API/SDK Instruction for UHF Reader

ASKA
ul. Wędkarska 2A/B1, 04-869 Warszawa
tel. 22 4985908/9, fax 22 6177020
e-mail: ask@aska.com.pl www.kodykreskowe.com

Instruction

The API Development Kit is designed for user developing application software conveniently, which provide to user in the form of dynamic link file.

It supports Visual C++、VB、C++ Builder and Delphi platform developing environment. User can correct develop their application software efficiently by reading this instruction manual.

According to the function features, the API function can be classified into two types function: “reader management function” and “tag operating function”; Tag operating function can be classified as “public function” and “tag operating function in specific protocol

Function List

- **Definition of Data Structure**
  1. TagIds structure definition
  2. Gen2TagList structure definition
  3. Gen2LockFlags structure definition
- **Reader management function**
  1. OpenReader ----------Open the reader
  2. CloseReader ----------Close the reader
  3. SetBaudRate ----------Configure the baud rate between PC and reader
  4. SetParameter ----------Configure reader working parameters
  5. GetParameter ---------- ----------Read the working parameters of reader
  6. SetIdmData ----------Write the required ID matching data to reader.
  7. GetIdmData----------Read the ID match data
  8. ResetReader ----------Reset the reader
  9. GetFirmwareVersion ----------Read the program version of reader firmware
  10. BeepControl----------Control the reader buzzer
  11. SelectAntenna----------Choose the working antenna
  12. SetWorkAntenna ----------Configure the working antenna for multi-port reader
  13. QueryAntenna ----------Query the antenna connecting condition
  14.StopRFwork---------- Stop the reader launching power
  15. GetTriggerState ----------Query reader trigger input state
  16. SetRelayState ----------Set reader relay action
  17. Synchronize ----------Synchronize the reader
- **18000-6B Tag Operating Function**
  1. SingleTagIdentify ----------Single tag identification
  2. SingleTagIdentifyEx----------Single tag identifying for multi-port reader
  3. MultipleTagIdentify ----------Multiple tag identification
  4. MultipleTagIdentifyEx----------Multiple tags identifying for multi-port reader
  5. ReadSingleTag----------Identify single tag and read designated memory data
  6. ReadSingleTag0----------Read memory data of designated tag
  7. WriteSingleTag----------Identify single tag and write data to designated memory address
  8. WriteSingleTag0----------Write parameters to the designated memory address of specific tag
  9. LockTag ----------Identify single tag, and lock the designated tag memory address
  10. LockTag0----------Lock the designated memory address of specific tag
  11. QueryLock----------Query the locking status of designated memory address
  12. QueryLock0 ----------Query the locking status of designated tag memory address
- **EPC GEN2 Tag Operating Function**

1. InitGen2Tag ---------- Initialize the EPC Gen2 tag
2. InitGen2EpcCode---------- Initialize the EPC code for EPC Gen2 tag
3. SingleTagIdentify----------Single tag identification
4. MultipleTagIdentify----------Multiple tag identification
5. Gen2ReadTag---------- Read the memory data of EPC Gen2 tag
6. Gen2WriteTag----------Write data to the memory area of EPC Gen2 tag
7. Gen2ReadTagwithEpc----------Read the memory data of designated EPC Gen2 tag
8. Gen2WriteTagwithEpc----------Modify the EPC code for designated EPC Gen2 tag
9. Gen2ReadTagwithEpcPW---------- Read the memory data of designated cryptographic tag
10. Gen2WriteTagwithEpcPW---------- Modify the EPC code of designated cryptographic tag
11. Gen2SetAccessPassword----------Modify the access password for designated tag.
12. Gen2GetAccessPassword----------Read the access password of designated tag.
13. Gen2SetKillPassword----------Modify the kill password for designated tag
14. Gen2GetKillPassword----------Read the kill password of designated tag
15. Gen2KillTag----------Kill the designated tag
16. Gen2LockTag----------Lock the designated tag
17. Gen2ChangeEAS----------Modify the EAS bit for designated tag
18. Gen2EASAlarm----------Tag alarm
19. Gen2TagContentList----------List the selected contents of EPC Gen2 tag

● Definition of Data Structure

1. TagIds Structure Definition

```
typedef struct
{
    nsigned char TagType;
    nsigned char AntNum;
    nsigned char Length;
    nsigned char Ids[ID_Length];
}TagIds;
```

The definition of TagIds Structure is as below：

- TagType—Tag type
- AntNum—Antenna Serial Number
- Length—Byte length of tag ID
- Ids— Tag ID value

2. Gen2TagList Structure Definition

```
typedef  struct
{
   unsigned char DataMask;
   unsigned char EpcLength;
   unsigned char TidLength;
   unsigned char UserLength;
   unsigned char Epc[24];
   unsigned char Tid[12];
   unsigned char User[64];
}Gen2TagList;
```

“Gen2TagList” is the required structure for listing EPC, TID and USER data at a time.. Its definition as below:

▲ DataMask ---The storage area needs to be read, using the bit mask format
D0 is “1” -- Reading the EPC code,
D0 is “0 “– Doesn’t reading the EPC code
D1 is “1” -- Reading the TID code

D1 is “0” -- Doesn’t reading the TID code
D2 is “1” -- Reading the data of User area
D2 is “0” -- Doesn’t reading the data of User area
D3 ~ D7 ---The five bits data are reserved

▲ EpcLength：The double-byte length of EPC code read (D0 is “1”)
▲ TidLength：The double-byte length of TID code read (D1 is “1”)
▲ UserLength：The double-byte length of User data read (D2 is “1”)
▲ Epc：EPC code read
▲ Tid：TID code read
▲ User：User data read

3、Gen2LockFlags structure definition

```
typedef struct
{
    unsigned char Mask;
    unsigned char Action;
}Gen2LockFlag;

typedef struct
{
    Gen2LockFlag Kill;
    Gen2LockFlag Access;
    Gen2LockFlag EPC;
    Gen2LockFlag TID;
    Gen2LockFlag User;
}Gen2LockFlags;
```

It stipulates the operating of tag KILL password, ACCESS password, EPC, TID and locking user area. The definition of Gen2LockFlag is as below:

- Mask：Operating range of locking area
- Action：Locking type

Gen2LockFlags structure definition

- Kill：Operational definition for locking KILL password
- Access：Operational definition for locking Access password
- EPC：Operational definition for locking EPC memory area
- TID：Operational definition for locking TID memory area
- User：Operational definition for locking USER memory area

● Reader Management Function

Reader management function is used to achieve the work of configuring the reader working condition, status query and function operating by host PC.

1. OpenReader

Function prototype: short OpenReader (HANDLE * hCom, unsigned char LinkType,char *com_port)

Explanation: Open the reader

Input parameter:

- hCom-- Handle
- LinkType—connecting type between reader and PC, “1” for serial port, “2” for network port, “3” for USB
- Com_port—It is serial port name when serial port connecting; It is reader’s IP address when network connecting. It is empty character string or any character string when USB connecting.

Return result: the result is zero, which shows the action is correct, others are wrong.

Explanation：It needs to call ”OpenReader” function before operating the reader

2. CloseReader

Function prototype：short CloseReader (HANDLE hCom)

Function explanation：close the reader

Input parameter:

- hCom-- Handle

Return results: the result is zero, which shows the action is correct, others are wrong

Explanation：It need to call CloseReader：when quit the application program so as to release system resource.

3. SetBaudRate

Function prototype：short SetBaudRate (HANDLE hCom, unsigned short BaudRate)

Function explanation：Configure the baud rate between PC and reader

Input parameter：

- BaudRate：configuring BaudRate, the value is 0,1,2,3, 4, the corresponding baudrate is 9600, 19200, 38400, 57600 and 115200bps.

Return results: The result is zero, which shows the action is correct, others are wrong.

Expalination: ”SetBaudRate” function is only suitable for modifying the reader Baudrate when connecting the reader by RS-232

4.SetParameter

Function prototype：short SetParameter(HANDLE hCom, unsigned int Addr, unsigned char Count, unsigned char * Parameter)

Function explanation： configuring the reader working parameters.

Input parameter：

- hCom-- Handle
- Addr：The first working parameter address needs to be configured, range from 2 to 255
- value：The working parameters' value needs to be configured. For details, please refer to the instruction manual for ReaderSetup.exe

Return results: the result is zero, which shows the action is correct, others are wrong.

Explanation：It's internal function, usually users no need to use it.

5. GetParameter

Function prototype：short GetParameter(HANDLE hCom, unsigned int Addr, unsigned char Count,unsigned char *Value)

Function explanation：Reading the multiple working parameters of reader

Input parameter：

- hCom-- Handle
- Addr：The first address of working parameters
- Count: The number of parameters
- value：The return value of reading working parameters

Return results: the result is zero, which shows the action is correct, others are wrong.

Explanation：It's internal function, usually users no need to use it.

6、SetIdmData

Function prototype：short SetIdmData(HANDLE hCom, unsigned char order,unsigned char match_length, unsigned int count,TagIds *value)

Function explanation：Before using the ID matching function, it's necessary to call the "SetIdmData" function and write all the required ID data to the reader.

Input parameter:

- hCom—Handle
- order— Tag ID sort order, 0 for descending order, 1 for ascending order'
- match_length—Byte length needs to be matched.
- count— The number of tag ID
- Value—Tag data. Please refer to the definition of TagIds

Return results: the result is zero, which shows the action is correct, others are wrong

9. GetIdmData

Function prototype：GetIdmData (HANDLE hCom, unsigned char * moreFlag,unsigned int *count,TagIds *value)

Function explanation：Read the ID match data in reader

Input parameter:

- hCom—Handle
- moreFlag：Mark for unfinished ID match data
- count：The number of expected ID matching data when calling this function. It also refers to the number of actual ID matching data read when function returns
- Value—Tag data. Please refer to the definition of TagIds

Return results: the result is zero, which shows the action is correct, others are wrong

8. ResetReader

Function prototype：short ResetReader(HANDLE hCom)

Function explanation: Reset the reader

Input parameter:

- hCom-- Handle

Return results: the result is zero, which shows the action is correct, others are wrong

9. GetFirmwareVersion

Function prototype：short GetFirmwareVersion (HANDLE hCom, Version * version)

Function explanation：Read the program version of reader firmware.

Input parameter：

- hCom-- Handle
- Version：value of firmware program, please refers to the Version data structure

Return results: the result is zero, which shows the action is correct, others are wrong.

10、BeepControl

Function prototype：short BeepControl(HANDLE hCom, unsigned char ControlType)

Function explanation：Control the reader buzzer

Input parameter：

- hCom—Handle
- ControlType— controlling type of buzzer, defined as: 1 start buzzer; 2, stop the buzzer; 3. start the buzzer and automatically stop the buzzer.

Return results: the result is zero, which shows the action is correct, others are wrong.

Explanation：User can control the buzzer action by calling this function.

11、SelectAntenna

Function prototype：short SelectAntenna(HANDLE hCom, unsigned char AntennaNum)

Function explanation： Choose the working antenna

Input parameter：

- hCom—Handle
- AntennaNum：the selected working antenna

Return results: the result is zero, which shows the action is correct, others are wrong.

Explanation： it's only used for multi-ports reader(NFC-9812,NFC-9814 etc.). User can operate the designated antenna and do reading, writing, kill operating for specific tag. This function is not valid for single ports reader.

12□SetWorkAntenna

Function prototype□short SetWorkAntenna (HANDLE hCom, unsigned char Antenna)

Function explanation□Configure the working antenna(channel) for multi-port reader

Input parameter□

- hCom—Handle
- Antenna—Configured antenna, bit mask mode, If D0-D7 is '1', which indicates antena1~8 can work. If it's zero, the corresponding antenna doesn't work.

Return results: the result is zero, which shows the action is correct, others are wrong.

Explanation□It's internal function, usually users no need to use it.

13. QueryAntenna

Function prototype□short QueryAntenna(HANDLE hCom, unsigned char * Antenna)

Function explanation□Query the antenna connecting condition(For multi-port reader)

Input parameter□

- hCom—Handle
- Antenna□Mark of antenna connecting condition, If D0-D7 bit is "1", ndicate the corresponding antenna connect to reader. If it's"0", it means the antenna doesn't connect with reader

Return results: the result is zero, which shows the action is correct, others are wrong.

Explanation□It's internal function, usually users no need to use it.

14□StopRFwork

Function prototype□short StopRFwork(HANDLE hCom)

Function explanation□stop the reader launching power

Input parameter□

- hCom— Handle

Return results: the result is zero, which shows the action is correct, others are wrong.

15□GetTriggerState

Function prototype□short GetTriggerState(HANDLE hCom)

Function explanation□Query reader trigger input state.

Input parameter□

- hCom— Handle

Return results: The returning results is trigger input state, using bit mark encoding, the minimum significance bit "D0" indicates the state of trigger 1, If the value is"0", it means the trigger is invalid, if it's "1", the trigger is valid; "D1" indicates the state of trigger 2, there is no definition for "D2-D7"

16□SetRelayState

Function prototype□short SetRelayState(HANDLE hCom, unsigned char Mask, unsigned char State)

Function explanation□.Set reader relay action

Input parameter□

- hCom— Handle

- Mark— Relay bit mask needs to be operated; D0 is for operating"relay 1".

  If D0 value is"1", it means the state value will affect the change of reply state. If the value is "0", then there is no impact. The definition for D1-D3 is the same as D0.
- State—The bit mask for reply state needs to be changed,, "1" indicates the corresponding replay is closed, "0" indicates the corresponding replay is disconnected.

Return results: the result is zero, which shows the action is correct, others are wrong.

17□Synchronize

Function prototype□Synchronize(HANDLE hCom)

Function explanation□Synchronize the reader

Input parameter□

- hCom— Handle

Return results: the result is zero, which shows the action is correct, others are wrong.

Explanation: This function is used for multiple readers work together and uses the synchronization function. (The host PC maintains the readers' synchronization)

●18000-6B Tag Operating Function

Tag operating function is used to achieve the work of identifying tag, reading/writing tag, lock tag memory and query memory status etc.

1□SingleTagIdentify

Function prototype□short SingleTagIdentify(HANDLE hCom, unsigned int TagType, unsigned char *ID)

Function explanation□Single tag identification

Input parameter□

- hCom— Handle
- TagType□Tag type identification,
  "1" ----ISO18000-6B tag,
  "4"----EPC Gen2 tag.
  "6" ----TID of Gen2 tag
- ID□ return value of identified tag

Return results: the result is zero, which shows the action is correct, others are wrong.

2□SingleTagIdentifyEx

Function prototype □ short SingleTagIdentifyEx(HANDLE hCom, unsigned int TagType, unsigned int * Count,TagIds *value)

Function explanation□single tag identifying for multi-port reader

Input parameter□

- hCom— Handle
- TagType□Tag type identification,
  "1" ----ISO18000-6B tag,
  "4"----EPC Gen2 tag.
  "6" ----TID of Gen2 tag
- Count□The number of Identified tags

- ID： Return value of identified tag. Please refer to the definition of TagIds

Return results: the result is zero, which shows the action is correct, others are wrong.

Explanation：The function is suitable for single tag identifying of multi ports reader.(Model No.NFC-9812, NFC-9814)

3、MultipleTagIdentify

Function prototype： short MultipleTagIdentify(HANDLE hCom, unsigned int TagType, unsigned char * Count,TagIds *value)

Function explanation： Multiple tag identification

Input parameter：

- hCom— Handle
- TagType：Tag type identification,
  "1" ----ISO18000-6B tag,
  "4"----EPC Gen2 tag.
  "6" ----TID of Gen2 tag
- Count：The number of Identified tags
- Value： Return ID value of identified tags. Please refers to the definition of TagIds

Return results: the result is zero, which shows the action is correct, others are wrong.

4、MultipleTagIdentifyEx

Function prototype： short MultipleTagIdentifyEx(HANDLE hCom, unsigned int TagType, unsigned int * Count，unsigned char *ID)

Function explanation：Multiple tags identifying for multi-port reader

Input parameter：

- hCom— Handle
- TagType：Tag type identification,
  "1" ----ISO18000-6B tag,
  "4"----EPC Gen2 tag.
  "6" ----TID of Gen2 tag
- Count：The number of Identified tags
- Value： Return ID value of identified tags.

Return results: the result is zero, which shows the action is correct, others are wrong.

Explanation: By using this function, it can identify all the tags in front of multi-antenna. It's suitable for multi-port reader. (Model No.NFC-9812, NFC-9814)

5、ReadSingleTag

Function prototype：short ReadSingleTag(HANDLE hCom, unsigned int TagType, unsigned char Addr,unsigned char Length,unsigned char *Id,unsigned char *value)

Function explanation：Identify single tag, and read the designated memory data.

Input parameter：

- hCom— Handle
- TagType— Tag type identification, "1" for ISO18000-6B tag,
- Addr：Byte address of tag memory.

- Length：tag byte length needs to be read
- Id：return value of identified tag ID
- Value：return value of reading memory data

Return results: the result is zero, which shows the action is correct, others are wrong.

6、ReadSingleTag0

Function prototype：short ReadSingleTag0(HANDLE hCom, unsigned int TagType, unsigned char *Id,unsigned char Addr,unsigned char Length,unsigned char *value)

Function explanation： Read memory data of designated tag.

Input parameter：

- hCom— Handle
- TagType— Tag type identification, “1” for ISO18000-6B tag,
- Addr：byte address of tag memory
- Length：byte length read
- Id：ID of designated tag
- Value：return value of reading memory data

Return results: the result is zero, which shows the action is correct, others are wrong.

7、WriteSingleTag

Function prototype：short WriteSingleTag(HANDLE hCom, unsigned int TagType, unsigned char *Id,unsigned char Addr,unsigned char Length,unsigned char *value，unsigned char * Result)

Function explanation：identify single tag , and write data to the designated memory address

Input parameter：

- hCom— Handle
- TagType— Tag type identification, “1” for ISO18000-6B tag,
- Addr：byte address of tag memory
- Length：byte length needs to be written
- Id：return value of identified tag ID
- Value：the value which needs to be written into tag memory.
- Result：The returning result for writing tag, the results are:
  “0” for writing failed;
  “1” for writing succeeded;
  “2” for not writable;
  ”3” for unknown results

Return results: the result is zero, which shows the action is correct, others are wrong.

8、WriteSingleTag0

Function prototype：short WriteSingleTag0(HANDLE hCom, unsigned int TagType, unsigned char *Id,unsigned char Addr,unsigned char Length,unsigned char *value,unsigned char * Result)

Function explanation：Write parameters to the designated memory address for specific tag.(The Parameters

with designated byte length)

Input parameter：

- hCom— Handle
- TagType— Tag type identification, “1” for ISO18000-6B tag,
- Addr：byte address of tag memory
- Length：byte length needs to be written
- Id：return value of identified tag ID
- Value：the value which needs to be written into tag memory.
- Result：The returning value for writing tags, the results are:
  “0” for writing failed;
  “1” for writing succeeded;
  “2” for not writable;
  ”3” for unknown results

Return results: the result is zero, which shows the action is correct, others are wrong.

9、LockTag

Function prototype：short LockTag(HANDLE hCom, unsigned int TagType, unsigned char *Id,unsigned char Addr)

Function explanation：Identify single tag, and lock the designated tag memory address

Input parameter：

- hCom— Handle
- TagType— Tag type identification, “1” for ISO18000-6B tag,
- Addr：byte address of tag memory which needs to be locked.
- ID：Identified tag ID No.

Return results: the result is zero, which shows the action is correct, others are wrong.

10、LockTag0

Function prototype：short LockTag0(HANDLE hCom, unsigned int TagType, unsigned char *Id,unsigned char Addr)

Function explanation：Lock the designated memory address of specific tag

Input parameter：

- hCom— Handle
- TagType— Tag type identification, “1” for ISO18000-6B tag,
- Addr：byte address of tag memory which needs to be locked.
- ID： designated tag ID No.

Return results: the result is zero, which shows the action is correct, others are wrong.

11、QueryLock

Function prototype：short QueryLock(HANDLE hCom, unsigned int TagType, unsigned char *Id,unsigned char Addr, unsigned char Length,unsigned char * Value)

Function explanation：Identify single tag, and query the locking status of designated memory address

Input parameter：

- hCom— Handle
- TagType— Tag type identification, “1” for ISO18000-6B tag,
- Addr：byte address of tag memory which needs to query the locking status.
- ID： Returning value of identified tag ID.
- Length：The number of bytes which needs to query the locking status.
- Value：The locking status of corresponding bytes(returned by reader).

“1” for locked; “2”for unlocked;”3” for unknown locking status.

Return results: the result is zero, which shows the action is correct, others are wrong.

12、QueryLock0

Function prototype：short QueryLock0(HANDLE hCom, unsigned int TagType, unsigned char *Id,unsigned char Addr，unsigned char Length,unsigned char * Value)

Function explanation：query the locking status of designated memory address for specific tag

Input parameter：

- hCom— Handle
- TagType— Tag type identification, “1” for ISO18000-6B tag,
- Addr：byte address of tag memory which needs to query the locking status.
- ID：designated tag ID value
- Length：The number of bytes which needs to query the locking status.
- Value：The locking status of corresponding bytes(returned by reader).

“1” for locked; “0”for unlocked;”2” for unknown locking status.

Return results: the result is zero, which shows the action is correct, others are wrong.

●EPC GEN2 Tag Operating Function

1、InitGen2Tag

Function prototype：short InitGen2Tag(HANDLE hCom)

Function explanation：Initialize the EPC Gen2 tag

Input parameter：

- hCom— Handle

Return results: the result is zero, which shows the action is correct, others are wrong.
Explanation: It aims at the uninitialized tag. Please contact the tag supplier if the tag is initialized.

2、InitGen2EpcCode

Function prototype: short InitGen2EpcCode(HANDLE hCom,unsigned char MustInit,unsigned char * EpcCode)
Function explanation：Initialize the EPC code for EPC Gen2 tag
Input parameter：

- hCom— Handle

- MustInit：Whether the tag need to initialize parameters or not.. “1” for yes,”0” for no need.
- EpcCode：The EPC code which needs to be written into tag

Return results: the result is zero, which shows the action is correct, others are wrong.
Explanation: It contains “InitGen2Tag” function when calling the function of “InitGen2EpcCode”

3. SingleTagIdentify

Function prototype：short SingleTagIdentify(HANDLE hCom, unsigned int TagType, unsigned char *ID)
Function explanation： single tag identification
Input parameters：

- hCom— Handle
- TagType：Tag type identification, 4 for EPC code, 6 for TID code.
- ID：The returning ID value of identified tag

Return results: the result is zero, which shows the action is correct, others are wrong.

4. MultipleTagIdentify

Function prototype： short MultipleTagIdentify(HANDLE hCom, unsigned int TagType, unsigned int * Count，unsigned char *ID)
Function explanation：Multiple tag identification
Input parameters：

- hCom— Handle
- TagType：Tag type identification, 4 for EPC code, 6 for TID code.
- Count：The number of identified tags
- ID：The returning ID value of identified tags

Return results: the result is zero, which shows the action is correct, others are wrong.

5. Gen2ReadTag

Function prototype: short Gen2ReadTag(HANDLE hCom,unsigned char MemBank,unsigned Addr,unsigned char Length,unsigned char * Data)
Function explanation：Read the memory data of EPC Gen2 tag
Input parameters：

- hCom— Handle
- MemBank：Designated memory area： 1 for EPC, 2 for TID, 3 for USER
- Addr：Designated memory address needs to be read
- Length：Data length in designated memory area which needs to be read
- Data：The returning data value when read designated memory area.

Return results: the result is zero, which shows the action is correct, others are wrong.

Explanation: The calling of” Gen2ReadTag” function is related with tag, please contact the tag supplier if the tag with TID and USER.

6. Gen2WriteTag

Function prototype: short Gen2WriteTag(HANDLE hCom,unsigned char MemBank,unsigned Addr, unsigned

char Length ,unsigned char * Data)

Function explanation：Write data to the memory area of EPC Gen tag

Input parameters：

- hCom— Handle
- MemBank：Designated memory area： 1 for EPC, 3 for USER
- Addr：Designated memory address needs to be written
- Length：Double byte data length in designated memory area which needs to be written
- Data：The required writing data.

Return results: the result is zero, which shows the action is correct, others are wrong.

Explanation: The calling of" Gen2WriteTag" function is related with tag, please contact the tag supplier if the tag with USER area.

7、Gen2ReadTagWithEpc

Function prototype： short Gen2ReadTagWithEpc(HANDLE hCom,unsigned char MemBank,unsigned Addr,unsigned char ReadLen,unsigned char *ReadData,unsigned char Scope,unsigned char Length,unsigned char *MaskData)

Function explanation： Read the memory data of designated EPC Gen2 tag

Input parameters：

- hCom— Handle
- MemBank：Memory type
  Value is 1 --- Read the EPC memory data.
  Value is 3 --- Read the USER memory data.
- Addr：The start reading address
  EPC：Range 2~7,
  USER：Double byte address, starts from 0
- ReadLen：Double data byte length needs to be read
- ReadData：Data read
- Scope：value=1, comparing with EPC memory;
  Value=2, comparing with TID memory.
- Length：Byte length of compared data
- MaskData：The data needs to be compared; Scope value=1, stands for EPC code; Scope value=2, stands for TID code.

Return results: the result is zero, which shows the action is correct, others are wrong.

8、Gen2WriteTagWithEpc

Function prototype: short Gen2WriteTagWithEpc(HANDLE hCom,unsigned char MemBank,unsigned Addr,unsigned char WriteLen,unsigned char *WriteData,unsigned char Scope,unsigned char Length,unsigned char *MaskData)

Function explanation：Modify the EPC code for designated EPC Gen2 tag

Input parameters：

- hCom— Handle

- MemBank：Memory type
  Value is 1 --- Write data to EPC memory
  Value is 3 --- Write data to USER memory
- Addr：The start writing address
  EPC：Range 2~7,
  USER：Double byte address, starts from 0
- WriteLen：Double data byte length needs to be written,
- WriteData：Data needs to be written
- Scope：value=1, comparing with EPC memory;
  Value=2, comparing with TID memory.
- Length：Byte length of compared data
- MaskData：The data needs to be compared; Scope value=1, stands for EPC code; Scope value=2, stands for TID code.

Return results: the result is zero, which shows the action is correct, others are wrong.

9、Gen2ReadTagWithEpcPW

Function prototype: short Gen2ReadTagWithEpcPW(HANDLE hCom,unsigned int Password,unsigned char MemBank,unsigned Addr,unsigned char ReadLen,unsigned char *ReadData,unsigned char Scope,unsigned char Length,unsigned char *MaskData

Function explanation：Read the memory data of designated cryptographic tag

Input parameters：

- hCom— Handle
- Password：access password of tag
- MemBank：Memory type
  Value is 1 --- Read EPC memory data
  Value is 3 --- Read USER memory data
- Addr：The start reading address
  EPC：Range 2~7,
  USER：Double byte address, starts from 0
- ReadLen：Double data byte length needs to be read,
- ReadData：Data read
- Scope：value=1, comparing with EPC memory;
  Value=2, comparing with TID memory.
- Length：Byte length of compared data
- MaskData：The data needs to be compared; Scope value=1, stands for EPC code; Scope value=2, stands for TID code.

Return results: the result is zero, which shows the action is correct, others are wrong.

10、Gen2WriteTagWithEpcPW

Function prototype: short Gen2WriteTagWithEpcPW(HANDLE hCom,unsigned char Password,unsigned char MemBank,unsigned Addr,unsigned char WriteLen,unsigned char *WriteData,unsigned char Scope,unsigned char Length,unsigned char *MaskData)

Function explanation：Modify the EPC code of designated cryptographic tag

Input parameters：

- hCom— Handle
- Password：access password of tag
- MemBank：Memory type

  Value is 1 --- Write data to EPC memory

  Value is 3 --- Write data to USER memory
- Addr：The start writing address

  EPC：Range 2~7,

  USER：Double byte address, starts from 0
- WriteLen：Double data byte length needs to be written,
- WriteData：Data needs to be written
- Scope：value=1, comparing with EPC memory;

  Value=2, comparing with TID memory.
- Length：Byte length of compared data
- MaskData：The data needs to be compared; Scope value=1, stands for EPC code; Scope value=2, stands for TID code.

Return results: the result is zero, which shows the action is correct, others are wrong.

11、Gen2SetAccessPassword

Function prototype: short Gen2SetAccessPassword(HANDLE hCom,unsigned int Password,unsigned int Password2,unsigned char Scope,unsigned char Length,unsigned char *MaskData)

Function explanation：Modify the access password for designated tag.

Input parameters：

- hCom— Handle
- Password：original access password of tag
- Password2：New access password needs to be configured
- Scope：value=1, comparing with EPC memory;

  Value=2, comparing with TID memory.
- Length：Byte length of compared data
- MaskData：The data needs to be compared; Scope value=1, stands for EPC code; Scope value=2, stands for TID code.

Return results: the result is zero, which shows the action is correct, others are wrong.

12、Gen2GetAccessPassword

Function prototype: short Gen2GetAccessPassword(HANDLE hCom,unsigned int Password,unsigned int * Password2,unsigned char Scope,unsigned char Length,unsigned char *MaskData)

Function: Read the access password of designated tag.

Input parameters：

- hCom— Handle
- Password：original access password of tag
- Password2：access password read

- Scope：value=1, comparing with EPC memory;
  Value=2, comparing with TID memory.
- Length：Byte length of compared data
- MaskData：The data needs to be compared; Scope value=1, stands for EPC code; Scope value=2, stands for TID code.

Return results: the result is zero, which shows the action is correct, others are wrong.

13. Gen2SetKillPassword

Function prototype： short Gen2SetKillPassword(HANDLE hCom,unsigned int Password,unsigned int Password2,unsigned char Scope,unsigned char Length,unsigned char *MaskData)

Function explanation： modify the kill password for designated tag

Input parameters：

- hCom— Handle
- Password：Access password of tag
- Password2：The kill password needs to be configured
- Scope：value=1, comparing with EPC memory;
  Value=2, comparing with TID memory.
- Length：Byte length of compared data
- MaskData：The data needs to be compared; Scope value=1, stands for EPC code; Scope value=2, stands for TID code.

Return results: the result is zero, which shows the action is correct, others are wrong.

14、Gen2GetKillPassword

Function prototype： short Gen2GetKillPassword(HANDLE hCom,unsigned int Password,unsigned int * Password2,unsigned char Scope,unsigned char Length,unsigned char *MaskData)

Function explanation：Read the kill password of designated tag

Input parameters：

- hCom— Handle
- Password：Access password of tag
- Password2：The kill password read
- Scope：value=1, comparing with EPC memory;
  Value=2, comparing with TID memory.
- Length：Byte length of compared data
- MaskData：The data needs to be compared; Scope value=1, stands for EPC code; Scope value=2, stands for TID code.

Return results: the result is zero, which shows the action is correct, others are wrong.

15、Gen2KillTag

Function prototype：short Gen2KillTag(HANDLE hCom,unsigned int Password,unsigned char Scope,unsigned char Length,unsigned char *MaskData)

Function explanation：Kill the designated tag

Input parameters：

- hCom— Handle

- Password：The kill password of tag
- Scope：value=1, comparing with EPC memory;
  Value=2, comparing with TID memory.
- Length：Byte length of compared data
- MaskData：The data needs to be compared; Scope value=1, stands for EPC code; Scope value=2, stands for TID code.

Return results: the result is zero, which shows the action is correct, others are wrong.

16、Gen2LockTag

Function prototype： short Gen2LockTag(HANDLE hCom,unsigned int Password, Gen2LockFlags* LockFlag,unsigned char Scope,unsigned char Length,unsigned char *MaskData)

Function explanation：lock the designated tag

Input parameters：

- hCom— Handle
- Password：The access password of tag
- Gen2LockFlags： Operating scope and content needs to be locked. It's a data structure.
- Scope：value=1, comparing with EPC memory;
  Value=2, comparing with TID memory.
- Length：Byte length of compared data
- MaskData：The data needs to be compared; Scope value=1, stands for EPC code; Scope value=2, stands for TID code.

Return results: the result is zero, which shows the action is correct, others are wrong.

17、Gen2ChangeEAS

Function prototype： short Gen2ChangeEAS(HANDLE hCom,unsigned int Password,unsigned char EASFlag,unsigned char Scope,unsigned char Length,unsigned char *MaskData)

Function explanation：Modify the EAS bit for designated tag

Input parameters：

- hCom— Handle
- Password—Modify the access password for designated tag
- EASFlag： The EAS mark bit needs to be configured
- Scope：value=1, comparing with EPC memory;
  Value=2, comparing with TID memory.
- Length：Byte length of compared data
- MaskData：The data needs to be compared; Scope value=1, stands for EPC code; Scope value=2, stands for TID code.

Return results: the result is zero, which shows the action is correct, others are wrong.

Explanation: Please contact the tag supplier if the tag with EAS bit.

18、Gen2EASAlarm

Function prototype：short Gen2EASAlarm(HANDLE hCom)

Function explanation：Tag alarm

Input parameters：

- hCom— Handle

Return results: the result is zero, which means giving alarm, others are wrong.

19、Gen2TagContentList

Function prototype：short Gen2TagContentList(HANDLE hCom, Gen2TagList * Data)

Function explanation：List the selected contents of EPC Gen2 tag

Input parameters：

- hCom— Handle
- Data—*Gen2TagList* structure, explain for the required listing tag parts and returned tag data. For details, please refer to the definition of “Gen2TagList”

Return results: the result is zero, which shows the action is correct, others are wrong.