

Advanced Card Systems Ltd.



ACR120 Contactless Reader/Writer



APPLICATION PROGRAMMING INTERFACE

Version 1.4 03-2005

Advanced Card Systems Ltd.
Unit 2910-2913, 29/F, The Center,
99 Queen's Road Central, Hong Kong

Tel: +852 2796 7873
Website: www.acs.com.hk

Fax: +852 2796 1286
Email: info@acs.com.hk

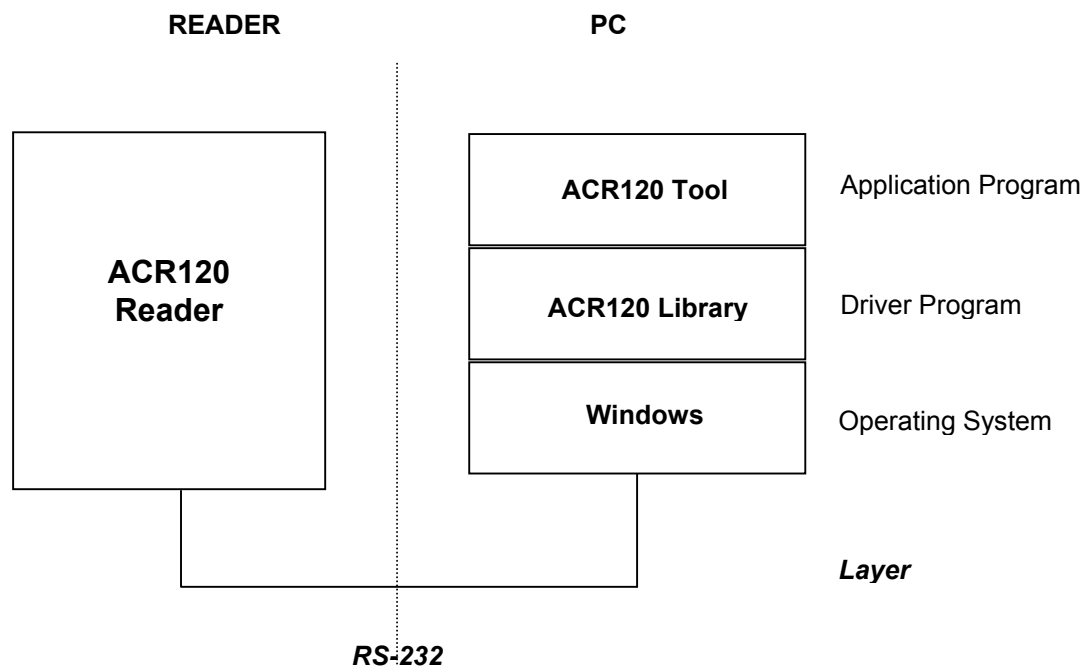
Contents

1. Introduction.....	3
2. ACR120.....	4
2.1 Overview	4
2.2 Communication Speed	4
2.3 ACR120 API	4
2.3.1 <i>Interface Function Prototypes</i>	4
2.3.1.1 ACR120_Open	4
2.3.1.2 ACR120_Close	5
2.3.1.3 ACR120_Reset	6
2.3.1.4 ACR120_Select	6
2.3.1.5 ACR120_Login.....	8
2.3.1.6 ACR120_Read.....	10
2.3.1.7 ACR120_ReadValue	13
2.3.1.8 ACR120_ReadEEPROM.....	15
2.3.1.9 ACR120_ReadLowLevelRegister	16
2.3.1.10 ACR120_Write.....	17
2.3.1.11 ACR120_WriteValue	19
2.3.1.12 ACR120_WriteEEPROM.....	21
2.3.1.13 ACR120_WriteLowLevelRegister	22
2.3.1.14 ACR120_WriteMasterKey	22
2.3.1.15 ACR120_Inc.....	23
2.3.1.16 ACR120_Dec.....	25
2.3.1.17 ACR120_Copy	27
2.3.1.18 ACR120_Power.....	29
2.3.1.19 ACR120_ReadUserPort.....	30
2.3.1.20 ACR120_WriteUserPort.....	30
2.3.1.21 ACR120_GetID	31
2.3.1.22 ACR120_ListTag.....	32
2.3.1.23 ACR120_MultiTagSelect	33
2.3.1.24 ACR120_TxDataTelegram.....	35
2.3.1.25 ACR120_RequestVersionInfo	35
Appendix A: Table of Error Codes	37
Appendix B: Sector Number Adaptation on Mifare 4K Card	38

1. Introduction

This manual describes the use of ACR120 interface software to program the ACR120 readers. It is a set of library functions implemented for the application programmers to operate the ACR120 readers and the presented cards. Currently, it is supplied in the form of 32-bit DLL (for Windows 98/2K/XP). It can be programmed using the popular development tools like Visual C/C++, Visual Basic, Delphi, etc... ACR120 readers can be connected to the PC via the RS/232 interface.

The architecture of the ACR120 library can be visualized as the following diagram:



2. ACR120

2.1 Overview

ACR120 is a set of high-level functions provided for the application software to use. It provides a consistent application-programming interface (ACR120 API) for the application to operate on the ACR120 reader and the corresponding presented card. ACR120 communicates with the ACR120 reader via the communication port facilities provided by the operating system.

2.2 Communication Speed

The ACR120 library controls the communication speed between the reader and the PC. The default communication baud rate (factory setting) is 9600bps, no parity, eight bits and one stop bits. A higher speed of 115200bps can be achieved by using software command issuing from the host. If you are not sure about the factory setting of your readers, you can use the Analyze Reader Function of ACR120 Tools to detect the current ACR120 reader settings.

2.3 ACR120 API

The ACR120 Application Programming Interface (API) defines a common way of accessing the ACR120 reader. Application programs invoke ACR120 reader through the interface functions and perform operations on the presented card.

The header file ACR120.h is available for the program developer, which contains all the function prototypes and macros described below.

2.3.1 Interface Function Prototypes

Generally, a program is required to call ACR120_Open first to obtain a handle. The handle is required for all ACR120 function call except for ACR120_Open.

2.3.1.1 ACR120_OPEN

Format:

```
DLLEAPI INT16 AC_DECL ACR120_Open ( INT16 ReaderPort,  
                                     INT16 BaudRate );
```

Function Description:

This function opens the port (connection) to ACR120 reader.

Parameters	Description
ReaderPort	The port number where the ACR120 reader is connected. Available choices are "ACR120_COM1" to "ACR120_COM8".

Parameters	Description	
BaudRate	The port baud rate. Available choices are "ACR120_COM_BAUDRATE_9600" to "ACR120_COM_BAUDRATE_115200".	
Return Value	INT16	Result code: 0 means success.

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.

Example:

// open a port to an ACR120 reader connected at COM1 with a baud rate of 9600 bps.

```
INT16 rHandle;

rHandle = ACR120_Open( ACR120_COM1,
                      ACR120_COM_BAUDRATE_9600 );
```

2.3.1.2 ACR120_CLOSE**Format:**

```
DLLAPI INT16 AC_DECL ACR120_Close ( INT16 rHandle );
```

Function Description:

This function closes the port (connection) to ACR120 reader.

Parameters	Description	
rHandle	The handle to ACR120 reader returned by ACR120_Open	
Return Value	INT16	Result code: 0 means success.

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.

Example:

// close the port (connection) to ACR120 reader.

```
INT16 RetCode;

RetCode = ACR120_Close ( rHandle );
```

2.3.1.3 ACR120_RESET

Format:

```
DLLAPI INT16 AC_DECL ACR120_Reset ( INT16 rHandle, UINT8 stationID );
```

Function Description:

This function resets the reader.

Parameters	Description	
rHandle	The handle to ACR120 reader returned by ACR120_Open.	
stationID	The StationID of ACR120 Reader.	
Return Value	INT16	Result code: 0 means success.

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.

Example:

```
// reset the reader (reader stationID:1)
```

```
INT16 RetCode;
```

```
RetCode = ACR120_Reset ( rHandle, 1 );
```

2.3.1.4 ACR120_SELECT

Format:

```
DLLAPI INT16 AC_DECL ACR120_Select ( INT16  rHandle,
                                     UINT8   stationID,
                                     BOOL*    pHaveTag,
                                     UINT8*   pTAG,
                                     UINT8   pSN[ACR120_SN_LEN] );
```

Function Description:

This function Selects a single card in range and returns the card ID (Serial Number).

Parameters	Description
rHandle	The handle to ACR120 reader returned by ACR120_Open.
stationID	The StationID of ACR120 Reader.
pHaveTag	Output Variable that will indicate whether the TAG Type Identification is returned: (TRUE) or (FALSE).

Parameters	Description	
pTAG	Output Variable that will contain the TAG Type Identification if returned (*pHaveTag = TRUE).	
pSN	Output Variable that will contain the card ID (Serial Number), AC_MIFARE_SN_LEN_4 (4 bytes long), AC_MIFARE_SN_LEN_7 (7 bytes long), AC_MIFARE_SN_LEN (10 bytes long).	
Return Value	INT16	Result code. 0 means success.

TAG Type Identification:

Tag Type Value	Tag Type Description	Serial Number Length
0x01	Mifare Light	4
0x02	Mifare 1K	4
0x03	Mifare 4K	4
0x04	Mifare DESFire	7
0x05	Mifare UltrLight	7
0x06	JCOP30	4
0x07	Shanghai Transport	4
0x08	MPCOS Combi	4
0x80	ISO type B, Calypso	4

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.

Notes:

You have to select the card first before you can Login and manipulate the card.

When there's more than one card in antenna range, you can use ACR120_MultiTagSelect.

Example:

```
// Select a single card in range (reader stationID: 1)
```

```
INT16 RetCode;
```

```
UINT8 SID;
BOOL pHaveTag;
UINT8 pTAG;
UINT8 pSN[3];
CString StrMsg;
```

```
SID = 1;
```

```
RetCode = ACR120_Select ( rHandle, SID, &pHaveTag, &pTAG, pSN );
```

```
// Get Serial Number Returned
```

```
StrMsg.Format("Card Serial: %X %X %X %X",pSN[0],pSN[1],pSN[2],pSN[3] );
```

2.3.1.5 ACR120_LOGIN

Format:

```

DLLAPI INT16 AC_DECL ACR120_Login ( INT16    rHandle,
                                     UINT8     stationID,
                                     UINT8     sector,
                                     UINT8     keyType,
                                     INT        storedNo,
                                     UINT8     pKey[ACR120_KEY_LEN] );

```

Function Description:

This function performs authentication to access one sector of the card. Only one sector can be accessed at a time.

Parameters	Description
rHandle	The handle to ACR120 reader returned by ACR120_Open
stationID	The StationID of ACR120 Reader.
Sector *	The sector number to login in.
keyType	The type of key. It can be: ACR120_LOGIN_KEYTYPE_AA, ACR120_LOGIN_KEYTYPE_BB, ACR120_LOGIN_KEYTYPE_FF, ACR120_LOGIN_KEYTYPE_STORED_A and ACR120_LOGIN_KEYTYPE_STORED_B
storedNo	The stored no. of key to use, IF keyType = ACR120_LOGIN_KEYTYPE_STORED_A or ACR120_LOGIN_KEYTYPE_STORED_B.
pKey	The login key, IF keyType = ACR120_LOGIN_KEYTYPE_AA or ACR120_LOGIN_KEYTYPE_BB. ACR120_KEY_LEN is 6 bytes long.
Return Value	INT16 Result code. 0 means success.

* Please refer to Appendix B for logging in Mifare 4K cards.

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.

Notes:

If keyType = ACR120_LOGIN_KEYTYPE_AA, or
If keyType = ACR120_LOGIN_KEYTYPE_BB,

Then storedNo. will not be used and can be just zero. While pKey must contain the 6 bytes key.

If keyType = ACR120_LOGIN_KEYTYPE_FF

Then the transport code: 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF will be use.

If keyType = ACR120_LOGIN_KEYTYPE_STORED_A, or

If keyType = ACR120_LOGIN_KEYTYPE_STORED_B,

Then pKey will not be use and can be just zero's while storedNo is the keyNo of the MasterKey you want to use. (Refer to ACR120_WriteMasterKey)

Before you can manipulate the card e.g read, write, copy, readvalue writevalue etc.. You have to successfully Login first to the card sector you want to manipulate.

Example:

```
// Login to sector 1 using keyType ACR120_LOGIN_KEYTYPE_AA
```

```
// (reader stationID: 1)
```

```
INT16 RetCode;
```

```
UINT8 SID;
```

```
UINT8 sector;
```

```
UINT8 keyType;
```

```
Int storedNo;
```

```
UINT8 pKey[5];
```

```
SID = 1;
```

```
sector = 1;
```

```
keyType = ACR120_LOGIN_KEYTYPE_AA
```

```
storedNo = 0;
```

```
pKey[0] = 255;
```

```
pKey[1] = 255;
```

```
pKey[2] = 255;
```

```
pKey[3] = 255;
```

```
pKey[4] = 255;
```

```
pKey[5] = 255;
```

```
RetCode = ACR120_Login(rHandle, SID, sector, keyType, storedNo, pKey);
```

```
// Login to sector 1 using keyType ACR120_LOGIN_KEYTYPE_FF
```

```
// (reader stationID: 1)
```

```
INT16 RetCode;
```

```
UINT8 SID;
```

```
UINT8 sector;
```

```
UINT8 keyType;
```

```
Int storedNo;
```

```
UINT8 pKey[5];
```

```

SID = 1;
sector = 1;
keyType = ACR120_LOGIN_KEYTYPE_AA
storedNo = 0;

RetCode = ACR120_Login(rHandle, SID, sector, keyType, storedNo, pKey);

```

```

// Login to sector 1 using keyType ACR120_LOGIN_KEYTYPE_STORED_A
// masterkey is stored to ( keyNo: 3 ) using the ACR120_WriteMasterKey
// (reader stationID: 1)

```

```
INT16 RetCode;
```

```

UINT8 SID;
UINT8 sector;
UINT8 keyType;
Int storedNo;
UINT8 pKey[5];

```

```

SID = 1;
sector = 1;
keyType = ACR120_LOGIN_KEYTYPE_STORED_A
storedNo = 3;

```

```
RetCode = ACR120_Login(rHandle, SID, sector, keyType, storedNo, pKey);
```

2.3.1.6 ACR120_READ

Format:

```

DLLAPI INT16 AC_DECL ACR120_Read ( INT16    rHandle,
                                   UINT8    stationID,
                                   UINT8    block,
                                   UINT8    pBlockData[ACR120_DATA_LEN]);

```

Function Description:

This function reads a block within the sector where you Login.

Parameters	Description	
rHandle	The handle to ACR120 reader returned by ACR120_Open	
stationID	The StationID of ACR120 Reader.	
block	The block number you want to read.	
pBlockData	Output Variable that will Contain the data read. ACR120_DATA_LEN is 16 bytes long.	
Return Value	INT16	Result code. 0 means success.

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.

Notes:

Memory Organization is based from Standard Card IC MF1 IC S50 which is 16 sectors with 4 blocks of 16 bytes each.

		Byte Number within a Block															
Sector	Block	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
15	3	Key A					Access Bits					Key B					
	2																
	1																
	0																
14	3	Key A					Access Bits					Key B					
	2																
	1																
	0																
:	:																
:	:																
:	:																
1	3	Key A					Access Bits					Key B					
	2																
	1																
	0																
0	3	Key A					Access Bits					Key B					
	2																
	1																
	0																

For you to access the exact block, you have to multiply the sector number by 4 plus the block number :

Block = (Sector * 4) + BlockNumber

Example:

// Read block 1 of sector 1 (reader stationID: 1)

// let's assume we've successfully Login to sector 1

```
INT16 RetCode;
```

```
UINT8 SID;
```

```
UINT8 block;
```

```

UINT8 pBlockData[16];
CString StrMsg;

SID = 1;
block = (1 * 4) + 1
RetCode = ACR120_Read(rHandle, SID, block, pBlockData);

// Data Read
StrMsg.Format("Data Read: %X %X %X %X %X %X %X %X %X %X %X %X %X %X %X",
              pBlockData[0],pBlockData[1],
              pBlockData[2],pBlockData[3],
              pBlockData[4], pBlockData[5],
              pBlockData[6], pBlockData[7],
              pBlockData[8], pBlockData[9],
              pBlockData[10],pBlockData[11],
              pBlockData[12],pBlockData[13],
              pBlockData[14],pBlockData[15]);

// Read block 2 of sector 4 (reader stationID: 1)
// let's assume we've successfully Login to sector 4

INT16 RetCode;

UINT8 SID;
UINT8 block;
UINT8 pBlockData[16];
CString StrMsg;
SID = 1;
block = (4 * 4) + 2

RetCode = ACR120_Read(rHandle, SID, block, pBlockData);

// Data Read
StrMsg.Format("Data Read: %X %X %X %X %X %X %X %X %X %X %X %X %X %X %X",
              pBlockData[0],pBlockData[1],
              pBlockData[2],pBlockData[3],
              pBlockData[4], pBlockData[5],
              pBlockData[6], pBlockData[7],
              pBlockData[8], pBlockData[9],
              pBlockData[10],pBlockData[11],
              pBlockData[12],pBlockData[13],
              pBlockData[14],pBlockData[15]);

```

2.3.1.7 ACR120_READVALUE

Format:

```

DLLAPI INT16 AC_DECL ACR120_ReadValue( INT16    rHandle,
                                         UINT8    stationID,
                                         UINT8    block,
                                         INT32*   pValueData);

```

Function Description:

This function reads value block within the sector where you Login.

Parameters	Description	
rHandle	The handle to ACR120 reader returned by ACR120_Open	
stationID	The StationID of ACR120 Reader.	
block	The value block number you want to read.	
pValueData	Output Variable that will Contain the value read.	
Return Value	INT16	Result code. 0 means success.

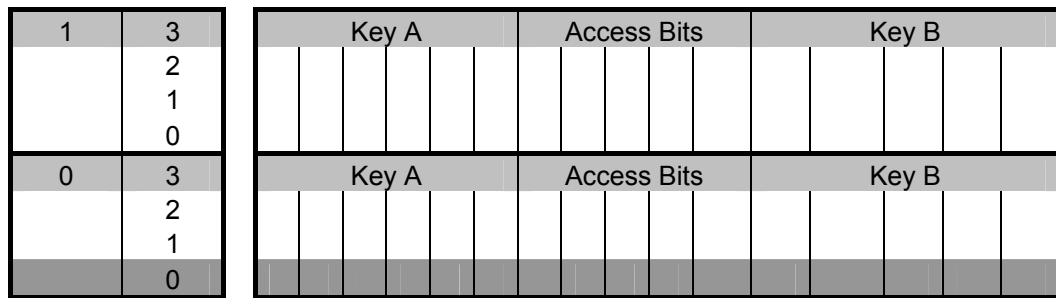
Returns:

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.

Notes:

Memory Organization is based on Standard Card IC MF1 IC S50, which are 16 sectors with 4 blocks of 16 bytes each.

		Byte Number within a Block															
Sector	Block	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
15	3	Key A				Access Bits				Key B							
	2																
	1																
	0																
14	3	Key A				Access Bits				Key B							
	2																
	1																
	0																
:	:																
:	:																
:	:																



For you to access the exact block, you have to multiply the sector number by 4 plus the block number :
 Block = (Sector * 4) + BlockNumber.

The difference between the ACR120_Read and ACR120_ReadValue is that the ACR120_Read reads the 16 Bytes data within the block while ACR120_ReadValue reads the INT32 value in the value block (block that was formatted by ACR120_WriteValue). "The block must be a value before reading; refer to ACR120_WriteValue"

Example:

```
// Read value of block 1 of sector 1 (reader stationID: 1)
// Let's assume logging into sector 1 was successful and a value is written to
// block 1 using ACR120_WriteValue

INT16 RetCode;

UINT8 SID;
UINT8 block;
UINT32 pValueData;
CString StrMsg;

SID = 1;
block = (1 * 4) + 1

RetCode = ACR120_ReadValue(rHandle, SID, block, &pValueData);

// Value Read
StrMsg.Format("Value Read: %d",pValueData);

// Read value of block 2 of sector 4 (reader stationID: 1)
// Let's assume logging into sector 4 was successful and a value is written to
// block 2 using ACR120_WriteValue

INT16 RetCode;

UINT8 SID;
```

```
UINT8 block;
UINT32 pValueData;
CString StrMsg;

SID = 1;
block = (4 * 4) + 2;

RetCode = ACR120_ReadValue(rHandle, SID, block, &pValueData);

// Value Read
StrMsg.Format("Value Read: %d", pValueData);
```

2.3.1.8 ACR120_READEEPROM

Format:

```
DLLAPI INT16 AC_DECL ACR120_ReadEEPROM ( INT16    rHandle,
                                          UINT8    stationID,
                                          UINT8    reg,
                                          UINT8*   pEEPROMData);
```

Function Description:

This function reads the internal EEPROM of the ACR120 reader

Parameters	Description
rHandle	The handle to ACR120 reader returned by ACR120_Open
stationID	The StationID of ACR120 Reader.
reg	The register number.
pEEPROMData	Output Variable that will Contain the EEPROM register's value.
Return Value	INT16 Result code. 0 means success.

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.

Notes:

The details for the register map is shown below:

ACR120 Reader Module EEPROM Memory Organization		
Register Number	Name	Description
00h...03h	Unique device ID (32bit)	This number is unique for each device and therefore read only.
04h	Station ID	Indicates the address ID for every station. The ID is used for addressing within a party line.
05h	Protocol Configuration	Set Protocol type, power on behavior. 00h -> ACR120 reader in ASCII mode 01h -> ACR120 reader in Binary mode
06h	Baud Rate Selection	Defines Communication speed. 00h -> 9600 baud 01h -> 19200 baud 02h -> 38400 baud 03h -> 57600 baud
07h...0Fh	Reserved	
10h...13h	User Date	Free Usage

Example:

// Read Baud rate (register 06h) of EEPROM (reader stationID: 1)

```

INT16 RetCode;

UINT8 SID;
UINT8 reg;
UINT8 pEEPROMData;
CString StrMsg;

SID = 1;
reg = 6;

RetCode = ACR120_ReadEEPROM (rHandle, SID, reg, &pEEPROMData);

// Value Read
StrMsg.Format("EEPROM Data Read:: %d",pEEPROMData);

```

2.3.1.9 ACR120_READLOWLEVELREGISTER**Format:**

```

ACR120_DLLAPI INT16 ACR120_DECLACR120_ReadLowLevelRegister(
    INT16      hReader,
    UINT8      stationID,
    UINT8      reg,
    UINT8*     pRegData);

```

* This command should be used under manufacturer's recommendation.

Function Description:

This function reads the internal register value.

Parameters	Description	
hReader	The handle to our reader returned by ACR120_Open.	
stationID	The station ID of our reader.	
reg	The register number.	
pRegData	Contains the register's value.	
Return Value	INT16	Result code. 0 means success.

2.3.1.10 ACR120_WRITE**Format:**

```

DLLAPI INT16 AC_DECL ACR120_Write ( INT16    rHandle,
                                     UINT8     stationID,
                                     UINT8     block,
                                     UINT8     pBlockData[ACR120_DATA_LEN] );

```

Function Description:

This function reads a block within the sector where you Login.

Parameters	Description	
rHandle	The handle to ACR120 reader returned by ACR120_Open	
stationID	The StationID of ACR120 Reader.	
block	The block number where you want to write.	
pBlockData	The 16 bytes Data to Write. ACR120_DATA_LEN is 16 bytes long.	
Return Value	INT16	Result code. 0 means success.

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.

Notes:

Memory Organization is based on Standard Card IC MF1 IC S50, which are 16 sectors with 4 blocks of 16 bytes each.

		Byte Number within a Block															
Sector	Block	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
15	3	Key A				Access Bits				Key B							

[illegible]

For you to access the exact block, you have to multiply the sector number by 4 plus the block number :

$$\text{Block} = (\text{Sector} * 4) + \text{BlockNumber}$$

Example:

```
// Write to block 1 of sector 1 (reader stationID: 1)
```

```
// Let's assume logging into sector 1 was successful
```

```
INT16 RetCode;
```

```
UINT8 SID;  
UINT8 block;  
UINT8 pBlockData[16];  
CString StrMsg;
```

```
SID = 1;
block = (1 * 4) + 1
```

```
pBlockData[0] = 255;
pBlockData[1] = 255;
pBlockData[2] = 255;
pBlockData[3] = 255;
pBlockData[4] = 255;
pBlockData[5] = 255;
pBlockData[6] = 255;
pBlockData[7] = 255;
pBlockData[8] = 255;
pBlockData[9] = 255;
```

```

pBlockData[10] = 255;
pBlockData[11] = 255;
pBlockData[12] = 255;
pBlockData[13] = 255;
pBlockData[14] = 255;
pBlockData[15] = 255;

RetCode = ACR120_Write(rHandle, SID, block, pBlockData);

```

2.3.1.11 ACR120_WRITEVALUE

Format:

```

DLLAPI INT16 AC_DECL ACR120_WriteValue( INT16    rHandle,
                                         UINT8    stationID,
                                         UINT8    block,
                                         INT32    ValueData);

```

Function Description:

This function writes INT32 value to a block within the sector where you Login.

Parameters	Description	
rHandle	The handle to ACR120 reader returned by ACR120_Open	
stationID	The StationID of ACR120 Reader.	
block	The block number where you want to write.	
ValueData	The value you want to write.	
Return Value	INT16	Result code. 0 means success.

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.

Notes:

Memory Organization is based on Standard Card IC MF1 IC S50, which are 16 sectors with 4 blocks of 16 bytes each.

Sector	Block	Byte Number within a Block															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
15	3	Key A				Access Bits				Key B							
	2																
	1																
	0																

14	3												
	2												
	1												
	0												
:	:												
:	:												
:	:												
1													
	2												
	1												
	0												
0	3												
	2												
	1												
	0												

For you to access the exact block, you have to multiply the sector number by 4 plus the block number :
 $\text{Block} = (\text{Sector} * 4) + \text{BlockNumber}$.

Example:

// write value to block 1 of sector 1 (reader stationID: 1)

// Let's assume logging into sector 1 was successful

```
INT16 RetCode;
```

```
UINT8 SID;
```

```
UINT8 block;
```

```
UINT32 ValueData;
```

```
CString StrMsg;
```

```
SID = 1;
```

```
block = (1 * 4) + 1;
```

```
ValueData = 5000;
```

```
RetCode = ACR120_WriteValue(rHandle, SID, block, ValueData);
```

2.3.1.12 ACR120_WRITEEEPROM

Format:

```
DLLAPI INT16 AC_DECL ACR120_WriteEEPROM ( INT16  rHandle,
                                           UINT8  stationID,
                                           UINT8  reg,
                                           UINT8  EEPROMData);
```

Function Description:

This function writes to internal EEPROM of the ACR120 reader.

Parameters	Description	
rHandle	The handle to ACR120 reader returned by ACR120_Open	
stationID	The StationID of ACR120 Reader.	
reg	The register number.	
EEPROMData	The value to write at the ACR120 reader EEPROM reg.	
Return Value	INT16	Result code. 0 means success.

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.

Notes:

The details for the register map is shown below:

ACR120 Reader Module EEPROM Memory Organization		
Register Number	Name	Description
00h...03h	Unique device ID (32bit)	This number is unique for each device and therefore read only.
04h	Station ID	Indicates the address ID for every station. The ID is used for addressing within a party line.
05h	Protocol Configuration	Set Protocol type, power on behavior. 00h -> ACR120 reader in ASCII mode 01h -> ACR120 reader in Binary mode
06h	Baud Rate Selection	Defines Communication speed. 00h -> 9600 baud 01h -> 19200 baud 02h -> 38400 baud 03h -> 57600 baud
07h...0Fh	Reserved	
10h...13h	User Date	Free Usage

Example:

```
// Write/Set Baud rate to 57600, (register 06h) of EEPROM (reader stationID: 1)

INT16 RetCode;

UINT8 SID;
UINT8 reg;
UINT8 EEPROMData;
CString StrMsg;

SID = 1;
reg = 6;
EEPROMData = 3;

RetCode = ACR120_WriteEEPROM (rHandle, SID, reg, EEPROMData);
```

2.3.1.13 ACR120_WRITELOWLEVELREGISTER**Format:**

```
ACR120_DLLAPI INT16 ACR120_DECLACR120_WriteLowLevelRegister(
    INT16      hReader,
    UINT8      stationID,
    UINT8      reg,
    UINT8      registerData);
```

Function Description:

This function writes the internal register.

Parameters	Description	
hReader	The handle to our reader returned by ACR120_Open.	
stationID	The station ID of our reader.	
reg	The register number.	
registerData	Contains the register's value to write.	
Return Value	INT16	Result code. 0 means success.

* This command should be used under manufacturer's recommendation.

2.3.1.14 ACR120_WRITEMASTERKEY**Format:**

```
DLLAPI INT16 AC_DECL ACR120_WriteMasterKey ( INT16      rHandle,
    UINT8      stationID,
    UINT8      keyNo,
    UINT8      pKey[ACR120_KEY_LEN]);
```

Function Description:

This function writes Master key to internal EEPROM of the ACR120 reader.

Parameters	Description	
rHandle	The handle to ACR120 reader returned by ACR120_Open	
stationID	The StationID of ACR120 Reader.	
keyNo	The master key number.	
pKey	6 bytes key to write.	
Return Value	INT16	Result code. 0 means success.

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.

Notes:

ACR120 reader currently can store up to 32 keys (0 - 31). Keys stored in the reader can be used to Login to a card sector by using the KeyType ACR120_LOGIN_KEYTYPE_STORED_A or ACR120_LOGIN_KEYTYPE_STORED_B.

Example:

// Write master key: AAh AAh AAh AAh AAh AAh ; keyNO:2 (reader stationID: 1)

```
INT16 RetCode;
```

```
UINT8 SID;
UINT8 keyNo;
UINT8 pKey(5);
CString StrMsg;
SID = 1;
keyNo = 2;
```

```
pKey[0]=170;
pKey[1]=170;
pKey[2]=170;
pKey[3]=170;
pKey[4]=170;
pKey[5]=170;
```

```
RetCode = ACR120_WriteMasterKey (rHandle, SID, keyNo, pKey);
```

2.3.1.15 ACR120_INC**Format:**

```
DLLAPI INT16 AC_DECL ACR120_Inc (INT16 rHandle,
                                UINT8 stationID,
                                UINT8 block,
                                INT32 value,
                                INT32* pNewValue);
```

Function Description:

This function Increments a value block by adding a value to previously stored value.

Parameters	Description	
rHandle	The handle to ACR120 reader returned by ACR120_Open	
stationID	The StationID of ACR120 Reader.	
block	Value Block Number.	
value	Value to be added to previously stored value in the block.	
pNewValue	The updated value after increment.	
Return Value	INT16	Result code. 0 means success.

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.

Notes:

Memory Organization is based on Standard Card IC MF1 IC S50, which are 16 sectors with 4 blocks of 16 bytes each.

		Byte Number within a Block															
Sector	Block	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
15	3	Key A				Access Bits				Key B							
	2																
	1																
	0																
14	3	Key A				Access Bits				Key B							
	2																
	1																
	0																
:	:																
:	:																
:	:																
1	3	Key A				Access Bits				Key B							
	2																
	1																
	0																
0	3	Key A				Access Bits				Key B							
	2																
	1																



Block must contain a Value before Incrementing.
"Refer to ACR120_WriteValue".

Example:

// Increment value block 1 of sector 1 by 500. (reader stationID: 1)

```
INT16 RetCode;

UINT8 SID;
UINT8 block;
UINT8 value;
UINT8 pNewValue;
CString StrMsg;

SID = 1;
Block = ( 1 * 4 ) + 1;
value = 500;

RetCode = ACR120_Inc (rHandle, SID, block, value, &pNewValue);

// Updated Value after increment
StrMsg.Format("Incremented Value: %d",pNewValue);
```

2.3.1.16 ACR120_DEC

Format:

```
DLLAPI INT16 AC_DECL ACR120_Dec (INT16  rHandle,
                                UINT8   stationID,
                                UINT8   block,
                                INT32    value,
                                INT32*   pNewValue);
```

Function Description:

This function decrements a value block by subtracting a value to previously stored value.

Parameters	Description	
rHandle	The handle to ACR120 reader returned by ACR120_Open	
stationID	The StationID of ACR120 Reader.	
block	Value Block Number.	
value	Value to be subtracted to previously stored value in the block.	
pNewValue	The updated value after decrement.	
Return Value	INT16	Result code. 0 means success.

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.

Notes:

Memory Organization is based on Standard Card IC MF1 IC S50 which are 16 sectors with 4 blocks of 16 bytes each.

Sector	Block	Byte Number within a Block															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
15	3	Key A						Access Bits						Key B			
	2																
	1																
	0																
14	3	Key A						Access Bits						Key B			
	2																
	1																
	0																
:	:																
:	:																
:	:																
1	3	Key A						Access Bits						Key B			
	2																
	1																
	0																
0	3	Key A						Access Bits						Key B			
	2																
	1																
	0																

Block must contain a Value before decrementing.

"Refer to ACR120_WriteValue".

Example:

// decrement value block 1 of sector 1 by 500. (reader stationID: 1)

```
INT16 RetCode;
```

```
UINT8 SID;
```

```
UINT8 block;
```

```
UINT8 value;
```

```
UINT8 pNewValue;
```

```
CString StrMsg;
```

```

SID = 1;
Block = ( 1 * 4 ) + 1;
value = 500;

RetCode = ACR120_dec (rHandle, SID, block, value, &pNewValue);

// Updated Value after decrement
StrMsg.Format("Decrement Value: %d",pNewValue);

```

2.3.1.17 ACR120_COPY

Format:

```

DLLAPI INT16 AC_DECL ACR120_Copy (INT16  rHandle,
                                   UINT8   stationID,
                                   UINT8   srcBlock,
                                   UINT8   desBlock,
                                   INT32*  pNewValue);

```

Function Description:

This function copies a value block to another block of the same sector.

Parameters	Description	
rHandle	The handle to ACR120 reader returned by ACR120_Open	
stationID	The StationID of ACR120 Reader.	
srcBlock	The source block number.	
desBlock	The target block number.	
pNewValue	The updated value after copy.	
Return Value	INT16	Result code. 0 means success.

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.

Notes:

Memory Organization is based on Standard Card IC MF1 IC S50, which are 16 sectors with 4 blocks of 16 bytes each.

		Byte Number within a Block															
Sector	Block	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
15	3	Key A				Access Bits				Key B							
	2																
	1																
	0																
14	3	Key A				Access Bits				Key B							

[illegible]

Source block must contain a Value before copying to another block in the same sector. “Refer to *ACR120_WriteValue*”.

The destination or target block need not to be a value block.

Example:

```
// copy value block 1 of sector 1 to block 2 of sector 1. (reader stationID: 1)
// Lets assume that logging into sector 1 was successful and block one is a value
// block. "Refer to ACR120_WriteValue".
```

```
INT16 RetCode;

UINT8 SID;
UINT8 srcBlock;
UINT8 desBlock;
UINT8 pNewValue;
CString StrMsg;

SID = 1;
srcBlock = ( 1 * 4 ) + 1;
desBlock= ( 1 * 4 ) + 2;

RetCode = ACR120_Copy (rHandle, SID, srcBlock, desBlock, &pNewValue);

// Updated Value of target block after copy.
StrMsg.Format("Block 2 Value: %d",pNewValue);
```

2.3.1.18 ACR120_POWER

Format:

```
DLLAPI INT16 AC_DECL ACR120_Power (INT16 rHandle,  
                                   UINT8  stationID,  
                                   BOOL    bOn);
```

Function Description:

This function is used to turn the antenna power on/off for reducing power consumption.

Parameters	Description	
rHandle	The handle to ACR120 reader returned by ACR120_Open	
stationID	The StationID of ACR120 Reader.	
bOn	Turn on (TRUE) or off (FALSE)..	
Return Value	INT16	Result code. 0 means success.

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.

Notes:

The antenna power will be turned on automatically before TAG access commands like "ACR120_Select" and "ACR120_MultiTagSelect".

Example:

// Turns antenna power off (reader stationID: 1)

```
INT16 RetCode;  
  
UINT8 SID;  
BOOL bOn;  
  
SID = 1;  
bOn = false;  
RetCode = ACR120_Power (rHandle, SID,bOn);
```

// Turns antenna power on (reader stationID: 1)

```
INT16 RetCode;  
  
UINT8 SID;  
BOOL bOn;  
  
SID = 1;  
bOn = true;  
  
RetCode = ACR120_Power (rHandle, SID,bOn);
```

2.3.1.19 ACR120_READUSERPORT

Format:

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

Function Description:

This function is used to read in the state of user port (PIN 14 of the OEM module).

Parameters	Description	
rHandle	The handle to ACR120 reader returned by ACR120_Open	
stationID	The StationID of ACR120 Reader.	
pUserPortState	Contains the port state (only LSB is used).	
Return Value	INT16	Result code. 0 means success.

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.

Example:

```
// Read User port (reader stationID: 1)
```

```
INT16 RetCode;
```

```
UINT8 SID;
UINT8 pUserPortState;
```

```
SID = 1;
```

```
RetCode = ACR120_ReadUserPort (rHandle, SID, &pUserPortState);
```

2.3.1.20 ACR120_WRITEUSERPORT

Format:

```
DLLAPI INT16 AC_DECL ACR120_WriteUserPort (INT16 rHandle,
                                           UINT8 stationID,
                                           UINT8 userPortState);
```

Function Description:

This function sets the state of user port (PIN 14 of the OEM module).

Parameters	Description
rHandle	The handle to ACR120 reader returned by ACR120_Open
stationID	The StationID of ACR120 Reader.

Parameters	Description	
userPortState	Clear the port pin if userPortState = 0. Otherwise it's set.	
Return Value	INT16	Result code. 0 means success.

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.

Example:

```
// Clear User port (reader stationID: 1)
```

```
INT16 RetCode;

UINT8 SID;
UINT8 userPortState;

SID = 1;
userPortState = 0;

RetCode = ACR120_WriteUserPort (rHandle, SID, userPortState);
```

2.3.1.21 ACR120_GETID**Format:**

```
DLLAPI INT16 AC_DECL ACR120_GetID (INT16 rHandle,
                                   UINT8* pNumID,
                                   UINT8* pStationID);
```

Function Description:

This function gets the station ID's for all reader modules on the bus.

Parameters	Description	
rHandle	The handle to ACR120 reader returned by ACR120_Open	
pNumID	The number of station ID returned.	
pStationID	Contains the list of station ID returned.	
Return Value	INT16	Result code. 0 means success.

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.

Example:

```
// Get station ID's
```

```
INT16 RetCode;
```

```
UINT8 pNumID;
```

```
UINT8 pStationID[255];
```

```
RetCode = ACR120_GetID(rHandle, &pNumID, pStationID);
```

2.3.1.22 ACR120_LISTTAG**Format:**

```
DLLAPI INT16 AC_DECL ACR120_ListTag( INT16    rHandle,
                                     UINT8     stationID,
                                     UINT8*    pNumTagFound,
                                     BOOL*     pHaveTag,
                                     UINT8*    pTAG,
                                     UINT8*    pSN);
```

Function Description:

This function lists the serial numbers of all tags, which are in readable antenna range.

Parameters	Description	
rHandle	The handle to ACR120 reader returned by ACR120_Open	
stationID	The StationID of ACR120 Reader.	
pNumTagFound	Contains of number of TAG listed.	
pHaveTag	Whether the TAG Type Identification is listed.	
pTAG	The list of TAG Type Identification. If pHaveTag is false, this is an array of serial number length of the cards detected. If pHaveTag is true, this is an array of Tag type. The corresponding serial number length could then be determined from the Tag type.	
pSN	The flat array of serial numbers. All serial numbers are concatenated with length of either 4, 7 or 10 numbers. The lengths are indicated in pTag field.	
Return Value	INT16	Result code. 0 means success.

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.

Example:

```
// List all Tag's in antenna range (stationID: 1)
```

```
INT16 RetCode;

UINT8  SID;
UINT8* pNumTagFound;
BOOL*  pHaveTag;
UINT8* pTAG;
UINT8* pSN[199];
UINT8  ctr;
UINT8  ctrl;

SID=1;

RetCode = ACR120_ListTag(rHandle, SID, &pNumTagFound, &pHaveTag, &pTAG, pSN);

StrMsg.Format("Number of Tag Found: %d", pNumTagFound);

//Display Serial Numbers Found
// Loop to Number of TagFound (pNUmTagFound)

ctrl = 0;
for( ctr = 0 ; ctr < pNumTagFound; ctr++)
{

StrMsg.Format("SN[%d]: %X %X %X %X", ctr,
SN[ctrl+0],SN[ctrl+1],SN[ctrl+2],SN[ctrl+3]);
ctrl += 4;

}
```

2.3.1.23 ACR120_MULTITAGSELECT**Format:**

```
DLLAPI INT16 AC_DECL ACR120_MultiTagSelect( INT16      rHandle,
                                           UINT8  stationID,
                                           UINT8  pSN[ACR120_SN_LEN],
                                           BOOL*  pHaveTag,
                                           UINT8*   pTAG,
                                           UINT8  pResultSN[ACR120_SN_LEN]);
```

Function Description:

This function selects a single card in range and returns the card ID (Serial Number).

Parameters	Description
rHandle	The handle to ACR120 reader returned by ACR120_Open
stationID	The StationID of ACR120 Reader.

Parameters	Description	
pSN	Contains the serial number of the TAG to be selected. It's ACR120_SN_LEN is 4 bytes long. AC_MIFARE_SN_LEN_4 (4 bytes long), AC_MIFARE_SN_LEN_7 (7 bytes long), AC_MIFARE_SN_LEN (10 bytes long).	
pHaveTag	Whether the TAG Type Identification of selected tag is returned.	
pTAG	The TAG Type Identification of selected tag.	
pResultSN	The serial number of selected TAG. It's ACR120_SN_LEN is 4 bytes long. AC_MIFARE_SN_LEN_4 (4 bytes long), AC_MIFARE_SN_LEN_7 (7 bytes long), AC_MIFARE_SN_LEN (10 bytes long).	
Return Value	INT16	Result code. 0 means success.

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.

Example:

```
// Select a card in range (reader stationID: 1)
// Let's assume that there were 2 cards in range and you wanted to select the one
// with serial number ( FFh FFh FFh FFh )

INT16 RetCode;

UINT8 SID;
UINT8 pSN[3];
BOOL* pHaveTag;
UINT8* pTAG;
UINT8 pResultSN[3];

SID = 1;

pSN[0]=FF;
pSN[1]=FF;
pSN[2]=FF;
pSN[3]=FF;

RetCode = ACR120_MultiTagSelect(rHandle, SID, pSN, &pHaveTag, &pTAG, pResultSN);

// Get Serial Number Returned
StrMsg.Format("Card Serial Selected: %X %X %X %X",
              pResultSN[0], pResultSN[1],
              pResultSN [2], pResultSN [3] );
```

2.3.1.24 ACR120_TXDATATELEGRAM

Format:

```
ACR120_DLLAPI INT16 ACR120_DECL
ACR120_TxDataTelegram(
    INT16      hReader,
    UINT8      stationID,
    UINT8      length,
    BOOL       bParity,
    BOOL       bEvenParity,
    BOOL       bCRCCGen,
    BOOL       bCRCCheck,
    BOOL       bCryptoInactive,
    UINT8      bitFrame,
    UINT8*     data,
    UINT8*     pRecvLen,
    UINT8*     recvData);
```

Function Description:

This function transfers user specific data frames.

Parameters	Description	
hReader	The handle to our reader returned by ACR120_Open.	
stationID	The station ID of our reader.	
length	The length of user specific data frame.	
bParity	TRUE if parity generation is enabled.	
bEvenParity	TRUE if parity is even. Otherwise it's odd.	
bCRCCGen	TRUE if CRC generation for transmission is enabled.	
bCRCCheck	TRUE if CRC checking for receiving is enabled.	
bCryptoInactive	TRUE if Crypto unit is deactivated before transmission start.	
bitFrame	Bit Framing (number of bits from last byte transmitted).	
data	Contains the user specific data frame.	
pRecvLen	It returns the length of response data received.	
recvData	Contains the response data received.	
Return Value	INT16	Result code. 0 means success.

2.3.1.25 ACR120_REQUESTVERSIONINFO

Format:

```
ACR120_DLLAPI INT16 ACR120_DECL
ACR120_RequestVersionInfo(
    INT16      hReader,
    UINT8      stationID,
```

```
UINT8*    pVersionInfoLen,  
UINT8*    pVersionInfo);
```

Function Description:

This function gets the reader's firmware version information

Parameters	Description	
hReader	The handle to our reader returned by ACR120_Open.	
pNumID	The number of station ID returned.	
pVersionInfoLen	It returns the length of the Firmware Version string.	
PVersionInfo	It returns the Firmware Version string.	
Return Value	INT16	Result code. 0 means success.

Appendix A: Table of Error Codes

Code	Meaning
ERR_ACR120_INTERNAL_UNEXPECTED(1000)	Library internal unexpected error
ERR_ACR120_PORT_INVALID(2000)	The port is invalid
ERR_ACR120_PORT_OCCUPIED(2010)	The port is occupied by another application
ERR_ACR120_HANDLE_INVALID(2020)	The handle is invalid
ERR_ACR120_INCORRECT_PARAM(2030)	Incorrect Parameter
ERR_ACR120_READER_NO_TAG(3000)	No TAG in reachable range / selected
ERR_ACR120_READER_READ_FAIL_AFTER_OP(3010)	Read fail after operation
ERR_ACR120_READER_NO_VALUE_BLOCK(3020)	Block doesn't contain value
ERR_ACR120_READER_OP_FAILURE(3030)	Operation failed
ERR_ACR120_READER_UNKNOWN(3040)	Reader unknown error
ERR_ACR120_READER_LOGIN_INVALID_STORED_KEY_FORMAT(4010)	Invalid stored key format in login process
ERR_ACR120_READER_WRITE_READ_AFTER_WRITE_ERROR(4020)	Reader can't read after write operation
ERR_ACR120_READER_DEC_FAILURE_EMPTY(4030)	Decrement failure (empty)

Appendix B: Sector Number Adaptation on Mifare 4K Card

Sector Number on Card	Sector Number for Log-in	Block Number	Card Type	
0x00	0x00	0x00-0x03	Mifare 1K	Mifare 4K
0x01	0x01	0x04-0x07	Standard sectors	Standard sectors
0x02	0x02	0x08-0x0B		
0x03	0x03	0x0C-0x0F		
0x04	0x04	0x10-0x13		
0x05	0x05	0x14-0x17		
0x06	0x06	0x18-0x1B		
0x07	0x07	0x1C-0x1F		
0x08	0x08	0x20-0x23		
0x09	0x09	0x24-0x27		
0x0A	0x0A	0x28-0x2B		
0x0B	0x0B	0x2C-0x2F		
0x0C	0x0C	0x30-0x33		
0x0D	0x0D	0x34-0x37		
0x0E	0x0E	0x38-0x3B		
0x0F	0x0F	0x3C-0x3F		
0x10	0x10	0x40-0x43		Big sectors
0x11	0x11	0x44-0x47		
0x12	0x12	0x48-0x4B		
0x13	0x13	0x4C-0x4F		
0x14	0x14	0x50-0x53		
0x15	0x15	0x54-0x57		
0x16	0x16	0x58-0x5B		
0x17	0x17	0x5C-0x5F		
0x18	0x18	0x60-0x63		
0x19	0x19	0x64-0x67		
0x1A	0x1A	0x68-0x6B		
0x1B	0x1B	0x6C-0x6F		
0x1C	0x1C	0x70-0x73		
0x1D	0x1D	0x74-0x77		
0x1E	0x1E	0x78-0x7B		
0x1F	0x1F	0x7C-0x7F		
0x20	0x20	0x80-0x8F		Big sectors
0x21	0x24	0x90-0x9F		
0x22	0x28	0xA0-0xAF		
0x23	0x2C	0xB0-0xBF		
0x24	0x30	0xC0-0xCF		
0x25	0x34	0xD0-0xDF		
0x26	0x38	0xE0-0xEF		
0x27	0x3C	0xF0-0xFF		