Advanced Card Systems Ltd.
Card & Reader Technologies

ACR1251T

便携式 NFC 读写器 II

（USB 接口）

参考手册 V1.02

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

版本历史

发布日期	修订说明	版本号
2018/6/4	• 初始发布	1.00
2020/1/10	• 新增 MIFARE 商标及归属说明 • 更新 2.0 节：**特性** • 更新 5.1 节：PCSC API • 更新 5.3.6 节：设置自动 PICC 轮询（Set Automatic PICC Polling） • 更新 5.3.7 节：读取自动 PICC 轮询（Read Automatic PICC Polling） • 更新 5.3.10 节：设置自动 PPS（Set Auto PPS） • 更新 5.3.11 节：读取自动 PPS（Read Auto PPS） • 重新归类 5.2 节与非接触智能卡相关的命令	1.01
2020/08/28	• 更新 5.3.4 节：设置 LED 状态指示器操作（Set LED Status Indicator Behavior） • 更新 5.3.8 节：设置 PICC 操作参数（Set PICC Operating Parameter） • 更新 5.3.9 节：读取 PICC 操作参数（Read PICC Operating Parameter）	1.02

目录

图目录

表目录

1.0. 简介

ACR1251T 是一款基于 13.56 MHz 非接触技术研发的智能卡读写器，是 ACR1251U 连机 NFC 智能卡读写器的便携版。它继全球首款符合 CCID 标准的非接触式 NFC 读写器 ACR122U 的便携版 ACR122T 之后，为用户提供了更多先进功能，它既支持 ISO 14443 A 类和 B 类卡，也支持 MIFARE®, FeliCa 以及全部四种 NFC 标签和设备。

做为计算机与卡片的中间设备，ACR1251T 会执行来自于计算机的命令，专门与非接触式标签或设备部件（LED ）进行通信。其 PICC 读写器接口符合 PC/SC 规范。本参考手册将详细介绍如何执行 PC/SC APDU 命令来支持非接触式接口并控制 ACR1251T 的外围设备。

2.0. 特性

- USB 全速接口
- 符合 CCID 标准
- 智能卡读写器：
  - 非接触接口：
    - 读写速率可达 424 Kbps
    - 内置天线用于读写非接触式标签，读取智能卡的距离达 30 mm（视标签的类型而定）
    - 支持 ISO 14443 第 4 部分 A 类和 B 类卡、MIFARE Classic®卡、FeliCa 卡和全部四种 NFC 标签（ISO/IEC 18092）
    - 内建防冲突特性（任何时候只能访问 1 张标签）
    - NFC 支持：
      - 卡片读/写模式
- 内置外设：
  - 用户可控的双色 LED 指示灯
- 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）
- 具有 USB 固件升级能力
- 支持 Android™ 3.1 及更高版本[1]
- 符合下列标准：
  - EN 60950/IEC 60950
  - ISO 14443
  - ISO 18092
  - PC/SC
  - CCID
  - CE
  - FCC
  - RoHS
  - REACH
  - VCCI（日本）
  - MIC（日本）
  - Microsoft® WHQL

[1] *使用 ACS 定义的安卓库*

3.0. 缩略语

缩略语	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	取消选择请求和取消选择响应
PSL	参数选择请求和参数选择响应
RLS	释放请求和释放响应
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
Gi	发起人可选信息域
PFB	交易控制信息
FSL	帧长度的最大值
LLCP	逻辑链路控制协议

表1 ： 缩略语

4.0. 架构

ACR1251T 与计算机之间的数据通讯采用 CCID 协议。 PICC 间的通信符合 PC/SC 标准。

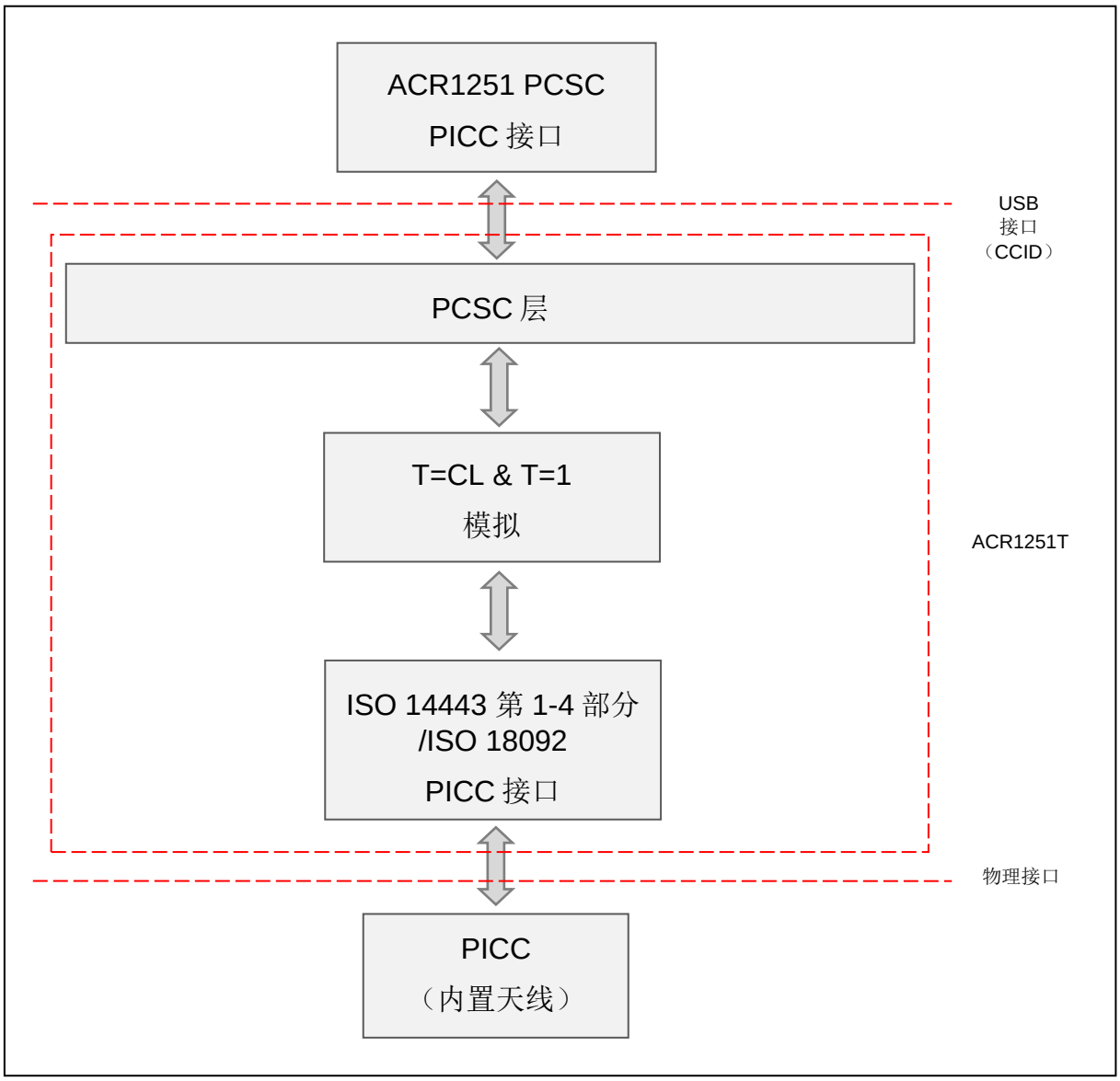


图1 ：ACR1251T 的结构

5.0. 主机编程（连机）API

5.1. PCSC API

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

5.1.1. 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

执行其他 PCSC 操作前，应当执行此函数。.

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
   }
}
```

5.1.2. 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

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
      }
   }
}
```

5.1.3. 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

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 ACR1251 CL Reader PICC 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
         }
      }
   }
```

5.1.4. 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: 外设控制 *节的命令使用此 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 ACR1251 CL Reader PICC 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
            }
```

5.1.5. 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。

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 ACR1251 CL Reader PICC 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];
           }
        }
```

5.1.6. 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

此函数会终止 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
      }
   }
}
```

5.1.7. APDU 流程图

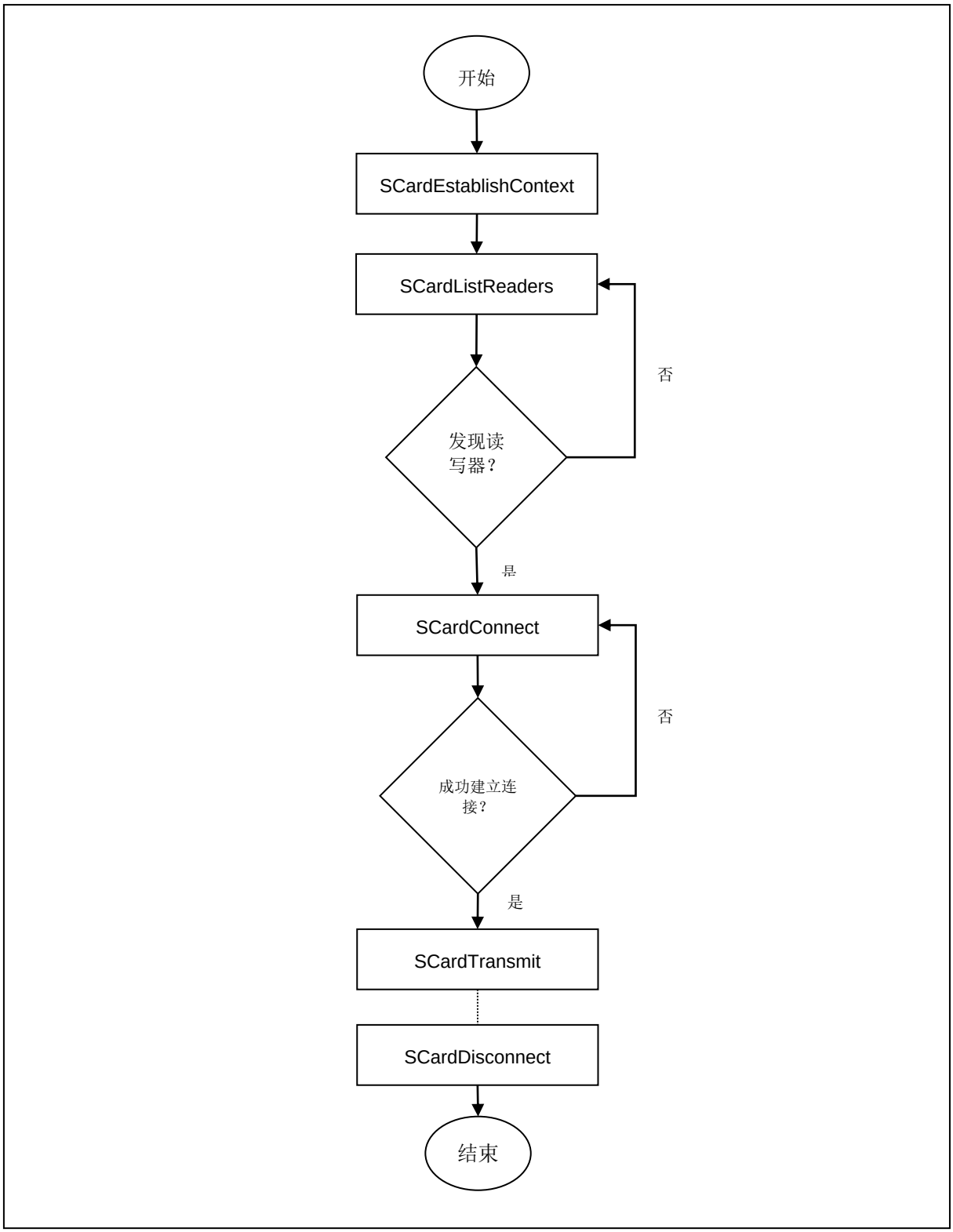


图2 ： ACR1251T APDU 流程图

5.1.8. 直接命令（Escape Command）流程图

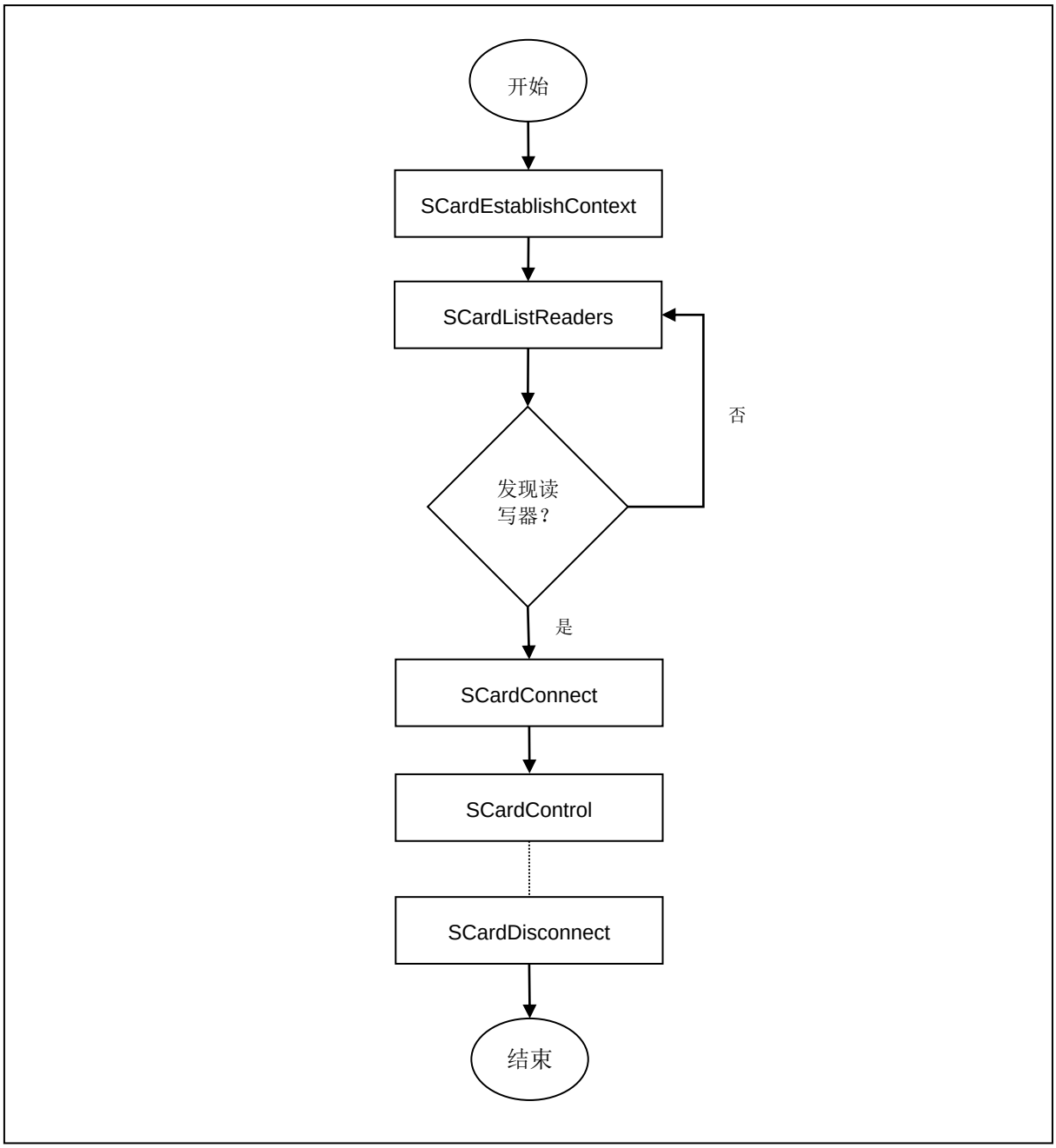


图3 ：ACR1251T 直接命令流程图

5.2. 非接触式智能卡协议

5.2.1. 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。

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

字节	值	标记	说明
0	3Bh	初始头部	
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 表示历史字符的个数（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

例如：

MIFARE Classic 1K 卡的 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 02: MIFARE Classic 4K
00 03: MIFARE Ultralight®
00 26: MIFARE Mini®
00 3A: MIFARE Ultralight® C
00 36: MIFARE Plus® SL1 2K
00 37: MIFARE Plus® SL1 4K
00 38: MIFARE Plus® SL2 2K
00 39: MIFARE Plus® SL2 4K
00 30: Topaz 和 Jewel
00 3B: FeliCa
FF 28: JCOP 30
FF [SAK]: 未定义标签

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

<table>
<tr><th>字节</th><th>值</th><th>标记</th><th>说明</th></tr>
<tr><td>0</td><td>3Bh</td><td>初始头部</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 表示历史字符的个数（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 字节 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

5.2.2. 非接触接口的私有 APDU 指令

5.2.2.1. 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）

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

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

如果 P1 = 01h，获取 ISO 14443 A 类卡的 ATS（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	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};
```

5.2.3. PCSC 2.0 第 3 部分的 APDU 命令（版本 2.02 或更高）

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。

5.2.3.1. 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	数据长度	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

响应格式

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
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 SW1 SW2 值。

如果 C-APDU 数据域中存在多个数据对象，并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。

5.2.3.2. 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（Manage Session）命令

命令	CLA	INS	P1	P2	Lc	命令数据域
Manage Session	FFh	C2h	00h	00h	数据长度	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

会话管理响应数据对象

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

5.2.3.2.1. 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	-

5.2.3.2.2. 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	-

5.2.3.2.3. 版本数据对象

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

5.2.3.2.4. 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	-

5.2.3.2.5. 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	-

5.2.3.2.6. 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）

5.2.3.2.7. 获取参数数据对象

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

5.2.3.2.8. 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

5.2.3.3. 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

5.2.3.3.1. 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

5.2.3.3.2. 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。

5.2.3.3.3. 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。

5.2.3.3.4. 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）

5.2.3.3.5. 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	-

5.2.3.3.6. 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）

5.2.3.3.7. 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	

5.2.3.3.8. 响应数据对象

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)

5.2.3.4. 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

5.2.3.4.1. 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（Switch Protocol Data Object）

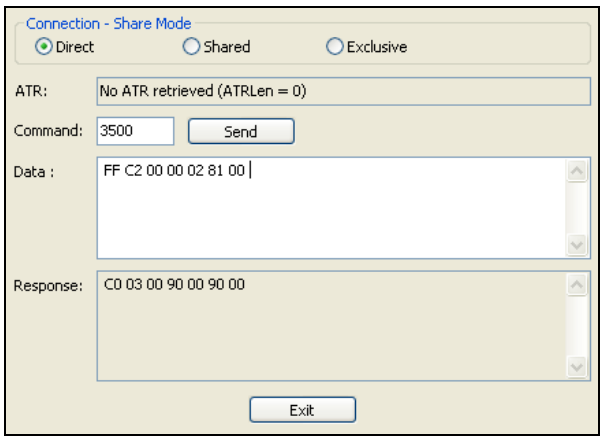
标签	长度（1 字节）	值	
		字节 0	字节 1
8Fh	02h	00h – ISO/IEC14443 Type A 01h – ISO/IEC14443 Type B 03h – FeliCa 其它 – RFU	00h – 如果没有分层 02h – 切换到第二层 03h – 切换并激活到第三层 04h – 激活到第四层 其它 - RFU

5.2.3.5. PCSC 2.0 第 3 部分示例

1. 开始透明会话

   命令：**FF C2 00 00 02 81 00**

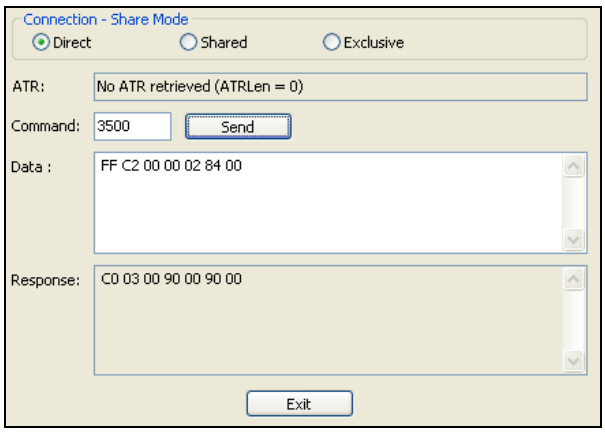
   响应：**C0 03 00 90 00 90 00**



2. 打开天线场

   命令：**FF C2 00 00 02 84 00**

   响应：**C0 03 00 90 00 90 00**

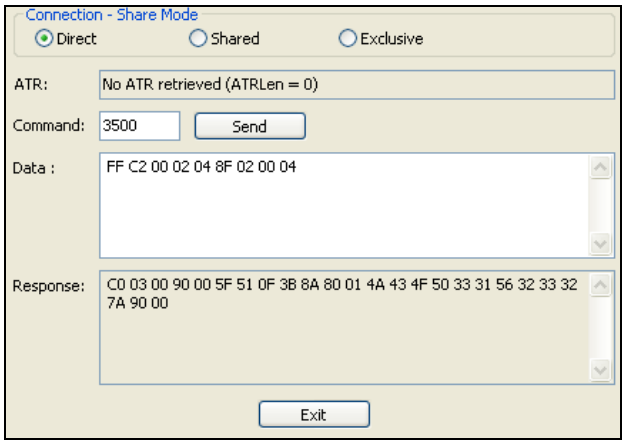


3. ISO 14443-4A 有效。

   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）

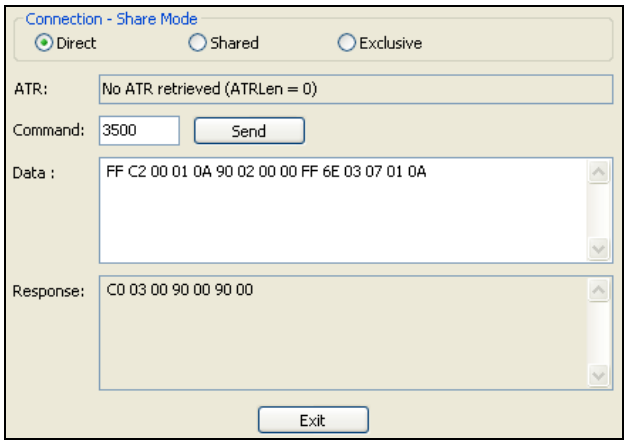
   C0 03 00 90 00 5F 51 [ATR] 90 00



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**

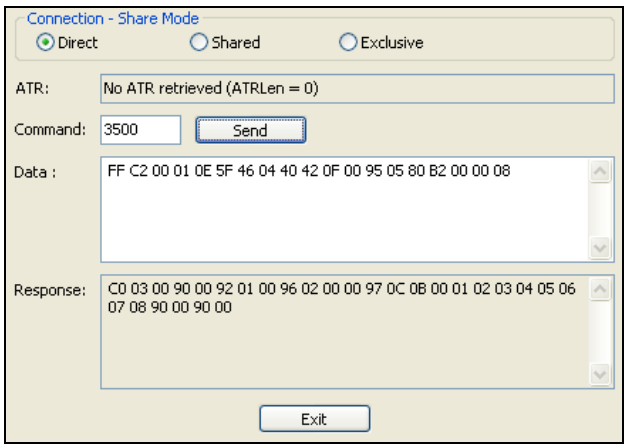
   响应：**C0 03 00 90 00 90 00**



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**

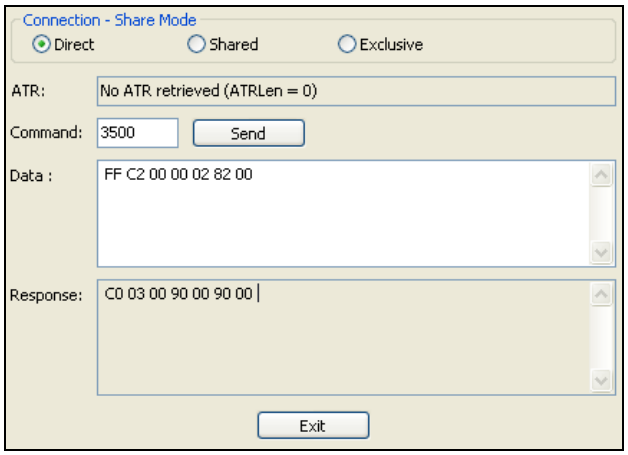
   响应：**C0 03 00 90 00 92 01 00 96 02 00 00 97 0C [卡片响应] 90 00**



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**



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

5.2.4.1. 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 = {FF FF FF FF FF FFh}*

密钥 6 个字节。

载入读写器的密钥值。例如：FF FF FF FF FF FFh

Load Authentication Keys 的响应结构（2 个字节）

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	90h	00h	操作成功完成。
错误	63h	00h	操作失败。

例如：

// 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}

5.2.4.2. 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）。其中用到两种认证密钥：TYPE_A 和 TYPE_B。

Load Authentication Keys 的 APDU 结构（6 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 字节）

字节 1	字节 2	字节 3	字节 4	字节 5
版本号 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，读取同一扇区内的其他块不需要再次进行验证。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。*

密钥类型 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

密钥号 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 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 字节)	尾部块 (1 个块，16 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

1K 字节

表2 ： 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 字节)	尾部块 (1 个块，16 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 字节)	尾部块 (1 个块，16 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3 ： 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	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

512 位
或
64 个字节

表4 ：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，密钥号 00h}验证块 04h。PC/SC V2.07

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

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

5.2.4.3. 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 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，多块模式仅用于访问数据块。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 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 个字节（MIFARE Classic 4K）

// 块 80h 至块 8Eh（15 个块）

APDU = FF B0 00 80 F0h

5.2.4.4. 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 卡的待更新字节的长度应该是 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 个字节，或 6 个字节。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 字节）。从起始块至起始+2 块。（多块模式）

注：*出于安全原因，多块模式仅用于访问数据块。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访问。*

Update Binary Block 的响应状态码（2 个字节）

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

5.2.4.5. 值块操作命令（加、减、保存）（Value Block Operation）

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 个字节）{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，然后该块将变为值块。

01h = 使值块的值增加 VB_Value，该命令仅用于操作值块。

02h = 使值块的值减少 VB_Value，该命令仅用于操作值块。

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 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	90 00h	操作成功完成。
错误	63 00h	操作失败。

5.2.4.6. 读取值块（Read Value Block）

此命令用于获取值块中的数值，该命令仅用于操作值块。

Read Value Block 的 APDU 结构（5 个字节）

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

其中：

块号 1 字节。待访问的值块

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。

5.2.4.7. 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 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。

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}

5.2.5. 访问符合 PCSC 的标签（ISO 14443-4）

基本上，所有符合 ISO 14443-4 标准的卡片（PICC 卡）都可以理解 ISO 7816-4 规定的 APDU。ACR1251T 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。ACR1251T 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 **5.2.2 节**。

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	含义
成功	90 00h	操作成功完成。
错误	63 00h	操作失败。

典型的操作顺序是：

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。

要实现这一点：

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 应用数据 = 00 00 00 00，ATQB 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。*

例如：

// 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]

5.2.6. 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。命令符合 FeliCa 规范，加了一个命令头。

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 字节）

响应	响应数据域
结果	响应数据

以读取内存块为例：

1. 与 FeliCa 建立连接。

   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。

   命令 = FF CA 00 00 00h

   响应 = [IDM (8 字节)] 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。

   命令 = 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

   响应 = 内存块数据

5.3. 外设控制

读写器的外设控制命令通过***控制码为 SCARD_CTL_CODE(3500)的 SCardControl*** 来实现。

5.3.1. 获取固件版本（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

例如：

响应 = E1 00 00 00 0F 41 43 52 31 32 35 31 55 5F 56 36 32 31 2E 30

固件版本号 (HEX) = 41 43 52 31 32 35 31 55 5F 56 36 32 31 2E 30

固件版本号 (ASCII) = “ACR1251T_V621.0”

5.3.2. 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 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	说明	说明
Bit 0	红色 LED	1 = 开；0 = 关
Bit 1	绿色 LED	1 = 开；0 = 关
Bit 2 - 7	RFU	RFU

5.3.3. 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	说明	说明
Bit 0	红色 LED	1 = 开；0 = 关
Bit 1	绿色 LED	1 = 开；0 = 关
Bit 2 - 7	RFU	RFU

5.3.4. 设置 LED 状态指示器操作（Set LED Status Indicator Behavior）

此命令用于设置 LED 作为状态指示器的各种操作。

注：*该设置将保存在非易失存储器中。*

Set LED Status Indicator Behavior 的命令结构（6 字节）

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

操作（1 字节）

操作	模式	说明
Bit 0	卡片操作闪烁 LED	LED 在 PICC 卡片被读写时会闪烁。
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 界面的激活状态 1 = 启用；0 = 禁用
Bit 3 – 5	RFU	RFU
Bit 6	选择颜色（绿色）	绿色 LED 表示状态更改 1 = 启用；0 = 禁用
Bit 7	选择颜色（红色）	红色 LED 表示状态更改 1 = 启用；0 = 禁用

注：*默认操作值 = 47h.*

Set LED 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	默认操作

5.3.5. 读取 LED 状态指示灯操作（Read LED Status Indicator Behavior）

此命令用于读取 LED 的当前默认操作。

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

命令	CLA	INS	P1	P2	Lc
Read LED Status Indicator Behavior	E0h	00h	00h	21h	00h

Read LED 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	卡片操作闪烁 LED	LED 在 PICC 卡片被读写时会闪烁。
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 界面的激活状态 1 = 启用；0 = 禁用
Bit 3 - 5	RFU	RFU
Bit 6	选择颜色（绿色）	绿色 LED 表示状态更改 1 = 启用；0 = 禁用
Bit 7	选择颜色（红色）	红色 LED 表示状态更改 1 = 启用；0 = 禁用

注：*默认操作值 = 47h.*

5.3.6. 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上，读写器的 PICC 轮询功能就会启动 PICC 扫描，以确定是否有 PICC 被放置于/移出了内置天线的范围。

您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：*该设置将保存在非易失存储器中。*

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	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	RFU
Bit 7	强制执行 ISO 14443-A 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 130mA。**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。*

5.3.7. 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	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	RFU
Bit 7	强制执行 ISO 14443-A 第 4 部分	1 = 启用；0 = 禁用。

注：*轮询设置的默认值 = 8Bh。*

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

此命令用于设置 PICC 操作参数。

注：*该设置将保存在非易失存储器中。*

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

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

Set 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 - 7	RFU	RFU	RFU

注：*操作参数的默认值 = 1Fh。*

5.3.9. 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 PICC Operating Parameter 的命令结构（5 字节）

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

Read 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 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 - 7	RFU	RFU	RFU

注：*操作参数的默认值 = 1Fh。*

5.3.10. 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。

Set Auto PPS 的命令结构（7 字节）

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

Set Auto PPS 的响应结构（9 字节）

响应	CLA	INS	P1	P2	Le	响应数据域			
结果	E1h	00h	00h	00h	04h	Max Tx Speed	Current Tx Speed	Max Rx Speed	Current Rx Speed

其中：

Max Tx Speed 最大 Tx 速度（1 字节）

Current Tx Speed 当前 Tx 速度（1 字节）

Max Rx Speed 最大 Rx 速度（1 字节）

Current Rx Speed 当前 Rx 速度（1 字节）

00h = 106 Kbps；默认设置，等同于没有设置自动 PPS

01h = 212 Kbps

02h = 424 Kbps

注：

1. *通常来讲，应用程序应当知道正在被使用的 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。读写器只是使用建议的通信速率来与 PICC 进行对话。如果 PICC 或周围环境不能满足建议的通信速率的要求，PICC 将不可访问。*
2. *读写器支持不同的数据发送速度和接收速度。*

5.3.11. 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 字节）

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

Read Auto PPS 的响应结构（9 字节）

响应	CLA	INS	P1	P2	Le	响应数据域			
结果	E1h	00h	00h	00h	04h	Max Tx Speed	Current Tx Speed	Max Rx Speed	Current Rx Speed

其中：

Max Tx Speed 最大 Tx 速度（1 字节）

Current Tx Speed 当前 Tx 速度（1 字节）

Max Rx Speed 最大 Rx 速度（1 字节）

Current Rx Speed 当前 Rx 速度（1 字节）

00h = 106 Kbps；默认设置，等同于没有设置自动 PPS

01h = 212 Kbps

02h = 424 Kbps

5.4. ACR122U 兼容命令

5.4.1. 双色 LED 控制（Bi-color LED Control）

此命令用于控制双色 LED 指示灯的状态。

Bi-color LEDs Control 的命令结构（9 个字节）

命令	CLA	INS	P1	P2	Lc	命令数据域（4 字节）
Bi-color LED Control	FFh	00h	40h	LED 状态控制	04h	闪烁周期控制

P2 LED 状态控制

双色 LED 控制的结构（1 个字节）

命令	项	说明
Bit 0	红色 LED 最终状态	1 = 开；0 = 关
Bit 1	绿色 LED 最终状态	1 = 开；0 = 关
Bit 2	红色 LED 状态掩码	1 = 更新其状态 0 = 不更改
Bit 3	绿色 LED 状态掩码	1 = 更新其状态 0 = 不更改
Bit 4	红色 LED 初始闪烁状态	1 = 开；0 = 关
Bit 5	绿色 LED 初始闪烁状态	1 = 开；0 = 关
Bit 6	红色 LED 闪烁掩码	1 = 闪烁 0 = 不闪烁
Bit 7	绿色 LED 闪烁掩码	1 = 闪烁 0 = 不闪烁

命令数据域 闪烁周期控制。

Bi-color LED Blinking Duration Control 的命令结构（4 字节）

字节 0	字节 1	字节 2	字节 3
T1 周期 初始闪烁状态 （单位 = 100 ms）	T2 周期 切换闪烁状态 （单位 = 100 ms）	重复次数	00h

响应数据域　　SW1 SW2。读卡器返回的状态码。

状态码

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

LED 当前状态（1 字节）

状态	项	说明
Bit 0	当前红色 LED	1 = 开；0 = 关
Bit 1	当前绿色 LED	1 = 开；0 = 关
Bits 2 – 7	保留	

提示：

1. *LED 状态操作是在 LED 闪烁操作之后进行的。*
2. *如果相应的 LED 状态掩码未启用，则 LED 状态不会发生改变。*
3. *如果相应的 LED 闪烁掩码未启用，则 LED 不会闪烁。同时，重复次数的值必须大于 0。*
4. *T1 和 T2 周期参数主要用于控制 LED 闪烁的工作周期和蜂鸣器的鸣响时间。比如说，如果 T1=1，T2=1，则工作周期 = 50%。*

   注：*工作周期 = T1/(T1 + T2)。*

5.4.2. 获取固件版本（Get Firmware Version）

此命令用于获取读写器的固件版本号。

Get Firmware Version 的命令结构（5 字节）

命令	CLA	INS	P1	P2	Le
Get Firmware	FFh	00h	48h	00h	00h

Get Firmware Version 的响应结构（X 字节）

响应	响应数据域
结果	固件版本号

例如：

响应 = 41 43 52 31 32 35 31 54 5F 56 44 30 30 2E 30h = ACR1251T_VD00.0 (ASCII)

5.4.3. 获取 PICC 操作参数（Get PICC Operating Parameter）

此命令用于获取读写器的 PICC 操作参数。

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

命令	CLA	INS	P1	P2	Le
Get PICC Operating Parameter	FFh	00h	50h	00h	00h

Get the PICC Operating Parameter 的响应结构（2 字节）

响应	响应数据域	
结果	90h	PICC 操作参数

PICC 操作参数

位	参数	说明	选项
7	自动 PICC 轮询	启用 PICC 轮询。	1 = 启用 0 = 停用
6	自动 ATS 生成	每次激活 ISO 14443-4 A 类标签都发送 ATS 请求。	1 = 启用 0 = 停用
5	轮询时间间隔	设置连续 PICC 轮询之间的时间间隔	1 = 250 ms 0 = 500 ms
4	FeliCa 424 Kbps	PICC 轮询要检测的标签类别	1 = 检测 0 = 跳过
3	FeliCa 212 Kbps		1 = 检测 0 = 跳过
2	Topaz		1 = 检测 0 = 跳过
1	ISO 14443 B 类		1 = 检测 0 = 跳过
0	ISO 14443 A 类 ***注：****要检测 MIFARE 标签，必须首先禁用自动 ATS 生成。*		1 = 检测 0 = 跳过

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

此命令用于设置读写器的 PICC 操作参数。

Set PICC Operating Parameter 的命令结构（5 字节）

命令	CLA	INS	P1	P2	Le
Set PICC Operating Parameter	FFh	00h	51h	PICC 操作参数	00h

Set PICC Operating Parameter 的响应结构（2 字节）

响应	响应数据域	
结果	90h	PICC 操作参数

PICC 操作参数

位	参数	说明	选项
7	自动 PICC 轮询	启用 PICC 轮询。	1 = 启用 0 = 停用
6	自动 ATS 生成	每次激活 ISO 14443-4 A 类标签都发送 ATS 请求。	1 = 启用 0 = 停用
5	轮询时间间隔	设置连续 PICC 轮询之间的时间间隔	1 = 250 ms 0 = 500 ms
4	FeliCa 424 Kbps	PICC 轮询要检测的标签类别	1 = 检测 0 = 跳过
3	FeliCa 212 Kbps		1 = 检测 0 = 跳过
2	Topaz		1 = 检测 0 = 跳过
1	ISO 14443 B 类		1 = 检测 0 = 跳过
0	ISO 14443 A 类 ***注：****要检测 MIFARE 标签，必须首先禁用自动 ATS 生成。*		1 = 检测 0 = 跳过