Advanced Card Systems Ltd.
Card & Reader Technologies

AMR220-C1

ACS 安全蓝牙™ mPOS 读写器

参考手册 V1.04

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

版本历史

发布日期	修订说明	版本号
2018-02-08	• 首次发布	1.00
2018-03-02	• 更新 2.0 节 – 特性 • 更新 3.1.2 节 相片 • 新增 3.1.2.3 节 – 设备重置针孔	1.01
2018-07-19	• 更新 3.1.1.2 电池寿命	1.02
2018-09-04	• 更新 2.0 节 – 特性	1.03
2019-06-20	• 更新读写器的市场营销名 • 更新 5.3.1 节 – 获取固件版本 • 更新 5.3.2 节 –休眠模式选项 • 更新 5.3.3 节 – 天线场控制 • 更新 5.3.4 节 – 自动 PICC 轮询 • 更新 5.3.5 节 – PICC 操作参数 • 更新 5.3.7 节 – LED 控制	1.04

目录

图目录

表目录

1.0. 简介

AMR220-C1 ACS 安全蓝牙™ mPOS 读写器通过蓝牙（Bluetooth®）技术与智能设备进行通信。它符合 ISO 7816 和 ISO 14443 标准, 支持接触式和非接触式智能卡。除此之外, 它还符合 EMV® Level 1 和 Level 2, Mastercard® Contactless (原名 MasterCard PayPass) 和 Visa® Contactless 标准, 支持的卡片范围更广, 提升了 ACS 移动读卡器产品线在支付领域的适用度。

目标客户包括微商, 移动商贩（例如美食车, 流动餐车, 快递公司）, 零售商等。

1.1. 符号和缩写

缩写	说明
AC	应用密文 Application Cryptogram
AID	应用标识符 Application Identifier
AIP	应用交互特征 Application Interchange Profile
AOSA	可用脱机消费金额 Available Offline Spending Amount
APDU	应用协议数据单元 Application Protocol Data Unit
ATC	应用交易计数器 Application Transaction Counter
BT	蓝牙 Bluetooth
BLE	低功耗蓝牙 Bluetooth Low Energy
CA	认证中心 Certification Authority
CED	客户专属数据 Customer Exclusive Data
CID	密文信息数据 Cryptogram Information Data
CVM	持卡人验证方法 Cardholder Verification Method
CVR	卡片验证结果 Card Verification Results
DD	可随意使用数据 Discretionary Data
DF	专用文件 Dedicated File
FFI	形状系数标识符 Form Factor Indicator
FW	固件 Firmware
IAD	发卡行应用数据 Issuer Application Data
IFD	接口设备 Interface Device
JCB	吉士美卡 Japan Credit Bureau

缩写	说明
PAN	主账号 Primary Account Number
PBOC	中国人民银行标准 People's Bank of China specifications
PCD	邻近耦合设备（读写器） Proximity Coupling Device
PIN	个人识别码 Personal Identification Number
POS	销售点 Point of Sale
PSN	应用 PAN 序号 Application PAN Sequence Number
RID	注册应用提供商标识 Registered Application Provider Identifier
QPBOC	快速借记/贷记应用 Quick PBOC (非接触式 EMV 应用的中国应用)
TAC	终端行为代码 Terminal Action Code
TTQ	终端交易属性 Terminal Transaction Qualifiers
TVR	终端验证结果 Terminal Verification Results

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，具有高度灵活性
- 智能卡读写器：
  - 非接触接口：
    - 读写速率可达 848 Kbps
    - 内置天线用于读写非接触式标签，读取距离可达 50 mm（视标签的类型而定）
    - 支持 ISO 14443 第 4 部分 A 类和 B 类卡，MIFARE® 卡，FeliCa 卡和全部 4 种 NFC（ISO/IEC 18092）标签
    - 支持符合 Mastercard® Contactless 和 Visa payWave® 的卡
    - 内建防冲突特性（任何时候只能访问 1 张标签）
    - 支持的 NFC 模式：
      - **卡片读/写模式**
  - 接触式接口：
    - 读写速率高达 600 Kbps
    - 支持 ISO 7816 A 类, B 类和 C 类（5V, 3V, 1.8V）标准尺寸卡
    - 支持符合 T=0 或 T=1 协议的微处理器卡
    - 支持协议和参数选择（PPS）
    - 具有短路保护功能
- 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）
- 内置外设：
  - LED 指示灯：
    - 4 个用户可控的单色 LED（绿色）
    - 1 个电池充电状态 LED（红色）
    - 1 个蓝牙状态 LED（蓝色）
  - 按键：
    - 电源开关
    - 蓝牙开关
  - 用户可控的扬声器（单音指示）
- 支持多种加密算法（按需定制），例如 AES, DES 和 3DES
- 具有 USB 固件升级能力 [1]
- 支持 Android™ 4.4 及更高版本 [2]
- 支持 iOS 8.0 及更高版本 [3]

---

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

- 符合下列标准：
  - EN 60950/IEC 60950
  - ISO 7816
  - ISO 14443
  - ISO 18092
  - EMV® Level 1 和 Level 2
  - Mastercard® Contactless
  - Visa payWave®
  - Bluetooth™
  - PC/SC
  - CCID
  - CE
  - FCC
  - RoHS 3
  - REACH
  - MIC（日本）
  - Microsoft® WHQL

3.0. 架构

3.1. 硬件设计

采用的 Cortex M3 级主处理器可通过蓝牙接口或 USB 接口与平板电脑或计算机通信，也可控制外设并与 ICC 通信。NFC 芯片作为收发器在非接触标签与主处理器之间建立 RF 通道。

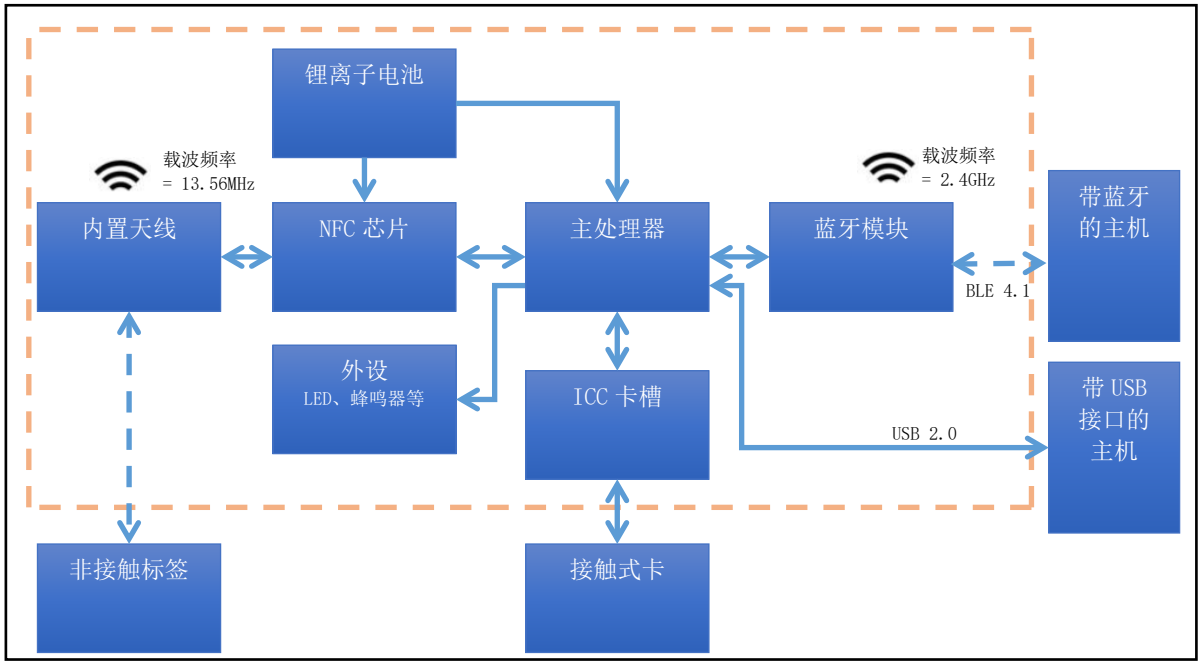


图1：硬件结构

3.1.1. 电池

AMR220-C1 使用容量为 450 mAh 的锂离子充电电池。

3.1.1.1. 电池充电

AMR220-C1 可连接电源插座为电池充电。

3.1.1.2. 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： 接触式接口	9 天*(1)
工作模式： 非接触接口	7 天*(1)
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。

3.1.2. 用户接口

3.1.2.1. LED 操作

充电状态 LED

LED 状态		说明
充电 LED 开启		正在充电
充电 LED 关闭		已充满

蓝牙状态 LED

LED 状态	说明
缓慢闪烁（0.5 秒 - 开启, 4.5 秒 - 关闭）	已配对
缓慢闪烁（0.5 秒 - 开启, 1.5 秒 - 关闭）	配对中
关	蓝牙停用

LED 状态	
蓝牙状态 LED 开启	
蓝牙状态 LED 关闭	

EMV 非接触 LED 操作

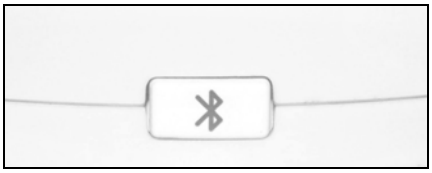
注：*如需了解更多信息, 请参考 EMV 非接触规范：https://www.emvco.com/wp-content/uploads/2017/05/Book_A_Architecture_and_General_Rqmts_v2_6_Final_20160422011856105.pdf。*

3.1.2.2. 开关操作

3.1.2.2.1. 电源开关操作

- 开机时, 长按电源开关 1-2 秒钟。
- 关机时, 长按电源开关, 直到听到“哔”的一声, 松开开关。

3.1.2.2.2. 蓝牙开关操作



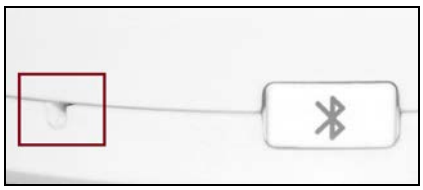
激活/取消激活蓝牙配对：

- 要激活蓝牙配对, 只需按一下蓝牙开关。
- 要取消激活蓝牙配对, 需要在两秒钟内两次按压蓝牙开关。

设置设备进入固件升级模式：

- 如果设备已经开机, 长按蓝牙开关 10 秒钟。
- 如果设备关机, 同时按下蓝牙开关和电源开关。

3.1.2.3. 设备重置针孔



重置设备：

- 重置设备，找到蓝牙按钮旁边的针孔，然后使用一个针按下重置按钮。

3.2. 软件设计

3.2.1. USB 接口

通过 MS-CCID 驱动，仅可支持一个单槽设备，并且只能使用 PICC 接口。若想使用两个接口，就需要用到 ACS 驱动。AMR220-C1 的 USB 接口符合 CCID 协议，定义了两个插槽，一个用于 ICC 接口，一个用于 PICC 接口。

AMR220-C1 是一款 CCID 设备，所以主机应用程序完全符合 PCSC 标准。

注：*关于这些 API 的更多细节，请参考 Microsoft MSDN 库或 PCSC 工作组。*

一些经常用到的 PCSC API 将在 **PCSC API** 中进行介绍。

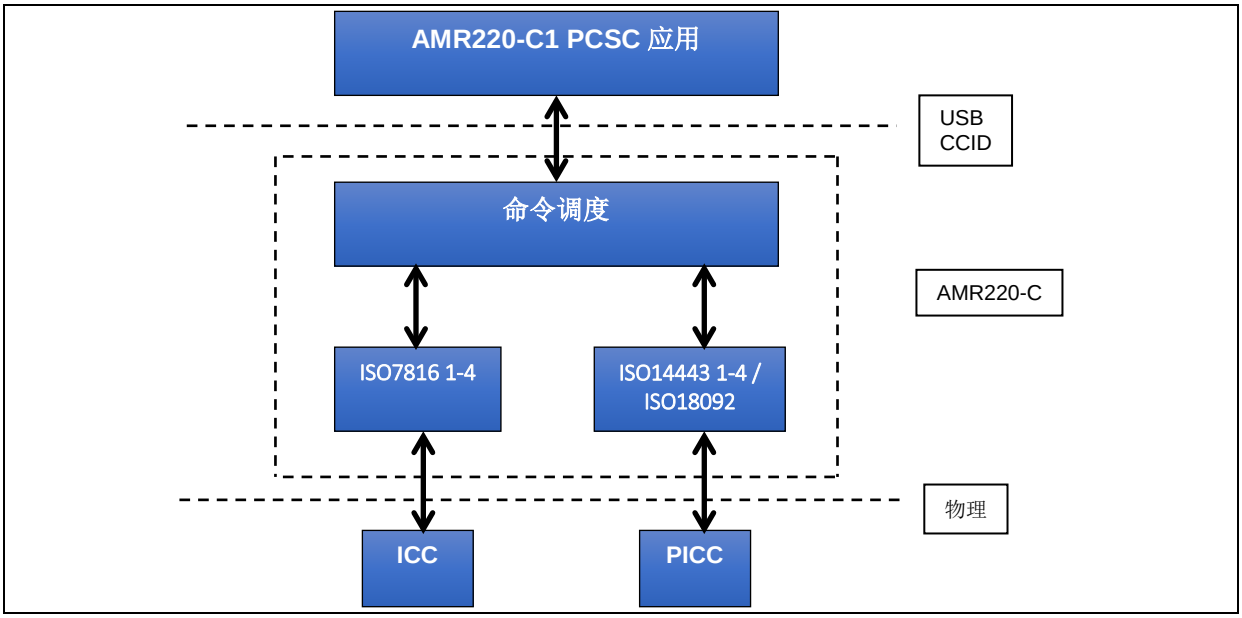


图2：软件架构 - USB 接口

3.2.2. 蓝牙接口

AMR220-C1 的蓝牙接口符合低功耗蓝牙（BLE）4.1 标准。ACS 提供一个高级别的 Android/IOS 库来简化应用程序编程。蓝牙 API 将在**蓝牙库**中进行介绍。

以下是 BLE 的架构, 供用户参考。

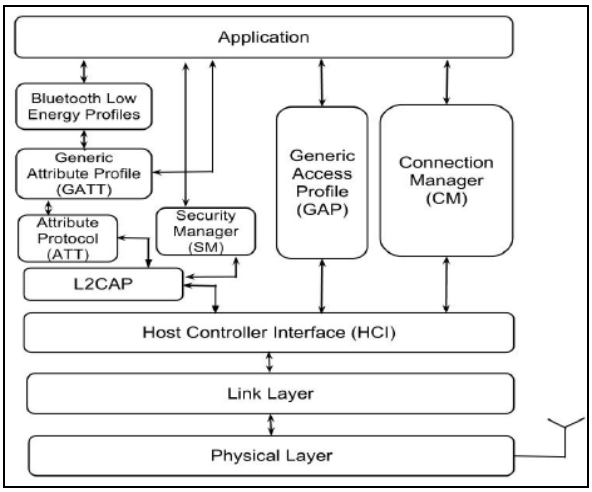


图3： 软件架构 - 蓝牙接口

4.0. 主机口程

4.1. PCSC API

本节将介绍一些使用 USB 接口进行应用程序编程的 PCSC API。关于这些 API 的更多细节，请参考 Microsoft MSDN 库或 PCSC 工作组。

4.1.1. SCardEstablishContext

SCardEstablishContext 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

4.1.2. SCardListReaders

SCardListReaders 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

4.1.3. SCardConnect

SCardConnect 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

4.1.4. SCardControl

SCardControl 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 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：直接命令介绍的命令要使用此 API 进行发送。

4.1.5. ScardTransmit

SCardTransmit 函数用来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（**即非接触接口的私有 APDU 指令**发送给所连接卡片的命令）要使用此 API 进行发送。.*

4.1.6. ScardDisconnect

SCardDisconnect 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

4.1.7. APDU 流程图

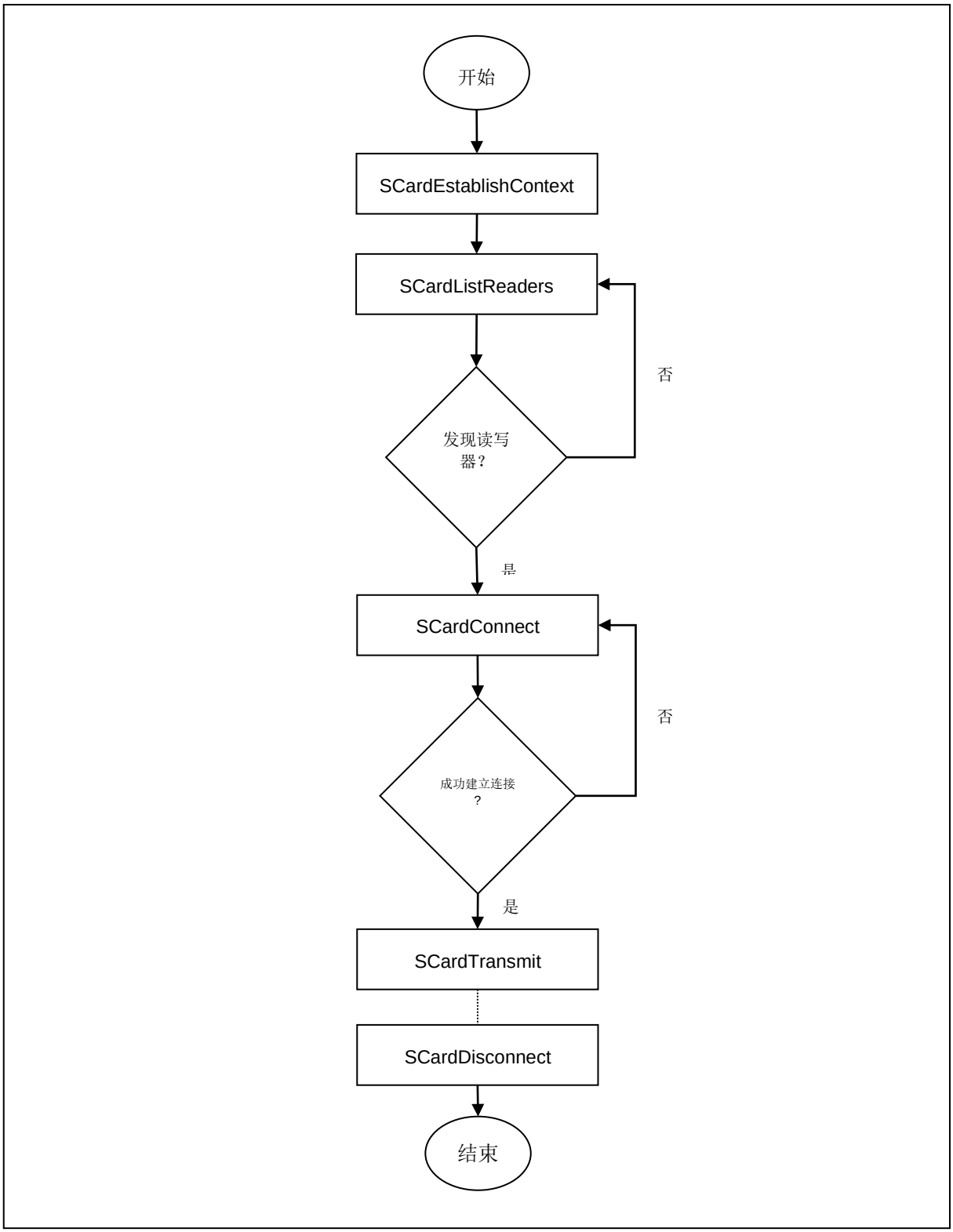


图4： APDU 流程图

4.1.8. 直接（Escape）命令流程图

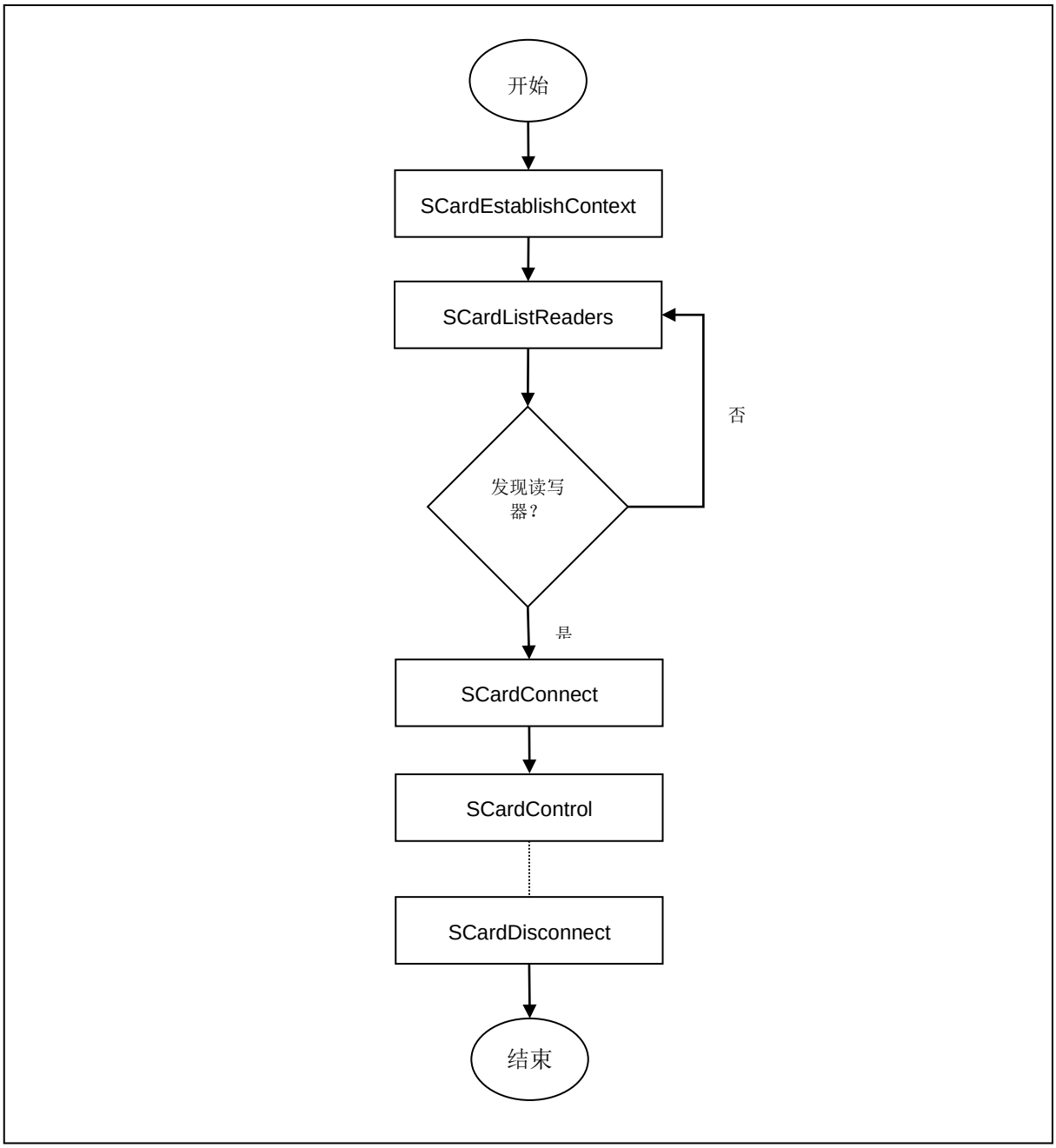


图5： 直接（Escape）命令流程图

4.2. 蓝牙库

本节介绍了 ACS 提供给开发人员使用的蓝牙库。更多详情, 请参考库封装文件。

4.2.1. 设置 BLE

如果设备支持 BLE, 启用蓝牙功能才能连接卡片终端。要启用蓝牙功能, 需要从 BluetoothManager 获取 BluetoothAdapter 实例。

```
private BluetoothAdapter mBluetoothAdapter;
 ...
 // Initializes Bluetooth adapter.
final BluetoothManager bluetoothManager = (BluetoothManager)
      getSystemService(Context.BLUETOOTH_SERVICE);
mBluetoothAdapter = bluetoothManager.getAdapter();
```

要查看蓝牙是否启用, 只需从 BluetoothAdapter 调用 isEnabled(), 然后再调用 startActivityForResult()来请求用户许可启用蓝牙功能。执行完 onActivityResult()后返回结果。

```
// Ensures Bluetooth is available on the device and it is enabled. If not,
// displays a dialog requesting user permission to enable Bluetooth.
if (mBluetoothAdapter == null || !mBluetoothAdapter.isEnabled()) {
   Intent enableBtIntent = new
   Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
    startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
 }
```

4.2.2. 初始化 Java 智能卡 I/O API

要使用 Java 智能卡 I/O API 和 BLE 卡终端, 需要调用 BluetoothSmartCard 的 getInstance()。这是一个将 Context 作为参数的单例类。如果在 Activity 中调用这个方法, 他们可以将此传递给参数。

```
private BluetoothTerminalManager mManager;
private TerminalFactory mFactory;
 ...
 // Get the Bluetooth terminal manager.
 mManager = BluetoothSmartCard.getInstance(this).getManager();

 if (mManager == null) {
     Toast.makeText(this, R.string.error_bluetooth_not_supported,
              Toast.LENGTH_SHORT).show();
     finish();
     return;
 }

 // Get the terminal factory.
 mFactory = BluetoothSmartCard.getInstance(this).getFactory();

 if (mFactory == null) {
     Toast.makeText(this, R.string.error_bluetooth_provider_not_found,
           Toast.LENGTH_SHORT).show();
     finish();
     return;
 }
```

4.2.3. 发现 BLE 卡终端

要找到 BLE 卡终端，需用到 BluetoothTerminalManager 的 startScan()方法。提供终端类型和回调来返回 CardTerminal 对象。

找到卡片终端后，必须调用 stopScan()来停止扫描，以避免智能设备电池电量快速流失而影响操作。

对于 Android 6.0 或更高版本，开发人员必须在运行时间请求 ACCESS_COARSE_LOCATION 或 ACCESS_FINE_LOCATION 权限才能扫描 BLE 设备。

```
private Handler mHandler;
private Button mScanButton;
private TerminalAdapter mTerminalAdapter;
...
// Initialize Scan button.
mHandler = new Handler();
mScanButton = (Button) findViewById(R.id.activity_main_button_scan);
mScanButton.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        // Request access coarse location permission.
        if (ContextCompat.checkSelfPermission(MainActivity.this,
              Manifest.permission.ACCESS_COARSE_LOCATION)
              != PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(MainActivity.this,
               new String[]{Manifest.permission.ACCESS_COARSE_LOCATION},
               REQUEST_ACCESS_COARSE_LOCATION);
        } else {
           mScanButton.setEnabled(false);
           mTerminalAdapter.clear();

        // Start the scan.

       mManager.startScan(BluetoothTerminalManager.TERMINAL_TYPE_AMR220_C,
            new BluetoothTerminalManager.TerminalScanCallback() {
                @Override
                public void onScan(final CardTerminal terminal) {
                 runOnUiThread(new Runnable() {
                   @Override
                   public void run() {
                     mTerminalAdapter.addTerminal(terminal);
                   }
                 });
               }
           });

         // Stop the scan.
         mHandler.postDelayed(new Runnable() {
          @Override
          public void run() {
          mManager.stopScan();
          mScanButton.setEnabled(true);
         }
        }, SCAN_PERIOD);
      }
    }
});
```

从 onRequestPermissionsResult()返回结果。获得授权后, 应用可以开始扫描。

```
@Override
public void onRequestPermissionsResult(int requestCode,
      @NonNull String[] permissions,
      @NonNull int[] grantResults) {
  if (requestCode == REQUEST_ACCESS_COARSE_LOCATION) {
      if ((grantResults.length > 0)
            && (grantResults[0] == PackageManager.PERMISSION_GRANTED)) {
          mScanButton.setEnabled(false);
          mTerminalAdapter.clear();

        // Start the scan.

       mManager.startScan(BluetoothTerminalManager.TERMINAL_TYPE_AMR220_C,
           new BluetoothTerminalManager.TerminalScanCallback() {
             @Override
             public void onScan(final CardTerminal terminal) {
               runOnUiThread(new Runnable() {
                 @Override
                 public void run() {
                   mTerminalAdapter.addTerminal(terminal);
                 }
               });
             }
          });

        // Stop the scan.
        mHandler.postDelayed(new Runnable() {
          @Override
          public void run() {
            mManager.stopScan();
            mScanButton.setEnabled(true);
          }
        }, SCAN_PERIOD);
      }
   } else {
      super.onRequestPermissionsResult(requestCode, permissions,
              grantResults);
   }
 }
```

4.2.4. 连接卡片

要连接卡片, 需要调用 CardTerminal 对象的 connect()来返回卡片对象。可用的协议包括 T=0, T=1, T=0 或 T=1 和直接模式。

```
// Protocol:
 // "T=0"    - T=0
 // "T=1"    - T=1
 // "*"      - T=0 or T=1
 // "direct" - Direct mode
 try {
     Card card = terminal.connect("*");
 } catch (CardException e) {
     e.printStackTrace();
 }
```

4.2.5. 断开卡片连接

使用完卡片后, 可以调用卡片对象的 disconnect()来断开卡片连接。如果断开连接后需要重置卡片, 可发送 true 给参数。

```
try {
     card.disconnect(false);
 } catch (CardException e) {
     e.printStackTrace();
 }
```

4.2.6. 传输 APDU

成功连接卡片后, 必须开放一个通道来传输 APDU。方法是调用卡片对象的 getBasicChannel()或 openLogicalChannel() 。返回 CardChannel 对象后, 即可使用 CardChannel 对象的 transmit(CommandAPDU)或 transmit(ByteBuffer, ByteBuffer)来传输 APDU。

```
byte[] command = { 0x00, (byte) 0x84, 0x00, 0x00, 0x08 };

 try {
     Card card = terminal.connect("*");
     CardChannel channel = card.getBasicChannel();
     CommandAPDU commandAPDU = new CommandAPDU(command);
     ResponseAPDU responseAPDU = channel.transmit(commandAPDU);
 } catch (CardException e) {

     e.printStackTrace();
 }
```

4.2.7. 传输控制命令

成功连接卡片后, 可以调用卡片对象的 transmitControlCommand()来传输控制命令。

```
int controlCode = BluetoothTerminalManager.IOCTL_ESCAPE;
 try {
     Card card = terminal.connect("direct");
     card.transmitControlCommand(controlCode, command);
 } catch (CardException e) {
     e.printStackTrace();
 }
```

4.2.8. 断开 BLE 卡终端连接

当调用 CardTerminal 对象的 connect()时, 库会自动连接 BLE 卡终端。要手动终止蓝牙连接, 必须调用 BluetoothTerminalManager 对象的 disconnect()。

```
mManager.disconnect(terminal);
```

5.0. 命令集

5.1. 平板电脑和 AMR220-C1 之间的 API

***注：**此部分内容仅针对不准备使用 ACS BT 库开发自己应用程序的开发人员，如果您使用的是 ACS BT 库，请忽略此部分内容。*

5.1.1. BT 通信帧格式

AMR220-C1 和平板电脑之间采用预定义的帧格式进行通信。

帧格式的定义如下：

STX (0x02)	Data Len MSB	Data Len LSB	Sequence Number	Frame Type	Data Field (N 个字节)	CheckSum

其中：

STX 文本起点，必须是 0x02

Data Len MSB 数据字段长度，MSB（1 字节）

Data Len LSB 数据字段长度，LSB（1 字节）

Sequence Number BT 消息的序号，每条消息加 1，bit 7 表示是否为链接数据包（“1”表示是链接数据包；“0”表示最后一个数据包）

Frame Type 通信帧类型

不带加密的数据帧 = 0x00　　加密的数据帧 = 0x01

ACK 帧 = 0x02　　NAK 帧 = 0x03

Abort 帧 = 0x04　　INT 帧 = 0x05

Inter 字符超时帧 = 0xF1　　校验和错误帧 = 0xF2

数据长度错误帧 = 0xF3

Data Field： 消息正文（N 个字节），数据加密部分

Checksum： XOR {数据长度 MSB, 数据长度 LSB, 序列号, 帧类型, 数据字段}

状态响应：

1. **成功接收或已接受链接 - ACK**

   响应格式：

0x02	0x00	0x00	Seq	0x02	CS

   ACK 应答后紧跟着响应帧

2. **接收失败 - 否定应答 - NAK**

   响应格式：

0x02	0x00	0x00	Seq	0x03	CS

3. **Inter 字符超时 - 每个字符间的超时值（比如说 5ms）**

   响应格式：

0x02	0x00	0x00	Seq	0xF1	CS

4. **校验和检查 - 校验和错误**

   响应格式：

0x02	0x00	0x00	Seq	0xF2	CS

5. **数据长度错误 - 数据长度超出最大限制**

   响应格式：

0x02	0x00	0x00	Seq	0xF3	CS

5.1.2. 数据字段格式– 命令

本节定义了命令帧的数据字段内容格式。

数据字段				
INS (1 字节)	Counter (1 字节)	Command Payload Length (2 字节)	Command Payload (N 字节)	CS (1 字节)

其中：

INS	命令头 (1 字节) 用于说明特定命令属性的命令头部
Counter	计数器用于防止重放攻击 (1 字节） 每个命令增加一次, 用于防止重放攻击
Command Payload Length	命令数据包的长度（2 字节）
Command Payload	命令消息（N 字节）
CS	校验和，XOR {INS, Counter, Command Payload Length, Command Payload}

5.1.3. 数据字段格式– 响应

本节定义了响应帧的数据字段内容格式。

数据字段				
RSP (1 字节)	Counter (1 字节)	Response Payload Length (2 字节)	ResponsePayload (N 字节)	CS (1 字节)

其中：

RSP	响应头 (1 字节) 对应特定命令头的响应头部
Counter	计数器用于防止重放攻击(1 字节） 每个响应增加一次
Response Payload Length	响应数据包的长度 (2 字节）
ResponsePayload	响应报文 (N 字节)
CS	校验和, XOR {INS, Counter, ResponsePayload Length, ResponsePayload}

5.1.4. BT 命令和响应

5.1.4.1. 命令(从智能设备至 AMR220-C1)

命令名称	说明	INS
SPH_to_RDR_EmvExchangeData	在智能设备和 EMV L2 内核之间交换数据	0x44
SPH_to_RDR_PcdPowerOn	请求 AMR220-C1 在定义的范围内轮询 PCD 标签	0x80
SPH_to_RDR_PcdPowerOff	请求 AMR220-C1 取消选择已经激活的 PCD 标签	0x81
SPH_to_RDR_PcdExAPDU	请求 AMR220-C1 与已经激活的 PCD 标签交换 APDU ***注：****必须首先完成 SPH_to_RDR_PcdPowerOn*。	0x82
SPH_to_RDR_IccPowerOn	请求 AMR220-C1 激活 ICC/SAM 卡。	0xA0
SPH_to_RDR_IccPowerOff	请求 AMR220-C1 下电激活的 ICC/SAM 卡。	0xA1
SPH_to_RDR_IccExAPDU	请求 AMR220-C1 与激活的 ICC/SAM 卡交换 APDU ***注：****必须首先完成 SPH_to_RDR_IccPowerOn*。	0xA2
SPH_to_RDR_IccSetParameter	请求 AMR220-C1 与激活的 ICC/SAM 卡进行 PPS ***注：****必须首先完成 SPH_to_RDR_IccPowerOn*。	0xA3
SPH_to_RDR_ExEscape	请求 AMR220-C1 交换 Escape 命令来控制/配置外设	0xC0

表3： 从智能设备至 AMR220-C1 的 BT 命令

5.1.4.1.1. SPH_to_RDR_PcdPowerOn

此命令请求 AMR220-C1 在定义的范围内轮询 PCD 标签。

偏移	数据域	大小	值	说明
0	INS	1	0x80	命令头
1	Counter	1		命令计数器
2	Command Payload Length	2	0x0006	命令数据包的长度
4	Command Payload	6		Byte 1 – 卡片类型 Bit 0 = 0 – 禁用 Type A 轮询 1 – 启用 Type A 轮询 Bit 1 = 0 – 禁用 Type B 轮询 1 – 启用 Type B 轮询 Bit 2 = 0 – 禁用 FeliCa212 轮询 1 – 启用 FeliCa212 轮询 Bit 3 = 0 – 禁用 FeliCa424 轮询 1 – 启用 FeliCa424 轮询 Bit 4-6 = RFU Bit 7 = 0 – 不发送 RATS 1 – 自动发送 RATS Byte 2 – 轮询重试 Byte 3 – 6 – 轮询间隔 每次轮询间的间隔 （单位：1ms)
10	CS	1		XOR 上述字段

此报文的响应是 RDR_to_SPH_PcdPowerOnRsp

5.1.4.1.2. SPH_to_RDR_PcdPowerOff

此命令请求 AMR220-C1 取消选择已经激活的 PCD 标签

偏移	数据域	大小	值	说明
0	INS	1	0x81	命令头
1	Counter	1		命令计数器
2	Command Payload Length	2	0x0000	命令数据包的长度
4	Command Payload	0		-
4	CS	1		XOR 上述字段

此报文的响应是 RDR_to_SPH_PcdPowerOffRsp

5.1.4.1.3. SPH_to_RDR_PcdExAPDU

此命令请求 AMR220-C1 与激活的 PCD 标签交换 APDU。

注：*必须首先完成 SPH_to_RDR_PcdPowerOn*

偏移	数据域	大小	值	说明
0	INS	1	0x82	命令头
1	Counter	1		命令计数器
2	Command Payload Length	2	N	命令数据包的长度
4	Command Payload	N		用于激活 PCD 标签的 APDU
N+4	CS	1		XOR 上述字段

此报文的响应是 RDR_to_SPH_PcdExAPDURsp

5.1.4.1.4. SPH_to_RDR_IccPowerOn

此命令请求 AMR220-C1 激活 ICC/SAM 卡。

偏移	数据域	大小	值	说明
0	INS	1	0xA0	命令头
1	Counter	1		命令计数器
2	Command Payload Length	2	0x0001	命令数据包的长度
4	Command Payload	1		ICC 卡槽选择 0x01 = ICC 卡槽
5	CS	1		XOR 上述字段

此报文的响应是 RDR_to_SPH_IccPowerOnRsp

5.1.4.1.5. SPH_to_RDR_IccPowerOff

此命令请求 AMR220-C1 下电激活的 ICC/SAM 卡。

偏移	数据域	大小	值	说明
0	INS	1	0xA1	命令头
1	Counter	1		命令计数器
2	Command Payload Length	2	0x0001	命令数据包的长度
4	Command Payload	1		ICC 卡槽选择 0x01 = ICC 卡槽
5	CS	1		XOR 上述字段

此报文的响应是 RDR_to_SPH_IccPowerOffRsp

5.1.4.1.6. SPH_to_RDR_IccExAPDU

此命令请求 AMR220-C1 与激活的 ICC/SAM 卡交换 APDU。

注：*必须首先完成 SPH_to_RDR_IccPowerOn。*

偏移	数据域	大小	值	说明
0	INS	1	0xA2	命令头
1	Counter	1		命令计数器
2	Command Payload Length	2	N	命令数据包的长度
4	Command Payload	N		字节 1 - ICC 卡槽选择 0x01 = ICC 卡槽 字节 2 - … - APDU 用于激活 ICC 卡的 APDU
N+4	CS	1		XOR 上述字段

此报文的响应是 RDR_to_SPH_IccExAPDURsp

5.1.4.1.7. SPH_to_RDR_IccSetParameter

此命令请求 AMR220-C1 与激活的 ICC/SAM 卡进行 PPS。

注：*必须先完成 SPH_to_RDR_IccPowerOn。*

偏移	数据域	大小	值	说明
0	INS	1	0xA3	命令头
1	Counter	1		命令计数器
2	Command Payload Length	2	5	命令数据包的长度
4	Command Payload	5		字节 1 - ICC 卡槽选择 0x01 = ICC 卡槽 字节 2 - 5 - PPS
9	CS	1		XOR 上述字段

此报文的响应是 RDR_to_SPH_IccSetParameterRsp。

5.1.4.1.8. SPH_to_RDR_ExEscape

此命令请求 AMR220-C1 交换 Escape 命令来控制/配置外设。

偏移	数据域	大小	值	说明
0	INS	1	0xC0	命令头
1	Counter	1		命令计数器
2	Command Payload Length	2	5	命令数据包的长度
4	Command Payload	N		直接（Escape）命令
N+4	CS	1		XOR 上述字段

此报文的响应是 RDR_to_SPH_ExEscapeRsp

5.1.4.2. 响应 (从 AMR220-C1 至智能设备)

响应名称	说明	INS
RDR_to_SPH_EmvExchangeData	SPH_to_RDR_EmvExchangeData 的响应	0x54
RDR_to_SPH_PcdPowerOnRsp	SPH_to_RDR_PcdPowerOn 的响应	0x90
RDR_to_SPH_PcdPowerOffRsp	SPH_to_RDR_PcdPowerOff 的响应	0x91
RDR_to_SPH_PcdExAPDURsp	SPH_to_RDR_PcdExAPDU 的响应	0x92
RDR_to_SPH_IccPowerOnRsp	SPH_to_RDR_IccPowerOn 的响应	0xB0
RDR_to_SPH_IccPowerOffRsp	SPH_to_RDR_IccPowerOff 的响应	0xB1
RDR_to_SPH_IccExAPDURsp	SPH_to_RDR_IccExAPDU 的响应	0xB2
RDR_to_SPH_IccSetParameterRsp	SPH_to_RDR_IccSetParameter 的响应	0xB3
RDR_to_SPH_ExEscapeRsp	SPH_to_RDR_ExEscape 的响应	0xD0

表4： 从 AMR220-C1 至智能设备的 BT 响应

5.1.4.2.1. RDR_to_SPH_PcdPowerOnRsp

此命令是对 SPH_to_RDR_PcdPowerOn 的响应。

偏移	数据域	大小	值	说明
0	RSP	1	0x90	响应的头部
1	Counter	1		响应计数器
2	Response Payload Length	2	N	响应数据包的长度
4	Response Payload	N		字节 1 – 错误代码 0x00 = 无错误 其它 = 参考错误代码表 字节 2 - ... - 卡片 ATR
N+4	CS	1		XOR 上述字段

5.1.4.2.2. RDR_to_SPH_PcdPowerOffRsp

此命令是对 SPH_to_RDR_PcdPowerOff 的响应。

偏移	数据域	大小	值	说明
0	RSP	1	0x91	响应的头部
1	Counter	1		响应计数器
2	Response Payload Length	2	0x0001	响应数据包的长度
4	Response Payload	1		0x00 = 无错误 其它 = 参考错误代码表
5	CS	1		XOR 上述字段

5.1.4.2.3. RDR_to_SPH_PcdExAPDURsp

此命令是对 RDR_to_SPH_PcdExAPDURsp 的响应。

偏移	数据域	大小	值	说明
0	RSP	1	0x92	响应的头部
1	Counter	1		响应计数器
2	Response Payload Length	2	N	响应数据包的长度
4	Response Payload	N		字节 1 – 错误代码 0x00 = 无错误 其它 = 参考错误代码表 字节 2 - ... - APDU 响应
N+4	CS	1		XOR 上述字段

5.1.4.2.4. RDR_to_SPH_IccPowerOnRsp

此命令是对 SPH_to_RDR_IccPowerOn 的响应。

偏移	数据域	大小	值	说明
0	RSP	1	0xB0	响应的头部
1	Counter	1		响应计数器
2	Response Payload Length	2	N	响应数据包的长度
4	Response Payload	N		字节 1 - ICC 卡槽选择 0x01 = ICC 卡槽 字节 2– 错误代码 0x00 = 无错误 其它 = 参考错误代码表 字节 3 - … - 卡片 ATR
N+4	CS	1		XOR 上述字段

5.1.4.2.5. RDR_to_SPH_IccPowerOffRsp

此命令是对 SPH_to_RDR_IccPowerOff 的响应。

偏移	数据域	大小	值	说明
0	RSP	1	0xB1	响应的头部
1	Counter	1		响应计数器
2	Response Payload Length	2	0x0002	响应数据包的长度
4	Response Payload	2		字节 1 - ICC 卡槽选择 0x01 = ICC 卡槽 字节 2– 错误代码 0x00 = 无错误 其它 = 参考错误代码表
6	CS	1		XOR 上述字段

5.1.4.2.6. RDR_to_SPH_IccExAPDURsp

此命令是对 SPH_to_RDR_IccExAPDU 的响应。

偏移	数据域	大小	值	说明
0	RSP	1	0xB2	响应的头部
1	Counter	1		响应计数器
2	Response Payload Length	2	N	响应数据包的长度
4	Response Payload	N		字节 1 - ICC 卡槽选择 0x01 = ICC 卡槽 字节 2– 错误代码 0x00 = 无错误 其它 = 参考错误代码表 字节 3 - … - APDU 响应
N+4	CS	1		XOR 上述字段

5.1.4.2.7. RDR_to_SPH_IccSetParameterRsp

此命令是对 SPH_to_RDR_IccSetParameter 的响应。

偏移	数据域	大小	值	说明
0	RSP	1	0xB3	响应的头部
1	Counter	1		响应计数器
2	Response Payload Length	2	N	响应数据包的长度
4	Response Payload	N		字节 1 - ICC 卡槽选择 0x01 = ICC 卡槽 字节 2– 错误代码 0x00 = 无错误 其它 = 参考错误代码表 字节 3 - … - PPS 响应
N+4	CS	1		XOR 上述字段

5.1.4.2.8. RDR_to_SPH_ExEscapeRsp

此命令是对 SPH_to_RDR_ExEscape 的响应。

偏移	数据域	大小	值	说明
0	RSP	1	0xD0	响应的头部
1	Counter	1		响应计数器
2	Response Payload Length	2	N	响应数据包的长度
4	Response Payload	N		Escape 命令响应
N+4	CS	1		XOR 上述字段

5.2. 非接触式智能卡协议

5.2.1. ATR 的生成

PICC 接口的 ATR 符合 PCSC 规范。

注：如需了解更多信息，□参考
http://pcscworkgroup.com/Download/Specifications/pcsc3_v2.01.09.pdf。

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

字节	值 (Hex)	标记	说明
0	0x3B	Initial	
1	0x8N	T0	高半字节“8”表示： • 不存在 TA1, TB1 和 TC1；存在 TD1 低半字节“N”表示： • 历史字节的数量
2	0x80	TD1	高半字节“8”表示： • 不存在 TA2, TB2 和 TC2；存在 TD2 低半字节“0”表示： • 支持 T=0 协议
3	0x01	TD2	高半字节“0”表示： • 不存在 TA3, TB3, TC3 和 TD3 低半字节“1”表示： • 支持 T=1 协议
4 至 3+N	0x80	T1	类别指示字节 “80”表示在可选的 COMPACT-TLV 数据对象中或许存在一个状态标识符
	0x4F	… Tk	应用标识符存在标识
	0x0C		长度
	RID		注册应用提供商标识（RID） 等于“0xA0 0x00 0x00 0x03 0x06”
	SS		标准字节
	C0 ..C1		卡片名称字节
	0x00 0x00 0x00 0x00	RFU	RFU
4+N	UU	TCK	T0 至 Tk 的所有字节按位异或

例如：Mifare 1K 的 ATR = {0x3B 0x8F 0x80 0x01 0x80 0x4F 0x0C 0xA0 0x00 0x00 0x03 0x06 0x03 0x00 0x01 0x00 0x00 0x00 0x00 0x6A}

长度 (YY) = 0x0C

RID = {0xA0 0x00 0x00 0x03 0x06} (PC/SC 工作组)

标准 (SS) = 0x03 (ISO14443A, Part 3)

卡片名称 (C0 ..C1) = {0x00 0x01} (Mifare 1K)

标准 (SS)

0x03：ISO14443A, 第 3 部分　0x11：FeliCa

卡片名称 (C0 ..C1)

0x00 0x01:Mifare 1K
0x00 0x02:Mifare 4K
0x00 0x03:Mifare Ultralight
0x00 0x26:Mifare Mini
0x00 0x30:Topaz
0x000x3B:FeliCa
0x00 0x38:Mifare Plus SL2_2K
0x00 0x39:Mifare Plus SL2_4K
0xFF [SAK]: 尚未定义的标签

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

<table>
<tr><th>字节</th><th>值
(Hex)</th><th>标记</th><th>说明</th></tr>
<tr><td>0</td><td>0x3B</td><td>Initial</td><td></td></tr>
<tr><td>1</td><td>0x8N</td><td>T0</td><td>高半字节“8”表示：
• 不存在 TA1, TB1 和 TC1；存在 TD1
低半字节“N”表示：
• 历史字节的数量</td></tr>
<tr><td>2</td><td>0x80</td><td>TD1</td><td>高半字节“8”表示：
• 不存在 TA2, TB2 和 TC2；存在 TD2
低半字节“0”表示：
• 支持 T=0 协议</td></tr>
<tr><td>3</td><td>0x01</td><td>TD2</td><td>高半字节“0”表示：
• 不存在 TA3, TB3, TC3 和 TD3
低半字节“1”表示：
• 支持 T=1 协议</td></tr>
<tr><td rowspan="2">4
至
3 + N</td><td>XX</td><td>T1</td><td rowspan="2">历史字节：

ISO14443A：
来自 ATS 应答的历史字节。
请参考 ISO14443-4 规范。

ISO14443B：
<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

…
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>

例如 1：Desfire 的 ATR = {0x3B 0x81 0x80 0x01 0x80 0x80} // 6 字节的 ATR

注：使用 APDU""0xFF 0xCA 0x01 0x00 0x00"来区分是符合 ISO14443A-4 的 PICC 还是符合 ISO14443B-4 的 PICC，并且如果有的话，取回完整的 ATS。ISO14443A-3 和 ISO14443B-3/4 的 PICC 会返回 ATS。

APDU 命令 = 0xFF 0xCA 0x01 0x00 0x00

APDU 响应 = 0x06 0x75 0x77 0x81 0x02 0x80 0x90 0x00

ATS = {0x06 75 77 81 02 80}

例如 2：EZ-link 的 ATR = {0x3B 0x880x80 0x01 0x1C 0x2D 0x94 0x11 0xF7 0x71 0x85 0x00 0xBE}

ATQB 的应用数据 = 0x1C0x2D 0x94 0x11

ATQB 的协议信息 =0xF7 0x71 0x85

ATTRIB 的 MBLI = 0x00

5.2.2. 非接触接口的私有 APDU 指令

5.2.2.1. 获取数据命令

获取数据（Get Data）命令依据所插入的卡片，获取卡片的信息。可用于各种非接触卡。

Get Data 的命令结构(5 字节)

命令	CLA	INS	P1	P2	Le
Get Data	0xFF	0xCA	0x00 0x01	0x00	0x00

如果 P1 = 0x00, UID 响应报文结构（UID + 2 个字节）

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

如果 P1 = 0x01, ISO 14443 A 类卡的 ATS 响应结构（ATS + 2 个字节）

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	0x90	0x00	操作成功完成。
警告	0x62	0x82	UID/ATS 的末尾先于 Le 字节到达 (Le 大于 UID 的长度)。

错误	0x6C	XX	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	0x63	0x00	操作失败。
错误	0x6A	0x81	不支持此功能

5.2.2.2. PCSC 2.0 第 3 部分（2.02 及以上版本）的 APDU 命令

PCSC 2.0 第三部分规定的命令用于将数据从应用程序透明地传递给非接触式标签，将接收到的数据透明地返回给应用程序和协议，以及同时切换协议。

5.2.2.2.1. 命令和响应的 APDU 格式

命令格式：

CLA	INS	P1	P2	Lc	命令数据域
0xFF	0xC2	0x00	Function	DataLen	Data[DataLen]

Functions 1 字节

0x00 = 管理会话

0x01 = 透明交换

0x02 = 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 SW2 符合 ISO 7816 的定义。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
0xC0	0x03	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

错误状态	说明
0x00 0x90 0x00	未发生错误
XX 0x62 0x82	数据对象 XX 告警, 请求信息不存在
XX 0x63 0x00	未有信息
XX 0x63 0x01	由于其它数据对象失败, 停止执行
XX 0x6A 0x81	不支持数据对象 XX
XX 0x67 0x00	意外长度的数据对象 XX
XX 0x6A 0x80	意外值的数据对象 XX
XX 0x64 0x00	数据对象 XX 执行错误（没有 IFD 响应）
XX 0x64 0x01	数据对象 XX 执行错误（没有 ICC 响应)
XX 0x6F 0x00	数据对象 XX 失败, 未有准确诊断

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

5.2.2.2.2. 管理会话命令

管理会话（Manage Session）命令用于管理透明会话, 其中包括开启透明会话, 结束透明会话, 管理操作环境, 管理透明会话中 IFD 的性能等。

Manage Session 命令

命令	CLA	INS	P1	P2	Lc	命令数据域
ManageSession	0xFF	0xC2	0x00	0x00	DataLen	DataObject（N 个字节）

其中：

Data Object (1 字节)

标签	数据对象
0x80	版本数据对象
0x81	开始透明会话
0x82	结束透明会话
0x83	关闭 RF 场
0x84	打开 RF 场
0x5F46	计时器
0xFF6D	获取参数
0xFF6E	设置参数

管理会话响应数据对象

标签	数据对象
0xC0	常见错误状态
0x80	版本数据对象
0xFF6D	IFD 参数数据对象

5.2.2.2.2.1. 开始会话数据对象

此命令开始透明会话。会话开始后，自动轮询功能被禁用，直到会话结束。

开始会话数据对象

标签	长度（1 字节）	值
0x81	0x00	-

5.2.2.2.2.2. 终止会话数据对象

此命令终止透明会话。自动轮询会被重置为会话开始前的状态。

终止会话数据对象

标签	长度（1 字节）	值
0x82	0x00	-

5.2.2.2.2.3. 版本数据对象

此命令返回 IFD 处理程序的版本号。

版本数据对象

标签	长度（1 字节）	值		
0x80	0x03	主版本	次版本	内部版本

5.2.2.2.2.4. 关闭 RF 数据对象

此命令关闭天线场

关闭 RF 场数据对象

标签	长度（1 字节）	值
0x83	0x00	-

5.2.2.2.2.5. 开启 RF 数据对象

此命令开启天线场

打开 RF 场数据对象

标签	长度（1 字节）	值
0x84	0x00	-

5.2.2.2.2.6. 计时器数据对象

此命令创建一个 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, 然后再重新开启 RF 场。

计时器数据对象

标签	长度（1 字节）	值
0x5F46	0x04	计时器 (4 字节)

5.2.2.2.2.7. 获取参数数据对象

此命令从 IFD 中获取参数。

获取参数数据对象

标签	长度（1 字节）	值		
		标签	长度	值
0xFF6D	变长	TLV_Objects		

TLV_Objects:

所需参数	标签	长度
IFD 帧长度整数（FSDI）	0x01	0x00
ICC 帧长度整数（FSCI）	0x02	0x00
帧等待时间整数（FWTI）	0x03	0x00
IFD 支持的最高通信速度	0x04	0x00
ICC 通信速度	0x05	0x00
调制指数	0x06	0x00
符合 ISO/IEC14443 的 PCB	0x07	0x00
符合 ISO/IEC14443 的 CID	0x08	0x00
符合 ISO/IEC14443 的 NAD	0x09	0x00
ISO/IEC14443 B 类的参数 1 - 4	0x0A	0x00

5.2.2.2.2.8. 设置参数数据对象

此命令设置 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
0xFF6E	变长	TLV_Objects		

TLV_Objects:

所需参数	标签	长度
IFD 帧长度整数（FSDI）	0x01	0x01
ICC 帧长度整数（FSCI）	0x02	0x01
帧等待时间整数（FWTI）	0x03	0x01
IFD 支持的最高通信速度	0x04	0x01
PICC 的通信速度	0x05	0x01
调制指数	0x06	0x01
符合 ISO/IEC14443 的 PCB	0x07	0x01

所需参数	标签	长度
符合 ISO/IEC14443 的 CID	0x08	0x01
符合 ISO/IEC14443 的 NAD	0x09	0x01
ISO/IEC14443 B 类的参数 1 - 4	0x0A	0x04

5.2.2.2.3. Transparent Exchange 命令

透明交换(Transparent Exchange)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	0xFF	0xC2	0x00	0x01	DataLen	DataObject（N 个字节）

其中：

Data Object (1 字节)

标签	数据对象
0x90	发送和接收标志
0x91	发送位成帧
0x92	接收位成帧
0x93	发送
0x94	接收
0x95	收发 - 发送和接收
0xFF6D	获取参数
0xFF6E	设置参数

透明交换响应数据对象

标签	数据对象
0xC0	常见错误状态
0x92	所接收数据中最后一个字节的有效位数
0x96	响应报文状态字

标签	数据对象
0x97	ICC 响应
0xFF6D	IFD 参数数据对象

5.2.2.2.3.1. 发送和接收标志数据对象

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
0x90	0x02	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

5.2.2.2.3.2. 发送位成帧数据对象

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
0x91	0x01	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。

如果该数据对象不存在, 则表明所有的位都有效。

5.2.2.2.3.3. 接收位成帧数据对象

在命令 APDU 中, 此数据对象用于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, 此数据对象用于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
0x92	0x01	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

如果该数据对象不存在, 则表明所有的位都有效。

5.2.2.2.3.4. 发送数据对象

要从 IFD 向 ICC 传输数据, 传输完成后 ICC 不会返回响应。

发送数据对象

标签	长度（1 字节）	值
0x93	DataLen	数据（N 个字节）

5.2.2.2.3.5. 接收数据对象

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
0x94	0x00	-

5.2.2.2.3.6. 收发数据对象

此数据对象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
0x95	DataLen	数据（N 个字节）

5.2.2.2.3.7. 响应状态数据对象

此数据对象用于提示在响应中接收到的数据状态。

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	
0x96	0x02	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。否则会显示“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	

5.2.2.2.3.8. 响应数据对象

此数据对象用于提示响应中接收到的数据。

响应数据对象

标签	长度（1 字节）	值
0x97	数据长度	响应数据 (N 字节)

5.2.2.2.4. 切换协议命令

切换协议（Switch Protocol）命令用于指定透明会话中的协议和不同标准层。

Switch Protocol 命令

命令	CLA	INS	P1	P2	Lc	命令数据域
SwProtocol	0xFF	0xC2	0x00	0x02	数据长度	数据对象（N 个字节）

其中：

数据对象 (1 字节)

标签	数据对象
0x8F	切换协议数据对象
0xFF6D	获取参数
0xFF6E	设置参数

切换协议响应数据对象

标签	数据对象
0xC0	常见错误状态
0xFF6D	IFD 参数数据对象

5.2.2.2.4.1. 切换协议数据对象

用于指定协议和不同标准层。

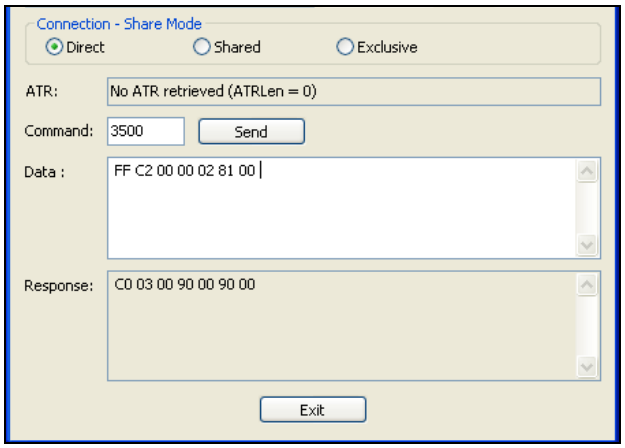
切换协议数据对象

标签	长度（1 字节）	值	
		字节 0	字节 1
0x8F	0x02	0x00 – ISO/IEC14443 Type A 0x01 – ISO/IEC14443 Type B 0x03 – FeliCa 其它 – RFU	0x00 – 如果没有分层 0x02 – 切换到第二层 0x03 – 切换或激活到第三层 0x04 – 激活到第四层 其它 - RFU

5.2.2.2.5. PCSC 2.0 第 3 部分示例

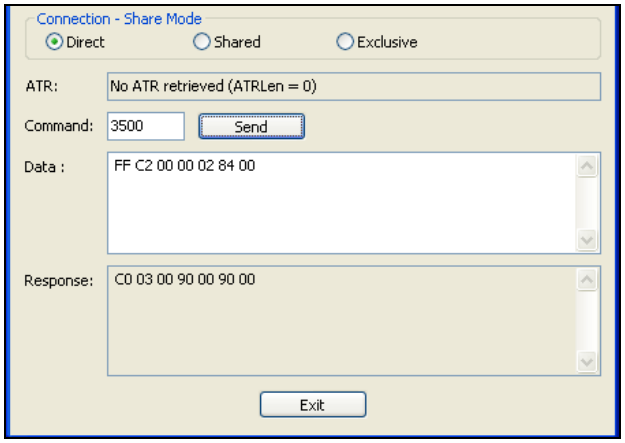
Step 1. 开始透明会话

命令：0xFF 0xC2 0x00 0x00 0x02 0x81 0x00
响应：0xC0 0x03 0x00 0x90 0x00 0x90 0x00



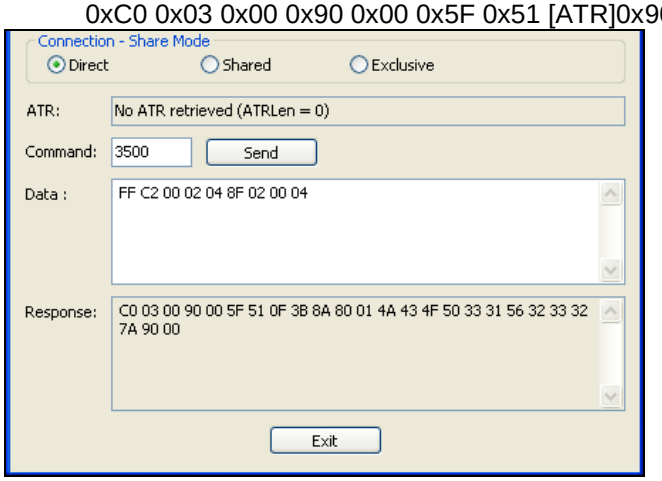
Step 2. 开启天线场

命令：0xFF 0xC2 0x00 0x00 0x02 0x84 0x00
响应：0xC0 0x03 0x00 0x90 0x00 0x90 0x00



Step 3. ISO14443-4A 生效

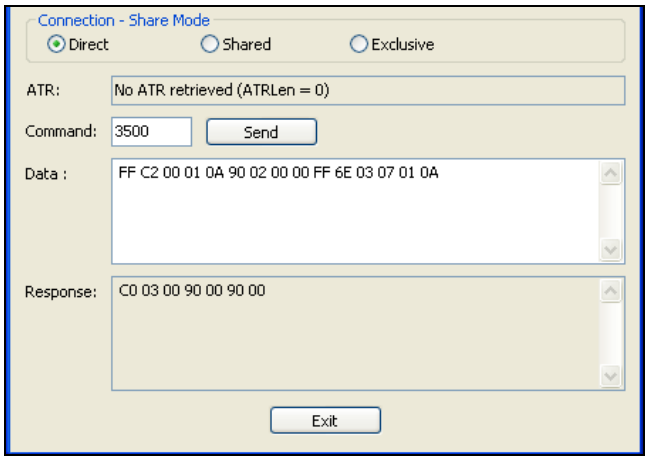
命令：0xFF 0xC2 0x00 0x02 0x04 0x8F 0x02 0x00 0x04
响应：0xC0 0x03 0x01 0x64 0x01 0x90 0x00 (如果卡片不存在)
0xC0 0x03 0x00 0x90 0x00 0x5F 0x51 [ATR]0x90 0x00



Step 4. 将 PCB 设为 0x0A 并在传输数据中启用 CRC, 奇偶校验和协议头。

命令：0xFF 0xC2 0x00 0x01 0x0A 0x90 0x02 0x00 0x00 0xFF 0x6E 0x03 0x07 0x01 0x0A

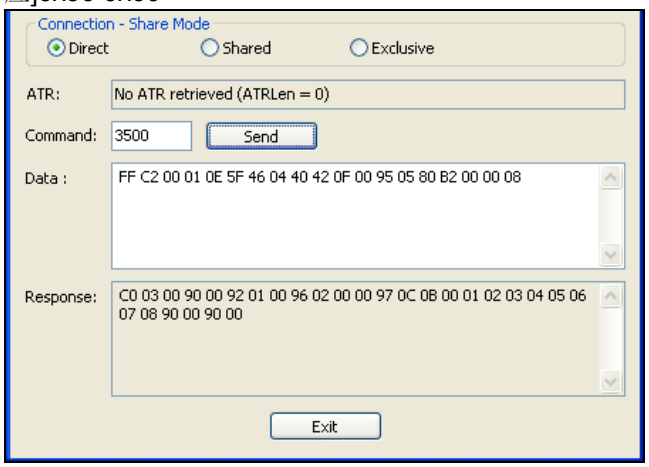
响应：0xC0 0x03 0x00 0x90 0x00 0x90 0x00



Step 5. 发送 APDU “0x80 0xB2 0x00 0x00 0x08”至卡片并取响应

命令：0xFF 0xC2 0x00 0x01 0x0E 0x5F 0x46 0x04 0x40 0x42 0x0F 0x00 0x95 0x05 0x80 0xB2 0x00 0x00 0x08

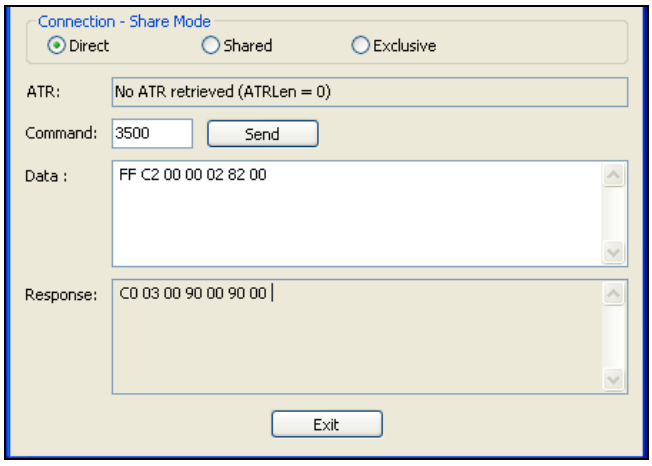
响应：0xC0 0x03 0x00 0x90 0x00 0x92 0x01 0x00 0x96 0x02 0x00 0x00 0x97 0x0C [卡片响应]0x90 0x00



Step 6. 结束透明会话

命令：0xFF 0xC2 0x00 0x00 0x02 0x82 0x00

响应：0xC0 0x03 0x00 0x90 0x00 0x90 0x00



5.2.2.3. Mifare 1K/4K 存储卡的 PICC 命令（T=CL 模拟）

5.2.2.3.1. 加载认证密钥

加载认证密钥（Load Authentication Keys）命令用于向读写器加载认证密钥。该认证密钥用于验证 Mifare 1K/4K 存储卡的特定扇区。AMR220-C1 仅提供了易失密钥存储位置。

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	0xFF	0x82	密钥结构	密钥号	0x06	密钥 (6 字节)

密钥结构（1 个字节）：

0x00 = 密钥被载入读写器的易失存储器。

密钥号（1 个字节）：

0x00 ~ 0x01 = 用于存储临时密钥的易失存储器。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。
默认值 = {0xFF 0xFF 0xFF 0xFF 0xFF 0xFF}

密钥（6 个字节）：

载入读写器的密钥值。例如 {0xFF 0xFF 0xFF 0xFF 0xFF 0xFF}

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 的响应状态码

结果	SW1	SW2	含义
成功	0x90	0x00	操作成功完成。
错误	0x63	0x00	操作失败。

例如：

// 向易失性存储位置 0x00 加载秘钥{0xFF 0xFF 0xFF 0xFF 0xFF FF}。

APDU = {0xFF 0x82 0x00 0x00 0x06 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF}

5.2.2.3.2. MIFARE 1K/4K 卡认证

认证（Authentication）命令用存储在读写器内的密钥来验证 MIFARE 1K/4K 卡。其中会用到两种认证密钥：TYPE_A 和 TYPE_B。

Authentication 的 APDU 结构#1（6 个字节）

命令	CLA	INS	P1	P2	P3	命令数据域
Authentication	0xFF	0x88	0x00	块号	密钥类型	密钥号

Authentication 的 APDU 结构#2（10 个字节）

命令	CLA	INS	P1	P2	Lc	命令数据域
Authentication	0xFF	0x86	0x00	0x00	0x05	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
版本号 0x01	0x00	块号	密钥类型	密钥号

块号（1 个字节）：

待验证的存储块。

Mifare 1K 卡的内存分为 16 个扇区, 每个扇区包含 4 个连续的块。例如：
扇区 0x00 包含块{0x00, 0x01, 0x02 和 0x03}
扇区 0x01 包含块{0x04, 0x05, 0x06 和 0x07}
最后一个扇区 0x0F 包含块{0x3C, 0x3D, 0x3E 和 0x3F}

验证成功后, 读取同一个扇区内的其他块不需要再进行验证。详情请参考 Mifare 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。*

密钥类型（1 个字节）：

0x60 = 该密钥被用作 TYPE A 密钥进行验证。

0x61 = 该密钥被用作 TYPE B 密钥进行验证。

密钥号（1 个字节）：

0x00 ~ 0x01 = 用于存储密钥的易失存储器。读写器与电脑断开连接后, 密钥即会消失。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 的响应状态码

结果	SW1	SW2	含义
成功	0x90	0x00	操作成功完成。
错误	0x63	0x00	操作失败。

MIFARE 1K 卡的内存结构

扇区 (总共 16 个扇区) (每个扇区包含 4 个连续的块)	数据块 (3 个块, 每个块 16 个字节)	尾部块 (1 个块, 16 字节)
扇区 0	0x00 ~ 0x02	0x03
扇区 1	0x04 ~ 0x06	0x07
…	…	…
…	…	…
扇区 14	0x38 ~ 0x0A	0x3B
扇区 15	0x3C ~ 0x3E	0x3F

1K 字节

MIFARE 4K 卡的内存结构

扇区 (总共 32 个扇区) (每个扇区包含 4 个连续的块)	数据块 (3 个块, 每个块 16 个字节)	尾部块 (1 个块, 16 字节)
扇区 0	0x00 ~ 0x02	0x03
扇区 1	0x04 ~ 0x06	0x07
…	…	…
…	…	…
扇区 30	0x78 ~ 0x7A	0x7B
扇区 31	0x7C ~ 0x7E	0x7F

2K 字节

扇区 (总共 8 个扇区) (每个扇区包含 16 个连续的块)	数据块 (15 个块, 每个块 16 个字节)	尾部块 (1 个块, 16 字节)
扇区 32	0x80 ~ 0x8E	0x8F
扇区 33	0x90 ~ 0x9E	0x9F
…	…	…
…	…	…
扇区 38	0xE0 ~ 0xEE	0xEF
扇区 39	0xF0 ~ 0xFE	0xFF

2K 字节

例如：

// 要使用{TYPE A, 密钥号 0x00}验证块 0x04

// PC/SC V2.01, 弃用

APDU = {0xFF 0x88 0x00 0x040x600x00};

// 要使用{TYPE A, 密钥号 0x00}验证块 0x04

// PC/SC V2.07

APDU = {0xFF 0x86 0x00 0x00 0x05 0x01 0x00 0x040x600x00}

注：MIFARE Ultralight 不需要进行验证, 其内存可以自由访问。

MIFARE Ultralight 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	内部	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

64 个字节

5.2.2.3.3. 读二进制块

读二进制块（Read Binary Blocks）命令用于从 Mifare 卡片中取回多个“数据块”。执行此命令前，必须先对数据块/尾部块进行验证。

Read Binary 的 APDU 结构（5 个字节）

命令	CLA	INS	P1	P2	Le
Read Binary Blocks	0xFF	0xB0	0x00	块号	待读取的字节数

块号（1 个字节）：

起始块

待读取的字节数（1 个字节）：

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

- MIFARE Ultralight 卡的待读字节数最大为 16。

- MIFARE 1K 卡的待读字节数最大为 48。（多块模式；3 个连续的块）
- MIFARE 4K 卡的待读字节数最大为 240。（多块模式；15 个连续的块）

例 1：0x10 (16 字节) -> 仅读取起始块。（单块模式）

例 2：0x40 (64 字节) -> 读取起始块至起始+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被访问, 请使用单块模式对其进行访问。*

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

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	0x90	0x00	操作成功完成。
错误	0x63	0x00	操作失败。

例如：

// 从二进制块 0x04 中读取 16 个字节（MIFARE 1K 或 4K）

APDU = {0xFF 0xB0 0x00 0x04 0x10}

// 从二进制块 0x80 开始读取 240 个字节（MIFARE 4K）

// 块 0x80 至块 0x8E（15 个块）

APDU = {0xFF 0xB0 0x00 0x80 0xF0}

5.2.2.3.4. 更新二进制块

更新二进制块（Update Binary Blocks）命令用于向 Mifare 写入多个“数据块”。执行此命令前, 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	0xFF	0xD6	0x00	块号	待更新的字节数	块数据 (16 字节的倍数)

块号（1 个字节）：

待更新的起始块

待更新的字节数（1 个字节）：

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

例 1：0x10 (16 字节) -> 仅写入起始块。（单块模式）

例 2：0x30 (48 字节) -> 写入起始块至起始 + 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访问, 请使用单块模式对其进行访问。*

块数据（16 的倍数 + 2 个字节, 或 6 个字节）：

待写入二进制块的数据。

Update Binary Block 的响应状态码（2 个字节）

结果	SW1	SW2	含义
成功	0x90	0x00	操作成功完成。
错误	0x63	0x00	操作失败。

例如：

// 将 MIFARE 1K/4K 卡中的二进制块 0x04 的数据更新为{0x00 0x01 ..0x0F}

APDU = {0xFF 0xD6 0x00 0x04 0x10 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0A 0x0B 0x0C 0x0D 0x0E 0x0F}

// 将 MIFARE Ultralight 中二进制块 0x04 的数据更新为{0x00 0x01 0x02 0x03}

APDU = {0xFF 0xD6 0x00 0x04 0x04 0x00 0x01 0x02 0x03}

5.2.2.3.5. 值块操作（INC, DEC, STORE）

值块操作（Value Block Operation）命令用于进行数值操作, 例如：增加值块的值等。

Value Block Operation 的 APDU 结构（10 个字节）

命令	CLA	INS	P1	P2	Lc	命令数据域	
Value Block Operation	0xFF	0xD7	0x00	块号	0x05	VB_OP	VB_Value (4 个字节) {MSB ...LSB}

块号（1 个字节）：

待操作的值块

VB_OP（1 个字节）：

0x00 = 将 VB_Value 存入该块。然后该块将变为值块。

0x01 = 使值块的值增加 VB_Value。该命令仅用于操作值块。

0x02 = 使值块的值减少 VB_Value。该命令仅用于操作值块。

VB_Value（4 个字节）：

用于算数运算的数值, 是一个有符号长整数（4 个字节）。

例如 1：Decimal “–4” = {0xFF, 0xFF, 0xFF, 0xFC}

VB_Value			
MSB			LSB
0xFF	0xFF	0xFF	0xFC

例如 2：Decimal “1” = {0x00, 0x00, 0x00, 0x01}

VB_Value			
MSB			LSB
0x00	0x00	0x00	0x01

Value Block Operation 的响应结构（2 个字节）

响应	响应数据域	
结果	SW1	SW2

Value Block Operation 的响应状态码

结果	SW1	SW2	含义
成功	0x90	0x00	操作成功完成。
错误	0x63	0x00	操作失败。

5.2.2.3.6. 读取值块

读取值块（Read Value Block）命令用于获取值块中的数值, 仅用于操作值块。

Value Block Operation 的 APDU 结构（5 字节）

命令	CLA	INS	P1	P2	Le
Read Value Block	0xFF	0xB1	0x00	块号	0x04

块号（1 个字节）：

待访问的值块。

Read Value Block 的响应结构(4 + 2 个字节)

响应	响应数据域		
结果	值 {MSB ..LSB}	SW1	SW2

值（4 个字节）：

卡片返回的数值, 是一个有符号长整数（4 个字节）。

例如 1：Decimal “–4” = {0xFF, 0xFF, 0xFF, 0xFC}

值			
MSB			LSB
0xFF	0xFF	0xFF	0xFC

例如 2：Decimal “1” = {0x00, 0x00, 0x00, 0x01}

值			
MSB			LSB
0x00	0x00	0x00	0x01

Read Value Block 命令的响应状态码

结果	SW1	SW2	含义
成功	0x90	0x00	操作成功完成。
错误	0x63	0x00	操作失败。

5.2.2.3.7. 复制值块

复制值块（Copy Value Block）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 命令的 APDU 结构（7 字节）

命令	CLA	INS	P1	P2	Lc	命令数据域	
Copy Value Block	0xFF	0xD7	0x00	源块号	0x02	0x03	目标块号

源块号（1 个字节）：

源值块中的值会被复制到目标值块。

目标块号（1 个字节）：

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 字节）

响应	响应数据域	
结果	SW1	SW2

Copy Value Block 的响应状态码

结果	SW1	SW2	含义
成功	0x90	0x00	操作成功完成。
错误	0x63	0x00	操作失败。

5.2.2.4. 访问符合 PCSC 的标签（ISO14443-4）

基本上，所有符合 ISO14443-4 标准的卡片（PICC 卡）都可以理解 ISO 7816-4 规定的 APDU。AMR220-C1 读写器与符合 ISO 14443-4 标准的卡片进行通信时，只需要交换 ISO 7816-4 规定的 APDU 和响应。AMR220-C1 会在内部处理 ISO 14443 第 1-4 部分协议。

另外 MIFARE 1K，4K，MINI 和 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 ISO14443-4 标签即可。

更多相关信息，请参阅 **Mifare 1K/4K 存储卡的 PICC 命令（T=CL 模拟）**。

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 7816-4 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 个字节）

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	0x90	0x00	操作成功完成。
错误	0x63	0x00	操作失败。

典型的操作顺序是：

- 出示标签并连接
- 读取/更新标签的存储内容

操作举例：

Step 1. 连接标签

标签的 ATR 是 0x3B 0x880x80 0x01 0x00 0x00 0x00 0x00 0x33 0x81 0x81 0x00 0x3A

其中, 这是一个 ISO14443-4 TypeB 标签,

ATQB 的应用数据 = 0x00 0x00 0x00 0x00

ATQB 的协议信息 = 0x33 0x81 0x81

Step 2. 发送 APDU, 例如, 取随机数

CMD: 0x00 0x84 0x00 0x00 0x08

RSP: 0x1A 0xF7 0xF3 0x1B 0xCD 0x2B 0xA9 0x58 [0x90 0x00]

注：*对于 ISO14443-4 A 类标签来说，可以通过 APDU"0xFF 0xCA 0x01 0x00 0x00"来获取 ATS。*

5.2.2.5. 访问 FeliCa 标签

访问 FeliCa 标签的命令与访问 PCSC 标签和 MIFARE 卡的命令有所不同。这些命令符合 FeliCa 规范，加了一个命令头，格式如下。

FeliCa 命令结构

命令	CLA	INS	P1	P2	Lc	命令数据域
FeliCa 命令	0xFF	0x00	0x00	0x00	命令数据域的长度	FeliCa 命令（开始于长度字节）

FeliCa 命令：

请参阅 FeliCa 卡片标准。

例如：

例 1 轮询命令

= {0x06, 0x00, 0xFF, 0xFF, 0x00, 0x00}

其中

0x00 = 轮询命令代码
0xFF 0xFF = 系统代码

例 2 非加密读取命令

= {0x10 0x06 0x01 0x01 0x06 0x01 0xCB 0x09 0x57 0x03 0x01 0x09 0x01 0x01 0x80 0x00}

其中

0x06 = 非加密读取命令代码
0x01 0x01 0x06 0x01 0xCB 0x09 0x57 0x03= Felica IDm (取决于卡片)
0x09 0x01= 服务代码
0x80 0x00 = 块

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	SW1	SW2

响应状态码

结果	SW1	SW2	含义
成功	0x90	0x00	操作成功完成。
错误	0x67	0x00	长度错误
	0x64	0x01	操作失败

操作举例：

Step 1. 连接 FeliCa

ATR = 0x3B 0x8F 0x80 0x01 0x80 0x4F 0x0C 0xA0 0x00 0x00 0x03 0x06 0x11 0x00 0x3B0x00 0x00 0x00 0x00 0x42

其中 11 00 3B = FeliCa

Step 2. 读取 FeliCa IDM

CMD = 0xFF 0xCA 0x00 0x00 0x00

RSP = [IDM (8bytes)] 0x90 0x00

例如：FeliCa IDM = **0x01 0x01 0x06 0x01 0xCB 0x09 0x57 0x03**

Step 3. FeliCa 命令访问（非加密读取命令的使用范例）

CMD = 0xFF0x00 0x00 0x00 0x10 0x10 0x06 **0x01 0x01 0x06 0x01 0xCB 0x09 0x57 0x03**0x01 0x09 0x01 0x01 0x80 0x00

其中

Felica 命令 = 0x10 0x06 **0x01 0x01 0x06 0x01 0xCB 0x09 0x57 0x03**0x01 0x09 0x01 0x01 0x80 0x00

Felica IDm = 0x01 0x01 0x06 0x01 0xCB 0x09 0x57 0x03

RSP = 0x1D 0x07 **0x01 0x01 0x06 0x01 0xCB 0x09 0x57 0x03** 0x00 0x00 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x90 0x00

其中

响应代码 = 0x07

Felica IDm = 0x01 0x01 0x06 0x01 0xCB 0x09 0x57 0x03

状态标识 = 0x00 0x00

块数据 = 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

5.3. 直接命令

直接（Escape）命令用于控制外设或进行特殊操作。

命令通过 dwControlCode = SCARD_CTL_CODE(3500) 的 SPH_to_RDR_ExEscape 或 PCSC SCardControl 发送。

5.3.1. 获取固件版本

获取固件版本（Get Firmware Version）命令用于获取 AMR220-C1 的固件信息。

Get Firmware Version 的结构（4 个字节）

命令	CLA	INS	P1	P2
Get Firmware Version	0xFC	0x00	0xA1	0xFF

Get Firmware Version 的响应结构（3 个字节 + 固件信息的长度）

响应				响应数据域
结果	0x00	0x30	0x30	固件版本号

例如：响应 = 0x00 0x30 0x30 0x31 0x2E 0x30 0x2E 0x31 0x34

固件版本(HEX) = 0x31 0x2E 0x30 0x2E 0x31 0x34

固件版本(ASCII) = “1.0.14”

5.3.2. 休眠模式选项

设置休眠时间间隔（Set Sleep Time Interval）命令用于获取/设置 AMR220-C1 进入休眠模式前的闲置时间。

设置 Sleep Time Interval 的命令格式(6 个字节)

命令	CLA	INS	P1	P2	Lc	命令数据域
Sleep Time Interval	0xE0	0x00	0x00	0x48	0x01	关闭时间

或

获取 Sleep Time Interval 的命令格式(5 个字节)

命令	CLA	INS	P1	P2	Lc
Sleep Time Interval	0xE0	0x00	0x00	0x48	0x00

Sleep Time Interval 的响应结构(6 个字节)

响应	CLA	INS	P1	P2	Le	响应数据域
结果	0xE1	0x00	0x00	0x00	0x01	关闭时间

关闭时间(1 字节)：以秒为单位

Data In = 01 至 FF

默认 = 0x78 (120 秒)

5.3.3. 天线场控制

天线场控制（Antenna Field Control）命令用于控制天线场。

注：*天线场受自动轮询影响。*

Antenna Field Control 命令格式（6 字节）

命令	CLA	INS	P1	P2	Lc	命令数据域
Antenna Field Control	0xE0	0x00	0x00	0x41	0x01	0x00 – 天线关闭 0x01 – 天线开启

Antenna Field Control 的响应结构（6 字节）

响应	CLA	INS	P1	P2	Le	响应数据域
结果	0xE1	0x00	0x00	0x00	0x01	0x00 – 天线关闭 0x01 – 天线开启

5.3.4. 自动 PICC 轮询

自动 PICC 轮询（Automatic PICC Polling）命令用于设置使用 USB 进行通讯时读写器的轮询模式。

每当 AMR220-C1 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。

我们可以发送一个命令来停用 PICC 轮询功能。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，读写器的电能消耗更低。

注:

1. *自动轮询功能仅用于 PICC 模式。*
2. *AMR220-C1 重置时，会被设置成默认值。*

设置 Automatic PICC Polling 的命令格式（6 个字节）

命令	CLA	INS	P1	P2	Lc	命令数据域
Automatic PICC Polling	0xE0	0x00	0x00	0x23	0x01	轮询设置

或

获取 Automatic PICC Polling 的命令格式（5 个字节）

命令	CLA	INS	P1	P2	Lc
Automatic PICC Polling	0xE0	0x00	0x00	0x23	0x00

Automatic PICC Polling 的响应结构（6 个字节）

响应	CLA	INS	P1	P2	Le	响应数据域
结果	0xE1	0x00	0x00	0x00	0x01	轮询设置

轮询设置（1 个字节）：

轮询设置	说明	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。

**轮询设置参数的默认值 = 0x8B*

注：

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。*
3. *读写器会自动激活"ISO14443A-4 PICC"的 ISO 14443A-4 模式。Type B PICC 不受此选项影响。*
4. *JCOP30 卡片有两种模式：ISO14443A-3 (MIFARE 1K)和 ISO14443A-4 模式。一旦 PICC 被激活，应用就必须选定一种模式。*

5.3.5. PICC 操作参数

PICC 操作参数（PICC Operating Parameter）命令用于设置自动轮询的检测卡片类型。

注:

1. *自动轮询功能仅用于 PICC 模式。*
2. *AMR220-C1 重置时, 会被设置成默认值。*

设置 PICC Operating Parameter 的命令格式（6 个字节）

命令	CLA	INS	P1	P2	Lc	命令数据域
PICC Operating Parameter	0xE0	0x00	0x00	0x20	0x01	卡片类型

或

获取 PICC Operating Parameter 的命令格式（5 个字节）

命令	CLA	INS	P1	P2	Lc
PICC Operating Parameter	0xE0	0x00	0x00	0x20	0x00

PICC Operating Parameter 的响应格式（6 个字节）

响应	CLA	INS	P1	P2	Le	响应数据域
结果	0xE1	0x00	0x00	0x00	0x01	卡片类型

卡片类型（1 个字节）：

卡片类型	参数	说明	选项
Bit0	ISO14443 Type A	PICC 轮询要检测的标签类别	1 = 检测 0 = 跳过
Bit1	ISO14443 Type B		1 = 检测 0 = 跳过
Bit2	Felica 212kbps		1 = 检测 0 = 跳过
Bit3	Felica 424kbps		1 = 检测 0 = 跳过
Bit4 - 7	RFU	RFU	RFU

**卡片类型默认值 = 0x0F*

5.3.6. 蜂鸣器控制

蜂鸣器控制（Buzzer Control）命令用于控制蜂鸣器声音。

Buzzer Control 的格式 #1 (6 个字节)

命令	CLA	INS	P1	P2	Lc	命令数据域
Buzzer Control	0xE0	0x00	0x00	0x28	0x01	蜂鸣器鸣响时间

蜂鸣器鸣响时间（1 个字节）：

0x00 = 关闭

0x01 - 0xFF = 持续时间 (单位：10ms), 频率 = 1500Hz

或

Buzzer Control 的格式 #2 (8 个字节)

命令	CLA	INS	P1	P2	Lc	命令数据域		
Buzzer Control	0xE0	0x00	0x00	0x28	0x03	蜂鸣器鸣响时间	蜂鸣器关闭时间	重复次数

蜂鸣器鸣响时间（1 个字节）：

0x00 = 关闭

0x01 - 0xFF = 开启时间 (单位：10ms), 频率 = 1500Hz

蜂鸣器关闭时间（1 个字节）：

0x00 = 开启, 频率 = 1500Hz

0x01 - 0xFF = 关闭时间 (单位：10ms）

重复次数 (1 个字节)：

开启和关闭模式重复次数

或

Buzzer Control 的格式 #3 (12 个字节)

命令	CLA	INS	P1	P2	Lc	命令数据域			
Buzzer Control	0xE0	0x00	0x00	0x28	0x07	蜂鸣器鸣响时间	蜂鸣器关闭时间	重复次数	频率 (MSB … LSB)

蜂鸣器鸣响时间（1 个字节）：

0x00 = 关闭

0x01 - 0xFF = 开启时间 (单位：10ms）

蜂鸣器关闭时间（1 个字节）：

0x00 = Turn ON

0x01 - 0xFF = 关闭时间 (单位：10ms）

重复次数 (1 个字节)：

开启和关闭模式重复次数

频率 (4 个字节)：

蜂鸣器输出频率
频率 = 1500 -> 1500Hz
频率 = 750 -> 750Hz
频率 = 其它值 -> RFU

Buzzer Control 的响应格式（6 个字节）

响应	CLA	INS	P1	P2	Le	响应数据域
结果	0xE1	0x00	0x00	0x00	0x01	0x00

5.3.7. LED 控制

LED 控制（LED Control）命令用于控制 LED。

设置 LED Control 的命令结构（6 个字节）

命令	CLA	INS	P1	P2	Lc	命令数据域
LED Control	0xE0	0x00	0x00	0x29	0x01	LED 状态

或

获取 LED Control 的命令结构（5 个字节）

命令	CLA	INS	P1	P2	Lc
LED Control	0xE0	0x00	0x00	0x29	0x00

LED Control 的响应结构（6 个字节）

响应	CLA	INS	P1	P2	Le	响应数据域
结果	0xE1	0x00	0x00	0x00	0x01	LED 状态

LED 状态 (1 个字节)：

轮询设置	说明	说明
Bit 0	绿色 1 LED	1 = 开；0 = 关
Bit 1	绿色 2 LED	1 = 开；0 = 关
Bit 2	绿色 3 LED	1 = 开；0 = 关
Bit 3	绿色 4 LED	1 = 开；0 = 关
Bit 4 ...7	RFU	RFU