Advanced Card Systems Ltd.
Card & Reader Technologies

ACR1281S1-C8 Contactless Reader/Writer

Application Programming Interface V1.00

Subject to change without prior notice

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

Table of Contents

List of Figures

List of Tables

1.0. Introduction

The ACR1281S-C8 is the new version of ACS's ACR120S Contactless Smart Card Reader. This manual describes the use of ACR1281S1-C8 interface software to program the ACR1281S1-C8 readers. It contains a set of library functions implemented for the application programmers to operate the ACR1281S1-C8 readers and the presented cards. The library functions are supplied in the form of DLL and it can be programmed using the popular development tools like Visual C/C++, Visual Basic, Delphi, etc. ACR1281S1-C8 readers can be connected to the PC via the RS/232 interface.

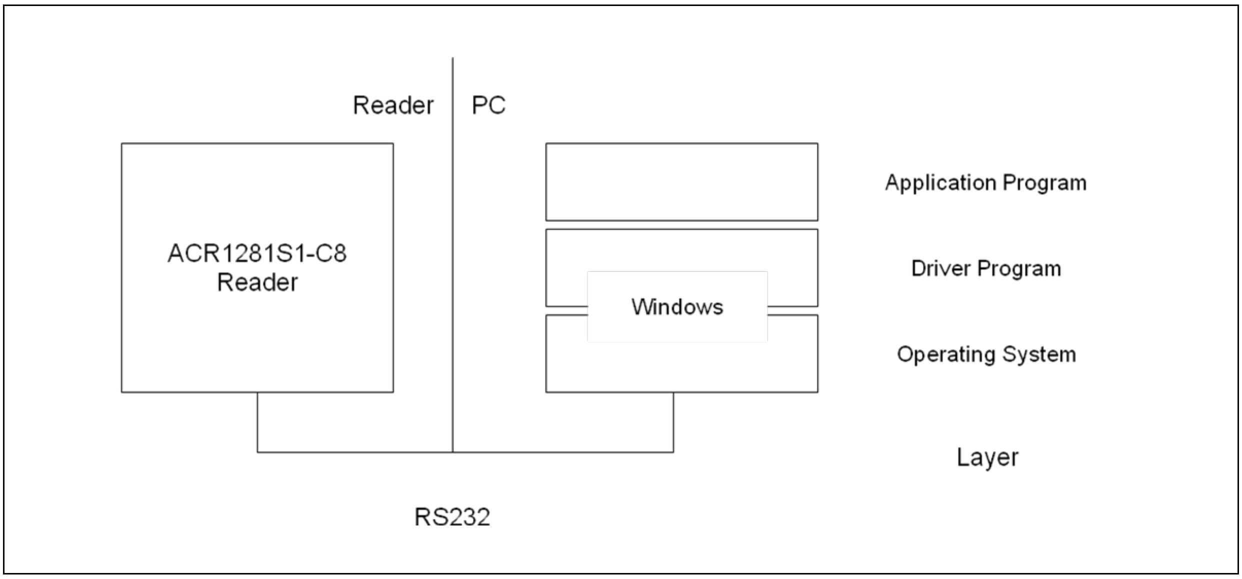


Figure 1: Architecture of the ACR1281S1-C8 Library

1.1. Features

- Serial RS232 interface (also available in RS485 *upon request*)
- 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
- OEM PCBA module version (*upon request*)
- Firmware Upgradability
- Compliant with the following standards:
  - CE
  - FCC
  - RoHS

2.0. Overview

ACR1281S1-C8 contains a set of high-level functions for the application software's use. It provides a consistent application programming interface (ACR120 API) to operate on the ACR1281S1-C8 reader and the corresponding presented card. The PC communicates with the ACR1281S1-C8 reader via the communication port facilities provided by the operating system.

2.1. Communication Speed

The library controls the communication speed between the reader and the PC. The default communication baud rate (factory setting) is 9600 bps, no parity, 8 bits and one stop bits. A higher speed of 115200 bps can be achieved by using software command issued 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 determine the reader settings.

2.2. Application Programming Interface

The Application Programming Interface (API) defines a common way of accessing the ACR1281S1-C8 reader. Application programs invoke ACR1281S1-C8 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.

Interface Function Types

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*.

Note: *All Card API's involving* ***SECTOR*** *and* ***BLOCK*** *parameters, please refer to* ***Appendix C*** *for further discussion.*

2.2.1. ACR120_Open

Format:

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

Function Description:

This function opens the port (connection) to ACR1281S1-C8 reader.

<table>
<tr><th>Parameters</th><th colspan="2">Description</th></tr>
<tr><td>ReaderPort</td><td colspan="2">The port number where the ACR1281S1-C8 reader is connected.
Available choices are “ACR120_COM1” to “ACR120_COM8.”</td></tr>
<tr><td>BaudRate</td><td colspan="2">The port baud rate.
Available choices are “ACR120_COM_BAUDRATE_9600” to “ACR120_COM_BAUDRATE_115200.”</td></tr>
<tr><td><b>Return Value</b></td><td>INT16</td><td>Result code: 0 means success</td></tr>
</table>

Table 1: ACR120_Open Function 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 - Table of Error Codes**.

Example:

```
// Open a port to an ACR1281S1-C8 reader connected at COM1 with a baud rate
of 9600 bps


INT16 rHandle;

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

2.2.2. ACR120_Close

Format:

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

Function Description:

This function closes the port (connection) to ACR1281S1-C8 reader.

Parameters	Description	
rHandle	The handle to ACR1281S1-C8 reader returned by ACR120_Open	
Return Value	INT16	Result code: 0 means success.

Table 2: ACR120_Close Function 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 - Table of Error Codes**.

Example:

```
// Close the port (connection) to ACR1281S1-C8 reader.


INT16 RetCode;

RetCode = ACR120_Close (rHandle);
```

2.2.3. 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 ACR1281S1-C8 reader returned by ACR120_Open
stationID	The Station ID of ACR1281S1-C8 reader

Parameters	Description	
Return Value	INT16	Result code: 0 means success

Table 3: ACR120_Reset Function 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 - Table of Error Codes**.

Example:

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


INT16 RetCode;

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

2.2.4. 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 ACR1281S1-C8 reader returned by ACR120_Open	
stationID	The Station ID of ACR1281S1-C8 reader	
pHaveTag	Output Variable that will indicate whether the TAG Type Identification is returned: (TRUE) or (FALSE)	
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

Table 4: ACR120_Select Function Description

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

Table 5: TAG Type Identification

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 - Table of Error Codes**.

Notes:

1. *You have to select the card first before you can Login and manipulate the card.*
2. *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]);
```

2.2.5. 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 ACR1281S1-C8 reader returned by ACR120_Open	
stationID	The Station ID of ACR1281S1-C8 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.

Table 6: ACR120_Login Function Description

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

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 – Table of Error Codes**.

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: 0xFFh 0xFFh 0xFFh 0xFFh 0xFFh 0xFFh will be used.

If keyType = ACR120_LOGIN_KEYTYPE_STORED_A, or

If keyType = ACR120_LOGIN_KEYTYPE_STORED_B,

> *Then pKey will not bed used and can be just 0's while storedNo is the keyNo of the MasterKey you want to use. Please refer to* ***2.2.14 - 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);
```

2.2.6. 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 ACR1281S1-C8 reader returned by ACR120_Open	
stationID	The Station ID of ACR1281S1-C8 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

Table 7: ACR120_Read Function 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 - Table of Error Codes**.

Note: *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  %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  %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]);
```

2.2.7. 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 ACR1281S1-C8 reader returned by ACR120_Open	
stationID	The Station ID of ACR1281S1-C8 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.

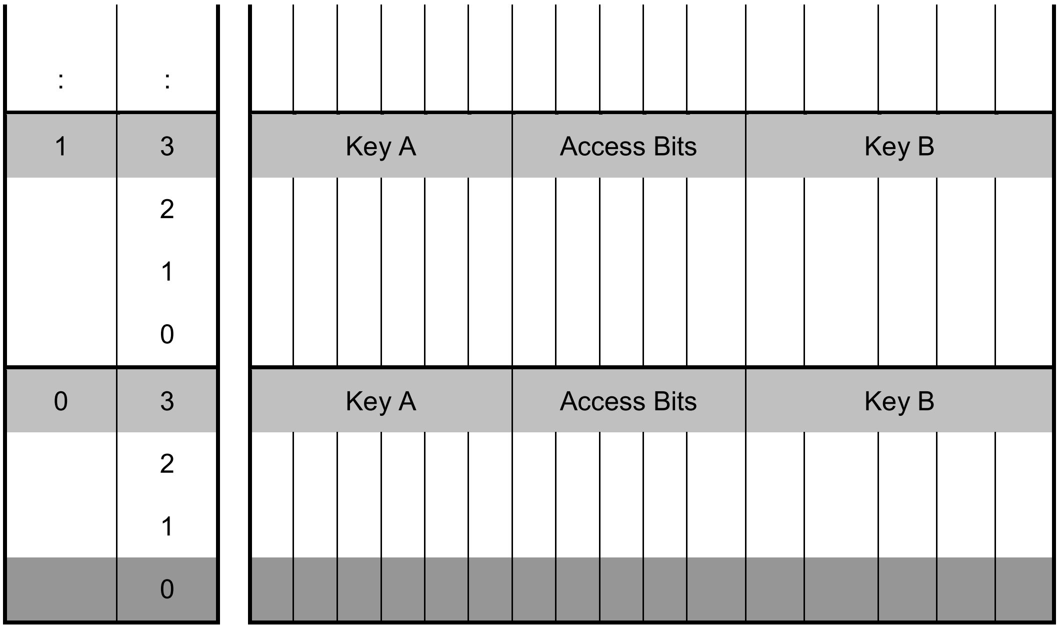
Table 8: ACR120_ReadValue Function Description

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 - Table of Error Codes**.

Note: *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 ACR20_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. Please refer to **Section 2.2.11 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);
```

2.2.8. 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 ACR1281S1-C8 reader.

Parameters	Description	
rHandle	The handle to ACR1281S1-C8 reader returned by ACR120_Open	
stationID	The Station ID of ACR1281S1-C8 reader	
reg	The register number	
pEEPROMData	Output Variable that will contain the EEPROM register’s value	
Return Value	INT16	Result code. 0 means success.

Table 9: ACR120_ReadEEPROM Function 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 - Table of Error Codes**.

The details for the register map are shown below:

ACR1281S1-C8 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 = ACR1281S1-C8 reader in ASCII mode 01h = ACR1281S1-C8 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);
```

2.2.9. ACR120_ReadLowLevelRegister

Format:

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

Note: *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 the reader	
reg	The register number	
pRegData	Contains the register's value	
Return Value	INT16	Result code. 0 means success.

Table 10: ACR120_ReadLowLevelRegister Function Description

2.2.10. 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 ACR1281S1-C8 reader returned by ACR120_Open	
stationID	The Station ID of ACR1281S1-C8 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.

Table 11: ACR120_Write Function 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 - Table of Error Codes**.

Note: *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																
	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:

```
// 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);
```

2.2.11. 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 ACR1281S1-C8 reader returned by ACR120_Open	
stationID	The Station ID of ACR1281S1-C8 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.

Table 12: ACR120_WriteValue Function 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 - Table of Error Codes**.

Note: *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																
	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:

```
// 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);
```

2.2.12. 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 ACR1281S1-C8 reader.

Parameters	Description	
rHandle	The handle to ACR1281S1-C8 reader returned by ACR120_Open	
stationID	The Station ID of ACR1281S1-C8 reader	
reg	The register number	
EEPROMData	The value to write at the ACR1281S1-C8 reader EEPROM reg	
Return Value	INT16	Result code. 0 means success.

Table 13: ACR120_WriteEEPROM Function 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 - Table of Error Codes**.

The details for the register map are shown below:

ACR1281S1-C8 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 = ACR1281S1-C8 reader in ASCII mode 01h = ACR1281S1-C8 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);
```

2.2.13. 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 ACR1281S1-C8 returned by ACR120_Open	
stationID	The Station ID of ACR1281S1-C8 reader	
reg	The register number	
registerData	Contains the register's value to write	
Return Value	INT16	Result code. 0 means success.

Table 14: ACR120_WriteLowLevelRegister Function Description

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

2.2.14. 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 ACR1281S1-C8 reader.

Parameters	Description	
rHandle	The handle to ACR1281S1-C8 reader returned by ACR120_Open	
stationID	The Station ID of ACR1281S1-C8 reader	
keyNo	The master key number	
pKey	6 bytes key to write	
Return Value	INT16	Result code. 0 means success.

Table 15: ACR120_WriteMasterKey Function 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 - Table of Error Codes**.

Note: ACR1281S1-C8 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);
```

2.2.15. 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 ACR1281S1-C8 reader returned by ACR120_Open	
stationID	The Station ID of ACR1281S1-C8 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.

Table 16: ACR120_Inc Function 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 - Table of Error Codes**.

Note: *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																
	0																

*Block must contain a Value before Incrementing. Please refer to **Section 2.2.11 – 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);
```

2.2.16. 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 ACR1281S1-C8 reader returned by ACR120_Open	
stationID	The Station ID of ACR1281S1-C8 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.

Table 17: ACR120_Dec Function 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 - Table of Error Codes**.

Note: *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																
	0																

Block must contain a Value before Incrementing. Please refer to ***2.2.11 – 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("Decremented Value: %d",pNewValue);
```

2.2.17. 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 ACR1281S1-C8 reader returned by ACR120_Open	
stationID	The Station ID of ACR1281S1-C8 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.

Table 18: ACR120_Copy Function 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 - Table of Error Codes**.

Note: *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																
	0																

*Source block must contain a Value before copying to another block in the same sector. Please refer to **2.2.11 – 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);
```

2.2.18. 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 ACR1281S1-C8 reader returned by ACR120_Open	
stationID	The Station ID of ACR1281S1-C8 reader	
bOn	Turn on (TRUE) or off (FALSE)	
Return Value	INT16	Result code. 0 means success.

Table 19: ACR120_Power Function 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 - Table of Error Codes**.

Note: *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);
```

2.2.19. 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 ACR1281S1-C8 reader returned by ACR120_Open	
stationID	The Station ID of ACR1281S1-C8 reader	
pUserPortState	Contains the port state (only LSB is used)	
Return Value	INT16	Result code. 0 means success.

Table 20: ACR120_ReadUserPort Function 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 - Table of Error Codes**.

Example:

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


INT16 RetCode;

UINT8 SID;
UINT8 pUserPortState;
```

```
SID = 1;

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

2.2.20. ACR120_WriteUserPort

Format:

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

Function Description:

For ACR1281S1-C8, this function sets the state of the LED.

For ACM1281S1-Z8, a relay is tied to the LED control. An additional control is made available for controlling the on board buzzer. This function sets the states of Relay (together with LED) and Buzzer.

Note: *The LED state of some readers may have been tied to indicate operation status by software option in factory default. In this case, the user may not be able to change the Relay/LED independently. To release this tie, please use the ACR120_WRITEEEPROM function to write a value of 0x00h to a special EEPROM address of 0xFE then do a power reset to the reader. Doing this operation only once is enough to change the option permanently.*

Parameters	Description	
rHandle	The handle to ACR1281S1-C8 reader returned by ACR120_Open	
stationID	The Station ID of ACR1281S1-C8 reader	
userPortState	**Value**	**Action**
	0x00h	Relay/LED and Buzzer OFF
	0x01h	Relay/LED ON, Buzzer OFF
	0x02h	Relay/LED OFF Buzzer ON
	0x03h	Relay/LED and Buzzer ON
Return Value	INT16	Result code. 0 means success.

Table 21: ACR120_WriteUserPort Function 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 - Table of Error Codes**.

Example:

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


INT16 RetCode;

UINT8 SID;
UINT8 userPortState;
```

```
SID = 1;
userPortState = 0;

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

2.2.21. 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 ACR1281S1-C8 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.

Table 22: ACR120_GetID Function 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 - Table of Error Codes**.

Example:

```
// Get station ID's


INT16 RetCode;

UINT8 pNumID;
UINT8 pStationID[255];


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

2.2.22. 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 ACR1281S1-C8 reader returned by ACR120_Open	
stationID	The Station ID of ACR1281S1-C8 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 4, 7 or 10 numbers. The lengths are indicated in *pTag* field.	
Return Value	INT16	Result code. 0 means success.

Table 23:ACR120_ListTag Function 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 - Table of Error Codes**.

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 ctr1;

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;

}
```

2.2.23. 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).

<table>
<tr><th>Parameters</th><th colspan="2">Description</th></tr>
<tr><td>rHandle</td><td colspan="2">The handle to ACR1281S1-C8 reader returned by ACR120_Open</td></tr>
<tr><td>stationID</td><td colspan="2">The Station ID of ACR1281S1-C8 reader</td></tr>
<tr><td>pSN</td><td colspan="2">Contains the serial number of the TAG to be selected. Its 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).</td></tr>
<tr><td>pHaveTag</td><td colspan="2">Whether the TAG Type Identification of selected tag is returned.</td></tr>
<tr><td>pTAG</td><td colspan="2">The TAG Type Identification of selected tag.</td></tr>
<tr><td>pResultSN</td><td colspan="2">The serial number of selected TAG. Its 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).</td></tr>
<tr><td><b>Return Value</b></td><td>INT16</td><td>Result code. 0 means success.</td></tr>
</table>

Table 24: ACR120_MultiTagSelect Function 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 - Table of Error Codes**.

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] );
```

2.2.24. ACR120_TxDataTelegram

Format:

```
ACR120_DLLAPI INT16 ACR120_DECL
ACR120_TxDataTelegram(
   INT16    hReader,
   UINT8    stationID,
   UINT8    length,
   BOOL     bParity,
   BOOL     bOddParity,
   BOOL     bCRCGen,
   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 ACR1281S1-C8 reader
length	The length of user specific data frame

Parameters	Description	
bParity	TRUE if parity generation is enabled	
bOddParity	TRUE if parity is odd. Otherwise it's even	
bCRCGen	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.

Table 25: ACR120_TxData Telegram Function Description

2.2.25. 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 ACR1281S1-C8 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.

Table 26: ACR120_RequestVersionInfo Function Description

2.2.26. PICC_InitBlockNumber

Format:

```
DLLAPI INT16 AC_DECL PICC_InitBlockNumber (INT16 FrameSizeIndex);
```

Function Description:

This function resets the block number to be used during the ISO 14443 Part 4 (T=CL) communication. It also sets the frame length of the Card (PICC). By default the frame length is 16 bytes. The frame length of the card is reported by the ATS in Type A and the ATQB in Type B cards.

Parameters	Description	
Frame Size Index	An index to a maximum frame size which the card can accept	
Return Value	INT16	The actual frame length selected.

Table 27: PICC_InitBlockNumber Function Description

The argument only accepts the following:

Frame Size Index	Frame Length (in bytes)
0	16
1	24
2	32
3	40
4	48
5	64
6	96
7	128
8	256
Otherwise	16

Returns:

The actual frame length selected will be returned as a confirmation. For example, if 4 is used as calling parameter, the value 48 is returned.

Notes:

1. *This function should be called after each time with the ACR120_Select() or ACR120_MultiTagSelect() function.*
2. *It is suggested to execute this function for Type A card or the function ACR120_READATQB for Type B card, just after the ACR120_Select operation, then call the PICC_InitBlockNumber according to the result of the respective functions.*

Example:

```
//=====================================================================
//'Selects a single card and returns the card ID (Serial Number)
//=====================================================================

   //Variable Declarations
   BYTE ResultSN[11];
   BYTE TagType;
```

```
    BYTE ResultTag;
    char SN[100];
    UINT8 SID=1;
    BYTE  DataLength, pData[10], ResponseDataLength, pResponseData[100];
    INT16 TimeOut=50, i, CardFrameSize;
    char  pdata[500];
    char  *ATS_ATQB;


      retcode = ACR120_Select(rHandle, SID, &TagType, &ResultTag, ResultSN);


     //'Check if Retcode is Error
     if (retcode >=0 )
     {
        if ((TagType == 4) || (TagType == 5)) {
          // Type A cards
          memcpy(SN,ResultSN, 7);
        } else {
           memcpy(SN,ResultSN, ResultTag);
        }


       // Get the Info Bytes, if it is a type B card

       CardFrameSize=0;
       pdata[0]='\0';
       ResponseDataLength=0;

       if (TagType==0x80) {
       // Type B Cards
          if (ACR120_ReadATQB(rHandle, SID, pResponseData)==0) {
ResponseDataLength=7;
          CardFrameSize=pResponseData[10]>>4;
          }
       } else if (TagType < 0x80 || TagType == 0xff) {
       // Type A Cards
          if     (PICC_RATS(rHandle,     SID,     4,     &ResponseDataLength,
pResponseData)>=0) {
             CardFrameSize=pResponseData[1]&0x0f;
          }
       }

    PICC_InitBlockNumber(CardFrameSize);      // Set communiation frame size

     } else  {

          // Card Selection Error handling here
     }
```

2.2.27. PICC_Xch_APDU

Format:

```
DLLAPI INT16 AC_DECL PICC_Xch_APDU (
            INT16 rHandle,
            UINT8 station_ID,
            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	The handle to ACR1281S1-C8 reader returned by ACR120_Open	
station_D	The Station ID of ACR1281S1-C8 reader	
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 28: PICC_Xch_APDU Function 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 - Table of Error Codes**.

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
    }
}
```

2.2.28. PICC_RATS

Format:

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

Function Description:

This function is only valid for ISO 14443 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 ISO 14443 mode.

Parameters	Description	
rHandle	The handle to ACR1281S1-C8 reader returned by ACR120_Open	
station_ID	The Station ID of ACR1281S1-C8 reader	
FSDI	An index to a maximum frame size which the reader can accept. The value should not exceed 4, i.e. 48 bytes.	
typeA	A Boolean value indicates the card type. This value should always be TRUE.	
pATSlen	A pointer to the location storing the length of the ATS received	
pATS	A pointer to the ATS received	
Return Value	INT16	Result code. 0 means success.

Table 29: PICC_RATS Function Description

The FSDI to (Frame Size for proximity coupling Device) FSD conversion:

FSDI	FSD (in bytes)
0	16
1	24
2	32
3	40
4	48
5	64

FSDI	FSD (in bytes)
6	96
7	128
8	256
Otherwise	RFU

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 - Table of Error Codes**. For detailed meaning of the ATS, please refer to corresponding documents.

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

2.2.29. PICC_Deselect

Format:

```
DLLAPI INT16 AC_DECL PICC_Deselect(
            INT16 rHandle,
            UINT8 station_ID,
            BOOL typeA);
```

Function Description:

This function sends DESELECT (card close) signal to the cards running ISO 14443 Part 4 (T=CL) protocol.

Parameters	Description	
rHandle	The handle to ACR1281S1-C8 reader returned by ACR120_Open	
station_ID	The Station ID of ACR1281S1-C8 reader	
typeA	A Boolean value indicates the card type, TRUE for Type A cards, FALSE for Type B cards	
Return Value	INT16	Result code. 0 means success.

Table 30: PICC_Deselect Function 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 - Table of Error Codes**.

2.2.30. ACR120_ReadATQB

Format:

```
DLLAPI INT16 AC_DECL ACR120_ReadATQB(INT16   rHandle,
                  UINT8 stationID,
                  UINT8 *pATQB);
```

Function Description:

This function reads the ATQB data from the card. This function only works after a successful Select command on an ISO 14443 Type B card.

Parameters	Description
rHandle	The handle to ACR1281S1-C8 reader returned by ACR120_Open
stationID	The Station ID of ACR1281S1-C8 reader
pATQB	A pointer to a 7 byte data array containing the ATQB. The first 4 bytes and last 3 bytes being the Application Data and Protocol Info respectively.

Table 31: ACR120_ReadATQB Function Description

Returns:

The return value is 0 if the function is successful. Otherwise, it returns a negative value containing the error code. An error will return if the *ACR120_Select* command is not previously executed with success on a Type B card.

Note: *This function only works after a successful Select command on an ISO 14443 Type B card.*

Example:

```
      INT16 RetCode;
      UINT8 SID;
      UINT8 pSN[4]
      UINT8 pATQB[7];
      BOOL pHaveTag;
      UINT8 pTAG;

SID=1;
// Select a type B card
RetCode = ACR120_Select (rHandle, SID, &pHaveTag, &pTAG, pSN);

RetCode = ACR120_ReadATQB (rHandle, SID, pATQB);

if (RetCode==0) {
StrMsg.Format(“Card ATQB = %02X%02X%02X%02X%02X%02X%02X”,
   pATQB[0], pATQB[1], pATQB[2], pATQB[3], pATQB[4],
pATQB[5], pATQB[6]);
}
```

2.2.31. ACR120_SetFWI

```
ACR120_SetFWI( INT16 hReader,
            UINT8 stationID,
            UINT8 *pFWI)
```

Function Description:

This function alters the default Frame Waiting Index (FWI) which the ISO14443 cards reported during the initial card operation. The value of the reader is adopted through the ACR120_RATS() operation in type A cards and the ACR120_Select() operation in type B cards. In some instances, the frame waiting time may need to extend to wait for certain computation intensive operations on the card, which the card will request for a Waiting Time Extension (WTX) inside the ISO14443 part 4 communication.

This function is called by the ACR120_Xch_APDU() API and is usually not needed to be called by high level application explicitly.

Parameters	Description	
RHandle	The handle to ACR1281S1-C8 reader returned by ACR120_Open	
stationID	The Station ID of ACR1281S1-C8 reader	
pFWI	Contains the new FWI to be set (value <= 0x0Eh)	
Return Value	INT16	Result code. 0 means success.

Table 32: ACR120_SetFWI

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 - Table of Error Codes**. The new FWI value is updated by the function.

Note: *According to the ISO 14443 Part 4 Specifications, the maximum value of FWI is 0x0Eh. The FWI value will be updated by the maximum value the card reader that can support. The actual waiting time FWT is calculated by the following formula:*

*FWT = (256 * 16 / 13560000) * (2 ^ FWI)*

which gives 4.94s if FWI = 14

2.2.32. ACR120_FlipUserPort

Format:

```
DLLAPI INT16 AC_DECL ACR120_FlipUserPort(INT16  rHandle,
                    UINT8 stationID,
                    UINT8 PortFlipAction);
```

Function Description:

This function is added to ease the LED/Relay flipping and buzzer sounding operation. The *ACR120_WriteUserPort* only turns *ON* or *OFF* of the corresponding devices according to the argument userPortState (c.f. ACR120_WRITEUSERPORT function), it could be difficult for the controlling PC program to time the activation duration precisely. This function call activates the LED/Relay and Buzzer for a precise duration defined in EEPROM values in address 0x07h and 0x08h respectively. This function will not take any action when called if the value is zero (0x00h) in the respective EEPROM locations.

Parameters	Description	
RHandle	The handle to ACR1281S1-C8 reader returned by ACR120_Open	
stationID	The Station ID of ACR1281S1-C8 reader	
userPortState	**Value**	**Action**
	0x00h	No action
	0x01h	Turns on LED/Relay on for m milliseconds
	0x02h	Turns on Buzzer on for m milliseconds
	0x03h	Turns on LED/Relay and Buzzer on for the respective durations
Return Value	INT16	Result code. 0 means success.

Table 33: ACR120_FlipUserPort

m = 200ms x (the value in EEPROM location 0x07h)

n = 200ms x (the value in EEPROM location 0x08h)

Returns:

The return value is always 0 indicates a successful execution.

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

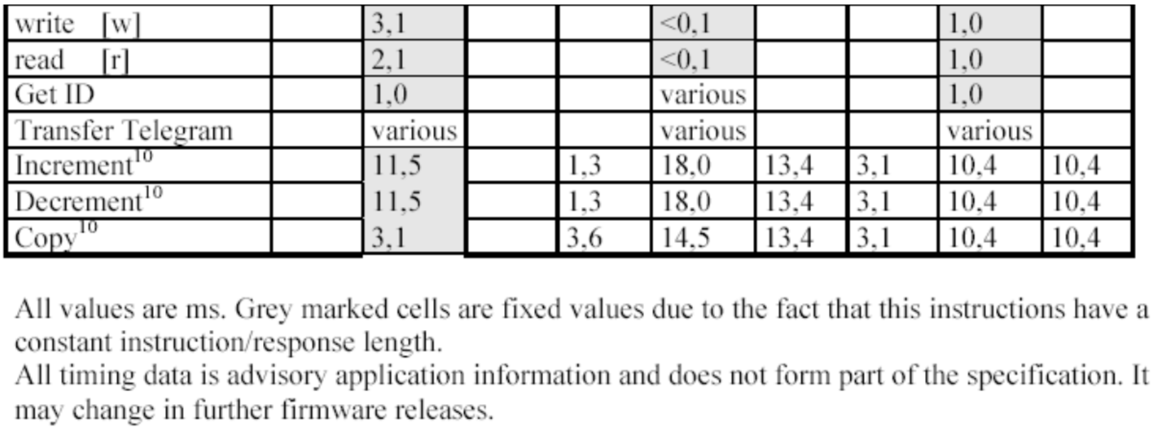
Sector Number on Card	Sector Number for Log-in	Block Number	Card Type	
0x00h	0x00h	0x00h-0x03h	Mifare 1K Standard Sectors	Mifare 4K Standard Sectors
0x01h	0x01h	0x04h-0x07h		
0x02h	0x02h	0x08h-0x0Bh		
0x03h	0x03h	0x0Ch-0x0Fh		
0x04h	0x04h	0x10h-0x13h		
0x05h	0x05h	0x14h-0x17h		
0x06h	0x06h	0x18h-0x1Bh		
0x07h	0x07h	0x1Ch-0x1Fh		
0x08h	0x08h	0x20h-0x23h		
0x09h	0x09h	0x24h-0x27h		
0x0Ah	0x0Ah	0x28h-0x2Bh		
0x0Bh	0x0Bh	0x2Ch-0x2Fh		
0x0Ch	0x0Ch	0x30h-0x33h		
0x0Dh	0x0Dh	0x34h-0x37h		
0x0Eh	0x0Eh	0x38h-0x3Bh		
0x0Fh	0x0Fh	0x3Ch-0x3Fh		
0x10h	0x10h	0x40h-0x43h		
0x11h	0x11h	0x44h-0x47h		
0x12h	0x12h	0x48h-0x4Bh		
0x13h	0x13h	0x4Ch-0x4Fh		
0x14h	0x14h	0x50h-0x53h		
0x15h	0x15h	0x54h-0x57h		
0x16h	0x16h	0x58h-0x5Bh		
0x17h	0x17h	0x5Ch-0x5Fh		
0x18h	0x18h	0x60h-0x63h		
0x19h	0x19h	0x64h-0x67h		
0x1Ah	0x1Ah	0x68h-0x6Bh		
0x1Bh	0x1Bh	0x6Ch-0x6Fh		
0x1Ch	0x1Ch	0x70h-0x73h		
0x1Dh	0x1Dh	0x74h-0x77h		
0x1Eh	0x1Eh	0x78h-0x7Bh		
0x1Fh	0x1Fh	0x7Ch-0x7Fh		
0x20h	0x20h	0x80h-0x8Fh		Big sectors

Sector Number on Card	Sector Number for Log-in	Block Number	Card Type	
0x21h	0x24h	0x90h-0x9Fh		
0x22h	0x28h	0xA0h-0xAFh		
0x23h	0x2Ch	0xB0h-0xBFh		
0x24h	0x30h	0xC0h-0xCFh		
0x25h	0x34h	0xD0h-0xDFh		
0x26h	0x38h	0xE0h-0xEFh		
0x27h	0x3Ch	0xF0h-0xFFh		

Appendix C. 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



write [w]		3,1			<0,1			1,0	
read [r]		2,1			<0,1			1,0	
Get ID		1,0			various			1,0	
Transfer Telegram		various			various			various	
Increment[10]		11,5		1,3	18,0	13,4	3,1	10,4	10,4
Decrement[10]		11,5		1,3	18,0	13,4	3,1	10,4	10,4
Copy[10]		3,1		3,6	14,5	13,4	3,1	10,4	10,4

All values are ms. Grey marked cells are fixed values due to the fact that this instructions have a constant instruction/response length.
All timing data is advisory application information and does not form part of the specification. It may change in further firmware releases.