Advanced Card Systems Ltd.
Card & Reader Technologies

ACR122S
NFC 读写器（串口）

应用程序编程接口 V2.03

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

目录

1.0. 简介 3

2.0. 特性 4

3.0. 应用程序编程接口概述 5

3.1. 读写器 5
3.1.1. 预定义文档 5
3.1.2. 函数文档 6
3.2. LED 16
3.2.1. 函数文档 16
3.3. 卡片 18
3.3.1. 函数文档 18

附录 A.数据结构 28

附录 A.1. _ACR122_TIMEOUTS 结构体引用 28
附录 A.2. _ACR122_LED_CONTROL 结构体引用 28

附录 B.高阶 API 返回的错误代码 29

图目录

图 1 ：ACR122S 库架构 3

1.0. 简介

本 API 文档介绍了如何利用 ACR122S 软件接口在 ACR122S 读写器上进行应用开发。该软件接口以 32/64 位的 DLL（Dynamic Link Library，动态链接库）的形式提供，可使用常见开发工具，如 Java、Delphi、Visual Basic、Visual C++、Visual C# 和 Visual Basic .NET 进行编程。

ACR122S 读写器可通过 RS-232 接口连接 PC。

ACR122S 库架构如下图所示：

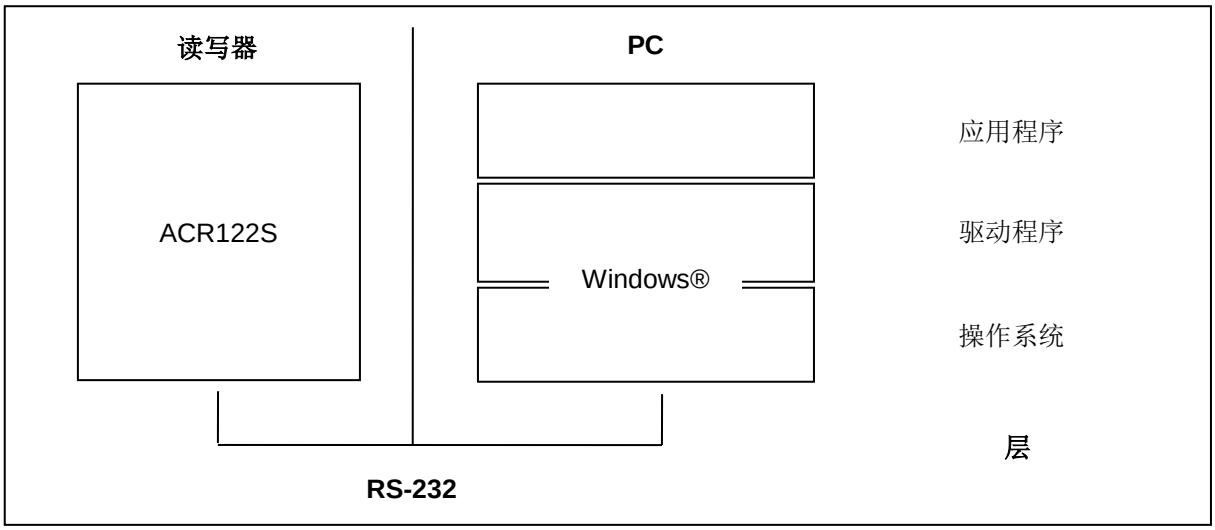

图1：ACR122S 库架构

2.0. 特性

- RS-232 串行接口：波特率=115200 bps, 8-N-1
- USB 接口供电
- 仿 CCID 架构（二进制格式）
- 智能卡读写器：
  - 读/写速率高达 424 Kbps
  - 内置天线用于读写非接触标签，读取智能卡的距离可达 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）
  - 符合 ISO 7816 标准的 SAM 卡槽
- 内置外围设备：
  - 2 个用户可控的 LED 指示灯
  - 1 个用户可控的蜂鸣器
- 符合下列标准：
  - ISO 18092
  - ISO 14443
  - CE
  - FCC
  - KC
  - VCCI
  - RoHS 2

3.0. 应用程序编程接口概述

ACR122S DLL 是应用软件可用的高阶函数集。应用程序可调用 DLL 提供的统一 API（Application Programming Interface，应用程序编程接口）操作 ACR122S 和卡片。ACR122S DLL 通过操作系统的通讯端口与 ACR122S 读写器进行通讯。

ACR122S 读写器的 API 定义了访问 ACR122S 的通用方法。应用程序可以通过接口函数调用 ACR122S，进而操作卡片。

程序开发人员可使用头文件 ACR122.h 的所有函数原型和宏，如下文所述。

3.1. 读写器

3.1.1. 预定义文档

3.1.1.1. ACR122_GetFirmwareVersion 和 ACR122_GetFirmwareVersionA

如果定义了 Unicode，则 ACR122_GetFirmwareVersion 将映射函数 ACR122_GetFirmwareVersionW()。否则，ACR122_GetFirmwareVersion 将映射函数 ACR122_GetFirmwareVersionA()。

```
#define ACR122_GetFirmwareVersion ACR122_GetFirmwareVersionA
```

3.1.1.2. ACR122_Open 和 ACR122_OpenA

如果定义了 Unicode，则 ACR122_Open 将映射函数 ACR122_OpenW()。否则，ACR122_Open 将映射函数 ACR122_OpenA()。

```
#define ACR122_Open ACR122_OpenA
```

3.1.2. 函数文档

3.1.2.1. ACR122_OpenA

此函数用于打开读写器并返回一个引用句柄值。

```
DWORD WINAPI ACR122_OpenA ( LPCSTR   portName,
                            LPHANDLE phReader
                          )
```

参数	说明	
[in] portName	端口名。"\\.\COM1"表示读写器连接的是 Windows®的 COM1 端口。	
[out] phReader	指向 HANDLE 变量的指针。	
返回值	ERROR_SUCCESS	操作成功完成。
	Failure	错误代码。更多详情，请参见 Windows API 和 ACR122 两者的错误代码。

3.1.2.2. ACR122_OpenW

此函数用于打开读写器并返回一个引用句柄值。

```
DWORD WINAPI ACR122_OpenW ( LPCWSTR  portName,
                            LPHANDLE phReader
                          )
```

参数	说明	
[in] portName	端口名。"\\.\COM1"表示读写器连接的是 Windows 的 COM1 端口。	
[out] phReader	指向变量 HANDLE 的指针。	
返回值	ERROR_SUCCESS	操作成功完成。
	Failure	错误代码。更多详情，请参见 Windows API 和 ACR122 两者的错误代码。

源代码举例

```
HANDLE hReader;
DWORD ret;
// Open reader using COM1
ret = ACR122_Open(TEXT("\\\\.\\COM1"), &hReader);
```

3.1.2.3. ACR122_Close

此函数用于关闭读写器并释放资源。

```
DWORD WINAPI ACR122_Close ( HANDLE hReader
)
```

参数	说明	
[in] hReader	函数 ACR122_Open()返回的引用值。	
返回值	ERROR_SUCCESS	操作成功完成。
	Failure	错误代码。更多详情，请参见 Windows API 和 ACR122 两者的错误代码。

源代码举例

```
DWORD ret;
// Close reader
ret = ACR122_Close(hReader);
```

3.1.2.4. ACR122_GetNumSlots

此函数用于获取卡槽数。

```
DWORD WINAPI ACR122_GetNumSlots ( HANDLE  hReader,
                                  LPDWORD pNumSlots
                                )
```

参数	说明	
[in] hReader	函数 ACR122_Open()返回的引用值。	
[out] pNumSlots	指向变量 DWORD 的指针，该变量用于返回卡槽数。	
返回值	ERROR_SUCCESS	操作成功完成。
	Failure	错误代码。更多详情，请参见 Windows API 和 ACR122 两者的错误代码。

源代码举例

```
DWORD numSlots;
DWORD ret;
// Get number of slots
ret = ACR122_GetNumSlots(hReader, &numSlots);
```

3.1.2.5. ACR122_GetBaudRate

此函数用于获取读写器的波特率。

```
DWORD WINAPI ACR122_GetBaudRate ( HANDLE  hReader,
                                  LPDWORD pBaudRate
                                )
```

参数	说明	
[in] hReader	函数 ACR122_Open()返回的引用值。	
[out] pBaudRate	指向变量 DWORD 的指针，该变量用于返回通信波特率。	
返回值	ERROR_SUCCESS	操作成功完成。
	Failure	错误代码。更多详情，请参见 Windows API 和 ACR122 两者的错误代码。

源代码举例

```
DWORD baudRate;
DWORD ret;
// Get baud rate
ret = ACR122_GetBaudRate(hReader, &baudRate);
```

3.1.2.6. ACR122_SetBaudRate

此函数用于设置读写器的通讯波特率。支持两种波特率：9600 bps 和 115200 bps。

```
DWORD WINAPI ACR122_SetBaudRate ( HANDLE hReader,
                                  DWORD baudRate
                                )
```

参数		说明
[in] hReader		函数 ACR122_Open()返回的引用值。
[in] baudRate		波特率必须设为 9600 bps 或 115200 bps。
返回值	ERROR_SUCCESS	操作成功完成。
	Failure	错误代码。更多详情，请参见 Windows API 和 ACR122 两者的错误代码。

源代码举例

```
DWORD ret;
// Set baud rate to 115200 bps
ret = ACR122_SetBaudRate(hReader, 115200);
```

3.1.2.7. ACR122_GetTimeouts

此函数用以获取读写器状态和响应操作的超时参数。

```
DWORD WINAPI ACR122_GetTimeouts ( HANDLE          hReader,
                                  PACR122_TIMEOUTS pTimeouts
                                )
```

参数	说明	
[in] hReader	函数 ACR122_Open()返回的引用值。	
[out] pTimeouts	指向 ACR122_TIMEOUTS 结构体的指针，该结构体用于返回超时信息。	
返回值	ERROR_SUCCESS	操作成功完成。
	Failure	错误代码。更多详情，请参见 Windows API 和 ACR122 两者的错误代码。

***注：**更多 PACR122_TIMEOUTS 的细节，请参见**附录 A.1 _ACR122_TIMEOUTS** 结构体引用。*

源代码举例

```
ACR122_TIMEOUTS timeouts;
DWORD ret;
// Get timeouts
ret = ACR122_GetTimeouts(hReader, &timeouts);
```

3.1.2.8. ACR122_SetTimeouts

此函数用以设置读写器状态和响应操作的超时参数。

```
DWORD WINAPI ACR122_SetTimeouts ( HANDLE  hReader,
                                  const PACR122_TIMEOUTS pTimeouts
                                  )
```

参数	说明	
[in] hReader	函数 ACR122_Open()返回的引用值。	
[in] pTimeouts	指向 ACR122_TIMEOUTS 结构体的指针，该结构体包含新设置的超时时间值。	
返回值	ERROR_SUCCESS	操作成功完成。
	Failure	错误代码。更多详情，请参见 Windows API 和 ACR122 两者的错误代码。

*注：更多 PACR122_TIMEOUTS 的细节，请参**见 Appendix A.1 _ACR122_TIMEOUTS 结构体**引用。*

源代码举例

```
ACR122_TIMEOUTS timeouts;
DWORD ret;
// Get timeouts
// ...
// Modify status timeout to 100 ms
timeouts.statusTimeout = 100;
// Set timeouts
ret = ACR122_SetTimeouts(hReader, &timeouts);
```

3.1.2.9. ACR122_GetFirmwareVersionA

此函数用于获取 ANSI 字符串形式的卡槽固件版本号。

```
DWORD WINAPI ACR122_GetFirmwareVersionA ( HANDLE  hReader,
DWORD   slotNum,
LPSTR   firmwareVersion,
LPDWORD pFirmwareVersionLen
)
```

参数	说明	
[in] hReader	函数 ACR122_Open()返回的引用值。	
[in] slotNum	卡槽号。	
[out] firmwareVersion	指向缓冲区的指针，该缓冲区用于接收读写器返回的固件版本号。	
[in,out] pFirmwareVersionLen	参数固件版本号的长度（字节数），并接收读写器返回的实际字节数。	
返回值	ERROR_SUCCESS	操作成功完成。
	Failure	错误代码。更多详情，请参见 Windows API 和 ACR122 两者的错误代码。

3.1.2.10. ACR122_GetFirmwareVersionW

此函数用于获取 Unicode 字符串形式的卡槽固件版本号。

```
DWORD WINAPI ACR122_GetFirmwareVersionW ( HANDLE  hReader,
DWORD   slotNum,
LPWSTR  firmwareVersion,
LPDWORD pFirmwareVersionLen
)
```

参数	说明	
[in] hReader	函数 ACR122_Open()返回的引用值。	
[in] slotNum	卡槽号。	
[out] firmwareVersion	指向缓存的指针，该缓存用于接收读写器返回的固件版本号。	
[in,out] pFirmwareVersionLen	参数固件版本号的长度（字节数），并接收读写器返回的实际字节数。	
返回值	ERROR_SUCCESS	操作成功完成。
	Failure	错误代码。更多详情，请参见 Windows API 和 ACR122 两者的错误代码。

源代码举例

```
TCHAR firmwareVersion[20];
DWORD firmwareVersionLen;
DWORD ret;
// Get firmware version
firmwareVersionLen = sizeof(firmwareVersion) / sizeof(TCHAR);
ret      =      ACR122_GetFirmwareVersion(hReader,      firmwareVersion,
&firmwareVersionLen);
```

3.2. LED

3.2.1. 函数文档

3.2.1.1. ACR122_SetLedStatesWithBeep

此函数用于控制读写器的 LED0、LED1 和蜂鸣器操作。

```
DWORD WINAPI ACR122_SetLedStatesWithBeep ( HANDLE            hReader,
                                           PACR122_LED_CONTROL controls,
                                           DWORD             numControls,
                                           DWORD             t1,
                                           DWORD             t2,
                                           DWORD             numTimes,
                                           DWORD             buzzerMode
                                         )
```

参数	说明
[in] hReader	函数 ACR122_Open()返回的引用值。
[in] controls	指向 ACR122_LED_CONTROL 数据结构体的指针。
[in] numControls	控制器的数量必须为 2。
[in] t1	T1 周期，单位为毫秒。取值范围：0-25500
[in] t2	T2 周期，单位为毫秒。取值范围：0-25500
[in] numTimes	次数。取值范围：0-255
[in] buzzerMode	蜂鸣器状态掩码。可行值可以用于 OR 操作。 值 / 含义： ACR122_BUZZER_MODE_OFF — 蜂鸣器不开启。 ACR122_BUZZER_MODE_ON_T1 — 蜂鸣器在 T1 周期内开启。 ACR122_BUZZER_MODE_ON_T2 — 蜂鸣器在 T2 周期内开启。

返回值		
	ERROR_SUCCESS	操作成功完成。
	Failure	错误代码。更多详情，请参见 Windows API 和 ACR122 两者的错误代码。

*注：更多 PACR122_LED_CONTROL 的细节，请参见 **附录 A.2 _ ACR122_LED_CONTROL** 结构体引用。*

源代码举例

```
ACR122_LED_CONTROL controls[2];
DWORD ret;
// Set LED0 to ON
controls[0].finalState = ACR122_LED_STATE_ON;
controls[0].updateEnabled = TRUE;
controls[0].initialBlinkingState = ACR122_LED_STATE_OFF;
controls[0].blinkEnabled = FALSE;

// Set LED1 to blink
controls[1].finalState = ACR122_LED_STATE_OFF;
controls[1].updateEnabled = FALSE;
controls[1].initialBlinkingState = ACR122_LED_STATE_OFF;
controls[1].blinkEnabled = TRUE;

// Beep on T1 where T1 and T2 are equal to 100 ms
ret  =  ACR122_SetLedStatesWithBeep(hReader,  controls,  2,  100,  100,
ACR122_BUZZER_MODE_ON_T1);
```

3.3. 卡片

3.3.1. 函数文档

3.3.1.1. ACR122_DirectTransmit

此函数用于发送标签命令并接收读写器非接触接口返回的响应。

```
DWORD WINAPI ACR122_DirectTransmit ( HANDLE       hReader,
 const LPBYTE sendBuffer,
 DWORD        sendBufferLen,
 LPBYTE       recvBuffer,
 LPDWORD      pRecvBufferLen
 )
```

参数	说明	
[in] hReader	函数 ACR122_Open()返回的引用值。	
[in] sendBuffer	指向要写入卡片的实际数据的指针。	
[in] sendBufferLen	参数 sendBuffer 的长度（字节数）。	
[in] recvBuffer	指向卡片返回数据的指针。	
[in,out] pRecvBufferLen	参数 recvBuffer 的长度（字节数），并接收卡片返回的实际字节数。	
返回值	ERROR_SUCCESS	操作成功完成。
	Failure	错误代码。更多详情，请参见 Windows API 和 ACR122 两者的错误代码。

源代码举例

```
BYTE command[] = { 0xD4, 0x4A, 0x01, 0x00 }; // Poll Type A card
DWORD commandLen = sizeof(command);
BYTE response[300];
DWORD responseLen = sizeof(response);
DWORD ret;
ret = ACR122_DirectTransmit(hReader, command, commandLen, response,
&responseLen);
```

本 API 可与非接触接口交换命令和响应。以下为可能涉及到的命令组：

组 1. 不同标签类型的 PICC 轮询。例如，ISO 14443-4 A 类，ISO 14443-4 B 类，FeliCa 和 MIFARE 卡的 PICC 轮询

实例：ISO 14443-4 A 类

==========

命令 = {D4 4A 01 00}

响应 = {D5 4B 01 01 00 08 28 04 85 82 2F A0 07 77 F7 80 02 47 65 90 00}

其中， 查找到的标签数量 = [01]

目标编号 = 01

SENS_RES = 00 08

SEL_RES = 28

UID 长度 = 4

UID = 85 82 2F A0

ATS = 07 77 F7 80 02 47 65

操作完成 = 90 00

或

响应 = {D5 4B 00 90 00} (未找到标签)

实例：ISO 14443-4 B 类

==========

命令 = {D4 4A 01 03 00}

响应 = {D5 4B 01 01 50 00 01 32 F4 00 00 00 00 33 81 81 01 21 90 00}

其中， 查找到的标签数量 = [01]

目标编号 = 01

ATQB = 50 00 01 32 F4 00 00 00 00 33 81 81

ATTRIB_RES 长度 = 01

ATTRIB_RES = 21

操作完成 = 90 00

或

响应 = {D5 4B 00 90 00} (未找到标签)

实例：MIFARE Classic® 1K/4K/MIFARE® Ultralight®

==========

命令 = {D4 4A 01 00}

响应 = {D5 4B 01 01 00 44 00 07 04 6E 0C A1 BF 02 84 90 00}

其中，查找到的标签数量 = [01]

目标编号 = 01

SENS_RES = 00 44

SEL_RES = 00

UID 长度 = 7

UID = 04 6E 0C A1 BF 02 84

操作完成 = 90 00

或

响应 = {D5 4B 00 90 00} (未找到标签)

注： *可通过识别 SEL_RES 来鉴定标签类型。*

几种常见标签类型的 SEL_RES：

00 = MIFARE Ultralight

08 = MIFARE 1K

09 = MIFARE MINI

18 = MIFARE 4K

20 = MIFARE® DESFire®

28 = JCOP30

98 = Gemplus MPCOS

实例：FeliCa 212 K

==========

命令 = {D4 4A 01 01 00 FF FF 00 00}

响应 = {D5 4B 01 01 14 01 01 01 05 01 86 04 02 02 03 00 4B 02 4F 49 8A 8A 80 08 90 00}

其中， 查找到的标签数量 = [01]

目标编号 = 01

POL_RES 长度 = 14

响应代码 = 01

NFCID2 = 01 01 05 01 86 04 02 02

PAD = 03 00 4B 02 4F 49 8A 8A 80 08

操作完成 = 90 00

或

响应 = {D5 4B 00 90 00} (未找到标签)

实例：FeliCa 424K

==========

命令 = {D4 4A 01 02 00 FF FF 00 00}

响应 = {D5 4B 01 01 14 01 01 01 05 01 86 04 02 02 03 00 4B 02 4F 49 8A 8A 80 08 90 00}

其中， 查找到的标签数量 = [01]

目标编号 = 01

POL_RES 长度 = 14

响应代码 = 01

NFCID2 = 01 01 05 01 86 04 02 02

PAD = 03 00 4B 02 4F 49 8A 8A 80 08

操作完成 = 90 00

或

响应 = {D5 4B 00 90 00}(未找到标签)

组 2. 为符合 ISO 14443-4 的标签交换标签命令和响应。

C_APDU = 00 84 00 00 08

命令 = {D4 40 01 00 84 00 00 08}

响应 = {D5 41 [00] 62 89 99 ED C0 57 69 2B 90 00 90 00}

其中，响应数据 = 62 89 99 ED C0 57 69 2B 90 00

组 3. 交换 FeliCa 命令和响应，例如，读存储块。

命令 = {D4 40 01 10 06 01 01 05 01 86 04 02 02 01 09 01 01 80 00}

响应 = {D5 41 [00] 1D 07 01 01 05 01 86 04 02 02 00 00 01 00 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 90 00}

注：*更多详情，请参见 FeliCa 标准。*

组 4. 交换 MIFARE 1K/4K Classic 命令和响应

例 1：用于认证密钥 A 的命令

块 04

密钥 = FF FF FF FF FF FF

UID = F6 8E 2A 99

命令 = {D4 40 01 60 04 FF FF FF FF FF FF F6 8E 2A 99}

响应 = {D5 41 [00] 90 00}

例 2：用于认证密钥 B 的命令

块 04

密钥 = FF FF FF FF FF FF

UID = F6 8E 2A 99

命令 = {D4 40 01 61 04 FF FF FF FF FF FF F6 8E 2A 99}

响应 = {D5 41 [00] 90 00}

例 3：用于读取数据块的命令

读取值块 04 的内容

命令 = {D4 40 01 30 04}

响应 = {D5 41 [00] 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 90 00}

其中，块数据 = 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16

例 4：用于更新数据块的命令

更新 Block 04 的内容

命令 = {D4 40 01 A0 04 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10}

响应 = {D5 41 [00] 90 00}

注：*将返回错误代码 [XX]。*

[00] = 有效

其他 = 错误

更多详情请，参见错误代码表。

组 5. 交换 MIFARE Classic 1K/4K 值块的命令和响应

值块用于执行电子钱包的功能，例如增值，减值，恢复，传输等。值块的固定数据结构使得值块可进行差错检验、差错校正和备份管理。

字节号	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
说明	值				值				值				地址	地址	地址	地址

其中：

值 一个有符号的 4 字节值。

一个值的最低有效字节存储在最低地址字节。取反的字节以标准 2 的补码格式保存。

地址 表示一个 1 字节地址，可用于保存一个块的存储地址（块）。

例如：

值 100（十进制） = 64 (十六进制)，假设值块 = 05h

格式化的值块 = 64 00 00 00 9B FF FF FF 64 00 00 00 05 FA 05 FA

1. 用值 100（十进制）更新值块 05 的内容。

   命令 = {D4 40 01 **A0** 05 64 00 00 00 9B FF FF FF 64 00 00 00 05 FA 05 FA}

   响应 = {D5 41 [00] 90 00}

2. 使值块 05 的值增加 1（十进制）。

   命令 = {D4 40 01 **C1** 05 01 00 00 00}

   响应 = {D5 41 [00] 90 00}

   ***注：** // 使值块 05 的值减少 1（十进制）。*

   命令 = {D4 40 01 **C0** 05 01 00 00 00}

3. 传输值块 05 先前计算的值（十进制）.

   Command = {D4 40 01 **B0** 05}

   响应 = {D5 41 [00] 90 00}

4. 恢复值块 05 的值（取消先前的增值或减值操作）。

   Command = {D4 40 01 **C2** 05}

5. 读取值块 05 的内容

   命令 = {D4 40 01 **30** 05}

   响应 = {D5 41 [00] 65 00 00 00 9A FF FF FF 65 00 00 00 05 FA 05 FA 90 00}

   其中，值 = 101 （十进制）

6. 复制值块 05 的值到值块 06 (十进制)。

   命令 = {D4 40 01 **C2** 05}

   响应 = {D5 41 [00] 90 00}

   Command = {D4 40 01 **B0** 06}

   响应 = {D5 41 [00] 90 00}

7. 读取值块 06 的内容

命令 = {D4 40 01 **30** 06}

响应 = {D5 41 [00] 65 00 00 00 9A FF FF FF 65 00 00 00 05 FA 05 FA 90 00}

其中，值 = 101 （十进制）地址“05 FA 05 FA”表明该值是从值块 05 复制的。

组 6. PICC 操作参数的设置

例 1： 设置 PICC 轮询重试次数的命令

命令 = {D4 32 05 00 00 [00]} // 重试次数 = 00

响应 = {D5 33}

例 2： 使 ISO 14443-4 A 类卡能激活的命令

例如：进入 JCOP 的 ISO 14443-4 模式。

命令 = {D4 12 24}

响应 = {D5 12}

例 3： 使 ISO 14443-4 A 类卡能去激活的命令

例如：进入 JCOP 的 MIFARE 模拟模式。

命令 = {D4 12 34}

响应 = {D5 12}

例 4： 开启 PICC 天线的命令

命令 = {D4 32 01 01}

响应 = {D5 33}

例 5： 关闭 PICC 天线的命令

命令 = {D4 32 01 00}

响应 = {D5 33}

3.3.1.2. ACR122_ExchangeApdu

此函数用于给卡片发送 APDU 命令并接收卡片的 APDU 响应。

```
DWORD WINAPI ACR122_ExchangeApdu ( HANDLE        hReader,
                                   DWORD         slotNum,
                                   const LPBYTE  sendBuffer,
                                   DWORD         sendBufferLen,
                                   LPBYTE        recvBuffer,
                                   LPDWORD       pRecvBufferLen
                                 )
```

参数	说明	
[in] hReader	函数 ACR122_Open()返回的引用值。	
[in] slotNum	卡槽号。	
[in] sendBuffer	指向要写入卡片的实际数据的指针。	
[in] sendBufferLen	参数 sendBuffer 的长度（字节数）。	
[out] recvBuffer	指向卡片返回数据的指针。	
[in,out] pRecvBufferLen	参数 recvBuffer 的长度（字节数），并接收卡片返回的实际字节数。	
返回值	ERROR_SUCCESS	操作成功完成。
	Failure	错误代码。更多详情，请参见 Windows API 和 ACR122 两者的错误代码。

源代码举例

```
BYTE command[] = { 0x80, 0x84, 0x00, 0x00, 0x08 };
DWORD commandLen = sizeof(command);
BYTE response[300];
DWORD responseLen = sizeof(response);
DWORD ret;
// Exchange APDU on slot 0
ret = ACR122_ExchangeApdu(hReader, 0, command, commandLen, response,
&responseLen);
```

3.3.1.3. ACR122_PowerOffIcc

此函数用于使卡槽中的 ICC 下电。

```
DWORD WINAPI ACR122_PowerOffIcc ( HANDLE hReader,
DWORD  slotNum
)
```

参数		说明
[in] hReader		函数 ACR122_Open()返回的引用值。
[in] slotNum		卡槽号。
返回值	ERROR_SUCCESS	操作成功完成。
	Failure	错误代码。更多详情，请参见 Windows API 和 ACR122 两者的错误代码。

源代码举例

```
DWORD ret;
// Power off slot 0
ret = ACR122_PowerOffIcc(hReader, 0);
```

3.3.1.4. ACR122_PowerOnIcc

此函数用于使卡槽中的卡片上电并返回 ATR 字符串。

```
DWORD WINAPI ACR122_PowerOnIcc ( HANDLE  hReader,
   DWORD   slotNum,
   LPBYTE  atr,
   LPDWORD pAtrLen
   )
```

参数	说明	
[in] hReader	函数 ACR122_Open()返回的引用值。	
[in] slotNum	卡槽号。	
[out] atr	指向缓冲区的指针，该缓冲区用于接收卡片返回的 ATR 字符串。	
[in,out] pAtrLen	参数 atr 的长度（字节数），并接收卡片返回的实际字节数。	
返回值	ERROR_SUCCESS	操作成功完成。
	Failure	错误代码。更多详情，请参见 Windows API 和 ACR122 两者的错误代码。

源代码举例

```
BYTE atr[64];
DWORD atrLen = sizeof(atr);
DWORD ret;
// Power on slot 0
ret = ACR122_PowerOnIcc(hReader, 0, atr, &atrLen);
```

附录A. 数据结构

附录A.1. _ACR122_TIMEOUTS 结构体引用

此数据结构体用于函数 **ACR122_GetTimeouts()**和 **ACR122_SetTimeouts()**。

- `DWORD _ACR122_TIMEOUTS::numResponseRetries`

  响应重试次数。

  默认为 1 次。

- `DWORD _ACR122_TIMEOUTS::numStatusRetries`

  状态重试次数。

  默认为 1 次。

- `DWORD _ACR122_TIMEOUTS::responseTimeout`

  响应超时时间，单位为毫秒。

  默认为 10000 毫秒。

- `DWORD _ACR122_TIMEOUTS::statusTimeout`

  状态超时时间，单位为毫秒。

  默认为 2000 毫秒。

附录A.2. _ACR122_LED_CONTROL 结构体引用

此数据结构体用于函数 **ACR122_SetLedStatesWithBeep()**。

- `BOOL _ACR122_LED_CONTROL::blinkEnabled`

  使 LED 指示灯闪烁。

  设为 TRUE 使 LED 指示灯闪烁。其他情况下，均设为 FALSE。

- `DWORD _ACR122_LED_CONTROL::finalState`

  最终状态。

  可能出现的值为 ACR122_LED_STATE_OFF 和 ACR122_LED_STATE_ON。

- `DWORD _ACR122_LED_CONTROL::initialBlinkingState`

  初始闪烁状态。

  可能出现的值为 ACR122_LED_STATE_OFF 和 ACR122_LED_STATE_ON。

- `BOOL _ACR122_LED_CONTROL::updateEnabled`

  使能更新。

  设为 TRUE 以更新状态。其他情况下，设为 FALSE 以保持原有状态。

附录B. 高阶 API 返回的错误代码

- `ACR122_ERROR_NO_MORE_HANDLES ((DWORD) 0x20000001L)`
  句柄无效。
- `ACR122_ERROR_UNKNOWN_STATUS ((DWORD) 0x20000002L)`
  读写器发生未知错误。
- `ACR122_ERROR_OPERATION_FAILURE ((DWORD) 0x20000003L)`
  操作失败。
- `ACR122_ERROR_OPERATION_TIMEOUT ((DWORD) 0x20000004L)`
  操作超时。
- `ACR122_ERROR_INVALID_CHECKSUM ((DWORD) 0x20000005L)`
  校验和计算错误。
- `ACR122_ERROR_INVALID_PARAMETER ((DWORD) 0x20000006L)`
  不正确的参数输入。