915MHz RFID Reader

Dynamic Library User Guide

Table of contents

1. Function library description

There are 8 files in the dynamic library, including adpcom.dll, adpcom.lib, adpnet.dll, adpnet.lib, adppub.dll, adppub.lib, adpusb.dll, adpusb.lib,. The dynamic library is the interface standard of Microsoft Windows, and the popular software development tools VC, VB, VF, Delphi, C++ Builder, Power Builder, etc. can all be used.

This function library is only applicable to the card reader working in passive mode or answering mode; when the card reader is working in active mode, the function library May cause data conflicts or overlapping errors;

2. Serial interface device

It is only applicable to connect readers with **RS-232** or **RS-485 .** Select **adpcom.dll, adpcom.lib** dynamic library;

2.1. Device management function

2.1.1.ap_open (connect serial device)

HANDLE _stdcall ap_open (int nPort, int nBaud)

Function: Initialize serial port

parameters: nPort: serial port number, the value is 1~250

nBaud: communication baud rate 1200~115200, (default value for non-customized type: 9600) error code table

example: int m_handle;

m_handle = ap_open (2,9600);//initialize serial port 1, baud rate 9600

If it is a WIN32 program, m_handle is the device handle, see the following example:

HANDLE m_handle; m_handle = ap_open (2, 9600); if(m_handle<0)

MessageBox("ap_open error");

2.1.2.ap_close (release serial port)

void _stdcall ap_close (HANDLE m_handle)

Function: Release the serial port Parameters: m_handle: communication device identifier Return: None

example: ap_close (m_handle);

Note: In the WIN32 environment, m_handle is the device handle of the serial port, which must be released before it can be connected again.

2.1.3.ap_getaddress (Get device communication address and version information)

int _stdcall ap_getaddress (HANDLE m_handle, int *m_oAddress, int *m_oVer)

Function: Obtain device communication

address Parameters: m_handle: Communication device identifier

m_iAddress: Obtained device communication address pointer address

m_oVer: Obtained device version information pointer address

Return: Return 0 successfully

example: int m_iAddress; int st; st

= ap_getaddress

(m_handle, m_iAddress, m_oVer);

2.1.4.ap_setaddress (set device communication address)

int _stdcall ap_setaddress (HANDLE m_handle, int m_iAddress , int idata)

Function: Set device communication address

Parameters: m_handle: communication device identifier

m_iAddress: device current communication address

idata: device communication address to be set

Return: Return 0 successfully

example: int idata = 65534; int st; st =

ap_setaddress (m_handle,

m_iAddress, idata);

2.1.5.ap_getconfig (get basic device parameters)

int _stdcall ap_getconfig (HANDLE m_handle, int m_iAddress , unsigned char *oData, unsigned char *oSize)

Function: Get the basic parameters of the

device Parameters: m_handle: communication device identifier

m_iAddress: current communication address

oData: return the basic parameter pointer address (see the basic parameter table)

oSize: return the basic parameter byte number pointer address

Return: Return 0 successfully

example: unsigned char oData[28]; unsigned char oSize; int st; st = ap_getconfig (m_handle, m_iAddress, oData, oSize);

2.1.6.ap_setconfig (set the basic parameters of the device)

int _stdcall ap_setconfig (HANDLE m_handle, int m_iAddress , unsigned char *iData, unsigned char iSize)

Function: Set the basic parameters of the device Parameters: m_handle: communication device identifier
m_iAddress: current communication address
iData: pointer address of basic parameters to be set (see basic parameter table)
iSize: number of bytes of basic parameters to be set

Return: Return 0 successfully

example: unsigned char iData[] = {0x01,}; unsigned char iSize = 28; int st; st = ap_setconfig (m_handle, m_iAddress, iData, iSize);

2.1.7.ap_gettcpip (get device network communication parameters)

int _stdcall ap_gettcpip (HANDLE m_handle, int m_iAddress , unsigned char *oData, unsigned char *oSize)

Function: Get device network communication parameters Parameters: m_handle: communication device
identifier m_iAddress: current communication
address oData: return parameter pointer address (see TCPIP parameter
table) oSize: return parameter byte number pointer address

Return: Return 0 successfully

example: unsigned char oData[20]; unsigned char oSize; int st; st = ap_gettcpip (m_handle, m_iAddress, oData, oSize);

2.1.8.ap_settcpip (set device network communication parameters)

int _stdcall ap_settcpip (HANDLE m_handle, int m_iAddress , unsigned char *iData, unsigned char iSize)

Function: Set device network communication parameters

Parameters: m_handle: communication device identifier

m_iAddress: current communication address

iData: pointer address of parameters to be set (see TCPIP parameter table)

iSize: number of bytes of parameters to be set

Return: Return 0 successfully

example: unsigned char iData[]={0x01,}; unsigned char iSize=20; int st; st = ap_setconfig (m_handle, m_iAddress, iData, iSize);

2.2.ISO18000-6B protocol label operation function

2.2.1.ap_identify6b (identify IS0180000-6B protocol label)

int _stdcall ap_identify6b (HANDLE m_handle, int m_iAddress, unsigned char *oData, unsigned char *oSize)

Function: Identify IS0180000-6B protocol label Parameters:

m_handle: communication device identifier m_iAddress:

current communication address oData: return

data pointer address oSize: return data byte

number pointer address

Return: Return 0 successfully

example: unsigned char oData[12]; unsigned char oSize; int st; st = ap_identify6b (m_handle, m_iAddress, oData, oSize);

Note: Get the unique **ID number in the label,** and get **12 bytes of data. Generally, the first 8** bytes are the card number, and the last **4** bytes are **0** by default ;

2.2.2.ap_read6b (read the data in the IS0180000-6B protocol label)

int _stdcall ap_read6b (HANDLE m_handle, int m_iAddress , unsigned char *oData, unsigned char iAddr, unsigned char iSize)

Function: Read the data in the IS0180000-6B protocol label Parameters:

m_handle: communication device identifier m_iAddress: current

communication address oData: return data pointer address

iAddr: data address to be obtained oSize: return data byte

number pointer address

Return: Return 0 successfully

example: unsigned char oData[12];

unsigned char iAddr = 18; unsigned char iSize
= 12; int st; st = ap_read6b (m_handle,
m_iAddress, oData, iAddr, oSize); Note: **Get**
the custom data in the label, the address starts from **0 ;**

2.2.3.ap_write6b (write data to IS0180000-6B protocol tag)

int _stdcall ap_write6b (HANDLE m_handle, int m_iAddress , unsigned char *iData, unsigned char iAddr, unsigned char iSize)

Function: Write data to the IS0180000-6B protocol label Parameters:
m_handle: Communication device identifier m_iAddress: Current
communication address iData: Return data iAddr: Data
address to be written iSize: Number of data bytes to be
written

Return: Return 0 successfully

example: unsigned char iData [2]={0x01,0x02}; unsigned char iAddr =
18; unsigned char iSize = 2; int st; st = ap_write6b (m_handle,
m_iAddress, iData, iAddr, iSize);

Note: When writing data into the label, the address starts from 18 , and the data at the previous position is an unmodifiable area;

2.3. EPC (GEN 2) protocol label operation function

2.3.1.ap_identify6c (identify EPC (GEN 2) protocol label)

int _stdcall ap_identify6c (HANDLE m_handle, int m_iAddress, unsigned char *oData, unsigned char *oSize)

Function: Identify EPC (GEN 2) protocol label Parameters:
m_handle: communication device identifier m_iAddress:
current communication address oData:
return data pointer address oSize: return
data byte count pointer address

Return: Return 0 successfully

example: unsigned char oData[12]; unsigned char
oSize; int st; st = ap_identify6c (m_handle,
m_iAddress, oData, oSize);

Note: Get the **12-** byte data in the **EPC** area of the label ;

2.3.2.ap_identify6cmult (identify EPC (GEN 2) protocol multi-label)

int _stdcall ap_identify6cmult (HANDLE m_handle, int m_iAddress, unsigned char *oData, unsigned char *oSize)

Function: Identify EPC (GEN 2) protocol multi-label
parameters: m_handle: communication device identifier
m_iAddress: current communication address
oData: return data pointer address oSize:
return data byte number pointer address

Return: Return 0 successfully

example: unsigned char oData[12]; unsigned char
oSize; int st; st = ap_identify6c (m_handle,
m_iAddress, oData, oSize);

Note: Obtain **12** bytes of data in **the EPC** area of multiple tags ;

2.3.3.ap_read6c (read data in EPC(GEN 2) protocol tag)

int _stdcall ap_read6c (HANDLE m_handle, int m_iAddress , unsigned char *oData, unsigned char iMem, unsigned char iAddr, unsigned char iSize)

Function: Read the data in the EPC (GEN 2) protocol label
Parameters: m_handle: Communication device identifier
m_iAddress: Current communication address
oData: Return data pointer address iMem: Data
block address to be obtained iAddr: Data address
to be obtained oSize: Return data word Section
number

Return: Return 0 successfully

example: unsigned char oData[12]; unsigned char
iMem = 1; unsigned char iAddr = 2;
unsigned char iSize = 2; int st; st =
ap_read6c (m_handle, m_iAddress, oData,
iMem, iAddr, oSize);

Card storage division:

block name	store content	Block address	byte capacity	read/write	
Reserved	store KILL PASSWORD and ACCESS PASSWORD	00H	8	64bits read only	
EPC(EPC ÿ)	Access EPC number	01H	12	96bits	
TIME(TIME ÿ)	access tag identification number,	02H	24	Readable and writable 196bits read only	

	Each TID number should be unique				
USER(USER ÿ)	Access user-defined data	03H	64	512bits	readable and writable

Note: The data address in **the EPC** area starts from **2 ;** in any area, each address stores **2** bytes;

for example:

EPC area card number: **01 02 03 04 05 06 07 08 09 10 11 12;** iMem = 1; iAddr = 2; iSize = 4; The obtained data is: 01 01 02 03 04 (the front 01 is the antenna number, integrated card reader The default is 01, and the multi-channel card reader is the channel number)

iMem = 1; iAddr = 3; iSize = 4; The obtained data is: 01 03 04 05 06 ;

iMem = 1; iAddr = 4; iSize = 4; The obtained data is: 01 05 06 07 08 ;

analogy…

2.3.4.ap_write6c (write data to EPC(GEN 2) protocol tag)

int _stdcall ap_write6c (HANDLE m_handle, int m_iAddress , unsigned char *iData, unsigned char iMem, unsigned char iAddr, unsigned char iSize)

Function: Write data to the EPC (GEN 2) protocol tag Parameters:
m_handle: communication device identifier m_iAddress: current communication address iData: pointer address of data to be written iMem: address of data block to be written iAddr: address of data to be written iSize: the number of data bytes to be written

Return: Return 0 successfully

example: unsigned char iData [2]={0x01,0x02}; unsigned char iMem = 1; unsigned char iAddr = 2; unsigned char iSize = 2; int st; st = ap_write6b (m_handle, m_iAddress, iData, iMem, iAddr, iSize);

2.3.5.ap_encrypt (EPC(GEN 2) protocol label encryption)

int _stdcall ap_encrypt (HANDLE m_handle, int m_iAddress)

Function: EPC (GEN 2) protocol label encryption
parameters: m_handle: communication device identifier
m_iAddress: current communication address

Return: Return 0 successfully

example: int st;

st = ap_encrypt (m_handle, m_iAddress);

3. Network interface device

It is only applicable to readers connected by **TCPIP ,** and the card reader is only used on the

server side. Select **adpnet.dll, adpnet.lib** dynamic library;

3.1. Device management function

3.1.1.an_open (connect network device)

SOCKET _stdcall an_open (char * ulAddress, int iPort)

Function: Connect network equipment

Parameters: ulAddress: remote IP address, default value 192.168.1.115 iPort: remote IP port,

default value 49152

Return: If successful, it will return the network connection handle, if it fails, it will return the value, see the error code table

example: int m_handle;

m_handle = an_open (“192.168.1.115”, 49152);

3.1.2.an_close (release network)

void _stdcall an_close (SOCKET m_handle)

Function: Release the serial

port Parameters: m_handle: communication device identifier

Return: None

example: an_close (m_handle);

Note: In the WIN32 environment, the device handle of the m_handle network must be released before it can be connected again.

3.1.3.an_getaddress (Get device communication address and version information)

int _stdcall an_getaddress (SOCKET m_handle, int *m_oAddress, int *m_oVer)

Function: Obtain device communication

address Parameters: m_handle: Communication device

identifier m_iAddress: Obtained device communication address pointer

address m_oVer: Obtained device version information pointer address

Return: Return 0 successfully

3.1.4.an_setaddress (set device communication address)

int _stdcall an_setaddress (SOCKET m_handle, int m_iAddress , int idata)

Function: Set device communication

address Parameters: m_handle: communication device

identifier m_iAddress: device current communication

address idata: device communication address to be set

Return: Return 0 successfully

3.1.5.an_getconfig (Get basic parameters of the device)

int _stdcall an_getconfig (SOCKET m_handle, int m_iAddress , unsigned char *oData, unsigned char *oSize)

Function: Get the basic parameters of the

device Parameters: m_handle: communication device identifier

m_iAddress: current communication address

oData: return the basic parameter pointer address (see the basic parameter

table) oSize: return the basic parameter byte number pointer address

Return: Return 0 successfully

3.1.6.an_setconfig (set the basic parameters of the device)

int _stdcall an_setconfig (SOCKET m_handle, int m_iAddress , unsigned char *iData, unsigned char iSize)

Function: Set the basic parameters of the

device Parameters: m_handle: communication device identifier

m_iAddress: current communication address

iData: pointer address of basic parameters to be set (see basic parameter table)

iSize: number of bytes of basic parameters to be set

Return: Return 0 successfully

3.1.7.an_gettcpip (get device network communication parameters)

int _stdcall an_gettcpip(SOCKET m_handle, int m_iAddress , unsigned char *oData, unsigned char *oSize)

Function: Get device network communication

parameters Parameters: m_handle: communication device
identifier m_iAddress: current communication
address oData: return parameter pointer address (see TCPIP parameter
table) oSize: return parameter byte number pointer address

Return: Return 0 successfully

Note: When the network version sets **TCPIP** parameters, the card reader will restart and needs to be reconnected;

3.1.8.an_settcpip (set device network communication parameters)

int _stdcall an_settcpip (SOCKET m_handle, int m_iAddress , unsigned char *iData, unsigned char iSize)

Function: Set device network communication

parameters Parameters: m_handle: communication device
identifier m_iAddress: current communication
address iData: pointer address of parameters to be set (see TCPIP parameter
table) iSize: number of bytes of parameters to be set

Return: Return 0 successfully

3.2.ISO18000-6B protocol label operation function

3.2.1.an_identify6b (identify IS0180000-6B protocol label)

int _stdcall an_identify6b (SOCKET m_handle, int m_iAddress, unsigned char *oData, unsigned char *oSize)

Function: Identify IS0180000-6B protocol label Parameters:

m_handle: communication device identifier m_iAddress:
current communication address oData: return
data pointer address oSize: return data byte
count pointer address

Return: Return 0 successfully

3.2.2.an_read6b (read the data in the IS0180000-6B protocol label)

int _stdcall an_read6b (SOCKET m_handle, int m_iAddress , unsigned char *oData, unsigned char iAddr, unsigned char iSize)

Function: Read the data in the IS0180000-6B protocol label Parameters:

m_handle: communication device identifier m_iAddress: current
communication address

oData: return data pointer address
iAddr: data address to be obtained
oSize: return data byte number pointer address

Return: Return 0 successfully

3.2.3.an_write6b (write data to IS0180000-6B protocol tag)

int _stdcall an_write6b (SOCKET m_handle, int m_iAddress , unsigned char *iData, unsigned char iAddr, unsigned char iSize)

Function: Write data to the IS0180000-6B protocol label Parameters:
m_handle: Communication device identifier m_iAddress: Current
communication address iData: Return data iAddr: Data
address to be written iSize: Number of data bytes to be
written

Return: Return 0 successfully

3.3. EPC (GEN 2) protocol label operation function

3.3.1.an_identify6c (identify EPC (GEN 2) protocol label)

int _stdcall an_identify6c (SOCKET m_handle, int m_iAddress, unsigned char *oData, unsigned char *oSize)

Function: Identify EPC (GEN 2) protocol label
Parameters: m_handle: communication device identifier
m_iAddress: current communication
address oData: return data pointer
address oSize: return data byte count pointer address

Return: Return 0 successfully

3.3.2.an_identify6cmult (identify EPC (GEN 2) protocol multi-label)

int _stdcall an_identify6cmult (SOCKET m_handle, int m_iAddress, unsigned char *oData, unsigned char *oSize)

Function: Identify EPC (GEN 2) protocol multi-label
parameters: m_handle: communication device identifier
m_iAddress: current communication
address oData: return data pointer address
oSize: return data byte count pointer address

Return: Return 0 successfully

3.3.3.an_read6c (read data in EPC(GEN 2) protocol tag)

int _stdcall an_read6c (SOCKET m_handle, int m_iAddress , unsigned char *oData, unsigned char iMem, unsigned char iAddr, unsigned char iSize)

Function: Read the data in the EPC (GEN 2) protocol label
Parameters: m_handle: Communication device identifier
m_iAddress: Current communication address
oData: Return data pointer address iMem: Data
block address to be obtained iAddr: Data address
to be obtained oSize: Return data word Section
number

Return: Return 0 successfully

3.3.4.an_write6c (write data to EPC(GEN 2) protocol tag)

int _stdcall an_write6c (SOCKET m_handle, int m_iAddress , unsigned char *iData, unsigned char iMem, unsigned char iAddr, unsigned char iSize)

Function: Write data to the EPC (GEN 2) protocol tag Parameters:
m_handle: communication device identifier m_iAddress: current
communication address iData: pointer address of data
to be written iMem: address of data block to be written
iAddr: address of data to be written iSize: the number
of data bytes to be written

Return: Return 0 successfully

3.3.5.an_encrypt (EPC(GEN 2) protocol tag encryption)

int _stdcall an_encrypt (SOCKET m_handle, int m_iAddress)

Function: EPC (GEN 2) protocol label encryption
parameters: m_handle: communication device identifier
m_iAddress: current communication address

Return: Return 0 successfully

4. USB interface device

It is only applicable to readers connected by **USB .**

Select **adpusb.dll, adpusb.lib** dynamic library;

4.1. Device management function

4.1.1.ad_open (connect USB device)

HANDLE __stdcall ad_open ()

Function: Connect the USB device

Return: If successful, return the device connection handle; if failed, return value, see error code table

example:

```
int m_handle = -1;  While
(m_handle != 0)  {

        m_handle = ad_open ();
}
```

4.1.2.ad_close (release USB device)

void __stdcall ad_close(HANDLE m_handle)

Function: Release the serial

port Parameters: m_handle: communication device

identifier Return: None

example:

```
ad_close (m_handle);
```

4.1.3.ad_exitprogram (exit programming mode)

int _stdcall ad_exitprogram (HANDLE m_handle) Function: Exit programming

mode Parameters: m_handle: communication device identifier Returns: None

example:

```
ad_close (m_handle);
```

4.1.4.ad_getaddress (Get device communication address and version information)

int _stdcall ad_getaddress (HANDLE m_handle, int *m_oAddress, int *m_oVer)

Function: Obtain device communication

address Parameters: m_handle: Communication device

identifier m_iAddress: Obtained device communication address pointer

address m_oVer: Obtained device version information pointer address

Return: Return 0 successfully

4.1.5.ad_setaddress (set device communication address)

int _stdcall ad_setaddress(HANDLE m_handle, int m_iAddress , int idata)

Function: Set device communication

address Parameters: m_handle: communication device

identifier m_iAddress: device current communication

address idata: device communication address to be set

Return: Return 0 successfully

4.1.6.ad_getconfig (get the basic parameters of the device)

int _stdcall ad_getconfig(HANDLE m_handle, int m_iAddress , unsigned char *oData, unsigned char *oSize)

Function: Get the basic parameters of the

device Parameters: m_handle: communication device identifier

m_iAddress: current communication address

oData: return the basic parameter pointer address (see the basic parameter

table) oSize: return the basic parameter byte number pointer address

Return: Return 0 successfully

4.1.7.ad_setconfig (set the basic parameters of the device)

int _stdcall ad_setconfig (HANDLE m_handle, int m_iAddress , unsigned char *iData, unsigned char iSize)

Function: Set the basic parameters of the

device Parameters: m_handle: communication device identifier

m_iAddress: current communication address

iData: pointer address of basic parameters to be set (see basic parameter table)

iSize: number of bytes of basic parameters to be set

Return: Return 0 successfully

4.1.8.ad_getoutstatus (get output mode parameters)

int _stdcall ad_getoutstatus (HANDLE m_handle, int m_iAddress , unsigned char *oData, unsigned char *oSize)

Function: Get device network communication

parameters Parameters: m_handle: communication device

identifier m_iAddress: current communication

address oData: return parameter pointer address (see output mode parameter

table) oSize: return parameter byte number pointer address

Return: Return 0 successfully

4.1.9.ad_setoutstatus (set output mode parameters)

int _stdcall ad_setoutstatus (HANDLE m_handle, int m_iAddress , unsigned char *iData, unsigned char iSize)

Function: Set device network communication

parameters Parameters: m_handle: communication device

identifier m_iAddress: current communication

address iData: pointer address of parameter to be set (see output mode parameter

table) iSize: number of bytes of parameter to be set

Return: Return 0 successfully

4.2.ISO18000-6B protocol label operation function

4.2.1.ad_identify6b (identify IS0180000-6B protocol label)

int _stdcall ad_identify6b(HANDLE m_handle, int m_iAddress, unsigned char *oData, unsigned char *oSize)

Function: Identify IS0180000-6B protocol label Parameters:

m_handle: communication device identifier m_iAddress:

current communication address oData: return

data pointer address oSize: return data byte

number pointer address

Return: Return 0 successfully

4.2.2.ad_read6b (read the data in the IS0180000-6B protocol label)

int _stdcall ad_read6b (HANDLE m_handle, int m_iAddress , unsigned char *oData, unsigned char iAddr, unsigned char iSize)

Function: Read the data in the IS0180000-6B protocol label

Parameters: m_handle: communication device identifier
m_iAddress: current communication address oData:
return data pointer address iAddr: data address to be
obtained oSize: return data byte number pointer
address

Return: Return 0 successfully

4.2.3.ad_write6b (write data to IS0180000-6B protocol tag)

int _stdcall ad_write6b (HANDLE m_handle, int m_iAddress , unsigned char *iData, unsigned char iAddr, unsigned char iSize)

Function: Write data to the IS0180000-6B protocol label Parameters:
m_handle: Communication device identifier m_iAddress: Current
communication address iData: Return data iAddr: Data
address to be written iSize: Number of data bytes to be
written

Return: Return 0 successfully

4.3. EPC (GEN 2) protocol label operation function

4.3.1.ad_identify6c (identify EPC (GEN 2) protocol label)

int _stdcall ad_identify6c (HANDLE m_handle, int m_iAddress, unsigned char *oData, unsigned char *oSize)

Function: Identify EPC (GEN 2) protocol label
Parameters: m_handle: communication device identifier
m_iAddress: current communication
address oData: return data pointer
address oSize: return data byte count pointer address

Return: Return 0 successfully

4.3.2.ad_read6c (read data in EPC(GEN 2) protocol tag)

int _stdcall ad_read6c (HANDLE m_handle, int m_iAddress , unsigned char *oData, unsigned char iMem, unsigned char iAddr, unsigned char iSize)

Function: Read the data in the EPC (GEN 2) protocol label
Parameters: m_handle: Communication device identifier
m_iAddress: Current communication address
oData: Return data pointer address iMem: Data
block address to be obtained iAddr: Data address
to be obtained oSize: Return data word Section
number

Return: Return 0 successfully

4.3.3.ad_write6c (write data to EPC(GEN 2) protocol tag)

int _stdcall ad_write6c (HANDLE m_handle, int m_iAddress , unsigned char *iData, unsigned char iMem, unsigned char iAddr, unsigned char iSize)

Function: Write data to the EPC (GEN 2) protocol tag Parameters:
m_handle: communication device identifier m_iAddress: current
communication address iData: pointer address of data
to be written iMem: address of data block to be written
iAddr: address of data to be written iSize: the number
of data bytes to be written

Return: Return 0 successfully

4.3.4.ad_encrypt (EPC(GEN 2) protocol label encryption)

int _stdcall ad_encrypt (HANDLE m_handle, int m_iAddress)

Function: EPC (GEN 2) protocol label encryption
parameters: m_handle: communication device identifier
m_iAddress: current communication address

Return: Return 0 successfully

5. Error code table

error code	error state	error code	error state	error code	error state

201 Failed to open 202		211 Failed to connect to USB 212 Failed		252 Communication address error	
Failed to get parameters 203		to enter programming mode 253 Incomplete protocol		213 Failed to exit programming	
Failed to set parameters 204 Failed		mode 256 Data conversion error 259 Incomplete data			
to set timeout 205 Failed to send					
data 206 Failed to receive data 207					
Failed to close 208 Failed to send					
timeout 209 Timeout to receive					

6. Appendix parameter table

6.1. Basic parameter table

parameter	Explain	The
Para1	the size of the power	reference value can adjust the distance from the card reader to read the tag Default value: 30 Reference value: (decimal format) 0~30 Can set the
Para2	frequency hopping enable	fixed frequency or frequency hopping mode Default value: 1 Reference value: (decimal format) 1-fixed frequency, 2,-Frequency hopping default value: 110 (915MHz) reference
Para3	Fixed frequency value	value: (decimal format) 0~200 (860MHz ~ 960MHz) default value: 84 (902MHz) reference value: (decimal format) 0~200 (860MHz ~ 960MHz) default Value: 93
Para4	Hop value 1	(906.5MHz) Reference value: (decimal format) 0~200 (860MHz ~ 960MHz) Default value: 102 (911MHz) Reference value: (decimal format) 0~200 (860MHz ~ 960MHz) Default value:
Para5	Hopping value 2	110 (915MHz ÿ Decimal format) 0~200(860MHz ~ 960MHz) Response mode: The card reader stops working, the host computer sends commands, the card reader works, and acts
Para6	Hopping value 3	according to the command; Active mode: The card reader works normally, and takes the initiative when it recognizes the tag Send data; Passive mode: the card reader works
Para7	Hop value 4	normally, and the tag is cached in the card reader when the tag is recognized, and the host computer sends a command to obtain the tag data; Default value: 2 Reference value:
Paragraph 8	Hopping value 5	(decimal format) 1-Response mode 2 -Active mode 3-Passive mode Default value: 10 (*1ms) Reference value: (decimal format) 5~255 (*1ms)
Para9	Hopping value 6	
Paragraph 10	Operating mode	
Paragraph 11	Timed sending interval	

Para12	External trigger mode	Default value: 0 Reference value: (decimal format) 0-off 2-low level active Default
Para13	output method	value: 1 Reference value: (decimal format) 1- RS232 2- RS485 3- TCPIP 4- CANBUS 5- SYRIA 6- Wiegand26 7- Wiegand34 Specific
Paragraph 14	Wiegand parameter 1 - data offset	reference Wiegand protocol default value: 0 Reference value: 0~20 Specific reference Wiegand protocol
Paragraph 15	Wiegand parameter 2 - output period	default value: 30 (* 10ms) Reference value: 0~255 (* 10ms) Specific reference Wiegand protocol default
money16	Wiegand Parameter 3 – Pulse Width	value: 10 (* 10us) Reference value: 0~255 (* 10us) Refer to the Wiegand protocol default value: 15 (* 100us)
Para17	Wiegand Parameter 4 - Pulse Period	Reference value: 0~255 (* 100us) One byte of data, the lower 4 bits represent 4 antennas, for example: use an antenna
Paragraph 18	antenna settings	1: 01H (binary 0000 0001) Using antenna 3: 04H (binary 0000 0100) Using antenna 1 and antenna 3: 05H (binary 0000 0101)
Para19	Card reader type	Default value: 16 Reference value: (decimal format) 1-ISO18000-6B single card 16-EPC(GEN 2) single card 17-EPC(GEN 2) + ISO18000-6B 32-EPC(GEN 2) Multicard 64-EPC(GEN 2)+Other partitions Default
Para20	Same ID output interval	value: 1s Reference value: (decimal format) 0~255s Default value: 1 Reference
Para21	buzzer	value: (decimal format) 0-Disable 1-Enable card reading category is [EPC(GEN 2)+ other partitions], this
Para22	Read other partition selections	parameter is selected for other partitions: Default value: 1 Reference value: (decimal format) 1-TID area (worldwide unique number area) 2-USER area (user-defined data area) When the card
Para23	Read other partition addresses	reading type is [EPC(GEN 2)+other partitions], this parameter is the starting address selection for other partition data acquisition: Default value: 0 Reference value: (decimal format) 0~31 When the card reading type is [EPC(GEN 2)+other partitions], this parameter is the data acquisition length selection for other partitions: Default value: 2
Para24	Read other partition lengths	Reference value: (decimal format) 1~12 Enable the encrypted reading of the card reader Card;
Para25	Encryption enabled	

		Default value: 0 Reference value: (decimal format) 0-common version, no encryption; 1-card reader encryption; Default
Para26 Para27	encrypted password	value: 0000 Reference value: (decimal format) 0000~9999 Example: Password 0123 (decimal) = 00H 7BH (Hexadecimal) Default value: 32 Reference value:
Couple28	Maximum number of cards read	(Decimal format) 10~64

6.2.TCPIP parameter table

parameter	illustrate	Default
Para1 Para2 Para3 Para4	IP address (4 bytes)	reference value: 192.168.5.105 Example: IP = 192.168.5.105 is expressed as C0 A8 05 69H
Para5 Para6 Para7 Paragraph 8	subnet mask (4 bytes)	The subnet mask is used to shield part of the IP address to distinguish network identification and host identification, and to indicate whether the IP address is on a local area network or a remote network. Default value: 255.255.255.0 Example: SubNet Mask = 255.255.255.0 means FF FF FF 00H Default value: 192.168.5.1 Example: Gateway = 192.168.5.1 means C0 A8 05 01H
Para9 Paragraph 10 Paragraph 11 Para12	default gateway (4 bytes)	
Para13 Paragraph 14	IP port	Default value: 49152 Example: IP Port = 49152 means C0 00H Default value:
Paragraph 15 money16 Para17 Paragraph 18 Para19 Para20	Physical address (6 bytes)	5E-45-A2-6C-30-1E Example: MAC = 5E-45-A2-6C-30-1E is expressed as 5E 45 A2 6C 30 1EH

6.3. Output mode parameter list

parameter	Describe	Data type
Para1	the output type	of reference value output Default value: 0 Reference value: (decimal format) 0-Decimal (1747988) 1-Hex (1AAC14) 2-Wiegand (02644052)
Para2	output digits	Fixed frequency or frequency hopping mode can be set. Default value: 8 Reference value: (decimal format) 8- output digits 8 digits (01747988)

		9-output digit 9 digits (001747988) 10- output digit 10 digits (0001747988) default
Para3	Whether to bring carriage return	value: 0 reference value: (decimal format) 0- without carriage return 1-with carriage return

7. **DELPHI** call function

//---RS232

function ap_open (nPort : Integer; nBaud : Integer): Integer; stdcall; external adpcom.dll'; function ap_close(m_hCom : Integer):bool; stdcall; external adpcom.dll'; function ap_getaddress(m_hCom : Thandle; oAddress : PDWORD; oVer : PDWORD) : integer; stdcall; external adpcom.dll'; //Get device communication address

function ap_setaddress(m_hCom : Thandle; iAddress : Integer; iData : Integer) : integer; stdcall; external adpcom.dll'; //Set device communication address

function ap_getconfig(m_hCom : Thandle; iAddress : Integer; oData : PDWORD; oSize : PDWORD) : integer; stdcall; external adpcom.dll'; //Get basic parameters function ap_setconfig(m_hCom : Thandle; iAddress : Integer; iSize : Integer) : integer; stdcall; external adpcom.dll'; //Set basic parameters

function ap_gettcpip(m_hCom : Thandle; iAddress : Integer; oData : PDWORD; oSize : PDWORD) : integer; stdcall; external adpcom.dll'; //ÿÿ TCPIP ÿÿ function ap_settcpip(m_hCom : Thandle; iAddress : Integer; iData : PDWORD; iSize : Integer) : integer; stdcall; external adpcom.dll'; //Set TCPIP parameters

function ap_identify6b(m_hCom : Thandle; iAddress : Integer; oData : PDWORD; oSize : PDWORD) : integer; stdcall; external adpcom.dll'; //Get 6B card number

function ap_read6b(m_hCom : Thandle; iAddress : Integer; oData : PDWORD; iAddr : Integer; iSize : Integer) : integer; stdcall; external adpcom.dll'; //Get 6B data

function ap_write6b(m_hCom : Thandle; iAddress : Integer; iData : PDWORD; iAddr : Integer; iSize : Integer) : integer; stdcall; external adpcom.dll'; //Set 6B data

function ap_identify6c(m_hCom : Thandle; iAddress : Integer; oData : PDWORD; oSize : PDWORD) : integer; stdcall; external adpcom.dll'; //Get 6c card number

function ap_identify6cmult(m_hCom : Thandle; iAddress : Integer; oData : PDWORD; oSize : PDWORD) : integer; stdcall; external adpcom.dll'; //Get 6c card number (multi-card) function ap_read6c(m_hCom : Thandle; iAddress : Integer; oData : PDWORD; iMem : Integer; iAddr : Integer; iSize : Integer) : integer; stdcall; external adpcom.dll'; //Get 6c data

function ap_write6c(m_hCom : Thandle; iAddress : Integer; iData : PDWORD; iMem : Integer; iAddr : Integer; iSize : Integer) : integer; stdcall; external adpcom.dll'; //Set 6c data

function ap_encrypt(m_hCom : Thandle; iAddress : Integer) : integer; stdcall; external adpcom.dll'; // encrypt 6c card

function ap_softreset(m_hCom : Thandle; iAddress : Integer) : integer; stdcall; external adpcom.dll'; // software reset device

function ap_getautocard(m_hCom : Thandle; oData : PDWORD; oSize : PDWORD) : integer; stdcall; external adpcom.dll'; //Get the card number sent automatically

//---TCPIP

function an_open(ip : string; port : Integer): Integer; stdcall; external adpnet.dll'; function

an_close(m_hCom : Integer):bool; stdcall; external adpnet.dll'; function

an_getaddress(m_hCom : Thandle; oAddress : PDWORD; oVer : PDWORD) : integer; stdcall;
external adpnet.dll'; //Get device communication address

function an_setaddress(m_hCom : Thandle; iAddress : Integer; iData : Integer) : integer; stdcall;
external 'adpnet.dll'; //Set device communication address

function an_getconfig(m_hCom : Thandle; iAddress : Integer; oData : PDWORD; oSize : PDWORD) :
integer; stdcall; external 'adpnet.dll'; //Get basic parameters

function an_setconfig(m_hCom : Thandle; iAddress : Integer; iData : PDWORD; iSize : Integer) :
integer; stdcall; external 'adpnet.dll'; //Set basic parameters

function an_gettcpip(m_hCom : Thandle; iAddress : Integer; oData : PDWORD; oSize : PDWORD) : integer; stdcall;
external 'adpnet.dll'; //ÿÿ TCPIP ÿÿ function an_settcpip(m_hCom : Thandle; iAddress : Integer; iData : PDWORD; iSize :
Integer) : integer;
stdcall; external 'adpnet.dll'; //Set TCPIP parameters

function an_identify6b(m_hCom : Thandle; iAddress : Integer; oData : PDWORD; oSize : PDWORD) :
integer; stdcall; external 'adpnet.dll'; //Get 6B card number

function an_read6b(m_hCom : Thandle; iAddress : Integer; oData : PDWORD; iAddr : Integer; iSize :
Integer) : integer; stdcall; external 'adpnet.dll'; //Get 6B data

function an_write6b(m_hCom : Thandle; iAddress : Integer; iData : PDWORD; iAddr : Integer; iSize :
Integer) : integer; stdcall; external 'adpnet.dll'; //Set 6B data

function an_identify6c(m_hCom : Thandle; iAddress : Integer; oData : PDWORD; oSize : PDWORD) :
integer; stdcall; external 'adpnet.dll'; //Get 6c card number

function an_identify6cmult(m_hCom : Thandle; iAddress : Integer; oData : PDWORD; oSize : PDWORD) : integer; ;
oData : PDWORD; iMem : Integer; iAddr :

Integer; iSize : Integer) : integer; stdcall; external 'adpnet.dll'; //Get 6c data

function an_write6c(m_hCom : Thandle; iAddress : Integer; iData : PDWORD; iMem : Integer; iAddr :
Integer; iSize : Integer) : integer; stdcall; external 'adpnet.dll'; //Set 6c data

function an_encrypt(m_hCom : Thandle; iAddress : Integer) : integer; stdcall; external 'adpnet.dll'; //ÿ ÿ 6c ÿ function
an_getautocard(m_hCom : Thandle; oData : PDWORD; oSize : PDWORD) : integer; stdcall;

external 'adpnet.dll'; //Get the card number sent automatically

//---USB

function ad_open(): Integer; stdcall; external 'adpusb.dll'; function

ad_close(m_hCom : Integer):bool; stdcall; external 'adpusb.dll'; function

ad_getaddress(m_hCom : Thandle; oAddress : PDWORD; oVer : PDWORD) : integer; stdcall;
external 'adpusb.dll'; //Get device communication address

function ad_setaddress(m_hCom : Thandle; iAddress : Integer; iData : Integer) : integer; stdcall;
external 'adpusb.dll'; //Set device communication address

function ad_getconfig(m_hCom : Thandle; iAddress : Integer; oData : PDWORD; oSize : PDWORD) : integer; stdcall;
external 'adpusb.dll'; //Get basic parameters

function ad_setconfig(m_hCom : Thandle; iAddress : Integer; iData : PDWORD; iSize : Integer) :
integer; stdcall; external 'adpusb.dll'; //Set basic parameters

function ad_getoutstatus(m_hCom : Thandle; iAddress : Integer; oData : PDWORD; oSize : PDWORD) : integer; PDWORD; iSize : Integer) : integer; stdcall; external 'adpusb.dll'; //Set output type parameter function ad_identify6b(m_hCom : Thandle; iAddress : Integer; oData : PDWORD; oSize : PDWORD) :

integer; stdcall; external 'adpusb.dll'; //Get 6B card number
function ad_read6b(m_hCom : Thandle; iAddress : Integer; oData : PDWORD; iAddr : Integer; iSize : Integer) : integer; stdcall; external 'adpusb.dll'; //Get 6B data
function ad_write6b(m_hCom : Thandle; iAddress : Integer; iData : PDWORD; iAddr : Integer; iSize : Integer) : integer; stdcall; external 'adpusb.dll'; //Set 6B data
function ad_identify6c(m_hCom : Thandle; iAddress : Integer; oData : PDWORD; oSize : PDWORD) : integer; stdcall; external 'adpusb.dll'; //Get 6c card number
function ad_read6c(m_hCom : Thandle; iAddress : Integer; oData : PDWORD; iMem : Integer; iAddr : Integer; iSize : Integer) : integer; stdcall; external 'adpusb.dll'; //Get 6c data
function ad_write6c(m_hCom : Thandle; iAddress : Integer; iData : PDWORD; iMem : Integer; iAddr : Integer; iSize : Integer) : integer; stdcall; external 'adpusb.dll'; //Set 6c data
function ad_encrypt(m_hCom : Thandle; iAddress : Integer) : integer; stdcall; external 'adpusb.dll'; //ÿ ÿ 6c ÿ function ad_getautocard(m_hCom : Thandle; oData : PDWORD; oSize : PDWORD) : integer; stdcall;

external 'adpusb.dll'; //Get the card number sent automatically

8. C#.NET call function

```
#region ---RS232---
[DllImport("adpcom.dll")]
public static extern int ap_open(int iPort, int iBaudrate); // ap_open - open serial port
[DllImport("adpcom.dll")] public static extern void ap_close(int iHandle); // ap_close - close serial
port [DllImport("adpcom.dll")] public static extern int ap_getaddress(int iHandle, ref int oAddress,
ref int oVer);// ap_getaddress - get address


[DllImport("adpcom.dll")]
public static extern int ap_setaddress(int iHandle, int iAddress, int iData);// ap_setaddress - ÿÿÿ
site
[DllImport("adpcom.dll")]
public static extern int ap_getconfig(int iHandle, int iAddress, IntPtr oData, ref byte oSize);//
ap_getconfig - get parameters
[DllImport("adpcom.dll")]
public static extern int ap_setconfig(int iHandle, int iAddress, byte[] iData, byte iSize);// ap_setconfig - set parameters
[DllImport("adpcom.dll")] public static extern int ap_gettcpip(int iHandle, int iAddress, IntPtr oData, ref byte oSize);// ap_gettcpip
- get tcpip parameters [DllImport("adpcom.dll")]
```

```
        public static extern int ap_settcpip(int iHandle, int iAddress, byte[] iData, byte iSize);// ap_settcpip - set tcpip
parameters [DllImport("adpcom.dll")] public static extern int ap_identify6b(int iHandle, int iAddress, IntPtr oData, ref byte
        oSize);// ap_identify6b - identify 6b card number


        [DllImport("adpcom.dll")]
        public static extern int ap_read6b(int iHandle, int iAddress, IntPtr oData, byte iAddr, byte iSize);//  ap_read6b - ÿ
6bÿÿ
        [DllImport("adpcom.dll")]
        public static extern int ap_write6b(int iHandle, int iAddress, byte[] iData, byte iAddr, byte iSize);// ap_write6b - ÿ
6bÿÿ
        [DllImport("adpcom.dll")]
        public static extern int ap_identify6c(int iHandle, int iAddress, IntPtr oData, ref byte oSize);// ap_identify6c
- identify 6C card number
        [DllImport("adpcom.dll")]
        public static extern int ap_identify6cmult(int iHandle, int iAddress, IntPtr oData, ref byte oSize);//
ap_identify6cmult - identify 6C card number - multi-card
        [DllImport("adpcom.dll")]
        public static extern int ap_read6c(int iHandle, int iAddress, IntPtr oData, byte iMem, byte iAddr, byte
iSize);// ap_read6c - read 6C data
        [DllImport("adpcom.dll")]
        public static extern int ap_write6c(int iHandle, int iAddress, byte[] iData, byte iMem, byte iAddr, byte
iSize);// ap_write6c - write 6C data
        [DllImport("adpcom.dll")]
        public static extern int ap_encrypt(int iHandle, int iAddress);// ap_encrypt - ÿÿ

        [DllImport("adpcom.dll")]
        public static extern int ap_softreset(int iHandle, int iAddress);// ap_softreset – soft reset device

        [DllImport("adpcom.dll")]
        public static extern int ap_getautocard(int iHandle, IntPtr oData, ref byte oSize);//ap_getautocard - Get the card
number sent automatically #endregion


        #region ---TCPIP---
        [DllImport("adpnet.dll")]
        public static extern int an_open(string ip, int port);
        [DllImport("adpnet.dll")]  public static extern int
        an_close(int iHandle);  [DllImport("adpnet.dll")]  public
        static extern int an_getaddress(int iHandle, ref int
        oAddress, ref int oVer);  [DllImport("adpnet.dll")]  public static extern int an_setaddress(int
        iHandle, int iAddress, int iData);  [DllImport("adpnet.dll")]  public static extern int
        an_getconfig(int iHandle, int iAddress, IntPtr oData, ref byte oSize);  [DllImport("adpnet.dll")]
```

```
public static extern int an_setconfig(int iHandle, int iAddress, byte[] iData, byte iSize);
[DllImport("adpnet.dll")] public static extern int an_gettcpip(int iHandle, int iAddress,
IntPtr oData, ref byte oSize); [DllImport("adpnet.dll")] public static extern int an_settcpip(int
iHandle, int iAddress, byte[] iData, byte iSize); [DllImport("adpnet.dll")] public static extern
int an_identify6b(int iHandle, int iAddress, IntPtr oData, ref byte oSize);
[DllImport("adpnet.dll")] public static extern int an_read6b(int iHandle, int iAddress, IntPtr
oData, byte iAddr, byte iSize); [DllImport("adpnet.dll")] public static extern int an_write6b(int
iHandle, int iAddress, byte[] iData, byte iAddr, byte iSize); [DllImport("adpnet.dll")] public
static extern int an_identify6c(int iHandle, int iAddress, IntPtr oData, ref byte oSize);
[DllImport("adpnet.dll")] public static extern int an_identify6cmult(int iHandle, int iAddress, IntPtr
oData, ref byte oSize); [DllImport("adpnet.dll")] public static extern int an_read6c(int iHandle, int
iAddress, IntPtr oData, byte iMem, byte iAddr, byte iSize); [DllImport("adpnet.dll")] public static
extern int an_write6c(int iHandle, int iAddress, byte[] iData, byte iMem, byte iAddr, byte iSize);
[DllImport("adpnet.dll")] public static extern int an_encrypt(int iHandle, int iAddress);
[DllImport("adpnet.dll")] public static extern int an_getautocard(int iHandle, IntPtr oData, ref byte
oSize); #endregion
```

```
#region ---USB---
[DllImport("adpusb.dll")]
public static extern int ad_open();
[DllImport("adpusb.dll")] public
static extern int ad_close(int iHandle);
[DllImport("adpusb.dll")] public static extern
int ad_exitprogram(int iHandle);
[DllImport("adpusb.dll")] public static extern int
ad_getaddress(int iHandle, ref int oAddress, ref int oVer); [DllImport("adpusb.dll")]
public static extern int ad_setaddress(int iHandle, int iAddress, int iData);
[DllImport("adpusb.dll")] public static extern int ad_getconfig(int iHandle, int
iAddress, IntPtr oData, ref byte oSize); [DllImport("adpusb.dll")] public static
extern int ad_setconfig(int iHandle, int iAddress, byte[] iData, byte iSize);
[DllImport("adpusb.dll")] public static extern int ad_getoutstatus(int iHandle, int iAddress,
IntPtr oData, ref byte oSize); [DllImport("adpusb.dll")] public static extern int
ad_setoutstatus(int iHandle, int iAddress, byte[] iData, byte iSize);
```

```
[DllImport("adpusb.dll")]
public static extern int ad_identify6b(int iHandle, int iAddress, IntPtr oData, ref byte oSize);
[DllImport("adpusb.dll")] public static extern int ad_read6b(int iHandle, int iAddress, IntPtr
oData, byte iAddr, byte iSize); [DllImport("adpusb.dll")] public static extern int ad_write6b(int
iHandle, int iAddress, byte[] iData, byte iAddr, byte iSize); [DllImport("adpusb.dll")] public static
extern int ad_identify6c(int iHandle, int iAddress, IntPtr oData, ref byte oSize);
[DllImport("adpusb.dll")] public static extern int ad_read6c(int iHandle, int iAddress, IntPtr oData,
byte iMem, byte iAddr, byte iSize); [DllImport("adpusb.dll")] public static extern int ad_write6c(int
iHandle, int iAddress, byte[] iData, byte iMem, byte iAddr, byte iSize); [DllImport("adpusb.dll")]
public static extern int ad_encrypt(int iHandle, int iAddress); [DllImport("adpusb.dll")] public static
extern int ad_getautocard(int iHandle, IntPtr oData, ref byte oSize); #endregion
```