Technical Documents

Technical file name: RFIDStation

API library documentation

Technical paper no:

Version: V1.2

Proposed ________________________

Audit ________________________

Signatures ________________________

Standardisation ________________________

Approvals ________________________

____

____

____

Approvals ________________________

Table of Contents

Chapter 1 Preface 5

1.1 Description of the structure 5

1.1.1 HFREADER_OPRESULT 5

1.1.2 HFREADER_CONFIG 6

1.1.3 HFREADER_IO 8

1.1.4 HFREADER_ACTIVEFRAME 10

1.1.5 HFREADER_VERSION 10

1.1.6 ISO15693_UIDPARAM 11

1.1.7 ISO15693_BLOCKPARAM 11

1.1.8 ISO15693_TAGPARAM 12

1.1.9 ISO15693_DTU 13

1.1.10 ISO14443A_UID 13

1.1.11 ISO14443A_UIDPARAM 14

1.1.12 ISO14443A_BLOCKPARAM 15

1.1.13 ISO14443A_VALUEPARAM 16

1.1.14 ISO14443A_OPPARAM 17

1.1.15 ISO14443A_DTU 18

1.1.16 ISO14443B_INFO 19

1.1.17 ISO14443B_DTU 20

1.1.18 FELICA_UIDPARAM 21

1.1.19 FELICA_DTU 22

1.1.20 ISO14443A_IMPARAM 23

1.1.21 ISO14443A_IMINFO 24

1.1.22 ISO15693_IMOP 25

1.1.23 ISO15693_GATEUIDPARAM 26

Chapter 2 Readers System Control API 27

2.1 Library function descriptions 27

2.1.1 hfReaderOpenPort 27

2.1.2 hfReaderClosePort 27

2.1.3 hfReaderSetConfig 28

2.1.4 hfReaderGetConfig 29

2.1.5 hfReaderDefaultConfig 30

2.1.6 hfReaderCtrlRf 31

2.1.7 hfReaderTrigger 32

2.1.8 hfReaderSetIo 33

2.1.9 hfReaderGetIo 34

2.1.10 hfReaderCfgIo 35

2.1.11 hfReaderOpenUsb 36

2.1.12 hfReaderCloseUsb 36

2.1.13 hfReaderGetVersion 37

2.1.14 hfReaderSelectAnt 38

2.1.15 hfReaderSetPower 39

2.1.16 hfReaderScanUsbList........40
2.1.17 hfReaderOpenSocket........40
2.1.18 hfReaderCloseSocket........40
Chapter 3 The iso15693 protocol tagging API........42
3.1 Library function descriptions........42
3.1.1 iso15693GetUid........42
3.1.2 iso15693ReadBlock........43
3.1.3 iso15693WriteBlock........44
3.1.4 iso15693LockBlock........45
3.1.5 iso15693WriteAfi........46
3.1.6 iso15693LockAfi........47
3.1.7 iso15693WriteDsfid........48
3.1.8 iso15693LockDsfid........49
3.1.9 iso15693ReadTagInfo........50
3.1.10 iso15693SetEas........51
3.1.11 iso15693Dtu........52
3.1.12 iso15693OpTagsRUidAndBlock........53
3.1.13 iso15693ImOp........54
3.1.14 iso15693OpTagsWriteBlock........55
3.1.15 iso15693OpTagsWriteAfi........56
3.1.16 iso15693OpTagsReadAfi........57
3.1.17 iso15693GetGateAlarmUid........58
Chapter 4 iso14443A Protocol Tagging API........59
4.1 Library function descriptions........59
4.1.1 iso14443AGetUid........59
4.1.2 iso14443AAuthReadM1Block........60
4.1.3 iso14443AAuthWriteM1Block........61
4.1.4 iso14443AAuthReadM1Value........62
4.1.5 iso14443AAuthWriteM1Value........63
4.1.6 iso14443AAuthOpM1Value........64
4.1.7 iso14443AReadM0Block........65
4.1.8 iso14443AWriteM0Block........66
4.1.9 iso14443ARats........67
4.1.10 iso14443ACtrlEsam........68
4.1.11 iso14443AApdu........69
4.1.12 iso14443AHalt........70
4.1.13 iso14443ASDsel........71
4.1.14 iso14443ADtu........72
4.1.15 iso14443AReadM0Cnt........73
4.1.16 iso14443AReadM0Sig........74
4.1.17 iso14443AAuthM0........75
4.1.18 iso14443AAuthUltralightC........76
4.1.19 iso14443AReadTopazBlock........77
4.1.20 iso14443AWriteTopazBlock........78

4.1.21 iso14443AImOperation........79

Chapter 5 The iso14443B protocol tagging API........80

5.1 Library function descriptions........80

5.1.1 iso14443BSelect........80

5.1.2 iso14443BGetIDCardUid........81

5.1.3 iso14443BDtu........82

5.1.4 iso14443BHalt........83

Chapter 6 Felica Protocol Label Manipulation API........84

6.1 Library function descriptions........84

6.1.1 felicaGetUid........84

6.1.2 felicaDtu........85

Chapter 1 Preface

The HFReader.dll file is used for secondary development of the HF reader system software.

HFReader.dll is compiled in the standard C way (extern "C"). Because of the number of parameters involved in the interface functions, HFReader.dll uses structures to pass parameters. The structures involved in the system are described in detail below.

1.1 Description of the structure

Note: **The parameters inside the structure are in 4-byte alignment mode.**

1.1.1 HFREADER_OPRESULT

This structure defines the basic information of the response frame of the operation reader and the structure is defined as follows.

```
typedef struct
        hfReaderOpResult{ UINT
        srcAddr;
        UINT targetAddr;
        UINT flag;
        UINT errType;
        UINT t;
}HFREADER_OPRESULT;
```

The specific parameters are described in Table 1.1 below.

Table 1.1 Description of the HFREADER_OPRESULT structure parameters

Parameters	Detailed description	
srcAddr	The source address of the response frame, 0xFFFF is the broadcast address	
targetAddr	The destination address of the response frame, 0xFFFF is the broadcast address	
flag	Operation results	0：Successful operation
		1：Operation failed
errType	Type of error	0：No errors
		1：Label detected error
		2：Label Response Frame CRC Checksum Error
		3：Label not responding
		4：Parameter error
t	Response time in ms	

1.1.2 HFREADER_CONFIG

This structure defines the parameters of the reader operation, which are described in Table 1.2 below. The structure is defined as follows.

```
typedef struct hfReaderConfig{
        HFREADER_OPRESULT result;
        UINT workMode;
        UINT readerAddr;
        UINT cmdMode;
        UINT afiCtrl;
        UINT uidSendMode;
        UINT tagStatus;
        UINT baudrate;
        UINT beepStatus;
        UINT afi;
    }HFREADER_CONFIG;
```

Table 1.2 Description of the HFREADER_CONFIG structure parameters

Parameters	Detailed description	
result	Basic information about the operating parameters response frame of the operation reader	
workMode	Working mode	0x01：EAS mode for ISO15693, sending EAS commands 0x00：Inventory mode of ISO15693, sending inventory commands 0x10：ISO14443A 0x20：ISO14443B 0x30：Felica
readerAddr	Reader address	0x0000~0xFFFF：0xFFFF Broadcast address
cmdMode	Command mode	0：Auto mode, automatically sends commands to obtain UID
		1：Trigger mode, get UID by command trigger
afiCtrl	afi control	0：Disable AFI, inventory requests do not carry AFI parameters
		1：Enables AFI, inventory requests carry AFI parameters
uidSendMode	UID Sending mode	0：Active mode, active upload after UID acquisition
		1：Passive mode, UID stored in buffer, waiting for read UID command to be sent
tagStatus	Silent control	0：Silent enable, silent/hang up tag after successful inventory
		1：Silent disable, no silent/hang up tag after a successful inventory
baudrate	Baud rate	5：9600bps
		7：38400bps
		11：115200bps
beepStatus	Buzzer control	0：Disable, buzzer muted after successful inventory
		1：Enable, buzzer ’beep’ after a successful inventory
afi	0x00~0xFF	

1.1.3 HFREADER_IO

This structure defines the operating state of the reader IO port and has the following structure definition：

```
typedef struct
    hfReaderIo{ HFREADER_OPRE
    SULTresult;
    UINT out1State;
    UINT out1Frequent;
    UINT out1Cycle;
    UINT out2State;
    UINT out2Frequent;
    UINT out2Cycle;
    UINT relayState;
    UINT relayFrequent;
    UINT relayCycle;
    UINT in1;
    UINT in2;
    UINT in3;
    UINT in4;
}HFREADER_IO;
```

The specific parameters are described in Table 1.3 below.

Table 1.3 Description of the parameters of the HFREADER_IO structure

Parameters	Detailed description	
result	Basic information on the response frame of the IO port of the operation reader	
out1State	out1 Status	0：Low level
		1：High level
		2：No change in status
out1Frequent	out1 Output frequency	1：0.5Hz
		2：1Hz
		3：2Hz
		4：5Hz
		5：10Hz
out1Cycle	out1 Output cycle	0x00~0xFF
out2State	out2 Status	See out1 status notes
out2Frequent	out2 Output frequency	See out1 Output frequency description
out2Cycle	out2 output cycle	See out1 Output cycle description
relayState	Relay status	See out1 status notes
relayFrequent	Relay output frequency	See out1 Output frequency description
relayCycle	Relay output cycle	See out1 Output cycle description
in1	in1 Occurrence time	Calculated from power on， in 50ms
in2	in2 Occurrence time	Calculated from power on， in 50ms
in3	in3 Occurrence time	Calculated from power on， in 50ms
in4	in4 Occurrence time	Calculated from power on， in 50ms

1.1.4 HFREADER_ACTIVEFRAME

This structure defines the structure of a frame received as an unsolicited upload from the reader, the structure is defined as follows:

```
#define   HFREADER_FRAME_MAX_NUM            64
#define 255                     255
typedef struct
     hfReaderActiveFrame{ UINT
     num;
     UCHAR    uid[HFREADER_FRAME_MAX_NUM][255];
     UCHAR    frame[HFREADER_FRAME_MAX_NUM][255];
}HFREADER_ACTIVEFRAME;
```

The specific parameters are described in Table 1.4 below.

Table 1.4 Description of the parameters of the HFREADER_ACTIVEFRAME structure

Parameters	Detailed description	
num	Number of frames, up to 64 data frames	
uid	UID of the tag	Readers actively upload UIDs
frame	Data frames	Communication data frames

1.1.5 HFREADER_VERSION

This structure defines the received reader version information structure, the structure is defined as follows:

```
#define HFREADER_VERSION_SIZE       50
typedef struct
     hfReaderVersion{ HFREADER_
     OPRESULTresult;
     char   type[HFREADER_VERSION_SIZE];
     char   sv[HFREADER_VERSION_SIZE];
     char   hv[HFREADER_VERSION_SIZE];
}HFREADER_VERSION;
```

The specific parameters are described in Table 1.5 below.

Table 1.5 Description of the HFREADER_VERSION structure parameters

Parameters	Detailed description
result	Get basic information about the version's response frame
type	Reader models
sv	Software Versions
hv	Hardware versions

1.1.6 ISO15693_UIDPARAM

This structure defines the tag UID for obtaining the ISO15693 protocol and the structure is defined as follows:

```
    typedef struct iso15693UidParam{
        HFREADER_OPRESULTresult;
        UCHAR uid[25][8];
        UINT remainNum;
        UINT num;
    }ISO15693_UIDPARAM;
```

The specific parameters are described in Table 1.6 below.

Table 1.6 Description of the ISO15693_UIDPARAM structure parameters

Parameters	Detailed description
result	Get the basic information of the UID response frame
uid	UID of the tag, up to 25 UIDs can be received
remainNum	The number of UIDs remaining in the reader's cache
num	Number of UIDs read

1.1.7 ISO15693_BLOCKPARAM

This structure defines the result of manipulating the ISO15693 tag data block and the structure is defined as follows:

```
    typedef struct iso15693BlockParam{
        HFREADER_OPRESULTresult;
        UCHAR block[32]4];
        UINT addr;
        UINT num;
    }ISO15693_BLOCKPARAM;
```

The specific parameters are described in Table 1.7 below.

Table 1.7 Description of the parameters of the ISO15693_BLOCKPARAM structure

Parameters	Detailed description
result	Basic information about the response frame of the fuck data block
block	Data block data, up to 32 data blocks supported
addr	Data Block Header Address
num	Number of data blocks

1.1.8 ISO15693_TAGPARAM

This structure defines the result of obtaining ISO15693 label information and the structure is defined as follows:

```
typedef struct iso15693TagInfoParam{
        HFREADER_OPRESULTresult;
        UINT infoFlag;
        UINT dsfid;
        UINT afi;
        UINT blockNum;
        UINT  blockSize;
        UINT ic;
    }ISO15693_TAGPARAM;
```

The specific parameters are described in Table 1.8 below.

Table 1.8 Description of the parameters of the ISO15693_TAGPARAM structure

Parameters	Detailed description
result	Get the basic information of the tag information response frame
infoFlag	Information signs
dsfid	DSFID parameter of the label
afi	AFI parameters for labels
blockNum	Number of data blocks contained in the label
blockSize	The size of the data block contained in the label
ic	ic parameters of the label

1.1.9 ISO15693_DTU

This structure defines the result of the pass-through operation ISO15693 tag and the structure is defined as follows:

```
    typedef struct iso15693DtuParam{
        HFREADER_OPRESULT result;
        UINT txLen;
        UCHAR txFrame[255];
        UINT rxLen;
        UCHAR rxFrame[255];
        UINT timeout;
    }ISO15693_DTU;
```

The specific parameters are described in Table 1.9 below.

Table 1.9 Description of the parameters of the ISO15693_DTU structure

Parameters	Detailed description
result	Basic information on the response frame of the pass-through operation tag
txLen	Label request frame length
txFrame	Label request frame (without CRC)
rxLen	Label response frame length
rxFrame	Label response frames (without CRC)
timeout	Timeout time of the operation in us

1.1.10 ISO14443A_UID

This structure defines the ISO14443A tag UID and the structure is defined as follows:

```
    typedef struct iso14443aUid{
        UINT type;
        UINT len;
        UINT sak;
        UCHAR uid[10];
    }ISO14443A_UID;
```

The specific parameters are described in Table 1.10 below.

Table 1.10 Description of the ISO14443A_UID structure parameters

Parameters	Detailed description
type	Label type, 2 bytes
len	UID Length
uid	uid

sak	Label selection response

1.1.11 ISO14443A_UIDPARAM

This structure defines the tag UID for obtaining the ISO14443A protocol and the structure is defined as follows：

```
typedef struct iso14443aUidParam{
            HFREADER_OPRESULT result;
            ISO14443A_UID uid[15];
            UINT remainNum;
            UINT num;
      }ISO14443A_UIDPARAM;
```

The specific parameters are described in Table 1.11 below.

Parameters	Detailed description
result	Get the basic information of the UID response frame
uid	UID of the tag, up to 15 UIDs can be received
remainNum	The number of UIDs remaining in the reader's cache
num	Number of UIDs read

1.1.12 ISO14443A_BLOCKPARAM

This structure defines the result of manipulating the ISO14443A tag data block and the structure is defined as follows.

```
typedef struct iso14443ABlockParam{
    HFREADER_OPRESULT result;
    ISO14443A_UID uid;
    UINT keyType;
    UCHAR key[6];
    UCHAR block[16 * 13];
    UINT addr;
    UINT num;
}ISO14443A_BLOCKPARAM;
```

The specific parameters are described in Table 1.12 below.

Table 1.12 Description of the parameters of the ISO14443A_BLOCKPARAM structure

Parameters	Detailed description
result	Basic information about the response frame of the operation data block
uid	The tag UID, if uid.len is equal to 0, means that the tag does not need to be reselected before manipulating the data block. tag before manipulating the data block, generally used in trigger mode.
keyType	Key type, if equal to 0, means no password is required to manipulate the data block
key	Key
block	Data blocks, up to 13 M1 data blocks or 52 M0 data blocks
addr	Data Block Header Address
num	Number of data blocks

1.1.13 ISO14443A_VALUEPARAM

This structure defines the result of manipulating the ISO14443A tag value and the structure is defined as follows.

```
typedef struct iso14443AValueParam{
        HFREADER_OPRESULT result;
        ISO14443A_UID uid;
        UINT keyType;
        UCHAR key[6];
        UINT opCode;
        UINT blockAddr;
        UINT transAddr;
        int value;
    }ISO14443A_VALUEPARAM;
```

The specific parameters are described in Table 1.13 below.

Table 1.13 Description of the parameters of the ISO14443A_VALUEPARAM structure

Parameters	Detailed description
result	Basic information about the operation value response frame
uid	The tag UID, if uid.len is equal to 0, means that the tag does not need to be reselected before manipulating the data block. tag before manipulating the data block, generally used in trigger mode.
keyType	Key type, if equal to 0, means no password is required to manipulate the data block
key	Key
block	Data blocks, up to 13 M1 data blocks or 52 M0 data blocks
opCode	Opcodes, value added 0xC1, value subtracted 0xC0 and dumped 0xC2
blockAddr	Data block address
transAddr	Transfer address
value	Value

1.1.14 ISO14443A_OPPARAM

This structure defines the result of the operation of the ISO14443A label vendor custom command and is defined as follows:

```
  typedef struct iso14443AOperationParam{
      HFREADER_OPRESULT result;
      ISO14443A_UID uid;
      UINT keyType;
      UCHAR key[6];
      UCHAR txFrame[255];
      UINT txLen;
      UCHAR rxFrame[255];
      UINT rxLen;
}ISO14443A_OPPARAM;
```

The specific parameters are described in Table 1.14 below.

Table 1.14 Description of the parameters of the ISO14443A_OPPARAM structure

Parameters	Detailed description
result	Basic information about the operation tag response frame
uid	The tag UID, if uid.len is equal to 0, means that the tag does not need to be reselected before manipulating the data block. tag before manipulating the data block, generally used in trigger mode.
keyType	Key type, if equal to 0, means no password is required to manipulate the data block
key	Key
txFrame	Operation command request frame
txLen	Request frame length
rxFrame	Operation command result parameters
rxLen	Parameter length

1.1.15 ISO14443A_DTU

This structure defines the result of the pass-through operation ISO14443A tag, and the structure is defined as follows：

```
typedef struct iso14443ADtuParam{
        HFREADER_OPRESULT result;
        ISO14443A_UID uid;
        UINT txBit;
        UINT txLen;
        UCHAR txFrame[255];
        UINT rxBit;
        UINT rxLen;
        UCHAR rxFrame[255];
        UINT timeout;
    }ISO14443A_DTU;
```

The specific parameters are described in Table 1.15 below.

Table 1.15 Description of the parameters of the ISO14443A_DTU structure

Parameters	Detailed description
result	Basic information on the response frame of the pass-through operation tag
uid	The tag UID, if uid.len is equal to 0, means that it is not necessary to Re-select the tag, generally used in trigger mode.
txBit	The number of bits sent in the last byte of the label request frame, 0 means all are sent
txLen	Label request frame length
txFrame	Tag request frame
rxBit	The number of bits in the last byte of the label response frame, 0 means all valid
rxLen	Label response frame length
rxFrame	Tag response frames
timeout	Timeout time of the operation in us

1.1.16 ISO14443B_INFO

This structure defines the result of checking the ISO14443B label and the structure is defined as follows:

```
#define    HFREADER_ISO14443B_MAX_PUPI_SIZE             0x04
#define    HFREADER_ISO14443B_MAX_APPLI_SIZE            0x04
#define    HFREADER_ISO14443B_MAX_PROTOCOL_SIZE    0x04
typedef struct
      iso14443bInfo{ HFREADER_OP
      RESULTresult;
      UCHAR     pupi[HFREADER_ISO14443B_MAX_PUPI_SIZE];
      UCHAR     appField[HFREADER_ISO14443B_MAX_APPLI_SIZE];
      UCHAR     protocol[HFREADER_ISO14443B_MAX_PROTOCOL_SIZE];
}ISO14443B_INFO;
```

The specific parameters are described in Table 1.16 below.

Table 1.16 Description of the parameters of the ISO14443B_INFO structure

Parameters	Detailed description
result	Basic information on the response frame of the pass-through operation tag
pupi	Temporary ID
appField	Application information parameters
protocol	Protocol-related parameters

1.1.17 ISO14443B_DTU

This structure defines the result of the pass-through operation ISO14443B tag and is defined as follows:

```
    typedef struct iso14443BDtuParam{
        HFREADER_OPRESULT result;
        UINT txLen;
        UCHAR txFrame[255];
        UINT rxLen;
        UCHAR rxFrame[255];
        UINT timeout;
  }ISO14443B_DTU;
```

The specific parameters are described in Table 1.17 below.

Table 1.17 Description of the parameters of the ISO14443B_DTU structure

Parameters	Detailed description
result	Basic information on the response frame of the pass-through operation tag
txLen	Label request frame length
txFrame	Tag request frame
rxLen	Label response frame length
rxFrame	Tag response frames
timeout	Timeout time of the operation in us

1.1.18 FELICA_UIDPARAM

This structure defines the tag UID for obtaining the Felica protocol and the structure is defined as follows.

```
typedef struct felicaUidParam{
    HFREADER_OPRESULT result;
    UCHAR uid[10 * 16];
    UINT remainNum;
    UINT num;
}FELICA_UIDPARAM;
```

The specific parameters are described in Table 1.18 below.

Table 1.18 Description of the parameters of the FELICA_UIDPARAM structure

Parameters	Detailed description
result	Get the basic information of the UID response frame
uid	UID of the tag (16 bytes long), up to 10 UIDs can be received
remainNum	The number of UIDs remaining in the reader's cache
num	Number of UIDs read

1.1.19 FELICA_DTU

This structure defines the result of the pass-through operation Felica tag and the structure is defined as follows.

```
    typedef struct felicaDtuParam{
        HFREADER_OPRESULT result;
        UINT txLen;
        UCHAR txFrame[255];
        UINT rxLen;
        UCHAR rxFrame[255];
        UINT timeout;
   }FELICA_DTU;
```

The specific parameters are described in Table 1.19 below.

Table 1.19 Description of the parameters of the FELICA_DTU structure

Parameters	Detailed description
result	Basic information on the response frame of the pass-through operation tag
txLen	Label request frame length
txFrame	Tag request frame
rxLen	Label response frame length
rxFrame	Tag response frames
timeout	Timeout time of the operation in us

1.1.20 ISO14443A_IMPARAM

This structure defines the result of manipulating the ISO14443A tag intelligent manipulation data block and the structure is defined as follows:

```
typedef struct iso14443ABlockParam{
    R343_OPRESULT result;
    ISO14443A_UID uid;
    UINT antNum;
    UINT antWorkTime;
    UINT opMode;
    UINT opTimeout;
    UINT blockAddr;
    UINT blockNum;
    UCHAR block[1024];
}ISO14443A_IMPARAM;
```

The specific parameters are described in Table 1.20 below.

Table 1.20 Description of the parameters of the ISO14443A_IMPARAM structure

Parameters	Detailed description
result	Basic information about the response frame of the operation data block
uid	The tag UID, if uid.len is equal to 0, means that the tag does not need to be reselected before manipulating the data block.
antNum	Number of antennas, up to 4 supported
antWorkTime	Operating time per antenna, in ms
opMode	Operation modes: 0-Read only UID; 1-Read data block; 2-Write data block
opTimeout	Operation timeout time (t > antNum * antWorkTime), in ms
blockAddr	Data Block Header Address
blockNum	Number of data blocks
block	Data blocks

1.1.21 ISO14443A_IMINFO

This structure defines the operation information and antenna information during the intelligent operation of the ISO14443A tag and is defined as follows:

```
typedef struct
    iso14443AImOpInfo{ int
    opMode;
    int blockIndex;
    int uidLen;
    int tempr;
    int paTempr;
    int noise;
    int signal;
    int sofLv;
    int antIndex;
}ISO14443A_IMINFO;
```

The specific parameters are described in Table 1.21 below.

Table 1.21 Description of the parameters of the ISO14443A_IMINFO structure

Parameters	Detailed description
opMode	Current operation mode: 0-Read only UID; 1-Read data block; 2-Write data block
blockIndex	Number of label data blocks completed by the current operation
uidLen	UID length of the current tag
tempr	Air temperature
paTempr	Power amplifier temperature
noise	Noise signal strength in mv
signal	Label response signal strength in mv
sofLv	Noise signal discrete values
antIndex	Antenna Index

1.1.22 ISO15693_IMOP

This structure defines the operation information and antenna information during the intelligent operation of the ISO15693 tag as follows:

```
typedef struct
        iso15693ImParam{ HFREADER_OPRESULT
        result;
        UINT antNum;
        UCHAR antAddr[255];
        UINT mode;
        UINT tagNum;
        UCHAR
uid[HFREADER_ISO15693_IM_UID_NUM][HFREADER_ISO15693_SIZE_UID];
        UINT blockNum;
        UCHAR    blockAddr[HFREADER_ISO15693_IM_BLOCK_NUM];
        UCHAR    opBlockResult[HFREADER_ISO15693_IM_UID_NUM];
        UCHAR
block[HFREADER_ISO15693_IM_UID_NUM][HFREADER_ISO15693_IM_BLOCK_NUM][ HFR
EADER_ISO15693_SIZE_BLOCK];
        UINT timeout;
}ISO15693_IMOP;
```

The specific parameters are described in Table 1.22 below.

Table 1.22 Description of the parameters of the ISO15693_IMOP structure

Parameters	Detailed description
result	Basic information about IM operation response frames
antNum	Number of scanning antennas
antAddr	Antenna Address Index
mode	0x01-Reading UIDs 0x03-Read UID and data block
blockIndex	Number of label data blocks completed by the current operation
tagNum	Number of labels
uid	Tags UID
blockNum	Number of data blocks
blockAddr	Data block address
opBlockResult	Result of manipulating data blocks
block	Data block data
timeout	Waiting for response timeout

1.1.23 ISO15693_GATEUIDPARAM

This structure defines the operation information and antenna information during the intelligent operation of the ISO15693 channel door tag:

```
typedef struct iso15693GateParam{
        UCHAR uid[256 * HFREADER_ISO15693_SIZE_UID];
        UINT rssi[256];
        UINT ant;
        UINT num;
}ISO15693_GATEUIDPARAM;
```

The specific parameters are described in Table 1.23 below.

Table 1.22 Description of the ISO15693_GATEUIDPARAM structure parameters

Parameters	Detailed description
uid	Tags UID
rssi	Signal strength
ant	Antenna Address
num	Number of labels

Chapter 2 Readers System Control API

The Reader System Control API is responsible for interface communication, device parameter configuration, function control and IO control. A detailed functional description is given below.

2.1 Library function descriptions

2.1.1 hfReaderOpenPort

hfReaderOpenPort() opens the serial device of the reader system and returns a handle to the serial device. The details of the function are shown in Table 2.1 below：

Table 2.1 Function hfReaderOpenPort()

Function	Open the serial device	
Prototype	HANDLE hfReaderOpenPort(char *pPortName, char *pBaudrate)	
Parameters	pPortName（Input）	Serial port number, string type
	pBaudrate（Input）	Baud rate, string type Support 9600bps/38400 bps (default) /115200 bps
Back	HANDLE	-1: Failed to open the serial device Non-1: Serial device handle
Example	HANDLE hSerial = hfReaderOpenPort("COM1", "38400");	

2.1.2 hfReaderClosePort

hfReaderClosePort() is to close the serial port device of the reader system. A detailed description of the function is shown in Table 2.2 below.

Table 2.2 Function hfReaderClosePort()

Function	Close the serial device according to the specified serial device handle	
Prototype	void hfReaderClosePort(HANDLE hSerial)	
Parameters	hSerial（Input）	Serial device handles that have been opened
Back	void	No return value
Example	hfReaderClosePort(hSerial);	

2.1.3 hfReaderSetConfig

hfReaderSetConfig()is to configure the operating parameters of the reader according to the device type. The function is described in detail in Table 2.3 below shown.

Table 2.3 Functions hfReaderSetConfig()

Function	Configure reader operating parameters	
Prototype	int hfReaderSetConfig(HANDLE h, USHORT srcAddr, USHORT targetAddr, HFREADER_CONFIG *pConfig, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	pConfig（Input and output）	Input: configuration parameters Output: Operation Result For details see
	pTxFrame（Output）	
	pRxFrame（Output）	Received response frames
Back	int	0：No response frame
		>0：Response frame length
Example	rlt = hfReaderSetConfig(h, 0x000, 0x0001, pConfig, NULL, NULL);	

2.1.4 hfReaderGetConfig

hfReaderGetConfig()is to get the working parameters of the reader. The functions are described in detail in Table 2.4 below:

Table 2.4 Function hfReaderGetConfig()

Function	Obtaining reader operating parameters	
Prototype	int hfReaderGetConfig(HANDLE h, USHORT srcAddr, USHORT targetAddr, HFREADER_CONFIG *pConfig, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	pConfig（Output）	Output: Configuration parameters, and the result of the operation See 1.1.2 Description for details
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	0：No response frame
		>0：Response frame length
Example	rlt = hfReaderGetConfig(h, 0x000, 0x0001, pConfig, NULL, NULL);	

2.1.5 hfReaderDefaultConfig

hfReaderDefaultConfig()Is to set the reader to the default operating parameters. The function is described in detail in Table 2.5 below shows：

Table 2.5 Function hfReaderDefaultConfig()

Function	Setting the reader's default operating parameters	
Prototype	int hfReaderDefaultConfig(HANDLE h, USHORT srcAddr, USHORT targetAddr, HFREADER_CONFIG *pConfig, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	pConfig（Input and output）	Input: Configuration parameters, see the corresponding instruction manual for the reader for the default parameters Output: Result of the operation See 1.1.2 Description
	pTxFrame（output）	Request frames sent
	pRxFrame（output）	Received response frames
Back	int	0：No response frame
		>0：Response frame length
Example	rlt = hfReaderDefaultConfig(h, 0x000, 0x0001, pConfig, NULL, NULL);	

2.1.6 hfReaderCtrlRf

hfReaderCtrlRf()is the RF signal that controls the reader. A detailed description of the function is shown in Table 2.6 below：

Table 2.6 Functions hfReaderCtrlRf()

Function	Control of reader RF signals	
Prototype	int hfReaderCtrlRf(HANDLE h, USHORT srcAddr, USHORT targetAddr, UCHAR rfCtrl, HFREADER_OPRESULT *pResult, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	rfCtrl（Input）	0：Turning off the RF signal
		1：Turn on the RF signal
		2: Reset RF signal (off 20ms, then on again)
	pResult（Output）	Output: Result of the operation See 1.1.1 Description for details
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	0：No response frame
		>0：Response frame length
Example	rlt = hfReaderCtrlRf (h, 0x000, 0x0001, 0x00, pResult, NULL, NULL);	

2.1.7 hfReaderTrigger

hfReaderTrigger()is to trigger the reader to fire the inventory command to select the tag. A detailed description of the function is shown in Table 2.7 below.

Table 2.7 Function hfReaderTrigger()

Function	Trigger readers	
Prototype	int hfReaderTrigger(HANDLE h, USHORT srcAddr, USHORT targetAddr, UCHAR triggerCtrl, HFREADER_OPRESULT *pResult, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	triggerCtrl（Input）	0：Invalid
		1：Trigger Inventory
	pResult（Output）	Output: Result of the operation See 1.1.1 Description
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	0：No response frame
		>0：Response frame length
Example	rlt = hfReaderTrigger(h, 0x000, 0x0001, 0x01, pResult, NULL, NULL);	

2.1.8 hfReaderSetIo

hfReaderSetIo()is to control the IO port of the reader to output high and low levels. The functions are described in detail in Table 2.8 below:

Table 2.8 Function hfReaderSetIo()

Function	Control the IO port of the reader to output high and low levels	
Prototype	int hfReaderSetIo(HANDLE h, USHORT srcAddr, USHORT targetAddr, UCHAR *pOutState, HFREADER_OPRESULT *pResult, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	pOutState（Input）	Input: out port level, 8 bytes control 8 output pins 0：low level 1：high level Other: unchanged
	pResult（Output）	Output: Result of the operation See 1.1.1 Description for details
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	0：No response frame
		>0：Response frame length
Example	rlt = hfReaderSetIo(h, 0x000, 0x0001, 0x04, pResult, NULL, NULL);	

2.1.9 hfReaderGetIo

hfReaderGetIo()is to get the status of the reader IO port input pins. A detailed description of the function is shown in Table 2.9 below:

Table 2.9 Function hfReaderGetIo()

Function	Get the time of the reader IO port input event	
Prototype	int hfReaderGetIo(HANDLE h, USHORT srcAddr, USHORT targetAddr, UCHAR *pInState, HFREADER_OPRESULT *pResult, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	pInState（Output）	Output: IN port level, 8 bytes for 8 input pins 0：low level 1：High level
	pResult（Output）	Output: Result of the operation See 1.1.1 Description for details
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	0：No response frame
		>0：Response frame length
Example	rlt = hfReaderGetIo(h, 0x000, 0x0001, pIo, NULL, NULL);	

2.1.10 hfReaderCfgIo

hfReaderCfgIo()is to configure the frequency and number of cycles of the output pulses from the reader IO port. The functions are described in detail in Table 2.10 below shown in the following table：

Table 2.10 Functions hfReaderCfgIo()

Function	Configuration of the reader IO port output pulse frequency and number of cycles	
Prototype	int hfReaderCfgIo(HANDLE h, USHORT srcAddr, USHORT targetAddr, HFREADER_IO *pIo, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	triggerCtrl（Input）	0：Invalid
		1：Trigger Inventory
	pIo（Input and output）	Input: out port pulse frequency and number of cycles Output: Result of the operation See 1.1.3 Description for details
	pTxFrame（output）	Request frames sent
	pRxFrame（output）	Received response frames
Back	int	0：No response frame
		>0：Response frame length
Example	rlt = hfReaderCfgIo(h, 0x000, 0x0001, pIo, NULL, NULL);	

2.1.11 hfReaderOpenUsb

hfReaderOpenUsb()is to open the USB interface device of the reader system and return the device handle. A detailed description of the function is shown in Table 2.11 below：

Table 2.11 Function hfReaderOpenUsb()

Function	Turning on USB devices	
Prototype	HANDLE hfReaderOpenUsb(DWPRD vid, DWORD pid)	
Parameters	vid（Input）	0x0505
	pid（Input）	0x5050
Back	HANDLE	-1: Failed to open the device Non-1: Device handle
Example	HANDLE hSerial = hfReaderOpenUsb(0x0505, 0x5050);	

2.1.12 hfReaderCloseUsb

hfReaderCloseUsb() is to turn off the USB device of the reader system. A detailed description of the function is shown in Table 2.12 below：

Table 2.12 Function hfReaderCloseUsb()

Function	Shut down the device according to the specified USB device handle	
Prototype	void hfReaderCloseUsb(HANDLE hSerial)	
Parameters	hSerial（Input）	USB device handles that have been opened
Back	void	No return value
Example	hfReaderCloseUsb(hSerial);	

2.1.13 hfReaderGetVersion

hfReaderGetVersion() is to get the reader version information. A detailed description of the function is shown in Table 2.13 below：

Table 2.13 Function hfReaderGetVersion()

Function	Get version information	
Prototype	int hfReaderCfgIo(HANDLE h, USHORT srcAddr, USHORT targetAddr, HFREADER_VERSION *pVersion, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	triggerCtrl（Input）	0: Invalid
		1: Trigger inventory
	pVersion（Output）	Version information See 1.1.5 Description for details
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	0: no response frames
		>0：Response frame length
Example	rlt = hfReaderGetVersion(h, 0x000, 0x0001, pVerion, NULL, NULL);	

2.1.14 hfReaderSelectAnt

hfReaderSelectAnt() is to select the current RF output antenna port, only 1 group of antennas can be controlled at a time, the group can support 6 groups of antennas. A detailed description of the function is shown in Table 2.14 below：

Table 2.14 Function hfReaderSelectAnt()

Function	Control of reader RF signals	
Prototype	int hfReaderSelectAnt (HANDLE h, USHORT srcAddr, USHORT targetAddr, UCHAR *pAnt, HFREADER_OPRESULT *pResult, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	pAnt（Input）	Antenna control sequence, corresponding to Ant1~Ant6 on the device 0: turn off the antenna port 1: antenna port on
	pResult（Output）	Output: Result of the operation See 1.1.1 Description for details
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	0: no response frames
		>0：Response frame length
Example	rlt = hfReaderSelectAnt(h, 0x000, 0x0001, pAnt, pResult, NULL, NULL);	

2.1.15 hfReaderSetPower

hfReaderSetPower()is to set the power level of the RF output from each antenna group. The function is described in detail in Table 2.15 below
shown in the following table：

Table 2.15 Functions hfReaderSetAntPower()

Function	Control of reader RF signals	
Prototype	int hfReaderSetPower(HANDLE h, USHORT srcAddr, USHORT targetAddr, UCHAR ant, UCHAR pwr, HFREADER_OPRESULT *pResult, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	ant（Input）	Antenna index, starting from 0
	pwr（Input）	Antenna Output RF Power Control 0: lv0 - RF output low 1: lv1 2: lv2 3: lv3-RF output high Other: not valid
	pResult（Output）	Output: Result of the operation See 1.1.1 Description for details
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	0: no response frames
		>0：Response frame length
Example	rlt = hfReaderSetPower(h, 0x000, 0x0001, 0, 0, pResult, NULL, NULL);	

2.1.16 hfReaderScanUsbList

hfReaderScanUsbList()is to open all USB interface devices and return the device handle and the number of devices. The details of the function are shown in Table 2.16 below：

Table 2.16 Functions hfReaderScanUsbList()

Function	Scanning USB devices	
Prototype	int hfReaderScanUsbList(DWPRD vid, DWORD pid, HANDLE *pHandleList)	
Parameters	vid（Input）	0x0505
	pid（Input）	0x5050
	pHandleList（Output）	List of USB reader communication handles
Back	int	Number of USB devices
Example	int num = hfReaderScanUsbList(0x0505, 0x5050, pHandleList);	

2.1.17 hfReaderOpenSocket

hfReaderOpenSocket is to open the Ethernet interface device of the reader system and return the device handle. The function is described in detail in Table 2.17 below：

Table 2.17 Functions hfReaderOpenSocket()

Function	Scanning Ethernet devices	
Prototype	HANDLE hfReaderOpenSocket(char *pIP, DWORD iPort);	
Parameters	pIP（Input）	IP address of the device
	iPort（Input）	Device port number
Back	HANDLE	-1：Failed to open the device Non - 1: Device handle
Example	HANDLE hSerial = hfReaderOpenSocket(“192.168.1.7”, “10001”);	

2.1.18 hfReaderCloseSocket

hfReaderCloseSocket() is to turn off the Ethernet device of the reader system. The functions are described in detail in Table 2.12 below：

Table 2.12 Functions hfReaderCloseSocket()

Function	Shutdown the device based on the specified Ethernet device handle	
Prototype	void hfReaderCloseSocket(HANDLE hSerial)	
Parameters	hSerial（Input）	Ethernet devices that have been opened handle
Back	void	No return value
Example	hfReaderCloseSocket(hSerial);	

Chapter 3 The iso15693 protocol tagging API

3.1 Library function descriptions

3.1.1 iso15693GetUid

iso15693GetUid() gets the tag UID, which cannot be obtained when the reader system is in UID active sending mode:

Function	Get Tags UID	
Prototype	int iso15693GetUid(HANDLE h, USHORT srcAddr, USHORT targetAddr, UCHAR mode, ISO15693_UIDPARAM *pUid, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	mode（Input）	0：Normal mode
		1：Repeat mode, repeats the last UID sent
	pUid（Output）	Output: The UID obtained See 1.1.6 Description for details
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	0: no response frames
		>0：Response frame length
Example	rlt = iso15693GetUid(h, 0x000, 0x0001, 0x00, pUid, NULL, NULL);	

3.1.2 iso15693ReadBlock

iso15693ReadBlock()Reads data from a block of data for a specified tag. The functions are described in detail in Table 3.2 below:

Table 3.2 Function iso15693ReadBlock()

Function	Reads data from a data block with a specified label	
Prototype	int iso15693ReadBlock(HANDLE h, USHORT srcAddr, USHORT targetAddr, UCHAR *pUid, ISO15693_BLOCKPARAM *pReadBlock, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	pUid（Input）	Specified tag UID array, 8 bytes Example: 0xE0 0x04 0x00 0x01 0x02 0x03 0x04 0x05
	pReadBlock（Inputs Outputs）	Output: Data block data and operation results Input: data block address information See 1.1.7 Description for details
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	0: no response frames
		>0：Response frame length
Example	rlt = iso15693ReadBlock(h, 0x000, 0x0001, pUid, pReadBlock, NULL, NULL);	

3.1.3 iso15693WriteBlock

iso15693WriteBlock()Writes data to the data block of the specified label. A detailed description of the function is shown in Table 3.3 below:

Table 3.3 Functions iso15693WriteBlock()

Function	Write data to the data block of the specified label	
Prototype	int iso15693WriteBlock(HANDLE h, USHORT srcAddr, USHORT targetAddr, UCHAR *pUid, ISO15693_BLOCKPARAM *pWriteBlock, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	pUid（Input）	Specified tag UID array, 8 bytes 节 Example：0xE0 0x04 0x00 0x01 0x02 0x03 0x04 0x05
	pWriteBlock(Inputs Outputs)	Output: Result of the operation Input: data block address information See 1.1.7 Description for details
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	0: no response frames
		>0：Response frame length
Example	rlt = iso15693WriteBlock(h, 0x000, 0x0001, pUid, pWriteBlock, NULL, NULL);	

3.1.4 iso15693LockBlock

iso15693LockBlock()Locks the data block for the specified tag. A detailed description of the function is shown in Table 3.4 below：

Table 3.4 Functions iso15693LockBlock()

Function	Locking data blocks	
Prototype	int iso15693LockBlock(HANDLE h, USHORT srcAddr, USHORT targetAddr, UCHAR *pUid, UCHAR blockAddr, HFREADER_OPRESULT *pResult, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	pUid（Input）	Specified tag UID array, 8 bytes For example：0xE0 0x04 0x00 0x01 0x02 0x03 0x04 0x05
	blockAddr（Input）	Address of the locked data block
	pResult（Output）	Output: Result of the operation See 1.1.1 Description for details
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	0: no response frames
		>0：Response frame length
Example	rlt = iso15693LockBlock(h, 0x000, 0x0001, pUid, 0x00, pResult, NULL, NULL);	

3.1.5 iso15693WriteAfi

iso15693WriteAfi()Writes an AFI value to the specified tag. The function is described in detail in Table 3.5 below:

Table 3.5 Functions iso15693WriteAfi()

Function	Write AFI value	
Prototype	int iso15693WriteAfi(HANDLE h, USHORT srcAddr, USHORT targetAddr, UCHAR *pUid, UCHAR afi, HFREADER_OPRESULT *pResult, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	pUid（Input）	Specified tag UID array, 8 bytes For example：0xE0 0x04 0x00 0x01 0x02 0x03 0x04 0x05
	afi（Input）	afi value
	pResult（Output）	Output: Result of the operation See 1.1.1 Description for details
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	0: no response frames
		>0：Response frame length
Example	rlt = iso15693WriteAfi(h, 0x000, 0x0001, pUid, 0x00, pResult, NULL, NULL);	

3.1.6 iso15693LockAfi

iso15693LockAfi()Locks the specified tag AFI value. A detailed description of the function is shown in Table 3.6 below：

Table 3.6 Functions iso15693LockAfi()

Function	Lock AFI value	
Prototype	int iso15693LockAfi(HANDLE h, USHORT srcAddr, USHORT targetAddr, UCHAR *pUid, HFREADER_OPRESULT *pResult, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	pUid（Input）	Specified tag UID array, 8 bytes For example：0xE0 0x04 0x00 0x01 0x02 0x03 0x04 0x05
	pResult（Output）	Output: Result of the operation See 1.1.1 Description for details
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	0: no response frames
		>0：Response frame length
Example	rlt = iso15693LockAfi(h, 0x000, 0x0001, pUid, pResult, NULL, NULL);	

3.1.7 iso15693WriteDsfid

iso15693WriteDsfid()Writes a DSFID value to the specified label. A detailed description of the function is shown in Table 3.7 below:

Table 3.7 Functions iso15693WriteDsfid()

Function	Write DSFID value	
Prototype	int iso15693WriteDsfid(HANDLE h, USHORT srcAddr, USHORT targetAddr, UCHAR *pUid, UCHAR dsfid, HFREADER_OPRESULT *pResult, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h (Input)	Serial device handles that have been opened
	srcAddr (Input)	Source address
	targetAddr (Input)	Target address
	pUid (Input)	Specified tag UID array, 8 bytes For example: 0xE0 0x04 0x00 0x01 0x02 0x03 0x04 0x05
	dsfid (Input)	dsfid value
	pResult (Output)	Output: Result of the operation See 1.1.1 Description for details
	pTxFrame (Output)	Request frames sent
	pRxFrame (Output)	Received response frames
Back	int	0: no response frames
		>0: Response frame length
Example	rlt = iso15693WriteDsfid(h, 0x000, 0x0001, pUid, 0x00, pResult, NULL, NULL);	

3.1.8 iso15693LockDsfid

iso15693LockDsfid() Locks the DSFID value of the specified tag. A detailed description of the function is shown in Table 3.8 below:

Table 3.8 Functions iso15693LockDsfid()

Function	Lock the DSFID value	
Prototype	int iso15693LockDsfid(HANDLE h, USHORT srcAddr, USHORT targetAddr, UCHAR *pUid, HFREADER_OPRESULT *pResult, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h (Input)	Serial device handles that have been opened
	srcAddr (Input)	Source address
	targetAddr (Input)	Target address
	pUid (Input)	Specified tag UID array, 8 bytes For example: 0xE0 0x04 0x00 0x01 0x02 0x03 0x04 0x05
	pResult (Output)	Output: Result of the operation See 1.1.1 Description for details
	pTxFrame (Output)	Request frames sent
	pRxFrame (Output)	Received response frames
Back	int	0: no response frames
		>0: Response frame length
Example	rlt = iso15693LockDsfid(h, 0x000, 0x0001, pUid, pResult, NULL, NULL);	

3.1.9 iso15693ReadTagInfo

iso15693ReadTagInfo ()Gets information about the specified label. The function is described in detail in Table 3.9 below：

Table 3.8 Functions iso15693LockDsfid()

Function	Get label information	
Prototype	int iso15693ReadTagInfo(HANDLE h, USHORT srcAddr, USHORT targetAddr, UCHAR *pUid, ISO15693_TAGPARAM *pTagInfo, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	pUid（Input）	Specified tag UID array, 8 bytes For example：0xE0 0x04 0x00 0x01 0x02 0x03 0x04 0x05
	pTagInfo（Output）	Output: Label information and operation results See 1.1.8 Description for details
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	0: no response frames
		>0：Response frame length
Example	rlt = iso15693ReadTagInfo(h, 0x000, 0x0001, pUid, pTagInfo, NULL, NULL);	

3.1.10 iso15693SetEas

iso15693SetEas() Controls the EAS parameters for a given label. The functions are described in detail in Table 3.10 below:

Table 3.10 Functions iso15693SetEas()

Function	Control of EAS parameters (only valid for labels containing EAS parameters)	
Prototype	int iso15693SetEas(HANDLE h, USHORT srcAddr, USHORT targetAddr, UCHAR *pUid, , UCHAR cmd, HFREADER_OPRESULT *pResult, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	pUid（Input）	Specified tag UID array, 8 bytes For example：0xE0 0x04 0x00 0x01 0x02 0x03 0x04 0x05
	cmd（Input）	0x29：Positioning EAS
		0x2A：Reset EAS
		0x2B：Locking EAS
	pResult（Output）	Output: Result of the operation See 1.1.1 Description for details
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	0: no response frames
		>0：Response frame length
Example	rlt = iso15693SetEas(h, 0x000, 0x0001, pUid, 0x2A, pResult, NULL, NULL);	

3.1.11 iso15693Dtu

iso15693Dtu()Pass-through user command frame control tag. A detailed description of the function is shown in Table 3.11 below:

Table 3.11 Functions iso15693Dtu()

Function	Pass-through orders	
Prototype	int iso15693ADtu(HANDLE h, USHORT srcAddr, USHORT targetAddr, ISO15693_DTU *pDtu, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h (Input)	Serial device handles that have been opened
	srcAddr (Input)	Source address
	targetAddr (Input)	Target address
	pDtu (Inputs Outputs)	Input: request frame Output: response frame and operation result See 1.1.9 Description for details
	pTxFrame (Output)	Request frames sent
	pRxFrame (Output)	Received response frames
Back	int	0: no response frames
		>0: Response frame length
Example	rlt = iso15693Dtu(h, 0x000, 0x0001, pDtu, NULL, NULL);	

3.1.12 iso15693OpTagsRUidAndBlock

iso15693OpTagsRUidAndBlock()is a 2s loop that reads the UIDs of all tags in the field, as well as the tag's data block data. A detailed description of the function is shown in Table 3.12 below:

Table 3.12 Functions iso15693OpTagsRUidAndBlock()

Function	2s loop to read the UIDs of all tags in the field, and the tag's data block data	
Prototype	int iso15693OpTagsRUidAndBlock(HANDLE h, USHORT srcAddr, USHORT targetAddr, UCHAR *pBlockAddr , UCHAR num, UCHAR *pUid, UCHAR *pBlock, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	pBlockAddr(Input)	List of data block addresses, allowing discontinuous data block addresses
	num（Input）	0:Read tags only UID 1~4:Number of data blocks >4:invalid
	pUid（Output）	UID Arrays, such as:00112233445566e077889944556611e0…
	pBlock（Output）	Block data, if a block fails to be read, it is replaced by FFFFFFFF
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	-2: Device not responding -1: ag unstable, read timeout >0: pUid pUid array length, number of tags = array length / 8

Example	rlt = iso15693OpTagsRUidAndBlock(h, 0x000, 0x0001, pBlockAddr, 2,
	pUid, pBlock, NULL, NULL);	

3.1.13 iso15693ImOp

iso15693ImOp ()is a loop that reads the UIDs of all tags in the field, as well as the tag's data block data. A detailed description of the function is shown in Table 3.13 below:

Table 3.13 Functions iso15693ImOp ()

Function	Loop through the UIDs of all tags in the field, as well as the tag's data block data	
Prototype	int iso15693ImOp (HANDLE h, USHORT srcAddr, USHORT targetAddr, ISO15693_IMOP *pOp, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h (Input)	Serial device handles that have been opened
	srcAddr (Input)	Source address
	targetAddr (Input)	Target address
	pOp (Inputs Outputs)	Reference 1.1.22
	pTxFrame (Output)	Request frames sent
	pRxFrame (Output)	Received response frames
Back	int	pUid array length, number of labels = array length / 8
Example	rlt = iso15693ImOp (h, 0x000, 0x0001, pOp, NULL, NULL);	

3.1.14 iso15693OpTagsWriteBlock

iso156930pTagsWriteBlock()is to cycle through the data blocks of all the labels in the field. The function is described in detail in the following table 3.14 shows：

Table 3.14 Functions iso15693OpTagsWriteBlock()

Function	Cyclic writing of data block data for all tags in the field	
Prototype	int iso15693OpTagsWriteBlock(HANDLE h, USHORT srcAddr, USHORT targetAddr, USHORT tagRspTimeMin, USHORT tagRspTimeMax, UCHAR tagNum, UCHAR *pUid, UCHAR blockNum, UCHAR *pBlockAddr , UCHAR *pBlock, UCHAR *pResult, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	tagRspTimeMin（Input）	Minimum label response time in us
	tagRspTimeMax（Input）	Label response time maximum, in us
	tagNum（Input）	Number of labels
	pUid（Input）	UID Arrays, such as:00112233445566e077889944556611e0…
	blockNum（Input）	1~32:Number of data blocks
	pBlockAddr（Input）	List of data block addresses, allowing discontinuous data block addresses
	pBlock（Input）	Data to be written to the data block
	pResult（Output）	Result per data block: 1 - success, 0 - failure
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	Response frame length

Example	rlt = iso156930pTagsWriteBlock(h, 0x000, 0x0001, 4000, 5000, 2, pUid, 2, pBlockAddr, pBlock,
	pResult, NULL, NULL);	

3.1.15 iso15693OpTagsWriteAfi

iso15693OpTagsWriteAfi()is to cycle through the AFI values of all tags in the write field. The function is described in detail in Table 3.15 below shown in：

Table 3.15 Functions iso15693OpTagsWriteAfi()

Function	Cyclic writing of AFI for all tags in the field	
Prototype	int iso15693OpTagsWriteAfi(HANDLE h, USHORT srcAddr, USHORT targetAddr, USHORT tagRspTimeMin, USHORT tagRspTimeMax, UCHAR tagNum, UCHAR *pUid, UCHAR afi, UCHAR *pResult,	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	tagRspTimeMin（Input）	Minimum label response time in us
	tagRspTimeMax（Input）	Label response time maximum, in us
	tagNum（Input）	Number of labels
	pUid（Input）	UID　Arrays, such as:00112233445566e077889944556611e0…
	afi（Input）	afi value
	pResult（Output）	Result per data block: 1 - success, 0 - failure
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	Response frame length
Example	rlt = iso15693OpTagsWriteAfi(h, 0x000, 0x0001, 5000, 7000, 2, pUid, 2, pResult, NULL, NULL);	

3.1.16 iso15693OpTagsReadAfi

iso15693OpTagsReadAfi()is to cycle through the AFI values of all tags in the field. The function is described in detail in Table 3.16 below shown in the following table：

Table 3.16 Functions iso15693OpTagsReadAfi()

Function	Cycle through the AFI of all tags in the admission	
Prototype	int iso15693OpTagsReadAfi(HANDLE h, USHORT srcAddr, USHORT targetAddr, UCHAR tagNum, UCHAR *pUid, UCHAR *pAfiValue, UCHAR *pResult, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	tagNum（Input）	Number of labels
	pUid（Input）	UID Arrays, such as:00112233445566e077889944556611e0…
	pAfiValue（Output）	First address of the array of AFI values to be read
	pResult（Output）	Result per data block: 1 - success, 0 - failure
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	Response frame length
Example	rlt = iso15693OpTagsReadAfi(h, 0x000, 0x0001, 2, pUid, pAfiValue, pResult, NULL, NULL);	

3.1.17 iso15693GetGateAlarmUid

iso15693GetGateAlarmUid()is to cycle through the channel door alarm tags. The function is described in detail in Table 3.17 below shows:

Table 3.17 Functions iso15693GetAlarmUid()

Function	Cyclic reading of alarm tags	
Prototype	int iso15693GetGateAlarmUid(HANDLE h, USHORT srcAddr, USHORT targetAddr, ISO15693_GATEUIDPARAM *pUid,	
Parameters	h (Input)	Serial device handles that have been opened
	srcAddr (Input)	Source address
	targetAddr (Input)	Target address
	layerNum (Input)	Number of layers
	pUid (Output)	Output: Alarm tag UID
	pTxFrame (Output)	Request frames sent
	pRxFrame (Output)	Received response frames
Back	int	Response frame length
Example	rlt = iso15693GetGateAlarmUid(h, 0x000, 0x0001, pUid, NULL, NULL);	

Chapter 4 iso14443A protocol tagging operations API

4.1 Library function descriptions

4.1.1 iso14443AGetUid

iso14443AGetUID() This function does not obtain the UID when the reader system is in UID active mode, as detailed in Table 4.1 below:

Table 4.1 Functions iso14443AGetUID()

Function	Get Tags UID	
Prototype	int iso14443AGetUid(HANDLE h, USHORT srcAddr, USHORT targetAddr, UCHAR mode, ISO14443A_UIDPARAM *pUid, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	mode（Input）	0x26：Read idle tags
		0x52：Read all tags
	pUid（Output）	Output: The UID obtained See 1.1.11 Description for details
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	0: no response frames
		>0：Response frame length
Example	rlt = iso14443AGetUID(h, 0x000, 0x0001, 0x00, pUid, NULL, NULL);	

4.1.2 iso14443AAuthReadM1Block

iso14443AAuthReadM1Block()Reads data from a block with the specified label. The functions are described in detail in Table 4.2 below shows:

Table 4.2 Functions iso14443AAuthReadM1Block()

Function	Reads data from a data block with a specified label	
Prototype	int iso14443AAuthReadM1Block(HANDLE h, USHORT srcAddr, USHORT targetAddr, ISO14443A_BLOCKPARAM *pBlock, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	pBlock（Inputs Outputs）	Output: Data block data and operation results Input: data block address information, password information Input: data block address information, password information
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	0: no response frames
		>0：Response frame length
Example	rlt = iso14443AAuthReadM1Block(h, 0x000, 0x0001, pBlock, NULL, NULL);	

4.1.3 iso14443AAuthWriteM1Block

iso14443AAuthWriteM1Block()Writes data to the data block of the specified label. The function is described in detail in Table 4.3 below shown in：

Table 4.3 Functions iso14443AAuthWriteM1Block()

Function	Write data	
Prototype	int iso14443AAuthWriteM1Block(HANDLE h, USHORT srcAddr, USHORT targetAddr, ISO14443A_BLOCKPARAM *pBlock, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	pBlock（Inputs Outputs）	Output: Result of the operation Input: Data block address information,password information and data block data See 1.1.12 Description for details
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	0: no response frames
		>0：Response frame length
Example	rlt = iso14443AAuthWriteM1Block(h, 0x000, 0x0001, pBlock, NULL, NULL);	

4.1.4 iso14443AAuthReadM1Value

iso14443AAuthReadM1Value()Reads a block of data from a specified tag in the value format. Details of the functions are shown in Table 4.4 below：

Table 4.4 Functions iso14443AAuthReadM1Value()

Function	Read the data block data of the specified label in value format	
Prototype	int iso14443AAuthReadM1Block(HANDLE h, USHORT srcAddr, USHORT targetAddr, ISO14443A_VALUEPARAM *pValue, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	pValue（Inputs Outputs）	Output: value and result of the operation Input: data block address information,password information See 1.1.13 Description for details
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	0: no response frames
		>0：Response frame length
Example	rlt = iso14443AAuthReadM1Value(h, 0x000, 0x0001, pValue, NULL, NULL);	

4.1.5 iso14443AAuthWriteM1Value

iso14443AAuthWriteM1Value()Writes data to the data block of the specified label in the value format. The functions are described in detail in Table 4.5 below:

Table 4.5 Functions iso14443AAuthWriteM1Value()

Function	Write value	
Prototype	int iso14443AAuthWriteM1Value(HANDLE h, USHORT srcAddr, USHORT targetAddr, ISO14443A_VALUEPARAM *pValue, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	pValue（Inputs Outputs）	Output: Result of the operation Input: Data block address information,password information and data block data See 1.1.13 Description for details
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	0: no response frames
		>0：Response frame length
Example	rlt = iso14443AAuthWriteM1Value(h, 0x000, 0x0001, pValue, NULL, NULL);	

4.1.6 iso14443AAuthOpM1Value

iso14443AAuthOpM1Value()Performs a value operation on the data block of the specified label. The functions are described in detail in Table 4.6 below shown below:

Table 4.6 Functions iso14443AAuthOpM1Value()

Function	Value added, impairment and transfer values	
Prototype	int iso14443AAuthOpM1Value(HANDLE h, USHORT srcAddr, USHORT targetAddr, ISO14443A_VALUEPARAM *pValue, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	pValue（Inputs Outputs）	Output: Result of the operation Input: data block address information,password information, value and operation type See 1.1.13 Description for details
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	0: no response frames
		>0: Response frame length
Example	rlt = iso14443AAuthOpM1Value(h, 0x000, 0x0001, pValue, NULL, NULL);	

4.1.7 iso14443AReadM0Block

iso14443AReadM0Block()Retrieves data from the data page of the specified tag. The function is described in detail in Table 4.7 below:

Table 4.7 Functions iso14443AReadM0Block()

Function	Retrieve data from the data page of the specified tag	
Prototype	int iso14443AReadM0Block(HANDLE h, USHORT srcAddr, USHORT targetAddr, ISO14443A_BLOCKPARAM *pBlock, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	pBlock（Inputs Outputs）	Output: Data page data and operation results Input: Data page address information, password information See 1.1.12 Description for details
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	0: no response frames
		>0：Response frame length
Example	rlt = iso14443AReadM0Block(h, 0x000, 0x0001, pBlock, NULL, NULL);	

4.1.8 iso14443AWriteM0Block

iso14443AWriteM0Block()Writes data to the data page of the specified label. The function is described in detail in Table 4.8 below shows:

Table 4.8 Functions iso14443AWriteM0Block()

Function	Write data	
Prototype	int iso14443AWriteM0Block(HANDLE h, USHORT srcAddr, USHORT targetAddr, ISO14443A_BLOCKPARAM *pBlock, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	pBlock（Inputs Outputs）	Output: Result of the operation Input: data block address information, password information and data page data See 1.1.12 Description for details
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	0: no response frames
		>0：Response frame length
Example	rlt = iso14443AWriteM0Block(h, 0x000, 0x0001, pBlock, NULL, NULL);	

4.1.9 iso14443ARats

iso14443ARats()Gets the rats information for the specified tag. The function is described in detail in Table 4.9 below:

Table 4.9 Functions iso14443ARats()

Function	Get rats	
Prototype	int iso14443ARats(HANDLE h, USHORT srcAddr, USHORT targetAddr, ISO14443A_OPPARAM *pOpInfo, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	pOpInfo（Output）	Output: Result of the operation See 1.1.14 Description for details
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	0: no response frames
		>0: Response frame length
Example	rlt = iso14443ARats(h, 0x000, 0x0001, pOpInfo, NULL, NULL);	

4.1.10 iso14443ACtrlEsam

iso14443ACtrlEsam()Gets the rats information for the specified tag. The function is described in detail in Table 4.10 below:

Table 4.10 Functions iso14443ACtrlEsam()

Function	Get rats	
Prototype	int iso14443ACtrlEsam(HANDLE h, USHORT srcAddr, USHORT targetAddr, UCHAR index, UCHAR state, ISO14443A_OPPARAM *pOpInfo, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	index	1：ESAM1
		2：ESAM2
	state	0：Power off
		1：Power on, return rats message
		Other: reset, return rats information
	pOpInfo（Output）	Output: Result of the operation See 1.1.14 Description for details
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	0: no response frames
		>0：Response frame length
Example	rlt = iso14443ACtrlEsam(h, 0x000, 0x0001, 1, 1, pOpInfo, NULL, NULL);	

4.1.11 iso14443AApdu

iso14443AApdu()Pass-through of application level commands. A detailed description of the functions is shown in Table 4.11 below:

Table 4.11 Functions iso14443AApdu()

Function	Pass-through of application layer commands ISO14443A4	
Prototype	int iso14443AApdu(HANDLE h, USHORT srcAddr, USHORT targetAddr, UCHAR index, ISO14443A_OPPARAM *pOpInfo, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	index	0：Non-contact labels
		1：ESAM1
		2：ESAM2
	pOpInfo（Output）	Output: Result of the operation See 1.1.14 Description for details
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	0: no response frames
		>0：Response frame length
Example	rlt = iso14443AApdu(h, 0x000, 0x0001, 1, 1, pOpInfo, NULL, NULL);	

4.1.12 iso14443AHalt

iso14443AHalt()Puts up the label. The function is described in detail in Table 4.12 below:

Table 4.12 Functions iso14443AHalt()

Function	Hang up the label	
Prototype	int iso14443AHalt(HANDLE h, USHORT srcAddr, USHORT targetAddr, ISO14443A_OPPARAM *pOpInfo, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	pOpInfo（Output）	Output: Result of the operation See 1.1.14 Description for details
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	0: no response frames
		>0：Response frame length
Example	rlt = iso14443AHalt(h, 0x000, 0x0001, pOpInfo, NULL, NULL);	

4.1.13 iso14443ASDsel

iso14443ASDsel()Uncheck. A detailed description of the function is shown in Table 4.13 below:

Table 4.13 Functions iso14443ASDsel()

Function	Uncheck	
Prototype	int iso14443ASDsel(HANDLE h, USHORT srcAddr, USHORT targetAddr, ISO14443A_OPPARAM *pOpInfo, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	pOpInfo（Output）	Output: Result of the operation See 1.1.14 Description for details
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	0: no response frames
		>0：Response frame length
Example	rlt = iso14443ASDsel(h, 0x000, 0x0001, pOpInfo, NULL, NULL);	

4.1.14 iso14443ADtu

iso14443ADtu()Pass-through user command frame control labels. The functions are described in detail in Table 4.14 below:

Table 4.14 Functions iso14443ADtu()

Function	Pass-through orders	
Prototype	int iso14443ADtu(HANDLE h, USHORT srcAddr, USHORT targetAddr, ISO14443A_DTU *pDtu, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	pDtu（Inputs Outputs）	Input: request frame Output: response frame and result of the operation See 1.1.15 Description for details
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	0: no response frames
		>0：Response frame length
Example	rlt = iso14443ADtu(h, 0x000, 0x0001, pDtu, NULL, NULL);	

4.1.15 iso14443AReadM0Cnt

iso14443AReadM0Cnt()Reads the tag READ_CNT parameter. A detailed description of the function is shown in Table 4.15 below:

Table 4.15 Functions iso14443AReadM0Cnt()

Function	Read READ_CNT	
Prototype	int iso14443AReadM0Cnt(HANDLE h, USHORT srcAddr, USHORT targetAddr, ISO14443A_OPPARAM *pOpInfo, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	pOpInfo（Output）	Output: CNT and operation results See 1.1.14 Description for details
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	0: no response frames
		>0：Response frame length
Example	rlt = iso14443AReadM0Cnt(h, 0x000, 0x0001, pOpInfo, NULL, NULL);	

4.1.16 iso14443AReadM0Sig

iso14443AReadM0Sig()Reads the tag SIG parameter. A detailed description of the function is shown in Table 4.16 below:

Table 4.16 Functions iso14443AReadM0Sig()

Function	Read SIG	
Prototype	int iso14443AReadM0Sig(HANDLE h, USHORT srcAddr, USHORT targetAddr, ISO14443A_OPPARAM *pOpInfo, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h (Input)	Serial device handles that have been opened
	srcAddr (Input)	Source address
	targetAddr (Input)	Target address
	pOpInfo (Output)	Output: SIG and operation result See 1.1.14 Description for details
	pTxFrame (Output)	Request frames sent
	pRxFrame (Output)	Received response frames
Back	int	0: no response frames
		>0: Response frame length
Example	rlt = iso14443AReadM0Sig(h, 0x000, 0x0001, pOpInfo, NULL, NULL);	

4.1.17 iso14443AAuthM0

iso14443AAuthM0()The functions are described in detail in Table 4.17 below：

Table 4.17 Functions iso14443AAuthM0()

Function	Authorize tags and get PACK	
Prototype	int iso14443AAuthM0(HANDLE h, USHORT srcAddr, USHORT targetAddr, ISO14443A_OPPARAM *pOpInfo, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	pOpInfo（Output）	Output: PACK and the result of the operation See 1.1.14 Description for details
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	0: no response frames
		>0：Response frame length
Example	rlt = iso14443AAuthM0(h， 0x000， 0x0001， pOpInfo， NULL， NULL)；	

4.1.18 iso14443AAuthUltralightC

iso14443AAuthUltralightC()Authorizes the authentication tag according to the secret key 3DES. The functions are described in detail in Table 4.18 below shown in the following table:

Table 4.18 Functions iso14443AAuthUltralightC()

Function	3DES Authorised Certification UltralightC Label	
Prototype	int iso14443AAuthUltralightC (HANDLE h, USHORT srcAddr, USHORT targetAddr, UCHAR *pKey, HFREADER_OPRESULT *pResult, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h (Input)	Serial device handles that have been opened
	srcAddr (Input)	Source address
	targetAddr (Input)	Target address
	pKey (Input)	3DES Secret Key
	pResult (Output)	Output: Result of the operation See 1.1.1 Description for details
	pTxFrame (Output)	Request frames sent
	pRxFrame (Output)	Received response frames
Back	int	0: no response frames
		>0: Response frame length
Example	rlt = iso14443AAuthUltralightC(h, 0x000, 0x0001, pKey, pResult, NULL, NULL);	

4.1.19 iso14443AReadTopazBlock

iso14443AReadTopazBlock() Retrieves data from the Topaz labeled data block. The function is described in detail in Table 4.19 below
shown below：

Table 4.19 Functions iso14443AReadM0Block()

Function	Retrieve data from the data page of the specified tag	
Prototype	int iso14443AReadTopazBlock(HANDLE h, USHORT srcAddr, USHORT targetAddr, ISO14443A_BLOCKPARAM *pBlock, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	pBlock（Inputs Outputs）	Output: Data page data and operation results Input: Data page address information, password information See 1.1.12 Description for details
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	0: no response frames
		>0：Response frame length
Example	rlt = iso14443AReadTopazBlock(h, 0x000, 0x0001, pBlock, NULL, NULL);	

4.1.20 iso14443AWriteTopazBlock

iso14443AWriteTopazBlock()Writes data to the data page of the Topaz tag. The function is described in detail in the following table 4.20 shows：

Table 4.20 Functions iso14443AWriteTopazBlock()

Function	Write data	
Prototype	int iso14443AWriteTopazBlock(HANDLE h, USHORT srcAddr, USHORT targetAddr, ISO14443A_BLOCKPARAM *pBlock, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	pBlock（Inputs Outputs）	Output: Result of the operation Input: data block address information, password information and data page data See 1.1.12 Description for details
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	0: no response frames
		>0：Response frame length
Example	rlt = iso14443AWriteTopazBlock(h, 0x000, 0x0001, pBlock, NULL, NULL);	

4.1.21 iso14443AImOperation

iso14443AImOperation()Intelligent manipulation of tag data block data based on parameters. The functions are described in detail in the following table 4.21 shows：

Table 4.21 Functions iso14443AImOperation()

Function	Intelligent manipulation of labeled data block data	
Prototype	int iso14443AImOperation(HANDLE h, USHORT srcAddr, USHORT targetAddr, ISO14443A_IMPARAM *pImParams, iso14443aImDeviceInfo deviceInfo, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	pImParams（Inputs Outputs）	Output: operation parameters Input: Result of the operation, UID or data block See 1.1.20 Description for details
	deviceInfo（Input）	Callback functions for handling antenna information and operation information void(CALLBACK *iso14443aImDeviceInfo)(ISO14443A_IMINFO *pInfo)；
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	0: no response frames
		>0：Response frame length
Example	rlt = iso14443AImOperation(h, 0x000, 0x0001, pIm, imCallBack, NULL, NULL)；	

Chapter 5 The iso14443B protocol tagging API

5.1 Library function descriptions

5.1.1 iso14443BSelect

iso14443BSelect() selects the ISO14443B tag, this function cannot obtain a UID. the function is described in detail in Table 5.1 below shown in:

Table 5.1

Function	Get Tags UID	
Prototype	int iso14443BSelect(HANDLE h, USHORT srcAddr, USHORT targetAddr, UCHAR mode, ISO14443B_INFO *pInfo,	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	mode（Input）	Reserved
	pInfo（Output）	Output: Information about the selected tag See 1.1.16 Description for details
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	0: no response frames
		>0：Response frame length
Example	rlt = iso14443BSelect(h, 0x000, 0x0001, 0x00, pInfo, NULL, NULL);	

5.1.2 iso14443BGetIDCardUid

iso14443BGetIDCardUid() reads the UID of the Chinese ID tag. the function is described in detail in Table 5.2 below：

Table 5.2 Functions iso14443BGetIDCardUid()

Function	Get Tags UID	
Prototype	int iso14443BGetIDCardUid(HANDLE h, USHORT srcAddr, USHORT targetAddr, UCHAR *pUid, HFREADER_OPRESULT *pResult, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	pUid（Output）	ID tag UID, 8 bytes
	pResult（Output）	Output: Result of the operation See 1.1.1 Description for details
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	0: no response frames
		>0：Response frame length
Example	rlt = iso14443BGetIDCardUid(h, 0x000, 0x0001, pUid, pResult, NULL, NULL);	

5.1.3 iso14443BDtu

iso14443BDtu()Pass-through user command frame control labels. A detailed description of the function is shown in Table 5.3 below：

Table 5.3 Functions iso14443BDtu()

Function	Pass-through orders	
Prototype	int iso14443BDtu(HANDLE h, USHORT srcAddr, USHORT targetAddr, ISO14443A_DTU *pDtu, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	pDtu（Inputs Outputs）	Input: request frame Output: response frame and result of the operation See 1.1.17 Description for details
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	0: no response frames
		>0：Response frame length
Example	rlt = iso14443BDtu(h, 0x000, 0x0001, pDtu, NULL, NULL);	

5.1.4 iso14443BHalt

iso14443BHalt() hangs the tag. A detailed description of the function is shown in Table 5.4 below:

Table 5.4 Functions iso14443BHalt()

Function	Hang up the label	
Prototype	int iso14443BHalt(HANDLE h, USHORT srcAddr, USHORT targetAddr, HFREADER_OPRESULT *pResult, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	pResult（Output）	Output: Result of the operation See 1.1.1 Description for details
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	0: no response frames
		>0: Response frame length
Example	rlt = iso14443BHalt(h, 0x000, 0x0001, pOpInfo, NULL, NULL);	

Chapter 6 Felica Protocol Label Manipulation API

6.1 Library function descriptions

6.1.1 felicaGetUid

felicaGetUID() obtains the tag UID, which cannot be obtained when the reader system is in UID-active mode, as shown in Table 6.1 below:

Table 6.1 Functions felicaGetUID()

Function	Get Tags UID	
Prototype	int felicaGetUid(HANDLE h, USHORT srcAddr, USHORT targetAddr, UCHAR mode, ISO14443A_UIDPARAM *pUid, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h (Input)	Serial device handles that have been opened
	srcAddr (Input)	Source address
	targetAddr (Input)	Target address
	mode (Input)	Reserved
	pUid (Output)	Output: The UID obtained See 1.1.18 Description for details
	pTxFrame (Output)	Request frames sent
	pRxFrame (Output)	Received response frames
Back	int	0: no response frames
		>0: Response frame length
Example	rlt = felicaGetUID(h, 0x000, 0x0001, 0x00, pUid, NULL, NULL);	

6.1.2 felicaDtu

felicaDtu() transmits the user command frame control tag. A detailed description of the function is shown in Table 6.2 below:

Table 6.2 Function felicaDtu()

Function	Pass-through orders	
Prototype	int felicaDtu(HANDLE h, USHORT srcAddr, USHORT targetAddr, FELICA_DTU *pDtu, UCHAR *pTxFrame, UCHAR *pRxFrame)	
Parameters	h（Input）	Serial device handles that have been opened
	srcAddr（Input）	Source address
	targetAddr（Input）	Target address
	pDtu（Inputs Outputs）	Input: request frame Output: response frame and result of the operation See 1.1.19 Description
	pTxFrame（Output）	Request frames sent
	pRxFrame（Output）	Received response frames
Back	int	0: no response frames
		>0：Response frame length
Example	rlt = felicaDtu(h, 0x000, 0x0001, pDtu, NULL, NULL);	