Advanced Card Systems Ltd.
Card & Reader Technologies

ACR1281U-C8 Contactless Reader/Writer

Application Programming Interface V1.00

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

Table of Contents

List of Tables

1.0. Introduction

The ACR1281U-C8 is the new version of ACS's ACR120U Contactless Smart Card Reader. The ACR1281U-C8 USB Protocol defines the interface between the PC and the reader, as well as the communication channel between the PC and the supported contactless cards, including Mifare®, ISO 14443 Type A and B compatible cards. By using the High Level APIs, the users can develop applications that involve the use of contactless cards with minimum effort. For example:

- **Access control, Identification:** Reading the serial numbers of all cards in the field
- **Data Storage:** Performing encrypted read and write operations
- **Ticketing:** Performing read, write, increment and decrement operations in an encrypted environment
- **Multi applications:** Performing read, write, increment and decrement operations on various sectors of the card

1.1. Features

- USB Full Speed (12 Mbps)
- Read and write functionality
- Smart Card Reader:
  - Built-in antenna for contactless tag access, with card reading distance of up to 50 mm
  - Supports for ISO 14443 Type A and B cards, Mifare®
  - Built-in anti-collision feature (only one tag is accessed at any time)
  - Selective card polling capability (especially useful when multiple cards are presented)
- Built-in Peripherals:
  - LED
  - Buzzer
- Firmware Upgradability
- Compliant with the following standards:
  - CE
  - FCC
  - RoHS

2.0. USB Interface

The ACR1281U-C8 is connected to a computer through USB as specified in the USB Specification 2.0. The ACR1281U-C8 is working in full speed mode, i.e.12 Mbps.

The USB interface is used for firmware upgrade purpose.

Pin	Signal	Function
1	VCC	+5V power supply for the reader
2	D-	Differential signal transmits data between reader and PC
3	D+	Differential signal transmits data between reader and PC
4	GND	Reference voltage level for power supply

Table 1: USB Interface Wiring

***Note**: In order for the ACR1281U-C8 to function properly through USB interface, ACS proprietary device drive has to be installed.*

3.0. Reader Commands

3.1. ACR120_Open

Opens a port (connection) to the reader.

High Level API:

```
DLLAPI INT16 AC_DECL ACR120_Open(INT16 ReaderPort);
```

Parameters	Description
ReaderPort	Port number. Available choices are “ACR120_USB1” to “ACR120_USB8”.
Return Value	INT16 Handle for further operations. Error Code < 0

Table 2: ACR120_Open Command Description

Low Level Frame Structure:

Command: From PC to Reader (5 bytes)

Command Code (1 byte)	Command Data Length (1 bytes)	Instruction Code (1 byte)	Data Length (1 byte)
E0h	02h	05h	00h

Response: From Reader to PC (5 bytes) << Success or Fail >>

Response Code (1 byte)	Response Status (2 bytes)	Response Data Length (2 bytes)
E1h	INT16	0000h

3.2. ACR120_Close

Closes the port (connection) to the reader.

High Level API:

```
DLLAPI INT16 AC_DECL ACR120_Close(UINT16 hReader);
```

Parameters	Description
hReader	Handle to the Reader returned by AC_Open()
Return Value	INT16 0 = success; Error Code < 0

Table 3: ACR120_Close Command Description

Low Level Frame Structure:

Command: From PC to Reader (0 Bytes) <<No Frame will be sent to Reader>>

Response: From Reader to PC (0 Bytes) <<No Frame will be sent to PC>>

3.3. ACR120_Reset

Resets the Mifare Chip of the reader, then restores the factory settings.

High Level API:

```
DLLAPI INT16 AC_DECL ACR120_Reset(UINT16 hReader);
```

Parameters	Description
hReader	Handle to the Reader returned by AC_Open()
Return Value	INT16 0 = success; Error Code < 0

Table 4: ACR120_Reset Command Description

Low Level Frame Structure:

Command: From PC to Reader (5 Bytes)

Command Code (1 byte)	Command Data Length (1 bytes)	Instruction Code (1 bytes)	Data Length (1 byte)
E0h	02h	05h	00h

Response: From Reader to PC (5 Bytes) << Success or Fail >>

Response Code (1 Byte)	Response Status (2 bytes)	Response Data Length (2 bytes)
E1h	INT16	0000h

3.4. ACR120_Status

Returns the firmware version and the reader status.

High Level API:

```
DLLAPI INT16 AC_DECL
ACR120_Status(UINT16     hReader,
              UINT8      pFirmwareVersion[20],
              STRUCT_STATUS  pReaderStatus);
```

Parameters	Description
hReader	Handle to the Reader returned by AC_Open()
pFirmwareVersion	Firmware version will be returned (20 bytes)
pReaderStatus	Reader status
Return Value	INT16 0 = success; Error Code < 0

Table 5: ACR120_Status Command Description

Low Level Frame Structure:

Command: From PC to Reader (5 Bytes)

Command Code (1 Byte)	Command Data Length (1bytes)	Instruction Code (1 byte)	Data Length (1 byte)
E0h	02h	18h	00h

Response: From Reader to PC (35 Bytes) << Success or Fail >>

Response Code (1 byte)	Response Status (2 bytes)	Response Data Length (2 bytes)	Response Data	
			(20 bytes)	(10 bytes)
E1h	INT16	01Eh	pFirmwareVersion	pReaderStatus

```
Struct STRUCT_STATUS
{
// 0x01 = Type A; 0x02 = Type B; 0x03 = Type A + Type B
UINT8    MifareInterfaceType;
// Bit 0 = Mifare Light; Bit 1 = Mifare1K; Bit 2 = Mifare 4K; Bit 3 =
Mifare DESFire
// Bit 4 = Mifare UltraLight; Bit 5 = JCOP30; Bit 6 = Shanghai Transport
// Bit 7 = MPCOS Combi; Bit 8 = ISO type B, Calypso
// Bit 9 – Bit 31 = To be defined
UINT32      CardsSupported;
UINT8    CardOpMode; // 0x00 = Type A; 0x01 = Type B TAG is being processed
           // 0xFF = No TAG is being processed.
UINT8    FWI;  // the current FWI value (time out value)
UINT8    RFU; // to be defined
UINT16      RFU; // to be defined
} ReaderStatus;
```

Remark:

1. For UINT32 CardSupported, the LSB is transmitted first.
   e.g. Byte 0(LSB), Byte 1, Byte 2 & Byte 3(MSB)

3.5. ACR120_ReadRC531Reg

Reads the Mifare® register.

High Level API:

```
DLLAPI INT16 AC_DECL
ACR120_ReadRC531Reg(INT16      hReader,
                    UINT8      RegNo,
                    UINT8*        pValue);
```

Parameters	Description
hReader	Handle to the Reader returned by AC_Open()
RegNo	Register number
pValue	Mifare® register's value
Return Value	INT16 0 = success; Error Code < 0

Table 6: ACR120_ReadRC531Reg Command Description

Low Level Frame Structure:

Command: From PC to Reader (6 bytes)

Command Code (1 byte)	Command Data Length (1 bytes)	Instruction Code (1 byte)	Data Length (1 byte)	Data (1 byte)
E0h	03h	19h	01h	RegNo

Response: From Reader to PC (6 bytes) << Success or Fail >>

Response Code (1 byte)	Response Status (2 bytes)	Response Data Length (2 bytes)	Response Data (1 byte)
E1h	INT16	0001h	pValue

3.6. ACR120_WriteRC531Reg

Writes the Mifare® register.

High Level API:

```
DLLAPI INT16 AC_DECL
ACR120_WriteRC531Reg(UINT16   hReader,
                     UINT8 RegNo,
                     UINT8 Value);
```

Parameters	Description
hReader	Handle to the Reader returned by AC_Open()
RegNo	Register number
Value	Mifare® register's value to write
Return Value	INT16 Result code. 0 means success.

Table 7: ACR120_WriteRC531Reg Command Description

Low Level Frame Structure:

Command: From PC to Reader (7 bytes)

Command Code (1 byte)	Command Data Length (1 bytes)	Instruction Code (1 byte)	Data Length (1 byte)	Data	
				(1 byte)	(1 byte)
E0h	04h	1Ah	02h	RegNo	Value

Response: From Reader to PC (6 bytes) << Success or Fail >>

Response Code (1 byte)	Response Status (2 bytes)	Response Data length (2 bytes)	Response Data (1 byte)
E1h	INT16	0001h	ValueStored

Note: *The Response Data (ValueStored) is used for comparison only.*

3.7. ACR120_DirectSend

Directly sends data to the Mifare® Chip.

High Level API:

```
DLLAPI INT16 AC_DECL
ACR120_DirectSend( UINT16  hReader,
                   UINT8 DataLength,
                   UINT8*   pData,
                   UINT8*   pResponseDataLength,
                   UINT8*   pResponseData,
                   UINT16   TimedOut,);
```

Parameters	Description
hReader	Handle to the Reader returned by AC_Open()
DataLength (N)	Data length (maximum of 66 bytes)
Data	Data to be sent
pResponseDataLength (K)	Response data length
pResponseData	Response Data
TimedOut	Time Out for waiting the response data (ms)
Return Value	INT16 0 = success; Error Code < 0

Table 8: ACR120_DirectSend Command Description

Low Level Frame Structure:

Command: From PC to Reader (N + 3) bytes; 2<=N<=66

Command Code (1 byte)	Command Data Length (1 bytes)	Data (N bytes)
E0h	DataLength (2<=N<=66)	N bytes of Data Maximum of 66 bytes

Response: From Reader to PC (K + 5) bytes <<Success or Fail>>

Response Code (1 byte)	Response Status (2 bytes)	Response Data Length (2 bytes)	Response Data (K bytes)
E1h	INT16	pResponseDataLength (K)	pResponseData

Note: *ACR120_DirectSend()* & *ACR120_DirectReceive()* must be used in a pair.

3.8. ACR120_DirectReceive

Directly receives data from the Mifare® Chip.

High Level API:

```
DLLAPI INT16 AC_DECL
ACR120_DirectReceive(UINT16      hReader,
                     UINT8    RespectedDataLength,
                     UINT8*      pReceivedDataLength,
                     UINT8*      pReceivedData,
                     UINT16      TimedOut,);
```

Parameters	Description
hReader	Handle to the Reader returned by AC_Open()
RespectedDataLength	Respected data length to be received (maximum 64 bytes)
pReceivedDataLength (K)	Data length of the received data
pReceivedData	Received data
TimedOut	Time Out for waiting the response data (ms)
Return Value	INT16 0 = success; Error Code < 0

Table 9: ACR120_DirectReceive Command Description

Low Level Frame Structure:

Command: From PC to Reader (5 Bytes)

Command Code (1 byte)	Command Data Length (1 bytes)	Data (2 bytes)
E0h	02h	RespectedDataLength <= 64 bytes

Response: From Reader to PC (K + 5 Bytes) <<Success or Fail>>

Response Code (1 byte)	Response Status (2 bytes)	Response Data Length (2 bytes)	Response Data (K bytes)
E1h	INT16	PReceivedDataLength (K)	pReceivedData

Note: *ACR120_DirectSend() & ACR120_DirectReceive() must be used in a pair.*

3.9. ACR120_RequestDLLVersion

Gets the reader's API DLL version information.

High Level API:

```
DLLAPI INT16 AC_DECL
ACR120_RequestDLLVersion(  UINT8*   pVersionInfoLength,
                           UINT8*   pVersionInfo);
```

Parameters	Description
pVersionInfoLength	Returns the length of the DLL version string
pVersionInfo	Returns the DLL version string
Return Value	INT16 0 = success; Error Code < 0

Table 10: ACR120_RequestDLLVersion

Low Level Frame Structure:

Command: From PC to Reader (0 Bytes) <<No Frame will be sent to Reader>>

Response: From Reader to PC (0 Bytes) <<No Frame will be sent to PC>>

3.10. ACR120_ReadEEPROM

Reads the internal EEPROM.

High Level API:

```
DLLAPI INT16 AC_DECL
ACR120_ReadEEPROM(   INT16    hReader,
                     UINT8 RegNo,
                     UINT8*   pEEPROMData);
```

Parameters	Description
hReader	Handle to the Reader returned by AC_Open()
RegNo	Register number
pEEPROMData	Contains the EEPROM register's value
Return Value	INT16 Result code. 0 means success.

Table 11: ACR120_ReadEEPROM Command Description

Low Level Frame Structure:

Command: From PC to Reader (6 bytes)

Command Code (1 byte)	Command Data Length (1 bytes)	Instruction Code (1 byte)	Data Length (1 byte)	Data (1 byte)
E0h	03h	08h	01h	RegNo

Response: From Reader to PC (6 bytes) <<Success or Fail>>

Response Code (1 byte)	Response Status (2 bytes)	Response Data Length (2 bytes)	Response Data (1 byte)
E1h	INT16	0001h	pEEPROMData

3.11. ACR120_WriteEEPROM

Writes the internal EEPROM.

High Level API:

```
DLLAPI INT16 AC_DECL
ACR120_WriteEEPROM(  INT16    hReader,
                     UINT8 RegNo,
                     UINT8 EEPROMData);
```

Parameters	Description
hReader	Handle to the Reader returned by AC_Open()
RegNo	Register number
EEPROMData	EEPROM register's value to write
Return Value	INT16 Result code. 0 means success.

Table 12: ACR120_WriteEEPROM Command Description

Low Level Frame Structure:

Command: From PC to Reader (7 bytes)

Command Code (1 byte)	Command Data Length (2 byte)	Instruction Code (1 byte)	Data Length (1 byte)	Data	
				(1 byte)	(1 byte)
E0h	04h	0Ch	02h	RegNo	EEPROMData

Response: From Reader to PC (6 bytes) <<Success >>

Response Code (1 byte)	Response Status (2 bytes)	Response Data Length (2 bytes)	Response Data (1 byte)
E1h	INT16 (>=0)	0001h	EEPROMDataStored

Note: *The Response Data (EEPROMDataStored) is used for comparison only.*

Or

Response: From Reader to PC (5 bytes) << Fail>>

Response Code (1 byte)	Response Status (2 bytes)	Response Data Length (2 byes)
E1h	INT16 (<0)	0000h

3.12. ACR120_ReadUserPort

Reads the state of user port.

High Level API:

```
DLLAPI INT16 AC_DECL
ACR120_ReadUserPort( INT16    hReader,
                     UINT8*   pUserPortState);
```

Parameters	Description
hReader	Handle to the reader returned by AC_Open()
pUserPortState	Contains the port state (only bit 2 and bit 6 are used)
Return Value	INT16 Result code. 0 means success.

Table 13: ACR120_ReadUserPort Command Description

Low Level Frame Structure:

Command: From PC to Reader (5 bytes)

Command Code (1 byte)	Command Data Length (2 bytes)	Instruction Code (1 byte)	Data Length (1 byte)
E0h	0002h	14h	00h

Response: From Reader to PC (6 bytes) <<Success or Fail>>

Response Code (1 byte)	Response Status (2 bytes)	Response Data Length (2 bytes)	Response Data (1 byte)
E1h	INT16	0001h	pUserPortState

UserPortState:

Bit 0-1: Not Used

Bit 2: Buzzer (0 = OFF; 1 = ON)

Bit 3-5: Not Used

Bit 6: LED (0 = OFF; 1 = ON)

Bit 7: Not Used

3.13. ACR120_WriteUserPort

Reads the state of user port.

High Level API:

```
DLLAPI INT16 AC_DECL
ACR120_WriteUserPort(   INT16    hReader,
                        UINT8 UserPortState);
```

Parameters	Description
hReader	Handle to the reader returned by AC_Open()
pUserPortState	Contains the port state to write (only bit 2 and bit 6 are used)
Return Value	INT16 Result code. 0 means success.

Table 14: ACR120_WriteUserPort Command Description

Low Level Frame Structure:

Command: From PC to Reader (6 bytes)

Command Code (1 byte)	Command Data Length (1 bytes)	Instruction Code (1 byte)	Data Length (1 byte)	Data (1 byte)
E0h	03h	15h	01h	UserPortState

Response: From Reader to PC (6 bytes) << Success or Fail >>

Response Code (1 byte)	Response Status (2 bytes)	Response Data Length (2 bytes)	Response Data (1 byte)
E1h	INT16	0001h	UserPortStateStored

Note: *The Response Data (UserPortStateStored) is used for comparison only.*

UserPortState:

Bit 0-1: Not Used

Bit 2: Buzzer (0 = OFF; 1 = ON)

Bit 3-5: Not Used

Bit 6: LED (0 = OFF; 1 = ON)

Bit 7: Not Used

3.14. ACR120_Power

Turns the antenna power on or off.

High Level API:

```
DLLAPI INT16 AC_DECL
ACR120_Power(INT16      hReader,
             INT8       State);
```

Parameters	Description
hReader	Handle to the reader returned by AC_Open()
State	Turn OFF (0) or ON (1)
Return Value	INT16 Result code. 0 means success.

Table 15: ACR120_Power Command Description

Low Level Frame Structure:

Command: From PC to Reader (5 bytes)

Command Code (1 byte)	Command Data Length (1 bytes)	Instruction Code (1 byte)	Data Length (1 byte)
E0h	02h	12 (if State=0) or 13 (if State=1)	00h

Response: From Reader to PC (6 bytes) <<Success or Fail>>

Response Code (1 byte)	Response Status (2 bytes)	Response Data Length (2 bytes)	Response Data (1 byte)
E1h	INT16	0001h	StateStored

Note: *The Response Data (StateStored) is used for comparison only.*

4.0. General Card Commands

Note: *All Card APIs involving SECTOR and BLOCK parameters please refer to* ***Appendix E*** *for further explanation.*

4.1. ACR120_Select

Selects a single card and returns the card ID (serial number).

High Level API:

```
DLLAPI INT16 AC_DECL
ACR120_Select( UINT16   hReader,
               UINT8*   pResultTagType,
               UINT8*   pResultTagLength,
               UINT8 pResultSN[10]);
```

Parameters	Description
hReader	Handle to the reader returned by AC_Open()
pResultTagType	Contains the selected TAG type
pResultTagLength	Contains the length of the selected TAG
pResultSN	If the *pResultTagLength* = 4 or 7 or 10, the *pSN* contains the selected card ID (Serial Number). The ID may be 4 or 7 or 10 bytes long.
Return Value	INT16 Result code. 0 means success.

Table 16: ACR120_Select Command Description

Low Level Frame Structure:

Command: From PC to Reader (5 Bytes)

Command Code (1 byte)	Command Data Length (1 byte)	Instruction Code (1 byte)	Data Length (1 byte)
E0h	02h	00h	00h

Response: From Reader to PC (18 Bytes) << Success>>

Response Code (1 byte)	Response Status (2 bytes)	Response Data Length (2 bytes)	Response Data			
			(1 byte)	(1 byte)	(1 byte)	(10 bytes)
E1h	INT16 (>=0)	000Dh	01h	pResultTagType	pResultTagLength	pResultSN (Serial Number)

Or

Response: From Reader to PC (5 bytes) << Fail>>

Response Code (1 byte)	Response Status (2 bytes)	Response Data Length (2 byes)
E1h	INT16 (<0)	0000h

4.2. ACR120_ListTags

Lists out the serial numbers of all tags which are in readable antenna range.

High Level API:

```
DLLAPI INT16 AC_DECL
ACR120_ListTags(  INT16    hReader,
                  UINT8*   pNumTagFound,
                  UINT8 pTagType[4],
                  UINT8 pTagLength[4],
                  UINT8 pSN[4][10]);
```

Parameters	Description
hReader	Handle to the reader returned by AC_Open()
pNumTagFound	Contains the number of TAG listed
pTagType[4]	Contains the TAG type
pTagLength[4]	Contains the length of the serial number
pSN[4][10]	The flat array of serial numbers. All serial numbers are concatenated with fixed length (10 bytes)
Return Value	INT16 Result code. 0 means success.

Table 17: ACR120_LIstTags Command Description

Low Level Frame Structure:

Command: From PC to Reader (5 bytes)

Command Code (1 byte)	Command Data Length (1 byes)	Instruction Code (1 byte)	Data Length (1 byte)
E0h	02h	03h	00h

Response: From Reader to PC (6 + N x 12 bytes) << Success or Fail >>

Response Code (1 byte)	Response Status (2 bytes)	Response Data Length (2 bytes)	Response Data (1 byte)
E1h	INT16 (>=0)	01h + N x 0Ch	pResultTagFound (N)

+ Response Data Block 0

Mifare Response Data Block [0]		
(1 byte)	(1 byte)	(10 bytes)
pTagType[0]	pTagLength[0]	pResultSN[0] (Serial Number)

+ Response Data Blocks 1… (N-1)

Mifare Response Data Block [N-1]		
(1 byte)	**(1 byte)**	**(10 bytes)**
pTagType[N-1]	pTagLength[N-1]	pResultSN[N-1] (Serial Number)

Or

Response: From Reader to PC (5 bytes) << Fail>>

Response Code (1 byte)	Response Status (2 bytes)	Response Data Length (2 byes)
E1h	INT16 (<0)	0000h

4.3. ACR120_MultiTagSelect

Selects a TAG with specific serial number.

High Level API:

```
DLLAPI INT16 AC_DECL
ACR120_MultiTagSelect(  INT16    hReader,
                        UINT8 TagLength,
                        UINT8 SN[10],
                        UINT8*   pResultTagType,
                        UINT8*   pResultTagLength,
                        UINT8*   pResultSN);
```

Parameters	Description
hReader	Handle to the reader returned by AC_Open()
TagLength (N)	Contains the length of the serial number of the TAG to be selected. The *TagLength* may be 4, 7 or 10 bytes long.
SN	Contains the serial number of the TAG to be selected
pResultTagType	Contains the selected TAG type
pResultTagLength (K)	Contains the length of the serial number of the selected TAG. The *pResultTagLength* may be 4, 7 or 10 bytes long.
pResultSN	Serial number of the selected TAG
Return Value	INT16 Result code. 0 means success.

Table 18: ACR120_MultiTagSelect Command Description

Low Level Frame Structure:

Command: From PC to Reader (Taglength(N)+6 bytes)

Command Code (1 byte)	Command Data Length (2 bytes)	Instruction Code (1 byte)	Data Length (1 byte)	Data	
				(1 byte)	(N byte)
E0h	Taglength (N) +0003h	03h	Taglength (N) +01h	TagLength (N)	SN

Response: From Reader to PC (18 Bytes) << Success>>

Response Code (1 byte)	Response Status (2 bytes)	Response Data Length (2 bytes)	Response Data			
			(1 byte)	(1 byte)	(1 byte)	(10 bytes)
E1h	INT16 (>=0)	000Dh	01h	pResultTagType	PResultTagLength (K)	pResultSN (Serial Number)

Or

Response: From Reader to PC (5 bytes) << Fail>>

Response Code (1 byte)	Response Status (2 bytes)	Response Data Length (2 byes)
E1h	INT16 (<0)	0000h

4.4. ACR120_TxDataTelegram

Sends data to the Mifare® Chip.

High Level API:

```
DLLAPI INT16 AC_DECL
ACR120_TxDataTelegram(INT16      hReader,
                      UINT8      SendDataLength,
                      UINT8*     pSendData
                      UINT8*     pReceivedDataLength,
                      UINT8*     pReceivedData);
```

Parameters	Description
hReader	Handle to the reader returned by AC_Open()
SendDataLength (N)	Length of data to be sent
pSendData	Data to be sent
pReceivedDataLength (K)	Length of received data
pReceivedData	Received data
Return Value	INT16 Result code. 0 means success.

Table 19: ACR120_TxDataTelegram Command Description

Low Level Frame Structure:

Command: From PC to Reader (N+5 bytes)

Command Code (1 Byte)	Command Data Length (2 bytes)	Instruction Code (1 byte)	Data Length (1 byte)	Data (N bytes)
E0h	N+0002h	16h	N	SendData

Response: From Reader to PC (K+5 bytes) <<Success >>

Response Code (1 byte)	Response Status (2 bytes)	Response Data Length (2 bytes)	Response Data (K bytes)
E1h	INT16 (>=0)	Kh	pReceivedData

Or

Response: From Reader to PC (5 bytes) << Fail>>

Response Code (1 byte)	Response Status (2 bytes)	Response Data Length (2 byes)
E1h	INT16 (<0)	0000h

5.0. Card Commands for Mifare® 1K/4K Cards

5.1. ACR120_Login

Performs an authentication to access one sector of the card. Only one sector can be accessed at a time.

High Level API:

```
DLLAPI INT16 AC_DECL
ACR120_Login(  INT16    hReader,
               UINT8 Sector,
               UINT8 KeyType,
               INT8     StoredNo,
               UINT8 pKey[6]);
```

Parameters	Description
hReader	Handle to the reader returned by AC_Open()
Sector	Sector number to login
KeyType	Type of key. It can be: AC_MIFARE_LOGIN_KEYTYPE_A, AC_MIFARE_LOGIN_KEYTYPE_B, AC_MIFARE_LOGIN_KEYTYPE_DEFAULT_A, AC_MIFARE_LOGIN_KEYTYPE_DEFAULT_B, AC_MIFARE_LOGIN_KEYTYPE_DEFAULT_F, AC_MIFARE_LOGIN_KEYTYPE_STORED_A and AC_MIFARE_LOGIN_KEYTYPE_STORED_B
StoredNo	The stored number of key if keyType = AC_MIFARE_LOGIN_KEYTYPE_STORED_A or AC_MIFARE_LOGIN_KEYTYPE_STORED_B.
pKey	The login key if keyType = AC_MIFARE_LOGIN_KEYTYPE_A or AC_MIFARE_LOGIN_KEYTYPE_B. It's AC_MIFARE_KEY_LEN(6) bytes long.
Return Value	INT16 Result code. 0 means success.

Table 20: ACR120_Login Command Description

Low Level Frame Structure:

Command: From PC to Reader (13 Bytes) <<Login Key is provided>>

If KeyType = AC_MIFARE_LOGIN_KEYTYPE_A or
AC_MIFARE_LOGIN_KEYTYPE_B.

Command Code (1 byte)	Command Data Length (1 bytes)	Instruction Code (1 byte)	Data Length (1 byte)	Data		
				(1 byte)	(1 byte)	(6 bytes)
E0h	0Ah	02h	08h	Sector	KeyType2	pKey

Or

Command: From PC to Reader (7 bytes) <<Login Key is not provided>>

If KeyType = AC_MIFARE_LOGIN_KEYTYPE_DEFAULT_A or,
AC_MIFARE_LOGIN_KEYTYPE_DEFAULT_B or
AC_MIFARE_LOGIN_KEYTYPE_DEFAULT_F or
AC_MIFARE_LOGIN_KEYTYPE_STORED_A or
AC_MIFARE_LOGIN_KEYTYPE_STORED_B.

Command Code (1 byte)	Command Data Length (2 bytes)	Instruction Code (1 byte)	Data Length (1 byte)	Data	
				(1 byte)	(1 byte)
E0h	0004h	02h	02h	Sector	KeyType2 or *StoredNo* + 10h (Used as Key A) or *StoredNo* + 30h (Used as Key B)

Response: From Reader to PC (5 bytes) <<Success or Fail>>

Response Code (1 byte)	Response Status (2 bytes)	Response Data Length (2 byes)
E1h	INT16	0000h

Constant Definition:

AC_MIFARE_LOGIN_KEYTYPE_A	0xAA (KeyType2 = 0xAA)
AC_MIFARE_LOGIN_KEYTYPE_B	0xBB (KeyType2 = 0xBB)
AC_MIFARE_LOGIN_KEYTYPE_DEFAULT_A	0xAD (KeyType2 = 0xAA)
AC_MIFARE_LOGIN_KEYTYPE_DEFAULT_B	0xBD (KeyType2 = 0xBB)
AC_MIFARE_LOGIN_KEYTYPE_DEFAULT_F	0xFD (KeyType2 = 0xFF)
AC_MIFARE_LOGIN_KEYTYPE_STORED_A	0xAF (KeyType2 = 0xAA)

AC_MIFARE_LOGIN_KEYTYPE_STORED_B	0xBF (KeyType2 = 0xBB)
AC_MIFARE_LOGIN_KEYTYPE_DEFAULT_F	0xFF

5.2. ACR120_Read

Reads a block.

High Level API:

```
DLLAPI INT16 AC_DECL
ACR120_Read(    INT16    hReader,
                UINT8 Block,
                UINT8 pBlockData[16]);
```

Parameters	Description
hReader	Handle to the reader returned by AC_Open()
Block	Block number
pblockData	Contains the data read. It is AC_MIFARE_DATA_LEN(16) bytes long.
Return Value	INT16 Result code. 0 means success.

Table 21: ACR120_Read Command Description

Low Level Frame Structure:

Command: From PC to Reader (6 bytes)

Command Code (1 byte)	Command Data Length (1 bytes)	Instruction Code (1 byte)	Data Length (1 byte)	Data (1 byte)
E0h	03h	06h	01h	Block

Response: From Reader to PC (21 bytes) <<Success>>

Response Code (1 byte)	Response Status (2 bytes)	Response Data Length (2 bytes)	Response Data (16 bytes)
E1h	INT16 (>=0)	0010h	pBlockData

Or

Response: From Reader to PC (5 bytes) << Fail>>

Response Code (1 byte)	Response Status (2 bytes)	Response Data Length (2 byes)
E1h	INT16 (<0)	0000h

5.3. ACR120_ReadValue

Reads a value.

High Level API:

```
DLLAPI INT16 AC_DECL
ACR120_ReadValue( INT16    hReader,
                  UINT8 Block,
                  INT32*   pValueData);
```

Parameters	Description
hReader	Handle to the reader returned by AC_Open()
Block	Block number
pValueData	Contains the value read. It is 32 bit signed integer.
Return Value	INT16 Result code. 0 means success.

Table 22: ACR120_ReadValue Command Description

Low Level Frame Structure:

Command: From PC to Reader (6 bytes)

Command Code (1 byte)	Command Data Length (1 byte)	Instruction Code (1 byte)	Data Length (1 byte)	Data (1 byte)
E0h	03h	07h	01h	Block

Response: From Reader to PC (9 bytes) <<Success>>

Response Code (1 byte)	Response Status (2 bytes)	Response Data Length (2 bytes)	Response Data (4 bytes)
E1h	INT16 (>=0)	0004h	pValueData

Or

Response: From Reader to PC (5 bytes) << Fail>>

Response Code (1 byte)	Response Status (2 bytes)	Response Data Length (2 byes)
E1	INT16 (<0)	0000

5.4. ACR120_Write

Writes a block.

High Level API:

```
DLLAPI INT16 AC_DECL
ACR120_Write(  INT16    hReader,
               UINT8 Block,
               UINT8 pBlockData[16]);
```

Parameters	Description
hReader	Handle to the reader returned by AC_Open()
Block	Block number
pblockData	Contains the data read. It is AC_MIFARE_DATA_LEN(16) bytes long.
Return Value	INT16 Result code. 0 means success.

Table 23: ACR120_Write Command Description

Low Level Frame Structure:

Command: From PC to Reader (22 bytes)

Command Code (1 byte)	Command Data Length (1 byte)	Instruction Code (1 byte)	Data Length (1 byte)	Data	
				(1 byte)	(16 bytes)
E0h	13h	0Ah	11h	Block	pBlockData

Response: From Reader to PC (21 bytes) <<Success>>

Response Code (1 byte)	Response Status (2 bytes)	Response Data length (2 bytes)	Response Data (16 bytes)
E1h	INT16 (>=0)	0010h	BlockDataStored

Note: *The Response Data (BlockDataStored) is used for comparison only.*

Or

Response: From Reader to PC (5 bytes) << Fail>>

Response Code (1 byte)	Response Status (2 bytes)	Response Data Length (2 byes)
E2h	INT16 (<0)	0000h

5.5. ACR120_WriteValue

Writes a value.

High Level API:

```
DLLAPI INT16 AC_DECL
ACR120_WriteValue(    INT16    hReader,
                      UINT8 Block,
                      INT32    ValueData);
```

Parameters	Description
hReader	Handle to the reader returned by AC_Open()
Block	Block number
ValueData	Contains the value to write. It is a 32-bit signed integer.
Return Value	INT16 Result code. 0 means success.

Table 24: ACR120_WriteValue Command Description

Low Level Frame Structure:

Command: From PC to Reader (10 bytes)

Command Code (1 byte)	Command Data Length (1 byte)	Instruction Code (1 byte)	Data Length (1 byte)	Data	
				(1 byte)	(4 bytes)
E0h	07h	0Bh	05h	Block	ValueData

Response: From Reader to PC (9 bytes) <<Success>>

Response Code (1 byte)	Response Status (2 bytes)	Response Data Length (2 bytes)	Response Data (4 bytes)
E1h	INT16 (>=0)	0004h	ValueDataStored

Note: *The Response Data (ValueDataStored) is used for comparison only.*

Or

Response: From Reader to PC (5 bytes) << Fail>>

Response Code (1 byte)	Response Status (2 bytes)	Response Data length (2 byes)
E2h	INT16 (<0)	0000h

5.6. ACR120_WriteMasterKey

Writes master keys.

High Level API:

```
DLLAPI INT16 AC_DECL
ACR120_WriteMasterKey(INT16      hReader,
                      UINT8      KeyNo,
                      UINT8      pKey[6]);
```

Parameters	Description
hReader	Handle to the reader returned by AC_Open()
KeyNo	Master key number
pKey	Key to write. It is AC_MIFARE_KEY_LEN(6) bytes long.
Return Value	INT16 Result code. 0 means success.

Table 25: ACR120_WriteMasterKey Command Description

Low Level Frame Structure:

Command: From PC to Reader (12 bytes)

Command Code (1 byte)	Command Data length (1 bytes)	Instruction Code (1 byte)	Data length (1 byte)	Data	
				(1 byte)	(6 bytes)
E0h	09h	0Dh	07h	KeyNo	pKey

Response: From Reader to PC (11 bytes) <<Success >>

Response Code	Response Status	Response Data Length	Response Data
(1 byte)	(2 bytes)	(2 bytes)	(6 bytes)
E1h	INT16 (>=0)	0006h	KeyStored

Note: *The Response Data (KeyStored) is used for comparison only.*

Or

Response: From Reader to PC (5 bytes) << Fail>>

Response Code	Response Status	Response Data Length
(1 byte)	(2 bytes)	(2 byes)
E1h	INT16 (<0)	0000h

5.7. ACR120_Inc

Increments a value block by adding a value to the value stored.

High Level API:

```
DLLAPI INT16 AC_DECL
ACR120_Inc   (  INT16    hReader,
                UINT8    Block,
                INT32    Value,
                INT32*   pNewValue);
```

Parameters	Description
hReader	Handle to the reader returned by AC_Open()
Block	Block number
Value	Value added to the block value
pNewValue	Updated value after increment
Return Value	INT16 Result code. 0 means success.

Table 26: ACR120_Inc Command Description

Low Level Frame Structure:

Command: From PC to Reader (10 bytes)

Command Code (1 byte)	Command Data Length (1 byte)	Instruction Code (1 byte)	Data Length (1 byte)	Data	
				(1 byte)	(4 bytes)
E0h	07h	0Fh	05h	Block	Value

Response: From Reader to PC (9 bytes) <<Success >>

Response Code (1 byte)	Response Status (2 bytes)	Response Data Length (2 bytes)	Response Data (4 bytes)
E1h	INT16 (>=0)	0004h	pNewValue

Or

Response: From Reader to PC (5 bytes) << Fail>>

Response Code (1 byte)	Response Status (2 bytes)	Response Data Length (2 byes)
E1h	INT16 (<0)	0000h

5.8. ACR120_Dec

Decrements a value block by subtracting a value stored.

High Level API:

```
DLLAPI INT16 AC_DECL
ACR120_Dec   (  INT16    hReader,
                UINT8 Block,
                INT32    Value,
                INT32*   pNewValue);
```

Parameters	Description
hReader	Handle to the reader returned by AC_Open()
Block	Block number
Value	Value subtracts
pNewValue	Updated value after decrement
Return Value	INT16 Result code. 0 means success.

Table 27: ACR120_Dec Command Description

Low Level Frame Structure:

Command: From PC to Reader (10 bytes)

Command Code (1 byte)	Command Data Length (1 byte)	Instruction Code (1 byte)	Data Length (1 byte)	Data	
				(1 byte)	(4 bytes)
E0h	07h	10h	05h	Block	Value

Response: From Reader to PC (9 bytes) <<Success >>

Response Code (1 byte)	Response Status (2 bytes)	Response Data Length (2 bytes)	Response Data (4 bytes)
E1h	INT16 (>=0)	0004h	pNewValue

Or

Response: From Reader to PC (5 bytes) << Fail>>

Response Code (1 byte)	Response Status (2 bytes)	Response Data Length (2 byes)
E1h	INT16 (<0)	0000h

5.9. ACR120_Copy

Copies a value block to another block of the same sector.

High Level API:

```
DLLAPI INT16 AC_DECL
ACR120_Copy(   INT16    hReader,
               UINT8 srcBlock,
               UINT8 desBlock,
               INT32*   pNewValue);
```

Parameters	Description
hReader	Handle to the reader returned by AC_Open()
srcBlock	Source block number
tgtBlock	Target block number
pNewValue	Updated value of the desBlock after copy
Return Value	INT16 Result code. 0 means success.

Table 28: ACR120_Copy Command Description

Low Level Frame Structure:

Command: From PC to Reader (7 bytes)

Command Code (1 byte)	Command Data length (1 byte)	Instruction Code (1 byte)	Data length (1 byte)	Data	
				(1 byte)	(1 byte)
E0h	04h	11h	02h	srcBlock	tgtBlock

Response: From Reader to PC (9 bytes) <<Success>>

Response Code (1 byte)	Response Status (2 bytes)	Response Data Length (2 bytes)	Response Data (4 bytes)
E1h	INT16 (>=0)	0004h	pNewValue

Or

Response: From Reader to PC (5 bytes) << Fail>>

Response Code (1 byte)	Response Status (2 bytes)	Response Data length (2 byes)
E1h	INT16 (<0)	0000h

6.0. Card Commands for ISO14443-4 Interface

6.1. PICC_Xch_APDU

Format:

```
DLLAPI INT16 AC_DECL PICC_Xch_APDU (
      INT16 rHandle,
      BOOL typeA,
      INT16 *pTransmitLength,
      UINT8 *pxData,
      INT16 *pReceiveLength,
      UINT8 *prData);
```

Function Description:

This function handles the APDU exchange in T=CL protocol. This routine will handle the Frame Waiting Time Extension (WTX) and chaining for long messages.

Parameters	Description
rHandle	Handle to the reader returned by ACR120_Open()
typeA	A Boolean value indicates the card type, TRUE for type A cards, FALSE for type B cards
pTransmitLength	A pointer to the location storing the length of the data to transmit, in bytes
pxData	A pointer to the transmit data storage
pReceiveLength	A pointer to the location storing the length of the data received, in bytes
prData	A pointer to the receive data storage
Return Value	INT16 Result code. 0 means success.

Table 29: PICC_Xch_APDU Command Description

Returns:

The return value is 0 if the function is successful. Otherwise, it returns a negative value containing the error code. For the detailed meaning of the error code, please refer to **Appendix A**.

Notes:

1. *The function `PICC_InitBlockNumber()` should be called each time between the `ACR120_Select()` or `ACR120_MultiTagSelect()` function and this function.*
2. *In many cases, the status code SW1 and SW2 are the last 2 bytes of the received data.*

Example:

```
INT16 rHandle;
UINT8 SID;
BOOT  typeA;
INT16 xLen, rLen;
UINT  rData[100];
```

```
    UINT8 Cmd[5]={0x94, 0xb2, 0x01, 0x3c, 0x1D};
    INT16 RetCode;

    xLen=5;
    SID=1;

    typeA = FALSE;    // Type B card

//Selects a single card and returns the card ID (Serial Number)
    retcode = ACR120_Select(rHandle, SID, &HaveTag, &tmpbyte, tmpArray);

if (retcode == 0)
{
    // If a card is selected, proceed to issue an APDU of 94B2013C1D
    PICC_InitBlockNumber(0);

       retcode  =  PICC_Xch_APDU(rHandle,  SID,  typeA,  &xLen,  Cmd,  &rLen,
       rData);
    //check if retcode is error

    if(retcode < 0){
       // Exchange APDU failed
    } else{
       // Exchange APDU successful
    }
}
```

6.2. Exchange ADPU Command

Low Level Frame Structure:

Command: From PC to Reader (5 bytes)

Command Code (1 Byte)	Command Data Length (1 bytes)	Instruction Code (1 byte)	Data length (1 byte)	Data	
				(1 byte)	(K byte)
E0h	03h	2Fh	K+1h	00h	APDU

Response: From Reader to PC (10 bytes) <<Success>>

Response Code (1 byte)	Response Status (2 bytes)	Response Data Length (2 bytes)	Data (K byte)
E1h	INT16	Kh	ADPU Response

Or

Response: From Reader to PC (5 bytes)

Response Code (1 byte)	Response Status (2 bytes)	Response Data Length (2 bytes)
E1h	INT16 (<0)	0000h

6.3. PICC_RATS

Format:

```
DLLAPI INT16 AC_DECL PICC_RATS (
      INT16 rHandle,
      UINT8 FSDI,
      UINT8 *pATSlen,
      UINT8 *pATS);
```

Function Description:

This function is only valid for ISO14443 type A cards. It requests an Answer-to-Select (ATS) message from the card after doing the ACR120_Select( ) operation. It tells the card how many bytes the reader can handle in a frame and also gets the operation parameters of the card when communicating in ISO14443 mode.

Parameters	Description
rHandle	Handle to the reader returned by ACR120_Open()
FSDI	Index to a maximum frame size which the reader can accept. The value should not exceed 4, i.e. 48 bytes.
pATSlen	Pointer to the location storing the length of the ATS received
pATS	Pointer to the ATS received
Return Value	INT16 Result code. 0 means success.

Table 30: PICC_RATS Command Description

FSDI	FSD (in bytes)
0	16
1	24
2	32
3	40
4	48
5	64
6	96
7	128
8	256
Otherwise	RFU

Table 31: FSDI to (Frame Size for Proximity Coupling Device) FSD Conversion

Returns:

The return value is zero if the function is successful. Otherwise, it returns a negative value containing the error code. For the detailed meaning of the error code, please refer to Appendix A.. For detailed meaning of the ATS, please refer to corresponding documents.

Note: *There is no need for calling this function in Type B cards.*

6.4. Auto-RATS

Low Level Frame Structure:

Command: From PC to Reader (5 bytes)

Command Code (1 Byte)	Command Data Length (1 bytes)	Instruction Code (1 byte)	Data Length (1 byte)	Data (1 byte)
E0h	03h	17h	01h	bEnable

bEnable (1 Byte):

0x00h – Disable RATS During Power On Command

0x01h – Enable RATS During Power On Command

Response: From Reader to PC (5 bytes)

Response Code (1 byte)	Response Status (2 bytes)	Response Data Length (2 byes)
E1h	INT16 (<0)	0000h

6.5. Firmware Upgrade Mode

Low Level Frame Structure:

Command: From PC to Reader (5 bytes)

Command Code (1 Byte)	Command Data length (1 bytes)	Instruction Code (1 byte)	Data Length (1 byte)
E0h	02h	20h	00h

Response: From Reader to PC (5 bytes)

Response Code (1 byte)	Response Status (2 bytes)	Response Data Length (2 byes)
E1h	INT16 (<0)h	0000h

Appendix A. Error Codes returned by High Level APIs

SUCCESS_READER_OP(0)

Successful operation. No Error Found.

#Handled by the DLL. The DLL has to do the consistent checking even a "Success Response Status" is returned by the device.

#Corresponding to the << Response Status 'L', 'P', 'A' & 'G' >>.

ERR_INTERNAL_UNEXPECTED(-1000)

Library internal unexpected error.

#Handled by the DLL

ERR_PORT_INVALID(-2000)

The port is invalid.

#Handled by the DLL

ERR_PORT_OCCUPIED(-2010)

The port is occupied by another application.

#Handled by the DLL

ERR_HANDLE_INVALID(-2020)

The handle is invalid.

#Handled by the DLL

ERR_INCORRECT_PARAM(-2030)

Incorrect Parameter.

#Handled by the DLL.

ERR_READER_NO_TAG(-3000, or 0xF448)

No TAG in reachable range / selected.

#Corresponding to the << Response Status 'N' >>.

ERR_READER_OP_FAILURE(-3030, or 0xF42A)

Operation failed.

#Corresponding to the << Response Status 'F' >>.

ERR_READER_UNKNOWN(-3040, or 0xF420)

Reader unknown error.

#Corresponding to the << Response Status 'C', 'O', 'X' & '?' >>.

ERR_READER_LOGIN_INVALID_STORED_KEY_FORMAT(-4010, or 0xF056)

Invalid stored key format in login process.

#Handled by the DLL.

ERR_READER_LOGIN_FAIL(-4011, or 0xF055)

Login failed.

#Corresponding to the << Response Status 'I' >>.

ERR_READER_OP_AUTH_FAIL(-4012, or 0xF054)

The operation or access is not authorized.

#Corresponding to the << Response Status 'I' >>.

ERR_READER_VALUE_DEC_EMPTY(-4030, or 0xF042)

Decrement failure (empty).

#Corresponding to the << Response Status 'E' >>.

ERR_READER_VALUE_INC_OVERFLOW(-4031, or 0xF041)

Increment Overflow.

#Corresponding to the << Response Status 'E' >>.

ERR_READER_VALUE_OP_FAILURE (-4032, 0xF040)

Value Operations failure. E.g. Value Increment

#Corresponding to the << Response Status 'I' >>.

ERR_READER_VALUE_INVALID_BLOCK(-4033, 0xF03F)

Block doesn't contain value.

#Corresponding to the << Response Status 'F' >>.

ERR_READER_VALUE_ACCESS_FAILURE (-4034, 0xF03E)

Value Access failure.

#Corresponding to the << Response Status 'U' >>.

Appendix B. Possible TAG Types

TAG Type Value	TAG Type Description	TAG SN Length
0x01h	Mifare Light	4
0x02h	Mifare 1K	4
0x03h	Mifare 4K	4
0x04h	Mifare DESFire	7
0x05h	Mifare Ultralight	7
0x06h	JCOP30	4
0x07h	Shanghai Transport	4
0x08h	MPCOS Combi	4
0x80h	ISO Type B, Calypso	4

Table 32: Possible TAG Types

Appendix C. USB ID and Drivers for ACR1281U-C8

- VID_0x072F & PID_0x8003 as the USB ID of ACR1281U-C8
- ACR120US.SYS will be used as the driver name for ACR1281U-C8
- ACR120UT.SYS will be used as the driver name for ACR1281U-C8

Appendix D. Standard Program Flow

1. Before any card commands, get the Reader Handle first.
2. Select the TAG.
3. Login the TAG.
4. Access the TAG.
5. Close the Reader Handle.

// ACR120_Sample.c; a very simple program

```
#include "acr120.h"

void main(void)
{
   INT16 hReader;
   UINT8 Length, SN[10], Data[16], Type;
     // Open a communication channel (USB Interface)
     hReader=ACR120_Open(ACR120_USB1);

   if(hReader<0){ // error happened!!! };

   // Assume the Reader Handle is ready, then "Select a TAG"
    ACR120_Select(hReader, &Type, &Length, SN);

   // Assume a TAG is selected, then "Login Sector 0x00"
    ACR120_Login(hReader, 0x00, AC_MIFARE_LOGIN_KEYTYPE_DEFAULT_A,0,NULL);

   // Assume the Sector is authorized, then "Read data from Block 0x02 of
Sector 0x00"
     ACR120_Read(hReader, 0x02, Data);

/*
Some operations.
*/
     ACR120_Close(hReader); // Close the port and quit the program
     return;
}
```

Appendix E. Physical and Logical Block/Sector Calculation

1. Mifare 1K
   - Logical Sector is equal to Physical sector, which are 0 to 15
   - Logical block of each sector is from 0 to 3
   - Physical blocks = ((Sector * 4) + Logical block)
2. Mifare 4K
   - Case 1: If {0 <= Logical Sector <= 31}
     - Physical sector is equal to Logical
     - Logical block of each sector is from 0 to 3
     - Physical blocks = ((Sector * 4) + Logical block)
   - Case 2: If {32 <= Logical Sector <= 39}
     - Physical Sector = Logical Sector + ((Logical Sector - 32) * 3)
     - Logical block of each sector is from 0 to 15
     - Physical blocks = ((Logical Sector - 32) * 16) + 128 + Logical block