Advanced Card Systems Ltd.
Card & Reader Technologies

ACR1255U-J1

ACS 安全蓝牙™ NFC 读写器

参考手册 V1.13

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

版本历史

发布日期	修订说明	版本号
2015-09-07	• 初始发布	1.00
2016-02-02	• 更新 2.0 节 – 特性 • 更新 4.1.2 节 – 电池寿命 • 更新 4.5.1 节 – 按键 • 更新 4.5.3 节 – LED 状态指示灯 • 更新 4.5.4 节 – 蜂鸣器 • 更新 5.3 节 – 认证 • 更新 6.7.2 节 – 获取序列号 • 更新 6.7.16 节 – 休眠模式选项 • 更新 6.6.18 节 – 读取 Tx 功率值	1.01
2016-06-06	• 更新 5.6.5 节–卡片状态通知命令 • 更新 6.2.1 节 – ATR 的生成 • 更新 6.4.1 节 – 加载认证密钥 • 更新 6.7.6 节 – 设置 LED 和蜂鸣器状态指示器 • 更新 6.7.7 节 – 读取 LED 和蜂鸣器状态指示器 • 更新 6.7.8 节 – 设置自动 PICC 轮询 • 更新 6.7.10 节 – 设置 PICC 操作参数 • 更新 6.7.11 节 – 读取 PICC 操作参数 • 更新 6.7.12 节 – 设置自动 PPS • 更新 6.7.13 节 – 读取自动 PPS • 更新 6.7.16 节 – 休眠模式选项 • 更新 6.7.17 节 修改 Tx 功率命令 • 更新 6.7.18 节 读取 Tx 功率值 • 增加 6.7.19 节 – 重写客户主密钥	1.02
2016-09-16	• 更新产品推广名称	1.03
2017-01-10	• 更新命令格式及示例 • 增加相互认证通信示例（5.7.1 节） • 更新表 5 值 • 删除 PIN 密码说明 • 更新 2.0 节 – 特性 • 更新 4.1.2 节 – 电池寿命 • 更新 5.6.5 节 – 卡片状态通知命令 • 增加 6.4 节 – PC/SC 2.0 第 3 部分 APDU 命令 • 更新 6.7.4 节 – LED 颜色 • 更新 6.7.17 节 – 修改 Tx 功率命令 • 更新 6.7.18 节 – 读取 Tx 功率值	1.04

发布日期	修订说明	版本号
2017-10-02	• 更新表 5 – 服务句柄和 UUID 消息列表 • 更新 5.3 节 – 认证 • 更新 5.7 节 – 相互认证 • 新增 6.7 节 – 访问 FeliCa 标签	1.05
2017-12-28	• 更新产品照片	1.06
2018-05-24	• 更新 5.2 节 – 配置文件选择 • 更新 5.6 节 – 蓝牙通信协议	1.07
2018-09-13	• 更新格式 • 更新 2.0 节 – 特性 • 更新 3.0 节 – 系统框图 • 更新 5.6.4 节 – APDU 命令 • 更新 5.6.6 节 – 硬件错误响应 • 更新 5.7.6 节 – RDR_to_SPH_ACK (错误处理) • 更新 6.3.5 节– 蜂鸣器控制（Buzzer Control） • 更新 6.3.6 节– 设置 LED 和蜂鸣器状态指示器 • 更新 6.3.8 节 – 设置自动 PICC 轮询 • 更新 6.3.10 节 – 设置 PICC 操作参数 • 更新 6.3.11 节 – 读取 PICC 操作参数 • 更新 6.3.12 节 – 设置自动 PPS • 更新 6.3.16 节 – 设置休眠模式选项 • 新增 6.3.17 节 – 读取休眠模式选项 • 更新 6.3.18 节 – 设置 Tx 功率值 • 更新 6.3.19 节 – 读取 Tx 功率值 • 新增 6.3.21 节 – 获取电池电量	1.08
2018-10-08	• 新增 6.3.17 节 – 读取休眠模式选项 • 更新 6.3.18 节 – 设置 Tx 功率值 • 新增 6.3.21 节 – 获取电池电量	1.09
2019-06-14	• 更新产品推广名称 • 更新 5.0 节 – 软件设计 • 更新 5.1.3 节 – 认证	1.10
2019-07-08	• 更新 5.1.3 节 – 认证	1.11

发布日期	修订说明	版本号
2019-08-28	• 更新 6.2.4.7 节 - 复制值块 (Copy Value Block） • 更新 6.3.8 节 - 设置自动 PICC 轮询（Set Automatic PICC Polling） • 更新 6.3.9 节 - 读取自动 PICC 轮询（Read Automatic PICC Polling） • 更新 6.3.17 节 - 读取休眠模式选项	1.12
2020-06-16	• 新增 6.2.2.2 节 – 获取 PICC 数据 • 新增 6.3.22 节 – 读取 PICC 类型 • 更新 6.1 节 – PC/SC API • 更新 6.3.6 节 – 设置 LED 和蜂鸣器状态指示器 • 更新 6.3.7 节 – 读取 LED 和蜂鸣器状态指示器 • 更新 6.3.9 节 – 读取自动 PICC 轮询	1.13

目录

图目录

表目录

1.0. 简介

ACR1255U-J1 ACS 安全蓝牙™ NFC 读写器是计算机/移动设备与非接触式智能卡/NFC 设备之间的通信接口。它可以为多种卡片建立一个从计算机/移动设备到智能卡的统一接口。它兼顾了卡片的各种特性而使得计算机软件程序员不需要负责有关智能卡操作的技术细节, 在许多情况下, 这些细节与智能卡系统的实施并无太大关系。

1.1. 符号和缩写

缩写	说明
ATR	属性请求和属性响应
DEP	数据交换协议请求及数据交换协议响应
DSL	取消请求和取消响应
PSL	参数选择请求和参数选择响应
RLS	发布请求和发布响应
WUP	唤醒请求和唤醒响应
DID	设备 ID
BS	发送比特周期
BR	接收比特周期
PP	协议参数

表1： 符号和缩写

2.0. 特性

- USB 全速接口
- 蓝牙™接口
- 即插即用—支持 CCID 标准, 具有高度的灵活性
- 智能卡读写器：
  - 非接触接口：
    - 读写速率高达 424 Kbps
    - 内置天线用于读写非接触式标签, 读取智能卡的距离可达 50 mm（视标签的类型而定）
    - 支持符合 ISO 14443 第 4 部分的 A 类和 B 类卡, MIFARE®卡, FeliCa 卡和全部四种 NFC（ISO/IEC 18092）标签
    - 内建防冲突特性（任何时候都只能访问 1 张标签）
    - 支持 AES-128 加密算法（CBC 加密模式）
    - NFC 支持：
      - 卡片读/写模式
- 应用程序编程接口：
  - 支持 PC/SC
  - 支持 CT-API（通过 PC/SC 上一层的封装）
- 内置部件：
  - 两个用户可控的双色 LED 指示灯
  - 用户可控的蜂鸣器
- 具有 USB 固件升级能力[1]
- 支持 Android™ 4.3 及以上版本[2]
- 支持 iOS 8.0 及以上版本[3]
- 符合下列标准：
  - EN 60950/IEC 60950
  - ISO 14443
  - ISO 18092
  - 蓝牙
  - PC/SC
  - CCID
  - CE
  - FCC
  - RoHS
  - REACH
  - VCCI（日本）
  - TELEC （日本）
  - Microsoft® WHQL

[1] *适用于连接计算机模式。*
[2] *使用 ACS 定义的 Android 库*
[3] *使用 ACS 定义的 iOS 库*

3.0. 系统框图

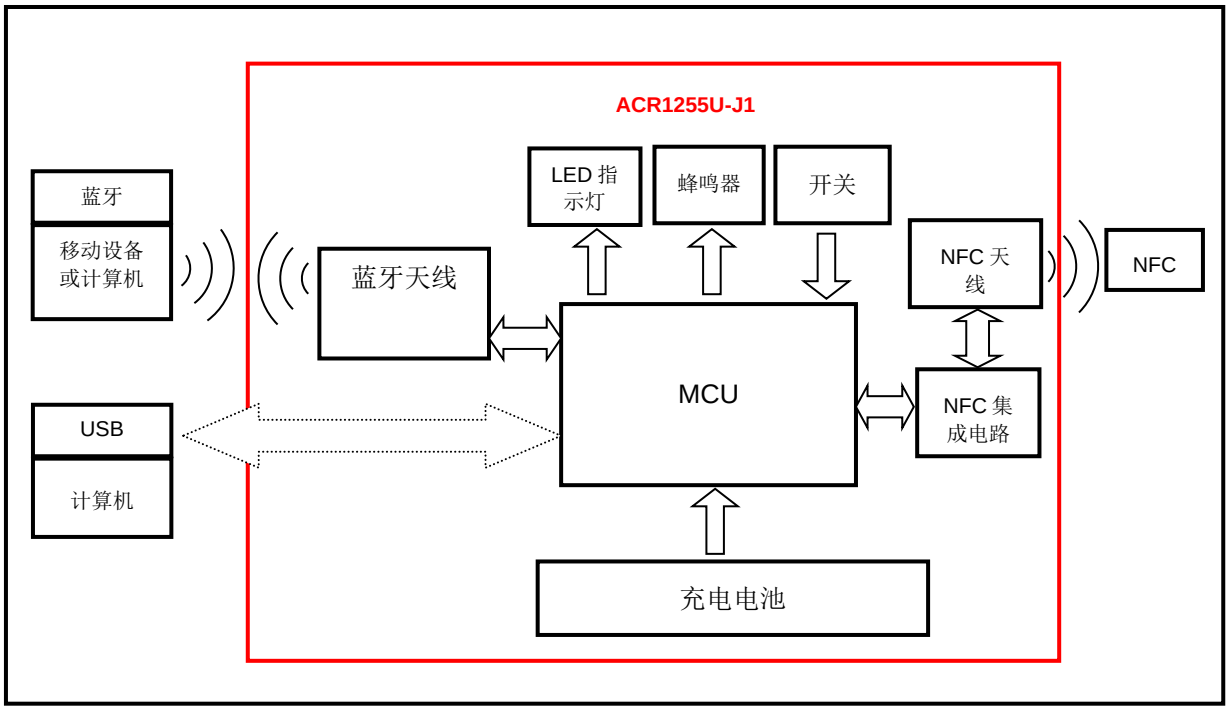


图1： ACR1255U-J1 架构

ACR1255U-J1 与其他设备之间的 USB 通信架构采用 CCID 协议，而所有的 NFC 通信则符合 PC/SC 标准。

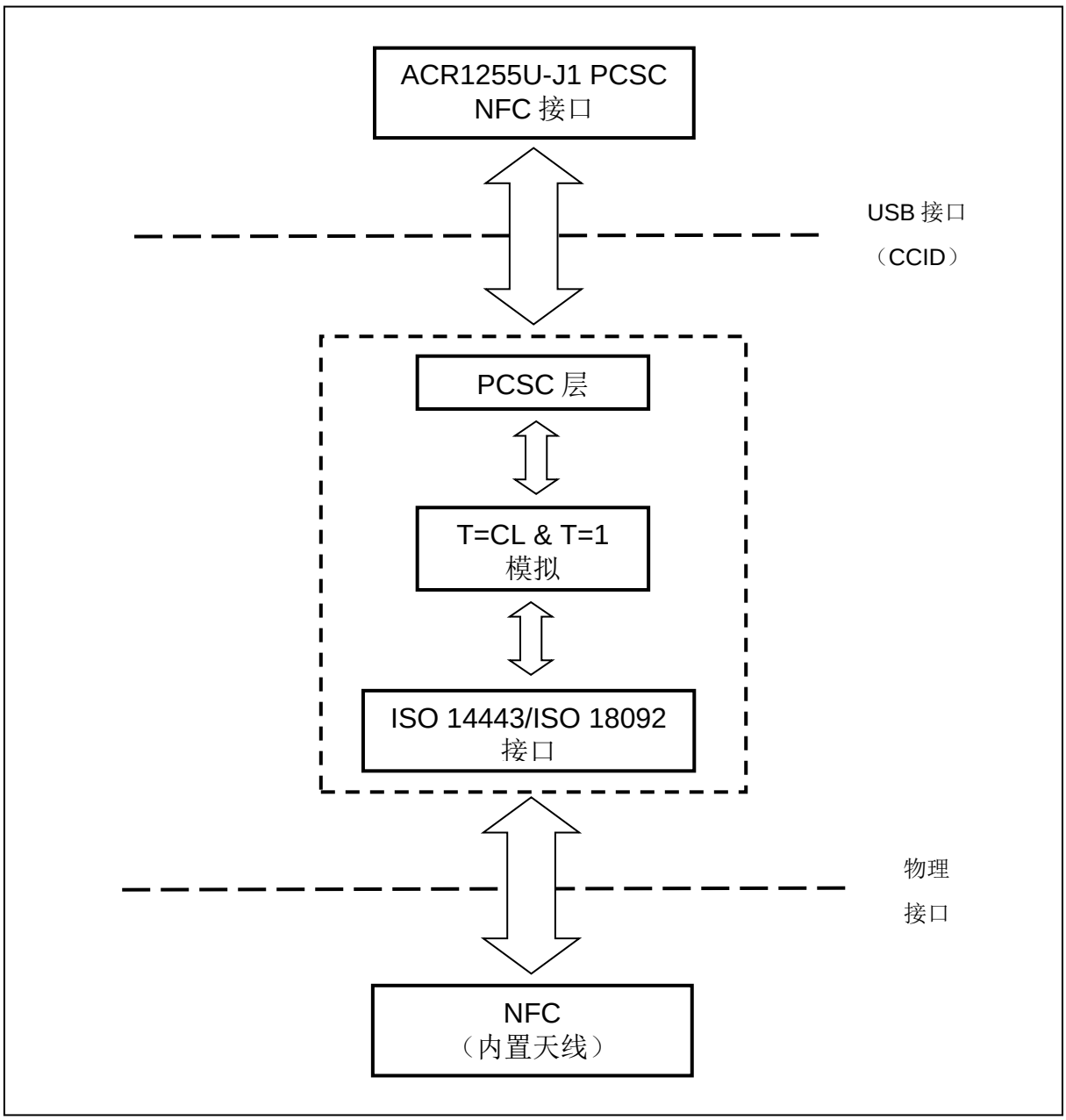


图2： ACR1255U-J1 的 USB 通信架构

蓝牙协议栈的架构如下：

<table>
<tr><td>GAP 从角色配置</td><td>SIM 卡访问配置</td><td>GAP 从机绑定管理器</td></tr>
<tr><td colspan="2" rowspan="3">GAP</td><td>GATT</td></tr>
<tr><td rowspan="2">SMP</td><td>ATT</td></tr>
<tr><td>L2CAP</td></tr>
<tr><td colspan="3">HCI</td></tr>
<tr><td colspan="3">链路层</td></tr>
<tr><td colspan="3">物理层</td></tr>
</table>

图3： 蓝牙协议栈

4.0. 硬件设计

4.1. 电池

ACR1255U-J1 使用容量为 320 mAh 的锂离子充电电池。

4.1.1. 电池充电

电池没电时 ACR1255U-J1 可以多种模式充电：关机模式，USB 模式，蓝牙模式；前提条件是读写器连接了电源插座。

4.1.2. 电池寿命

电池寿命与设备使用情况相关。以下是各种工作条件下预估的电池寿命：

模式	预计电池寿命
工作模式	14 天* (1)
待机模式	22 天 (2)
关机模式	2 年

表2：预计电池寿命长度

***注：** 结果可能因采用的智能卡不同而发生变化。.*

(1) 在蓝牙模式下，每天进行 10 次操作，每次操作一分钟。

(2) 在蓝牙模式下，休眠时间设为 60 秒，每天唤醒一次。

4.2. 蓝牙接口

ACR1255U-J1 使用蓝牙作为设备和计算机/移动设备的匹配媒介。

4.3. USB 接口

ACR1255U-J1 可通过 mini USB 作为电池充电端口来连接计算机。另外利用此端口，还可以在计算机连接模式下操作 ACR1255U-J1。

4.3.1. 通信参数

ACR1255U-J1 通过符合 USB 2.0 规范的 USB 端口连接计算机。它支持 USB 全速模式，速率为 12Mbps。

引脚	信号	功能
1	V_{BUS}	为读写器提供+5 V 的电源
2	D-	ACR1255U-J1 和计算机之间以差分信号传输数据。
3	D+	ACR1255U-J1 和计算机之间以差分信号传输数据。
4	GND	参考电压等级

表3：USB 接口配线

4.3.2. 端点

ACR1255U-J1 通过如下端点与主计算机进行通信：

控制端点（Control Endpoint）	用于进行设置和控制
批量输出（Bulk OUT）	用于从主机发送至 ACR1255U-J1 的命令 （数据包的大小为 64 字节）
批量输入（Bulk IN）	用于从 ACR1255U-J1 发送至主机的响应 （数据包的大小为 64 字节）
中断输入（Interrupt IN）	用于从 ACR1255U-J1 发送至主机的卡片状态消息 （数据包的大小为 8 字节）

4.4. NFC 接口

ACR1255U-J1 与非接触卡之间的接口符合 ISO 14443 或 ISO 18092 规范，并进行了某些限制或提升来增强读写器的实用功能。

4.4.1. 载波频率

ACR1255U-J1 的载波频率为 13.56 MHz。

4.4.2. 卡片轮询

ACR1255U-J1 会自动检测进入天线场内的非接触卡。此功能支持 ISO 14443-4 的 A 类卡和 B 类卡，FeliCa 卡，Topaz 卡，MIFARE 卡以及 NFC 标签。

4.5. 用户接口

4.5.1. 按键

ACR1255U-J1 有三个按键，位置如下图所示：

按键	说明
ANT_KEY	唤醒休眠
MODEL_KEY	模式选择
UPDATE_KEY	使用启动载入器

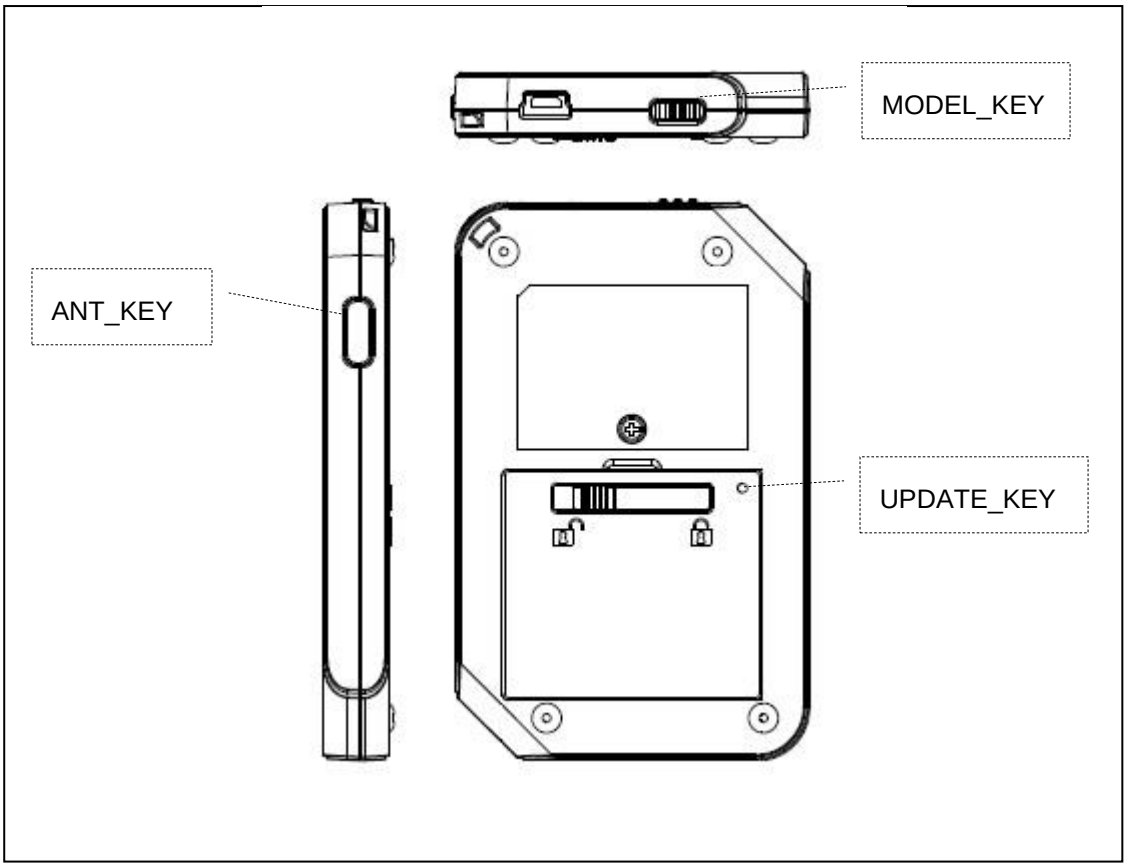


图4： ACR1255U-J1 的按键

4.5.2. 模式选择开关

ACR1255U-J1 提供三种模式：USB, 关机和蓝牙。用户一次可以选择一种模式作为数据传输接口。

符号	开关	启用模式
USB	USB	连机
	关闭	断电
	蓝牙	蓝牙

4.5.3. LED 状态指示灯

ACR1255U-J1 提供二个双色 LED 指示灯, 用以显示不同的操作状态, 包括：

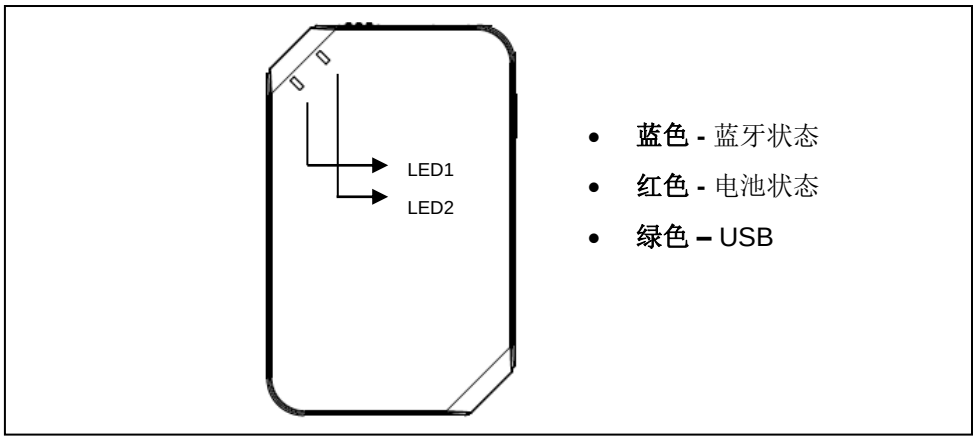


图5： LED 操作状态

模式	颜色	LED 操作	状态
蓝牙模式	蓝色 (LED1)	关闭	• 读写器断电 • 没有配对的蓝牙设备 • 读写器处于 USB 模式
		缓慢闪亮 v1[4]	配对的蓝牙设备成功通过认证, 正在等待指令
		缓慢闪亮 v2[5]	蓝牙设备配对成功, 正在进行卡片轮询, 没有发现卡片
		快速闪亮	读写器和匹配设备之间正在传输数据
		快速—缓慢闪亮	• 上电 • 搜索配对设备
		开	• 蓝牙设备匹配了读写器 • 卡片存在
	红色 (LED2)	关闭	• 电池已经充满 • 读写器断电 • 电池电压高于 3.5 V 并且没有 USB 充电电源
		缓慢闪亮	电量低
		开	电池正在充电
USB 模式	绿色 (LED2)	关闭	读写器断电
		缓慢闪亮	没有卡片或读写器, 正等待指令
		快速闪亮	智能卡和读写器之间在进行读/写
		开	存在卡片或读写器正等待指令

[4] *LED 亮 500 毫秒，熄 500 毫秒*
[5] *LED 亮 200 毫秒，熄 1800 毫秒*

模式	颜色	LED 操作	状态
	橙色 [6] (LED2)	开	设备已上电
	红色 (LED1)	关闭	• 电池已经充满 • 读写器断电 • 电池电压高于 3.5 V 并且没有 USB 充电电源
		缓慢闪亮	电量低
		开	电池正在充电

4.5.4. 蜂鸣器

ACR1255U-J1 配有一个蜂鸣器, 用于通知用户卡片轮询, 蓝牙连接, 休眠模式, 以及电量不足等状态。

蜂鸣器操作	事件
1 声短的鸣响	检测到卡片或卡片被移出
1 声长的鸣响	读写器已断电

[6] *红色和绿色的 LED 均开启。*

5.0. 软件设计

5.1. 蓝牙通信

5.1.1. 蓝牙连接流程

以下是蓝牙连接的程序流程：

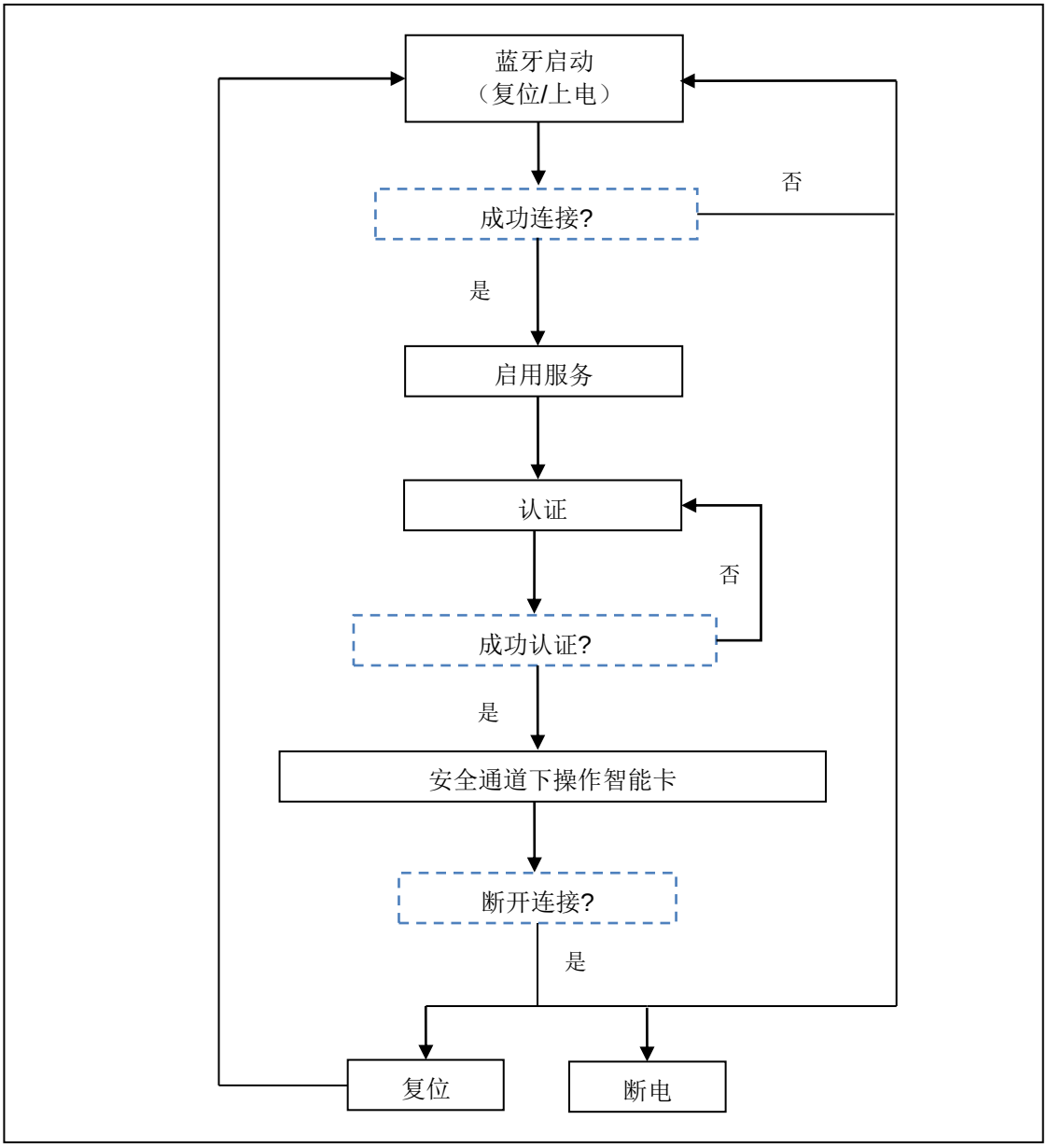


图6： 蓝牙连接流程

5.1.2. 配置文件选择

ACR1255U-J1 被设计为一款使用蓝牙技术作为数据传输接口的智能卡读写器。它提供了能与三条通道进行命令通信的客制化服务：第一条通道用于命令请求，第二条通道用于命令响应/卡片通知，第三条通道 RFU。

并且，当读写器处于蓝牙模式时，当前读写器的电池状态很重要，因而客制化的电池服务可以将当前电池状态告知匹配设备。当电池状态发生变化，读写器也将通过指定通道通知匹配设备。简而言之，电池电量级别分为三类：电量充足(≥ 3.78 V)，电量低(< 3.78 V and ≥ 3.68 V)，和无电量 (< 3.68 V).

最后，为了给用户提供更多读写器信息，增加了客制化设备信息服务。可以人工读取或者由应用请求读取，该特性包括型号，序列号，固件版本和厂商名称。

服务	UUID	通道
智能卡	3C4AFFF1-4783-3DE5-A983-D348718EF133	命令请求
	3C4AFFF2-4783-3DE5-A983-D348718EF133	命令响应/卡片通知
	3C4AFFF3-4783-3DE5-A983-D348718EF133	RFU
电池	2A19	电量级别
设备信息	2A23	系统 ID
	2A24	型号
	2A25	序列号
	2A26	固件版本
	2A27	硬件版本
	2A29	生产商名称

表4： ACR1255U-J1 的蓝牙服务

属性名称	UUID	句柄
DeviceName	2A00	03h
Send(Reader → Paired device)	3C4AFFF1-4783-3DE5-A983-D348718EF133	2Ah
Receive(Paired device → Reader)	3C4AFFF2-4783-3DE5-A983-D348718EF133	2Dh
BatteryLevel	2A19	14h
Manufacturer	2A29	1Eh
SerialNumber	2A25	16h
FW_Version	2A26	18h
ModelNumber	2A24	14h

表5： ACR1255U-J1 服务句柄和 UUID 消息列表

5.1.3. 认证

向 ACR1255U-J1 加载任意敏感数据之前，数据处理服务器必须通过 ACR1255U-J1 的认证，然后才有权修改读写器内部的安全数据。ACR1255U-J1 使用了相互认证的方法。

为了更好的进行说明，请参考下图（图中省去了桥接设备，以便更简单明了的进行说明）：

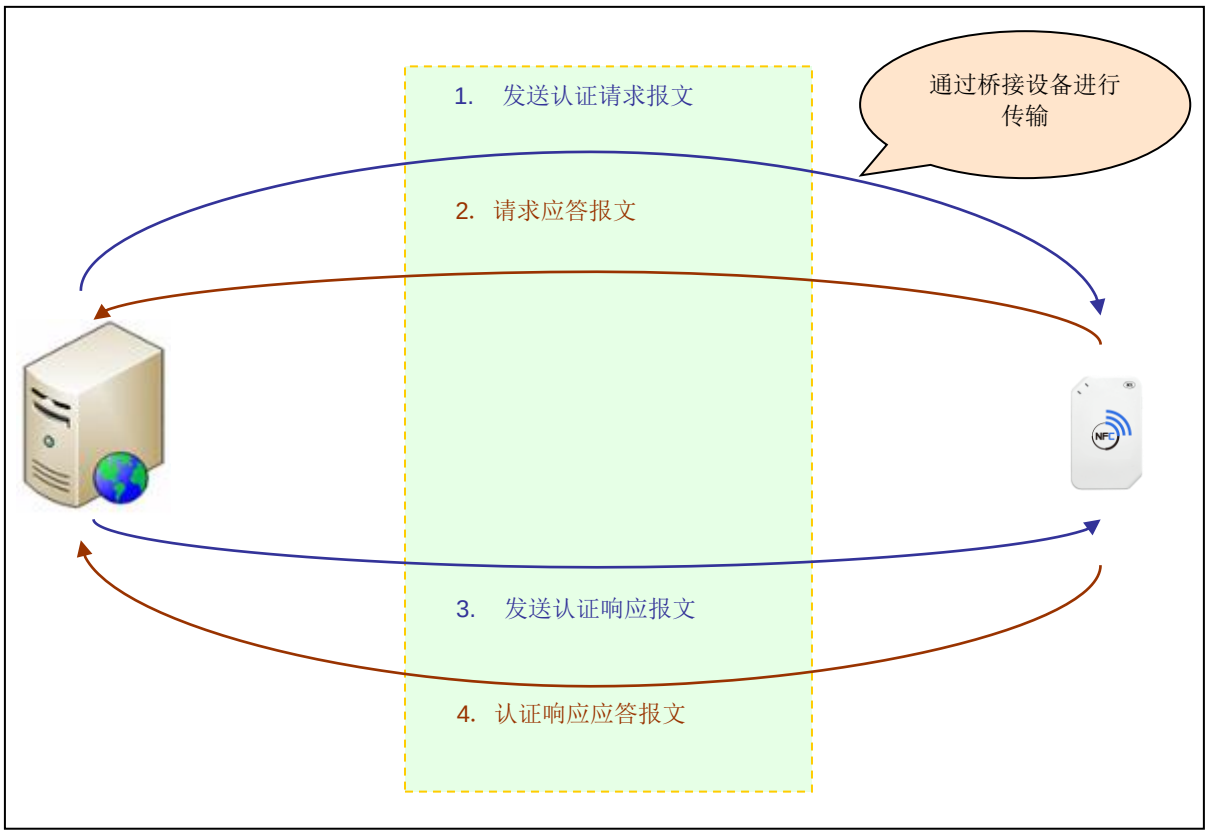


图7： 认证步骤

认证成功后，在 ACR1255U-J1 和数据服务器中分别生成一个 16 字节过程密钥。

默认客户主密钥: **41 43 52 31 32 35 35 55 2D 4A 31 20 41 75 74 68**

注：*认证密钥最多错误六（6）次，一旦超过，读写器会被锁定并无法使用。*

有关更多详细信息，您可以联系 ACS 销售代表

5.1.4. 通信配置

通信配置文件应为：

开始字节（Start byte） + 长度（Len） + 数据块（Datablock） + 校验（Check） + 终止字节（Stop byte）

数据域	大小（字节）	说明
Start byte	1	值：05h
Len	2	Len 表示 Datablock 数据域中的字节数
Datablock	N	数据（帧格式的说明见**帧格式**)
Check	1	Check 表示 Len 和 Datablock 数据域中各字节的 XOR 值
Stop byte	1	值：0Ah

5.1.5. 帧格式

5.1.5.1. 数据帧格式

命令

HID 帧	长度（字节）	说明
CmdMessageType	1	命令
Length	2	数据长度
Slot	1	默认值：00h
Seq	1	序号
Param	1	参数
Checksum	1	校验和
Data	0-N	数据

帧格式应为：

CmdMessageType + Length + Slot + Seq + Param + Checksum + Data

如果命令总长度（包括标识符，长度和数据包）大于 20 字节，读写器或匹配的设备将自动将其划分为多个帧。

响应

HID 帧	长度（字节）	说明
RspMessageType	1	响应
Length	2	数据长度
Slot	1	默认值：00h
Seq	1	序号
Slot Status/Param	1	卡槽状态/参数
Checksum	1	校验和
Data	0-N	数据

数据校验和用于检测数据无线传输过程中可能引发的错误。计算数据校验和：

XOR { RspMessageType, Length, Slot, Seq, SlotStatus/Param, Data }.

例： 62 00 00 00 01 00 63 => 校验和 = 63h

5.1.6. 蓝牙通信协议

ACR1255U-J1 采用预定义协议, 通过蓝牙接口与匹配的设备进行通信。协议类似于 CCID 命令通道和响应通道的格式。

命令	支持模式	发送方	说明
62h	已认证	匹配设备	PICC 上电
63h	已认证	匹配设备	PICC 下电
65h	已认证	匹配设备	获取卡片状态
6Fh	已认证	匹配设备	交换 APDU
6Bh	已认证	匹配设备	直接命令

表6： 命令码总表

命令	支持模式	发送方	说明
80h	已认证	读写器	数据块的响应
81h	已认证	读写器	卡槽状态的响应
83h	已认证	读写器	直接命令的响应
50h	已认证	读写器	通知卡片状态
51h	已认证	读写器	硬件错误的响应

表7： 响应码总表

5.1.6.1. 卡片上电

此命令用于发送上电请求给读写器。

命令格式

偏移	数据域	大小	值	说明
0	*CmdMessageType*	1	**62h**	-
1	*Length*	2	00 00h	数据长度
3	*Slot*	1	00h	-
4	*Seq*	1	00h	序号
5	*Param*	1	00h	参数
6	*Checksum*	1	-	*Checksum* 表示命令中所有字节的 XOR 值
7	*Data*	N	-	数据

响应数据格式

偏移	数据域	大小	值	说明
0	*RspMessageType*	1	**80h**	-
1	*Length*	2	-	数据长度
3	*Slot*	1	00h	-
4	*Seq*	1	00h	序号
5	*Param*	1	-	参数 Bit0 为 LSB Bit 0-1 = 00h = 有卡（激活） = 01h = 有卡（未激活） = 02h = 无卡 Bit 6 = “0” = 未发生错误 = “1” = 发生错误 例如：参数= 41h = 0100 0001b 在有卡（未激活）时发生错误
6	*Checksum*	1	-	*Checksum* 表示响应中所有字节的 XOR 值
7	*Data*	N	-	数据

例：

响应 = 80 00 08 00 00 00 B3 3B 83 80 01 41 07 00 44

ATR = 3B 83 80 01 41 07 00 44

5.1.6.2. 卡片下电

此命令用于发送下电请求给读写器。

命令格式

偏移	数据域	大小	值	说明
0	*CmdMessageType*	1	**63h**	-
1	*Length*	2	0000h	数据长度
3	*Slot*	1	00h	-
4	*Seq*	1	00h	序号
5	*Param*	1	00h	参数
6	*Checksum*	1	-	*Checksum* 表示命令中所有字节的 XOR 值
7	*Data*	N	-	数据

响应数据格式

偏移	数据域	大小	值	说明
0	*RspMessageType*	1	**81h**	-
1	*Length*	2	-	数据长度
3	*Slot*	1	00h	-
4	*Seq*	1	00h	序号
5	*Param*	1	-	参数 Bit 0 为 LSB Bit 0-1 = 00h = 有卡（激活） = 01h = 有卡（未激活） = 02h = 无卡 Bit 6 = “0” = 未发生错误 = “1” = 发生错误 例如：参数 = 41h = 0100 0001b 在有卡（未激活）时发生错误
6	*Checksum*	1	-	*Checksum* 表示响应中所有字节的 XOR 值
7	*Data*	N	-	数据

5.1.6.3. 获取卡槽状态

此命令用于检查是否有卡片插入。

命令格式

偏移	数据域	大小	值	说明
0	*CmdMessageType*	1	**65h**	-
1	*Length*	2	00 00h	数据长度
3	*Slot*	1	00h	-
4	*Seq*	1	00h	序号
5	*Param*	1	00h	参数
6	*Checksum*	1	-	*Checksum* 表示命令中所有字节的 XOR 值
7	*Data*	N	-	数据

响应数据格式

偏移	数据域	大小	值	说明
0	*RspMessageType*	1	**81h**	-
1	*Length*	2	0000h	数据长度
3	*Slot*	1	00h	-
4	*Seq*	1	00h	序号
5	*Param*	1	-	卡片状态： Bit 0 为 LSB Bit 0-1 = 00h = 有卡（激活） = 01h = 有卡（未激活） = 02h = 无卡 Bit 6 = “0” = 未发生错误 = “1” = 发生错误 例如：参数 = 41h = 0100 0001b 在有卡（未激活）时发生错误
6	*Checksum*	1	-	*Checksum* 表示响应中所有字节的 XOR 值
7	*Data*	N	-	数据

5.1.6.4. APDU 命令

此命令用于发送 APDU 命令给读写器。

注：支持扩展 APDU 的“Param”选项适用于 2.01.00 及以上版本固件。

命令格式

偏移	数据域	大小	值	说明
0	*CmdMessageType*	1	**6Fh**	-
1	*Length*	2	-	数据长度; 最大长度：256 字节
3	*Slot*	1	00h	-
4	*Seq*	1	00h	序号
5	*Param*	1	-	参数 短 APDU 级 00h – 默认 扩展 APDU 级 00h –APDU 命令随本命令开始和结束。 01h –APDU 命令随本命令开始，并在下一个 APDU 命令中继续。 02h –此数据字段继续传递 APDU 命令并结束该 APDU 命令。 03h – 此数据字段继续传递命令 APDU，后面跟随另外一个数据块。 10h – 空的数据字段，下一个响应会继续传递响应 APDU
6	*Checksum*	1	-	*Checksum* 表示命令中所有字节的 XOR 值
7	*Data*	N	-	数据

响应数据格式

偏移	数据域	大小	值	说明
0	*RspMessageType*	1	**80h**	-
1	*Length*	2		数据长度
3	*Slot*	1	00h	-
4	*Seq*	1	00h	序号

偏移	数据域	大小	值	说明
5	*Param*	1	-	参数 对于 Bit0-5, Bit 0 为 LSB 短 APDU 级 00h – default 扩展 APDU 级 00h – 响应 APDU 随本命令开始和结束。 01h –响应 APDU 随本命令开始，并会继续。 02h – 此数据字段继续传递响应 APDU 并结束该响应 APDU。 03h – 此数据字段继续传递响应 APDU, 后面跟随另外一个数据块。 10h – 空的数据字段。下一个命令将继续传递命令 APDU。, Bit 6 用作错误位 Bit 6 = “0” = 未发生错误 = “1” = 发生错误
6	*Checksum*	1	-	*Checksum* 表示响应中所有字节的 XOR 值
7	*Data*	N	-	数据

范例：

向卡片发送 600 字节数据

1. 命令 = 6F 01 00 00 XX 01 校验和 (256 字节数据)
   响应 = 80 00 00 00 XX 10 校验和
2. 命令 = 6F 01 00 00 XX 03 校验和 (256 字节数据)
   响应 = 80 00 00 00 XX 10 校验和
3. 命令 = 6F 00 58 00 XX 02 校验和 (88 字节数据)
   响应 = 80 00 02 00 XX 00 校验和 90 00

从卡片接收 600 字节数据

1. 命令 = 6F 00 07 00 XX 00 校验和 00 B0 87 00 00 02 58
   响应 = 80 01 00 00 XX 01 校验和 (256 字节数据)
2. 命令 = 6F 00 00 00 XX 10 校验和
   响应 = 80 01 00 00 XX 03 校验和 (256 字节数据)

3. 命令 = 6F 00 00 00 XX 10 校验和

   响应 = 80 00 5A 00 XX 02 校验和 (88 字节数据) 90 00

5.1.6.5. 卡片状态通知命令

此命令用于检查卡片的状态。

偏移	数据域	大小	值	说明
0	*RspMessageType*	1	**50h**	-
1	*Length*	2	00 00h	数据长度
3	*Slot*	1	00h	-
4	*Seq*	1	00h	序号
5	*Param*	1	-	状态： 02h = 无卡 03h = 有卡
6	*Checksum*	1	-	*Checksum* 表示消息中所有字节的 XOR 值
7	*Data*	N	-	数据

注：*若收到消息 "52 00 00 00 00 01 53"，表示读写器已进入睡眠模式。*

5.1.6.6. 硬件错误响应

此消息用于在收到不正确的命令时返回错误消息。

响应数据格式

偏移	数据域	大小	值	说明
0	*RspMessageType*	1	51h	-
1	*Length*	2	-	数据长度
3	*Slot*	1	00h	-
4	*Seq*	1	00h	序号
5	*Param*	1	-	状态： 01h = 校验和错误 02h = 超时 03h = 命令错误 04h = 未被授权 05h = 未定义错误 06h = 接收数据错误 07h = 认证次数超过限制错误
6	*Checksum*	1	-	Checksum 表示消息中所有字节的 XOR 值
7	*Data*	N	-	数据

5.1.6.7. 直接命令

此命令用于访问读写器的扩展功能。

命令格式

偏移	数据域	大小	值	说明
0	*CmdMessageType*	1	**6Bh**	-
1	*Length*	2	-	数据长度
3	*Slot*	1	00h	-
4	*Seq*	1	00h	序号
5	*Param*	1	00h	参数
6	*Checksum*	1	-	Checksum 表示命令中所有字节的 XOR 值
7	*Data*	N	-	数据

响应数据格式

偏移	数据域	大小	值	说明
0	*RspMessageType*	1	**83h**	-
1	*Length*	2	-	数据长度
3	*Slot*	1	00h	-
4	*Seq*	1	00h	序号
5	*Param*	1	00h	参数
6	*Checksum*	1	-	*Checksum* 表示响应中所有字节的 XOR 值
7	*Data*	N	-	数据

5.1.6.7.1. 直接命令示例

1. 开启自动轮询

   请求命令：E0 00 00 40 01

   响应命令：E1 00 00 40 01

   例：

   请求：6B 00 05 00 00 00 CF E0 00 00 40 01

   响应： 83 00 05 00 00 00 66 E1 00 00 40 01

2. 关闭自动轮询

   请求命令：E0 00 00 40 00

   响应命令：E1 00 00 40 00

例：

请求：6B 00 05 00 00 00 CE E0 00 00 40 00

响应： 83 00 05 00 00 00 67 E1 00 00 40 00

3. 验证请求

请求命令： E0 00 00 45 00

响应命令：E1 00 00 45 00 + Data(16 字节)

其中：Data = 16 字节的随机数

例：

请求 (SPH_to_RDR_ReqAuth): 6B 00 05 00 00 00 CB E0 00 00 45 00

响应 (RDR_to_SPH_AuthRsp1): 83 00 15 00 00 00 21 E1 00 00 45 00 77 59 E8 62 B7 80 0D 0A CE 9A 03 9B E9 48 EF 05

4. 验证响应

请求命令：E0 00 00 46 00 + Data(32 字节)

响应命令：E1 00 00 46 00 + Data(16 字节)

例：

请求(SPH_to_RDR_AuthRsp): 6B 00 25 00 00 00 EA E0 00 00 46 00 A6 81 17 91 9F 46 07 AE AE 4E 94 8E 05 14 E8 C8 25 F7 90 05 76 F8 DE 7D 6D ED 55 3F 80 10 C2 CA

响应 (RDR_to_SPH_AuthRsp2): 83 00 15 00 00 00 51 E1 00 00 46 00 47 D5 50 54 F3 49 D4 17 B1 65 40 21 9B DA C9 B2

5.1.7. 相互认证和加密协议

蓝牙模式下, 相互认证成功后将对**蓝牙通信协议**中的通信协议进行加密和传输。

5.1.7.1. 蓝牙认证程序流程

如**认证**所示, 相互认证用于避免中间人攻击。整个相互认证过程用到的命令概述如下表：

序号	命令	支持模式	发送方	说明
1	6Bh	已连接	配对设备	SPH_to_RDR_ReqAuth
2	83h	已连接	读写器	RDR_to_SPH_AuthRsp1
3	6Bh	已连接	配对设备	SPH_to_RDR_AuthRsp
4	83h	已连接	读写器	RDR_to_SPH_AuthRsp2

表8： 相互认证命令概述

5.1.7.2. SPH_to_RDR_ReqAuth

此命令请求 ACR1255U-J1 对匹配的密钥生成设备进行认证。

有关认证流程的更多信息, 请参考**认证**。

偏移	数据域	大小	值	说明	加密
0	*bMessageType*	1	6Bh	-	否
1	*LEN1 LEN2 (wLength)*	2	0005h	长度为两个字节, 表示此数据域中的额外字节的数量, LEN1 是 MSB, 而 LEN2 是 LSB。	
3	*卡槽号*	1	00h	-	
4	*序号*	1	00h	-	
5	*参数*	1	00h	卡槽状态	
6	*wChecksum*	1	CBh	CSUM 表示命令中所有字节的 XOR 值。	
7	*数据*	5	E0 00 00 45 00h	-	否

如果接收的命令消息无误, 将收到响应 RDR_to_SPH_AuthRsp1。否则, 则收到包含错误信息的响应 RDR_to_SPH_ACK 。

5.1.7.3. RDR_to_SPH_AuthRsp1

此命令是由 ACR1255U-J1 发送的对 SPH_to_RDR_ReqAuth 的响应。

偏移	数据域	大小	值	说明	加密
0	*bMessageType*	1	**83h**	-	否
1	*LEN1 LEN2 (wLength)*	2	0015h	长度为两个字节, 表示 abRndNum 数据域中的额外字节的数量, LEN1 是 MSB, 而 LEN2 是 LSB。	否
3	*卡槽号*	1	00h	-	否
4	*序号*	1	00h	-	否
5	*参数*	1	00h	卡槽状态	-
6	*wChecksum*	1	-	wChecksum 表示命令中所有字节的 XOR 值	否
7	*abRndNum*	21	E1 00 00 45 00 + 16 字节随机数	abRndNum[0:15] – 16 字节随机数 所有的 16 字节数据都必须使用当前存储在 ACR1255U-J1 内的客户主密钥进行加密。 “E1 00 00 45 00”不需要加密	是

5.1.7.4. SPH_to_RDR_AuthRsp

此命令属于认证流程的第二阶段。设备发送 SPH_to_RDR_ReqAuth 命令给 ACR1255U-J1 之后，如果命令无误，读写器将返回 RDR_to_SPH_AuthRsp1 消息。

RDR_to_SPH_AuthRsp1 包含一系列通过客户主密钥加密的 16 字节随机数。匹配的密钥生成设备应当使用正确的客户主密钥进行解密，并将其添加到 16 字节随机数的末尾。然后使用客户主密钥解密全部 32 字节的随机数，使用此命令将结果返回给 ACR1255U-J1，以成功完成认证。

偏移	数据域	大小	值	说明	加密
0	*bMessageType*	1	6Bh	-	否
1	*LEN1 LEN2 (wLength)*	2	0025h	长度为两个字节，表示 abAuthData 数据域中的额外字节的数量，LEN1 是 MSB，而 LEN2 是 LSB。	否
3	*卡槽号*	1	00h	-	否
4	*序号*	1	00h	-	否
5	*参数*	1	00h	卡槽状态	否
6	*wCheckSum*	1	-	wCheckSum 表示命令中所有字节的 XOR 值	否
7	*abAuthData*	37	E0 00 00 46 00 + 32 字节随机数	abAuthData[0:15] – 数据处理服务器生成的 16 字节随机数 abAuthData[16:31] – 接收自 ACR1255U-J1 的 16 字节的解密后的随机数 所有 32 字节数据在 AES128 CBC 加密模式下通过客户主密钥进行解密处理。 “E0 00 00 46 00”不需要解密	是

如果接收的命令消息无误并且由配对设备返回的随机数也是正确的，响应为 RDR_to_SPH_AuthRsp2。否则会收到包含错误信息的响应 RDR_to_SPH_ACK 。

5.1.7.5. RDR_to_SPH_AuthRsp2

此命令是由 ACR1255U-J1 发送的对 SPH_to_RDR_AuthRsp 的响应。

偏移	数据域	大小	值	说明	加密
0	*bMessageType*	1	**83h**	-	否
1	*LEN1 LEN2 (wLength)*	2	0015h	长度为两个字节, 表示 abRndNum 数据域中的额外字节的数量, LEN1 是 MSB, 而 LEN2 是 LSB。	否
3	*卡槽号*	1	00h	-	否
4	*序号*	1	00h	-	否
5	*参数*	1	00h	卡槽状态	否
6	*wCheckSum*	1		wCheckSum 表示命令中所有字节的 XOR 值	否
20	*abRndNum*	21	E1 00 00 46 00 + 16 字节随机数	abRndNum[0:15] – 接收自数据处理服务器的 16 字节随机数 所有的 16 字节数据都必须使用当前存储在 ACR1255U-J1 内的客户主密钥进行加密。 “E1 00 00 46 00”不需要加密	是

5.1.7.6. RDR_to_SPH_ACK (错误处理)

这是由 ACR1255U-J1 发送给配对设备的错误处理确认信息，用于确认接受一些命令消息。在通信过程中，指定的错误消息均使用 RDR_to_SPH_ACK 进行传输。命令不用加密。

命令	支持模式	发送方	说明
51h	已连接/已认证	读写器	RDR_to_SPH_ACK (错误处理)

此消息还会在需要时包含错误代码。

偏移	数据域	大小	值	说明	加密
0	*bMessageType*	1	51h	错误处理响应头	否
1	*LEN1LEN2 (wLength)*	2	00h	-	
3	*卡槽号*	1	00h	-	
4	*序号*	1	00h	-	
3	*bErrorCode*	1	-	指定先前处理命令消息时所发生错误的代码。可能的错误代码见下表。	
4	*wChecksum*	1	-	CSUM 表示命令中所有字节的 XOR 值。	

数据域	值	说明
bErrorCode	01h	校验和错误
	02h	超时
	03h	命令错误
	04h	未授权
	05h	未定义错误
	06h	接收到的数据错误
	07h	认证次数超过限制错误

表9： 相互认证错误代码

6.0. 主机编程（连机）API

6.1. PC/SC API (For Windows®)

这一章节将会描述一些用于应用程序编程的 PC/SC API。关于这些 API 的更多细节，请参考 Microsoft MSDN 库或 PC/SC 工作组。

6.1.1. SCardEstablishContext

此函数用于建立进行设备数据库操作的资源管理器上下文。

请参考：http://msdn.microsoft.com/en-us/library/windows/desktop/aa379479%28v=vs.85%29.aspx

执行其他 PCSC 操作前，必须先执行此函数。

例:

```
#define SCARD_SCOPE_USER 0

SCARDCONTEXT hContext;
int retCode;
void main ()
{
   // To establish the resource manager context and assign it to "hContext"
   retCode = SCardEstablishContext(SCARD_SCOPE_USER,
                  NULL,
                  NULL,
                  &hContext);
   if (retCode != SCARD_S_SUCCESS)
   {
      // Establishing resource manager context failed
   }
   else
   {
      // Establishing resource manager context successful
      // Further PCSC operation can be performed
   }
}
```

6.1.2. SCardListReaders

此函数可以给出系统中在指定读卡器组集合中的读卡器名字列表（去掉重复的）。调用者提供一个读卡器组列表，函数返回这些指定组里面的读卡器名字列表。无法识别的组名会被忽略。这个函数只会返回当前系统中可以使用的组里面的读卡器。

请参考：http://msdn.microsoft.com/en-us/library/windows/desktop/aa379793%28v=vs.85%29.aspx

例:

```
#define SCARD_SCOPE_USER 0

SCARDCONTEXT hContext; // Resource manager context
int retCode;
char readerName [256]; // List reader name

void main ()
{
   // To establish the resource manager context and assign to "hContext"
   retCode = SCardEstablishContext(SCARD_SCOPE_USER,
                  NULL,
                  NULL,
                  &hContext);
   if (retCode != SCARD_S_SUCCESS)
   {
      // Establishing resource manager context failed
   }
   else
   {
      // Establishing resource manager context successful
      // List the available reader which can be used in the system
      retCode = SCardListReaders (hContext,
                  NULL,
                  readerName,
                  &size);
      if (retCode != SCARD_S_SUCCESS)
      {
         // Listing reader fail
      }
      if (readerName == NULL)
      {
         // No reader available
      }
      else
      {
         // Reader listed
      }
   }
}
```

6.1.3. SCardConnect

此函数利用特定资源管理器上下文，在应用程序与包含在特定读卡器中的智能卡之间建立一条连接。如果特定读卡器中没有卡片，会返回一条错误信息。

请参考：http://msdn.microsoft.com/en-us/library/windows/desktop/aa379473%28v=vs.85%29.aspx

例:

```
#define SCARD_SCOPE_USER 0

SCARDCONTEXT      hContext;          // Resource manager context
SCARDHANDLE       hCard;             // Card context handle
unsigned long     dwActProtocol;     // Establish active protocol
int               retCode;
char              readerName [256]; // List reader name
char              rName [256];       // Reader name for connection

void main ()
{
   …
      if (readerName == NULL)
      {
         // No reader available
      }
      else
      {
         // Reader listed
      rName = "ACS ACR1255U-J1 PICC Reader 0";  // Depends on what
                                                reader be used
                                                // Should connect to
                                                PICC interface
         retCode = SCardConnect(hContext,
               rName,
               SCARD_SHARE_SHARED,
               SCARD_PROTOCOL_T0,
               &hCard,
               &dwActProtocol);
         if (retCode != SCARD_S_SUCCESS)
         {
            // Connection failed (May be because of incorrect reader
         name, or no card was detected)
         }
         else
         {
            // Connection successful
         }
      }
   }
```

6.1.4. SCardControl

此函数提供对读卡器的直接控制。你可以在 *SCardConnect* 函数成功调用后，但在 *ScardDisconnect* 函数成功调用前随时调用此函数。它对读卡器状态的影响取决于控制码。

请参考：http://msdn.microsoft.com/en-us/library/windows/desktop/aa379474%28v=vs.85%29.aspx

注：外设控制*中的命令要使用此 API 进行发送。*

例:

```
#define SCARD_SCOPE_USER    0

#define EscapeCommand 0x310000 + 3500*4
   SCARDCONTEXT          hContext;          // Resource manager context
   SCARDHANDLE           hCard;             // Card context handle
   unsigned long         dwActProtocol;     // Established active protocol
   int                   retCode;
   char                  readerName [256]; // Lists reader name
   char                  rName [256];       // Reader name for connection
   BYTE                  SendBuff[262],     // APDU command buffer
                         RecvBuff[262];     // APDU response buffer
   BYTE                  FWVersion [20],    // For storing firmware
                                               version message
   BYTE                  ResponseData[50]; // For storing card response
   DWORD                 SendLen,           // APDU command length
                         RecvLen;           // APDU response length
void main ()
   {
      …
      rName = "ACS ACR1255U-J1 PICC Reader 0";      // Depends on what
                                                    reader will be used
                                                    // Should connect to
                                                    PICC interface

         retCode = SCardConnect(hContext,
            rName,
            SCARD_SHARE_DIRECT,
            SCARD_PROTOCOL_T0| SCARD_PROTOCOL_T1,
            &hCard,
            &dwActProtocol);
         if (retCode != SCARD_S_SUCCESS)
         {
            // Connection failed (may be because of incorrect reader
         name, or no card was detected)
         }
         else
         {
            // Connection successful
            RecvLen = 262;
            // Get firmware version
            SendBuff[0] = 0xE0;
            SendBuff[1] = 0x00;
            SendBuff[2] = 0x00;
            SendBuff[3] = 0x18;
            SendBuff[4] = 0x00;
            SendLen = 5;
```

```
            retCode = SCardControl ( hCard,
                     EscapeCommand,
                     SendBuff,
                     SendLen,
                     RecvBuff,
                     RecvLen,
                     &RecvLen);
            if (retCode != SCARD_S_SUCCESS)
            {
               // APDU sending failed
               return;
            }
            else
            {
               // APDU sending successful
               // The RecvBuff stores the firmware version message.
               for (int i=0;i< RecvLen-5;i++)
               {
                  FWVersion[i] = RecvBuff [5+i];
               }
            }
            // Connection successful
            RecvLen = 262;

            // Turn Green LED on, turn Red LED off
            SendBuff[0] = 0xE0;
            SendBuff[1] = 0x00;
            SendBuff[2] = 0x00;
            SendBuff[3] = 0x29;
            SendBuff[4] = 0x01;
            SendBuff[5] = 0x02;  // Green LED On, Red LED off
            SendLen = 6;
            retCode = SCardControl ( hCard,
                     EscapeCommand,
                     SendBuff,
                     SendLen,
                     RecvBuff,
                     RecvLen,
                     &RecvLen);
            if (retCode != SCARD_S_SUCCESS)
            {
               // APDU sending failed
               return;
            }
            else
            {
               // APDU sending success
            }
```

6.1.5. ScardTransmit

此函数用于发送服务请求给智能卡，并接收从智能卡返回的数据。

请参考：http://msdn.microsoft.com/en-us/library/windows/desktop/aa379804%28v=vs.85%29.aspx

注：*APDU 命令（即：发送给已建立连接的卡片的命令，***MIFARE Classic (1K/4K)存储卡的 PICC 命令***和***非接触接口的私有 APDU 指令***）使用此 API 进行发送。*

例:

```
#define SCARD_SCOPE_USER      0

SCARDCONTEXT      hContext;          // Resource manager context
SCARDHANDLE       hCard;             // Card context handle
unsigned long     dwActProtocol;     // Established active protocol
int               retCode;
char              readerName [256]; // List reader name
char              rName [256];       // Reader name for connect
BYTE              SendBuff[262],     // APDU command buffer
                  RecvBuff[262];     // APDU response buffer
BYTE              CardID [8],        // For storing the FeliCa IDM/
                                     MIFARE UID
BYTE              ResponseData[50]; // For storing card response
DWORD             SendLen,           // APDU command length
                  RecvLen;           // APDU response length
SCARD_IO_REQUEST  ioRequest;

void main ()
{
   …
   rName = "ACS ACR1255U-J1 PICC Reader 0";  // Depends on what reader
                                             should be used
                                             // Should connect to PICC
                                             interface

      retCode = SCardConnect(hContext,
                     rName,
                     SCARD_SHARE_SHARED,
                     SCARD_PROTOCOL_T0,
                     &hCard,
                     &dwActProtocol);
      if (retCode != SCARD_S_SUCCESS)
      {
         // Connection failed (May be because of incorrect reader name,
         or no card was detected)
      }
      else
      {
         // Connection successful
         ioRequest.dwProtocol = dwActProtocol;
         ioRequest.cbPciLength = sizeof(SCARD_IO_REQUEST);
         RecvLen = 262;
```

```
// Get MIFARE UID/ FeliCa IDM
SendBuff[0] = 0xFF;
SendBuff[1] = 0xCA;
SendBuff[2] = 0x00;
SendBuff[3] = 0x00;
SendBuff[4] = 0x00;
SendLen = 5;
retCode = SCardTransmit( hCard,
                  &ioRequest,
                  SendBuff,
                  SendLen,
                  NULL,
                  RecvBuff,
                  &RecvLen);

if (retCode != SCARD_S_SUCCESS)
{
   // APDU sending failed
   return;
}
else
{
   // APDU sending successful
   // The RecvBuff stores the IDM for FeliCa / the UID for
   MIFARE.
   // Copy the content for further FeliCa access
   for (int i=0;i< RecvLen-2;i++)
   {
      CardID [i] = RecvBuff[i];
   }
}
```

6.1.6. ScardDisconnect

此函数用来断开先前在应用程序和目标读卡器中的智能卡之间建立的连接。

请参考：http://msdn.microsoft.com/en-us/library/windows/desktop/aa379475%28v=vs.85%29.aspx

此函数用于结束 PCSC 操作。

例:

```
#define SCARD_SCOPE_USER 0

SCARDCONTEXT       hContext;          // Resource manager context
SCARDHANDLE        hCard;             // Card context handle
unsigned long      dwActProtocol;     // Established active protocol
int                retCode;

void main ()
{
   …
         // Connection successful
      …
      retCode = SCardDisconnect(hCard,SCARD_RESET_CARD);
      if (retCode != SCARD_S_SUCCESS)
      {
         // Disconnection failed
      }
      else
      {
         // Disconnection successful
      }
   }
}
```

6.1.7. APDU 流程图

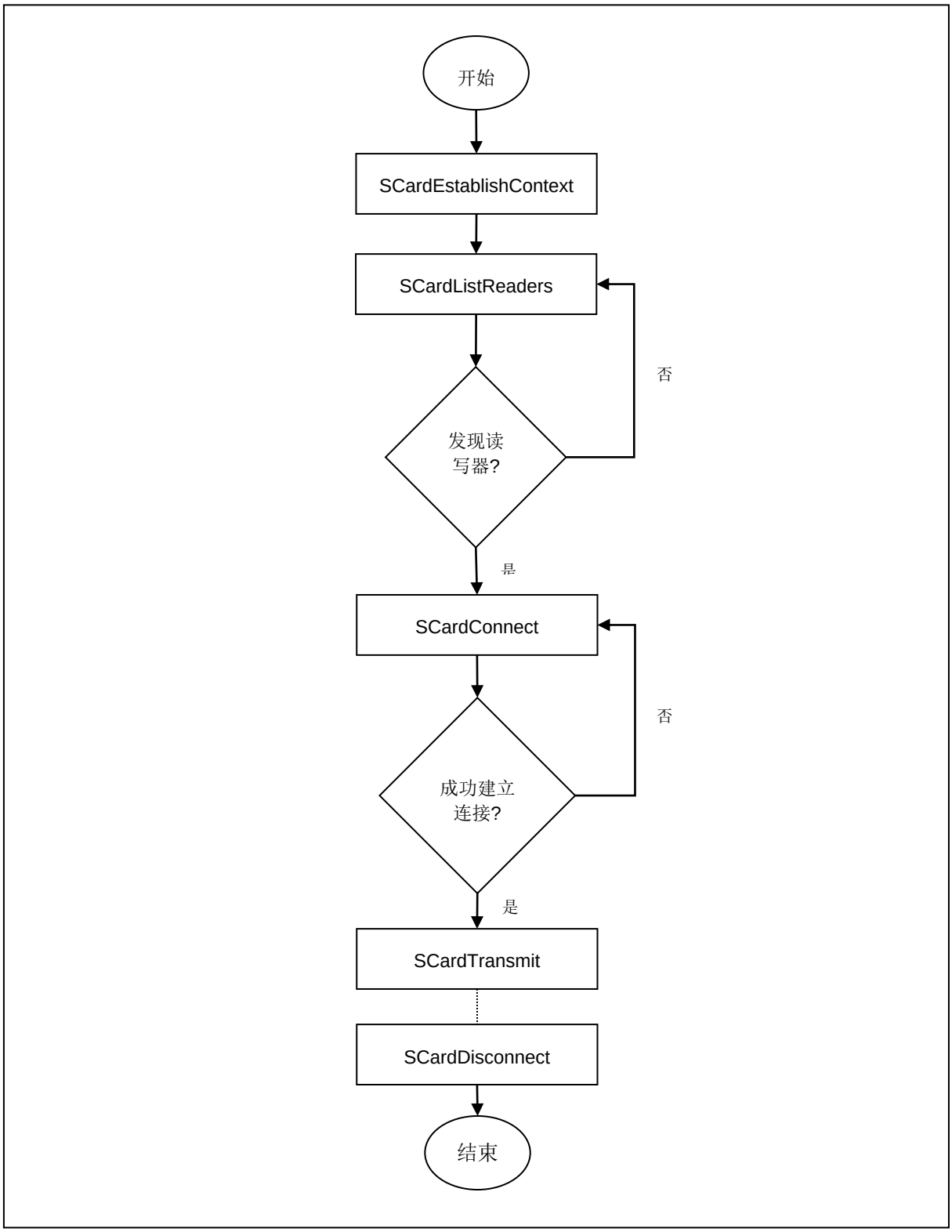


图8： ACR1255U-J1 APDU 流程图

6.1.8. 直接命令（Escape Command）流程图

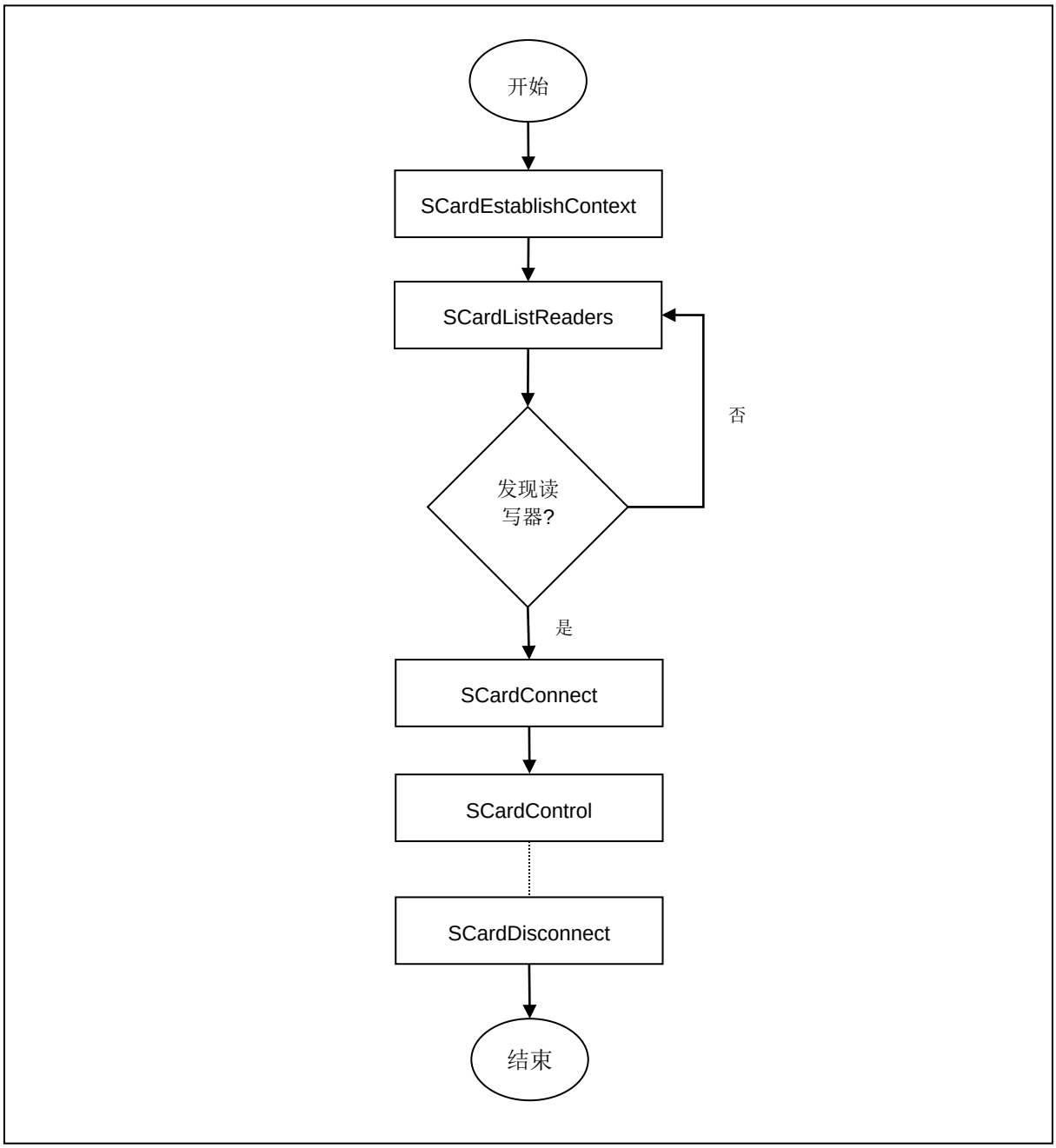


图9： ACR1255U-J1 直接命令流程图

6.2. 非接触式智能卡协议

6.2.1. ATR 的生成

读写器检测到 PICC 后, 一个 ATR 会被发送至 PCSC 驱动来识别 PICC。

6.2.1.1. ATR 信息格式（适用于 ISO 14443-3 PICC）

字节	值	标记	说明
0	3Bh	初始字符	-
1	8Nh	T0	高半字节 8 表示：后续不存在 TA1, TB1 和 TC1, 只存在 TD1。 低半字节 N 指出历史字符的个数（HistByte 0 - HistByte N-1)
2	80h	TD1	高半字节 8 表示：后续不存在 TA2, TB2 和 TC2, 只存在 TD2。 低半字节 0 表示协议类型为 T=0
3	01h	TD2	高半字节 0 表示后续不存在 TA3, TB3, TC3 和 TD3。 低半字节 1 表示协议类型为 T=1
4 至 3+N	80h	T1	类别指示字节, 80 表示在可选的 COMPACT-TLV 数据对象中可能存在状态指示。
	4Fh	Tk	应用标识符存在标识。
	0Ch		长度
	RID		注册的应用提供商标识(RID) # A0 00 00 03 06
	SS		标准字节。
	C0 ..C1h		卡片名称字节。
	00 00 00 00h	RFU	RFU # 00 00 00 00
4+N	UU	TCK	T0 至 Tk 的所有字节按位异或

例：

MIFARE Classic 1K 的 ATR = {3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 01 00 00 00 00 6Ah}

其中：

长度 (YY) = 0Ch

RID = A0 00 00 03 06h (PC/SC 工作组)

标准 (SS) = 03h (ISO 14443A, 第 3 部分)

卡片名称(C0 ..C1) = [00 01h] (MIFARE Classic 1K)

标准 (SS) = 03h：ISO 14443A, 第 3 部分

= 11h：FeliCa

卡片名称(C0 ..C1)

00 01: MIFARE Classic 1K	00 38: MIFARE Plus SL2 2K
00 02: MIFARE Classic 4K	00 39: MIFARE Plus SL2 4K
00 03: MIFARE Ultralight®	00 30: Topaz and Jewel
00 26: MIFARE Mini®	00 3B: FeliCa
00 3A: MIFARE Ultralight® C	FF 28: JCOP 30
00 36: MIFARE Plus® SL1 2K	FF [SAK]: undefined tags
00 37: MIFARE Plus SL1 4K	00 07: SRIX512

6.2.1.2. ATR 信息格式（适用于 ISO 14443-4 PICC）

<table>
<tr><th>字节</th><th>值</th><th>标记</th><th>说明</th></tr>
<tr><td>0</td><td>3Bh</td><td>初始字符</td><td>-</td></tr>
<tr><td>1</td><td>8N</td><td>T0</td><td>高半字节 8 表示：后续不存在 TA1, TB1 和 TC1, 只存在 TD1。
低半字节 N 指出历史字符的个数（HistByte 0 - HistByte N-1)</td></tr>
<tr><td>2</td><td>80h</td><td>TD1</td><td>高半字节 8 表示：后续不存在 TA2, TB2 和 TC2, 只存在 TD2。
低半字节 0 表示协议类型为 T=0</td></tr>
<tr><td>3</td><td>01h</td><td>TD2</td><td>高半字节 0 表示后续不存在 TA3, TB3, TC3 和 TD3。
低半字节 1 表示协议类型为 T=1</td></tr>
<tr><td rowspan="2">4
至
3 + N</td><td>XX</td><td>T1</td><td rowspan="2">历史字节：

ISO 14443-A：
来自 ATS 应答的历史字节。参考 ISO 14443-4 标准。

ISO 14443-B：
<table><tr><th>Byte1-4</th><th>Byte5-7</th><th>Byte8</th></tr><tr><td>ATQB 的应用数据</td><td>ATQB 的协议信息字符</td><td>高半字节 =ATTRIB 命令的 MBLI；低半字节 (RFU)=0</td></tr></table></td></tr>
<tr><td>XX
XX
XX</td><td>Tk</td></tr>
<tr><td>4+N</td><td>UU</td><td>TCK</td><td>T0 至 Tk 的所有字节按位异或</td></tr>
</table>

例 1：

MIFARE® DESFire®的 ATR = {3B 81 80 01 80 80h} // 6 个字节的 ATR

注：*使用 APDU "FF CA 01 00 00h"来区分是符合 ISO 14443A-4 的 PICC 还是符合 ISO 14443B-4 的 PICC, 并且如果有的话, 取回完整的 ATS。符合 ISO 14443A-3 或 ISO 14443B-3/4 的 PICC 会返回 ATS。*

APDU 命令 = FF CA 01 00 00h

APDU 响应 = 06 75 77 81 02 80 90 00h

ATS = {06 75 77 81 02 80h}

例 2：

EZ-link 的 ATR = {3B 88 80 01 1C 2D 94 11 F7 71 85 00 BEh}

ATQB 的应用数据 = 1C 2D 94 11h

ATQB 的协议信息 = F7 71 85h

ATTRIB 的 MBLI = 00h

6.2.2. 非接触接口的私有 APDU 指令

6.2.2.1. 获取数据（Get Data）

此命令用于获取“已建立连接的 PICC”的序列号或 ATS。

GET UID 的 APDU 结构（5 个字节）

命令	CLA	INS	P1	P2	Le
Get Data	FFh	CAh	00h 01h	00h	00h (最大长度)

若 P1 = 00h, Get UID 的响应报文结构（UID + 2 字节）

响应	响应数据域					
结果	UID (LSB)	…	…	UID (MSB)	SW1	SW2

如果 P1 = 01h, 获取 ISO 14443 A 类卡的 ATS（ATS + 2 字节）

响应	响应数据域		
结果	ATS	SW1	SW2

响应状态码

结果	SW1	SW2	含义
成功	90h	00h	操作成功完成。
警告	62h	82h	UID/ATS 的末尾先于 Le 字节到达（Le 大于 UID 的长度）。
错误	6Ch	**XX**h	长度错误（错误的 Le：‘**XX**’表示确切的数字），如果 Le 小于 UID 的长度。
错误	63h	00h	操作失败。
错误	6Ah	81h	不支持此功能

例如：

获取“已经建立连接的 PICC”的序列号：

```
UINT8 GET_UID[5] = {FF, CA, 00, 00, 00};
```

获取“已经建立连接的 ISO 14443-A PICC”的 ATS：

```
UINT8 GET_ATS[5] = {FF, CA, 01, 00, 00};
```

6.2.2.2. 获取 PICC 数据（Get PICC Data）

此命令用于获取“已建立连接的 PICC”的数据。

注：*仅适用于 2.04.xx 及以上版本的固件。*

Get PICC Data 的 APDU 格式（5 字节）

命令	CLA	INS	P1	P2	Le
Get PICC Data	FFh	CAh	00h	02h	00h

如果是 A 类卡，获取 ATQA + UID + SAK 的响应格式（2 字节 + 4/7/10 字节 + 1 字节 + 2 字节）

响应	响应数据域								
结果	ATQA	ATQA	UID (LSB)	…	…	UID (MSB)	SAK	SW1	SW2

如果是 B 类卡，获取 ATQB（12 字节+2 字节）

响应	响应数据域		
结果	ATQB	SW1	SW2

响应状态码

结果	SW1	SW2	含义
成功	90h	00h	操作成功完成
错误	63h	00h	操作失败
错误	6Ah	81h	不支持此功能

6.2.3. PC/SC 2.0 第 3 部分的 APDU 命令（2.02 或更高版本）

PC/SC 2.0 第三部分规定的命令用于将数据从应用程序透明传递给非接触标签，将接收到的数据透明返回给应用程序和协议，以及同时切换协议。

6.2.3.1. 命令和响应的 APDU 格式

命令格式

CLA	INS	P1	P2	Lc	命令数据域
FFh	C2h	00h	功能	DataLen	Data[DataLen]

其中：

功能 1 字节

00h = 会话管理

01h = 透明交换

02h = 切换协议

其它 = RFU

应答格式

响应数据域	SW1	SW2
编码的数据域 BER-TLV		

每个命令都会返回 SW1 和 SW2 加上响应数据域（如有）。SW1 和 SW2 基于 ISO 7816 的规定。下述 C0 数据对象的 SW1 SW2 也要用到。

C0 数据元格式

标签	长度（1 字节）	SW2
C0h	03h	错误状态

错误状态说明

错误状态	说明
XX SW1 SW2	XX = APDU 中不良数据对象的数量 00 = APDU 常见错误 01 = 第 1 个数据对象有错误 02 = 第 2 个数据对象有错误
00 90 00h	未发生错误
XX 62 82h	数据对象 XX 告警，请求信息不存在
XX 63 00h	未有信息
XX 63 01h	由于其它数据对象失败，停止执行

错误状态	说明
XX 6A 81h	不支持数据对象 XX
XX 67 00h	意外长度的数据对象 XX
XX 6A 80h	意外值的数据对象 XX
XX 64 00h	数据对象 XX 执行错误（没有 IFD 响应）
XX 64 01h	数据对象 XX 执行错误（没有 ICC 响应)
XX 6F 00h	数据对象 XX 失败, 没有准确诊断

第一个字节的值表示数据对象 XX 的数量, 而最后两个字节是对错误的解释。允许使用 ISO 7816 规定的 SW1 SW2 值。

如果在 C-APDU 数据域中存在多个数据对象, 而且其中一个数据对象失败, 那么在其它数据对象不依赖于失败的数据对象的情况下, IFD 可以处理接下来的数据对象。

6.2.3.2. 会话管理命令（Manage Session Command）

此命令用于管理透明会话，包括开始和终止透明会话。您也可以通过此命令管理操作环境以及透明会话内 IFD 的功能。

会话管理命令

命令	CLA	INS	P1	P2	Lc	命令数据域
Manage Session	FFh	C2h	00h	00h	DataLen	数据对象（N 个字节）

其中：

数据对象（1 个字节）

标签	数据对象
80h	版本数据对象
81h	开始透明会话
82h	结束透明会话
83h	关闭 RF 场
84h	打开 RF 场
5F 46h	计时器
FF 6Dh	获取参数
FF 6Eh	设置参数

管理会话响应数据对象

标签	数据对象
C0h	常见错误状态
80h	版本数据对象
FF 6Dh	IFD 参数数据对象

6.2.3.2.1. 开始会话数据对象（Start Session Data Object）

此命令用于开始透明会话。会话开始后，自动轮询功能将被禁用，直到会话结束。

开始会话数据对象

标签	长度（1 字节）	值
81h	00h	-

6.2.3.2.2. 终止会话数据对象（End Session Data Object）

此命令用于终止透明会话。在新的会话开始之前，重置为自动轮询状态。

终止会话数据对象

标签	长度（1 字节）	值
82h	00h	-

6.2.3.2.3. 版本数据对象（Version Data Object）

此命令用于返回 IFD 处理程序的版本号。

版本数据对象

标签	长度（1 字节）	值		
80h	03h	主版本	次版本	内部版本

6.2.3.2.4. 关闭 RF 数据对象（Turn Off the RF Data Object）

此命令用于关闭天线场。

关闭 RF 场数据对象

标签	长度（1 字节）	值
83h	00h	-

6.2.3.2.5. 开启 RF 数据对象（Turn On the RF Data Object）

此命令用于开启天线场。

打开 RF 场数据对象

标签	长度（1 字节）	值
84h	00h	-

6.2.3.2.6. 计时器数据对象（Timer Data Object）

此命令用于创建一个 32 位计时器数据对象，以 1 µs 为单位。

例如：如果在关闭 RF 数据对象和开启 RF 数据对象之间有 5000 µs 的计时器数据对象，读写器会关闭 RF 场大约 5000µs，然后再开启 RF 场。

计时器数据对象

标签	长度（1 字节）	值
5F 46h	04h	计时器（4 个字节）

6.2.3.2.7. 获取参数数据对像（Get Parameter Data Object）

此命令用于从 IFD 中获取各种参数。

获取参数数据对象

标签	长度（1 字节）	值		
		标签	长度	值
FF 6Dh	变长	TLV_Objects		

TLV_Objects

所需参数	标签	长度
IFD 帧长度整数（FSDI）	01h	00h
ICC 帧长度整数（FSCI）	02h	00h
帧等待时间整数（FWTI）	03h	00h
IFD 支持的最高通信速度	04h	00h
ICC 通信速度	05h	00h
调制指数	06h	00h
符合 ISO/IEC14443 的 PCB	07h	00h
符合 ISO/IEC14443 的 CID	08h	00h
符合 ISO/IEC14443 的 NAD	09h	00h
ISO/IEC14443 B 类的参数 1 - 4	0Ah	00h

6.2.3.2.8. 设置参数数据对象（Set Parameter Data Object）

此命令用于设置 IFD 的各种参数。

设置参数数据对象

标签	长度（1 字节）	值		
		标签	长度	值
FF 6Eh	变长	TLV_Objects		

TLV_Objects

所需参数	标签	长度
IFD 帧长度整数（FSDI）	01h	01h
ICC 帧长度整数（FSCI）	02h	01h
帧等待时间整数（FWTI）	03h	01h
IFD 支持的最高通信速度	04h	01h
ICC 通信速度	05h	01h
调制指数	06h	01h
符合 ISO/IEC14443 的 PCB	07h	01h
符合 ISO/IEC14443 的 CID	08h	01h
符合 ISO/IEC14443 的 NAD	09h	01h
ISO/IEC14443 B 类的参数 1 - 4	0Ah	04h

6.2.3.3. 透明交换命令（Transparent Exchange Command）

此命令用于发送和接收来自 ICC 的任何位或字节。

透明交换命令

命令	CLA	INS	P1	P2	Lc	命令数据域
TranspEx	FFh	C2h	00h	01h	DataLen	数据对象（N 个字节）

其中：

数据对象（1 个字节）

标签	数据对象
90h	发送和接收标志
91h	发送位成帧
92h	接收位成帧
93h	发送
94h	接收
95h	收发 - 发送和接收
FF 6Dh	获取参数
FF 6Eh	设置参数

透明交换会话响应数据对象

标签	数据对象
C0h	常见错误状态
92h	所接收数据中最后一个字节的有效位数
96h	响应报文状态字
97h	ICC 响应
FF 6Dh	IFD 参数数据对象

6.2.3.3.1. 发送和接收标志数据对象（Transmission and Reception Flag Data Object）

此命令用于为下列传输定义成帧参数和 RF 参数。

发送和接收标志数据对象

标签	长度（1 字节）	值	
		位	说明
90h	02h	0	0 – 在传输的数据后添加 CRC 1 – 不在传输的数据后添加 CRC
		1	0 – 丢弃接收数据中的 CRC 1 – 不丢弃接收数据中的 CRC（即不进行 CRC 检查）
		2	0 – 在传输的数据中插入奇偶校验位 1 – 不插入奇偶校验位
		3	0 – 期望接收的数据中含有奇偶校验位 1 – 不期望接收的数据中含有奇偶校验位（即不进行奇偶校验）
		4	0 – 在传输数据中添加协议头, 或者从响应中丢弃 1 – 不添加或者丢弃协议头（如有）（例如 PCB, CID, NAD）
		5-15	RFU

6.2.3.3.2. 发送位成帧数据对象（Transmission Bit Framing Data Object）

此命令用于定义待发送或待收发数据中最后一个字节的有效位数量。

发送位成帧数据对象

标签	长度（1 字节）	值	
		位	说明
91h	01h	0-2	最后一个字节中的有效位数量（0 表示所有的位都有效）
		3-7	RFU

发送位成帧数据对象只能和“发送”或“收发”数据对象一起使用。如果不存在此数据对象，则表明所有的位都有效。

6.2.3.3.3. 接收位成帧数据对象（Reception bit Framing Data Object）

在命令 APDU 中, 此数据对象定义接收到的数据中最后一个字节的预期有效位数量。

在响应 APDU 中, 此数据对象告知接收到的数据中最后一个字节的有效位数量。

接收位成帧数据对象

标签	长度（1 字节）	值	
		位	说明
92h	01h	0-2	最后一个字节中的有效位数量（0 表示所有的位都有效）
		3-7	RFU

如果不存在此数据对象, 则表明所有的位都有效。

6.2.3.3.4. 发送数据对象（Transmit Data Object）

此命令用于将数据从 IFD 发送至 ICC。完成传输后, 不期待收到 ICC 的响应。

发送数据对象

标签	长度（1 字节）	值
93h	DataLen	数据（N 个字节）

6.2.3.3.5. 接收数据对象（Receive Data Object）

此命令用于强制读写器在下述计时器对象规定的时间段内进入接收模式。

接收数据对象

标签	长度（1 字节）	值
94h	00h	-

6.2.3.3.6. 收发数据对象（Transceive Data Object）

此命令用于发送和接收来自 ICC 的数据。数据发送完成后, 读写器会保持等待状态, 直到计时器数据对象规定的时间结束。

如果没有在数据域中定义计时器数据对象, 读写器会保持等待状态直到设置参数 FWTI 数据对象规定的时间段结束。如果没有设置 FWTI, 读写器会等待大约 302 µs。

收发数据对象

标签	长度（1 字节）	值
95h	DataLen	数据（N 个字节）

6.2.3.3.7. 响应状态数据对象（Response Status Data Object）

在响应中, 此命令用于提示接收到的数据状态。

响应状态数据对象

标签	长度（1 字节）	值		
		字节 0		字节 1
		位	说明	
96h	02h	0	0 – CRC 正确, 或未进行校验 1 – CRC 校验失败	如果检测到冲突, 这些字节会说明冲突的位置。否则显示“00h”。
		1	0 – 无冲突 1 – 检测到冲突	
		2	0 – 无奇偶校验位错误 1 – 检测到奇偶校验位错误	
		3	0 – 无成帧错误 1 – 检测到成帧错误	
		4 - 7	RFU	

6.2.3.3.8. 响应数据对象 (Response Data Object)

在响应中, 此命令用于提示接收到的数据状态。

响应数据对象

标签	长度（1 字节）	值
97h	DataLen	响应数据 (N 字节)

6.2.3.4. 切换协议命令（Switch Protocol Command）

此命令用于指定透明会话中的协议和不同标准层。

切换协议命令

命令	CLA	INS	P1	P2	Lc	命令数据域
SwProtocol	FFh	C2h	00h	02h	DataLen	数据对象（N 个字节）

其中：

数据对象（1 个字节）

标签	数据对象
8Fh	切换协议数据对象
FF 6Dh	获取参数
FF 6Eh	设置参数

切换协议响应数据对象

标签	数据对象
C0h	常见错误状态
FF 6Dh	IFD 参数数据对象

6.2.3.4.1. 切换协议数据对象（Switch Protocol Data Object）

此命令用于指定协议和不同标准层。

切换协议数据对象

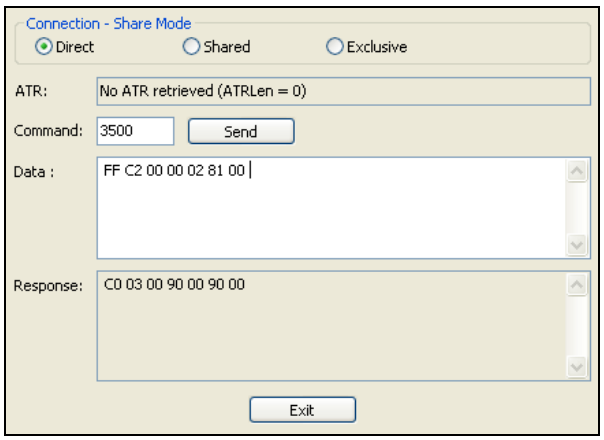
标签	长度（1 字节）	值	
		字节 0	字节 1
8Fh	02h	00h – ISO/IEC14443 Type A 01h – ISO/IEC14443 Type B 03h – FeliCa 其它 – RFU	00h – 如果没有分层 02h – 切换到第二层 03h – 切换并激活到第三层 04h – 激活到第四层 其它 - RFU

6.2.3.5. PC/SC 2.0 第 3 部分示例

1. 开始透明会话

   命令：**FF C2 00 00 02 81 00**

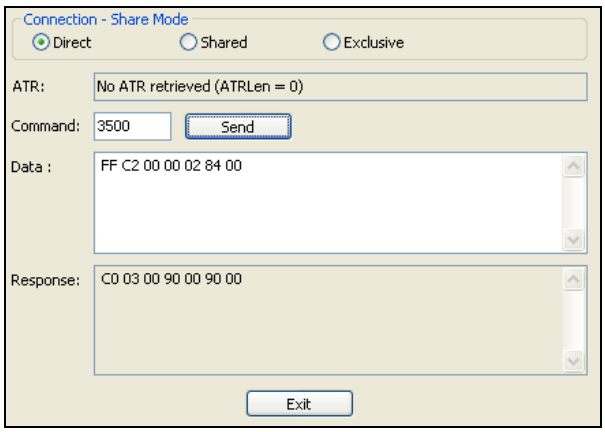
   响应：**C0 03 00 90 00 90 00**



2. 打开天线场

   命令：**FF C2 00 00 02 84 00**

   响应：**C0 03 00 90 00 90 00**

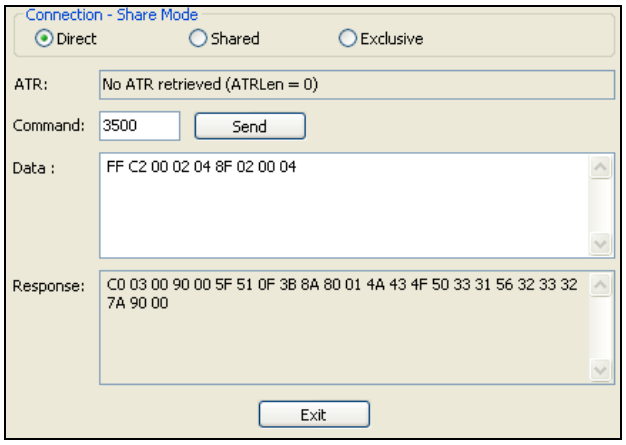


3. ISO 14443-4A 有效。

   命令： **FF C2 00 02 04 8F 02 00 04**

   响应： **C0 03 01 64 01 90 00** (如果卡不存在）

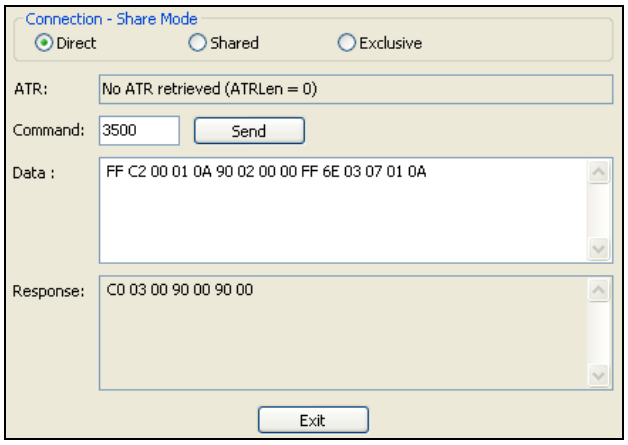
   C0 03 00 90 00 5F 51 [ATR] 90 00



4. 将 PCB 设为 0Ah, 并在传输数据中启用 CRC, 奇偶校验和协议头。

   命令：**FF C2 00 01 0A 90 02 00 00 FF 6E 03 07 01 0A**

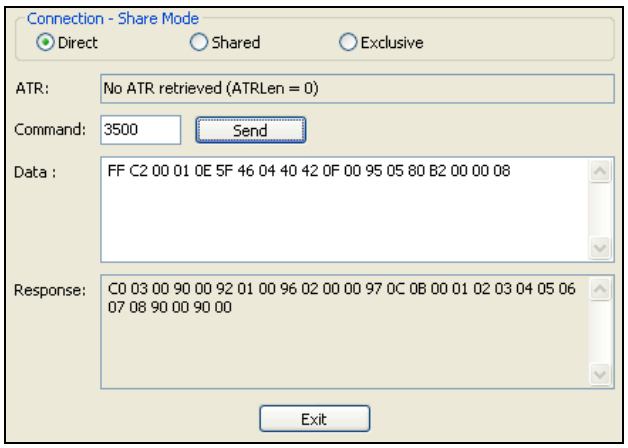
   响应：**C0 03 00 90 00 90 00**



5. 发送 APDU “80B2000008”至卡片并取响应。

   命令：**FF C2 00 01 0E 5F 46 04 40 42 0F 00 95 05 80 B2 00 00 08**

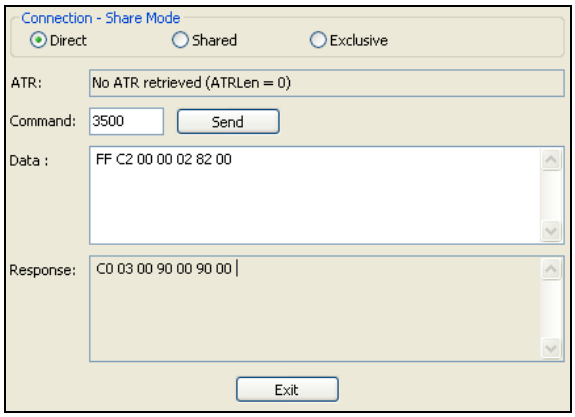
   响应：**C0 03 00 90 00 92 01 00 96 02 00 00 97 0C [卡片响应] 90 00**



6. 结束透明会话。

   命令：**FF C2 00 00 02 82 00**

   响应：**C0 03 00 90 00 90 00**



6.2.4. MIFARE Classic (1K/4K)存储卡的 PICC 命令

6.2.4.1. 加载认证密钥（Load Authentication Keys）

此命令用于向读写器加载认证密钥。该认证密钥用于验证 MIFARE Classic (1K/4K)存储卡的特定扇区。

Load Authentication Keys 的 APDU 结构（11 个字节）

命令	CLA	INS	P1	P2	Lc	命令数据域
Load Authentication Keys	FFh	82h	密钥结构	密钥号	06h	密钥（6 字节）

其中：

密钥结构 1 字节。

00h = 密钥被载入读写器的易失存储器。

其它 = 保留。

密钥号 1 字节。

00h ~ 01h =用于临时存储密钥的易失存储器。一旦读写器与电脑断开连接，密钥就会消失。易失密钥有两个，可以用作不同会话的过程密钥。

注：*默认值是 FF FF FF FF FF FFh.*

密钥 6 个字节。

载入读写器的密钥值。例：FF FF FF FF FF FFh

Load Authentication Keys 的响应结构（2 字节）

响应	响应数据域	
结果	SW1	SW2

Load Authentication Keys 命令的响应状态码

结果	SW1	SW2	含义
成功	90h	00h	操作成功完成。
错误	63h	00h	操作失败。

例：

// 向易失存储器位置 00h 加载密钥 {FF FF FF FF FF FFh}。

APDU = {FF 82 00 00 06 FF FF FF FF FF FFh}

6.2.4.2. MIFARE Classic (1K/4K)卡认证（Authentication for MIFARE Classic (1K/4K）

此命令使用存储在读写器内的密钥来验证 MIFARE Classic（1K/4K）卡（PICC）。涉及两种认证密钥：TYPE_A 和 TYPE_B。

Load Authentication Keys 的 APDU 结构（6 个字节）[弃用]

命令	CLA	INS	P1	P2	P3	命令数据域
Authentication	FFh	88h	00h	块号	密钥类型	密钥号

Load Authentication Keys 的 APDU 结构（10 个字节）

命令	CLA	INS	P1	P2	Lc	命令数据域
Authentication	FFh	86h	00h	00h	05h	认证数据字节

认证数据字节（5 个字节）

字节 1	字节 2	字节 3	字节 4	字节 5
版本 01h	00h	块号	密钥类型	密钥号

其中：

块号 1 字节。待验证的存储块。

MIFARE Classic 1K 卡的内存划分为 16 个扇区，每个扇区包含 4 个连续的块。（例如：扇区 00h 包含块{00h, 01h, 02h 和 03h}；扇区 01h 包含块{04h, 05h, 06h 和 07h}；最后一个扇区 0Fh 包含块{3Ch, 3Dh, 3Eh 和 3Fh}。验证通过后，读取同一个扇区内的其他块无需再次验证。详情请参考 MIFARE Classic 1K/4K 卡标准。

注： *一旦该块被成功验证，即可访问属于同一扇区的所有块。*

秘钥类型 1 字节。

60h = 该密钥被用作 TYPE A 密钥进行验证

61h = 该密钥被用作 TYPE B 密钥进行验证

秘钥号 1 字节。

00 ~ 01h = 用于存储密钥的易失存储器。一旦读写器与电脑断开连接，密钥就会消失。易失密钥有两个，可以用作不同会话的过程密钥。

Load Authentication Keys 的响应结构（2 字节）

响应	响应数据域	
结果	SW1	SW2

Load Authentication Keys 命令的响应状态码

结果	SW1	SW2	含义
成功	90	00h	操作成功完成。
错误	63	00h	操作失败。

扇区 (共 16 个扇区, 每个扇区包含 4 个连续的块)	数据块 (3 个块, 每块 16 个字节)	尾部块 (1 个块, 16 个字节)
扇区 0	00h – 02h	03h
扇区 1	04h – 06h	07h
..	..	..
..	..	..
扇区 14	38h – 0Ah	3Bh
扇区 15	3Ch – 3Eh	3Fh

1 KB

表10 ： MIFARE Classic 1K 卡的内存结构

扇区 (共 32 个扇区, 每个扇区包含 4 个连续的块)	数据块 (3 个块, 每块 16 个字节)	尾部块 (1 个块, 16 个字节)
扇区 0	00h – 02h	03h
扇区 1	04h – 06h	07h
..	..	..
..	..	..
扇区 30	78h – 7Ah	7Bh
扇区 31	7Ch – 7Eh	7Fh

2 KB

扇区(共 8 个扇区, 每个扇区包含 16 个连续的块)	数据块(15 个块, 每块 16 个字节)	尾部块(1 个块, 16 个字节)
扇区 32	80h – 8Eh	8Fh
扇区 33	90h – 9Eh	9Fh
..	..	..
..	..	..
扇区 38	E0h – EEh	EFh
扇区 39	F0h – FEh	FFh

2 KB

表11 ： MIFARE Classic 4K 卡的内存结构

字节号	0	1	2	3	页
序列号	SN0	SN1	SN2	BCC0	0
序列号	SN3	SN4	SN5	SN6	1
内部/锁	BCC1	Internal	Lock0	Lock1	2
OTP	OPT0	OPT1	OTP2	OTP3	3
数据读/写	Data0	Data1	Data2	Data3	4
数据读/写	Data4	Data5	Data6	Data7	5
数据读/写	Data8	Data9	Data10	Data11	6
数据读/写	Data12	Data13	Data14	Data15	7
数据读/写	Data16	Data17	Data18	Data19	8
数据读/写	Data20	Data21	Data22	Data23	9
数据读/写	Data24	Data25	Data26	Data27	10
数据读/写	Data28	Data29	Data30	Data31	11
数据读/写	Data32	Data33	Data34	Data35	12
数据读/写	Data36	Data37	Data38	Data39	13
数据读/写	Data40	Data41	Data42	Data43	14
数据读/写	Data44	Data45	Data46	Data47	15

512 位
或
64 字节

表12 ：MIFARE Ultralight® 卡的内存结构

例如：

//要使用{TYPE A, 密钥号 00h}验证块 04h。PC/SC V2.01, 弃用

APDU = {FF 88 00 04 60 00h};

// 要使用{TYPE A, key number 00h}验证块 04h。PC/SC V2.07

APDU = {FF 86 00 00 05 01 00 04 60 00h}

注： *MIFARE Ultralight 无需验证, 其内存可以自由访问。*

6.2.4.3. 读二进制块（Read Binary Blocks）

此命令用于从 PICC 卡片中取回多个“数据块”。执行本命令前, 必须先对数据块/尾部块进行验证。

Read Binary 的 APDU 结构（5 字节）

命令	CLA	INS	P1	P2	Le
Read Binary Blocks	FFh	B0h	00h	块号	待读取的字节数

其中：

块号 1 字节。起始块。

待读取的字节数 1 个字节。

MIFARE Classic 1K/4K 卡的待读字节的长度应该是 16 字节的倍数；MIFARE Ultralight® 卡应该是 4 字节的倍数。

MIFARE Ultralight® 卡的待读字节数最大为 16。

MIFARE Classic 1K 卡的待读字节数最大为 48。（多块模式；3 个连续的块）

MIFARE Classic 4K 卡的待读字节数最大为 240。（多块模式；15 个连续的块）

例 1： 10h（16 个字节）。仅起始块。（单块模式）

例 2： 40h（64 个字节）。从起始块至起始+3 块。（多块模式）

注： *出于安全原因, 多块模式仅用于读写数据块。尾部块不能在多块模式下读写, 请使用单块模式对其进行读写。*

Read Binary Block 的响应结构（4/16 的倍数 + 2 字节）

响应	响应数据域		
结果	数据（4/16 字节的倍数）	SW1	SW2

Read Binary Block 命令的响应状态码

结果	SW1	SW2	含义
成功	90h	00h	操作成功完成。
错误	63h	00h	操作失败。

例如：

// 从二进制块 04h 中读取 16 字节（MIFARE Classic 1K/4K）

APDU = FF B0 00 04 10h

// 从二进制块 80h 开始读取 240 字节（MIFARE Classic 4K）

// 块 80h 至块 8Eh（15 个块）

APDU = FF B0 00 80 F0h

6.2.4.4. 更新二进制块（Update Binary Blocks）

此命令用于向 PICC 卡写入多个“数据块”。执行本命令前，必须先对数据块/尾部块进行验证。

Update Binary 的 APDU 结构（16 的倍数 + 5 字节）

命令	CLA	INS	P1	P2	Lc	命令数据域
Update Binary Blocks	FFh	D6h	00h	块号	待更新的字节数	块数据（16 字节的倍数）

其中：

块号 1 字节。待更新的起始块

待更新的字节数 1 字节。

- MIFARE Classic 1K/4K 卡的待更新字节数是 16 字节的倍数；MIFARE Ultralight 卡是 4 字节的倍数。
- MIFARE Classic 1K 卡的待更新字节数最大为 48。（多块模式；3 个连续的块）
- MIFARE Classic 4K 卡的待更新字节数最大为 240。（多块模式；15 个连续的块）

块数据 16 字节的倍数 + 2 字节，或 6 字节。待写入二进制块的数据。

例 1： 10h（16 个字节）。仅起始块。（单块模式）

例 2： 30h（48 个字节）。从起始块至起始+2 块。（多块模式）

注：*出于安全原因，多块模式仅用于读写数据块。尾部块不能在多块模式下被读写，请使用单块模式对其进行读写。*

Update Binary Block 的响应码（2 字节）

结果	SW1	SW2	含义
成功	90h	00h	操作成功完成。
错误	63h	00h	操作失败。

例如：

// 将 MIFARE Classic 1K/4K 卡中的二进制块 04h 的数据更新为{00 01 ..0Fh}

APDU = FF D6 00 04 10 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0Fh

// 将 MIFARE Ultralight 卡中的二进制块 04h 的数据更新为{00 01 02 03h}

APDU = FF D6 00 04 04 00 01 02 03h

6.2.4.5. 值块操作命令(增加，减少，存储) [Value Block Operation (INC, DEC, STORE)]

此命令用于对基于数值的交易进行操作（例如：增加值块的值）。

Value Block Operation 的 APDU 结构（10 字节）

命令	CLA	INS	P1	P2	Lc	命令数据域	
Value Block Operation	FFh	D7h	00h	块号	05h	VB_OP	VB_Value（4 字节）{MSB ..LSB}

其中：

块号 1 字节。待操作的值块。

VB_OP 1 字节。

00h = 将 VB_Value 存入该块, 然后该块变为一个值块。

01h = 使值块的值增加 VB_Value。此命令仅适用于对值块的操作。

02h = 使值块的值减少 VB_Value。此命令仅适用于对值块的操作。

VB_Value 4 字节。用于算数运算的数值, 是一个有符号长整数（4 个字节）。

例 1： Decimal –4 = {FFh, FFh, FFh, FCh}

VB_Value			
MSB			LSB
FFh	FFh	FFh	FCh

例 2： Decimal 1 = {00h, 00h, 00h, 01h}

VB_Value			
MSB			LSB
00h	00h	00h	01h

Value Block Operation 的响应结构（2 字节）

响应	响应数据域	
结果	SW1	SW2

Value Block Operation 响应码

结果	SW1	SW2	含义
成功	90h	00h	操作成功完成。
错误	63h	00h	操作失败。

6.2.4.6. 读值块（Read Value Block）

此命令用于获取值块中的数值, 此命令仅适用于对值块的操作。

Read Value Block 的 APDU 结构（5 字节）

命令	CLA	INS	P1	P2	Le
Read Value Block	FFh	B1h	00h	块号	04h

其中：

块号 1 字节。待读写的值块。

Read Value Block 的响应结构（4 + 2 字节）

响应	响应数据域		
结果	值 {MSB ..LSB}	SW1	SW2

其中：

值 4 字节。卡片返回的数值, 是一个有符号长整数（4 个字节）。

例 1： Decimal –4 = {FFh, FFh, FFh, FCh}

值			
MSB			**LSB**
FFh	FFh	FFh	FCh

例 2： Decimal 1 = {00h, 00h, 00h, 01h}

值			
MSB			**LSB**
00h	00h	00h	01h

Read Value Block 命令的响应状态码

结果	SW1	SW2	含义
成功	90h	00h	操作成功完成。
错误	63h	00h	操作失败。

6.2.4.7. 复制值块（Copy Value Block）

此命令用于将一个值块中的数值复制到另外一个值块。

Copy Value Block 的 APDU 结构（7 字节）

命令	CLA	INS	P1	P2	Lc	命令数据域	
Copy Value Block	FFh	D7h	00h	源块号	02h	03h	目标块号

其中：

源块号 1 字节。源值块中的值会被复制到目标值块。

目标块号 1 字节。要恢复的值块。源值块和目标值块必须位于同一个扇区。

Copy Value Block 的响应结构（2 个字节）

响应	响应数据域	
结果	SW1	SW2

Copy Value Block 命令的响应状态码

结果	SW1	SW2	含义
成功	90h	00h	操作成功完成。
错误	63h	00h	操作失败。

例如：

// 将数值“1”存入块 05h

APDU = FF D7 00 05 05 00 00 00 00 01h

// 读取值块 05h

APDU = FF B1 00 05 04h

//将数值从值块 05h 复制到值块 06h

APDU = FF D7 00 05 02 03 06h

// 使值块 05h 的值增加“5”

APDU = FF D7 00 05 05 01 00 00 00 05h

6.2.5. 访问符合 PC/SC 标准的标签（ISO 14443-4）

所有符合 ISO 14443-4 标准的卡片（PICC）都理解符合 ISO 7816-4 规定的 APDU。ACR1255U-J1 读写器与符合 ISO 14443-4 标准的卡片进行通信时，只需要对 ISO 7816-4 规定的 APDU 和响应进行转换。ACR1255U-J1 会在内部处理 ISO 14443 第 1-4 部分协议。

另外 MIFARE Classic® (1K/4K), MIFARE® Mini 和 MIFARE Ultralight® 标签是通过 T=CL 模拟进行支持的。只要将 MIFARE 标签视作标准的 ISO 14443-4 标签即可。更多信息请参阅 **MIFARE Classic (1K/4K)存储卡的 PICC 命令**。

ISO 7816-4 规定的 APDU 报文结构

命令	CLA	INS	P1	P2	Lc	命令数据域	Le
ISO 7816 第 4 部分规定的命令					命令数据域的长度		期望返回的响应数据的长度

ISO 7816-4 规定的响应报文的结构（数据 + 2 字节）

响应	响应数据域		
结果	响应数据	SW1	SW2

通用的 ISO 7816-4 命令的响应状态码

结果	SW1	SW2	含义
成功	90h	00h	操作成功完成。
错误	63h	00h	操作失败。

典型的操作顺序为：

1. 出示标签, 与 PICC 接口建立连接。
2. 读取/更新标签的存储内容。

要实现这些, 需要：

1. 与标签建立连接。

   标签的 ATR 为 3B 88 80 01 00 00 00 00 33 81 81 00 3Ah.

   其中：

   ATQB 的应用数据 = 00 00 00 00

   ATQB 的协议信息 = 33 81 81。

   这是一个 ISO 14443-4 Type B 标签。

2. 发送 APDU, 取随机数。

   << 00 84 00 00 08h

   >> 1A F7 F3 1B CD 2B A9 58h [90 00h]

   注：*对于 ISO 14443-4 Type A 标签来说, 可以通过 APDU"FF CA 01 00 00h"来获取 ATS。*

例：

// 从 ISO 14443-4 Type B PICC（ST19XR08E）中读取 8 字节

APDU = 80 B2 80 00 08h

CLA = 80h

INS = B2h

P1 = 80h

P2 = 00h

Lc = 无

命令数据域 = 无

Le = 08h

应答：00 01 02 03 04 05 06 07h [$9000h]

6.2.6. 读写 MIFARE DESFire 标签(ISO 14443-3)

MIFARE® DESFire® 支持 ISO7816-4 APDU 包模式和本地模式。一旦 MIFARE® DESFire® 标签被激活，发送至 MIFARE® DESFire® 标签的第一个 APDU 就会确定“命令的模式”。如果第一个 APDU 采用“本地模式”，则其余的 APDU 命令都必须是“本地模式”。同样，如果第一个 APDU 采用“ISO 7816-4 APDU 包模式”，则其余的 APUDU 都必须是“ISO 7816-4 APDU 包模式”。

例 1：MIFARE® DESFire® ISO 7816-4 APDU 包。

//从 ISO 14443-4 Type A PICC（DESFIRE）中读取 8 个字节的随机数

APDU = {90 0A 00 00 01 00 00}

CLA = 90h; INS = 0Ah (DESFire Instruction); P1 = 00h; P2 = 00h

Lc = 01h; Data In = 00h; Le = 00h (Le = 00h 表示最大长度)

应答：7B 18 92 9D 9A 25 05 21 [$91AF]

状态码{91 AF}由 DESFIRE 标准定义，详情请参阅 DESFIRE 标准。

例 2：MIFARE® DESFire® 分页链接（ISO 7816 APDU 包模式）

// 在本例中，应用涉及到"分页链接"。

// 要获得 DESFIRE 卡的版本号：

步骤 1：发送 APDU{90 60 00 00 00}来获取第一个数据页。INS=60h

应答：04 01 01 00 02 18 05 91 AF [$91AF]

步骤 2：发送 APDU {90 AF 00 00 00}来获取第二个数据页。INS=AFh

应答：04 01 01 00 06 18 05 91 AF [$91AF]

步骤 3：发送 APDU {90 AF 00 00 00} 来获取最后一个数据页。INS=AFh

应答：04 52 5A 19 B2 1B 80 8E 36 54 4D 40 26 04 91 00 [$9100]

6.2.7. 访问 FeliCa 标签

访问 FeliCa 标签的命令与访问 PCSC 标签和 MIFARE 标签的命令有所不同。命令符合 FeliCa 规范，增加一个命令头。

FeliCa 命令结构

命令	CLA	INS	P1	P2	Lc	命令数据域
FeliCa 命令	FFh	00h	00h	00h	命令数据域的长度	FeliCa 命令 (开始于长度字节)

FeliCa 的响应结构 (数据 + 2 个字节)

响应	响应数据域
结果	响应数据

以读取内存块为例：

1. 连接 FeliCa。

   ATR = 3B 8F 80 01 80 4F 0C A0 00 00 03 06 11 00 3B 00 00 00 00 42h

   其中 ： 11 00 3Bh = FeliCa

2. 读取 FeliCa IDM.

   CMD = FF CA 00 00 00h

   RES = [IDM (8bytes)] 90 00h

   例如 FeliCa IDM = 01 01 06 01 CB 09 57 03h

3. FeliCa 命令访问。

   例如：“读取”内存块。

   CMD = FF 00 00 00 10 10 06 01 01 06 01 CB 09 57 03 01 09 01 01 80 00h

   其中：

   Felica 命令 = 10 06 01 01 06 01 CB 09 57 03 01 09 01 01 80 00h

   IDM = 01 01 06 01 CB 09 57 03h

   RES = 内存块数据

6.3. 外设控制

读写器的外设控制命令在蓝牙模式下采用 Escape 命令（0x6B），在 USB 模式下采用 PC_to_RDR_Escape 来实现。

在 PC 连机模式下：

命令格式

偏移	数据域	大小	值	说明
0	*bMessageType*	1	6Bh	-
1	*dwLength*	4	-	此消息的 abData 数据域的大小
5	*bSlot*	1	-	标识命令的插槽号
6	*bSeq*	1	-	命令的序号
7	*abRFU*	3	-	保留为将来使用
10	*abData*	字节型数组	-	送至 CCID 的数据块

其中 *abData* 是指命令

在蓝牙模式下：

命令格式

偏移	数据域	大小	值	说明
0	*CmdMessageType*	1	6Bh	-
1	*Length*	2	-	数据长度
3	*Slot*	1	00h	-
4	*Seq*	1	00h	序号
5	*Param*	1	00h	参数
6	*Checksum*			Checksum 是指命令中所有字节的 XOR 值
7	*Data*	N	-	数据

其中 *Data* 是指命令

6.3.1. 获取固件版本号（Get Firmware Version）

此命令用于获取读写器的固件信息。

Get Firmware Version 的命令结构（5 字节）

命令	CLA	INS	P1	P2	Lc
Get Firmware Version	E0h	00h	00h	18h	00h

Get Firmware Version 的响应结构（5 字节 + 固件信息的长度）

响应	CLA	INS	P1	P2	Le	响应数据域
结果	E1h	00h	00h	00h	待接收的字节数	固件版本号

例：

响应 = E1 00 00 00 14 41 43 52 31 32 35 35 55 2D 4A 31 20 53 57 56 20 31 2E 30 35

固件版本号（HEX） = 41 43 52 31 32 35 35 55 2D 4A 31 20 53 57 56 20 31 2E 30 35

固件版本号（ASCII） = “ACR1255U-J1 SWV 1.05”

6.3.2. 获取序列号（Get Serial Number）

此命令用于获取读写器的序列号。

Get Serial Number 的命令结构（5 字节）

命令	CLA	INS	P1	P2	Lc
Get Serial Number	E0h	00h	00h	47h	00h

Get Serial Number 的响应结构（5 字节）

响应	CLA	INS	P1	P2	Le	响应数据域
结果	E1h	00h	00h	00h	待接收的字节数	序列号

例：

响应 = E1 00 00 00 0C 52 52 34 33 31 2D 30 30 30 30 31 36

序列号(HEX) = 52 52 34 33 31 2D 30 30 30 30 31 36

序列号(ASCII) = “RR431-000016”

6.3.3. LED 控制（LED Control）

此命令用于控制 LED 的输出。

LED Control 命令的结构（6 字节）

命令	CLA	INS	P1	P2	Lc	命令数据域
LED Control	E0h	00h	00h	29h	01h	LED 状态

LED Control 命令的响应结构（6 字节）

响应	CLA	INS	P1	P2	Le	响应数据域
结果	E1h	00h	00h	00h	01h	LED 状态

LED 状态（1 字节）

LED 状态	LED	颜色	说明
Bit 0	1	绿色	1 = 开；0 = 关
Bit 1	1	红色	1 = 开；0 = 关
Bit 2	2	蓝色	1 = 开；0 = 关
Bit 3	2	红色	1 = 开；0 = 关
Bit 4 - 7		RFU	RFU

6.3.4. LED 状态（LED Status）

此命令用于检查当前 LED 的状态。

LED Status 命令的结构（5 字节）

命令	CLA	INS	P1	P2	Lc
LED Status	E0h	00h	00h	29h	00h

LED Status 的响应结构（6 字节）

响应	CLA	INS	P1	P2	Le	响应数据域
结果	E1h	00h	00h	00h	01h	LED 状态

LED 状态（1 字节）

LED 状态	LED	颜色	说明
Bit 0	1	绿色	1 = 开；0 = 关
Bit 1	1	红色	1 = 开；0 = 关
Bit 2	2	蓝色	1 = 开；0 = 关
Bit 3	2	红色	1 = 开；0 = 关
Bit 4 - 7		RFU	RFU

6.3.5. 蜂鸣器控制（Buzzer Control）

此命令用于控制蜂鸣器的输出。

Buzzer Control 的命令结构（6 字节）

命令	CLA	INS	P1	P2	Lc	命令数据域
Buzzer Control	E0h	00h	00h	28h	01h	蜂鸣器鸣响时间

其中：

蜂鸣器鸣响时间 1 字节

01 - FFh = 持续时间（单位：10 ms）

Buzzer Control 的响应结构（6 字节）

响应	CLA	INS	P1	P2	Le	响应数据域
结果	E1h	00h	00h	00h	01h	00h

6.3.6. 设置 LED 和蜂鸣器状态指示器（Set LED and Buzzer Status Indicator Behavior）

此命令用于设置 LED 和蜂鸣器作为状态指示器的各种操作。

***注：** 该设置将被保存到非易失存储器。（从固件版本 2.03.xx 开始）*

Bit4 和 Bit5 操作选项只在 2.04.xx 及以上版本的固件中提供。

Set LED and Buzzer Status Indicator Behaviors 的 命令结构（6 字节）

命令	CLA	INS	P1	P2	Lc	命令数据域
Set LED and Buzzer Status Indicator Behavior	E0h	00h	00h	21h	01h	默认操作

操作（1 字节）

操作	模式	说明
Bit 0	电池充电状态 LED	显示电池充电状态 1 = 启用；0 = 禁用
Bit 1	PICC 轮询状态 LED	显示 PICC 轮询状态 1 = 启用；0 = 禁用
Bit 2	PICC 激活状态 LED	显示 PICC 接口的激活状态 1 = 启用；0 = 禁用
Bit 3	卡片插入事件蜂鸣器	每次检测到卡片插入发出哔的一声 1 = 启用；0 = 禁用
Bit 4	卡片移除事件蜂鸣器	每次检测到卡片移除发出哔的一声 1 = 启用；0 = 禁用 *（要使用此功能，必须先启用 Bit3）*
Bit 5	读写器上电蜂鸣器	读写器上电时发出哔的一声 1 = 启用；0 = 禁用
Bit 6	RFU	
Bit 7	卡片操作闪烁 LED	LED 在卡片读写时会亮起。

注：

***(1)** 操作的默认值 = BFh*

***(2)** USB 或蓝牙模式下，充电时红色 LED 指示灯常亮，除非充电状态设置处于关闭状态，或读写器已经充满电。*

***(3)** 蓝牙模式下，蓝色 LED 指示灯（LED 2）持续闪烁，不能更改。*

***(4)** USB 模式下，绿色 LED 指示灯（LED 2）常亮，除非轮询状态设置处于关闭状态。*

Set LED and Buzzer Status Indicator Behaviors 的响应结构（6 字节）

响应	CLA	INS	P1	P2	Le	响应数据域
结果	E1h	00h	00h	00h	01h	默认操作

6.3.7. 读取 LED 和蜂鸣器状态指示器（Read LED and Buzzer Status Indicator Behavior）

此命令用于读取 LED 和蜂鸣器的当前默认操作。

***注：** Bit4 和 Bit5 操作响应只在 2.04.xx 及以上版本的固件中提供。*

Read LED and Buzzer Status Indicator Behavior 的命令结构（5 字节）

命令	CLA	INS	P1	P2	Lc
Read LED and Buzzer Status Indicator Behaviors	E0h	00h	00h	21h	00h

Read LED and Buzzer Status Indicator Behavior 的响应结构（6 字节）

响应	CLA	INS	P1	P2	Le	响应数据域
结果	E1h	00h	00h	00h	01h	操作

操作（1 字节）

操作	模式	说明
Bit 0	电池充电状态 LED	显示电池充电状态 1 = 启用；0 = 禁用
Bit 1	PICC 轮询状态 LED	显示 PICC 轮询状态 1 = 启用；0 = 禁用
Bit 2	PICC 激活状态 LED	显示 PICC 接口的激活状态 1 = 启用；0 = 禁用
Bit 3	卡片插入事件蜂鸣器	每次检测到卡片插入发出哔的一声 1 = 启用；0 = 禁用
Bit 4	卡片移除事件蜂鸣器	每次检测到卡片移除发出哔的一声 1 = 启用；0 = 禁用 *（要使用此功能，必须先启用 Bit3）*
Bit 5	读写器上电蜂鸣器	读写器上电时发出哔的一声 1 = 启用；0 = 禁用
Bit 6	RFU	
Bit 7	卡片操作闪烁 LED	LED 在卡片读写时会亮起。

***注：** 操作的默认值 = BFh*

6.3.8. 设置自动 PICC 轮询（Set Automatic PICC Polling）

此命令用于设置读写器的轮询模式。

每当读写器连接到电脑的时候，读写器的 PICC 轮询功能就会启动 PICC 扫描，以确定 PICC 是否被放置于/移出了内置天线的范围。

我们可以发送命令来停用 PICC 轮询功能。该命令通过 PCSC Escape 命令接口发送。为了满足节能要求，PICC 闲置，或者找不到 PICC 的时候，我们提供了几种关闭天线场的特殊模式。在省电模式下，读写器会消耗更低的电能。

注：*该设置将被保存到非易失存储器。（从固件版本 2.03.xx 开始）*

Set Automatic PICC Polling 的命令结构（6 字节）

命令	CLA	INS	P1	P2	Lc	命令数据域
Set Automatic PICC Polling	E0h	00h	00h	23h	01h	轮询设置

Set Automatic PICC Polling 的响应结构（6 字节）

响应	CLA	INS	P1	P2	Le	响应数据域
结果	E1h	00h	00h	00h	01h	轮询设置

轮询设置（1 字节）

轮询设置	模式	说明
Bit 0	自动 PICC 轮询	1 = 启用；0 = 禁用
Bit 1	如果没有找到 PICC，关闭天线场	1 = 启用；0 = 禁用
Bit 2	如果 PICC 闲置，关闭天线场	1 = 启用；0 = 禁用
Bit 3	RFU	-
Bit 5 ..4	PICC 轮询间隔	<Bit 5 – Bit 4> <0 – 0> = 250 ms <0 – 1> = 500 ms <1 – 0> = 1000 ms <1 – 1> = 2500 ms
Bit 6	RFU	-
Bit 7	强制执行 ISO14443A 第 4 部分	1 = 启用；0 = 禁用

注：*轮询设置的默认值 = 8Bh。*

提示：

1. *建议启用"如果 PICC 闲置，关闭天线场"选项，这样闲置的 PICC 就不会一直暴露在天线场中，可以防止 PICC"发热"。*
2. *PICC 轮询间隔时间越长，节能效果越好。然而，PICC 轮询的响应时间也会增加。在节能状态下，空闲时的电流消耗约为 60 mA；而在非节能状态下，空闲时的电流消耗约为 130 mA。*

   注：*空闲时的电流消耗=PICC 尚未激活。*
3. *读写器会自动激活"ISO14443A-4 PICC"的 ISO 14443A-4 模式。B 类 PICC 不受此选项影响。*
4. *JCOP30 卡片有两种模式：ISO 14443A-3（MIFARE Classic 1K）和 ISO 14443A-4 模式。一旦 PICC 被激活，应用就必须选定一种模式。*

6.3.9. 读取自动 PICC 轮询（Read Automatic PICC Polling）

此命令用于检查当前的 PICC 轮询设置。

Read Automatic PICC Polling 的命令结构（5 字节）

命令	CLA	INS	P1	P2	Lc
Read Automatic PICC Polling	E0h	00h	00h	23h	00h

Read Automatic PICC Polling 的响应结构（6 字节）

响应	CLA	INS	P1	P2	Le	响应数据域
结果	E1h	00h	00h	00h	01h	轮询设置

轮询设置（1 字节）

轮询设置	模式	说明
Bit 0	自动 PICC 轮询	1 = 启用；0 = 禁用
Bit 1	如果没有找到 PICC, 关闭天线场	1 = 启用；0 = 禁用
Bit 2	如果 PICC 闲置, 关闭天线场	1 = 启用；0 = 禁用
Bit 3	RFU	-
Bit 5 ..4	PICC 轮询间隔	<Bit 5 – Bit 4> <0 – 0> = 250 ms <0 – 1> = 500 ms <1 – 0> = 1000 ms <1 – 1> = 2500 ms
Bit 6	RFU	-
Bit 7	强制执行 ISO14443A 第 4 部分	1 = 启用；0 = 禁用

注：*轮询设置的默认值 = 8Bh。*

6.3.10. 设置 PICC 操作参数（Set PICC Operating Parameter）

此命令用于设置 PICC 操作参数。

注：*该设置将被保存到非易失存储器。（从固件版本 2.03.xx 开始）*

Set the PICC Operating Parameter 的命令结构（6 字节）

命令	CLA	INS	P1	P2	Lc	命令数据域
Set the PICC Operating Parameter	E0h	00h	00h	20h	01h	操作参数

Set the PICC Operating Parameter 的响应结构（6 字节）

响应	CLA	INS	P1	P2	Le	响应数据域
结果	E1	00h	00h	00h	01h	操作参数

操作参数（1 字节）

操作参数	参数	说明	选项
Bit 0	ISO 14443 A 类	PICC 轮询要检测的标签类别	1 = 检测 0 = 跳过
Bit 1	ISO 14443 B 类		1 = 检测 0 = 跳过
Bit 2	FeliCa 212 Kbps		1 = 检测 0 = 跳过
Bit 3	FeliCa 424 Kbps		1 = 检测 0 = 跳过
Bit 4	Topaz		1 = 检测 0 = 跳过
Bit 5	Calypso		1 = 检测 0 = 跳过
Bit 6	SRIX		1 = 检测 0 = 跳过
Bit 7	RFU	RFU	RFU

注：*操作参数的默认值 = 7Fh.*

6.3.11. 读取 PICC 操作参数（Read PICC Operating Parameter）

此命令用于检查当前的 PICC 操作参数。

Read the PICC Operating Parameter 的命令结构（5 字节）

命令	CLA	INS	P1	P2	Lc
Read the PICC Operating Parameter	E0h	00h	00h	20h	00h

Read the PICC Operating Parameter 的响应结构（6 字节）

响应	CLA	INS	P1	P2	Le	响应数据域
结果	E1h	00h	00h	00h	01h	操作参数

操作参数（1 字节）

操作参数	参数	说明	选项
Bit 0	ISO 14443 A 类	PICC 轮询要检测的标签类别	1 = 检测 0 = 跳过
Bit 1	ISO 14443 B 类		1 = 检测 0 = 跳过
Bit 2	FeliCa		1 = 检测 0 = 跳过
Bit 3	FeliCa 424 Kbps		1 = 检测 0 = 跳过
Bit 4	Topaz		1 = 检测 0 = 跳过
Bit 5	Calypso		1 = 检测 0 = 跳过
Bit 6	SRIX		1 = 检测 0 = 跳过
Bit 7	RFU	RFU	RFU

注：*操作参数的默认值 = 7Fh.*

6.3.12. 设置自动 PPS（Set Auto PPS）

每次识别出 PICC, 读写器都会尝试更改由最大连接速度定义的 PCD 和 PICC 间的通信速率。若卡片不支持建议的连接速度, 读写器会尝试以较慢的速度与卡片建立连接。

注：*该设置将被保存到非易失存储器。（从固件版本 2.03.xx 开始）*

Set Auto PPS 的命令结构（7 个字节）

命令	CLA	INS	P1	P2	Lc	命令数据域	
Set Auto PPS	E0h	00h	00h	24h	02h	最大 Tx 速度	最大 Rx 速度

Set Auto PPS 命令的响应结构（9 个字节）

响应	CLA	INS	P1	P2	Le	响应数据域			
结果	E1h	00h	00h	00h	04h	最大 Tx 速度	当前 Tx 速度	最大 Rx 速度	当前 Rx 速度

其中：

最大 Tx 速度 1 个字节。最高传输速度。

最大 Rx 速度 1 个字节。最高接收速度

当前 Tx 速度 1 个字节。当前的传输速度。

当前 Rx 速度 1 个字节。当前的接收速度。

值可以为： 00h = 106 Kbps

01h = 212 Kbps

02h = 424 Kbps

注：*默认设置为 00h。*

注：

1. *通常来讲, 应用程序应当知道正在使用的 PICC 的最大连接速率, 周围环境也会对最大可达速率有所影响。读写器会使用建议的通信速率与 PICC 交换信息。如果 PICC 或周围环境不能满足所建议通信速率的要求, PICC 将无法访问。*
2. *读写器支持不同的数据发送速度和接收速度。*

6.3.13. 读取自动 PPS（Read Auto PPS）

此命令用于检查当前的自动 PPS 设置。

Read Auto PPS 的命令结构（5 个字节）

命令	CLA	INS	P1	P2	Lc
Read Auto PPS	E0h	00h	00h	24h	00h

Read Auto PPS 命令的响应结构（9 个字节）

响应	CLA	INS	P1	P2	Le	响应数据域			
结果	E1	00h	00h	00h	04h	最大 Tx 速度	当前 Tx 速度	最大 Rx 速度	当前 Rx 速度

其中：

最大 Tx 速度 1 个字节。最高传输速度。

最大 Rx 速度 1 个字节。最高接收速度

当前 Tx 速度 1 个字节。当前的传输速度。

当前 Rx 速度 1 个字节。当前的接收速度。

值可以为：00h = 106 Kbps
01h = 212 Kbps
02h = 424 Kbps

注：*默认设置为 00h。*

6.3.14. 天线场控制（Antenna Field Control）

此命令用于打开/关闭天线场。

Antenna Field Control 的命令结构（6 个字节）

命令	CLA	INS	P1	P2	Lc	命令数据域
Antenna Field Control	E0h	00h	00h	25h	01h	状态

Antenna Field Control 的响应结构（6 个字节）

响应	CLA	INS	P1	P2	Le	响应数据域
结果	E1h	00h	00h	00h	01h	状态

其中：

状态 1 个字节。

01h = 启用天线场

00h = 停用天线场

注：*关闭天线场前, 要确保自动 PICC 轮询功能已经停用。*

6.3.15. 读取天线场状态（Read Antenna Field Status）

此命令用于检查当前的天线场状态。

Read Antenna Field Status 的命令结构（5 个字节）

命令	CLA	INS	P1	P2	Lc
Read Antenna Field Status	E0h	00h	00h	25h	00h

Read Antenna Field Status 的响应结构（6 个字节）

响应	CLA	INS	P1	P2	Le	响应数据域
结果	E1h	00h	00h	00h	01h	状态

其中：

状态 1 个字节。

00h = PICC 关闭

01h = PICC 闲置 [准备好进行非接触式标签轮询, 但没有检测到此类标签]

02h = PICC 就绪 [PICC 请求（参阅 ISO 14443）成功, 也就是说检测到了非接触式标签]

03h = PICC 已选定 [PICC 选择（参阅 ISO 14443）成功]

04h = PICC 已激活 [PICC 激活（参阅 ISO 14443）成功, 准备好进行 APDU 交换]

6.3.16. 设置休眠模式选项（Set Sleep Mode Option）

默认情况下, 如果 60 秒内没有操作, 读卡器会自动进入休眠状态。

此命令用于设置设备进入休眠模式前的空闲时长。

注：*该设置将被保存到非易失存储器。*

Set Sleep Time Interval 的命令格式（5 个字节）

命令	CLA	INS	P1	P2	Lc
Set Sleep Time Interval	E0h	00h	00h	48h	时长

Set Sleep Time Interval 的响应格式（6 个字节）

响应	CLA	INS	P1	P2	Le	响应数据域
结果	E1h	00h	00h	00h	01h	时长

其中：

时长 1 个字节

00h = 60 秒

01h = 90 秒

02h = 120 秒

03h = 180 秒

04h = 禁用

6.3.17. 读取休眠模式选项（Read Sleep Mode Option）

此命令用于查看进入休眠模式前的空闲时长。

注：*仅适用于 2.03.xx 及以上版本的固件。*

Read Sleep Time Interval 的命令格式（5 个字节）

命令	CLA	INS	P1	P2	Lc
Read Sleep Time Interval	E0h	00h	00h	50h	00h

Read Sleep Time Interval 的响应格式（6 个字节）

响应	CLA	INS	P1	P2	Le	响应数据域
结果	E1h	00h	00h	00h	01h	Time

其中：

时长 1 个字节

00h = 60 秒

01h = 90 秒

02h = 120 秒

03h = 180 秒

04h = 无休眠模式

6.3.18. 修改 Tx 功率命令（Change Tx Power command）

此命令用于设置蓝牙传输功率。

注： *该设置将被保存到非易失存储器。（从固件版本 2.03.xx 开始）*

Change Tx Power 的命令格式（5 个字节）

命令	CLA	INS	P1	P2	Lc
Set Tx power	E0h	00h	00h	49h	Tx 功率

Change Tx Power 的响应格式（5 个字节）

响应	CLA	INS	P1	P2	Le	响应数据域
结果	E1h	00h	00h	00h	01h	Tx 功率

其中：

Tx 功率 1 个字节

00h = -23 dBm (默认), 距离： ~3 米

01h = -6 dBm, 距离：~7 米

02h = 0 dBm, 距离：~17 米

03h = 4 dBm, 距离：~25 米

6.3.19. 读取 Tx 功率值（Read Tx Power Value）

此命令用于查看蓝牙传输功率。

注：*仅适用于 2.03.xx 及以上版本的固件。*

Read Tx Power Value 的命令格式（5 个字节）

命令	CLA	INS	P1	P2	Lc
Set Tx power	E0h	00h	00h	51h	00h

Read Tx Power Value 的响应格式（6 个字节）

响应	CLA	INS	P1	P2	Le	响应数据域
结果	E1h	00h	00h	00h	01h	Tx 功率

其中：

Tx 功率 1 个字节

00h = -23 dBm (默认)

01h = -6 dBm

02h = 0 dBm

03h = 4 dBm

6.3.20. 重写客户主密钥（Customer Master Key Rewrite）

此命令用于设置客户主密钥。

Customer Master Key reset 的命令格式（5 个字节）

命令	CLA	INS	P1	P2	Lc
Customer Master Key reset	E0h	00h	00h	60h	00h

Customer Master Key reset 的响应格式 (21 个字节) (成功)

响应	CLA	INS	P1	P2	Le	响应数据域
结果	E1h	00h	00h	00h	10h	16 字节随机数

Customer Master Key rewrite 的命令格式（36 字节）

命令	CLA	INS	P1	P2	Lc
Customer Master Key rewrite	E0h	00h	00h	61h	16 字节加密随机数 + 16 字节加密的新主密钥

Customer Master Key rewrite 的响应结构(7 字节) (成功)

响应	CLA	INS	P1	P2	Le	响应数据域
结果	E1h	00h	00h	00h	02h	90 00h

Customer Master Key rewrite 的响应结构 (5 字节) (失败)

响应	CLA	INS	P1	P2	Le
结果	E1h	00h	00h	00h	00h

例如：要写入客户主密钥。

读写器生成的随机数是 “11 22 33 44 55 66 77 88 99 00 11 22 33 44 55 66”

主机设备 → 读写器: (Customer Master Key Reset 请求)

Frame1 “E0 00 00 60 00”

读写器 → 主机设备: (Customer Master Key Reset 命令响应)

Frame1 “E1 00 00 00 10 11 22 33 44 55 66 77 88 99 00 11 22 33 44 55 66”

主机设备 → 读写器: (Rewrite Master Key 命令请求)

Frame1 “E0 00 00 61 xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx”

读写器 → 主机设备: (Rewrite Master Key 命令响应)

Frame1 “E1 00 00 00 02 90 00”

响应	说明
E1 00 00 00 02 90 00	成功
E1 00 00 00 00	失败

6.3.21. 获取电池电量（Get Battery Level）

此命令用于查看当前电池电量（仅用于蓝牙模式）

***注：**仅适用于 2.03.xx 及以上版本的固件，且读写器处于蓝牙模式。.*

Get Battery Level 的命令格式（5 个字节）

命令	CLA	INS	P1	P2	Lc
Get Battery Level	E0h	00h	00h	52h	00h

Get Battery Level 的响应格式（6 个字节）

响应	CLA	INS	P1	P2	Le	响应数据域
结果	E1h	00h	00h	00h	01h	电池电量

其中：

电池电量　1 个字节

100 = 100%电量

90 = 90%电量

80 = 80%电量

70 = 70%电量

60 = 60%电量

50 = 50%电量

40 = 40%电量

30 = 30%电量

20 = 20%电量

15 = 15%电量

6.3.22. 读取 PICC 类型（Read PICC Type）

此命令用于查看当前的 PICC 类型。

注：*仅适用于 2.04.xx 及以上版本的固件。*

Read PICC Type 的命令格式（5 字节）

命令	CLA	INS	P1	P2	Lc
Read PICC Type	E0h	00h	00h	35h	00h

Read PICC Type 的响应格式（6 字节）

响应	CLA	INS	P1	P2	Le	响应数据域	
结果	E1h	00h	00h	00h	02h	卡片类型	卡片状态

其中：

卡片类型 1 字节

CCh = 缺失

04h = Topaz

10h = Mifare

11h = Felica 212 Kbps

12h = Felica 424 Kbps

20h = ISO 14443-4 A 类

23h = ISO 14443-4 B 类

25h = Calypso

28h = Srix

卡片状态 1 字节

00h = PICC 下电

其它 = 检测到 PICC [检测到非接触标签]