡操作流程:

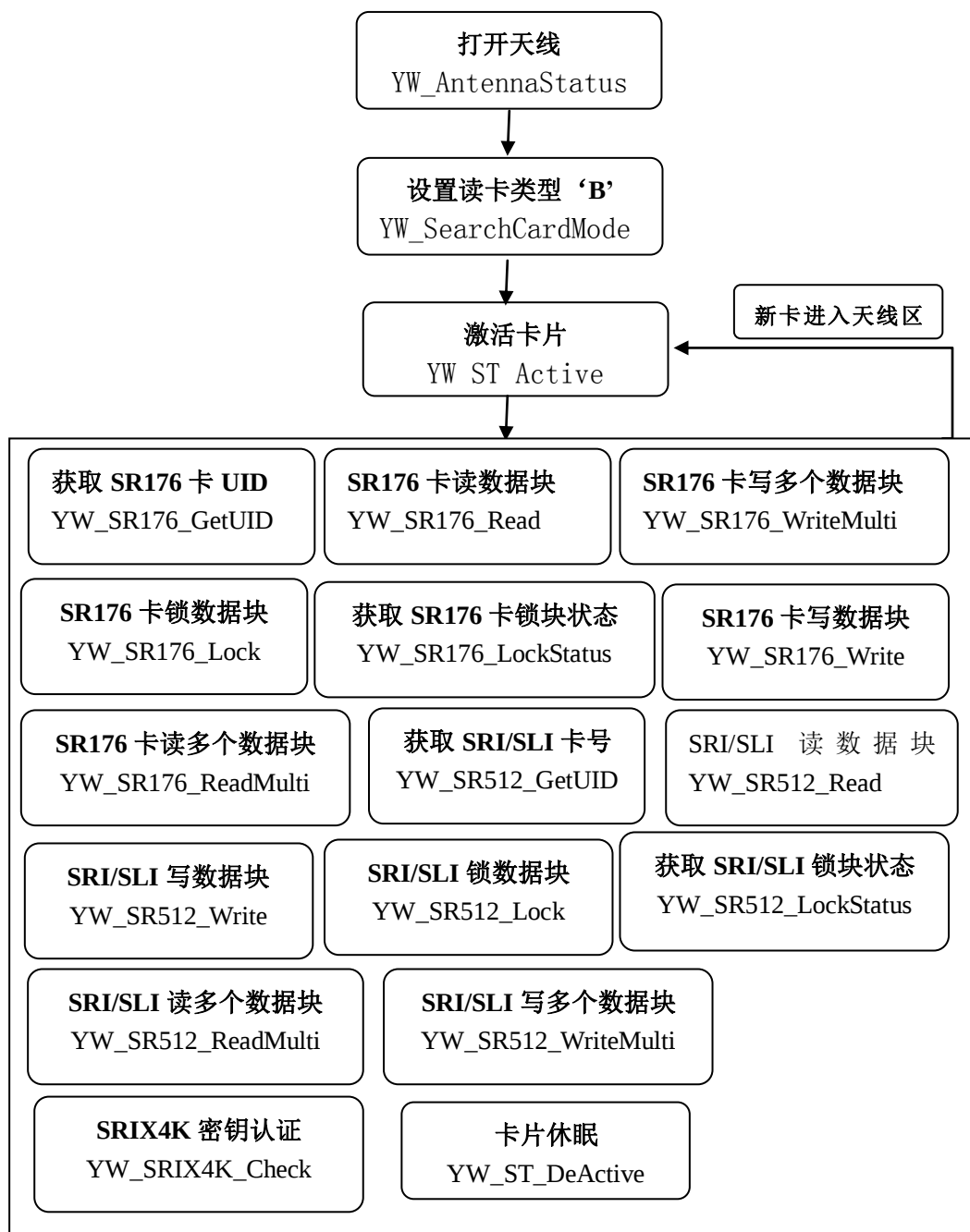


(图-8 ISO14443-4 TypeB CPU卡 操作流程图)

**AT88RF020/AT88RF080 卡片操作函数**

(图-9 AT88RF020/AT88RF080卡片 操作流程图)

## SR176、SRI512/1k/2k/4k、SLIX4K操作流程



(图-10 SR176、SRI512/1k/2k/4k、SLIX4K卡 操作流程图)

### 3.7.1 ISO14443 Type B 卡复位

**函数原形：** `int YW_TypeB_Reset(int ReaderID, unsigned char Mode, int *rtLen, unsigned char *pData);`

**参数列表：**

| 参数       | 类型              | 方向  | 含义                                           |
|----------|-----------------|-----|----------------------------------------------|
| ReaderID | int             | IN  | 所要获取的设备标示ID，范围0x0000-0xFFFF，如果未知，则ReaderID=0 |
| Mode     | unsigned char   | IN  | 寻卡的模式：<br>0x01->所有卡<br>0x00->未休眠卡            |
| rtLen    | int *           | OUT | Type B卡复位信息的长度                               |
| pData    | unsigned char * | OUT | Type B卡复位信息                                  |

**返回值：** 大于0为命令发送成功，小于0为命令发送失败

### 3.7.2 ISO14443 Type B 卡复位（扩展函数）

**函数原形：** `int YW_TypeB_ResetEx(int ReaderID, unsigned char Mode, unsigned char *Attrib, int *rtLen, unsigned char *pData);`

**参数列表：**

| 参数       | 类型              | 方向 | 含义                                           |
|----------|-----------------|----|----------------------------------------------|
| ReaderID | int             | IN | 所要获取的设备标示ID，范围0x0000-0xFFFF，如果未知，则ReaderID=0 |
| Mode     | unsigned char   | IN | 寻卡的模式：<br>0x01->所有卡<br>0x00->未休眠卡            |
| Attrib   | unsigned char * | IN | 复位时的Attrib参数，4个字节                            |

|       |                 |     |                |
|-------|-----------------|-----|----------------|
| rtLen | int *           | OUT | Type B卡复位信息的长度 |
| pData | unsigned char * | OUT | Type B卡复位信息    |

**返回值：**大于0为命令发送成功，小于0为命令发送失败

### 3.7.3 ISO14443 Type B 卡 COS 命令

**函数原形：**int YW\_TypeB\_COS(int ReaderID, int LenCOS, unsigned char \*Com\_COS, int \*DataLen, unsigned char \*pData);

**参数列表：**

| 参数       | 类型              | 方向  | 含义                                           |
|----------|-----------------|-----|----------------------------------------------|
| ReaderID | int             | IN  | 所要获取的设备标示ID，范围0x0000-0xFFFF，如果未知，则ReaderID=0 |
| LenCOS   | int             | IN  | 执行的COS命令的长度                                  |
| Com_COS  | unsigned char*  | IN  | 要执行的COS命令                                    |
| rtLen    | int *           | OUT | 执行COS后返回的数据的长度                               |
| pData    | unsigned char * | OUT | 执行COS后返回的数据                                  |

**返回值：**大于0为命令发送成功，小于0为命令发送失败

### 3.7.4 ISO14443 Type B 卡 Halt

**函数原形：**int YW\_TypeB\_Halt(ReaderID, unsigned char \*PUPI);

**参数列表：**

| 参数       | 类型              | 方向 | 含义                                           |
|----------|-----------------|----|----------------------------------------------|
| ReaderID | int             | IN | 所要获取的设备标示ID，范围0x0000-0xFFFF，如果未知，则ReaderID=0 |
| PUPI     | unsigned char * | IN | 卡片PUPI                                       |

**返回值：**大于0为命令发送成功，小于0为命令发送失败

### 3.7.5 获取中国居民身份证卡号

**函数原形：** `int YW_ChinaIDV2_RequestCardNo(int ReaderID, unsigned char *LengthOfCardNo, unsigned char *CardNo);`

**参数列表：**

| 参数             | 类型              | 方向  | 含义                                             |
|----------------|-----------------|-----|------------------------------------------------|
| ReaderID       | int             | IN  | 所要获取的设备标示ID，范围 0x0000-0xFFFF，如果未知，则 ReaderID=0 |
| LengthOfCardNo | unsigned char * | OUT | 居民二代证卡号长度                                      |
| CardNo         | unsigned char * | OUT | 返回居民二代证卡号                                      |

**返回值：** 大于0为命令发送成功，小于0为命令发送失败

**注意：** 仅仅是卡号，不是身份证号。

### 3.7.6 AT88RF020/080 密钥认证

**函数原形：** `int YW_AT88RF020_Check(int ReaderID, unsigned char *Key);`

**参数列表：**

| 参数       | 类型              | 方向 | 含义                                             |
|----------|-----------------|----|------------------------------------------------|
| ReaderID | int             | IN | 所要获取的设备标示ID，范围 0x0000-0xFFFF，如果未知，则 ReaderID=0 |
| Key      | unsigned char * | IN | 密钥，8 字节，出厂密钥是8个0x00                            |

**返回值：** 大于0为命令发送成功，小于0为命令发送失败

### 3.7.7 AT88RF020/080 读数据块

**函数原形：** `int YW_AT88RF020_Read(int ReaderID, int BlockID, unsigned char *pData);`

**参数列表：**

| 参数       | 类型              | 方向  | 含义                                                |
|----------|-----------------|-----|---------------------------------------------------|
| ReaderID | int             | IN  | 所要获取的设备标示ID, 范围 0x0000-0xFFFF, 如果未知, 则 ReaderID=0 |
| BlockID  | int             | IN  | 要读取的数据块地址                                         |
| pData    | unsigned char * | OUT | 读取的数据(每个块8字节数据)                                   |

**返回值:** 大于0为命令发送成功, 小于0为命令发送失败

### 3.7.8 AT88RF020/080 写数据块

**函数原形:** `int YW_AT88RF020_Write(int ReaderID, int BlockID, unsigned char *pData);`

**参数列表:**

| 参数       | 类型              | 方向 | 含义                                                |
|----------|-----------------|----|---------------------------------------------------|
| ReaderID | int             | IN | 所要获取的设备标示ID, 范围 0x0000-0xFFFF, 如果未知, 则 ReaderID=0 |
| BlockID  | int             | IN | 要写入的数据块号                                          |
| pData    | unsigned char * | IN | 要写入的数据(每个块8字节数据)                                  |

**返回值:** 大于0为命令发送成功, 小于0为命令发送失败

### 3.7.9 AT88RF020/080 锁数据块

**函数原形:** `int YW_AT88RF020_Lock(int ReaderID, unsigned char *LockByte);`

**参数列表:**

| 参数       | 类型  | 方向 | 含义                                                |
|----------|-----|----|---------------------------------------------------|
| ReaderID | int | IN | 所要获取的设备标示ID, 范围 0x0000-0xFFFF, 如果未知, 则 ReaderID=0 |

|          |                 |    |                           |
|----------|-----------------|----|---------------------------|
| LockByte | unsigned char * | IN | 锁块数据(4字节, 对应32块锁存, 详情见手册) |
|----------|-----------------|----|---------------------------|

**返回值:** 大于0为命令发送成功, 小于0为命令发送失败

### 3.7.10 AT88RF020/080 读取计数

**函数原形:** `int YW_AT88RF020_Count(int ReaderID, unsigned char *Signature);`

**参数列表:**

| 参数        | 类型              | 方向  | 含义                                              |
|-----------|-----------------|-----|-------------------------------------------------|
| ReaderID  | int             | IN  | 所要获取的设备标示ID, 范围0x0000-0xFFFF, 如果未知, 则ReaderID=0 |
| Signature | unsigned char * | OUT | 签名, 6 字节                                        |

**返回值:** 大于0为命令发送成功, 小于0为命令发送失败

### 3.7.11 AT88RF020/080 Deslect 当前卡片

**函数原形:** `int YW_AT88RF020_DeSel(int ReaderID);`

**参数列表:**

| 参数       | 类型  | 方向 | 含义                                              |
|----------|-----|----|-------------------------------------------------|
| ReaderID | int | IN | 所要获取的设备标示ID, 范围0x0000-0xFFFF, 如果未知, 则ReaderID=0 |

**返回值:** 大于0为命令发送成功, 小于0为命令发送失败

### 3.7.12 AT88RF020/080 读取多块

**函数原形:** `int YW_AT88RF020_ReadMulti(int ReaderID, int BlockID, int BlockNums, int *LenData, unsigned char *pData);`

**参数列表:**

| 参数 | 类型 | 方向 | 含义 |
|----|----|----|----|
|----|----|----|----|

|           |                 |     |                                                      |
|-----------|-----------------|-----|------------------------------------------------------|
| ReaderID  | int             | IN  | 所要获取的设备标示ID，范围<br>0x0000-0xFFFF，如果未知，则<br>ReaderID=0 |
| BlockID   | int             | IN  | 要读取的起始块号                                             |
| BlockNums | int             | IN  | 块数量                                                  |
| LenData   | int*            | IN  | 返回的数据长度                                              |
| pData     | unsigned char * | OUT | 返回的数据                                                |

**返回值：**大于0为命令发送成功，小于0为命令发送失败

### 3.7.13 AT88RF020/080 写多块

**函数原形：**`int YW_AT88RF020_WriteMulti(int ReaderID, int BlockID, int BlockNums, int LenData, unsigned char *pData);`

**参数列表：**

| 参数        | 类型                 | 方向 | 含义                                               |
|-----------|--------------------|----|--------------------------------------------------|
| ReaderID  | int                | IN | 所要获取的设备标示ID，范围<br>0x0000-0xFFFF，如果未知，则ReaderID=0 |
| BlockID   | int                | IN | 要写入数据的起始块号                                       |
| BlockNums | int                | IN | 块数量                                              |
| LenData   | int                | IN | 要写入的数据长度                                         |
| pData     | unsigned<br>char * | IN | 要写入的数据                                           |

**返回值：**大于0为命令发送成功，小于0为命令发送失败

### 3.7.14 激活 S T 系列卡

**函数原形：**`int YW_ST_Active(int ReaderID, unsigned char *ChipID);`

**参数列表：**

| 参数       | 类型  | 方向 | 含义             |
|----------|-----|----|----------------|
| ReaderID | int | IN | 所要获取的设备标示ID，范围 |

|        |                    |     |                                  |
|--------|--------------------|-----|----------------------------------|
|        |                    |     | 0x0000-0xFFFF, 如果未知, 则ReaderID=0 |
| ChipID | unsigned<br>char * | OUT | 卡的Chip ID, 一个字节                  |

**返回值:** 大于0为命令发送成功, 小于0为命令发送失败

**注意:** 本功能适合SR176、SRI512/1K/2K/4K、SLIX4K卡。

### 3.7.15 S T卡休眠

**函数原形:** `int YW_ST_DeActive(int ReaderID);`

**参数列表:**

| 参数       | 类型  | 方向 | 含义                                              |
|----------|-----|----|-------------------------------------------------|
| ReaderID | int | IN | 所要获取的设备标示ID, 范围0x0000-0xFFFF, 如果未知, 则ReaderID=0 |

**返回值:** 大于0为命令发送成功, 小于0为命令发送失败

**注意:** 本功能适合SR176、SRI512/1K/2K/4K、SLIX4K卡。

### 3.7.16 SR176 卡获取 UID

**函数原形:** `int YW_SR176_GetUID(int ReaderID, unsigned char *UID);`

**参数列表:**

| 参数       | 类型              | 方向  | 含义                                              |
|----------|-----------------|-----|-------------------------------------------------|
| ReaderID | int             | IN  | 所要获取的设备标示ID, 范围0x0000-0xFFFF, 如果未知, 则ReaderID=0 |
| UID      | unsigned char * | OUT | SR176卡的UID, 8个字节                                |

**返回值:** 大于0为命令发送成功, 小于0为命令发送失败

### 3.7.17 SR176 卡读数据块

**函数原形:** `int YW_SR176_Read(int ReaderID, int BlockID, unsigned char *pData);`

**参数列表:**

| 参数       | 类型              | 方向  | 含义                                              |
|----------|-----------------|-----|-------------------------------------------------|
| ReaderID | int             | IN  | 所要获取的设备标示ID, 范围0x0000-0xFFFF, 如果未知, 则ReaderID=0 |
| BlockID  | int             | IN  | 所读数据块号                                          |
| pData    | unsigned char * | OUT | 返回读取的数据, 2个字节                                   |

**返回值:** 大于0为命令发送成功, 小于0为命令发送失败

### 3.7.18 SR176 卡写数据块

**函数原形:** `int YW_SR176_Write(int ReaderID, int BlockID, unsigned char *pData);`

**参数列表:**

| 参数       | 类型              | 方向 | 含义                                              |
|----------|-----------------|----|-------------------------------------------------|
| ReaderID | int             | IN | 所要获取的设备标示ID, 范围0x0000-0xFFFF, 如果未知, 则ReaderID=0 |
| BlockID  | int             | IN | 所写数据块块号                                         |
| pData    | unsigned char * | IN | 要写入的数据, 2个字节                                    |

**返回值:** 大于0为命令发送成功, 小于0为命令发送失败

### 3.7.19 SR176 卡锁数据块

**函数原形:** `int YW_SR176_Lock(int ReaderID, unsigned char LockByte);`

**参数列表:**

| 参数       | 类型  | 方向 | 含义                                              |
|----------|-----|----|-------------------------------------------------|
| ReaderID | int | IN | 所要获取的设备标示ID, 范围0x0000-0xFFFF, 如果未知, 则ReaderID=0 |

|          |               |    |                     |
|----------|---------------|----|---------------------|
| LockByte | unsigned char | IN | 锁数据，1个字节，详情见SR176手册 |
|----------|---------------|----|---------------------|

**返回值：**大于0为命令发送成功，小于0为命令发送失败

### 3.7.20 SR176 卡获取锁块状态

**函数原形：**`int YW_SR176_LockStatus(int ReaderID, unsigned char *LockByte);`

**参数列表：**

| 参数       | 类型             | 方向  | 含义                                           |
|----------|----------------|-----|----------------------------------------------|
| ReaderID | int            | IN  | 所要获取的设备标示ID，范围0x0000-0xFFFF，如果未知，则ReaderID=0 |
| LockByte | unsigned char* | OUT | 返回锁块状态数据，一个字节                                |

**返回值：**大于0为命令发送成功，小于0为命令发送失败

### 3.7.21 SR176 卡读取多个数据块

**函数原形：**`int YW_SR176_ReadMulti(int ReaderID, int BlockID, int BlockNums, int *LenData, unsigned char *pData);`

**参数列表：**

| 参数        | 类型             | 方向  | 含义                                           |
|-----------|----------------|-----|----------------------------------------------|
| ReaderID  | int            | IN  | 所要获取的设备标示ID，范围0x0000-0xFFFF，如果未知，则ReaderID=0 |
| BlockID   | int            | IN  | 读取的起始块号                                      |
| BlockNums | int            | IN  | 块数量                                          |
| LenData   | int*           | OUT | 返回读取的数据长度                                    |
| pData     | unsigned char* | OUT | 返回读取的块数据                                     |

**返回值：**大于0为命令发送成功，小于0为命令发送失败

### 3.7.22 SR176 卡写多个数据块

**函数原形：** `int YW_SR176_WriteMulti(int ReaderID, int BlockID, int BlockNums, int LenData, unsigned char *pData);`

**参数列表：**

| 参数        | 类型             | 方向 | 含义                                             |
|-----------|----------------|----|------------------------------------------------|
| ReaderID  | int            | IN | 所要获取的设备标示ID，范围 0x0000-0xFFFF，如果未知，则 ReaderID=0 |
| BlockID   | int            | IN | 写入块的起始块号                                       |
| BlockNums | int            | IN | 块数量                                            |
| LenData   | int            | IN | 要写入的数据长度                                       |
| pData     | unsigned char* | IN | 要写入的数据                                         |

**返回值：** 大于0为命令发送成功，小于0为命令发送失败

### 3.7.23 SRI/SLI 卡获取 UID

**函数原形：** `int YW_SR512_GetUID(int ReaderID, unsigned char *UID);`

**参数列表：**

| 参数       | 类型             | 方向  | 含义                                             |
|----------|----------------|-----|------------------------------------------------|
| ReaderID | int            | IN  | 所要获取的设备标示ID，范围 0x0000-0xFFFF，如果未知，则 ReaderID=0 |
| UID      | unsigned char* | OUT | UID, 8个字节                                      |

**返回值：** 大于0为命令发送成功，小于0为命令发送失败

**注意：** 本功能适合SRI512/1K/2K/4K、SLIX4K卡。

### 3.7.24 SRI/SLI 卡读数据块

**函数原形：** `int YW_SR512_Read(int ReaderID, int BlockID, unsigned char *pData);`

**参数列表：**

| 参数       | 类型             | 方向  | 含义                                             |
|----------|----------------|-----|------------------------------------------------|
| ReaderID | int            | IN  | 所要获取的设备标示ID，范围 0x0000-0xFFFF，如果未知，则 ReaderID=0 |
| BlockID  | int            | IN  | 所读块的块号                                         |
| pData    | unsigned char* | OUT | 返回块数据, 4个字节                                    |

**返回值：** 大于0为命令发送成功，小于0为命令发送失败

**注意：** 本功能适合SRI512/1K/2K/4K、SLIX4K卡。

### 3.7.25 SRI/SLI 卡写数据块

**函数原形：** `int YW_SR512_Write(int ReaderID, int BlockID, unsigned char *pData);`

**参数列表：**

| 参数       | 类型             | 方向 | 含义                                             |
|----------|----------------|----|------------------------------------------------|
| ReaderID | int            | IN | 所要获取的设备标示ID，范围 0x0000-0xFFFF，如果未知，则 ReaderID=0 |
| BlockID  | int            | IN | 所写块的块号                                         |
| pData    | unsigned char* | IN | 要写入的数据, 4个字节                                   |

**返回值：** 大于0为命令发送成功，小于0为命令发送失败

**注意：** 本功能适合SRI512/1K/2K/4K、SLIX4K卡。

### 3.7.26 SRI/SLI 卡锁数据块

**函数原形：** `int YW_SR512_Lock(int ReaderID, unsigned short LockByte);`

**参数列表:**

| 参数       | 类型             | 方向 | 含义                                              |
|----------|----------------|----|-------------------------------------------------|
| ReaderID | int            | IN | 所要获取的设备标示ID, 范围0x0000-0xFFFF, 如果未知, 则ReaderID=0 |
| LockByte | unsigned short | IN | 锁数据 (2字节) 详情见卡手册                                |

**返回值:** 大于0为命令发送成功, 小于0为命令发送失败

**注意:** 本功能适合SRI512/1K/2K/4K、SLIX4K卡。

### 3.7.27 SRI/SLI 卡获取锁块状态

**函数原形:** `int YW_SR512_LockStatus(int ReaderID, unsigned short *LockByte)`

**参数列表:**

| 参数       | 类型              | 方向 | 含义                                              |
|----------|-----------------|----|-------------------------------------------------|
| ReaderID | int             | IN | 所要获取的设备标示ID, 范围0x0000-0xFFFF, 如果未知, 则ReaderID=0 |
| LockByte | unsigned short* | IN | 锁块数据 (2字节) 详情见卡手册                               |

**返回值:** 大于0为命令发送成功, 小于0为命令发送失败

**注意:** 本功能适合SRI512/1K/2K/4K、SLIX4K卡。

### 3.7.28 SRI/SLI 卡读多块

**函数原形:** `int YW_SR512_ReadMulti(int ReaderID, int BlockID, int BlockNums, int *LenData, unsigned char *pData)`

**参数列表:**

| 参数       | 类型  | 方向 | 含义                                    |
|----------|-----|----|---------------------------------------|
| ReaderID | int | IN | 所要获取的设备标示ID, 范围0x0000-0xFFFF, 如果未知, 则 |

|           |                 |     |            |
|-----------|-----------------|-----|------------|
|           |                 |     | ReaderID=0 |
| BlockID   | int             | IN  | 所读数据的起始块号  |
| BlockNums | int             | IN  | 块数量        |
| LenData   | int*            | OUT | 返回的数据长度    |
| pData     | unsigned char * | OUT | 返回的数据      |

**返回值：**大于0为命令发送成功，小于0为命令发送失败

**注意：**本功能适合SRI512/1K/2K/4K、SLIX4K卡。

### 3.7.29 SRI/SLI 卡写多块

**函数原形：**int YW\_SR512\_WriteMulti(int ReaderID, int BlockID, int BlockNums, int LenData, unsigned char \*pData)

**参数列表：**

| 参数        | 类型              | 方向 | 含义                                           |
|-----------|-----------------|----|----------------------------------------------|
| ReaderID  | int             | IN | 所要获取的设备标示ID，范围0x0000-0xFFFF，如果未知，则ReaderID=0 |
| BlockID   | int             | IN | 所写数据的起始块号                                    |
| BlockNums | int             | IN | 块数量                                          |
| LenData   | int             | IN | 写入数据长度                                       |
| pData     | unsigned char * | IN | 要写入的数据                                       |

**返回值：**大于0为命令发送成功，小于0为命令发送失败

**注意：**本功能适合SRI512/1K/2K/4K、SLIX4K卡。

### 3.7.30 SRIX4K 卡密钥认证

**函数原形：**int YW\_SRIX4K\_Check(int ReaderID, unsigned char \*Key, unsigned char \*Signature)

**参数列表：**

| 参数 | 类型 | 方向 | 含义 |
|----|----|----|----|
|----|----|----|----|

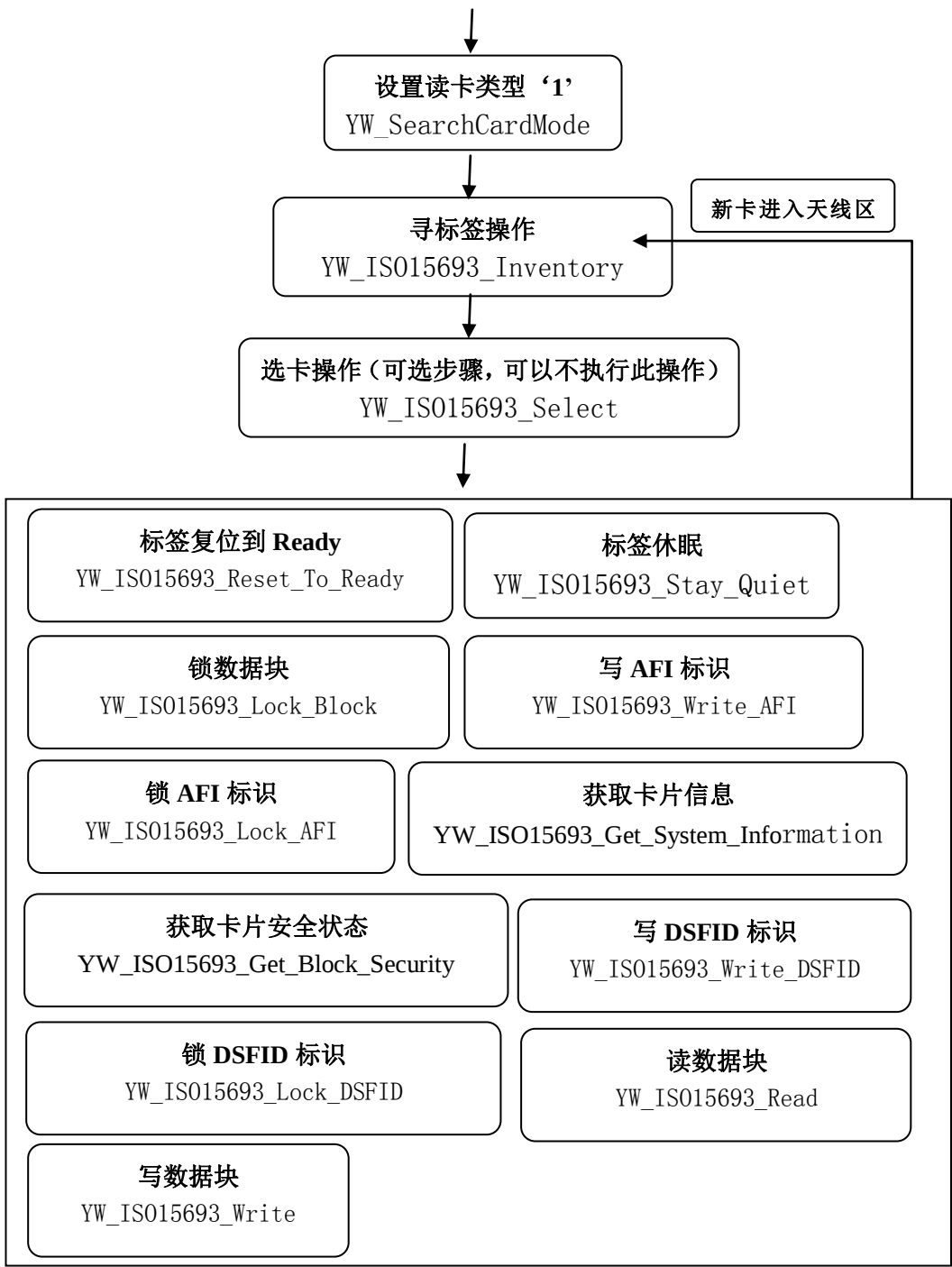
|           |                |     |                                                  |
|-----------|----------------|-----|--------------------------------------------------|
| ReaderID  | int            | IN  | 所要获取的设备标示ID，范围<br>0x0000-0xFFFF，如果未知，则ReaderID=0 |
| Key       | unsigned char* | IN  | 密钥， 6 个字节                                        |
| Signature | unsigned char* | OUT | 输出签名， 3 个字节                                      |

返回值：大于0为命令发送成功，小于0为命令发送失败

### 3.8 ISO15693 标签

ISO15693 Tag操作流程

打开天线  
YW\_AntennaStatus



(图-11 IS015693 标签操作流程图)

### 3.8.1 IS015693 标签 Inventory

函数原形: `int YW_IS015693_Inventory(int ReaderID, unsigned char *PData, unsigned char *PLen);`

参数列表:

| 参数       | 类型              | 方向  | 含义                                                  |
|----------|-----------------|-----|-----------------------------------------------------|
| ReaderID | int             | IN  | 所要获取的设备标示ID, 范围<br>0x0000-0xFFFF, 如果未知, 则ReaderID=0 |
| PData    | unsigned char * | OUT | 如果成功, 则返回数据<br>Flag(1byte)+DSFID(1Byte)+UID(8Byte)  |
| PLen     | unsigned char * | OUT | PData数据的长度                                          |

**返回值:** 大于0为成功, 小于0为失败

### 3.8.2 IS015693 标签 Stay Quiet

**函数原形:** `int YW_IS015693_Stay_Quiet(int ReaderID, unsigned char *PUID);`

**参数列表:**

| 参数       | 类型              | 方向 | 含义                                                      |
|----------|-----------------|----|---------------------------------------------------------|
| ReaderID | int             | IN | 所要获取的设备标示ID, 范围<br>0x0000-0xFFFF, 如果未知, 则<br>ReaderID=0 |
| PUID     | unsigned char * | IN | 卡的序列号UID(8Byte)                                         |

**返回值:** 大于0为成功, 小于0为失败

### 3.8.3 IS015693 标签选卡

**函数原形:** `int YW_IS015693_Select(int ReaderID, unsigned char Model, unsigned char *PUID);`

**参数列表:**

| 参数       | 类型              | 方向 | 含义                                                      |
|----------|-----------------|----|---------------------------------------------------------|
| ReaderID | int             | IN | 所要获取的设备标示ID, 范围<br>0x0000-0xFFFF, 如果未知, 则<br>ReaderID=0 |
| Model    | unsigned char   | IN | 默认: 0x02                                                |
| PUID     | unsigned char * | IN | 卡的序列号UID(8Byte)                                         |

**返回值:** 大于0为成功, 小于0为失败

**注意：**

1. 执行该函数后，标签被选定，除了Inventory API外，所有其它API的Model可以选择0x00，表示读卡器一直都是对选定的卡片操作，此时附带的API中UID可以给任意值。若Model=0x02，则表示执行的是指定UID的卡片，此时API中的UID是Inventory得到的UID。

### 3.8.4 IS015693 标签复位

**函数原形：** `int YW_IS015693_Reset_To_Ready(int ReaderID, unsigned char Model, unsigned char *PUID);`

**参数列表：**

| 参数       | 类型              | 方向 | 含义                                                                                                                                                                                                                               |
|----------|-----------------|----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ReaderID | int             | IN | 所要获取的设备标示ID，范围0x0000-0xFFFF，如果未知，则ReaderID=0                                                                                                                                                                                     |
| Model    | unsigned char   | IN | Model标识（Flag）：<br>0x01->对已选择的标签进行操作，PUID可为任意值，Option_flag = 0；<br>0x02->对PUID指定的标签进行操作，PUID为当前操作标签的UID，Option_flag = 0；<br>0x04->对已选择的标签进行操作，PUID可为任意值，Option_flag = 1；<br>0x06->对PUID指定的标签进行操作，PUID为当前操作标签的UID，Option_flag = 1； |
| PUID     | unsigned char * | IN | 卡的序列号UID(8Byte)                                                                                                                                                                                                                  |

**返回值：** 大于0为成功，小于0为失败

**注意：**

1. IS015693 中Model，其实为IS015693中的Flag标志位。

**Model中位0 (bit0)：** IS015693 Flag中 Select\_flag位(选择标识位)。表示处于选择状态标签执行该指令，后UID为任意值，执行前必须先执行YW\_IS015693\_Select API操作选定卡片。

**Model中位1 (bit1)** : IS015693 Flag中 Addres\_flag位(地址标识位).

表示读卡器与本API给定UID一致的标签执行该指令, 执行前可以不必先执行YW\_IS015693\_Select API操作选定卡片。

**Model中位2 (bit2)** : IS015693中Option\_flag位, 不同标签指令, 该值各不相同。

### 3.8.5 IS015693 标签读块

**函数原形:** `int YW_IS015693_Read(int ReaderID, unsigned char Model, unsigned char *PUID, unsigned char StartBlockID, unsigned char BlockNums, unsigned char *PData, unsigned char *PLen);`

**参数列表:**

| 参数           | 类型              | 方向 | 含义                                                                                                                                                                                                                                                           |
|--------------|-----------------|----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ReaderID     | int             | IN | 所要获取的设备标示ID, 范围0x0000-0xFFFF, 如果未知, 则ReaderID=0                                                                                                                                                                                                              |
| Model        | unsigned char   | IN | Model标识 (Flag) :<br>0x01->对已选择的标签进行操作,<br>PUID可为任意值, Option_flag = 0;<br>0x02->对PUID指定的标签进行操作,<br>PUID为当前操作标签的UID,<br>Option_flag = 0;<br>0x04->对已选择的标签进行操作,<br>PUID可为任意值, Option_flag = 1;<br>0x06->对PUID指定的标签进行操作,<br>PUID为当前操作标签的UID,<br>Option_flag = 1; |
| PUID         | unsigned char * | IN | 卡的序列号UID(8Byte)                                                                                                                                                                                                                                              |
| StartBlockID | unsigned char   | IN | 开始块号                                                                                                                                                                                                                                                         |

|           |                 |     |               |
|-----------|-----------------|-----|---------------|
| BlockNums | unsigned char   | IN  | 块数量           |
| PData     | unsigned char * | OUT | 读到的数据         |
| PLen      | unsigned char * | OUT | 读到的数据PData的长度 |

**返回值：**大于0为成功，小于0为失败

**注意：**

1. Model的解释见 YW\_IS015693\_Reset\_To\_Ready 中注释。
2. 若Option\_Flag = 1, 返回的数据带安全状态字节 (Block Security Status), 格式为: 块安全状态字节 + 块数据; 反之不带安全状态字节。

### 3.8.6 IS015693 标签写块

**函数原形：**int YW\_IS015693\_Write(int ReaderID, unsigned char Model, unsigned char \*PUID, unsigned char BlockID, unsigned char \*PData);

**参数列表：**

| 参数       | 类型              | 方向 | 含义                                                                                                                                                                                                                                         |
|----------|-----------------|----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ReaderID | int             | IN | 所要获取的设备标示ID, 范围 0x0000-0xFFFF, 如果未知, 则 ReaderID=0                                                                                                                                                                                          |
| Model    | unsigned char   | IN | Model标识 (Flag) :<br>0x01->对已选择的标签进行操作, PUID可为任意值, Option_flag = 0;<br>0x02->对PUID指定的标签进行操作, PUID为当前操作标签的UID, Option_flag = 0;<br>0x04->对已选择的标签进行操作, PUID可为任意值, Option_flag = 1;<br>0x06->对PUID指定的标签进行操作, PUID为当前操作标签的UID, Option_flag = 1; |
| PUID     | unsigned char * | IN | 卡的序列号UID(8Byte)                                                                                                                                                                                                                            |
| BlockID  | unsigned char   | IN | 操作的绝对块号块号                                                                                                                                                                                                                                  |

|       |                 |    |            |
|-------|-----------------|----|------------|
| PData | unsigned char * | IN | 写到Tag中的块数据 |
|-------|-----------------|----|------------|

**返回值：**大于0为成功，小于0为失败

**注意：**

1. *Model*的解释见 *YW\_IS015693\_Reset\_To\_Ready* 中注释。
2. *Option\_Flag* 见 *IS015693* 标准，和返回数据字节无关。

### 3.8.7 IS015693 标签锁块

**函数原形：**int YW\_IS015693\_Lock\_Block(int ReaderID, unsigned char Model, unsigned char \*PUID, unsigned char BlockID);

**参数列表：**

| 参数       | 类型              | 方向 | 含义                                                                                                                                                                                                                               |
|----------|-----------------|----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ReaderID | int             | IN | 所要获取的设备标示ID，范围0x0000-0xFFFF，如果未知，则ReaderID=0                                                                                                                                                                                     |
| Model    | unsigned char   | IN | Model标识（Flag）：<br>0x01->对已选择的标签进行操作，PUID可为任意值，Option_flag = 0；<br>0x02->对PUID指定的标签进行操作，PUID为当前操作标签的UID，Option_flag = 0；<br>0x04->对已选择的标签进行操作，PUID可为任意值，Option_flag = 1；<br>0x06->对PUID指定的标签进行操作，PUID为当前操作标签的UID，Option_flag = 1； |
| PUID     | unsigned char * | IN | 卡的序列号UID(8Byte)                                                                                                                                                                                                                  |
| BlockID  | unsigned char   | IN | 绝对块号                                                                                                                                                                                                                             |

**返回值：**大于0为成功，小于0为失败

**注意：**

1. *Model*的解释见 *YW\_IS015693\_Reset\_To\_Ready* 中注释。
2. *Option\_Flag* 见 *IS015693* 标准，和返回数据字节无关。

### 3.8.8 ISO15693 标签写 AFI

**函数原形:** `int YW_ISO15693_Write_AFI(int ReaderID, unsigned char Model, unsigned char *PUID, unsigned char AFI);`

**参数列表:**

| 参数       | 类型              | 方向 | 含义                                                                                                                                                                                                                                         |
|----------|-----------------|----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ReaderID | int             | IN | 所要获取的设备标示ID, 范围 0x0000-0xFFFF, 如果未知, 则 ReaderID=0                                                                                                                                                                                          |
| Model    | unsigned char   | IN | Model标识 (Flag) :<br>0x01->对已选择的标签进行操作, PUID可为任意值, Option_flag = 0;<br>0x02->对PUID指定的标签进行操作, PUID为当前操作标签的UID, Option_flag = 0;<br>0x04->对已选择的标签进行操作, PUID可为任意值, Option_flag = 1;<br>0x06->对PUID指定的标签进行操作, PUID为当前操作标签的UID, Option_flag = 1; |
| PUID     | unsigned char * | IN | 卡的序列号UID(8Byte)                                                                                                                                                                                                                            |
| AFI      | unsigned char   | IN | AFI值                                                                                                                                                                                                                                       |

**返回值:** 大于0为成功, 小于0为失败

**注意:**

1. Model的解释见 *YW\_ISO15693\_Reset\_To\_Ready* 中注释。
2. Option\_Flag 见ISO15693 标准, 和返回数据字节无关。

### 3.8.9 ISO15693 标签锁 AFI

**函数原形:** `int YW_ISO15693_Lock_AFI(int ReaderID, unsigned char Model, unsigned char *PUID);`

**参数列表:**

| 参数 | 类型 | 方向 | 含义 |
|----|----|----|----|
|----|----|----|----|

|          |                 |    |                                                                                                                                                                                                                                                  |
|----------|-----------------|----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ReaderID | int             | IN | 所要获取的设备标示ID，范围<br>0x0000-0xFFFF，如果未知，则<br>ReaderID=0                                                                                                                                                                                             |
| Model    | unsigned char   | IN | Model标识（Flag）：<br>0x01->对已选择的标签进行操作，PUID可为<br>任意值，Option_flag = 0；<br>0x02->对PUID指定的标签进行操作，PUID为<br>当前操作标签的UID，Option_flag = 0；<br>0x04->对已选择的标签进行操作，PUID可为<br>任意值，Option_flag = 1；<br>0x06->对PUID指定的标签进行操作，PUID为<br>当前操作标签的UID，Option_flag = 1； |
| PUID     | unsigned char * | IN | 卡的序列号UID(8Byte)                                                                                                                                                                                                                                  |

**返回值：**大于0为成功，小于0为失败

**注意：**

1. Model的解释见 *YW\_IS015693\_Reset\_To\_Ready* 中注释。

2. Option\_Flag 见IS015693 标准，和返回数据字节无关。

### 3.8.10 IS015693 标签写 DSFID

**函数原形：** `int YW_IS015693_Write_DSFIID(int ReaderID, unsigned char Model, unsigned char *PUID, unsigned char DSFIID);`

**参数列表：**

| 参数       | 类型            | 方向 | 含义                                                                 |
|----------|---------------|----|--------------------------------------------------------------------|
| ReaderID | int           | IN | 所要获取的设备标示ID，范围<br>0x0000-0xFFFF，如果未知，则<br>ReaderID=0               |
| Model    | unsigned char | IN | Model标识（Flag）：<br>0x01->对已选择的标签进行操作，PUID可<br>为任意值，Option_flag = 0； |

|       |                 |    |                                                                                                                                                                                      |
|-------|-----------------|----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|       |                 |    | 0x02->对PUID指定的标签进行操作, PUID<br>为当前操作标签的UID, Option_flag = 0;<br>0x04->对已选择的标签进行操作, PUID可<br>为<br>任意值, Option_flag = 1;<br>0x06->对PUID指定的标签进行操作, PUID<br>为当前操作标签的UID, Option_flag = 1; |
| PUID  | unsigned char * | IN | 卡的序列号UID(8Byte)                                                                                                                                                                      |
| DSFID | unsigned char   | IN | DSFID值                                                                                                                                                                               |

**返回值:** 大于0为成功, 小于0为失败

**注意:**

1. Model的解释见 YW\_IS015693\_Reset\_To\_Ready 中注释。

2. Option\_Flag 见IS015693 标准, 和返回数据字节无关。

### 3.8.11 IS015693 标签锁 DSFID

**函数原形:** int YW\_IS015693\_Lock\_DSFID(int ReaderID, unsigned char Model, unsigned char \*PUID);

**参数列表:**

| 参数       | 类型            | 方向 | 含义                                                                                                                                                                                          |
|----------|---------------|----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ReaderID | int           | IN | 所要获取的设备标示ID, 范围<br>0x0000-0xFFFF, 如果未知, 则<br>ReaderID=0                                                                                                                                     |
| Model    | unsigned char | IN | Model标识 (Flag) :<br>0x01->对已选择的标签进行操作, PUID可为<br>任意值, Option_flag = 0;<br>0x02->对PUID指定的标签进行操作, PUID为<br>当前操作标签的UID, Option_flag = 0;<br>0x04->对已选择的标签进行操作, PUID可为<br>任意值, Option_flag = 1; |

|      |                 |    |                                                         |
|------|-----------------|----|---------------------------------------------------------|
|      |                 |    | 0x06->对PUID指定的标签进行操作, PUID为当前操作标签的UID, Option_flag = 1; |
| PUID | unsigned char * | IN | 卡的序列号UID(8Byte)                                         |

**返回值:** 大于0为成功, 小于0为失败

**注意:**

1. Model的解释见 *YW\_IS015693\_Reset\_To\_Ready* 中注释。

2. Option\_Flag 见IS015693 标准, 和返回数据字节无关。

### 3.8.12 获取 IS015693 标签系统信息

**函数原形:** `int YW_IS015693_Get_System_Information(int ReaderID, unsigned char Model, unsigned char *PUID, unsigned char *PData, unsigned char *PLen);`

**参数列表:**

| 参数       | 类型              | 方向  | 含义                                                                                                                                                                                                                                         |
|----------|-----------------|-----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ReaderID | int             | IN  | 所要获取的设备标示ID, 范围0x0000-0xFFFF, 如果未知, 则ReaderID=0                                                                                                                                                                                            |
| Model    | unsigned char   | IN  | Model标识 (Flag) :<br>0x01->对已选择的标签进行操作, PUID可为任意值, Option_flag = 0;<br>0x02->对PUID指定的标签进行操作, PUID为当前操作标签的UID, Option_flag = 0;<br>0x04->对已选择的标签进行操作, PUID可为任意值, Option_flag = 1;<br>0x06->对PUID指定的标签进行操作, PUID为当前操作标签的UID, Option_flag = 1; |
| PUID     | unsigned char * | IN  | 卡的序列号UID(8Byte)                                                                                                                                                                                                                            |
| PData    | unsigned char * | OUT | 系统返回的数据                                                                                                                                                                                                                                    |
| PLen     | unsigned char * | OUT | 系统返回的数据PData的长度                                                                                                                                                                                                                            |

**返回值:** 大于0为成功, 小于0为失败

**注意：**

1. *Model*的解释见 *YW\_IS015693\_Reset\_To\_Ready* 中注释。

2. *Option\_Flag* 见*IS015693* 标准，和返回数据字节无关。

### 3.8.13 获取 IS015693 标签块安全信息

**函数原形：** `int YW_IS015693_Get_Block_Security(int ReaderID, unsigned char Model, unsigned char *PUID, unsigned char StartBlockID, unsigned char BlockNums, unsigned char *PData, unsigned char *PLen);`

**参数列表：**

| 参数           | 类型              | 方向  | 含义                                                                                                                                                                                                                                                           |
|--------------|-----------------|-----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ReaderID     | int             | IN  | 所要获取的设备标示ID，范围 0x0000-0xFFFF，如果未知，则 ReaderID=0                                                                                                                                                                                                               |
| Model        | unsigned char   | IN  | Model标识（Flag）：<br>0x01->对已选择的标签进行操作，<br>PUID可为任意值，Option_flag = 0;<br>0x02->对PUID指定的标签进行操作，<br>PUID为当前操作标签的UID，<br>Option_flag = 0;<br>0x04->对已选择的标签进行操作，<br>PUID可为<br>任意值，Option_flag = 1;<br>0x06->对PUID指定的标签进行操作，<br>PUID为当前操作标签的UID，<br>Option_flag = 1; |
| PUID         | unsigned char * | IN  | 卡的序列号UID(8Byte)                                                                                                                                                                                                                                              |
| StartBlockID | unsigned char   | IN  | 开始块号                                                                                                                                                                                                                                                         |
| BlockNums    | unsigned char   | IN  | 块数量                                                                                                                                                                                                                                                          |
| PData        | unsigned char * | OUT | 系统返回的数据                                                                                                                                                                                                                                                      |

|      |                 |     |                 |
|------|-----------------|-----|-----------------|
| PLen | unsigned char * | OUT | 系统返回的数据PData的长度 |
|------|-----------------|-----|-----------------|

**返回值：**大于0为成功，小于0为失败

**注意：**

1. *Model*的解释见 *YW\_IS015693\_Reset\_To\_Ready* 中注释。

2. *Option\_Flag* 见IS015693 标准，和返回数据字节无关。

### 3.8.14 IS015693 标签防冲突寻多张卡

**函数原形：** `int YW_IS015693_Multi_Inventory(int ReaderID, unsigned char AFIEEnable, unsigned char AFI, unsigned char *PData, unsigned char *PLen);`

**参数列表：**

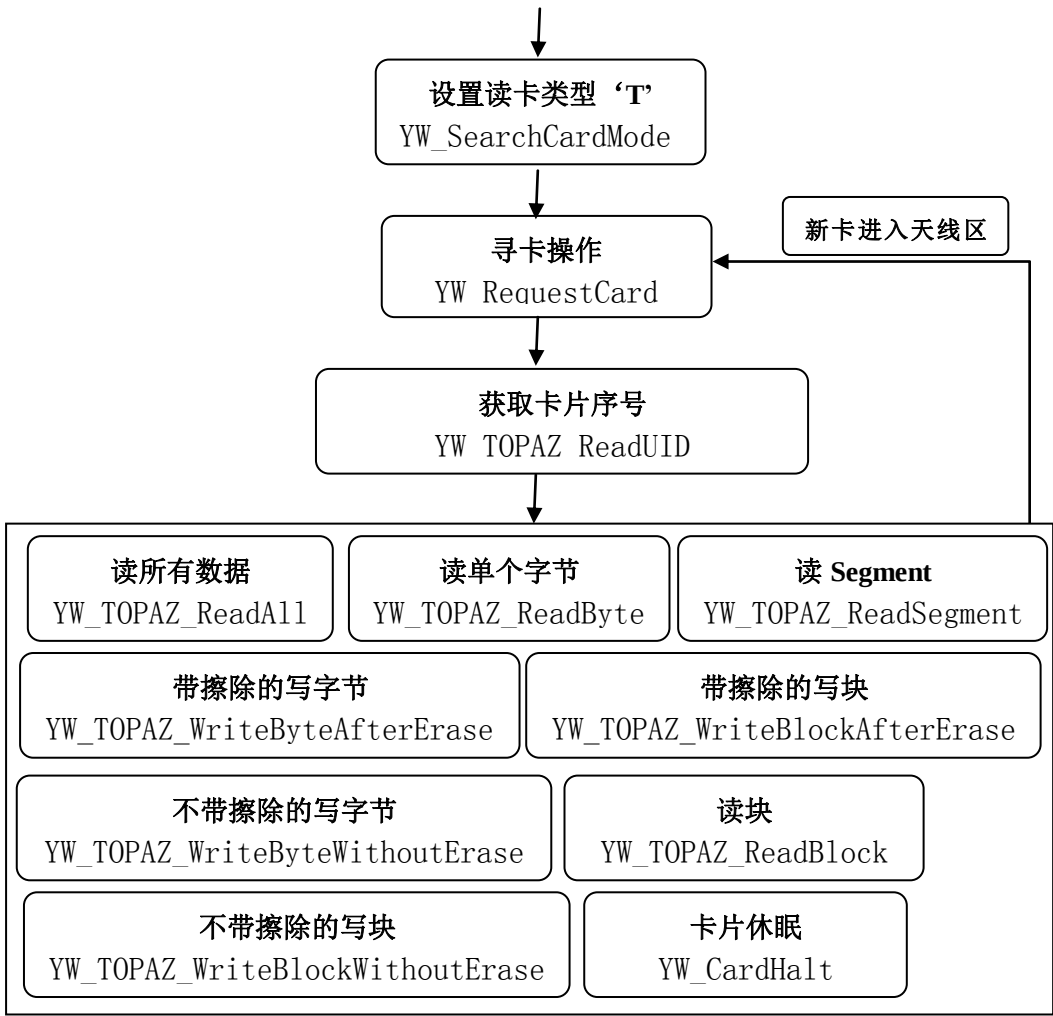
| 参数         | 类型              | 方向  | 含义                                           |
|------------|-----------------|-----|----------------------------------------------|
| ReaderID   | int             | IN  | 所要获取的设备标示ID，范围0x0000-0xFFFF，如果未知，则ReaderID=0 |
| AFIEEnable | unsigned char   | IN  |                                              |
| AFI        | unsigned char   | IN  |                                              |
| PData      | unsigned char * | OUT | 返回多个卡的序列号UID                                 |
| PLen       | unsigned char * | OUT | 多个卡序列号的长度                                    |

**返回值：**大于0为成功，小于0为失败

## 3.9 NFC-Type1 TOPAZ 标签

TOPAZ 卡片操作流程：

打开天线  
YW\_AntennaStatus



(图-12 TOPAZ卡 操作流程)

3.9.1 TOPAZ 卡片获取卡号

函数原形: `int YW_TOPAZ_ReadUID(int ReaderID, unsigned char *HR0, unsigned char *HR1, int *UIDLen, unsigned char *UID)`

参数列表:

| 参数        | 类型             | 方向  | 含义                                                |
|-----------|----------------|-----|---------------------------------------------------|
| Reader ID | int            | IN  | 所要获取的设备标示ID, 范围 0x0000-0xFFFF, 如果未知, 则 ReaderID=0 |
| HR0       | unsigned char* | OUT | TOPAZ卡的HR0                                        |
| HR1       | unsigned char* | OUT | TOPAZ卡的HR1                                        |

|        |               |     |             |
|--------|---------------|-----|-------------|
| UIDLen | int *         | OUT | TOPAZ卡的卡号长度 |
| UID    | unsigned char | OUT | TOPAZ卡的卡号   |

**返回值：**大于0为命令发送成功，≤0为命令发送失败

### 3.9.2 TOPAZ 卡片读所有数据

**函数原形：**`int YW_TOPAZ_ReadAll(int ReaderID, unsigned char *UID, int *DataLenRet, unsigned char *pDataRet)`

**参数列表：**

| 参数         | 类型             | 方向  | 含义                                           |
|------------|----------------|-----|----------------------------------------------|
| ReaderID   | int            | IN  | 所要获取的设备标示ID，范围0x0000-0xFFFF，如果未知，则ReaderID=0 |
| UID        | unsigned char* | IN  | TOPAZ卡的UID，目前4字节                             |
| DataLenRet | int *          | OUT | 读到的数据字节数                                     |
| pDataRet   | unsigned char  | OUT | 读到的数据                                        |

**返回值：**大于0为命令发送成功，≤0为命令发送失败

### 3.9.3 TOPAZ 卡片读单个字节

**函数原形：**`int YW_TOPAZ_ReadByte(int ReaderID, unsigned char *UID, unsigned char BlockID, unsigned char ByteAddr, unsigned char *pDataRet)`

**参数列表：**

| 参数       | 类型             | 方向 | 含义                                           |
|----------|----------------|----|----------------------------------------------|
| ReaderID | int            | IN | 所要获取的设备标示ID，范围0x0000-0xFFFF，如果未知，则ReaderID=0 |
| UID      | unsigned char* | IN | TOPAZ卡的UID，目前4字节                             |
| BlockID  | unsigned char  | IN | 要读取的块号                                       |

|          |                 |     |            |
|----------|-----------------|-----|------------|
| ByteAddr | unsigned char   | IN  | 要读取字节所在的地址 |
| pDataRet | unsigned char * | OUT | 读到的数据，一个字节 |

**返回值：** 大于0为命令发送成功，≤0为命令发送失败

### 3.9.4 TOPAZ 卡片先擦除再写单个字节

**函数原形：** `int YW_TOPAZ_WriteByteAfterErase(int ReaderID, unsigned char *UID, unsigned char BlockID, unsigned char ByteAddr, unsigned char DataWrite, unsigned char *pDataRet)`

**参数列表：**

| 参数        | 类型             | 方向  | 含义                                             |
|-----------|----------------|-----|------------------------------------------------|
| ReaderID  | int            | IN  | 所要获取的设备标示ID，范围 0x0000-0xFFFF，如果未知，则 ReaderID=0 |
| UID       | unsigned char* | IN  | TOPAZ卡的UID，目前4字节                               |
| BlockID   | unsigned char  | IN  | 要写入的块号                                         |
| ByteAddr  | unsigned char  | IN  | 要写入字节所在的地址                                     |
| DataWrite | unsigned char  | IN  | 要写入的数据内容，一个字节                                  |
| pDataRet  | unsigned char* | OUT | 写入后的数据内容，一个字节                                  |

**返回值：** 大于0为命令发送成功，≤0为命令发送失败

### 3.9.5 TOPAZ 卡片不带擦除写单个字节

**函数原形：** `int YW_TOPAZ_WriteByteWithoutErase(int ReaderID, unsigned char *UID, unsigned char BlockID, unsigned char ByteAddr, unsigned char DataWrite, unsigned char *pDataRet)`

**参数列表：**

| 参数       | 类型  | 方向 | 含义                                  |
|----------|-----|----|-------------------------------------|
| ReaderID | int | IN | 所要获取的设备标示ID，范围 0x0000-0xFFFF，如果未知，则 |

|           |                |     |                  |
|-----------|----------------|-----|------------------|
|           |                |     | ReaderID=0       |
| UID       | unsigned char* | IN  | TOPAZ卡的UID，目前4字节 |
| BlockID   | unsigned char  | IN  | 要写入的块号           |
| ByteAddr  | unsigned char  | IN  | 要写入字节所在的地址       |
| DataWrite | unsigned char  | IN  | 要写入的数据内容，一个字节    |
| pDataRet  | unsigned char* | OUT | 写入后的数据内容，一个字节    |

**返回值：**大于0为命令发送成功，≤0为命令发送失败

### 3.9.6 TOPAZ 卡片读 Segment

**函数原形：**`int YW_TOPAZ_ReadSegment(int ReaderID, unsigned char *UID, unsigned char Segment, int *DataLenRet, unsigned char *pDataRet)`

**参数列表：**

| 参数         | 类型             | 方向  | 含义                                           |
|------------|----------------|-----|----------------------------------------------|
| ReaderID   | int            | IN  | 所要获取的设备标示ID，范围0x0000-0xFFFF，如果未知，则ReaderID=0 |
| UID        | unsigned char* | IN  | TOPAZ卡的UID，目前4字节                             |
| Segment    | unsigned char  | IN  | 要读取的Segment，一个字节，0x00到0x03                   |
| DataLenRet | int *          | OUT | 读到的数据字节数                                     |
| pDataRet   | unsigned char* | OUT | 读到的数据                                        |

**返回值：**大于0为命令发送成功，≤0为命令发送失败

### 3.9.7 TOPAZ 卡片读块

**函数原形：**`int YW_TOPAZ_ReadBlock(int ReaderID, unsigned char *UID, unsigned char BlockID, unsigned char *pDataRet)`

**参数列表：**

| 参数 | 类型 | 方向 | 含义 |
|----|----|----|----|
|----|----|----|----|

|          |                |     |                                                         |
|----------|----------------|-----|---------------------------------------------------------|
| ReaderID | int            | IN  | 所要获取的设备标示ID, 范围<br>0x0000-0xFFFF, 如果未知, 则<br>ReaderID=0 |
| UID      | unsigned char* | IN  | TOPAZ卡的UID, 目前4字节                                       |
| BlockID  | unsigned char  | IN  | 要读取的块号                                                  |
| pDataRet | unsigned char* | OUT | 读到的数据, 8个字节                                             |

**返回值:** 大于0为命令发送成功, <=0为命令发送失败

### 3.9.8 TOPAZ 卡片先擦除再写块

**函数原形:** `int YW_TOPAZ_WriteBlockAfterErase(int ReaderID, unsigned char *UID, unsigned char BlockID, unsigned char *DataWrite, unsigned char *pDataRet)`

**参数列表:**

| 参数        | 类型             | 方向  | 含义                                                      |
|-----------|----------------|-----|---------------------------------------------------------|
| ReaderID  | int            | IN  | 所要获取的设备标示ID, 范围<br>0x0000-0xFFFF, 如果未知, 则<br>ReaderID=0 |
| UID       | unsigned char* | IN  | TOPAZ卡的UID, 目前4字节                                       |
| BlockID   | unsigned char  | IN  | 要写入的块号                                                  |
| DataWrite | unsigned char* | IN  | 要写入的数据, 8个字节                                            |
| pDataRet  | unsigned char* | OUT | 写入后的数据, 8个字节                                            |

**返回值:** 大于0为命令发送成功, <=0为命令发送失败

### 3.9.9 TOPAZ 卡片不带擦除写块

**函数原形:** `int YW_TOPAZ_WriteBlockWithoutErase(int ReaderID, unsigned char *UID, unsigned char BlockID, unsigned char *DataWrite, unsigned char *pDataRet)`

**参数列表:**

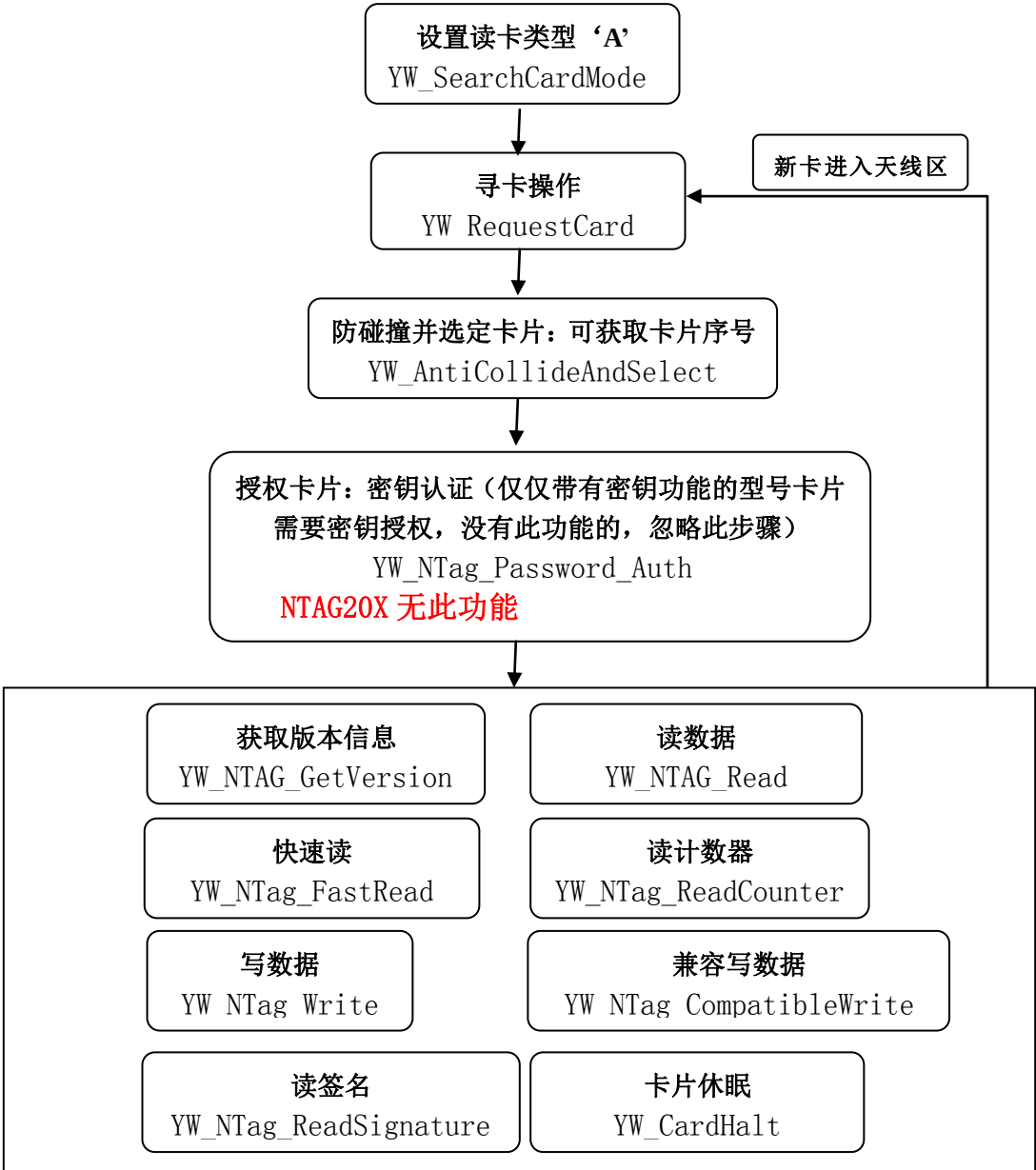
| 参数        | 类型             | 方向  | 含义                                           |
|-----------|----------------|-----|----------------------------------------------|
| ReaderID  | int            | IN  | 所要获取的设备标示ID，范围0x0000-0xFFFF，如果未知，则ReaderID=0 |
| UID       | unsigned char* | IN  | TOPAZ卡的UID，目前4字节                             |
| BlockID   | unsigned char  | IN  | 要写入的块号                                       |
| DataWrite | unsigned char* | IN  | 要写入的数据，8个字节                                  |
| pDataRet  | unsigned char* | OUT | 写入后的数据，8个字节                                  |

**返回值：**大于0为命令发送成功，<=0为命令发送失败

### 3.10 NFC-Type2 NTAG 系列标签

NTAG 标签包括 NTAG203、NTAG210/212/213/215/216 等卡片类型，不同的卡片可能支持的 API 可能不尽相同，详细信息请参考各个型号的数据手册。NTAG 操作流程如下：





(图-13 NTAG 标签操作流程)

3.10.1 NTAG 卡片获取版本信息

函数原形: `int YW_NTAG_GetVersion(int ReaderID, unsigned char *Version)`

参数列表:

| 参数       | 类型  | 方向 | 含义                                                  |
|----------|-----|----|-----------------------------------------------------|
| ReaderID | int | IN | 所要获取的设备标示ID, 范围<br>0x0000-0xFFFF, 如果未知, 则ReaderID=0 |

|         |                |     |                                                                                                                                                                                                                 |
|---------|----------------|-----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Version | unsigned char* | OUT | NTAG卡片的版本信息，8个字节， <b>NTAG20X无此指令。</b><br>NTAG21X执行此指令的返回结果中第6字节含义：<br><div style="text-align: right;">             0x0F为NTAG213卡<br/>             0x11为NTAG215卡<br/>             0x13为NTAG216卡           </div> |
|---------|----------------|-----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**返回值：**大于0为命令发送成功，≤0为命令发送失败

### 3.10.2 NTAG 卡片读数据

**函数原形：**int YW\_NTAG\_Read(int ReaderID, int StartPage, int \*DataLen, unsigned char \*pData)

**参数列表：**

| 参数        | 类型             | 方向  | 含义                                           |
|-----------|----------------|-----|----------------------------------------------|
| ReaderID  | int            | IN  | 所要获取的设备标示ID，范围0x0000-0xFFFF，如果未知，则ReaderID=0 |
| StartPage | int            | IN  | 要读取的开始页                                      |
| DataLen   | int*           | OUT | 读取得数据的字节数                                    |
| pData     | unsigned char* | OUT | 读取的数据，此处一般是读取4个数据页，即从开始页开始的连续的4个页，共16字节。     |

**返回值：**大于0为命令发送成功，≤0为命令发送失败

### 3.10.3 NTAG 卡片快速读数据

**函数原形：**int YW\_NTag\_FastRead(int ReaderID, int StartPage, int EndPage, int \*DataLen, unsigned char \*pData)

**参数列表：**

| 参数        | 类型             | 方向  | 含义                                           |
|-----------|----------------|-----|----------------------------------------------|
| ReaderID  | int            | IN  | 所要获取的设备标示ID，范围0x0000-0xFFFF，如果未知，则ReaderID=0 |
| StartPage | int            | IN  | 要读取的开始页                                      |
| EndPage   | int            | IN  | 要读取的结束页，由于各种限制，请保证读取页的跨度不要超过50页              |
| DataLen   | int*           | OUT | 读取得数据的字节数                                    |
| pData     | unsigned char* | OUT | 读取的数据                                        |

**返回值：**大于0为命令发送成功，≤0为命令发送失败

### 3.10.4 NTAG 卡片写数据

**函数原形：**int YW\_NTag\_Write(int ReaderID, int PageIndex, unsigned char \*pData)

**参数列表：**

| 参数        | 类型             | 方向 | 含义                                           |
|-----------|----------------|----|----------------------------------------------|
| ReaderID  | int            | IN | 所要获取的设备标示ID，范围0x0000-0xFFFF，如果未知，则ReaderID=0 |
| PageIndex | int            | IN | 要写入的页序号                                      |
| pData     | unsigned char* | IN | 写入的数据，4个字节                                   |

**返回值：**大于0为命令发送成功，≤0为命令发送失败

### 3.10.5 NTAG 卡片兼容写数据

**函数原形：**int YW\_NTag-CompatibleWrite(int ReaderID, int PageIndex, unsigned char \*pData)

**参数列表：**

| 参数 | 类型 | 方向 | 含义 |
|----|----|----|----|
|----|----|----|----|

|           |                |    |                                                      |
|-----------|----------------|----|------------------------------------------------------|
| ReaderID  | int            | IN | 所要获取的设备标示ID，范围<br>0x0000-0xFFFF，如果未知，则<br>ReaderID=0 |
| PageIndex | int            | IN | 要写入的页序号                                              |
| pData     | unsigned char* | IN | 写入的数据，16个字节，实际写入前<br>4个字节                            |

**返回值：**大于0为命令发送成功，≤0为命令发送失败

### 3.10.6 NTAG 卡片读计数器

**函数原形：**int YW\_NTag\_ReadCounter(int ReaderID, int \*Counter)

**参数列表：**

| 参数       | 类型    | 方向  | 含义                                               |
|----------|-------|-----|--------------------------------------------------|
| ReaderID | int   | IN  | 所要获取的设备标示ID，范围<br>0x0000-0xFFFF，如果未知，则ReaderID=0 |
| Counter  | int * | OUT | 读到的计数器内容                                         |

**返回值：**大于0为命令发送成功，≤0为命令发送失败

### 3.10.7 NTAG 卡片密码授权

**函数原形：**int YW\_NTag\_Password\_Auth(int ReaderID, unsigned char \*PWD, unsigned char \*AuthResult)

**参数列表：**

| 参数         | 类型             | 方向  | 含义                                                   |
|------------|----------------|-----|------------------------------------------------------|
| ReaderID   | int            | IN  | 所要获取的设备标示ID，范围<br>0x0000-0xFFFF，如果未知，则<br>ReaderID=0 |
| PWD        | unsigned char* | IN  | 密码，4个字节                                              |
| AuthResult | unsigned char* | OUT | 如果授权失败，此处显示失败的代<br>码，2字节                             |

**返回值：** 大于0为命令发送成功，<=0为命令发送失败

### 3.10.8 NTAG 卡片读签名

**函数原形：** `int YW_NTag_ReadSignature(int ReaderID, unsigned char *Signature)`

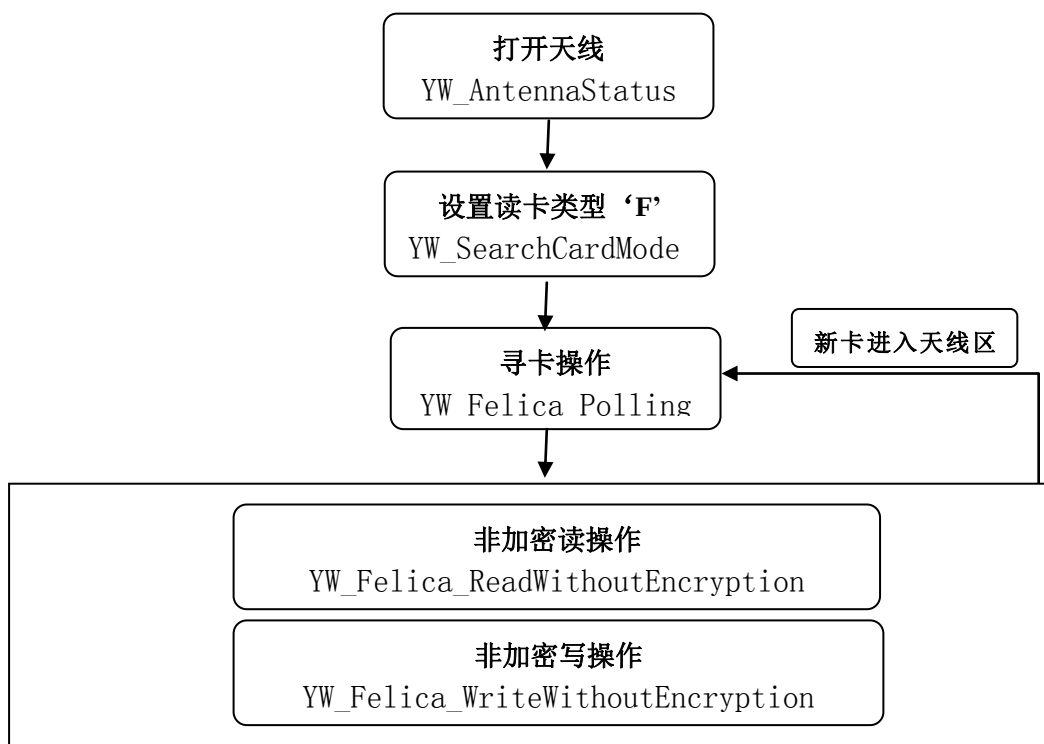
**参数列表：**

| 参数        | 类型             | 方向  | 含义                                           |
|-----------|----------------|-----|----------------------------------------------|
| ReaderID  | int            | IN  | 所要获取的设备标示ID，范围0x0000-0xFFFF，如果未知，则ReaderID=0 |
| Signature | unsigned char* | OUT | NTAG卡片的签名，32字节                               |

**返回值：** 大于0为命令发送成功，<=0为命令发送失败

## 3.11 NFC-Type3 Felica 卡

Felica 卡片包括加密的和非加密的卡片操作，目前提供非加密操作 API，后续加密 Felica 操作根据用户需求而增加。非加密 Felica 操作流程如下：



(图-14 TOPAZ卡 操作流程图)

### 3.11.1 Felica 卡片寻卡

函数原形：`int YW_Felica_Polling(int ReaderID, unsigned short SystemCode, unsigned char RequestCode, unsigned char Slot, unsigned char *IDM, unsigned char *PMm, unsigned char *RequestData)`

参数列表：

| 参数          | 类型             | 方向 | 含义                                             |
|-------------|----------------|----|------------------------------------------------|
| ReaderID    | int            | IN | 所要获取的设备标示ID，范围 0x0000-0xFFFF，如果未知，则 ReaderID=0 |
| SystemCode  | unsigned short | IN | Felica参数，System Code                           |
| RequestCode | unsigned char  | IN | Felica参数，Request Code                          |

|             |                |     |                                                                                                     |
|-------------|----------------|-----|-----------------------------------------------------------------------------------------------------|
|             |                |     | 0x00: No Request<br>0x01: System Code Request<br>0x02: Communication Performance Request<br>其他: RFU |
| Slot        | unsigned char  | IN  | 最大响应时间, 可以设置的值:<br>0x00, 0x01, 0x03, 0x07, 0x0F                                                     |
| IDM         | unsigned char* | OUT | 卡号, 8字节                                                                                             |
| PMm         | unsigned char* | OUT | PMm, 8字节                                                                                            |
| RequestData | unsigned char* | OUT | 只有当RequestCode=0x01和0x02<br>时, 才会返回Request data, 2字节                                                |

**返回值:** 大于0为命令发送成功, <=0为命令发送失败

### 3.11.2 Felica 卡片不加密读操作

**函数原形:** `int YW_Felica_ReadWithoutEncryption(int ReaderID, unsigned char *IDM, unsigned char NumberofService, unsigned char *ServiceCodeList, unsigned char NumberofBlock, unsigned char LengthofBlockList, unsigned char *BlockList, unsigned char *Status1, unsigned char *Status2, unsigned char *pBlockData)`

**参数列表:**

| 参数              | 类型             | 方向 | 含义                                              |
|-----------------|----------------|----|-------------------------------------------------|
| ReaderID        | int            | IN | 所要获取的设备标示ID, 范围0x0000-0xFFFF, 如果未知, 则ReaderID=0 |
| IDM             | unsigned char* | IN | Felica卡卡号, 8字节                                  |
| NumberofService | unsigned char  | IN | Service数量                                       |
| ServiceCodeList | unsigned char* | IN | Service内容, 字节数为2*NumberofService                |
| NumberofBlock   | unsigned char  | IN | Block块的数量                                       |

|                   |                |     |                                                                             |
|-------------------|----------------|-----|-----------------------------------------------------------------------------|
| LengthofBlockList | unsigned char  | IN  | Block列表的字节长度                                                                |
| BlockList         | unsigned char* | IN  | Block列表的内容                                                                  |
| Status1           | unsigned char* | OUT | 读取后的状态字节Status1, 1字节                                                        |
| Status2           | unsigned char* | OUT | 读取后的状态字节Status2, 1字节                                                        |
| pBlockData        | unsigned char* | OUT | 读到的块数据，只有当函数返回>0时，此处才有返回，字节数为<br>NumberofBlock*16，如果失败，请查看状态Status1，Status2 |

**返回值：**大于0为命令发送成功，≤0为命令发送失败

### 3.11.3 Felica 卡片不加密写操作

**函数原形：** `int YW_Felica_WriteWithoutEncryption(int ReaderID, unsigned char *IDM, unsigned char NumberofService, unsigned char *ServiceCodeList, unsigned char NumberofBlock, unsigned char LengthofBlockList, unsigned char *BlockList, unsigned char *pBlockData, unsigned char *Status1, unsigned char *Status2)`

**参数列表：**

| 参数              | 类型             | 方向 | 含义                                           |
|-----------------|----------------|----|----------------------------------------------|
| ReaderID        | int            | IN | 所要获取的设备标示ID，范围0x0000-0xFFFF，如果未知，则ReaderID=0 |
| IDM             | unsigned char* | IN | Felica卡卡号，8字节                                |
| NumberofService | unsigned char  | IN | Service数量                                    |
| ServiceCodeList | unsigned char* | IN | Service内容，字节数为2*                             |

|                   |                |     |                                  |
|-------------------|----------------|-----|----------------------------------|
|                   |                |     | NumberofService                  |
| NumberofBlock     | unsigned char  | IN  | Block块的数量                        |
| LengthofBlockList | unsigned char  | IN  | Block列表的字节长度                     |
| BlockList         | unsigned char* | IN  | Block列表的内容                       |
| pBlockData        | unsigned char* | IN  | 要写入的块数据，字节数为<br>NumberofBlock*16 |
| Status1           | unsigned char* | OUT | 如果写入失败，状态字节<br>Status1，1字节       |
| Status2           | unsigned char* | OUT | 如果写入失败，状态字节<br>Status2，1字节       |

**返回值：**大于0为命令发送成功，≤0为命令发送失败

## 4 订购方式

可以通过我们的网站或电话订购。或者联系当地的经销商。

### 北京友我科技有限公司

网站: <http://www.youwokeji.com.cn>

电话: 010-59395668 (传真)

手机: 18910685939 、 13671114914 、 13691531038

Email: [coodor@126.com](mailto:coodor@126.com)

## 附录 A

MifarePlus Level3 的数据及密钥存储结构和 Mifareone 有所区别，结构如下：

| 块相对地址    |                                        | 块地址    | AES<br>对应密钥块地址             |
|----------|----------------------------------------|--------|----------------------------|
| Sector0  |                                        |        |                            |
| Block0   | 数据块                                    | 0x0000 | A 密钥：0x4000<br>B 密钥：0x4001 |
| Block1   | 数据块                                    | 0x0001 |                            |
| Block2   | 数据块                                    | 0x0002 |                            |
| Block3   | Level1 :KeyA+KeyB, Level3: data+config | 0x0003 |                            |
| Sector1  |                                        |        |                            |
| Block0   | 数据块                                    | 0x0004 | A 密钥：0x4002<br>B 密钥：0x4003 |
| Block1   | 数据块                                    | 0x0005 |                            |
| Block2   | 数据块                                    | 0x0006 |                            |
| Block3   | Level1 :KeyA+KeyB, Level3: data+config | 0x0007 |                            |
| ....     |                                        |        |                            |
| Sector31 |                                        |        |                            |
| Block0   | 数据块                                    | 0x007C | A 密钥：0x403E<br>B 密钥：0x403F |
| Block1   | 数据块                                    | 0x007D |                            |
| Block2   | 数据块                                    | 0x007E |                            |
| Block3   | Level1 :KeyA+KeyB, Level3: data+config | 0x007F |                            |
| Sector32 |                                        |        |                            |
| Block0   | 数据块                                    | 0x80   | A 密钥：0x4040<br>B 密钥：0x4041 |
| Block1   | 数据块                                    | 0x81   |                            |
| ...      | 数据块                                    | ...    |                            |
| Block15  | Level1 :KeyA+KeyB, Level3: data+config | 0x8F   |                            |
| ...      |                                        |        |                            |
| Sector39 |                                        |        |                            |
| Block0   | 数据块                                    | 0xF0   | A 密钥：0x404E<br>B 密钥：0x404F |
| Block1   | 数据块                                    | 0xF1   |                            |
| ...      | 数据块                                    | ...    |                            |
| Block15  | Level1 :KeyA+KeyB, Level3: data+config | 0xFF   |                            |
| 配置块      |                                        |        |                            |
|          | MFP Configuration Block                | 0xB000 |                            |
|          | Installation Identifier                | 0xB001 |                            |

|       |                             |               |  |
|-------|-----------------------------|---------------|--|
|       | ATS Information             | 0xB002        |  |
|       | Field Configuration Block   | 0xB003        |  |
| Key 块 |                             |               |  |
|       | AES Sector Keys             | 0x4000~0x403F |  |
|       | AES Sector Keys             | 0x4040~0x404F |  |
|       | Originality Key             | 0x8000        |  |
|       | Card Master Key             | 0x9000        |  |
|       | Card Configuration Key      | 0x9001        |  |
|       | Level2 switch Key           | 0x9002        |  |
|       | Level3 switch Key           | 0x9003        |  |
|       | SL1 Card Authentication Key | 0x9004        |  |
|       | Select VC Key               | 0xA000        |  |
|       | Proximity Check Key         | 0xA001        |  |
|       | VC Polling ENC Key          | 0xA080        |  |
|       | VC Polling MAC Key          | 0xA081        |  |

**注意：**

- 1、蓝色和黄色部分是关联部分。即数据区和密钥区对应部分(仅仅是在 Level 2/3 才对应，因只有级别 2/3 才使用到 AES 密钥认证)。
- 2、在安全级别 Level 1，是和 Mifare classic 兼容的，每个扇区最后一块为密钥和配置块。
- 3、AES 密钥分为 A/B 密钥是人为划分，是为了同 Mifare classic 概念相同。在 PLUS 内部一个扇区是对应地址连续的 AES 密钥块。
- 4、主要掌握如下 key:

**AES Sector Keys:**

在 Level2/3 中对数据的授权采用 AES Key 授权。该密钥可以在 Level0 写入，或者通过 AES Sector Keys 对卡片授权而修改 AES Key。

**CardMasterKey:**

通过该 Key 的授权，可以改变 **Card Configuration Key** 和 **Level2/3 switch Key**

**Card Configuration Key:**

通过该 key 的授权，可以改变 MFP Configuration Block 配置块内容。

**Level2 switch Key:**

通过该 key 的授权，可以从 Level1 切换 Level2。

**Level3 switch Key:**

通过对该 key 的授权，可以从 Level2 切换 Level3，或从 Level1 切换到 Level3。

- 5、在 Level0，除了出厂写入的用户不能修改的密钥外，都可以以明文方式写入，一般在 Level0 做初始化操作。注意，必须在该安全级别写入 0x9000~0x9003 块。
- 6、Level3 级别支持明文、AES 加密、加密且带 MAC 方式读写方式。本读卡器采用的是最保密的方式读写 MifarePlus 块：*加密且带 MAC 方式*。