**TINY Application Programming Interface(HID)**

# What’s Inside

[What’s Inside 1](#_Toc33793963)

[1.Introduction 3](#_Toc33793964)

[2.General Command Set 4](#_Toc33793965)

[2.1 UhfReaderConnect () 4](#_Toc33793966)

[2.2 UhfReaderDisconnect () 5](#_Toc33793967)

[2.3 UhfOpenPort () 6](#_Toc33793968)

[2.4 UhfClosePort () 7](#_Toc33793969)

[2.5 UhfGetPaStatus () 8](#_Toc33793970)

[2.6 UhfGetPower () 9](#_Toc33793971)

[2.7 UhfSetPower () 10](#_Toc33793972)

[2.8 UhfGetFrequency () 11](#_Toc33793973)

[2.9 UhfSetFrequency () 12](#_Toc33793974)

[2.10 UhfGetRegister () 13](#_Toc33793975)

[2.11 UhfSetRegister () 14](#_Toc33793976)

[2.12 UhfResetRegister () 15](#_Toc33793977)

[2.13 UhfSaveRegister () 16](#_Toc33793978)

[2.14 UhfGetVersion () 17](#_Toc33793979)

[2.15 UhfGetReaderUID () 18](#_Toc33793980)

[2.16 UhfEnterSleepMode () 19](#_Toc33793981)

[2.17 UhfStopOperation () 20](#_Toc33793982)

[3.Inventory Command Set 21](#_Toc33793983)

[3.1 UhfStartInventory () 21](#_Toc33793984)

[3.2 UhfReadInventory () 22](#_Toc33793985)

[3.3 UhfInventorySingleTag () 23](#_Toc33793986)

[4.Access Command Set 24](#_Toc33793987)

[4.1 UhfReadDataByEPC () 24](#_Toc33793988)

[4.2 UhfReadDataFromSingleTag () 25](#_Toc33793989)

[4.3 UhfReadMaxDataByEPC () 26](#_Toc33793990)

[4.4 UhfReadMaxDataFromSingleTag () 27](#_Toc33793991)

[4.5 UhfWriteDataByEPC () 28](#_Toc33793992)

[4.6 UhfWriteDataToSingleTag () 29](#_Toc33793993)

[4.7 UhfBlockWriteDataByEPC () 30](#_Toc33793994)

[4.8 UhfBlockWriteDataToSingleTag() 31](#_Toc33793995)

[4.9 UhfEraseDataByEPC () 32](#_Toc33793996)

[4.10 UhfEraseDataFromSingleTag () 33](#_Toc33793997)

[4.11 UhfLockMemByEPC () 34](#_Toc33793998)

[4.12 UhfLockMemFromSingleTag () 35](#_Toc33793999)

[4.13 UhfKillTagByEPC () 36](#_Toc33794000)

[4.14 UhfKillSingleTag () 37](#_Toc33794001)

[4.15 UhfBlockWriteEPCByEPC() 38](#_Toc33794002)

[4.16 UhfBlockWriteEPCToSingleTag() 39](#_Toc33794003)

[4.17 UhfStartReadDataFromMultiTag () 40](#_Toc33794004)

[4.18 UhfGetDataFromMultiTag () 41](#_Toc33794005)

[5.Firmware Upgrate Command Set 42](#_Toc33794006)

[5.1 UhfUpdateInit () 42](#_Toc33794007)

[5.2 UhfUpdateSendRN32 () 43](#_Toc33794008)

[5.3 UhfUpdateSendSize () 44](#_Toc33794009)

[5.4 UhfUpdateSendData () 45](#_Toc33794010)

[5.5 UhfUpdateCommit () 46](#_Toc33794011)

[6.Other Command Set 47](#_Toc33794012)

[6.1 LockGenCode () 47](#_Toc33794013)

[6.2 UhfSearchHids () 48](#_Toc33794014)

[Appendix A : Description of the frequency parameters 49](#_Toc33794015)

[Appendix B : Instructions of the parameters about tag operation 51](#_Toc33794016)

# 1.Introduction

The document describes functions of RLM, includes Introduction, general command set, inventory command set, access command set, firmware upgrade command set, other command set and appendix.

HID is short for Human Interface Device, TINYis a UHF reader which uses HID to communicate. Users can use these functions following the steps bellow.

Firstly, connect TINYto a computer.

Secondly, call the function called UhfSearchHids to search HID serial number available on the computer.

Thirdly, select a serial number and use it as a parameter called cPort of function called UhfReaderConnect or UhfOpenPort to establish a connection for communication.

Lastly, use the value of hCom as a parameter of other functions.

# 2.General Command Set

## 2.1 UhfReaderConnect ()

**2.1.1 brief**

 Establish a connection with the RLM.

**2.1.2 Prototype**

int WINAPI UhfReaderConnect (HANDLE &hCom, char\* cPort, UCHAR flagCrc);

**2.1.3 Returns**

1 : success;

0 : fail;

**2.1.4 Parameters**

HANDLE &hCom : [out], handle for communication port;

char\* cPort : [in], number of HID (such as \\?\hid#vid\_0483&pid\_5750#6&20bfa97&0&0000#{4d1e55b2-f16f-11cf-88cb-001111000030});

UCHAR flagCrc : [in], using CRC16 or not, the value represents as the followings:

0x00 : use;

0x01 : not use;

## 2.2 UhfReaderDisconnect ()

**2.2.1 brief**

 Close the connection.

**2.2.2 Prototype**

 int WINAPI UhfReaderDisconnect (HANDLE &hCom, UCHAR flagCrc);

**2.2.3 Returns**

1 : success;

0 : fail;

**2.2.4 Parameters**

HANDLE &hCom : [out], handle for communication port;

UCHAR flagCrc : [in], using CRC or not,

0 : not use;

1 : use.

## 2.3 UhfOpenPort ()

**2.3.1 brief**

 Establish a connection with the RLM.

**2.3.2 Prototype**

 int WINAPI UhfOpenPort (HANDLE &hCom, char\* cPort, UCHAR flagCrc);

**2.3.3 Returns**

1 : success;

0 : fail;

**2.3.4 Parameters**

HANDLE &hCom : [out], handle for communication port;

char\* cPort : [in], number of HID (such as \\?\hid#vid\_0483&pid\_5750#6&20bfa97&0&0000#{4d1e55b2-f16f-11cf-88cb-001111000030});

 UCHAR flagCrc: [in], using CRC16 or not, the value represents as the followings:

0x00 : use;

0x01 : not use;

## 2.4 UhfClosePort ()

**2.4.1 brief**

 Close the connection.

**2.4.2 Prototype**

 int WINAPI UhfClosePort ();

**2.4.3 Returns**

1 : success;

0 : fail;

**2.4.4 Parameters**

None.

## 2.5 UhfGetPaStatus ()

**2.5.1 brief**

 Read RLM connection status.

**2.5.2 Prototype**

 int WINAPI UhfGetPaStatus (HANDLE hCom, UCHAR\* uStatus, UCHAR flagCrc);

**2.5.3 Returns**

1 : success;

0 : fail;

**2.5.4 Parameters**

HANDLE hCom : [in], handle for communication port;

UCHAR\* uStatus : [out] , connection status（the length of the array is 1, 0 : connect; other : disconnect）;

UCHAR flagCrc : [in] , using CRC16 or not ,

0 : not use;

1 : use.

## 2.6 UhfGetPower ()

**2.6.1 brief**

 Read RLM connection status.

**2.6.2 Prototype**

 int WINAPI UhfGetPower (HANDLE hCom, UCHAR\* uPower, UCHAR flagCrc);

**2.6.3 Returns**

1 : success;

0 : fail;

**2.6.4 Parameters**

HANDLE hCom : [in], handle for communication port;

UCHAR\* uPower : [out], value of power (the length of the array is 1);

UCHAR flagCrc : [in] , using CRC16 or not ,

0 : not use;

1 : use.

## 2.7 UhfSetPower ()

**2.7.1 brief**

 Set power of RLM.

**2.7.2 Prototype**

 int WINAPI UhfSetPower (HANDLE hCom, UCHAR uOption, UCHAR uPower, UCHAR flagCrc);

**2.7.3 Returns**

1 : success;

0 : fail;

**2.7.4 Parameters**

HANDLE hCom : [in], handle for communication port;

UCHAR uOption : [in], must be set to 0x01;

UCHAR uPower : [in], value to be set;

UCHAR flagCrc : [in], using CRC16 or not ,

0 : not use;

1 : use.

## 2.8 UhfGetFrequency ()

**2.8.1 brief**

 Get frequency of RLM.

**2.8.2 Prototype**

 int WINAPI UhfGetFrequency (HANDLE hCom, UCHAR\* uFreMode, UCHAR\* uFreBase, UCHAR\* uBaseFre, UCHAR\* uChannNum, UCHAR\* uChannSpc, UCHAR\* uFreHop, UCHAR flagCrc);

**2.8.3 Returns**

1 : success;

0 : fail;

**2.8.4 Parameters**

HANDLE hCom : [in], handle for communication port;

UCHAR\* uFreMode : [out] , mode of frequency (the array of length is 1);

UCHAR\* uFreBase : [out] , frequency base (the array of length is 1);

UCHAR\* uBaseFre: [out] , starting frequency (the array of length is 2);

UCHAR\* uChannNum : [out] , count of channels (the array of length is 1);

UCHAR\* uChannSpc : [out] , base of channel's bandwidth (the array of length is 1);

UCHAR\* uFreHop : [out] , mode of hopping (the array of length is 1);

UCHAR flagCrc : [in] , using CRC16 or not ,

0 : not use;

1 : use.

## 2.9 UhfSetFrequency ()

**2.9.1 brief**

 Set frequency of RLM.

**2.9.2 Prototype**

 int WINAPI UhfSetFrequency(HANDLE hCom, UCHAR uFreMode, UCHAR uFreBase, UCHAR\* uBaseFre, UCHAR uChannNum, UCHAR uChannSpc, UCHAR uFreHop, UCHAR flagCrc);

**2.9.3 Returns**

1 : success;

0 : fail;

**2.9.4 Parameters**

HANDLE hCom : [in], handle for communication port;

UCHAR uFreMode : [in] , mode of frequency ;

UCHAR uFreBase : [in] , frequency base;

UCHAR\* uBaseFre : [in] , starting frequency (the length of the array is 2);

UCHAR uChannNum : [in] , count of channels;

UCHAR uChannSpc : [in] , base of channel's bandwidth;

UCHAR uFreHop: [in] , mode of hopping;

byte flagCrc : [in] , using CRC16 or not ,

0 : not use;

1 : use.

## 2.10 UhfGetRegister ()

**2.10.1 brief**

 Read data from the specified address of register.

**2.10.2 Prototype**

int WINAPI UhfGetRegister (HANDLE hCom, int RADD, int RLEN, UCHAR\* STATUS, UCHAR\* REG, UCHAR flagCrc)

**2.10.3 Returns**

1 : success;

0 : fail;

**2.10.4 Parameters**

HANDLE hCom : [in], handle for communication port;

int RADD: [in] , address where you read;

int RLEN: [in] , how many registers you want to read;

UCHAR\* STATUS : [out] , the status of implementation (the length of the array is 1);

UHCAR\* REG: [out] , data you have read from registers;

UCHAR flagCrc: [in] , using CRC16 or not ,

0 : not use;

1 : use.

## 2.11 UhfSetRegister ()

**2.11.1 brief**

 Write data to the specified address of register.

**2.11.2 Prototype**

 int WINAPI UhfSetRegister (HANDLE hCom, int RADD, int RLEN, UCHAR\* REG, UCHAR\* STATUS, UCHAR flagCrc);

**2.11.3 Returns**

1 : success;

0 : fail;

**2.11.4 Parameters**

HANDLE hCom : [in], handle for communication port;

int RADD : [in] , address where you write;

int RLEN : [in] , how many registers you want to write;

UHCAR\* REG : [in] , data you want to write;

UCHAR\* STATUS : [out] , the status of implementation (the length of the array is 1);

UCHAR flagCrc : [in] , using CRC16 or not ,

0 : not use;

1 : use.

## 2.12 UhfResetRegister ()

**2.12.1 brief**

 Reset all values to factory defaults.

**2.12.2 Prototype**

 int WINAPI UhfResetRegister (HANDLE hCom, UCHAR flagCrc);

**2.12.3 Returns**

1 : success;

0 : fail;

**2.12.4 Parameters**

HANDLE hCom : [in], handle for communication port;

UCHAR flagCrc : [in] , using CRC16 or not ,

0 : not use;

1 : use.

## 2.13 UhfSaveRegister ()

**2.13.1 brief**

 Save the current settings to the corresponding registers.

**2.13.2 Prototype**

 int WINAPI UhfSaveRegister (HANDLE hCom, UCHAR flagCrc);

**2.13.3 Returns**

1 : success;

0 : fail;

**2.13.4 Parameters**

HANDLE hCom : [in], handle for communication port;

UCHAR flagCrc : [in] , using CRC16 or not ,

0 : not use;

1 : use.

## 2.14 UhfGetVersion ()

**2.14.1 brief**

 Read hardware serial number and software version number of RLM.

**2.14.2 Prototype**

 int WINAPI UhfGetVersion (HANDLE hCom, UCHAR\* uSerial, UCHAR\* uVersion, UCHAR flagCrc);

**2.14.3 Returns**

1 : success;

0 : fail;

**2.14.4 Parameters**

HANDLE hCom : [in], handle for communication port;

UCHAR\* uSerial : [out] , hardware serial number (the length of the array is 6);

UCHAR\* uVersion: [out] , software version number (the length of the array is 3);

UCHAR flagCrc : [in] , using CRC16 or not ,

0 : not use;

1 : use.

## 2.15 UhfGetReaderUID ()

**2.15.1 brief**

 Read the RLM’s uid that is a unique identifying number.

**2.15.2 Prototype**

int WINAPI UhfGetReaderUID (HANDLE hCom, UCHAR\* uUid, UCHAR flagCrc);

**2.15.3 Returns**

1 : success;

0 : fail;

**2.15.4 Parameters**

HANDLE hCom : [in], handle for communication port;

UCHAR\* uUid : [out] , uid（the length of the array is 12）;

UCHAR flagCrc : [in] , using CRC16 or not ,

0 : not use;

1 : use.

## 2.16 UhfEnterSleepMode ()

**2.16.1 brief**

 Enter sleep mode, but you can wake it up by sending any command.

**2.16.2 Prototype**

int WINAPI UhfEnterSleepMode (HANDLE hCom, UCHAR flagCrc);

**2.16.3 Returns**

1 : success;

0 : fail;

**2.16.4 Parameters**

HANDLE hCom : [in], handle for communication port;

UCHAR flagCrc: [in] , using CRC16 or not ,

0 : not use;

1 : use.

## 2.17 UhfStopOperation ()

**2.17.1 brief**

 Stop working and put RLM on standby.

**2.17.2 Prototype**

 int WINAPI UhfStopOperation (HANDLE hCom, UCHAR flagCrc);

**2.17.3 Returns**

1 : success;

0 : fail;

**2.17.4 Parameters**

HANDLE hCom : [in], handle for communication port;

UCHAR flagCrc: [in], using CRC16 or not ,

0 : not use;

1 : use.

# 3.Inventory Command Set

## 3.1 UhfStartInventory ()

**3.1.1 brief**

 Start to read uii in loop mode.

**3.1.2 Prototype**

 int WINAPI UhfStartInventory (HANDLE hCom, UCHAR flagAnti, UCHAR initQ, UCHAR flagCrc);

**3.1.3 Returns**

1 : success;

0 : fail;

**3.1.4 Parameters**

HANDLE hCom: [in], handle for communication port;

UCHAR flagAnti : [in] , mode of loop (1 : Anti-collision; 0 : single tag loop);

UCHAR initQ : [in] , Q value (This parameter valid when flagAnti is 1);

UCHAR flagCrc : [in] , using CRC16 or not ,

0 : not use;

1 : use.

Note:

UhfStartInventory () function is used to start to read uii in loop mode. when flagAnti=0, the RLM will read uii of one tag in loop mode. when flagAnti=1, the RLM will read uii of several tags in loop mode, which is called anti-collision. You should use this function first so that RLM uploads uiis to the buffer, then you use UhfReadInventory() to read uiis from the buffer, you must use UhfStopOperation() to stop the loop in the last.

## 3.2 UhfReadInventory ()

**3.2.1 brief**

 Read uii from buffer that RLM upload.

**3.2.2 Prototype**

 int WINAPI UhfReadInventory (HANDLE hCom, UCHAR\* uLenUii, UCHAR\* uUii);

**3.2.3 Returns**

1 : success;

0 : fail;

**3.2.4 Parameters**

HANDLE hCom: [in], handle for communication port;

UCHAR\* uLenUii : [out] , count of bytes in uii (the length of the array is 1);

UCHAR\* uUii: [out] , uii of a tag.

## 3.3 UhfInventorySingleTag ()

**3.3.1 brief**

 Read uii(including PC and EPC).

**3.3.2 Prototype**

 int WINAPI UhfInventorySingleTag (HANDLE hCom, UCHAR\* uLenUii, UCHAR\* uUii , UCHAR flagCrc);

**3.3.3 Returns**

1 : success;

0 : fail;

**3.3.4 Parameters**

HANDLE hCom: [in], handle for communication port;

UCHAR\* uLenUii: [out] , count of bytes in uii ( the length of array is 1);

UCHAR\* uUii: [out] , uii of a tag;

UCHAR flagCrc: [in] , using CRC16 or not ,

0 : not use;

1 : use.

# 4.Access Command Set

## 4.1 UhfReadDataByEPC ()

**4.1.1 brief**

 Read data from a specified tag with uii.

**4.1.2 Prototype**

 int WINAPI UhfReadDataByEPC (HANDLE hCom, UCHAR\* uAccessPwd, UCHAR uBank, UCHAR\* uPtr, UCHAR uCnt, UCHAR\* uUii, UCHAR\* uReadData, UCHAR\* uErrorCode, UCHAR flagCrc);

**4.1.3 Returns**

1 : success;

0 : fail;

**4.1.4 Parameters**

HANDLE hCom: [in], handle for communication port;

UCHAR\* uAccessPwd: [in] , access password stored in reserved area（refer to Appendix D）;

UCHAR uBank : [in] , storage area of a tag (refer to Appendix D);

UCHAR\* uPtr : [in] , address where you start read (refer to Appendix D);

UCHAR uCnt: [in] , how many words you want to read;

UCHAR\* uUii : [in] , uii of a tag;

UCHAR\* uReadData : [out] , data you have read(refer to Appendix D);

UCHAR\* uErrorCode: [out] , error code(refer to Appendix D);

UCHAR flagCrc: [in] , using CRC16 or not ,

0 : not use;

1 : use.

## 4.2 UhfReadDataFromSingleTag ()

**4.2.1 brief**

 Read uii from a tag without uii.

**4.2.2 Prototype**

 int WINAPI UhfReadDataFromSingleTag (HANDLE hCom, UCHAR\* uAccessPwd, UCHAR uBank, UCHAR\* uPtr, UCHAR uCnt, UCHAR\* uReadData, UCHAR\* uUii, UCHAR\* uLenUii, UCHAR\* uErrorCode, UCHAR flagCrc);

**4.2.3 Returns**

1 : success;

0 : fail;

**4.2.4 Parameters**

HANDLE hCom: [in], handle for communication port;

UCHAR\* uAccessPwd : [in] , access password stored in reserved area (refer to Appendix D);

UCHAR uBank : [in] , storage area of a tag (refer to Appendix D);

UCHAR\* uPtr: [in] , address where you start read (refer to Appendix D);

UCHAR uCnt : [in] , how many words you want to read;

UCHAR\* uReadData: [out] , data you have read (refer to Appendix D);

UCHAR\* uUii: [out] , uii of tag that you read from;

UCHAR\* uLenUii : [out] , how many bytes in uii;

UCHAR\* uErrorCode : error code (refer to Appendix D);

UCHAR flagCrc: [in] , using CRC16 or not ,

0 : not use;

1 : use.

## 4.3 UhfReadMaxDataByEPC ()

**4.3.1 brief**

 Read all words starting at the specified address from a specified tag with uii.

**4.3.2 Prototype**

int WINAPI UhfReadMaxDataByEPC (HANDLE hCom, UCHAR\* uAccessPwd, UCHAR uBank, UCHAR\* uPtr, UCHAR\* uUii, UCHAR\* Data\_len, UCHAR\* uReadData, UCHAR\* uErrorCode, UCHAR flagCrc);

**4.3.3 Returns**

1 : success;

0 : fail;

**4.3.4 Parameters**

HANDLE hCom: [in], handle for communication port;

UCHAR\* uAccessPwd : [in] , access password stored in reserved area (refer to Appendix D);

UCHAR uBank : [in] , storage area of a tag (refer to Appendix D);

UCHAR\* uPtr : [in] , address where you start read (refer to Appendix D);

UCHAR\* uUii : [in] , uii of a tag;

UCHAR\* Data\_len : [out] , how many words you hava read.

UCHAR\* uReadData: [out] , data you have read (refer to Appendix D);

UCHAR\* uErrorCode : [out] , error code (refer to Appendix D);

UCHAR flagCrc : [in] , using CRC16 or not ,

0 : not use;

1 : use.

Warning:

RLM100 can read no more than 90 words from a tag, RLM200 and RLM300 can read no more than 233 words from a tag. If count of the remaining words in the bank from the address is more than maximum RLM can support, function will return false;

## 4.4 UhfReadMaxDataFromSingleTag ()

**4.4.1 brief**

 Read all words starting at the specified address from a tag without uii.

**4.4.2 Prototype**

int WINAPI UhfReadMaxDataFromSingleTag (HANDLE hCom, UCHAR\* uAccessPwd, UCHAR uBank, UCHAR\* uPtr, UCHAR\* Data\_len, UCHAR\* uReadData, UCHAR\* uUii, UCHAR\* uLenUii, UCHAR\* uErrorCode, UCHAR flagCrc);

**4.4.3 Returns**

1 : success;

0 : fail;

**4.4.4 Parameters**

HANDLE hCom: [in], handle for communication port;

UCHAR\* uAccessPwd : [in] , access password stored in reserved area (refer to Appendix D);

UCHAR uBank : [in] , storage area of a tag (refer to Appendix D);

UCHAR\* uPtr: [in] , address where you start read (refer to Appendix D);

UCHAR\* Data\_len : [out] , how many bytes you have read;

UCHAR\* uReadData: [out] , data you have read;

UCHAR\* uUii: [out] , uii of tag that you read from;

UCHAR\* uLenUii: [out] , how many bytes in uii;

UCHAR\* uErrorCode : [out] , error code (refer to Appendix D);

UCHAR flagCrc : [in] , using CRC16 or not ,

0 : not use;

1 : use.

Warning:

RLM100 can read no more than 90 words from a tag, RLM200 and RLM300 can read no more than 233 words from a tag. If count of the remaining words in the bank from the address is more than maximum RLM can support, function will return false;

## 4.5 UhfWriteDataByEPC ()

**4.5.1 brief**

 Write one word to a specified tag with uii.

**4.5.2 Prototype**

 int WINAPI UhfWriteDataByEPC (HANDLE hCom, UCHAR\* uAccessPwd, UCHAR uBank, UCHAR\* uPtr, UCHAR uCnt, UCHAR\* uUii, UCHAR\* uWriteData, UCHAR\* uErrorCode, UCHAR flagCrc);

**4.5.3 Returns**

1 : success;

0 : fail;

**4.5.4 Parameters**

HANDLE hCom: [in], handle for communication port;

UCHAR\* uAccessPwd : [in] , access password stored in reserved area (refer to Appendix D);

UCHAR uBank : [in] , storage area of a tag (refer to Appendix D);

UCHAR\* uPtr : [in] , address where you start write (refer to Appendix D);

UCHAR uCnt: [in] , how many words you want to write（bCnt must be 1 here）;

UCHAR\* uUii: [in] , uii of a tag;

UCHAR\* uWriteData : [in] , data you want to write(refer to Appendix D);

UCHAR\* uErrorCode : [out] , error code(refer to Appendix D)

UCHAR flagCrc: [in] , using CRC16 or not ,

0 : not use;

1 : use.

## 4.6 UhfWriteDataToSingleTag ()

**4.6.1 brief**

 Write one word to a tag without uii.

**4.6.2 Prototype**

 int WINAPI UhfWriteDataToSingleTag (HANDLE hCom, UCHAR\* uAccessPwd, UCHAR uBank, UCHAR\* uPtr, UCHAR uCnt, UCHAR\* uWriteData, UCHAR\* uUii, UCHAR\* uLenUii, UCHAR\* uErrorCode, UCHAR flagCrc);

**4.6.3 Returns**

1 : success;

0 : fail;

**4.6.4 Parameters**

HANDLE hCom: [in], handle for communication port;

UCHAR\* uAccessPwd : [in] , access password stored in reserved area (refer to Appendix D);

UCHAR uBank: [in] , storage area of a tag (refer to Appendix D);

UCHAR\* uPtr : [in] , address where you start write (refer to Appendix D);

UCHAR uCnt : [in] , how many words you want to write;

UCHAR\* uWriteData : [in] , data you want to write;

UCHAR\* uUii : [out] , uii of tag that you want to write;

UCHAR\* uLenUii : [out] , how many bytes in uii;

UCHAR\* uErrorCode : [out] , error code (refer to Appendix D);

UCHAR flagCrc : [in] , using CRC16 or not ,

0 : not use;

1 : use.

## 4.7 UhfBlockWriteDataByEPC ()

**4.7.1 brief**

 Write several words to a specified tag with uii.

**4.7.2 Prototype**

int WINAPI UhfBlockWriteDataByEPC (HANDLE hCom, UCHAR\* uAccessPwd, UCHAR uBank, UCHAR\* uPtr, UCHAR uCnt, UCHAR\* uUii, UCHAR\* uWriteData, UCHAR\* uErrorCode, UCHAR\* uStatus, UCHAR\* uWritedLen, UCHAR\* RuUii, UCHAR flagCrc);

**4.7.3 Returns**

1 : success;

0 : fail;

**4.7.4 Parameters**

HANDLE hCom: [in], handle for communication port;

UCHAR\* uAccessPwd : [in] , access password stored in reserved area (refer to Appendix D);

UCHAR uBank : [in] , storage area of a tag (refer to Appendix D);

UCHAR\* uPtr : [in] , address where you start write (refer to Appendix D);

UCHAR uCnt: [in] , how many words you want to write;

UCHAR\* uUii : [in] , uii of a tag;

UCHAR\* uWriteData : [in] , data you want to write;

UCHAR\* uErrorCode : [out] , error code (refer to Appendix D);

UCHAR\* uStatus : [out] , the status of implementation (the length of the array is 1);

UCHAR\* uWritedLen : [out] , how many words have been written;

UCHAR\* RuUii: [out] , uii of a tag;

UCHAR flagCrc : [in] , using CRC16 or not ,

0 : not use;

1 : use.

## 4.8 UhfBlockWriteDataToSingleTag()

**4.8.1 brief**

 Write several words to a tag without uii.

**4.8.2 Prototype**

int WINAPI UhfBlockWriteDataToSingleTag (HANDLE hCom, UCHAR\* uAccessPwd, UCHAR uBank, UCHAR\* uPtr, UCHAR uCnt, UCHAR\* uWriteData, UCHAR\* uUii, UCHAR\* uLenUii, UCHAR\* uErrorCode, UCHAR\* uStatus,UCHAR\* uWritedLen, UCHAR flagCrc);

**4.8.3 Returns**

1 : success;

0 : fail;

**4.8.4 Parameters**

HANDLE hCom: [in], handle for communication port;

UCHAR\* uAccessPwd : [in] , access password stored in reserved area (refer to Appendix D);

UCHAR uBank : [in] , storage area of a tag (refer to Appendix D);

UCHAR\* uPtr : [in] , address where you start write (refer to Appendix D);

UCHAR uCnt : [in] , how many words you want to write;

UCHAR\* uWriteData : [in] , data you have write (refer to Appendix D);

UCHAR\* uUii : [out] , uii of a tag;

UCHAR\* uLenUii : [out] , how many bytes in uii;

UCHAR\* uErrorCode : [out] , error code (refer to Appendix D);

UCHAR\* uStatus : [out] , the status of implementation (the length of the array is 1);

UCHAR\* uWritedLen : [out] , how many words have been written;

UCHAR flagCrc : [in] , using CRC16 or not ,

0 : not use;

1 : use.

## 4.9 UhfEraseDataByEPC ()

**4.9.1 brief**

 Erase data in a bank of a specified tag with uii.

**4.9.2 Prototype**

 int WINAPI UhfEraseDataByEPC (HANDLE hCom, UCHAR\* uAccessPwd, UCHAR uBank, UCHAR\* uPtr, UCHAR uCnt, UCHAR\* uUii, UCHAR\* uErrorCode, UCHAR flagCrc);

**4.9.3 Returns**

1 : success;

0 : fail;

**4.9.4 Parameters**

HANDLE hCom: [in], handle for communication port;

UCHAR\* uAccessPwd : [in] , access password stored in reserved area (refer to Appendix D);

UCHAR uBank : [in] , storage area of a tag (refer to Appendix D);

UCHAR\* uPtr : [in] , address where you start erase (refer to Appendix D);

UCHAR uCnt : [in] , how many words you want to erase;

UCHAR\* uUii : [in] , uii of a tag;

UCHAR\* uErrorCode : [out] , error code(refer to Appendix D);

UCHAR flagCrc : [in] , using CRC16 or not ,

0 : not use;

1 : use.

## 4.10 UhfEraseDataFromSingleTag ()

**4.10.1 brief**

 Erase data in a bank of the tag without Uii.

**4.10.2 Prototype**

int WINAPI UhfEraseDataFromSingleTag (HANDLE hCom, UCHAR\* uAccessPwd, UCHAR uBank, UCHAR\* uPtr, UCHAR uCnt, UCHAR\* uUii, UCHAR\* uErrorCode, UCHAR flagCrc);

**4.10.3 Returns**

1 : success;

0 : fail;

**4.10.4 Parameters**

HANDLE hCom: [in], handle for communication port;

UCHAR\* uAccessPwd : [in] , access password stored in reserved area (refer to Appendix D);

UCHAR uBank : [in] , storage area of a tag (refer to Appendix D);

UCHAR\* uPtr : [in] , address where you start read (refer to Appendix D);

UCHAR uCnt : [in] , how many words you want to erase;

UCHAR\* uUii : [out] , uii of tag that you erase;

UCHAR\* uErrorCode : [out] , error code (refer to Appendix D);

UCHAR flagCrc : [in] , using CRC16 or not ,

0 : not use;

1 : use.

## 4.11 UhfLockMemByEPC ()

**4.11.1 brief**

 Lock a specified tag with uii.

**4.11.2 Prototype**

 int WINAPI UhfLockMemByEPC (HANDLE hCom, UCHAR\* uAccessPwd, UCHAR\* uLockData, UCHAR\* uUii, UCHAR\* uErrorCode, UCHAR flagCrc);

**4.11.3 Returns**

1 : success;

0 : fail;

**4.11.4 Parameters**

HANDLE hCom: [in], handle for communication port;

UCHAR\* uAccessPwd : [in] , access password stored in reserved area (refer to Appendix D);

UCHAR\* uLockData : [in] , lock code which means how you lock (refer to Appendix D);

UCHAR\* uUii : [in] , uii of a tag;

UCHAR\* uErrorCode : [out] ,error code(refer to Appendix D);

UCHAR flagCrc : [in] , using CRC16 or not ,

0 : not use;

1 : use.

## 4.12 UhfLockMemFromSingleTag ()

**4.12.1 brief**

 Lock a tag without uii.

**4.12.2 Prototype**

int WINAPI UhfLockMemFromSingleTag (HANDLE hCom, UCHAR\* uAccessPwd, UCHAR\* uLockData, UCHAR\* uUii, UCHAR\* uErrorCode, UCHAR flagCrc);

**4.12.3 Returns**

1 : success;

0 : fail;

**4.12.4 Parameters**

HANDLE hCom: [in], handle for communication port;

UCHAR\* uAccessPwd : [in] , access password stored in reserved area (refer to Appendix D);

UCHAR\* uLockData : [in] , lock code which means how you lock (refer to Appendix D);

UCHAR\* uUii : [out] , uii of tag that you lock;

UCHAR\* uErrorCode : [out] , error code (refer to Appendix D);

UCHAR flagCrc : [in] , using CRC16 or not ,

0 : not use;

1 : use.

## 4.13 UhfKillTagByEPC ()

**4.13.1 brief**

 Kill a specified tag with uii.

**4.13.2 Prototype**

 int WINAPI UhfKillTagByEPC (HANDLE hCom, UCHAR\* uKillPwd, UCHAR\* uUii, UCHAR\* uErrorCode, UCHAR flagCrc);

**4.13.3 Returns**

1 : success;

0 : fail;

**4.13.4 Parameters**

HANDLE hCom: [in], handle for communication port;

UCHAR\* uKillPwd : [in] , kill password stored in reserved area（refer to Appendix D）;

UCHAR\* uUii : [in] , uii of a tag;

UCHAR\* uErrorCode : [out] , error code (refer to Appendix D);

UCHAR flagCrc : [in] , using CRC16 or not ,

0 : not use;

1 : use.

## 4.14 UhfKillSingleTag ()

**4.14.1 brief**

 Kill a tag without uii.

**4.14.2 Prototype**

 int WINAPI UhfKillSingleTag (HANDLE hCom, UCHAR\* uKillPwd, UCHAR\* uUii, UCHAR\* uErrorCode, UCHAR flagCrc);

**4.14.3 Returns**

1 : success;

0 : fail;

**4.14.4 Parameters**

HANDLE hCom: [in], handle for communication port;

UCHAR\* uKillPwd : [in] , kill password stored in reserved area (refer to Appendix D);

UCHAR\* uUii : [out] , uii of tag that you kill;

UCHAR\* uErrorCode : [out] , error code (refer to Appendix D);

UCHAR flagCrc : [in] , using CRC16 or not ,

0 : not use;

1 : use.

## 4.15 UhfBlockWriteEPCByEPC()

**4.15.1 brief**

 Write epc to a specified tag with uii.

**4.15.2 Prototype**

int WINAPI UhfBlockWriteEPCByEPC (HANDLE hCom, UCHAR\* uAccessPwd, UCHAR uCnt, UCHAR\* uUii, UCHAR\* uWriteData, UCHAR\* uErrorCode, UCHAR\* uStatus, UCHAR\* uWritedLen, UCHAR\* RuUii, UCHAR flagCrc)

**4.15.3 Returns**

1 : success;

0 : fail;

**4.15.4 Parameters**

HANDLE hCom: [in], handle for communication port;

UCHAR\* uAccessPwd : [in] , access password stored in reserved area (refer to Appendix D);

UCHAR uCn: [in] , how many words you want to write;

UCHAR\* uUii : [in] , uii of a tag;

UCHAR\* uWriteData : [in] , data you want to write;

UCHAR\* uErrorCode : [out] , error code (refer to Appendix D);

UCHAR\* uStatus : [out] , the status of implementation (the length of the array is 1);

UCHAR\* uWritedLen : [out] , how many words have been written;

UCHAR\* RuUii: [out] , uii of a tag;

UCHAR flagCrc : [in] , using CRC16 or not ,

0 : not use;

1 : use.

## 4.16 UhfBlockWriteEPCToSingleTag()

**4.16.1 brief**

 Write epc to a tag without uii.

**4.16.2 Prototype**

int WINAPI UhfBlockWriteEPCToSingleTag (HANDLE hCom, UCHAR\* uAccessPwd, UCHAR uCnt, UCHAR\* uWriteData, UCHAR\* uUii, UCHAR\* uLenUii, UCHAR\* uStatus, UCHAR\* uErrorCode, UCHAR\* uWritedLen, UCHAR flagCrc);

**4.16.3 Returns**

1 : success;

0 : fail;

**4.16.4 Parameters**

HANDLE hCom: [in], handle for communication port;

UCHAR\* uAccessPwd : [in] , access password stored in reserved area (refer to Appendix D);

UCHAR uCnt : [in] , how many words you want to write;

UCHAR\* uWriteData : [in] , data you have write (refer to Appendix D);

UCHAR\* uUii : [out] , uii of a tag;

UCHAR\* uLenUii : [out] , how many bytes in uii;

UCHAR\* uErrorCode : [out] , error code (refer to Appendix D);

UCHAR\* uStatus : [out] , the status of implementation (the length of the array is 1);

UCHAR\* uWritedLen : [out] , how many words have been written;

UCHAR flagCrc : [in] , using CRC16 or not ,

0 : not use;

1 : use.

## 4.17 UhfStartReadDataFromMultiTag ()

**4.17.1 brief**

 Open the anti-collision function to read data from tags in loop mode.

**4.17.2 Prototype**

int WINAPI UhfStartReadDataFromMultiTag (HANDLE hCom, UCHAR\* uAccessPwd, UCHAR uBank, UCHAR\* uPtr, UCHAR uCnt, UCHAR uOption, UCHAR\* uPayLoad, UCHAR flagCrc)

**4.17.3 Returns**

1 : success;

0 : fail;

**4.17.4 Parameters**

HANDLE hCom: [in], handle for communication port;

UCHAR\* uAccessPwd : [in] , access password stored in reserved area (refer to Appendix D);

UCHAR uBank : [in] , storage area of a tag (refer to Appendix D);

UCHAR\* uPtr : [in] , address where you start read (refer to Appendix D);

UCHAR uCnt : [in] , how many words you want to read;

UCHAR uOption : [in] (0 : read one level data;1 : read two level data);

UCHAR\* uPayLoad : [in];

UCHAR flagCrc : [in] , using CRC16 or not ,

0 : not use; 1 : use.

Note:

When uOption=0, read one level data, the length of uPayLoad array is 2(first element is Q value range from 0 to 15, the second element must be 0x20). when uOption=1, read two level data, the length of uPayLoad array is 5 or 6(first element is the bank you want to read, the follow 1 or 2 elements is the address where you start to read, the next element is the count of words you want to read, the next element is Q value range from 0 to 15, the last element must be 0x20).

You should use this function first so that RLM uploads data to the buffer, then you use UhfGetDataFromMultiTag() to read data from the buffer, you must use UhfStopOperation() to stop the anti-collision in the last.

## 4.18 UhfGetDataFromMultiTag ()

**4.18.1 brief**

 Read data from buffer that RLM upload after open the anti-collision function.

**4.18.2 Prototype**

int WINAPI UhfGetDataFromMultiTag (HANDLE hCom, UCHAR\* uStatus, UCHAR\* ufData\_len, UCHAR\* ufReadData, UCHAR\* usData\_len, UCHAR\* usReadData, UCHAR\* uUii, UCHAR\* uLenUii, UCHAR\* uErrorCode);

**4.18.3 Returns**

1 : success;

0 : fail;

**4.18.4 Parameters**

HANDLE hCom: [in], handle for communication port;

UCHAR\* uStatus : [out] , the status of implementation (the length of the array is 1);

UCHAR\* ufData\_len : [out] , how many bytes you have read from the first level;

CHAR\* ufReadData : [out] , the data you have read from the first level;

UCHAR\* usData\_len : [out] , how many bytes you have read from the second level;

CHAR\* usReadData : [out] , how many bytes you have read from the second level;

UCHAR\* uUii : [out] , uii of a tag that you read from;

UCHAR\* uLenUii : [out] , how many bytes in the uii.

 UCHAR\* uErrorCode : [out] , error code (refer to Appendix D);

# 5.Firmware Upgrate Command Set

## 5.1 UhfUpdateInit ()

**5.1.1 brief**

 Inform the RLM to upgrade.

**5.1.2 Prototype**

int WINAPI UhfUpdateInit (HANDLE &hCom, char\* cPort, UCHAR\* STATUS, UCHAR\* RN32, UCHAR flagCrc);

**5.1.3 Returns**

1 : success;

0 : fail;

**5.1.4 Parameters**

HANDLE &hCom: [out], handle for communication port;

char\* cPort : [in], number of HID (such as \\?\hid#vid\_0483&pid\_5750#6&20bfa97&0&0000#{4d1e55b2-f16f-11cf-88cb-001111000030});

UCHAR\* STATUS : [out] , the status of implementation (the length of the array is 1);

UCHAR\* RN32 : [out] , RN32 the RLM responsed (the length of the array is 4);

UCHAR flagCrc : [in] , using CRC16 or not ,

0 : not use;

1 : use.

## 5.2 UhfUpdateSendRN32 ()

**5.2.1 brief**

 Send the radix-minus-one complement of RN32 to RLM.

**5.2.2 Prototype**

int WINAPI UhfUpdateSendRN32 (HANDLE hCom, UCHAR\* RN32, UCHAR\* STATUS, UCHAR flagCrc);

**5.2.3 Returns**

1 : success;

0 : fail;

**5.2.4 Parameters**

HANDLE hCom: [in], handle for communication port;

UCHAR\* RN32 : [in] , radix-minus-one complement of RN32 (the length of the array is 4);

UCHAR\* STATUS : [out] , the status of implementation (the length of the array is 1);

UCHAR flagCrc : [in] , using CRC16 or not ,

0 : not use;

1 : use.

## 5.3 UhfUpdateSendSize ()

**5.3.1 brief**

 Send the length of upgrading packet to RLM.

**5.3.2 Prototype**

int WINAPI UhfUpdateSendSize (HANDLE hCom, UCHAR\* STATUS, UCHAR\* FILESIZE, UCHAR flagCrc);

**5.3.3 Returns**

1 : success;

0 : fail;

**5.3.4 Parameters**

HANDLE hCom: [in], handle for communication port;

UCHAR\* STATUS : [out] , the status of implementation (the length of the array is 1);

UCHAR\* FILESIZE : [in] , the length of upgrading packet (the length of the array is 4);

UCHAR flagCrc : [in] , using CRC16 or not ,

0 : not use;

1 : use.

## 5.4 UhfUpdateSendData ()

**5.4.1 brief**

 Send upgrading packets to the RLM.

**5.4.2 Prototype**

int WINAPI UhfUpdateSendData (HANDLE hCom, UCHAR\* STATUS, UCHAR PACKNUM, UCHAR LASTPACK, int Data\_len, UCHAR\* TRANDATA, UCHAR flagCrc);

**5.4.3 Returns**

1 : success;

0 : fail;

**5.4.4 Parameters**

HANDLE hCom: [in], handle for communication port;

UCHAR\* STATUS : [out] , the status of implementation (the length of the array is 1);

UCHAR PACKNUM : [in] , the index of packet;

UCHAR LASTPACK : [in] , flag that indicates whether the current packet is the last packet or not. 0-the current packet is not the last packet , 1-the current packet is the last packet;

int Data\_len : [in] , how many bytes in the packet you want to send;

UCHAR\* TRANDATA : [in] , data packet;

UCHAR flagCrc : [in] , using CRC16 or not ,

0 : not use;

1 : use.

## 5.5 UhfUpdateCommit ()

**5.5.1 brief**

 Inform the RLM that upgrade has completed.

**5.5.2 Prototype**

int WINAPI UhfUpdateCommit(HANDLE hCom, UCHAR\* STATUS, UCHAR flagCrc);

**5.5.3 Returns**

1 : success;

0 : fail;

**5.5.4 Parameters**

HANDLE hCom: [in], handle for communication port;

UCHAR\* STATUS : [out] , the status of implementation (the length of the array is 1);

UCHAR flagCrc : [in] , using CRC16 or not ,

0 : not use;

1 : use.

# 6.Other Command Set

## 6.1 LockGenCode ()

**6.1.1 brief**

 Generate lock code

**6.1.2 Prototype**

 void LockGenCode(UCHAR bkill, UCHAR baccess, UCHAR buii, UCHAR btid, UCHAR buser, UCHAR\* LockCode);

**6.1.3 Returns**

None;

**6.1.4 Parameters**

UCHAR bkill : [in], lock mode about KillPwd (1 : lock; 2 : open; 3 : permanent lock; 4 : permanent open; other : keep status);

UCHAR baccess : [in], lock mode about AccessPwd (value is consistent with bkill);

UCHAR buii : [in], lock mode about UII (value is consistent with bkill);

UCHAR btid : [in], lock mode about TID (value is consistent with bkill);

UCHAR buser : [in], lock mode about USER (value is consistent with bkill);

UCHAR\* LockCode : [out], lock code;

## 6.2 UhfSearchHids ()

**6.2.1 brief**

 Search for HID serial number.

**6.2.2 Prototype**

 int WINAPI UhfSearchHids (char\* serials);

**6.2.3 Returns**

>0 : count of HIDs;

Other : fail;

**6.2.4 Parameters**

char\* serials : [out], HID serial number (serial numbers is separated by "|"; such as \\?\hid#vid\_0483&pid\_5750#6&20bfa97&0&0000#{4d1e55b2-f16f-11cf-88cb-001111000030});

# Appendix A : Description of the frequency parameters

|  |  |  |
| --- | --- | --- |
| field name | data type | description |
| uFreMode | UCHAR | mode of frequency, the value represents as the followings: 0 : China standard (920-925MHz);1 : China standard (840-845MHz);2 : ETSI standard;3 : Fixed frequency mode (922MHz);4 : User defined; |
| uFreBase | UCHAR | Frequency base, the value represents as the followings: 0 : 50MHz;1 : 125MHz; |
| uBaseFre | UCHAR\* | Starting frequency, range from 840 to 960; |
| uChannNum | UCHAR | The number of channels; |
| uChannSpc | UCHAR | The base of channel's bandwidth, the value represents as the followings: When bFreBase is equal to 50, it range from 1 to 20;When bFreBase is equal to 125, it range from 1 to 8; |
| uFreHop | UCHAR | Frequency hopping mode, the value represents as the followings: 0 : Random hopping;1 : Hopping from high to low;2 : Hopping from low to high;other : Random hopping; |
| There is a certain relationship between the following parameters:Starting frequency,Ending frequency,Frequency base,The number of channels,The base of channel's bandwidth: if(Frequency base == 0){ Ending frequency - Starting frequency = (The number of channels - 1) \* The base of channel's bandwidth \* 0.050}else{ Ending frequency - Starting frequency = (The number of channels - 1) \* The base of channel's bandwidth \* 0.125}For example : frequency range is 920.625-924.375MHzbFreBase = 1;now calculate bBaseFre: the decimal part of the starting frequency is 0.625;the integer Part of the starting frequency is 920;mantissa base of starting frequency : 0.625÷0.125 = 0x05;The binary representation of 920 is “11 1001 1000”uBaseFre[0] = (UCHAR)(11 1001 1000 >> 3); --->0x73uBaseFre[1] = (UCHAR) (11 1001 1000 << 5 | (0x05 & 0x1F)); --->0x05Ending frequency - Starting frequency = (The number of channels - 1) \* The base of channel's bandwidth \* 0.125924.375 -920.625 = （The number of channels - 1）\* The base of channel's bandwidth \* 0.125（The number of channels-1）\* The base of channel's bandwidth = 30If the number of channels is equal to 16 , then the base of channel's bandwidth should be 2.uFreMode = 4;uFreBase = 1;uFreBase = {0x73 , 0x05};uChannNum = 16;uChannSpc = 2;uFreHop = 0; |

# Appendix B : Instructions of the parameters about tag operation

|  |  |  |
| --- | --- | --- |
| field name | data type | description |
| uAccessPwd | UCHAR\* | Access password stored in reserved area, length of the array is 4 , for example : UCHAR uAccessPwd[] = {0x12 , 0x34 , 0x56 , 0x78}; |
| uKillPwd | UCHAR\* | Kill password stored in reserved area, length of the array is 4 , for example : UCHAR uKillPwd[] = {0x12 , 0x34 , 0x56 , 0x78}; |
| uBank | UCHAR | Bank of tag , the value represents as the followings: 0x00 : RESERVED;0x01 : UII;0x02 : TID;0x03 : USER; |
| uPtr | UCHAR\* | Address where you start to read or write (Address≥0), it means which word you start to read or write from.if (address > 127){UCHAR bPtr[2]; uPtr[0] = ((address >> 7) | 0x80); uPtr[1] = (address & 0x7F);}else{ UCHAR uPtr [1]; uPtr[0] = address;} |
| uCnt | UCHAR | How many words you want to read or write. |
| uUii | UCHAR\* | Uii of a tag. |
| uLenUii | UCHAR\* | Length of Uii. |
| uReadData | UCHAR\* | Data read from the tag. |
| uWriteData | UCHAR\* | Data that will be written to tag. |
| uWritedLen | UCHAR\* | length of data that have written to tag. |
| uStatus | UCHAR\* | Status, length of the array is 1. |
| uLockData | UCHAR\* | Lock code use for locking memery of the tag , the length of the array is 3. |
| uErrorCode | UCHAR\* | Error code, length of the array is 1. |