

# ACM Reader SDK Development Guideline

The SDK package is the software development kit developed by our company for users developing the application procedure conveniently. SDK is available to users in the form of dynamic linking library files.

Users using our ACM readers for users application software development, in accordance with the SDK development kit of our Company, users can efficiently and correctly complete DRF series reader application software development. SDK support the development of Visual C + +, VB, C + + Builder and Delphi.

SDK development guideline is the reference manual provided for users to do secondary development. Through reading the manual, users can quickly resolve the technical problems encountered in the process of secondary developing of our company's series of readers.

based on features of the functions, the SDK function can be divided into two types of functions: readers management function and label operation function.

## Reader management function

Reader management function is the function that achieves the functions that the host machine to set up working state of reader, enquiry the status and operate it.

## 1、OpenComm

Function prototype: `int OpenComm(HANDLE *hCom,char *com_port)`

Features: Open computer's serial port.

Input parameters:

●——hCom: Serial port handler

●——com\_port: Serial port file name

Using examples:

```
char comm[]="COM1";
OpenComm(&h_Com,comm);
if(h_Com != INVALID_HANDLE_VALUE)
{
    AfxMessageBox("Open the serial port successfully!");
}
else
```

```
{
    AfxMessageBox("Open the serial port for failure!");
}
```

## 2、CloseComm

Function prototype: void CloseComm(HANDLE comm)

Features: shut down the computer serial port.

Input parameters:

●——comm: Serial port handler

Using examples:

```
CloseComm(h_Com);
```

## 3、ReadSoftwareVersion

Function prototype: int ReadSoftwareVersion(HANDLE comm,char \*receive)

Features: Read software version number of reader.

Input parameters:

●——comm: Serial port handler

●——\*receive: to receive the array address of the software version. Which receive [0] is the main version value of the software, receive [1] is a minor version value of the software.

Returning results: the successful using returns 1, otherwise returns -1.

Using examples:

```
int flag;
char receive_data[2];
flag = ReadSoftwareVersion(h_Com,receive_data);
if(flag==1)
{
    CString string;
    string.Format("Software Version is %d.%d",receive_data[0],receive_data[1]);
    AfxMessageBox(string);
}
else
{
    AfxMessageBox("Read Software Version Failed!");
}
```

## 4、StopWorkSetting

Function prototype: int StopWorkSetting(HANDLE comm)

Features: stop transmitting power of reader

Input parameters:

●——comm: Serial port handler

Returning results: the successful using returns 1, otherwise returns -1.

Using examples:

```
int flag;
flag=StopWorkSetting(h_Com);
if(flag==1)
{
    AfxMessageBox("Operation OK!");
}
else
{
    AfxMessageBox("Operation Error!");
}
```

## 5、ResetReader

Function prototype: int ResetReader(HANDLE comm)

Features: reset readers.

Input parameters:

●——comm: Serial port handler

Returning results: the successful using returns 1, otherwise returns -1.

Using examples:

```
int flag;
flag = ResetReader(h_Com);
if(flag == 1)
{
    AfxMessageBox("Reset readers successfully!");
}
else
{
    AfxMessageBox("Reset reader for failure!");
}
```

## 6、QueryReaderSingleParameter

Function prototype: int QueryReaderSingleParameter(HANDLE comm,unsigned char msb,  
unsigned char lsb,unsigned char \*receive)

Features: enquiry for single parameter of readers.

Input parameters:

- comm: Serial port handler
- msb: need to enquiry for the high address of parameters
- lsb: need to enquiry for the low address of parameters
- \*receive: receive the array address of parameters

Returning results: the successful using returns 1, otherwise returns -1.

Using examples:

```
int flag;
unsigned char MSB=0x00;
unsigned char LSB=0x65;
unsigned char parameter[1];
flag = QueryReaderSingleParameter(h_Com,MSB,LSB,parameter);
if(1 == flag)
{
    CString string;
    string.Format("The parameter enquired is:%d",parameter[0]);
    AfxMessageBox(string);
}
else
{
    AfxMessageBox("Enquiring single parameter for failure!");
}
```

## 7、QueryReaderMultiParameter

Function prototype : int QueryReaderMultiParameter(HANDLE comm,unsigned char  
length,unsigned char msb,unsigned char  
lsb,unsigned char \*receive)

Features: enquire multiple parameters for readers

Input parameters:

- comm: Serial port handler
- msb: need to enquire starting high address of the parameters
- lsb: need to enquire starting low address of the parameters

●——\*receive: receive the array address of parameters

Returning results: the successful using returns 1, otherwise returns -1.

Using examples:

```
int flag;
unsigned char len=0x02;
unsigned char MSB=0x00;
unsigned char LSB=0x64;
unsigned char receive[30];
flag=QueryReaderMultiParameter(h_Com,len,MSB,LSB,receive);
if(1 == flag)
{
    CString string;
    string.Format("Enquiring a number of parameters are:%d-%d",receive[0],receive[1]);
    AfxMessageBox(string);
}
else
{
    AfxMessageBox("Enquire a number of parameters for failure!");
}
```

## 8、SetReaderSignleParameter

Function prototype : int SetReaderSignleParameter(HANDLE comm,unsigned char  
msb,unsigned char lsb,unsigned char data)

Features: Set single parameter of the reader

Input parameters:

●——comm: Serial port handler

●——msb: need to set the high address of the parameters

●——lsb: need to set the low address of the parameter

●——data: number set

Returning results: the successful using returns 1, otherwise returns -1.

Using examples:

```
unsigned char MSB=0x00;
unsigned char LSB=0x65;
unsigned char writevalue=0x78;//120
int flag;
flag = SetReaderSignleParameter(h_Com,MSB,LSB,writevalue);
```

```

if(1 == flag)
{
    AfxMessageBox("Setting single parameter successfully!");
}
else
{
    AfxMessageBox("Setting single parameter for failure!");
}

```

## 9、SetReaderMultiParameter

Function prototype : int SetReaderMultiParameter(HANDLE comm,unsigned char length,unsigned char msb,unsigned char lsb,unsigned char \*write\_data)

Features: set a number of parameters of reader at the same time

Input parameters:

●——comm: Serial port handler

●——msb: need to set the starting high address of the parameters

●——lsb: need to set the starting low address of the parameters

●——\*write\_data: write in the array address of data

Returning results: the successful using returns 1, otherwise returns -1.

Using examples:

```

unsigned char len=0x02;
unsigned char MSB=0x00;
unsigned char LSB=0x64;
unsigned char write[]={0xff,0x60};
int flag;
flag=SetReaderMultiParameter(h_Com,len,MSB,LSB,write);
if(flag == 1)
{

```

```

AfxMessageBox("Setting multiple parameters successfully!");

```

```

}

```

```

else

```

```

{

```

```

AfxMessageBox("Setting multiple parameters for failure!");

```

```

}

```

Reader label operation function

Reader label operation functions are the operating functions to achieve tags identifi

cation, labels reading, labels writing-in, locking labels memory and enquiring the locked state of labels memory, and other functions.

## 1、TagIdentify

Function prototype: `int TagIdentify(HANDLE comm,unsigned char *receive)`

Features: ID identification of single tag

Input parameters:

●——comm: Serial port handler

●——\*receive: to receive the array address of ID

Returning results: the successful using returns 1, otherwise returns -1.

Using examples:

```
int flag;
unsigned char receive[20];
flag=TagIdentify(h_Com,receive);
if(flag==1)
{
    CString string;
    string.Format("%02x %02x %02x %02x %02x %02x %02x %02x %02x %02x %02x %02x %02x",receive[0],receive[1],receive[2],receive[3],receive[4],receive[5],receive[6],receive[7],receive[8],receive[9],receive[10],receive[11]);
    AfxMessageBox(string);
}
else
    AfxMessageBox("Operation Error!");
```

## 2、InitializeTag

Function prototype: `int InitializeTag(HANDLE comm)`

Features: initialize tag

Input parameters:

●——comm: Serial port handler

Returning results: the successful using returns 1, otherwise returns -1.

Using examples:

```
int flag;
flag = InitializeTag(h_Com);
if(flag == 1)
{
```

```

        AfxMessageBox("Initialize Tag OK!");
    }
    else
    {
        AfxMessageBox("Initailize Tag Error!");
    }
}

```

### 3、TagLock

Function prototype: int TagLock(HANDLE comm,unsigned char locktype)

Features: Lock card

Input parameters:

●——comm: Serial port handler

●——locktype: designate to lock some area

```

0x00: lock USER area
0x01: lock TID area
0x02: lock EPC area

```

Returing results: the successful using returns 1, otherwise returns -1.

Using examples:

```

int flag;
flag = TagLock(h_Com,0x00);
if(1 == flag)
{
    AfxMessageBox("Lock Tag Success!");
}
else
{
    AfxMessageBox("Lock Tag Failed!");
}

```

### 4、TagKill

Function prototype: int TagKill(HANDLE comm,unsigned char pa1,unsigned char pa2,  
unsigned char pa3,unsigned char pa4)

Features: destruct labels

Input parameters:

●——comm: Serial port handler

●——pa1: password 1

●——pa2: password 2

●——pa3: password 3



●——pa4: password 4

Returning results: the successful using returns 1, otherwise returns -1.

## 5、TagWriteSingleWord

Function prototype : TagWriteSingleWord(HANDLE comm,unsigned char membank,unsigned char address,int data);

Features: to write in a single word in some region of card. (Note: When writing EPC district, the region of address is 0 and address is 1 can not be written in).

Input parameters:

●——comm: Serial port handler

●——membank: the region to write, the numbers are as follows:

0x00——Reserving area

0x01——EPC area

0x02——TID area

0x03——users area

●——address: the address to write in the area, ranged from 0-7.

●——data: the data to write in, the type is int.

Returning results: the successful using returns 1, otherwise returns -1.

Using examples: write in 0 x1234 in district 2 of the EPC

```
int flag;
unsigned char mem=0x01;
unsigned char add=0x02;
int write_data = 0x1234;
flag = TagWriteSingleWord(comm,mem,add,write_data);
if( flag == 1)
{
    MessageBox("Write Success!");
}
Else
{
    MessageBox("Write Failed!");
}
```

## 6、TagRead

Function prototype: `int TagRead(HANDLE comm,int length,unsigned char membank,  
unsigned char address,unsigned char *receive)`

Features: read the content of some domain in card.

Input parameters:

●——comm: Serial port handler

●——length: the length to read, the range is 1-8

●——membak: the region to read, the value is the same to the parameters written (reference to TagWriteSingleWord function)

●——address: to read the address in the region, ranged from 0-7

●——receive: to receive the first address of reading the data array

Returing results: the successful using returns 1, otherwise returns -1.

Using examples: (to read a Word in the Area 2 of EPC (1word = 2 Byte))

```
int flag;  
unsigned char mem=0x01;  
unsigned char add=0x02;  
int len = 0x01;  
unsigned char receive_data[10];  
flag = TagRead(comm,len,mem,sdd,receive_data);  
if(flag == 1)  
{  
    CString str;  
    str.Format("Read data is %02x-%02x",receive_data[0],receive_data[1]);  
    MessageBox(str);  
}  
else  
{  
    MessageBox("Read Failed!");  
}
```

## 7、ReadMultiTag

Function prototype: `int ReadMultiTag(HANDLE comm,unsigned char *count,unsigned  
char *id_buffer)`

Features: Reading multi-card

Input parameters:

●——comm: Serial port handler

●——count: memory the array address of reading the card pieces number

●——id\_buffer: Store the ID number of reading card (one ID number occupies 12 bytes, such as from id\_buffer [0] to id\_buffer [11] stores the ID number of reading the first card, and so on like this)

Returning results: the successful using returns 1, otherwise returns -1.

Using examples:

```
int flag;
unsigned char number_number[10];
unsigned char buffer[2048];
CString string;
flag = ReadMultiTag(h_Com,number_number,buffer);
if(flag == 1)
{
    string.Format("%02x-%02x-%02x-%02x-%02x-%02x-%02x-%02x-%02x-%02x-%02x-%02x-%02x\r\n",
    buffer[(number_number[0]-1)*12+0],buffer[(number_number[0]-1)*12+1],
    buffer[(number_number[0]-1)*12+2],buffer[(number_number[0]-1)*12+3],
    buffer[(number_number[0]-1)*12+4],buffer[(number_number[0]-1)*12+5],
    buffer[(number_number[0]-1)*12+6],buffer[(number_number[0]-1)*12+7],
    buffer[(number_number[0]-1)*12+8],buffer[(number_number[0]-1)*12+9],
    buffer[(number_number[0]-1)*12+10],buffer[(number_number[0]-1)*12+11]);
    MessageBox(string);
}
else
{
}
```

## 8、AfreshIdentifyTag

Function prototype: int AfreshIdentifyTag(HANDLE comm)

Features: re-identify multiple cards within the effective range of the reader.

Input parameters:

●——comm: Serial port handler

Returning results: the successful using returns 1, otherwise returns -1.

Using examples:

```
int flag;
flag = AfreshIdentifyTag(h_Com);
```

```

if(1 == flag)
{
    MessageBox("OK!");
}
else
{
    MessageBox("ERROR!");
}

```

## 9、AfreshGetData

Function prototype: int AfreshGetData(HANDLE comm,unsigned char \*count,unsigned char \*data)

Features: re-get the ID number reading last time from the memory of reader.

Input parameters:

●——comm: Serial port handler

●——\*count: store the first address of ID number pieces arrays

●——\*data: store the first address of ID number arrays (the first ID number is stored in the data [0] to the data [11], and so on like this for the other).

Returning results: the successful using returns 1, otherwise returns -1.

Using examples:

```

unsigned char receive_data[1024];
unsigned char count[3];
int flag;
flag = AfreshGetData(h_Com,count,receive_data);
if(1 == flag)
{

```

//treat ID number

```

}
Else
{

```

//Error Handling

```

}

```