Advanced Card Systems Ltd.
Card & Reader Technologies

ACR83 PINeasy

参考手册 V1.06

info@acs.com.hk
www.acs.com.hk

目录

图目录

1.0. 简介

ACR83 是一款性价比非常高的联机按键式读写器，它可以用作计算机（例如：个人电脑）与智能卡间的通信接口。不同类型的智能卡采用不同的命令和通信协议，而 ACR83 PINeasy 则为各种卡片建立了一个从计算机到智能卡的统一接口。

ACR83 通过 USB 接口与电脑进行连接，同时采用 CCID 接口与 USB 端进行通信。CCID（USB Chip/Smart Card Interface Devices-USB 芯片智能卡接口设备）规范定义了 USB 芯片智能卡接口设备的通讯协议和命令。

另外，ACR83 支持 CCID 安全 PIN 码输入（SPE）功能，为 PIN 码的输入提供了安全的用户界面，消除了 PIN 码被第三方获取的风险。ACR83 还可以在读卡器内进行 PIN 校验和修改，是一款特殊的智能卡读写器。

1.1. 参考文件

下列资料可以在 WWW.USB.ORG 下载。

- 通用串行总线规范 2.0（即 USB 规范），2000 年 4 月 27 日
- 通用串行总线通用类规范 1.0，1997 年 12 月 16 日
- 通用串行总线设备类：集成电路（S）卡接口设备的智能卡 CCID 规范，1.1 版，2005 年 4 月 22 日

下列文件可以在 WWW.ANSI.ORG 订购。

- ISO/IEC 7816-1：识别卡 — 带触点的集成电路卡 - 第一部分：物理特性
- ISO/IEC 7816-2：识别卡 — 带触点的集成电路卡 - 第二部分：触点的尺寸和位置
- ISO/IEC 7816-3：识别卡 — 带触点的集成电路卡 - 第三部分：电信号和传输协议

1.2. 符号和缩写

符号	缩写
ATR	复位应答 Answer-to-Reset
EMV	Europay、MasterCard、VISA 三大国际银行卡组织共同制定的芯片卡规范 Europay MasterCard VISA
PPS	协议与参数选择 Protocol and Parameters Selection
SPE	安全 PIN 码输入 Secure PIN Entry
USB	通用串行总线 Universal Serial Bus

2.0. 特性

- 14 键键盘
- 2 行 x 16 个字符的点阵式 LCD，每个字符 5x8 点
- 支持具有以下特性的 ISO 7816 CPU 卡：
  - A 类、B 类和 C 类（5 V、3 V、1.8 V）
  - T=0 和/或 T=1 协议
- 支持安全 PIN 码输入（SPE）
- 通过 EMV Level 1 认证
- USB 全速接口（12 Mbps）
- 符合下列标准：
  - PC/SC
  - WHQL
  - CCID
  - CE/FCC
  - RoHS

3.0. 支持的卡片类型

ACR83 支持符合 T=0 或 T=1 协议的 MCU 卡。若卡片产生的 ATR 指定了专用的操作模式（TA2 存在；TA2 中的 b5 位必须为 0），但 ACR83 PINeasy 不支持该特定模式，则 ACR83 会将卡片复位，使其置为协商模式。如果卡片不能被置为协商模式，ACR83 会拒绝读写该卡。

若卡片产生的 ATR 指定了协商模式（TA2 不存在时）和通信参数，而不是默认参数，则 ACR83 将执行 PPS 并尝试使用卡片在 ATR 中指定的通信参数。如果卡片不接受 PPS，读卡器会使用默认参数（F=372，D=1）。

对于上述参数的含义，请参照 ISO7816-3 标准。

4.0. 智能卡接口

ACR83 PINeasy 智能卡读写器带有 14 键键盘和可以显示 2 行 x 16 个字符的点阵式 LCD 显示屏。

4.1. 智能卡电源 VCC（C1）

插入的智能卡的电流消耗不得大于 100 mA。

4.2. 编程电压 VPP（C6）

根据 ISO7816-3 的规定，由智能卡上的触点 C6（VPP）为智能卡提供编程电压。但由于市面上的智能卡大多数基于 EEPROM，并且不需要为其提供外部编程电压，ACR83（CCID）的触点 C6（VPP）已被实现为普通的控制信号。此触点的电气规格与 RST 信号（触点 C2）的规格相同

4.3. 卡片类型选择

激活插入的卡片之前，处于控制地位的电脑总是需要向 ACR83 发送适当的命令来选择卡片类型。

对于基于 MCU 的卡片来说，读写器允许从 T=0 或 T=1 中选择首选的协议。但是只有当插入读写器的卡片对这两种协议类型都支持时，读写器才可以通过协议与参数选择（PPS）接受并执行这样的选择。当基于 MCU 的卡仅支持一种协议类型（T=0 或 T=1）时，读写器会自动采用该协议类型，而不管应用程序选择哪一种。

4.4. 微控制器卡接口

基于微控制器的智能卡只使用触点 C1（VCC）、C2（RST)、C3（CLK）、C5（GND）和 C7（I/O）。时钟信号（C3）的频率为 4 MHz。

4.5. 卡片插拔保护

ACR83（CCID）提供了一种机制来保护在上电状态下被突然拔出的卡片。当卡片被移出时，卡片的电源以及 ACR83（CCID）与卡之间的信号线路会立即被取消激活。但是作为惯例，只应在断电后才应从读卡器移出卡片，这样可以避免电气损伤。

注：*ACR83（CCID）本身不会接通向卡片的供电。此操作由处于控制地位的电脑向读卡器发送适当的命令来明确地完成。*

5.0. 电源

ACR83（CCID）需要 5 V，100 mA 的直流稳压电源，它通过与各类型读卡器一起提供的电缆由计算机供电。

6.0. USB 接口

ACR83（CCID）通过符合 USB 标准的 USB 接口与计算机建立连接。

6.1. 通信参数

ACR83（CCID）通过符合 USB 1.1 规范的 USB 端口与计算机建立连接，支持 USB 全速模式，速率为 12 Mbps。

引脚	信号	功能
1	VBUS	为读写器提供+5 V 的电源
2	D-	ACR83 和 PC 间以差分信号传输数据
3	D+	ACR83 和 PC 间以差分信号传输数据
4	GND	参考电压等级

表1 ：USB 接口配线

注：*ACR83 PINeasy 是一款* ***PC/SC 设备****，要通过 USB 接口正常工作，必须先安装* ***ACS PC/SC 驱动。****详情请参考设备驱动程序安装指南。*

6.2. 端点

ACR83（CCID）通过如下端点与主计算机进行通信：

Control Endpoint	用于设置和控制
Bulk OUT	用于从主计算机发送至 ACR83（CCID）的命令（数据包大小为 64 字节）
Bulk IN	用于从 ACR83（CCID）发送至主计算机的响应（数据包大小为 64 字节）
Interrupt IN	用于从 ACR83（CCID）发送至主计算机的卡片状态报文（数据包大小为 8 字节）

7.0. 通信协议

ACR83（CCID）通过 USB 与主机端（host）建立连接。现在的行业内规范 -- CCID 标准，已经为 USB 芯片-智能卡接口设备定义了与此相关的协议。CCID 涵盖了操作智能卡和 PIN 所需的全部协议。

ACR83（CCID）的 USB 端点配置和使用应当符合 CCID 标准第 3 部分的规定。概述总结如下：

- **控制命令**通过控制通道（缺省通道）发送。其中包括类特定请求和 USB 标准请求。由缺省通道发送的命令会通过缺省通道向主机反馈信息。
- **CCID 时间**通过中断通道发送。
- **CCID 命令**经由 BULK-OUT 端点发出。发送至 ACR83（CCID）的每个命令都有一个相关的最终响应，一些命令也可以有中间响应。
- **CCID 响应**经由 BULK-IN 端点发出。所有发送至 ACR83（CCID）的命令都必须同步发送。（即：对于 ACR83（CCID）来说，*bMaxCCIDBusySlots* 等同于 1）。

ACR83（CCID）支持的 CCID 功能由其类别描述符定义：

偏移	数据域	大小	值	说明
0	*bLength*	1	36h	描述符的字节数
1	*bDescriptorType*	1	21h	CCID 功能描述符的类别
2	*bcdCCID*	2	0100h	CCID 以二进制编码的十进制指定的版本号码
4	*bMaxSlotIndex*	1	00h	ACR83（CCID）有 1 个卡槽
5	*bVoltageSupport*	1	07h	ACR83（CCID）可支持 1.8V、3.0V 和 5.0V 的槽位电压。
6	*dwProtocols*	4	00000003h	ACR83（CCID）支持 T=0 和 T=1 协议
10	*dwDefaultClock*	4	00000FA0h	默认 ICC 时钟频率为 4 MHz
14	*dwMaximumClock*	4	00000FA0h	ICC 支持的最大时钟频率为 4 MHz
18	*bNumClockSupported*	1	00h	不支持手动设置时钟频率
19	*dwDataRate*	4	00002A00h	默认 ICC I/O 波特率为 10752 bps
23	*dwMaxDataRate*	4	0001F808h	ICC I/O 支持的最大波特率为 250000 bps
27	*bNumDataRatesSupported*	1	00h	不支持手动设置波特率
28	*dwMaxIFSD*	4	00000Feh	ACR83（CCID）T1 支持的最大 IFSD 为 254。
32	*dwSynchProtocols*	4	00000000h	ACR83（CCID）不支持同步卡。
36	*dwMechanical*	4	00000000h	ACR83（CCID）不支持特殊机制特性

偏移	数据域	大小	值	说明
40	*dwFeatures*	4	00010030h	ACR83（CCID）支持以下特性： • 自动根据参数改变 ICC 时钟频率 • 自动根据频率和 FI、DI 参数改变波特率 • 与 ACR83（CCID）进行 TPDU 级别交换
44	*dwMaxCCIDMessageLength*	4	0000010Fh	AACR83（CCID）可接受的最大信息长度为 271 字节。
48	*bClassGetResponse*	1	00h	对 TPDU 级别的交换没有影响
49	*bClassEnvelope*	1	00h	对 TPDU 级别的交换没有影响
50	*wLCDLayout*	2	0000h	无 LCD
52	*bPINSupport*	1	00h	无 PIN 校验
53	*bMaxCCIDBusySlots*	1	01h	可以同时忙的槽位数为 1

8.0. PC/SC SCardControl 应用程序编程接口

8.1. 具体的 ScardControl 函数

```
LONG SCardControl(
SCARDHANDLE hCard,
DWORD dwControlCode,
LPCVOID lpInBuffer,
DWORD nInBufferSize,
LPVOID lpOutBuffer,
DWORD nOutBufferSize,
LPDWORD lpBytesReturned
);
#define IOCTL_SMARTCARD_GET_FIRMWARE_VERSION SCARD_CTL_CODE(2078)
#define IOCTL_SMARTCARD_DISPLAY_LCD_MESSAGE SCARD_CTL_CODE(2079)
#define IOCTL_SMARTCARD_READ_KEY SCARD_CTL_CODE(2080
// PC/SC 2.0 Part 10
#define CM_IOCTL_GET_FEATURE_REQUEST SCARD_CTL_CODE(3400)
```

注：*数据以小端格式存储，最低有效字节（LSB）在前面。另外，必须在源代码中对 SCardControl 命令予以声明。*

8.2. 智能卡设备 IOCTL

本节介绍定义的智能卡设备输入/输出控制（IOCTL）。

8.2.1. CM_IOCTL_GET_FEATURE_REQUEST

CM_IOCTL_GET_FEATURE_REQUEST 会返回读写器支持的功能列表。

hCard	由函数 *SCardConnect* 返回的引用值
dwControlCode	CM_IOCTL_GET_FEATURE_REQUEST
lpInBuffer	NULL
nInBufferSize	必须是 *lpInBuffer* 的 **sizeof**(ULONG)
lpOutBuffer	根据 PC/SC 2.0 规范第 10 部分，定义了以下功能：

```
#define FEATURE_VERIFY_PIN_START 0x01
#define FEATURE_VERIFY_PIN_FINISH 0x02
#define FEATURE_MODIFY_PIN_START 0x03
#define FEATURE_MODIFY_PIN_FINISH 0x04
#define FEATURE_GET_KEY_PRESSED 0x05
#define FEATURE_VERIFY_PIN_DIRECT 0x06
#define FEATURE_MODIFY_PIN_DIRECT 0x07
#define FEATURE_MCT_READERDIRECT 0x08
#define FEATURE_MCT_UNIVERSAL 0x09
#define FEATURE_IFD_PIN_PROP 0x0A
#define FEATURE_ABORT 0x0B
```

ACR83 支持以下功能：

```
#define FEATURE_VERIFY_PIN_DIRECT 0x06
#define FEATURE_MODIFY_PIN_DIRECT 0x07
#define FEATURE_IFD_PIN_PROP 0x0A
```

如果使用的ACR83支持PC/SC 2.0第10部分规定，您将获得以下数据：

06 04 XX XX XX XX 07 04 XX XX XX XX 0A 04 XX XX XX XXh

其中，XX XX XX XXh 为功能的控制代码。

nOutBufferSize	*lpOutBuffer* 的 **sizeof**(ULONG)
lpBytesReturned	指向一个 DWORD 变量的指针，该变量用于接收存储进缓冲区的数据的大小（字节数），而该缓冲区由 *lpOutBuffer* 指定。

8.2.2. FEATURE_VERIFY_PIN_DIRECT

hCard 由函数 *SCardConnect* 返回的引用值

dwControlCode CM_IOCTL_GET_FEATURE_REQUEST

lpInBuffer

偏移	数据域	大小	值	说明
0	*bTimeOut*	1	-	秒数。若值等于 00h，则使用默认值
1	*bTimeOut2*	1	00h	不支持。第一次按键后的秒数
2	*bmFormatString*	1	-	PIN 码格式选项的几个参数，更多信息请参阅**附录 A**。
3	*bmPINBlockString*	1	-	定义 APDU 命令中 PIN 数据块的长度（字节数）。更多信息请参阅**附录 B**。
4	*bmPINLengthFormat*	1	-	允许在 APDU 命令中加入 PIN 码长度。更多信息请参阅**附录 C**。
5	*wPINMaxExtraDigit*	2	XXYYh	XXh：PIN 码最大长度（位数） YYh：PIN 码最小长度（位数）
7	*bEntryValidationCondition*	1	-	该值是一个位 OR 运算。 01h = 达到最大长度 02h = 按下确认键 04h = 出现超时
8	*bNumberMessage*	1	FFh	PIN 验证显示的消息数量
9	*wLangId*	2	0409h	消息的语言
11	*bMsgIndex*	1	00h	消息索引（应为 00h）
12	*bTeoPrologue*	3	000000h	要使用的 T=1 信息块（I-block）起始域（填写 00h）
15	*ulDataLength*	4	-	待发送至 ICC 的数据的长度。
19	*abData*	-	-	待发送至 ICC 的数据

nInBufferSize 19 + uLDataLength

lpOutBuffer

偏移	数据域	大小	值	说明
0	*abStatus*	2	-	6400h：SPE 操作超时 6401h：通过“取消”按钮取消了 SPE 操作。 6402h：两次输入的“新 PIN”不一致，PIN 修改操作失败 6403h：用户输入的 PIN 太短或太长，不符合最短/最长 PIN 码长度限制 **注：**ACR83 在 PIN 输入过程中检查 PIN 的长度，将不再返回此状态。 6B80h：传递的结构的参数无效 SW1SW2：来自卡片的结果

nOutBufferSize 2

lpBytesReturned 指向一个 DWORD 变量的指针，该变量用于接收存储进缓冲区的数据的长度（字节数），而该缓冲区由 *lpOutBuffer* 指定。

8.2.3. FEATURE_MODIFY_PIN_DIRECT

hCard 由函数 *SCardConnect* 返回的引用值

dwControlCode CM_IOCTL_GET_FEATURE_REQUEST

lpInBuffer

偏移	数据域	大小	值	说明
0	*bTimeOut*	1	-	秒数。若值等于 00h，则使用默认值
1	*bTimeOut2*	1	00h	不支持。第一次按键后的秒数
2	*bmFormatString*	1	-	PIN 码格式选项的几个参数，更多信息请参阅**附录 A**。
3	*bmPINBlockString*	1	-	定义 APDU 命令中 PIN 数据块的长度（字节数）。更多信息请参阅**附录 B**。
4	*bmPINLengthFormat*	1	-	允许在 APDU 命令中加入 PIN 码长度。更多信息请参阅**附录 C**。
5	*bInsertionOffsetOld*	1	-	当前 PIN 码的插入位置偏移（字节）
6	*bInsertionOffsetNew*	1	-	新 PIN 码的插入位置偏移（字节）
7	*wPINMaxExtraDigit*	2	XXYYh	XXh：PIN 码最大长度（位数） YYh：PIN 码最小长度（位数）
9	*bConfirmPIN*	1	00h, 01h, 02h, 03h	表示是否要在新的 PIN 码生效前进行确认（意思是用户要输入两次新 PIN 后，该 PIN 才会生效） 表示是否要在相同的 APDU 域输入并设置当前 PIN 码 b0：(0/1) 如果为 0 = 无需进行确认 如果为 1 = 需要进行确认 b1：(0/1) 如果为 0 = 无需输入当前 PIN（在此情况下，*bInsertinoOffsetOld* 的值不应被考虑在内） 如果为 1 = 需要输入当前 PIN b2 – b7：RFU
10	*bEntryValidationCondition*	1	-	该值是一个位 OR 运算。 01h = 达到最大长度 02h = 按下确认键 04h = 出现超时
11	*bNumberMessage*	1	FFh	PIN 验证显示的消息数量
12	*wLangId*	2	0409h	消息的语言

偏移	数据域	大小	值	说明
14	*bMsgIndex1*	1	00h	第一条提示信息的索引
15	*bMsgIndex2*	1	01h	第二条提示信息的索引
16	*bMsgIndex3*	1	02h	第三条提示信息的索引
17	*bTeoPrologue*	3	000000h	要使用的 T=1 信息块（I-block）起始域 （填写 00h）。
20	*ulDataLength*	4	-	待发送至 ICC 的数据的长度。
24	*abData*		-	待发送至 ICC 的数据

nInBufferSize 24 + ulDataLength

lpOutBuffer

偏移	数据域	大小	值	说明
0	*abStatus*	2	-	6400h：SPE 操作超时 6401h：通过“取消”按钮取消了 SPE 操作。 6402h：两次输入的“新 PIN”不一致，PIN 修改操作失败 6403h：用户输入的 PIN 太短或太长，不符合最短/最长 PIN 码长度限制 **注：**ACR83 在 PIN 输入过程中检查 PIN 的长度，将不再返回此状态。 6B80h：传递的结构参数无效 SW1SW2：来自卡片的结果

nOutBufferSize 2

lpBytesReturned 指向一个 DWORD 变量的指针，该变量用于接收存储进缓冲区的数据的大小（字节数），而该缓冲区由 *lpOutBuffer* 指定。

8.2.4. FEATURE_IFD_PIN_PROP

hCard 由函数 *SCardConnect* 返回的引用值。

dwControlCode 由 CM_IOCTL_GET_FEATURE_REQUEST 返回。

lpInBuffer NULL

LpOutBuffer

偏移	数据域	大小	值	说明
0	*wLcdLayout*	2	0210h	显示特性：2 行，每行 16 个字符
2	*bEntryValidationCondition*	1	07h	支持的超时时间到、达到 PIN 码最大长度以及按下确认按钮这三种条件
3	*bTimeOut2*	1	00h	0 = IFD 不能识别 *bTimeOut* 和 *bTimeOut2* 1 = IFD 可以识别 *bTimeOut* 和 *bTimeOut2*

nOutBufferSize 4

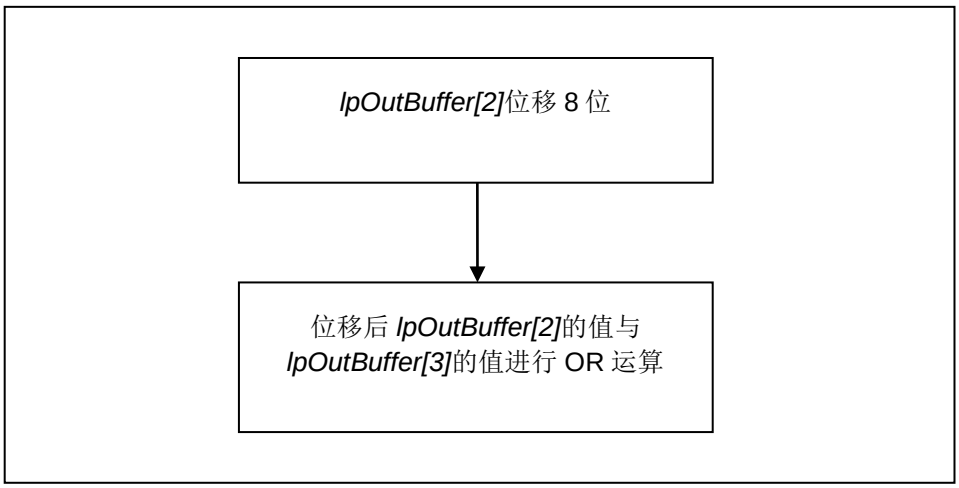
lpBytesReturned 指向一个 DWORD 变量的指针，该变量用于接收存储进缓冲区的数据的大小（字节数），而该缓冲区由 *lpOutBuffer* 指定。

8.2.5. IOCTL_SMARTCARD_GET_FIRMWARE_VERSION

IOCTL_SMARTCARD_GET_FIRMWARE_VERSION 用于启用 *Get Firmware Version* 命令。

8.2.5.1. 固件版本号

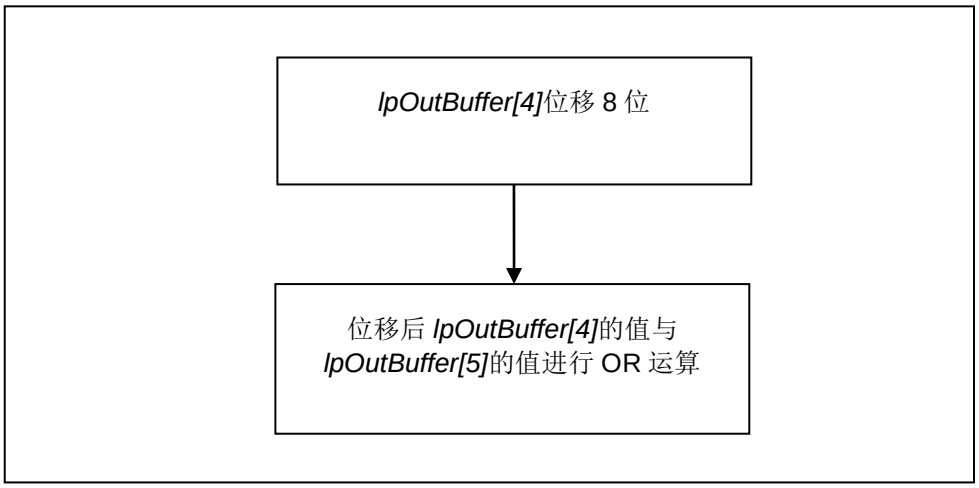
为了获得设备的固件版本号，要将接收到的缓冲数据的第三个数据元位移 8 位，之后再将位移后的结果与缓冲数据的第四个数据元进行 OR 运算。



例如：Firmware_Version = (Common.RecvBuff[2] << 8) | Common.RecvBuff[3]);

8.2.5.2. LCD

为了获取设备 LCD 的信息，要将接收到的缓冲数据的第五个数据元位移 8 位，之后再将位移后的结果与缓冲数据的第六个数据元进行 OR 运算。



输入数据：

hCard	由函数 *SCardConnect* 返回的引用值
dwControlCode	IOCTL_SMARTCARD_GET_FIRMWARE_VERSION

输出数据：

lpOutBuffer 命令输出的值

nOutBufferSize *lpOutBuffer* 的 **sizeof**(ULONG)

lpBytesReturned 指向一个 DWORD 变量的指针，该变量用于接收存储进缓冲区的数据的大小（字节数），而该缓冲区由 *lpOutBuffer* 指定。

偏移	数据域	大小	值	说明
0	*abStatus*	2	0000h	成功
2	*wACR83Firmware*	2		-
4	*LCD*	2		-

8.2.6. IOCTL_SMARTCARD_DISPLAY_LCD_MESSAGE

IOCTL_SMARTCARD_DISPLAY_LCD_MESSAGE 用于启用 display LCD message 命令。

hCard 由函数 *SCardConnect* 返回的引用值

dwControlCode IOCTL_SMARTCARD_DISPLAY_LCD_MESSAGE

lpInBuffer 设置 Display LCD message 选项的值

nInBufferSize *lpInBuffer* 的 **sizeof**(ULONG)

偏移	数据域	大小	值	说明
0	*abLCDmessage*	0-32	-	LCD 消息（最多 32 个字符）

输出数据：

lpOutBuffer 命令输出的值

nOutBufferSize *lpOutBuffer* 的 **sizeof**(ULONG)

lpBytesReturned 指向一个 DWORD 变量的指针，该变量用于接收存储进缓冲区的数据的大小（字节数），而该缓冲区由 *lpOutBuffer* 指定。

偏移	数据域	大小	值	说明
0	*abStatus*	2	0000h 0001h	成功 BAD_PARAMETER

8.2.7. IOCTL_SMARTCARD_READ_KEY

IOCTL_SMARTCARD_READ_KEY 用于启用 *Read Key* 命令。

输入数据：

hCard 由函数 *SCardConnect* 返回的引用值

dwControlCode IOCTL_SMARTCARD_READ_KEY

lpInBuffer 设置 Display LCD message 选项的值

nInBufferSize *lpInBuffer* 的 **sizeof**(ULONG)

偏移	数据域	大小	值	说明
0	*bTimeOut*	1	-	秒数。若值等于 00h，则使用默认值。
1	*wPINMaxExtraDigit*	2	XXYYh	XXh：PIN 码最大长度（位数） YYh：PIN 码最小长度（位数）
3	*bKeyReturnCondition*	1	-	该值是一个位 OR 运算。 01h：达到最大长度 02h：按下了按键[E] 04h：发生超时 08h：按下了按键[C]
4	*bEchoLCDStartPosition*	1	-	开始位置（0 – 31）
5	*bEchoLCDMode*	1	-	00h：通过按键输入的数据在 LCD 上显示为 ASCII 格式 01h：通过按键输入的数据在 LCD 上显示为星号“*”。

输出数据：

lpOutBuffer 命令输出的值

nOutBufferSize *lpOutBuffer* 的 **sizeof**(ULONG)

lpBytesReturned 指向一个 *DWORD* 变量的指针，该变量用于接收存储进缓冲区的数据的大小（字节数），而该缓冲区由 *lpOutBuffer* 指定。

偏移	数据域	大小	值	说明
0	*abStatus*	2	0000h 0001h	成功 BAD_PARAMETER
2	*bKeyReturnCondition*	1	31h 32h 33h 34h	达到最大长度 按下了按键[E] 发生超时 按下了按键[C]
3	*abNumericInputKeys*	0-32	-	-

9.0. PIN 码校验和修改的流程（PC/SC 第 10 部分）

ACR83 读写器支持 **PC/SC 2.0 第 10 部分规定的内容。**PIN 校验和修改的流程如下图所示：

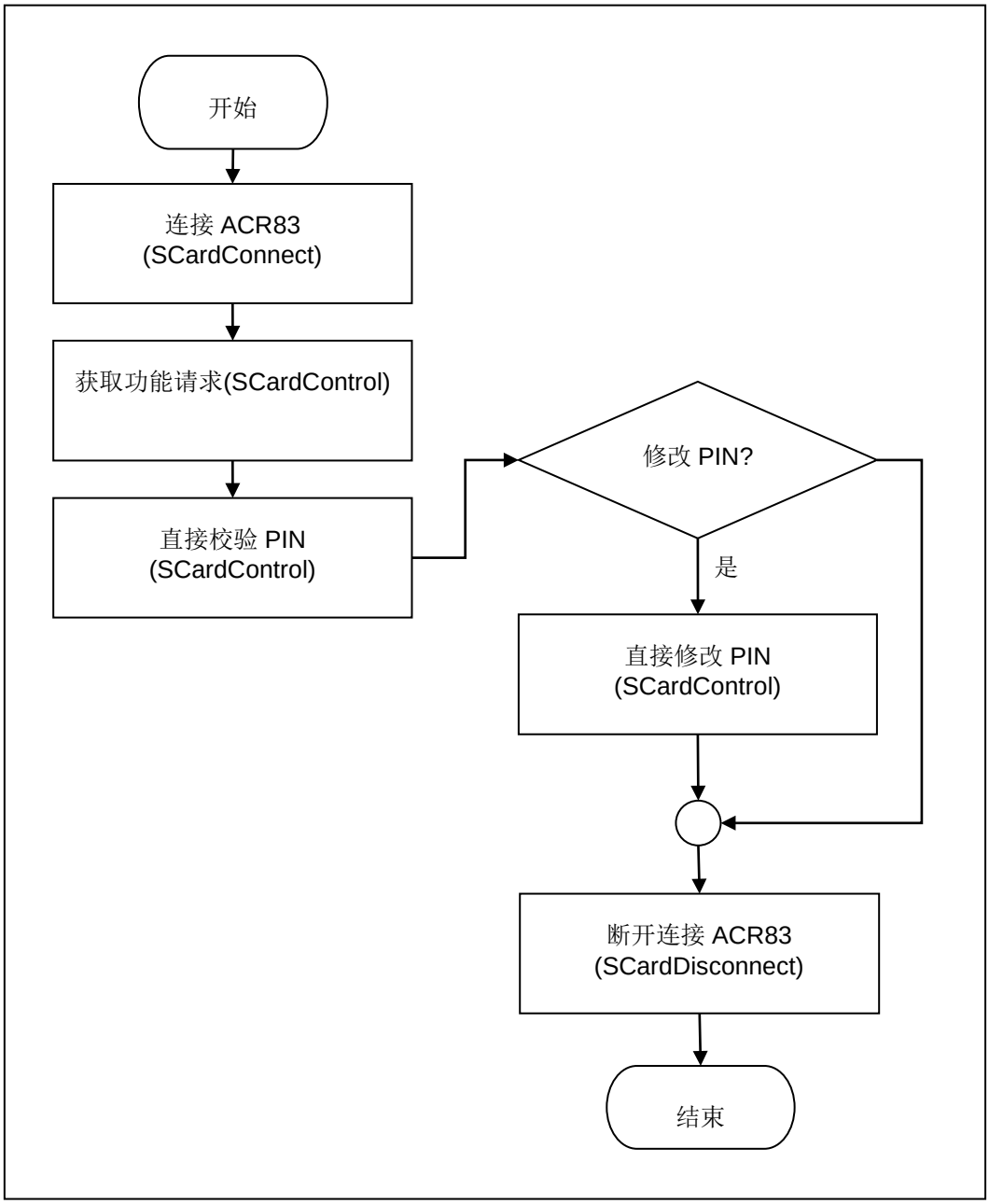


图1 ： PIN 校验和修改操作流程图

要使用 PIN 码校验和修改功能，*SCardControl* API 在调用时必须带有“获取功能请求”控制码，此 API 会返回读写器支持的功能的列表。

ACR83 仅支持直接校验 PIN、直接修改 PIN 以及 IFD PIN 属性三种功能。要使用这些功能，您可以从此列表中获得相应的控制码。如需了解更多信息，请参考 PC/SC 2.0 规范第 10 部分。

10.0.CCID SPE 数据结构

ACR83 的 SPE 功能基于完全符合 CCOD SPE 的有关 CCID SPE 规定。

如果应用程序采用 CCID SPE 方式，则必须使用 CCID 的 *PC_to_RDR_Secure* 命令来发送 APDU。

根据 CCID 规范，SPE 有两种模式：

1. PIN 校验
2. PIN 修改

这两种模式的设置均基于 CCID 规定的 *abPINOperationDataStructure*（请参考 CCID 规范 6.1.11.1 部分）。

bPINOperation：

00h：PIN 校验

01h：PIN 修改

ACR83 不支持其它值。

11.0.PIN 校验的数据结构

对于 PIN 校验，我们需要先了解 PIN 校验的数据结构。

bTimeOut：按键输入操作的秒数（00h：默认值=60 秒）

abPINApdu = CLA INS P1 P2 Lc XX XX XX XX …

例如：*abPINApdu* = 00 20 00 01 08 FF FF FF FF FF FF FF FFh

bmFormatString (Bit 7)：

0h：系统的单位为比特（bit）

1h：系统的单位为字节（byte）

Bit 6~3 (*SpePinPos)*：格式化后 PIN 在 APDU 命令中的位置

Bit2 (*SpeLeftRight*)：0=左，1=右

Bit1~0 (*SpePINTyp*)：

00h：二进制，例如：01 02 03 04 05 06

01h：BCD，例如：12 34 56

10h：ASCII，例如：31 32 33 34 35 36

bmPINBlockString：

Bit7~4 (*SpePINSize*)：

例如：2 表示 2*2 -1 = 4 - 1 允许最多 3 位数字的 PIN

若 *SpePINSize* = 0，表示不对 PIN 进行管理。

Bit3~0 (*SpePINLen*)：经过对齐和格式化后 PIN 数据块的大小，单位为字节

bmPINLengthFormat：

Bit3~0 (*SpePINLenPos*)：表示 PIN 码长度在 APDU 命令中的位置

若 *SpePINLenPos* =0，表示不对 PIN 进行管理。

Bit4：0：表示 *SpePINLenPos* 是以 bit 还是以 byte 为单位

wPINMaxExtraDigit：

XX：*(SpePinMin)* PIN 码最小长度

YY：*(SpePinMax)* PIN 码最大长度

bNumberMessage：

00h：不在 LCD 上显示消息

01h：在 LCD 上显示一条消息： “输入 PIN 码：”

FFh：默认值等于 01h

bMsgIndex：

00h：LCD 显示“输入 PIN 码：”

任何其它值均可能导致错误。

如果数据结构的格式发生错误，则 ACR83 会返回“6B 80h”。

系统以 bit 为单位（*bmFormatString* bit 7=0）。APDU 的格式与系统以 byte 为单位（*bmFormatString* bit 7=1）的 APDU 格式完全不同。

11.1. 错误检查 (Bit)

校验系统以 bit 为单位。

命令头		*SpePINLen*			
APDU 命令头	APDU 长度	偏移量 *SpePINPos*			PIN
CLA INS P1 P2	*Lc*	偏移量 *SpePINLenPos*	*SpePINSize*	未使用的数据域/ 可能不存在	PIN

检查 **PIN 校验数据结构**时要检查以下几点：

- *SpePINLen* 必须等于 *Lc*
- *SpePINPos* 必须大于等于 *SpePINLenPos* + *SpePINSize*
- *SpePINLen* – *SpePINPos* 必须大于等于 *SpePinMax* (如果是 BCD 格式，需为 4 的倍数)
- *SpePinMax* 必须大于等于 *SpePinMin*
- *SpePinMax* 不能超过 16 位数字，因为 LCD 每行只能显示 16 位数字
- *SpePinMin* 必须大于等于 1

11.2. 错误检查 (Byte)

检查 **PIN 校验数据结构**时要检查以下几点：

- *Lc* 必须等于 *SpePINLen* + *SpePINPos*
- *SpePINPos* 必须大于等于 *SpePINLenPos* + *SpePINSize*
- *SpePINLen* – *SpePINPos* 必须大于等于 *SpePinMax* (如果是 BCD 格式，需为 4 的倍数)
- *SpePinMax* 必须大于等于 *SpePinMin*
- *SpePinMax* 不能超过 16 位数字，因为 LCD 每行只能显示 16 位数字
- *SpePinMin* 必须大于等于 1

校验系统以 byte 为单位。

命令头		偏移量 *SpePINPos*			*SpePINLen*
APDU 命令头	APDU 长度	偏移量 *SpePINPos*			PIN
CLA INS P1 P2	Lc	偏移量 *SpePINLenPos*	*SpePINSize*	未使用的数据域	PIN

11.3. PIN 校验例 1

系统以 bit 为单位。

命令头		*SpePINLen*			
APDU 命令头	APDU 长度	偏移量 *SpePINPos*			PIN
CLA INS P1 P2	*Lc*	偏移量 *SpePINLenPos*	*SpePINSize*	未使用的数据域/可能不存在	PIN

检查 **PIN 校验数据结构**时要检查以下几点：

- *SpePINLen* 必须等于 *Lc*
- *SpePINPos* 必须大于等于 *SpePINLenPos* + *SpePINSize*
- *SpePINLen* – *SpePINPos* 必须大于等于 *SpePinMax* (如果是 BCD 格式，需为 4 的倍数)
- *SpePinMax* 必须大于等于 *SpePinMin*
- *SpePinMax* 不能超过 16 位数字，因为 LCD 每行只能显示 16 位数字
- *SpePinMin* 必须大于等于 1

abPINApdu = 00 20 00 01 09 57 30 30 30 30 30 30 30 30h

Lc (09h)后面的前 7 个位(0101011)是控制字符。

bmFormatString=39h

SpePinPos=7 bits，因为 *bmFormatString* 的 bit 7 = 0

SpeLeftRight=左

SpePINTyp=BCD

bmPINBlockString=49h

SpePINSize=4 bits

SpePINLen=9 bytes

bmPINLengthFormat=02h

SpePINLenPos=2 bits

wPINMaxExtraDigit=010Ah

SpePinMax=0Ah

SpePinMin=01h

PIN 输入 = 1 2 3 4 5 6 7 8 0

错误检查项：

- 第 1 点：*SpePINLen* (9h)等于 *Lc* (9h)
- 第 2 点：*SpePINPos* (7 bits) >= *SpePinLenPos* (2 bits) + *SpePINSize* (4 bits)
- 第 3 点：*SpePINLen* (9h) – *SpePinPos* (7 bits)[当做 1 byte] >=[*SpePinMax* (0Ah) * 4bits (BCD)] = 5 bytes

  ：8 bytes >=5 bytes
- 第 4 点：*SpePinMax* (0Ah) > *SpePinMin* (01h)
- 第 5 点：*SpePinMax* (0Ah)< =10h
- 第 6 点：*SpePinMin* (01h) >= 01h

命令头		***SpePINLen***			
APDU 命令头	APDU 长度	偏移量 *SpePINPos* 7 bits			PIN
00 20 00 01	09	偏移量 2bits	*SpePINSize* (4 bits)	未使用的数据域/可能不存在	PIN
00 20 00 01	09	01	输入 9 位数字	偏移量 6 bits（相对于 *Lc*)	PIN
00 20 00 01	09	**01**=**0**101011	**1001** (bits)	**1** (bit)=0101011	PIN
00 20 00 01	09	**0110011** (**1001** 替代原始的 0**1**0**1**0**1**1)			PIN

如何进行 PIN 码管理？

由于是采用左对齐及 BCD 格式。

	PIN (bits)
原始	1 0011 0000 0011 0000 0011 0000 0011 0000 0011 0000 0011 0000 0011 0000 1100 0000
输入 12 34 56 78 0 (更改为 bit 格式)	0001 0010 0011 0100 0101 0110 0111 1000 0000
原始	1 0011 0000 0011 0000 0011 0000 0011 0000 0011 0000 0011 0000 0011 0000 1100 0000
输入 PIN	0 0010 0100 0110 1000 1010 1100 1111 0000 000
结果 PIN	0 0010 0100 0110 1000 1010 1100 1111 0000 0001 0000 0011 0000 0011 0000 1100 0000
结果 APDU（bit 格式）	0110 0110 0010 0100 0110 1000 1010 1100 1111 0000 0001 0000 0011 0000 0011 0000 0011 0000
结果 APDU（byte 格式）	66 24 68 ac f0 10 30 30 30

byte 格式的整个 APDU 为：

00 20 00 01 09 66 24 68 AC F0 10 30 30 30h

如果采用右对齐：

bmFormatString 变为=3Dh

00 20 00 01 09 67 30 30 30 31 23 45 67 80h

11.4. PIN 校验例 2

系统以 bit 为单位。

abPINApdu = 00 20 00 01 08 57 A5 30 30 30 30 30 30h

Lc (08h)后面的前 11 个位(01010111)是控制字符。

bmFormatString=59h

SpePinPos=11 bits，因为 *bmFormatString* 的 bit 7 = 0

SpeLeftRight=左

SpePINTyp=BCD

bmPINBlockString=48h

SpePINSize=4 bits

SpePINLen=8 bytes

bmPINLengthFormat=06h

SpePINLenPos=6 bits

wPINMaxExtraDigit=0108h

SpePinMax=08h

SpePinMin=01h

PIN 输入 = 1 2 3 4 5

命令头		***SpePINLen***			
APDU 头	APDU 长度	偏移量 *SpePINPos* 7 bits			PIN
00 20 00 01	08	偏移量 6 bits	*SpePINSize* (4bits)	未使用的数据域/可能不存在	PIN
00 20 00 01	08	**010101**11 101	输入 5 位数字	偏移量 6 bits（相对于 *Lc*)	PIN
00 20 00 01	08	**010101**	**0101** (bits)	**1** (bit)= 01010111 10**1**	PIN
00 20 00 01	08	**01010101 011** (**0101** 替代原始的 01010**111 101**)			PIN

由于是采用左对齐及 BCD 格式。

	PIN (bits)
原始	0 0101 0011 0000 0011 0000 0011 0000 0011 0000 0011 0000 0011 0000
输入 12 34 5 (更改为 bit 格式)	0001 0010 0011 0100 0101
原始	0 0101 0011 0000 0011 0000 0011 0000 0011 0000 0011 0000 0011 0000
输入 PIN	0 0010 0100 0110 1000 101
结果 PIN	0 0010 0100 0110 1000 1010 0011 0000 0011 0000 0011 0000 0011 0000
结果 APDU（bit 格式）	0101 0101 0110 0010 0100 0110 1000 1010 0011 0000 0011 0000 0011 0000 0011 0000
结果 APDU（byte 格式）	55 62 46 8A 30 30 30 30

byte 格式的整个 APDU 为：

00 20 00 01 08 55 62 46 8A 30 30 30 30h

如果采用右对齐：

bmFormatString 变为=5Dh

00 20 00 01 08 55 65 30 30 30 31 23 45h

11.5. PIN 校验例 3

系统以 byte 为单位。

命令头		偏移量 *SpePINPos*			*SpePINLen*
APDU 命令头	APDU 长度	偏移量 *SpePINPos*			PIN
CLA INS P1 P2	*Lc*	偏移量 *SpePINLenPos*	*SpePINSize*	未被使用的数字域	PIN

检查 **PIN 校验数据结构**时要检查以下几点：

- *Lc* 必须等于 *SpePINLen* + *SpePINPos*
- *SpePINPos* 必须大于等于 *SpePINLenPos* + *SpePINSize*
- *SpePINLen* – *SpePINPos* 必须大于等于 *SpePinMax* (如果是 BCD 格式，需为 4 的倍数)
- *SpePinMax* 必须大于等于 *SpePinMin*
- *SpePinMax* 不能超过 16 位数字，因为 LCD 每行只能显示 16 位数字
- *SpePinMin* 必须大于等于 1

例 1：

abPINApdu = 00 20 00 01 09 57 30 30 30 30 30 30 30 30h

Lc (09h)后面的第 1 个字节(57h)是控制字符。

bmFormatString=89h

SpePinPos=1 byte，因为 *bmFormatString* bit 7 = 1

SpeLeftRight=左

SpePINTyp=BCD

bmPINBlockString=48h

SpePINSize=4 bits

SpePINLen=8 bytes

bmPINLengthFormat=04h

SpePINLenPos=4 bits

wPINMaxExtraDigit=010ah

SpePinMax=0ah

SpePinMin=01h

PIN 输入 = 1 2 3 4 5 6 7 8 0

- 第 1 点：*SpePINLen* (9)等于 *SpePINLen* (8) + *SpePinPos* (1)
- 第 2 点：*SpePINPos* (1 Byte) >= *SpePinLenPos* (4 bits) + *SpePINSize* (4 bits)
- 第 3 点：*SpePINLen* (9) – *SpePinPos* (1 Byte) >=[*SpePinMax* (0Ah) * 4bits(BCD)] = 5 Bytes
  ：8 Bytes >=5 Bytes
- 第 4 点：*SpePinMax* (0Ah) > *SpePinMin* (01h)
- 第 5 点：*SpePinMax* (0Ah) < =10h
- 第 6 点：*SpePinMin* (01h) >= 01h

命令头		***SpePINPos***			***SpePINLen***
APDU 命令头	*Lc*	偏移量 *SpePINPos* (1 字节)			PIN
00 20 00 01	09	偏移量 (4 bits)	*SpePINSize* (4 bits)	未使用的数据域	PIN
00 20 00 01	09	57h	输入 9 位数字	不存在	PIN
00 20 00 01	09	**0101**=**0101**0111	**1001** (bits)	不存在	PIN
00 20 00 01	09	**01011001** (59h) (**1001** 替代原始的 0101**0111**)			PIN

如何进行 PIN 码管理？

由于是采用左对齐及 BCD 格式

	PIN (Byte)
原始	00 20 00 01 09 57 30 30 30 30 30 30 30
输入数据	12 34 56 78 0
结果 PIN	00 20 00 01 09 59 12 34 56 78 00 30 30 30

byte 格式的整个 APDU 为：

00 20 00 01 09 59 12 34 56 78 00 30 30 30h

如果采用右对齐：

bmFormatString 变为=8Dh

00 20 00 01 08 59 30 30 30 31 23 45 67 80h

12.0.PIN 修改的数据结构

对于 PIN 修改，我们需要先了解 **PIN 修改的数据结构**。

bTimeOut：按键输入操作的秒数（00h：默认值=60 秒）

abPINApdu = *CLA INS P1 P2* Lc XX XX XX XXh …

对于 *bConfirmPIN* bit1 =0

例如：*abPINApdu* = 00 24 00 01 08 FF FF FF FF FF FF FF FFh

(新的 PIN)

对于 *bConfirmPIN* bit1 =1

例如：*abPINApdu* = 00 24 00 01 10 20 FF FF FF FF FF FF FF 20 FF FF FF FF FF FF FFh

(旧的/当前的 PIN) (新的 PIN)

bmFormatString (Bit 7)：

0: 系统以 bit 为单位

1: 系统以 byte 为单位

Bit 6~3 (*SpePinPos*)：格式化后 PIN 在 APDU 命令中的位置

Bit2 (*SpeLeftRight*)：0=左，1=右

Bit1~0 (*SpePINTyp*)：

00h：二进制，例如：01 02 03 04 05 06

01h：BCD，例如：12 34 56

10h：ASCII，例如：31 32 33 34 35 36

bmPINBlockString：

Bit7~4 (*SpePINSize*)：

例如：2 表示 2*2 -1 = 4 - 1 允许最多 3 位数字的 PIN

若 *SpePINSize* = 0，表示不对 PIN 进行管理。

Bit3~0 (*SpePINLen*)：经过对齐和格式化后 PIN 数据块的大小，单位为字节

bmPINLengthFormat：

Bit3~0 (*SpePINLenPos*)：表示 PIN 码长度在 APDU 命令中的位置

若 *SpePINLenPos* =0，表示不对 PIN 进行管理。

Bit4：0：表示 *SpePINLenPos* 是以 bit 还是以 byte 为单位

bInsertionOffsetOld (*SpeOffsetOld*)：当前 PIN 码的插入位置偏移（字节）

bInsertionOffsetOld (*SpeOffsetOld*)：新 PIN 码的插入位置偏移（字节）

bConfirmPIN：

Bit 0：0=无需确认新的 PIN 码　1：需确认新的 PIN 码

Bit 1：0=无需输入旧的（当前）PIN 码　2：需输入旧的（当前）PIN 码

00h：*bNumberMessage* 必须等于 00h 或 01h

01h：*bNumberMessage* 必须等于 02h

02h：*bNumberMessage* 必须等于 02h

03h：*bNumberMessage* 必须等于 03h

否则会发生错误。

wPINMaxExtraDigit：

XX：(*SpePinMin*) PIN 码最小长度

YY：(*SpePinMax*) PIN 码最大长度

bMsgIndex1：

00h：LCD 显示“输入 PIN 码：”

任何其它值均可能导致错误。

bMsgIndex2：

01h：LCD 显示“输入新的 PIN 码：”

任何其它值均可能导致错误。

bMsgIndex3：

02h：LCD 显示“再输一遍新 PIN 码：”

任何其它值均可能导致错误。

12.1. 修改（Bit）bConfirmPIN Bit1=0

修改 *bConfirmPIN* Bit1 = 0

（无需输入当前/旧的 PIN 码）

系统以 bit 为单位。

APDU 命令		*SpeOffsetNew*	*SpePINLen*			
APDU 头	APDU *Lc*	可能不	偏移量 *SpePINPos*			PIN
CLA INS P1 P2	*Lc*	存在	偏移量 *SpePINLenPos*	*SpePINSize*	未使用的数据域/不存在	PIN

12.2. 修改（Bit）bConfirmPIN Bit1=0 数据结构错误检查

检查 **PIN 修改数据结构**时要注意以下几点：

- *SpePINLen* + *SpeOffsetNew* 必须等于 *Lc*
- *SpePINPos* 必须大于等于 *SpePINLenPos* + *SpePINSize*
- *SpePINLen* – *SpePINPos* 必须大于等于 *SpePinMax* (如果是 BCD 格式，需为 4 的倍数)
- *SpePinMax* 必须大于等于 *SpePinMin*
- *SpePinMax* 不能超过 16 位数字，因为 LCD 每行只能显示 16 位数字
- *SpePinMin* 必须大于等于 1

12.3. 修改（Byte）bConfirmPIN Bit1=0

修改 *bConfirmPIN* Bit1 = 0

（无需输入当前/旧的 PIN 码）

系统以 byte 为单位。

命令头		*OffsetNew*	偏移量 *SpePINPos*			*SpePINLen*
APDU 头	APDU *Lc*	OffsetNew	偏移量 *SpePINPos*			PIN
CLA INS P1 P2	*Lc*	OffsetNew	偏移量 *SpePINLenPos*	*SpePINSize*	未使用的数据域	PIN

12.4. 修改（Byte）bConfirmPIN Bit1=0 数据结构错误检查

检查 **PIN 修改数据结构**时要注意以下几点：

- *Lc* 必须等于 *SpePINLen* + *SpePINPos* + *SpeOffsetNew*
- *SpePINPos* 必须大于等于 *SpePINLenPos* + *SpePINSize*
- *SpePINLen* – *SpePINPos* 必须大于等于 *SpePinMax* (如果是 BCD 格式，需为 4 的倍数)
- *SpePinMax* 必须大于等于 *SpePinMin*
- *SpePinMax* 不能超过 16 位数字，因为 LCD 每行只能显示 16 位数字
- *SpePinMin* 必须大于等于 1

12.5. 修改（Bit）bConfirmPIN Bit1=1

bConfirmPIN Bit1 = 1

（需要输入当前/旧的 PIN 码）

系统以 bit 为单位。

APDU 命令		*SpeOffsetOld*	*SpePINLen*			
APDU 头	APDU *Lc*	可能不	偏移量 *SpePINPos*			旧 PIN 码
CLA INS P1 P2	*Lc*	存在	偏移量 *SpePINLenPos*	*SpePINSize*	未使用的数据域	旧 PIN 码

SpeOffsetNew	*SpePINLen*			
可能不	偏移量 *SpePINPos*			新 PIN 码
存在	偏移量 *SpePINLenPos*	*SpePINSize*	未使用的数据域	新 PIN 码

12.6. 修改（Bit）bConfirmPIN Bit1=1 数据结构错误检查

检查 **PIN 修改数据结构**时要注意以下几点：

- *SpePINLen* + *SpeOffsetNew* 必须等于 *Lc*
- *SpeOffsetNew* >= *SpeOffsetOld* + *SpePINLen*
- *SpePINPos* 必须大于等于 *SpePINLenPos* + *SpePINSize*
- *SpePINLen* – *SpePINPos* 必须大于等于 *SpePinMax* (如果是 BCD 格式，需为 4 的倍数)
- *SpePinMax* 必须大于等于 *SpePinMin*
- *SpePinMax* 不能超过 16 位数字，因为 LCD 每行只能显示 16 位数字
- *PinMin* 必须大于等于 1

12.7. 修改（Byte）bConfirmPIN Bit1=1

bConfirmPIN Bit1 = 1

（需要输入当前/旧的 PIN 码）

系统以 byte 为单位。

APDU 命令		*SpeOffsetOld*	偏移量 *SpePINPos*			SpePINLen
APDU 头	APDU *Lc*	可能不	偏移量 *SpePINPos*			旧 PIN 码
CLA INS P1 P2	*Lc*	存在	偏移量 *SpePINLenPos*	*SpePINSize*	未使用的数据域	旧 PIN 码

SpeOffsetNew	偏移量 *SpePINPos*			*SpePINLen*
可能不	偏移量 *SpePINPos*			新 PIN 码
存在	偏移量 *SpePINLenPos*	*SpePINSize*	未使用的数据域	新 PIN 码

12.8. 修改（Byte）bConfirmPIN Bit1=1 数据结构错误检查

检查 PIN 修改数据结构时要注意以下几点：

- *SpePINLen* + *SpeOffsetNew* + *SpePINPos* 必须等于 *Lc*
- *SpeOffsetNew* >= *SpeOffsetOld* + *SpePINPos* + *SpePINLen*
- *SpePINPos* 必须大于等于 *SpePINLenPos* + *SpePINSize*
- *SpePINLen* – *SpePINPos* 必须大于等于 *SpePinMax* (如果是 BCD 格式，需为 4 的倍数)
- *SpePinMax* 必须大于等于 *SpePinMin*
- *SpePinMax* 不能超过 16 位数字，因为 LCD 每行只能显示 16 位数字
- *SpePinMin* 必须大于等于 1

12.9. PIN 修改例 1

修改 *bConfirmPIN* Bit1 = 0

（无需输入当前/旧的 PIN 码）

系统以 bit 为单位。

APDU 命令		*SpeOffsetNew*	*SpePINLen*			
APDU 头	APDU *Lc*	可能不	偏移量 *SpePINPos*			PIN
CLA INS P1 P2	*Lc*	存在	偏移量 *SpePINLenPos*	*SpePINSize*	未使用的数据域/不存在	PIN

检查 **PIN 修改数据结构**时要注意以下几点：

- *SpePINLen* + *SpeOffsetNew* 必须等于 *Lc*
- *SpePINPos* 必须大于等于 *SpePINLenPos* + *SpePINSize*
- *SpePINLen* – *SpePINPos* 必须大于等于 *SpePinMax* (如果是 BCD 格式，需为 4 的倍数)
- *SpePinMax* 必须大于等于 *SpePinMin*
- *SpePinMax* 不能超过 16 位数字，因为 LCD 每行只能显示 16 位数字
- *SpePinMin* 必须大于等于 1

abPINApdu = 00 24 00 01 0A 20 57 30 30 30 30 30 30 30 30h

bConfirmPIN = 00h（如果 *bConfirmPIN* = 00h，*bNumberMessage* 必须等于 01h, 00h）

输入新的 PIN 码一次。

bmFormatString=39h

SpePinPos=7 bits，因为 *bmFormatString* 的 bit 7 = 0

SpeLeftRight=左

SpePINTyp=BCD

bmPINBlockString=49h

SpePINSize=4 bits

SpePINLen=9 bytes

bmPINLengthFormat=02h

SpePINLenPos=2 bits

wPINMaxExtraDigit=010Ah

SpePinMax=0Ah

SpePinMin=0Ah

bInsertionOffsetNew(*SpeOffsetNew*)=01h

SpeOffsetNew =1 byte

bNumberMessage=01h

显示“输入新的 PIN 码：”

若 *bNumberMessage*=00h

不会显示任何信息，但用户需要输入 PIN 码。

输入新的 PIN 码 = 1 2 3 4 5 6 7 8 0

- 第 1 点：*Lc* (0A)等于 *SpePINLen* (09) + *SpeOffsetNew* (01h)
- 第 2 点：*SpePINPos* (7 bits) >= *SpePinLenPos* (2 bits) + *SpePINSize* (4 bits)
- 第 3 点：*SpePINLen* (*9*) – *SpePinPos* (7 bits)[当做 1 byte] >=[*SpePinMax* (0Ah) * 4bits(BCD)] = 5 bytes

  ：8 bytes >=5 bytes
- 第 4 点：*SpePinMax* (0Ah) > *SpePinMin* (01h)
- 第 5 点：*SpePinMax* (0Ah) < =10h
- 第 6 点：*SpePinMin* (01h) >= 01h

命令头		*OffsetNew*	*SpePINLen*			
APDU 头	APDU *Lc*	*OffsetNew*	偏移量 *SpePINPos* 7 bits			PIN
00 24 00 01	09	偏移量	偏移量 (2 bits)	*SpePINSize* (4 bits)	未使用的数据域/不存在	PIN
00 24 00 01	09	1 个字节	01	输入 9 位数字	偏移量 6 bits（相对于 *Lc*)	PIN
00 24 00 01	09	20	**01**=**01**01011	**1001** (bits)	**1** (bit)=0101011	PIN
00 24 00 01	09	20	**0110011** (**1001** 替代原始的 01**01011**)			PIN

如何进行 PIN 码管理？

由于是采用左对齐及 BCD 格式。

	PIN (bits)
原始	1 0011 0000 0011 0000 0011 0000 0011 0000 0011 0000 0011 0000 0011 0000 1100 0000
输入 12 34 56 78 0 (更改为 bit 格式)	0001 0010 0011 0100 0101 0110 0111 1000 0000
原始	1 0011 0000 0011 0000 0011 0000 0011 0000 0011 0000 0011 0000 0011 0000 1100 0000
输入新的 PIN	0 0010 0100 0110 1000 1010 1100 1111 0000 000
结果 PIN	0 0010 0100 0110 1000 1010 1100 1111 0000 0001 0000 0011 0000 0011 0000 1100 0000
结果 APDU（bit 格式）	0110 0110 0010 0100 0110 1000 1010 1100 1111 0000 0001 0000 0011 0000 0011 0000 0011 0000
结果 APDU（byte 格式）	66 24 68 ac f0 10 30 30 30

byte 格式的整个 APDU 为：

00 24 00 01 0A 20 66 24 68 ac f0 10 30 30 30h

如果采用右对齐：

bmFormatString 变为=3Dh

00 24 00 01 0A 20 67 30 30 30 31 23 45 67 80h

若 *SpeOffsetNew* = 00h 并且 *abPINApdu* = 00 24 00 01 09 57 30 30 30 30 30 30 30 30h

bmFormatString 变为=39h

结果格式化 APDU = 00 24 00 01 09 66 24 68 AC F0 10 30 30 30h

12.10. PIN 修改例 2

与例 1 相同，只是 *bConfirmPIN* 改为 01h。

abPINApdu = 00 24 00 01 0A 20 57 30 30 30 30 30 30 30 30h

bConfirmPIN = 01h（若 *bConfirmPIN*=01h，*bNumberMessage* 必须等于 02h）

输入新的 PIN 码，并再输一遍进行确认。

bmFormatString=39h

SpePinPos=7 bits，因为 *bmFormatString* 的 bit 7 = 0

SpeLeftRight=左

SpePINTyp=BCD

bmPINBlockString=49h

SpePINSize=4 bits

SpePINLen=9 bytes

bmPINLengthFormat=02h

SpePINLenPos=2 bits

wPINMaxExtraDigit=010Ah

SpePinMax=0Ah

SpePinMin=01h

bInsertionOffsetNew (*SpeOffsetNew*)=01h

SpeOffsetNew =1 byte

bNumberMessage=02h

显示“输入新的 PIN 码：”，并

显示“再输一遍新的 PIN 码”

输入新的 PIN 码 = 1 2 3 4 5 6 7 8 0

- 第 1 点：*Lc* (0Ah)等于 *SpePINLen* (09) + *SpeOffsetNew* (01h)
- 第 2 点：*SpePINPos* (7 bits) >= *SpePinLenPos* (2 bits) + *SpePINSize* (4 bits)
- 第 3 点：*SpePINLen* (9) – *SpePinPos* (7 bits) [当做 1 个字节] >= [*SpePinMax* (0Ah) * 4bits(BCD)] = 5 bytes

  ：8 bytes >=5 bytes
- 第 4 点：*SpePinMax* (0Ah) > *SpePinMin* (01h)
- 第 5 点：*SpePinMax* (0Ah) < 10h
- 第 6 点：*SpePinMin* (01h) >= 01h

命令头		*OffsetNew*	*SpePINLen*			
APDU 头	APDU *Lc*	*OffsetNew*	偏移量 *SpePINPos* 7 bits			PIN
00 24 00 01	09	偏移量	偏移量 (2 bits)	*SpePINSize* (4 bits)	未使用的数据域/不存在	PIN
00 24 00 01	09	1 个字节	01	输入 9 位数字	偏移量 6 bits（相对于 *Lc*)	PIN
00 24 00 01	09	20	**01**=**01**01011	**1001**(bits)	**1** (bit)=0101011	PIN
00 24 00 01	09	20	**0110011** (**1001** 替代原始的 0**1**01011)			PIN

如何进行 PIN 码管理?

由于是采用左对齐及 BCD 格式。

	PIN (bits)
原始	1 0011 0000 0011 0000 0011 0000 0011 0000 0011 0000 0011 0000 1100 0000
输入 12 34 56 78 0 (更改为 bit 格式)	0001 0010 0011 0100 0101 0110 0111 1000 0000
原始	1 0011 0000 0011 0000 0011 0000 0011 0000 0011 0000 0011 0000 1100 0000
输入新的 PIN	0 0010 0100 0110 1000 1010 1100 1111 0000 000
结果 PIN	0 0010 0100 0110 1000 1010 1100 1111 0000 0001 0000 0011 0000 0011 0000 1100 0000
结果 APDU（bit 格式）	0110 0110 0010 0100 0110 1000 1010 1100 1111 0000 0001 0000 0011 0000 0011 0000 0011 0000
结果 APDU（byte 格式）	66 24 68 AC F0 10 30 30 30

byte 格式的整个 APDU 为：

00 24 00 01 0A 20 66 24 68 ac f0 10 30 30 30h

如果采用右对齐：

bmFormatString 变为=3Dh

00 24 00 01 0A 20 67 30 30 30 31 23 45 67 80h

若 *SpeOffsetNew* = 00h 并且 *abPINApdu* = 00 24 00 01 09 57 30 30 30 30 30 30 30h

bmFormatString 变为=39h

结果格式化 APDU = 00 24 00 01 09 66 24 68 ac f0 10 30 30 30h

12.11. PIN 修改例 3

bConfirmPIN Bit1 = 0

（无需输入当前/旧的 PIN 码）

系统以 byte 为单位。

命令头		*OffsetNew*	偏移量 *SpePINPos*			*SpePINLen*
APDU 头	APDU *Lc*	*OffsetNew*	偏移量 *SpePINPos*			PIN
CLA INS P1 P2	*Lc*	*OffsetNew*	偏移量 *SpePINLenPos*	*SpePINSize*	未使用的数据域	PIN

检查 PIN 修改数据结构时要注意以下几点：

1. *Lc* 必须等于 *SpePINLen* + *SpePINPos* + SpeOffsetNew
2. *SpePINPos* 必须大于等于 *SpePINLenPos* + *SpePINSize*
3. *SpePINLen* – *SpePINPos* 必须大于等于 *SpePinMax* (如果是 BCD 格式，需为 4 的倍数)
4. *SpePinMax* 必须大于等于 *SpePinMin*
5. *SpePinMax* 不能超过 16 位数字，因为 LCD 每行只能显示 16 位数字
6. *SpePinMin* 必须大于等于 1

abPINApdu = 00 24 00 01 09 57 30 30 30 30 30 30 30 30h
Lc (09h)后面的第 1 个字节(57h)是控制字符。

bConfirmPIN = 01h（若 *bConfirmPIN*=01h，*bNumberMessage* 必须等于 02h）
输入新的 PIN 码，并再输一遍进行确认

bmFormatString=89h
SpePinPos=1 byte，因为 *bmFormatString* bit 7 = 1
SpeLeftRight=左
SpePINTyp=BCD

bmPINBlockString=48h
SpePINSize=4 bits
SpePINLen=8 bytes

bmPINLengthFormat=04h
SpePINLenPos=4 bits

wPINMaxExtraDigit=010Ah
SpePinMax=0Ah
SpePinMin=01h

bInsertionOffsetNew (*SpeOffsetNew*)=00h
SpeOffsetNew =00 byte

bNumberMessage=02h
显示“输入新的 PIN 码：”，并
显示“再输一遍新的 PIN 码”

PIN 输入 = 1 2 3 4 5 6 7 8 0

- 第 1 点：*Lc* (*9*)等于 *SpeOffsetNew* (0) + *SpePINLen* (*8*) + *SpePinPos* (*1*)
- 第 2 点：*SpePINPos* (1 Byte) >= *SpePinLenPos* (4 bits) + *SpePINSize* (4 bits)
- 第 3 点：*SpePINLen* (9) – *SpePinPos* (1 Byte) >=[*SpePinMax* (0Ah) * 4 bits(BCD)] = 5 bytes
  ：8 bytes >=5 bytes
- 第 4 点：*SpePinMax* (0Ah) > *SpePinMin* (01h)
- 第 5 点：*SpePinMax* (0Ah) < =10h
- 第 6 点：*SpePinMin* (01h) >= 01h

命令头		*SpePINPos*			*SpePINLen*
APDU 头	*Lc*	偏移量 *SpePINPos* 1 Byte			PIN
00 24 00 01	09	偏移量 (4 bits)	*SpePINSize* (4bits)	未使用的数据域	PIN
00 24 00 01	09	57h	输入 9 位数字	不存在	PIN
00 24 00 01	09	**0101**=**0101**0111	**1001**(bits)	不存在	PIN
00 24 00 01	09	**01011001** (59h) (**1001** 替代原始 0101**0111**)		-	PIN

如何进行 PIN 码管理？

由于是采用左对齐及 BCD 格式

	PIN (Byte)
原始	00 24 00 01 09 57 30 30 30 30 30 30 30 30h
输入数据	12 34 56 78 0h
结果 PIN	00 24 00 01 09 59 12 34 56 78 00 30 30 30h

byte 格式的整个 APDU 为：

00 24 00 01 09 59 12 34 56 78 00 30 30 30h

如果采用右对齐：

bmFormatString 变为=8Dh

00 24 00 01 08 59 30 30 30 31 23 45 67 80h

12.12. PIN 修改例 4

bConfirmPIN Bit1 = 1

（需要输入当前/旧的 PIN 码）

系统以 bit 为单位。

APDU 命令		*SpeOffsetOld*	*SpePINLen*			
APDU 头	APDU *Lc*	可能不	偏移量 *SpePINPos*			旧 PIN 码
CLA INS P1 P2	*Lc*	存在	偏移量 *SpePINLenPos*	*SpePINSize*	未使用的数据域	旧 PIN 码

SpeOffsetNew	*SpePINLen*			
可能不	偏移量 *SpePINPos*			新 PIN 码
存在	偏移量 *SpePINLenPos*	*SpePINSize*	未使用的数据域	新 PIN 码

检查 PIN 修改数据结构时要注意以下几点：

- *SpePINLen* + *SpeOffsetNew* 必须等于 Lc
- *SpeOffsetNew* > = *SpeOffsetOld* + *SpePINLen*
- *SpePINPos* 必须大于等于 *SpePINLenPos* + *SpePINSize*
- *SpePINLen* – *SpePINPos* 必须大于等于 *SpePinMax* (如果是 BCD 格式，需为 4 的倍数)
- *SpePinMax* 必须大于等于 *SpePinMin*
- *SpePinMax* 不能超过 16 位数字，因为 LCD 每行只能显示 16 位数字
- *SpePinMin* 必须大于等于 1

abPINApdu = 00 24 00 01 12 08 02 30 30 30 30 30 30 30 09 03 30 30 30 30 30 30 30h

bConfirmPIN = 02h（若 *bConfirmPIN*=02h，*bNumberMessage* 必须等于 02h）

bmFormatString=41h

SpePinPos=8 bit，因为 *bmFormatString* 的 bit 7 = 0

SpeLeftRight=左

SpePINTyp=BCD

bmPINBlockString=48h

SpePINSize=4 bits

SpePINLen=8 bytes

bmPINLengthFormat=04h

SpePINLenPos=4 bits

wPINMaxExtraDigit=010Ah

SpePinMax=0Ah

SpePinMin=01h

bInsertionOffsetNew(*SpeOffsetNew*)=0Ah

SpeOffsetNew =0Ah byte

bInsertionOffsetOld (*SpeOffsetOld*)=01h

SpeOffsetOld =01h byte

PIN 输入（旧的/当前的 PIN） = 1 2 3 4 5 6

PIN 输入（新 PIN 码）= 1 2 3 4 5 6 7 8 9 0

bNumberMessage=02h

显示“输入 PIN 码：”表示输入旧的/当前的 PIN 码，并

显示“输入新的 PIN 码”

- 第 1 点：*Lc* (12h)等于 *SpeOffsetNew* (0Ah) + *SpePINLen* (8)
- 第 2 点：*SpeOffsetNew* (0Ah) >= *SpeOffsetOld* (1) + *SpePINLen* (8)
- 第 3 点：*SpePINPos* (8 bits) >= *SpePinLenPos* (4 bits) + *SpePINSize* (4 bits)
- 第 4 点：*SpePINLen* (8) – *SpePinPos* (4 bits) >=[*SpePinMax* (0Ah) * 4bits(BCD)] = 5 bytes
  ：7.5 bytes >=5 bytes
- 第 5 点：*SpePinMax* (0Ah) > *SpePinMin* (01h)
- 第 6 点：*SpePinMax* (0Ah) < =10h
- 第 7 点：*SpePinMin* (01h) > = 01h

命令头		*OffsetOld*	*SpePINLen*			
APDU 头	APDU *Lc*	*OffsetOld*	偏移量 *SpePINPos* 8 bits = 1byte			旧 PIN 码
00 24 00 01	12	偏移量	偏移量 (4 bits)	*SpePINSize* (4 bits)	未使用的数据域	旧 PIN 码
00 24 00 01	12	1 个字节	02	输入 6 位数字	-	旧 PIN 码
00 24 00 01	12	08	0000=00000010	0110 (bits)	-	旧 PIN 码
00 24 00 01	12	08	00000110 (0110 替代原始的 00000010)		-	旧 PIN 码

OffsetNew	*SpePINLen*			
OffsetNew	偏移量 *SpePINPos* 8 bits = 1byte			新 PIN 码
偏移	偏移量 (4 bits)	*SpePINSize* (4 bits)	未使用的数据域	新 PIN 码
0A 字节	03	输入 10 位数字	-	新 PIN 码
相对于 *Lc*	00=00000011	1010 (bits)	-	新 PIN 码
09	00001010 (1010 替代原始的 00000011)		-	新 PIN 码

首先，处理旧 PIN 码。

	旧 PIN 码（Byte）
原始	00 24 00 01 12 08 02 30 30 30 30 30 30 30 09 03 30 30 30 30 30 30 30
输入数据	12 34 56
结果 PIN	00 24 00 01 12 08 06 12 34 56 30 30 30 30 09 03 30 30 30 30 30 30 30

之后，处理新 PIN 码。

	新 PIN 码（Byte）
原始	00 24 00 01 12 08 06 12 34 56 30 30 30 30 09 03 30 30 30 30 30 30 30h
输入数据	12 34 56 78 90h
结果 PIN	00 24 00 01 12 08 06 12 34 56 30 30 30 30 09 0A 12 34 56 78 90 30 30h

格式化后的整个 APDU 为：

00 24 00 01 12 08 06 12 34 56 30 30 30 30 09 0A 12 34 56 78 90 30 30h

12.13. PIN 修改例 5

BConfirmPIN Bit1 = 1

（需要输入当前/旧的 PIN 码）

系统以 byte 为单位。

APDU 命令		*SpeOffsetOld*	偏移量 *SpePINPos*			*SpePINLen*
APDU 头	APDU *Lc*	可能不	偏移量 *SpePINPos*			旧 PIN 码
CLA INS P1 P2	*Lc*	存在	偏移量 *SpePINLenPos*	*SpePINSize*	未使用的数据域	旧 PIN 码

SpeOffsetNew	偏移量 *SpePINPos*			*SpePINLen*
可能不	偏移量 *SpePINPos*			新 PIN 码
存在	偏移量 *SpePINLenPos*	*SpePINSize*	未使用的数据域	新 PIN 码

检查 PIN 修改数据结构时要注意以下几点：

- *SpePINLen* + *SpeOffsetNew* + *SpePINPos* 必须等于 *Lc*
- *SpeOffsetNew* >= *SpeOffsetOld* + *SpePINPos* + *SpePINLen*
- *SpePINPos* 必须大于等于 *SpePINLenPos* + *SpePINSize*

- *SpePINLen – SpePINPos* 必须大于等于 *SpePinMax* (如果是 BCD 格式，需为 4 的倍数)
- *SpePinMax* 必须大于等于 *SpePinMin*
- *SpePinMax* 不能超过 16 位数字，因为 LCD 每行只能显示 16 位数字
- *SpePinMin* 必须大于等于 1

abPINApdu = 00 24 00 01 12 2F 0A A6 30 30 30 30 30 30 2E FB C7 30 30 30 30 30 30h

bConfirmPIN = 03h (若 *bConfirmPIN* = 03h，则 *bNumberMessage* 必须等于 03h 或 FFh)

bmFormatString=91h

SpePinPos=2 bytes，因为 *bmFormatString* bit 7 = 1

SpeLeftRight=左

SpePINTyp=BCD

bmPINBlockString=46h

SpePINSize=4 bits

SpePINLen=6 bytes

bmPINLengthFormat=11h

SpePINLenPos=1 byte

wPINMaxExtraDigit=010Ah

SpePinMax=0Ah

SpePinMin=01h

bInsertionOffsetNew (*SpeOffsetNew*)=0Ah

SpeOffsetNew =0Ah byte

bInsertionOffsetOld (*SpeOffsetOld*)=01h

SpeOffsetOld =00h byte

PIN 输入（旧的/当前的 PIN） = 1 2 3 4 5 6

PIN 输入（新 PIN 码）= 1 2 3 4 5 6 7 8 9 0

bNumberMessage=03h 或 FFh

显示“输入 PIN 码：”表示输入旧的/当前的 PIN 码，并

显示“输入新的 PIN 码”

显示“再输一遍新 PIN 码”

- 第 1 点：*Lc* (12h)等于 *SpeOffsetNew* (0Ah) + *SpePINLen* (6) + *SpePinPos* (2)
- 第 2 点：*SpeOffsetNew* (0Ah) >= *SpeOffsetOld* (1) + *SpePINLen* (6) + *SpePinPos* (2)
- 第 3 点：*SpePINPos* (2 Bytes) >= *SpePinLenPos* (1 Byte) + *SpePINSize* (4 bits)
- 第 4 点：*SpePINLen* (6) = [*SpePinMax* (0Ah) * 4bits(BCD)] = 5 bytes

  ：6 bytes >=5 bytes
- 第 5 点：*SpePinMax* (0Ah) > *SpePinMin* (01h)
- 第 6 点：*SpePinMax* (0Ah) <=10h
- 第 7 点：*SpePinMin* (01h) >= 01h

命令头		***OffsetOld***	***SpePINLen***			
APDU 头	APDU *Lc*	*OffsetOld*	偏移量 *SpePINPos* = 2 bytes			旧 PIN 码
00 24 00 01	12	偏移量	偏移量 (1 Byte)	*SpePINSize* (4 bits)	未使用的数据域	旧 PIN 码
00 24 00 01	12	1 个字节	0A	输入 6 位数字	0110	旧 PIN 码
00 24 00 01	12	2F	0A	**0110** (bits)	0110	旧 PIN 码
00 24 00 01	12	2F	0A	**6**6 取代 **A**6	0110	旧 PIN 码

OffsetNew	***SpePINLen***			
OffsetNew	偏移量 *SpePINPos* 8 bits = 1 byte			新 PIN 码
偏移	偏移量 (1 Byte)	*SpePINSize* (4 bits)	未使用的数据域	新 PIN 码
0A 字节	FB	输入 10 位数字	0111	新 PIN 码
相对于 *Lc*	FB	**1010** (bits)	0111	新 PIN 码
2E	FB	**1010** (**1010** 取代原始 **C**7)	0111	新 PIN 码

首先，处理旧 PIN 码。

	旧 PIN 码（Byte）
原始	00 24 00 01 12 2F 0A A6 30 30 30 30 30 30 2E FB C7 30 30 30 30 30 30
输入数据	12 34 56
结果 PIN	00 24 00 01 12 2F 0A 66 12 34 56 30 30 30 2E FB C7 30 30 30 30 30 30

之后处理新 PIN 码。

	新 PIN 码（Byte）
原始	00 24 00 01 12 2F 0A 66 12 34 56 30 30 30 2E FB C7 30 30 30 30 30 30
输入数据	12 34 56 78 90
结果 PIN	00 24 00 01 12 2F 0A 66 12 34 56 30 30 30 2E FB A7 12 34 56 78 90 30

格式化后的整个 APDU 为：

00 24 00 01 12 2F 0A 66 12 34 56 30 30 30 2E FB A7 12 34 56 78 90 30h

附录A. bmFormatStrings 说明

位号	说明
Bit 7	系统单位的类型指示符 若为 0h：系统的单位是比特（Bit） 若为 1h：系统的单位是字节（Byte） 该位可量化下一个参数（单位移动）
Bit 6 – 3	定义格式化后 PIN 在 APDU 命令中的位置（相对于 Lc 后的第一个数据）。该位置基于系统单位的类型指示符来确定（最大值为 1111，15 个系统单位）
Bit 2	PIN 对齐的位掩码 若为 0h：数据左对齐 若为 1h：数据右对齐
Bit 1-0	PIN 格式类型 00h：二进制 01h：BCD 10h：ASCII

附录B. bmPINBlockString 说明

位号	说明
Bit 7 - 4	加入 APDU 命令的 PIN 长度的大小，单位为比特。（如果为 0，则不会在 APDU 命令中加入有效的 PIN 长度）。
Bit 3 - 0	PIN 长度信息：经过对齐和格式化后 PIN 数据块的大小，单位为字节

附录C. bmPINLengthFormat

位号	说明
Bit 7-5	RFU
Bit 4	系统单位的类型指示符 若为 0h：系统的单位是比特（Bit） 若为 1h：系统的单位是字节（Byte）
Bit 3 - 0	根据前述参数指出 PIN 长度在 APDU 命令中的位置 （最大值为 1111，15 个系统单位）。

附录D. 示例代码（PC/SC 2.0 第 10 部分）

```
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>
#include <winscard.h>

#define FEATURE_VERIFY_PIN_START    0x01
#define FEATURE_VERIFY_PIN_FINISH   0x02
#define FEATURE_MODIFY_PIN_START    0x03
#define FEATURE_MODIFY_PIN_FINISH   0x04
#define FEATURE_GET_KEY_PRESSED     0x05
#define FEATURE_VERIFY_PIN_DIRECT   0x06
#define FEATURE_MODIFY_PIN_DIRECT   0x07
#define FEATURE_MCT_READERDIRECT    0x08
#define FEATURE_MCT_UNIVERSAL       0x09
#define FEATURE_IFD_PIN_PROP        0x0A
#define FEATURE_ABORT               0x0B

#define FEATURE_SIZE                (FEATURE_ABORT + 1)

#define IOCTL_SMARTCARD_GET_FIRMWARE_VERSION    SCARD_CTL_CODE(2078)
#define IOCTL_SMARTCARD_DISPLAY_LCD_MESSAGE     SCARD_CTL_CODE(2079)
#define IOCTL_SMARTCARD_READ_KEY                SCARD_CTL_CODE(2080)
#define CM_IOCTL_GET_FEATURE_REQUEST            SCARD_CTL_CODE(3400)

#pragma pack(push, 1)
typedef struct _PIN_VERIFY_STRUCTURE {
    BYTE bTimeOut;
    BYTE bTimeOut2;
    BYTE bmFormatString;
    BYTE bmPINBlockString;
    BYTE bmPINLengthFormat;
    USHORT wPINMaxExtraDigit;
    BYTE bEntryValidationCondition;
    BYTE bNumberMessage;
    USHORT wLangId;
    BYTE bMsgIndex;
    BYTE bTeoPrologue[3];
    ULONG ulDataLength;
    BYTE abData[1];
} PIN_VERIFY_STRUCTURE, *PPIN_VERIFY_STRUCTURE;

typedef struct _PIN_MODIFY_STRUCTURE {
    BYTE bTimeOut;
    BYTE bTimeOut2;
    BYTE bmFormatString;
    BYTE bmPINBlockString;
    BYTE bmPINLengthFormat;
    BYTE bInsertionOffsetOld;
    BYTE bInsertionOffsetNew;
    USHORT wPINMaxExtraDigit;
    BYTE bConfirmPIN;
    BYTE bEntryValidationCondition;
    BYTE bNumberMessage;
    USHORT wLangId;
    BYTE bMsgIndex1;
    BYTE bMsgIndex2;
    BYTE bMsgIndex3;
    BYTE bTeoPrologue[3];
```

```
    ULONG ulDataLength;
    BYTE abData[1];
} PIN_MODIFY_STRUCTURE, *PPIN_MODIFY_STRUCTURE;

typedef struct _PIN_PROPERTIES_STRUCTURE {
    USHORT wLcdLayout;
    BYTE bEntryValidationCondition;
    BYTE bTimeOut2;
} PIN_PROPERTIES_STRUCTURE, *PPIN_PROPERTIES_STRUCTURE;

typedef struct _READ_KEY_OPTION {
    BYTE bTimeOut;
    WORD wPINMaxExtraDigit;
    BYTE bKeyReturnCondition;
    BYTE bEchoLCDStartPosition;
    BYTE bEchoLCDMode;
} READ_KEY_OPTION;
#pragma pack(pop)

int main(int argc, char *argv[])
{
    SCARDCONTEXT hSCardContext;
    LONG lReturn;

    lReturn    =    SCardEstablishContext(SCARD_SCOPE_USER,    NULL,    NULL,
&hSCardContext);
    if (lReturn != SCARD_S_SUCCESS)
    {
        printf("Error:SCardEstablishContext  failed  with  error  0x%08x\n",
lReturn);
        return 1;
    }

    char **readerName = NULL;
    int numReaders = 0;
    int i;

    LPTSTR pmszReaders = NULL;
    LPTSTR pReader;
    DWORD cch = SCARD_AUTOALLOCATE;

    lReturn = SCardListReaders(hSCardContext, NULL, (LPTSTR) &pmszReaders,
&cch);
    if (lReturn == SCARD_S_SUCCESS)
    {
        pReader = pmszReaders;
        while (*pReader != '\0')

        {
            printf("Reader:%s\n", pReader);

            // Advance to the next value
            pReader = pReader + strlen(pReader) + 1;

            numReaders++;
        }

        // Allocate reader name
        readerName = new char*[numReaders];
        if (readerName == NULL)
        {
```

```
            printf("Error: not enough memory\n");
            exit(1);
        }

        i = 0;
        pReader = pmszReaders;
        while (*pReader != '\0')
        {
            readerName[i] = new char[strlen(pReader) + 1];
            if (readerName[i] == NULL)
            {
                printf("Error: not enough memory\n");
                exit(1);
            }
            strcpy(readerName[i], pReader);
            i++;

            // Advance to the next value
            pReader = pReader + strlen(pReader) + 1;
        }

        // Free the memory
        SCardFreeMemory(hSCardContext, pmszReaders);
    }

    if (numReaders == 0)
    {
        printf("Error: cannot find reader in the system\n");
        return 1;
    }

    SCARDHANDLE hCard;
    DWORD dwAP;

    const int BUFFER_SIZE = 300;
    BYTE bSendBuffer[BUFFER_SIZE];
    DWORD dwSendBufferLen;
    BYTE bRecvBuffer[BUFFER_SIZE];
    DWORD dwRecvBufferLen;

    BYTE bOutputBuffer[100];

    DWORD dwNumBytesReturned;

    DWORD featureControlCodes[FEATURE_SIZE];
    DWORD controlCode;

    // Connect to the first reader
    printf("Connecting to %s...\n", readerName[0]);
    lReturn       =       SCardConnect(hSCardContext,       readerName[0],
SCARD_SHARE_SHARED,
        SCARD_PROTOCOL_T0 | SCARD_PROTOCOL_T1, &hCard, &dwAP);
    if (lReturn != SCARD_S_SUCCESS)
        printf("Error:SCardConnect failed with error 0x%08x\n", lReturn);
    else
    {
        // Get feature request
        printf("Getting feature request...\n");
        dwRecvBufferLen = sizeof(bRecvBuffer);
        lReturn = SCardControl(hCard, CM_IOCTL_GET_FEATURE_REQUEST,
            NULL, 0,
```

```
            bRecvBuffer, dwRecvBufferLen, &dwRecvBufferLen);
        if (lReturn != SCARD_S_SUCCESS)
            printf("Error:SCardControl   failed   with   error   0x%08x\n",
lReturn);
        else
        {
            printf("Response:");
            for (i = 0; i < dwRecvBufferLen; i++)
                printf("%02X ", bRecvBuffer[i]);
            printf("\n");

            memset(featureControlCodes, 0, sizeof(featureControlCodes));

            i = 0;
            while (i < dwRecvBufferLen)
            {
                // Get the feature
                if ((bRecvBuffer[i] >= FEATURE_VERIFY_PIN_START) &&
                    (bRecvBuffer[i] <= FEATURE_ABORT))
                {
                    // Get the TLV
                    if (i + 1 + 4 < dwRecvBufferLen)
                    {
                        // Get the length field
                        if (bRecvBuffer[i + 1] == 4)
                        {
                            controlCode  = bRecvBuffer[i + 2] << 24;
                            controlCode |= bRecvBuffer[i + 3] << 16;
                            controlCode |= bRecvBuffer[i + 4] << 8;
                            controlCode |= bRecvBuffer[i + 5];

                            featureControlCodes[bRecvBuffer[i]]           =
controlCode;
                        }

                    }
                }

                // Get the next feature
                if (i + 1 < dwRecvBufferLen)
                    i += bRecvBuffer[i + 1] + 2;
                else
                    break;
            }
        }

        printf("Beginning transaction...\n");
        lReturn = SCardBeginTransaction(hCard);
        if (lReturn != SCARD_S_SUCCESS)
            printf("Error:SCardBeginTransaction    failed    with    error
0x%08x\n", lReturn);

        // Send card command for PIN verification (ACOS3)
        dwSendBufferLen = 13;
        memcpy(bSendBuffer,
"\x80\x20\x06\x00\x08\xFF\xFF\xFF\xFF\xFF\xFF\xFF\xFF", dwSendBufferLen);

        // Create PIN verify structure
        PPIN_VERIFY_STRUCTURE  pPinVerify  =  (PPIN_VERIFY_STRUCTURE)  new
BYTE[sizeof(PIN_VERIFY_STRUCTURE) - 1 + dwSendBufferLen];
        if (pPinVerify == NULL)
```

```
        {
            printf("Error: not enough memory\n");
            exit(1);
        }

        // Initialize PIN verify structure (ACOS3)
        pPinVerify->bTimeOut                  = 0;
        pPinVerify->bTimeOut2                 = 0;
        pPinVerify->bmFormatString            = 0;
        pPinVerify->bmPINBlockString          = 0x08;
        pPinVerify->bmPINLengthFormat         = 0;
        pPinVerify->wPINMaxExtraDigit         = 0x0408;
        pPinVerify->bEntryValidationCondition = 0x03;
        pPinVerify->bNumberMessage            = 0x01;
        pPinVerify->wLangId                   = 0x0409;
        pPinVerify->bMsgIndex                 = 0;
        pPinVerify->bTeoPrologue[0]           = 0;
        pPinVerify->bTeoPrologue[1]           = 0;
        pPinVerify->bTeoPrologue[2]           = 0;
        pPinVerify->ulDataLength = dwSendBufferLen;
        memcpy(pPinVerify->abData, bSendBuffer, dwSendBufferLen);

        // Verify PIN
        printf("Verifying PIN using VERIFY_PIN_DIRECT...\n");
        dwRecvBufferLen = sizeof(bRecvBuffer);
        lReturn                    =                    SCardControl(hCard,
featureControlCodes[FEATURE_VERIFY_PIN_DIRECT],

            pPinVerify, sizeof(PIN_VERIFY_STRUCTURE) - 1 + dwSendBufferLen,
            bRecvBuffer, dwRecvBufferLen, &dwRecvBufferLen);
        if (lReturn != SCARD_S_SUCCESS)
            printf("Error:SCardControl   failed   with   error   0x%08x\n",
lReturn);
        else
        {
            printf("Response:");
            for (i = 0; i < dwRecvBufferLen; i++)
                printf("%02X ", bRecvBuffer[i]);
            printf("\n");
        }

        delete [] ((BYTE*) pPinVerify);

        // Send card command for PIN modification (ACOS3)
        dwSendBufferLen = 13;
        memcpy(bSendBuffer,
"\x80\x24\x00\x00\x08\xFF\xFF\xFF\xFF\xFF\xFF\xFF\xFF", dwSendBufferLen);

        // Create PIN modify structure
        PPIN_MODIFY_STRUCTURE  pPinModify  =  (PPIN_MODIFY_STRUCTURE)  new
BYTE[sizeof(PIN_MODIFY_STRUCTURE) - 1 + dwSendBufferLen];
        if (pPinModify == NULL)
        {
            printf("Error: not enough memory\n");
            exit(1);
        }

        // Initialize PIN modify structure (ACOS3)
        pPinModify->bTimeOut                  = 0;
        pPinModify->bTimeOut2                 = 0;
        pPinModify->bmFormatString            = 0;
```

```
        pPinModify->bmPINBlockString          = 0x08;
        pPinModify->bmPINLengthFormat         = 0;
        pPinModify->bInsertionOffsetOld       = 0;
        pPinModify->bInsertionOffsetNew       = 0;
        pPinModify->wPINMaxExtraDigit         = 0x0408;
        pPinModify->bConfirmPIN               = 0x01;
        pPinModify->bEntryValidationCondition = 0x03;
        pPinModify->bNumberMessage            = 0x02;
        pPinModify->wLangId                   = 0x0409;
        pPinModify->bMsgIndex1                = 0;
        pPinModify->bMsgIndex2                = 1;
        pPinModify->bMsgIndex3                = 0;
        pPinModify->bTeoPrologue[0]           = 0;
        pPinModify->bTeoPrologue[1]           = 0;
        pPinModify->bTeoPrologue[2]           = 0;
        pPinModify->ulDataLength = dwSendBufferLen;
        memcpy(pPinModify->abData, bSendBuffer, dwSendBufferLen);

        // Modify PIN

        printf("Modifying PIN using MODIFY_PIN_DIRECT...\n");
        dwRecvBufferLen = sizeof(bRecvBuffer);
        lReturn                    =                SCardControl(hCard,
featureControlCodes[FEATURE_MODIFY_PIN_DIRECT],
            pPinModify, sizeof(PIN_MODIFY_STRUCTURE) - 1 + dwSendBufferLen,
            bRecvBuffer, dwRecvBufferLen, &dwRecvBufferLen);
        if (lReturn != SCARD_S_SUCCESS)
            printf("Error:SCardControl   failed   with   error   0x%08x\n",
lReturn);
        else
        {
            printf("Response:");
            for (i = 0; i < dwRecvBufferLen; i++)
                printf("%02X ", bRecvBuffer[i]);
            printf("\n");
        }

        delete [] ((BYTE*) pPinModify);

        printf("Ending transaction...\n");
        lReturn = SCardEndTransaction(hCard, SCARD_LEAVE_CARD);
        if (lReturn != SCARD_S_SUCCESS)
            printf("Error:SCardEndTransaction failed with error 0x%08x\n",
lReturn);

        // Get IFD PIN properties
        printf("Getting IFD PIN properties...\n");
        dwRecvBufferLen = sizeof(bRecvBuffer);
        lReturn                    =                SCardControl(hCard,
featureControlCodes[FEATURE_IFD_PIN_PROP],
            NULL, 0,
            bRecvBuffer, dwRecvBufferLen, &dwRecvBufferLen);
        if (lReturn != SCARD_S_SUCCESS)
            printf("Error:SCardControl   failed   with   error   0x%08x\n",
lReturn);
        else
        {
            printf("Response:");
            for (i = 0; i < dwRecvBufferLen; i++)
                printf("%02X ", bRecvBuffer[i]);
            printf("\n");
```

```
        }

        // Display LCD message to ACR83
        char *msg = "Hello";
        printf("Displaying message (%s) to LCD...\n", msg);
        lReturn = SCardControl(hCard, IOCTL_SMARTCARD_DISPLAY_LCD_MESSAGE,
            msg, strlen(msg),
            bOutputBuffer, sizeof(bOutputBuffer), &dwNumBytesReturned);
        if (lReturn != SCARD_S_SUCCESS)
            printf("Error:SCardControl   failed   with   error   0x%08x\n",
lReturn);

        else
        {
            // Check status
            if ((dwNumBytesReturned >= 2) &&
                (bOutputBuffer[0] == 0) && (bOutputBuffer[1] == 0))
                printf("The message is displayed successfully\n");
            else
                printf("Error: cannot display LCD message\n");
        }

        // Read key from ACR83
        READ_KEY_OPTION readKeyOption;
        char keyString[100];
        DWORD len;
        BYTE keyReturnCondition;

        // Initialize read key option
        readKeyOption.bTimeOut = 0;
        readKeyOption.wPINMaxExtraDigit = 0x0408;
        readKeyOption.bKeyReturnCondition = 0x01;
        readKeyOption.bEchoLCDStartPosition = 0;
        readKeyOption.bEchoLCDMode = 0x01;

        printf("Reading key...\n");
        lReturn = SCardControl(hCard, IOCTL_SMARTCARD_READ_KEY,
            &readKeyOption, sizeof(READ_KEY_OPTION),
            bOutputBuffer, sizeof(bOutputBuffer), &dwNumBytesReturned);
        if (lReturn != SCARD_S_SUCCESS)
            printf("Error:SCardControl   failed   with   error   0x%08x\n",
lReturn);
        else
        {
            // Check status
            if ((dwNumBytesReturned >= 2) &&
                (bOutputBuffer[0] == 0) && (bOutputBuffer[1] == 0))
            {
                if (dwNumBytesReturned >= 3)
                    keyReturnCondition = bOutputBuffer[2];
                else
                    keyReturnCondition = 0;

                len = 0;
                if (dwNumBytesReturned >= 4)
                {
                    len = dwNumBytesReturned - 3;
                    memcpy(keyString, bOutputBuffer + 3, len);
                }

                // Set the last NULL character
```

```
                keyString[len] = '\0';

                printf("Key  Return  Condition:0x%02x,  Key  String:%s\n",
keyReturnCondition, keyString);
            }

            else
                printf("Error: cannot read key\n");
        }

        lReturn = SCardDisconnect(hCard, SCARD_LEAVE_CARD);
        if (lReturn != SCARD_S_SUCCESS)
            printf("Error:SCardDisconnect  failed  with  error  0x%08x\n",
lReturn);
    }

    lReturn = SCardReleaseContext(hSCardContext);
    if (lReturn != SCARD_S_SUCCESS)
        printf("Error:SCardReleaseContext  failed  with  error  0x%08x\n",
lReturn);

    // Deallocate reader name
    for (i = 0; i < numReaders; i++)
        delete [] readerName[i];
    delete readerName;

    return 0;}
```

附录E. *设置 bKeyReturnCondition*

bKeyReturnCondition	OR 操作数
若达到 PIN 码最大长度	01h
若按下 ACR83 设备的 KEY_E	02
若 ACR83 会话超时时间到	04h
若按下 ACR83 设备的 KEY_C	08h
若按下 ACR83 设备的 KEY_BACK	10h
若按下 ACR83 设备的 KEY_FN	20h

注：*将值设为对特定的 OR 操作数进行 OR 运算。*

附录F. *响应错误代码*

下表汇总了 ACR83（CCID）可能返回的错误代码：.

错误代码	状态
0001h	BAD_PARAMETER
0083h	SLOTERROR_LCDCOMMANDERROR
0084h	SLOTERROR_WRONGCONFIRMPIN
0085h	SLOTERROR_UNKNOWN_LCD
0086h	SLOTERROR_MAXPINSIZE_EQUAL_ZERO
00EFh	SLOTERROR_PIN_CANCELLED
00F0h	SLOTERROR_PIN_TIMEOUT