Advanced Card Systems Ltd.
Card & Reader Technologies

ACR122S
Serial NFC Reader

Application Programming Interface V2.03

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

Table of Contents

List of Figures

1.0. Introduction

This API document describes the use of ACR122S interface software to facilitate application development with the ACR122S reader. This interface software is supplied in the form of 32-bit and 64-bit DLL (Dynamic Link Library) which can be programmed using popular development tools like Java, Delphi, Visual Basic, Visual C++, Visual C# and Visual Basic .NET.

ACR122S can be connected to the PC through the RS-232 interface.

The architecture of the ACR122S library can be visualized as the following diagram:

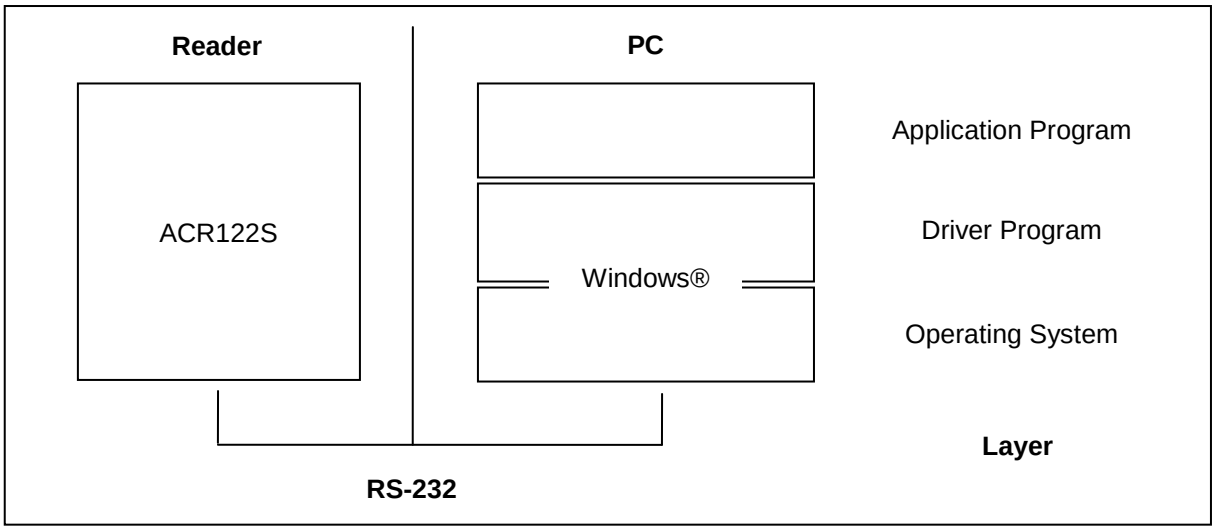

Figure 1: ACR122S Library Architecture

2.0. Features

- Serial RS-232 Interface: Baud Rate = 115200 bps, 8-N-1
- USB interface for power supply
- CCID-like frame format (Binary format)
- Smart Card Reader:
  - Read/Write speed of up to 424 Kbps
  - Built-in antenna for contactless tag access, with card reading distance of up to 50 mm (depending on tag type)
  - Support for ISO 14443 Part 4 Type A and B cards, MIFARE®, FeliCa, and all four types of NFC (ISO/IEC 18092 tags)
  - Built-in anti-collision feature (only one tag is accessed at any time)
  - ISO 7816-compliant SAM slot
- Built-in Peripherals:
  - Two user-controllable LEDs
  - User-controllable buzzer
- Compliant with the following standards:
  - ISO 18092
  - ISO 14443
  - CE
  - FCC
  - KC
  - VCCI
  - RoHS 2

3.0. Application Programming Interface Overview

The ACR122S DLL is a set of high-level functions provided for the application software use. It supplies a consistent API (Application Programming Interface) for the application to operate on ACR122S and on the corresponding presented card. The DLL communicates with the ACR122S via the communication port facilities provided by the operating system.

The ACR122S API defines a common way of accessing the ACR122S. Application programs invoke the ACR122S through the interface functions and perform operations on the presented card.

The header file ACR122.h is available for the program developer, which contains all the function prototypes and macros as described below.

3.1. Reader

3.1.1. Define Documentation

3.1.1.1. ACR122_GetFirmwareVersion and ACR122_GetFirmwareVersionA

`ACR122_GetFirmwareVersion` will be mapped to `ACR122_GetFirmwareVersionW()` function if Unicode is defined. Otherwise, it will be mapped to `ACR122_GetFirmwareVersionA()` function.

```
#define ACR122_GetFirmwareVersion ACR122_GetFirmwareVersionA
```

3.1.1.2. ACR122_Open and ACR122_OpenA

`ACR122_Open` will be mapped to `ACR122_OpenW()` function if Unicode is defined. Otherwise, it will be mapped to `ACR122_OpenA()` function.

```
#define ACR122_Open ACR122_OpenA
```

3.1.2. Function Documentation

3.1.2.1. ACR122_OpenA

This function is used to open the reader and return a handle value as a reference.

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

Parameter	Description	
`[in] portName`	Port name. "\\.\COM1" means that the reader is connected to COM1 in Windows®.	
`[out] phReader`	Pointer to the `HANDLE` variable.	
Return Value	`ERROR_SUCCESS`	The operation completed successfully.
	`Failure`	An error code. See Windows API error codes and ACR122 error codes.

3.1.2.2. ACR122_OpenW

This function is used to open a reader and return a handle value as reference.

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

Parameter	Description	
[in] portName	Port name. "\\.\COM1" means that the reader is connected to COM1 in Windows.	
[out] phReader	Pointer to the HANDLE variable.	
Return Value	ERROR_SUCCESS	The operation completed successfully.
	Failure	An error code. See Windows API error codes and ACR122 error codes.

Source Code Example

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

3.1.2.3. ACR122_Close

This function is used to close the reader and release the resources.

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

Parameter	Description	
[in] hReader	A reference value returned from ACR122_Open() function.	
Return Value	ERROR_SUCCESS	The operation completed successfully.
	Failure	An error code. See Windows API error codes and ACR122 error codes.

Source Code Example

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

3.1.2.4. ACR122_GetNumSlots

This function is used to retrieve the number of slots.

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

Parameter	Description	
[in] hReader	A reference value returned from ACR122_Open() function.	
[out] pNumSlots	Pointer to a DWORD variable in which the number of slots is returned.	
Return Value	ERROR_SUCCESS	The operation completed successfully.
	Failure	An error code. See Windows API error codes and ACR122 error codes.

Source Code Example

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

3.1.2.5. ACR122_GetBaudRate

This function is used to retrieve the baud rate of reader.

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

Parameter	Description	
[in] hReader	A reference value returned from ACR122_Open() function.	
[out] pBaudRate	Pointer to a DWORD variable in which the baud rate is returned.	
Return Value	ERROR_SUCCESS	The operation completed successfully.
	Failure	An error code. See Windows API error codes and ACR122 error codes.

Source Code Example

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

3.1.2.6. ACR122_SetBaudRate

This function is used to set the communication baud rate of reader. The reader supports 9600 bps and 115200 bps.

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

Parameter	Description	
[in] hReader	A reference value returned from ACR122_Open() function.	
[in] baudRate	Baud rate must be 9600 bps or 115200 bps.	
Return Value	ERROR_SUCCESS	The operation completed successfully.
	Failure	An error code. See Windows API error codes and ACR122 error codes.

Source Code Example

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

3.1.2.7. ACR122_GetTimeouts

This function is used to retrieve the timeout parameters for status and response operations of the reader.

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

Parameter	Description	
[in] hReader	A reference value returned from ACR122_Open() function.	
[out] pTimeouts	Pointer to a ACR122_TIMEOUTS structure in which the timeout information is returned.	
Return Value	ERROR_SUCCESS	The operation completed successfully.
	Failure	An error code. See Windows API error codes and ACR122 error codes.

Note: *For PACR122_TIMEOUTS, please see* ***Appendix A.1*** *_ACR122_TIMEOUTS Struct Reference.*

Source Code Example

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

3.1.2.8. ACR122_SetTimeouts

This function is used to set the timeout parameters for status and response operations on the reader.

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

Parameter	Description	
[in] hReader	A reference value returned from ACR122_Open() function.	
[in] pTimeouts	Pointer to a ACR122_TIMEOUTS structure that contains the new timeout values	
Return Value	ERROR_SUCCESS	The operation completed successfully.
	Failure	An error code. See Windows API error codes and ACR122 error codes.

Note: *For PACR122_TIMEOUTS, please see* ***Appendix A.1*** *_ACR122_TIMEOUTS Struct Reference.*

Source Code Example

```
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

This function is used to retrieve the firmware version in ANSI string of the slot.

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

Parameter	Description	
[in] hReader	A reference value returned from ACR122_Open() function.	
[in] slotNum	Slot number.	
[out] firmwareVersion	A pointer to the buffer that receives the firmware version returned from the reader.	
[in,out] pFirmwareVersionLen	The length in number of bytes of the firmware version parameter and receives the actual number of bytes received from the reader.	
Return Value	ERROR_SUCCESS	The operation completed successfully.
	Failure	An error code. See Windows API error codes and ACR122 error codes.

3.1.2.10. ACR122_GetFirmwareVersionW

This function is used to retrieve the firmware version in Unicode string of the slot.

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

Parameter	Description	
[in] hReader	A reference value returned from ACR122_Open() function.	
[in] slotNum	Slot number.	
[out] firmwareVersion	A pointer to the buffer that receives the firmware version returned from the reader.	
[in,out] pFirmwareVersionLen	The length in number of bytes of the firmware version parameter and receives the actual number of bytes received from the reader.	
Return Value	ERROR_SUCCESS	The operation completed successfully.
	Failure	An error code. See Windows API error codes and ACR122 error codes.

Source Code Example

```
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. Function Documentation

3.2.1.1. ACR122_SetLedStatesWithBeep

This function is used to control LED0, LED1 and buzzer operation of the reader.

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

Parameter	Description
[in] hReader	A reference value returned from ACR122_Open() function.
[in] controls	A pointer to the array of ACR122_LED_CONTROL data structure.
[in] numControls	Number of controls must be 2.
[in] t1	T1 in milliseconds. The value must be from 0 to 25500.
[in] t2	T2 in milliseconds. The value must be from 0 to 25500
[in] numTimes	Number of times. The values must be from 0 to 255.
[in] buzzerMode	A bitmask of buzzer mode. Possible values may be combined with the OR operation.

Value	Meaning
ACR122_BUZZER_MODE_OFF	The buzzer will not turn on.
ACR122_BUZZER_MODE_ON_T1	The buzzer will turn on during T1 duration.
ACR122_BUZZER_MODE_ON_T2	The buzzer will turn on during T2 duration.

Return Value	ERROR_SUCCESS	The operation completed successfully.
	Failure	An error code. See Windows API error codes and ACR122 error codes.

Note: *For PACR122_LED_CONTROL, please see* ***Appendix A.2*** *_ACR122_LED_CONTROL Struct Reference.*

Source Code Example

```
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. Card

3.3.1. Function Documentation

3.3.1.1. ACR122_DirectTransmit

This function is used to send tag command and receive response from the contactless interface of the reader.

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

Parameter	Description	
[in] hReader	A reference value returned from ACR122_Open() function.	
[in] sendBuffer	A pointer to the actual data to be written to the card.	
[in] sendBufferLen	The length in number of bytes of the sendBuffer parameter.	
[in] recvBuffer	A pointer to any data returned from the card.	
[in,out] pRecvBufferLen	The length in number of bytes of the recvBuffer parameter and receives the actual number of bytes received from the card.	
Return Value	ERROR_SUCCESS	The operation completed successfully.
	Failure	An error code. See Windows API error codes and ACR122 error codes.

Source Code Example

```
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);
```

This API is used for exchanging commands and responses with the contactless interface. Some possible command groups are:

Group 1. PICC Polling for different Tag Types. E.g. ISO 14443-4 Type A, ISO 14443-4 Type B, FeliCa and MIFARE

Case: ISO 14443-4 Type A

==========

Command = {D4 4A 01 00}

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

In which, Number of Tag found = [01]

Target number = 01

SENS_RES = 00 08

SEL_RES = 28

Length of the UID = 4

UID = 85 82 2F A0

ATS = 07 77 F7 80 02 47 65

Operation Finished = 90 00

or

Response = {D5 4B 00 90 00} (no tag found)

Case: ISO 14443-4 Type B

==========

Command = {D4 4A 01 03 00}

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

In which, Number of Tag found = [01]

Target number = 01

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

ATTRIB_RES Length = 01

ATTRIB_RES = 21

Operation Finished = 90 00

or

Response = {D5 4B 00 90 00} (no tag found)

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

==========

Command = {D4 4A 01 00}

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

In which, Number of Tag found = [01]

Target number = 01

SENS_RES = 00 44

SEL_RES = 00

Length of the UID = 7

UID = 04 6E 0C A1 BF 02 84

Operation Finished = 90 00

or

Response = {D5 4B 00 90 00} (no tag found)

Note: *The tag type can be determined by recognizing the SEL_RES.*

SEL_RES of some common tag types:

00 = MIFARE Ultralight

08 = MIFARE 1K

09 = MIFARE MINI

18 = MIFARE 4K

20 = MIFARE® DESFire®

28 = JCOP30

98 = Gemplus MPCOS

Case: FeliCa 212K

==========

Command = {D4 4A 01 01 00 FF FF 00 00}

Response = {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}

In which, Number of Tag found = [01]

Target number = 01

POL_RES Length = 14

Response Code = 01

NFCID2 = 01 01 05 01 86 04 02 02

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

Operation Finished = 90 00

or

Response = {D5 4B 00 90 00} (no tag found)

Case: FeliCa 424K

==========

Command = {D4 4A 01 02 00 FF FF 00 00}

Response = {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}

In which, Number of Tag found = [01]

Target number = 01

POL_RES Length = 14

Response Code = 01

NFCID2 = 01 01 05 01 86 04 02 02

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

Operation Finished = 90 00

or

Response = {D5 4B 00 90 00}(no tag found)

Group 2. Exchange Tag Commands and Responses for ISO 14443-4 compliant tags.

C_APDU = 00 84 00 00 08

Command = {D4 40 01 00 84 00 00 08}

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

In which, Response Data = 62 89 99 ED C0 57 69 2B 90 00

Group 3. Exchange FeliCa Commands and Responses, e.g. Read the memory block.

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

Response = {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}

Note: *Please refer to FeliCa specification for details.*

Group 4. Exchange MIFARE Classic 1K/4K Commands and Response

Example 1: CMD for KEY A Authentication

Block 04

KEY = FF FF FF FF FF FF

UID = F6 8E 2A 99

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

Response = {D5 41 [00] 90 00}

Example 2: CMD for KEY B Authentication

Block 04

KEY = FF FF FF FF FF FF

UID = F6 8E 2A 99

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

Response = {D5 41 [00] 90 00}

Example 3: CMD for Read Data Block

Read the content of Block 04

Command = {D4 40 01 30 04}

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

In which, Block Data = 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16

Example 4: CMD for Update Data Block

Update the content of Block 04

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

Response = {D5 41 [00] 90 00}

Note: *The error code [XX] will be returned.*

[00] = Valid

Other = Error

Please refer to Error Codes Table for more details.

Group 5. Exchange MIFARE Classic 1K/4K Value Block Commands and Response

The value blocks are used for performing electronic purse functions. E.g. Increment, Decrement, Restore and Transfer, etc. The value blocks have a fixed data format which permits error detection and correction and a backup management.

Byte Number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Description	Value				Value				Value				Adr	Adr	Adr	Adr

Where:

Value A signed 4-byte value.

The lowest significant byte off a value is stored in the lowest address byte. Negative values are stored in standard 2's complement format.

Adr 1-Byte address which can be used to save the storage address of a block (optional).

Example:

Value 100 (decimal) = 64 (Hex), assume Block = 05h

The formatted value block = 64 00 00 00 9B FF FF FF 64 00 00 00 05 FA 05 FA

1. Update the content of Block 05 with a value 100 (dec)

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

   Response = {D5 41 [00] 90 00}

2. Increment the value of Block 05 by 1 (dec)

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

   Response = {D5 41 [00] 90 00}

   Note*: Decrement the value of Block **05 by 1 (dec)***

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

3. Transfer the prior calculated value of Block 05 (dec)

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

   Response = {D5 41 [00] 90 00}

4. Restore the value of Block 05 (cancel the prior Increment or Decrement operation)

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

5. Read the content of Block 05

   Command = {D4 40 01 **30 05}**

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

   In which, the value = 101 (dec)

6. Copy the value of Block 05 to Block 06 (dec)

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

   Response = {D5 41 [00] 90 00}

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

   Response = {D5 41 [00] 90 00}

7. Read the content of Block 06

   Command = {D4 40 01 **30 06}**

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

   In which, the value = 101 (dec). The Adr "05 FA 05 FA" tells us the value is copied from Block 05.

Group 6: PICC Operation parameters setting.

Example 1: CMD for PICC Polling Retry Time

Command = {D4 32 05 00 00 [00]} // retry time = 00

Response = {D5 33}

Example 2: CMD for Enable ISO14443-4 Type A Activation. E.g. To enter the ISO14443-4 mode of JCOP

Command = {D4 12 24}

Response = {D5 12}

Example 3: CMD for Disable ISO14443-4 Type A Activation. E.g. To enter the MIFARE emulation mode of JCOP

Command = {D4 12 34}

Response = {D5 12}

Example 4: CMD for Turn On PICC Antenna

Command = {D4 32 01 01}

Response = {D5 33}

Example 5: CMD for Turn Off PICC Antenna

Command = {D4 32 01 00}

Response = {D5 33}

3.3.1.2. ACR122_ExchangeApdu

This function is used to send an APDU command and receive an APDU response from the card.

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

Parameter	Description	
[in] hReader	A reference value returned from ACR122_Open() function.	
[in] slotNum	Slot number.	
[in] sendBuffer	A pointer to the actual data to be written to the card.	
[in] sendBufferLen	The length in number of bytes of the sendBuffer parameter.	
[out] recvBuffer	A pointer to any data returned from the card.	
[in,out] pRecvBufferLen	The length in number of bytes of the recvBuffer parameter and receives the actual number of bytes received from the card.	
Return Value	ERROR_SUCCESS	The operation completed successfully.
	Failure	An error code. See Windows API error codes and ACR122 error codes.

Source Code Example

```
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

This function is used to power off the card in the slot.

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

Parameter	Description	
[in] hReader	A reference value returned from ACR122_Open() function.	
[in] slotNum	Slot number.	
Return Value	ERROR_SUCCESS	The operation completed successfully.
	Failure	An error code. See Windows API error codes and ACR122 error codes.

Source Code Example

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

3.3.1.4. ACR122_PowerOnIcc

This function is used to power on the card in the slot and then return the ATR string from the card.

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

Parameter	Description	
[in] hReader	A reference value returned from ACR122_Open() function.	
[in] slotNum	Slot number.	
[out] atr	A pointer to the buffer that receives the ATR string returned from the card.	
[in,out] pAtrLen	The length in number of bytes of the atr parameter and receives the actual number of bytes received from the card.	
Return Value	ERROR_SUCCESS	The operation completed successfully.
	Failure	An error code. See Windows API error codes and ACR122 error codes.

Source Code Example

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

Appendix A. Data Structures

Appendix A.1. _ACR122_TIMEOUTS Struct Reference

This data structure is used in **`ACR122_GetTimeouts()`** and **`ACR122_SetTimeouts()`** function.

- `DWORD _ACR122_TIMEOUTS::numResponseRetries`

  Number of response retries.

  Default is 1.

- `DWORD _ACR122_TIMEOUTS::numStatusRetries`

  Number of status retries.

  Default is 1.

- `DWORD _ACR122_TIMEOUTS::responseTimeout`

  Response timeout in milliseconds.

  Default is 10000 ms.

- `DWORD _ACR122_TIMEOUTS::statusTimeout`

  Status timeout in milliseconds.

  Default is 2000 ms.

Appendix A.2. _ACR122_LED_CONTROL Struct Reference

This data structure is used in **`ACR122_SetLedStatesWithBeep()`** function.

- `BOOL _ACR122_LED_CONTROL::blinkEnabled`

  Enable blink.

  Set to TRUE to enable blink. Otherwise, set to FALSE.

- `DWORD _ACR122_LED_CONTROL::finalState`

  Final state.

  Possible values are ACR122_LED_STATE_OFF and ACR122_LED_STATE_ON.

- `DWORD _ACR122_LED_CONTROL::initialBlinkingState`

  Initial blinking state.

  Possible values are ACR122_LED_STATE_OFF and ACR122_LED_STATE_ON.

- `BOOL _ACR122_LED_CONTROL::updateEnabled`

  Enable update.

  Set to TRUE to update the state. Otherwise, set to FALSE to keep the state unchanged.

Appendix B. Error Codes returned by high-level APIs

- `ACR122_ERROR_NO_MORE_HANDLES ((DWORD) 0x20000001L)`
  The handle is invalid.
- `ACR122_ERROR_UNKNOWN_STATUS ((DWORD) 0x20000002L)`
  Reader unknown error.
- `ACR122_ERROR_OPERATION_FAILURE ((DWORD) 0x20000003L)`
  Operation failed.
- `ACR122_ERROR_OPERATION_TIMEOUT ((DWORD) 0x20000004L)`
  Timeout operation.
- `ACR122_ERROR_INVALID_CHECKSUM ((DWORD) 0x20000005L)`
  Checksum calculation error
- `ACR122_ERROR_INVALID_PARAMETER ((DWORD) 0x20000006L)`
  Incorrect parameter input.