RLM Application Programming Interface (Bluetooth)

Ray-Links (Beijing) Technologies Co., Ltd.
Address:R.317,Zhongguancun Chuangye Tower,No.26 Shangdi Information Road,Haidian District,Beijing100085,P.R.C.
TEL : 010-84672430 FAX : 010-84672430-602
www.ray-links.com

What's Inside

1. Introduction

RLM is short for Ray-Links Module, and it's a UHF reader module series developed by Ray-Links. Currently, we have three kinds of products that have been introduced to market, including RLM100, RLM200 and RLM300. You can make RLM work through a variety of means. Undoubtedly, Bluetooth is a good choice. This document help developers to develop applications those can control RLM through Bluetooth on the android platform. The document consists of three parts: Introduction, API function definitions and appendices.

The library is a jar file. You can create a folder called libs, and copy the jar we provide to libs folder, then use it to develop. Two main classes are Function and ModuleControl, they are in the same package called "com.raylinks". You should know that RLM100 does not have all the features, but RLM200 and RLM300 have all features we provide in the document. You can learn more from Appendix D.

2. API function definitions

2.1 UhfReaderConnect()

2.1.1 brief

Establish a Bluetooth connection with the RLM.

2.1.2 Prototype

public boolean UhfReaderConnect (String macStr, byte[] bStatus, byte flagCrc);

2.1.3 Returns

true : success;

false : fail;

2.1.4 Parameters

String macStr : [in] , MAC address to be connected;

byte[] bStatus : [out] , connection status (the length of the array is 1);

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

0 : not use;

1 : use.

2.2 UhfReaderDisconnect ()

2.2.1 brief

Close Bluetooth connection.

2.2.2 Prototype

public boolean UhfReaderDisconnect ();

2.2.3 Returns

true : success;

false : fail;

2.2.4 Parameters

None.

2.3 UhfGetPaStatus ()

2.3.1 brief

Read RLM connection status.

2.3.2 Prototype

public boolean UhfGetPaStatus (byte[] uStatus, byte flagCrc);

2.3.3 Returns

true : success;

false : fail;

2.3.4 Parameters

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

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

0 : not use;

1 : use.

2.4 UhfGetPower ()

2.4.1 brief

Get power of RLM.

2.4.2 Prototype

public boolean UhfGetPower(byte[] bPower, byte flagCrc);

2.4.3 Returns

true : success;

false : fail;

2.4.4 Parameters

byte[] bPower : [out] , value of power(the length of the array is 1);

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

0 : not use;

1 : use.

2.5 UhfSetPower ()

2.5.1 brief

Set power of RLM.

2.5.2 Prototype

public boolean UhfSetPower(byte bOption, byte bPower, byte flagCrc);

2.5.3 Returns

true : success;

false : fail;

2.5.4 Parameters

byte bOption : [in] , must be set to 0x01;

byte bPower : [in] , value to be set;

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

0 : not use;

1 : use.

2.6 UhfGetFrequency ()

2.6.1 brief

Get frequency of RLM.

2.6.2 Prototype

public boolean UhfGetFrequency(byte[] bFreMode, byte[] bFreBase, byte[] bBaseFre, byte[] bChannNum, byte[] bChannSpc, byte[] bFreHop, byte flagCrc);

2.6.3 Returns

true : success;

false : fail;

2.6.4 Parameters

byte[] bFreMode : [out] , mode of frequency (the array of length is 1);

byte[] bFreBase : [out] , frequency base (the array of length is 1);

byte[] bBaseFre : [out] , starting frequency (the array of length is 2);

byte[] bChannNum : [out] , count of channels (the array of length is 1);

byte[] bChannSpc : [out] , base of channel's bandwidth (the array of length is 1);

byte[] bFreHop : [out] , mode of hopping (the array of length is 1);

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

0 : not use;

1 : use.

2.7 UhfSetFrequency ()

2.7.1 brief

Set frequency of RLM.

2.7.2 Prototype

public boolean UhfSetFrequency(byte bFreMode, byte bFreBase, byte[] bBaseFre, byte bChannNum, byte bChannSpc, byte bFreHop, byte flagCrc);

2.7.3 Returns

true : success;

false : fail;

2.7.4 Parameters

byte bFreMode : [in] , mode of frequency ;

byte bFreBase : [in] , frequency base;

byte[] bBaseFre : [in] , starting frequency (the length of the array is 2);

byte bChannNum : [in] , count of channels;

byte bChannSpc : [in] , base of channel's bandwidth;

byte bFreHop : [in] , mode of hopping;

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

0 : not use;

1 : use.

2.8 UhfGetVersion ()

2.8.1 brief

Read hardware serial number and software version number of RLM.

2.8.2 Prototype

public boolean UhfGetVersion(byte[] bSerial, byte[] bVersion, byte flagCrc);

2.8.3 Returns

true : success;

true : fail;

2.8.4 Parameters

byte[] bSerial : [out] , hardware serial number (the length of the array is 6);

byte[] bVersion : [out] , software version number (the length of the array is 3);

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

0 : not use;

1 : use.

2.9 UhfStartInventory ()

2.9.1 brief

Start to read uii in loop mode.

2.9.2 Prototype

public boolean UhfStartInventory(byte flagAnti, byte initQ, byte flagCrc);

2.9.3 Returns

true : success;

false : fail;

2.9.4 Parameters

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

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

byte 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.

2.10 UhfInventorySingleTag ()

2.10.1 brief

Read uii(including PC and EPC).

2.10.2 Prototype

public boolean UhfInventorySingleTag(byte[] bLenUii, byte[] bUii, byte flagCrc);

2.10.3 Returns

true : success;

false : fail;

2.10.4 Parameters

byte[] bLenUii : [out] , count of bytes in uii (the length of array is 1);

byte[] bUii : [out] , uii of a tag;

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

0 : not use;

1 : use.

2.11 UhfReadInventory ()

2.11.1 brief

Read uii from buffer that RLM upload.

2.11.2 Prototype

public boolean UhfReadInventory(byte[] bLenUii, byte[] bUii);

2.11.3 Returns

true : success;

fasle : fail;

2.11.4 Parameters

byte[] bLenUii : [out] , count of bytes in uii (the length of the array is 1);

byte[] bUii : [out] , uii of a tag.

2.12 UhfStopOperation ()

2.12.1 brief

Stop working and put RLM on standby.

2.12.2 Prototype

public boolean UhfStopOperation(byte flagCrc);

2.12.3 Returns

true : success;

false : fail;

2.12.4 Parameters

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

0 : not use;

1 : use.

2.13 UhfReadDataByEPC ()

2.13.1 brief

Read data from a specified tag with uii.

2.12.2 Prototype

public boolean UhfReadDataByEPC(byte[] bAccessPwd, byte bBank, byte[] bPtr, byte bCnt, byte[] bUii, byte[] bReadData, byte[] bErrorCode, byte flagCrc)

2.13.3 Returns

true : success;

false : fail;

2.13.4 Parameters

byte[] bAccessPwd : [in] , access password stored in reserved area（refer to Appendix B）;

byte bBank : [in] , storage area of a tag (refer to Appendix B);

byte[] bPtr : [in] , address where you start read (refer to Appendix B);

byte bCnt : [in] , how many words you want to read;

byte[] bUii : [in] , uii of a tag;

byte[] bReadData : [out] , data you have read(refer to Appendix B);

byte[] bErrorCode : [out] , error code(refer to Appendix B)

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

0 : not use;

1 : use.

2.14 UhfWriteDataByEPC ()

2.14.1 brief

Write one word to a specified tag with uii.

2.14.2 Prototype

public boolean UhfWriteDataByEPC(byte[] bAccessPwd, byte bBank, byte[] bPtr, byte bCnt, byte[] bUii, byte[] bWriteData, byte[] bErrorCode, byte flagCrc);

2.14.3 Returns

true : success;

false : fail;

2.14.4 Parameters

byte[] bAccessPwd : [in] , access password stored in reserved area (refer to Appendix B);

byte bBank : [in] , storage area of a tag (refer to Appendix B);

byte[] bPtr : [in] , address where you start write (refer to Appendix B);

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

byte[] bUii : [in] , uii of a tag;

byte[] bWriteData : [in] , data you want to write(refer to Appendix B);

byte[] bErrorCode : [out] , error code(refer to Appendix B)

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

0 : not use;

1 : use.

2.15 UhfEraseDataByEPC ()

2.15.1 brief

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

2.15.2 Prototype

public boolean UhfEraseDataByEPC(byte[] bAccessPwd, byte bBank, byte[] bPtr, byte bCnt, byte[] bUii, byte[] bErrorCode, byte flagCrc)

2.15.3 Returns

true : success;

false : fail;

2.15.4 Parameters

byte[] bAccessPwd : [in] , access password stored in reserved area (refer to Appendix B);

byte bBank : [in] , storage area of a tag (refer to Appendix B);

byte[] bPtr : [in] , address where you start erase (refer to Appendix B);

byte bCnt : [in] , how many words you want to erase;

byte[] bUii : [in] , uii of a tag;

byte[] bErrorCode : [out] , error code(refer to Appendix B);

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

0 : not use;

1 : use.

2.16 UhfLockMemByEPC ()

2.16.1 brief

Lock a specified tag with uii.

2.16.2 Prototype

public boolean UhfLockMemByEPC(byte[] bAccessPwd, byte[] bLockData, byte[] bUii, byte[] bErrorCode, byte flagCrc)

2.16.3 Returns

true : success;

true : fail;

2.16.4 Parameters

byte[] bAccessPwd : [in] , access password stored in reserved area (refer to Appendix B);

byte[] bLockData : [in] , lock code which means how you lock (refer to Appendix B);

byte[] bUii : [in] , uii of a tag;

byte[] bErrorCode : [out] ,error code(refer to Appendix B)

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

0 : not use;

1 : use.

2.17 UhfKillTagByEPC ()

2.17.1 brief

Kill a specified tag with uii.

2.17.2 Prototype

public boolean UhfKillTagByEPC(byte[] bKillPwd , byte[] bUii , byte[] bErrorCode , byte flagCrc);

2.17.3 Returns

true : success;

true : fail;

2.17.4 Parameters

byte[] bKillPwd : [in] , kill password stored in reserved area（refer to Appendix B）;

byte[] bUii : [in] , uii of a tag;

byte[] bErrorCode : [out] , error code (refer to Appendix B)

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

0 : not use;

1 : use.

2.18 UhfBlockWriteDataByEPC ()

2.18.1 brief

Write several words to a specified tag with uii.

2.18.2 Prototype

public boolean UhfBlockWriteDataByEPC(byte[] bAccessPwd, byte bBank, byte[] bPtr, byte bCnt, byte[] bUii, byte[] bWriteData, byte[] bErrorCode, byte[] bStatus, byte[] bWriteLen, byte[] RuUii, byte flagCrc)

2.18.3 Returns

true : success;

false : fail;

2.18.4 Parameters

byte[] bAccessPwd : [in] , access password stored in reserved area (refer to Appendix B);

byte bBank : [in] , storage area of a tag (refer to Appendix B);

byte[] bPtr : [in] , address where you start write (refer to Appendix B);

byte bCnt : [in] , how many words you want to write;

byte[] bUii : [in] , uii of a tag;

byte[] bWriteData : [in] , data you want to write;

byte[] bErrorCode : [out] , error code (refer to Appendix B);

byte[] bStatus : [out] , the status of implementation (the length of the array is 1);

byte[] bWriteLen : [out] , how many words have been written;

byte[] RuUii : [out] , uii of a tag;

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

0 : not use;

1 : use.

2.19 UhfReadDataFromSingleTag ()

2.19.1 brief

Read uii from a tag without uii.

2.19.2 Prototype

public boolean UhfReadDataFromSingleTag(byte[] bAccessPwd, byte bBank, byte[] bPtr, byte bCnt, byte[] bReadData, byte[] bUii, byte[] bLenUii, byte[] bErrorCode, byte flagCrc);

2.19.3 Returns

true : success;

false : fail;

2.19.4 Parameters

byte[] bAccessPwd : [in] , access password stored in reserved area (refer to Appendix B);

byte bBank : [in] , storage area of a tag (refer to Appendix B);

byte[] bPtr : [in] , address where you start read (refer to Appendix B);

byte bCnt : [in] , how many words you want to read;

byte[] bReadData : [out] , data you have read (refer to Appendix B);

byte[] bUii : [out] , uii of tag that you read from;

byte[] bLenUii : [out] , how many bytes in uii;

byte[] bErrorCode : error code (refer to Appendix B);

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

0 : not use;

1 : use.

2.20 UhfWriteDataToSingleTag ()

2.20.1 brief

Write data to a tag without uii.

2.20.2 Prototype

public boolean UhfWriteDataToSingleTag(byte[] bAccessPwd, byte bBank, byte[] bPtr, byte bCnt, byte[] bWriteData, byte[] bUii, byte[] bLenUii, byte[] bErrorCode, byte flagCrc);

2.20.3 Returns

true : success;

false : fail;

2.20.4 Parameters

byte[] bAccessPwd : [in] , access password stored in reserved area (refer to Appendix B);

byte bBank : [in] , storage area of a tag (refer to Appendix B);

byte[] bPtr : [in] , address where you start write (refer to Appendix B);

byte bCnt : [in] , how many words you want to write;

byte[] bWriteData : [in] , data you want to write;

byte[] bUii : [out] , uii of tag that you want to write;

byte[] bLenUii : [out] , how many bytes in uii;

byte[] bErrorCode : [out] , error code (refer to Appendix B);

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

0 : not use;

1 : use.

2.21 UhfEraseDataFromSingleTag ()

2.21.1 brief

Erase data in a bank of the tag without Uii.

2.21.2 Prototype

public boolean UhfEraseDataFromSingleTag(byte[] bAccessPwd, byte bBank, byte[] bPtr, byte bCnt, byte[] bUii, byte[] bErrorCode, byte flagCrc);

2.21.3 Returns

true : success;

false : fail;

2.21.4 Parameters

byte[] bAccessPwd : [in] , access password stored in reserved area (refer to Appendix B);

byte bBank : [in] , storage area of a tag (refer to Appendix B);

byte[] bPtr : [in] , address where you start read (refer to Appendix B);

byte bCnt : [in] , how many words you want to erase;

byte[] bUii : [out] , uii of tag that you erase;

byte[] bErrorCode : [out] , error code (refer to Appendix B);

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

0 : not use;

1 : use.

2.22 UhfLockMemFromSingleTag ()

2.22.1 brief

Lock a tag without uii.

2.22.2 Prototype

public boolean UhfLockMemFromSingleTag(byte[] bAccessPwd, byte[] bLockData, byte[] bUii, byte[] bErrorCode, byte flagCrc);

2.22.3 Returns

true : success;

false : fail;

2.22.4 Parameters

byte[] bAccessPwd : [in] , access password stored in reserved area (refer to Appendix B);

byte[] bLockData : [in] , lock code which means how you lock (refer to Appendix B);

byte[] bUii : [out] , uii of tag that you lock;

byte[] bErrorCode : [out] , error code (refer to Appendix B);

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

0 : not use;

1 : use.

2.23 UhfKillSingleTag ()

2.23.1 brief

Kill a tag without uii.

2.23.2 Prototype

public boolean UhfKillSingleTag(byte[] bKillPwd, byte[] bUii, byte[] bErrorCode, byte flagCrc);

2.23.3 Returns

true : success;

false : fail;

2.23.4 Parameters

byte[] bKillPwd : [in] , kill password stored in reserved area (refer to Appendix B);

byte[] bUii : [out] , uii of tag that you kill;

byte[] bErrorCode : [out] , error code (refer to Appendix B);

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

0 : not use;

1 : use.

2.24 UhfBlockWriteDataToSingleTag()

2.24.1 brief

Write several words to a tag without uii.

2.24.2 Prototype

public boolean UhfBlockWriteDataToSingleTag(byte[] bAccessPwd, byte bBank, byte[] bPtr, byte bCnt, byte[] bWriteData, byte[] bUii, byte[] bLenUii, byte[] bStatus, byte[] bErrorCode, byte[] bWriteLen, byte flagCrc)

2.24.3 Returns

true : success;

false : fail;

2.24.4 Parameters

byte[] bAccessPwd : [in] , access password stored in reserved area (refer to Appendix B);

byte bBank : [in] , storage area of a tag (refer to Appendix B);

byte[] bPtr : [in] , address where you start write (refer to Appendix B);

byte bCnt : [in] , how many words you want to write;

byte[] bWriteData : [in] , data you have write (refer to Appendix B);

byte[] bUii : [out] , uii of a tag;

byte[] bLenUii : [out] , how many bytes in uii;

byte[] bStatus : [out] , the status of implementation (the length of the array is 1);

byte[] bErrorCode : [out] , error code (refer to Appendix B);

byte[] bWriteLen : [out] , how many words have been written;

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

0 : not use;

1 : use.

2.25 UhfReadMaxDataByEPC ()

2.25.1 brief

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

2.25.2 Prototype

public boolean UhfReadMaxDataByEPC(byte[] bAccessPwd, byte bBank, byte[] bPtr, byte[] bUii, byte[] bDataLen, byte[] bReadData, byte[] bErrorCode, byte flagCrc);

2.25.3 Returns

true : success;

false : fail;

2.25.4 参数

byte[] bAccessPwd : [in] , access password stored in reserved area (refer to Appendix B);

byte bBank : [in] , storage area of a tag (refer to Appendix B);

byte[] bPtr : [in] , address where you start read (refer to Appendix B);

byte[] bUii : [in] , uii of a tag;

byte[] bDataLen : [out] , how many words you hava read.

byte[] bReadData : [out] , data you have read (refer to Appendix B);

byte[] bErrorCode : [out] , error code (refer to Appendix B);

byte 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;

2.26 UhfReadMaxDataFromSingleTag ()

2.26.1 brief

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

2.26.2 Prototype

public boolean UhfReadMaxDataFromSingleTag(byte[] bAccessPwd, byte bBank, byte[] bPtr, byte[] bDataLen, byte[] bReadData, byte[] bUii, byte[]bLenUii, byte[] bErrorCode, byte flagCrc)

2.26.3 Returns

true : success;

false : fail;

2.26.4 参数

byte[] bAccessPwd : [in] , access password stored in reserved area (refer to Appendix B);

byte bBank : [in] , storage area of a tag (refer to Appendix B);

byte[] bPtr : [in] , address where you start read (refer to Appendix B);

byte[] bDataLen : [out] , how many bytes you have read;

byte[] bReadData : [out] , data you have read;

byte[] bUii : [out] , uii of tag that you read from;

byte[]bLenUii : [out] , how many bytes in uii;

byte[] bErrorCode : [out] , error code (refer to Appendix B);

byte 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;

2.27 UhfEnterSleepMode ()

2.27.1 brief

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

2.27.2 Prototype

public boolean UhfEnterSleepMode(byte flagCrc);

2.27.3 Returns

true : success;

false : fail;

2.27.4 参数

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

0 : not use;

1 : use.

2.28 UhfGetReaderUID ()

2.28.1 brief

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

2.28.2 Prototype

public boolean UhfGetReaderUID(byte[] bUid, byte flagCrc);

2.28.3 Returns

true : success;

false : fail;

2.28.4 Parameters

byte[] bUid : [out] , uid（the length of the array is 12）;

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

0 : not use;

1 : use.

2.29 UhfStartReadDataFromMultiTag ()

2.29.1 brief

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

2.29.2 Prototype

public boolean UhfStartReadDataFromMultiTag(byte[] bAccessPwd, byte bBank, byte[] bPtr, byte bCnt, byte bOption, byte[] bPayLoad, byte flagCrc);

2.29.3 Returns

true : success;

false : fail;

2.29.4 Parameters

byte[] bAccessPwd : [in] , access password stored in reserved area (refer to Appendix B);

byte bBank : [in] , storage area of a tag (refer to Appendix B);

byte[] bPtr : [in] , address where you start read (refer to Appendix B);

byte bCnt : [in] , how many words you want to read;

byte bOption : [in]（0 : read one level data;1 : read two level data）;

byte[] bPayLoad : [in];

byte 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.

2.30 UhfGetDataFromMultiTag ()

2.30.1 brief

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

2.30.2 Prototype

public boolean UhfGetDataFromMultiTag(byte[] bStatus, byte[] bfDataLen, byte[] bfReadData, byte[] bsDataLen, byte[] bsReadData, byte[] bUii, byte[] bLenUii);

2.30.3 Returns

true : success;

false : fail;

2.30.4 Parameters

byte[] bStatus : [out] , the status of implementation (the length of the array is 1);

byte[] bfDataLen : [out] , how many bytes you have read from the first level;

byte[] bfReadData : [out] , the data you have read from the first level;

byte[] bsDataLen : [out] , how many bytes you have read from the second level;

byte[] bsReadData : [out] , how many bytes you have read from the second level;

byte[] bUii : [out] , uii of a tag that you read from;

byte[] bLenUii : [out] , how many bytes in the uii.

2.31 UhfGetRegister ()

2.31.1 brief

Read data from the specified address of register.

2.31.2 Prototype

public boolean UhfGetRegister(int RADD, int RLEN, byte[] bStatus, byte[] bReg, byte flagCrc);

2.31.3 Returns

true : success;

false : fail;

2.31.4 参数

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

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

byte[] bStatus : [out] , the status of implementation (the length of the array is 1);

byte[] bReg : [out] , data you have read from registers;

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

0 : not use;

1 : use.

2.32 UhfSetRegister ()

2.32.1 brief

Write data to the specified address of register.

2.32.2 Prototype

public boolean UhfSetRegister(int RADD, int RLEN, byte[] bRegData, byte[] bStatus, byte flagCrc);

2.32.3 Returns

true : success;

false : fail;

2.32.4 参数

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

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

byte[] bRegData : [in] , data you want to write;

byte[] bStatus : [out] , the status of implementation (the length of the array is 1);

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

0 : not use;

1 : use.

2.33 UhfResetRegister ()

2.33.1 brief

Reset all values to factory defaults.

2.33.2 Prototype

public boolean UhfResetRegister(byte flagCrc);

2.33.3 Returns

true : success;

false : fail;

2.33.4 参数

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

0 : not use;

1 : use.

2.34 UhfSaveRegister ()

2.34.1 brief

Save the current settings to the corresponding registers.

2.34.2 Prototype

public boolean UhfSaveRegister(byte flagCrc);

2.34.3 Returns

true : success;

false : fail;

2.34.4 参数

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

0 : not use;

1 : use.

2.35 UhfAddFilter ()

2.35.1 brief

Add select record to RLM.

2.35.2 Prototype

public boolean UhfAddFilter(Srecord[] pSrecord, byte[] bStatus, byte flagCrc);

2.35.3 Returns

true : success;

false : fail;

2.35.4 参数

Srecord[] pSrecord : [in] , select record that you want add (refer to Appendix A);

byte[] bStatus : [out] , the status of implementation (the length of the array is 1);

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

0 : not use;

1 : use.

2.36 UhfDeleteFilterByIndex ()

2.36.1 brief

Delete select record from RLM.

2.36.2 Prototype

public boolean UhfDeleteFilterByIndex(byte bSindex, byte[] bStatus, byte flagCrc);

2.36.3 Returns

true : success;

false : fail;

2.36.4 参数

byte bSindex : [in] , select record that you want to delete;

byte[] bStatus : [out] , the status of implementation (the length of the array is 1);

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

0 : not use;

1 : use.

2.37 UhfStartGetFilterByIndex ()

2.37.1 brief

Start to read select record.

2.37.2 Prototype

public boolean UhfStartGetFilterByIndex(byte bSindex, byte bSnum, byte[] bStatus, byte flagCrc);

2.37.3 Returns

true : success;

false : fail;

2.37.4 参数

byte bSindex : [in] , index of record where you want to read;

byte bSnum : [in] , how many records you want to read;

byte[] bStatus : [out] , the status of implementation (the length of the array is 1);

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

0 : not use;

1 : use.

Note:

UhfStartGetFilterByIndex() is used to read select records. You should use this function first so that RLM uploads select records to the buffer, then you use UhfReadFilterByIndex() to read select records from the buffer.

2.38 UhfReadFilterByIndex ()

2.38.1 brief

Read select records from buffer.

2.38.2 Prototype

public boolean UhfReadFilterByIndex(byte[] bStatus, Srecord[] pSrecord);

2.38.3 Returns

true : success;

false : fail;

2.38.4 参数

byte[] bStatus : [out] , the status of implementation (the length of the array is 1), the value represents as the followings:

0x00 : there are still some records exist in RLM;

0x01 : records have been read out;

0x10 : records have been read out, the index of record does not exist;

0x11 : this record is error;

Srecord[] pSrecord : [out] , refer to Appendix A.

2.39 UhfSelectFilterByIndex ()

2.39.1 brief

Choose the select records to filter tags when you do some operations about tags.

2.39.2 Prototype

public boolean UhfSelectFilterByIndex(byte bSindex, byte bSnum, byte[] bStatus, byte flagCrc);

2.39.3 Returns

true : success;

false : fail;

2.39.4 参数

byte bSindex : [in] , index of record where you want to choose;

byte bSnum : [in] , how many records you want to choose;

byte[] bStatus : [out] , the status of implementation (the length of the array is 1);

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

0 : not use;

1 : use.

2.40 UhfUpdateInit ()

2.40.1 brief

Inform the RLM to upgrade.

2.40.2 Prototype

public boolean UhfUpdataInit(byte[] bStatus, byte[] RN32, byte flagCrc);

2.40.3 Returns

true : success;

false : fail;

2.40.4 参数

byte[] bStatus : [out] , the status of implementation (the length of the array is 1);

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

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

0 : not use;

1 : use.

2.41 UhfUpdateSendRN32 ()

2.41.1 brief

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

2.41.2 Prototype

public boolean UhfUpdateSendRN32(byte[] RN32, byte[] bStatus, byte flagCrc);

2.41.3 Returns

true : success;

false : fail;

2.41.4 参数

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

byte[] bStatus : [out] , the status of implementation (the length of the array is 1);

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

0 : not use;

1 : use.

2.42 UhfUpdateSendSize ()

2.42.1 brief

Send the length of upgrading packet to RLM.

2.42.2 Prototype

public boolean UhfUpdateSendSize(byte[] bStatus, byte[] FILESIZE, byte flagCrc);

2.42.3 Returns

true : success;

false : fail;

2.42.4 参数

byte[] bStatus : [out] , the status of implementation (the length of the array is 1);

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

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

0 : not use;

1 : use.

2.43 UhfUpdateSendData ()

2.43.1 brief

Send upgrading packets to the RLM.

2.43.2 Prototype

public boolean UhfUpdateSendData(byte[] bStatus, byte PACKNUM, byte LASTPACK, int iDataLen, byte[] TRANDATA, byte flagCrc);

2.43.3 Returns

true : success;

false : fail;

2.43.4 参数

byte[] bStatus : [out] , the status of implementation (the length of the array is 1);

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

byte 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 iDataLen : [in] , how many bytes in the packet you want to send;

byte[] TRANDATA : [in] , data packet;

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

0 : not use;

1 : use.

2.44 UhfUpdateCommit ()

2.44.1 brief

Inform the RLM that upgrade has completed.

2.44.2 Prototype

public boolean UhfUpdataCommit(byte[] bStatus, byte flagCrc);

2.44.3 Returns

true : success;

false : fail;

2.44.4 参数

byte[] bStatus : [out] , the status of implementation (the length of the array is 1);

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

0 : not use;

1 : use.

2.45 bytesToHexString ()

2.45.1 brief

Convert bytes array to hex string.

2.45.2 Prototype

public String bytesToHexString(byte[] src, int len);

2.45.3 Returns

Hex string that have been converted;

2.45.4 Parameters

byte[] src : [in] , the bytes array to be converted;

int len : [in] , the length of the array.

2.46 byteToHexString ()

2.46.1 brief

Convert byte to hex string.

2.46.2 Prototype

public String byteToHexString(byte src);

2.46.3 Returns

hex string that have been converted

2.46.4 Parameters

byte src : [in] , byte to be converted;

2.47 HexStringToBytes ()

2.47.1 brief

Convert hex string to bytes array.

2.47.2 Prototype

public byte[] HexStringToBytes(String s);

2.47.3 Returns

Byte array that have been converted.

2.47.4 Parameters

String s : [in] , hex string to be convert.

2.48 isHex ()

2.48.1 brief

judge whether the passed string is hex string or not.

2.48.2 Prototype

public boolean isHex(String hex);

2.48.3 Returns

true : success;

false : fail;

2.48.4 Parameters

String hex : [in] hex string to be judged.

2.49 isDecimal ()

2.49.1 brief

Judge whether the passed string is decimal string or not.

2.49.2 Prototype

public boolean isDecimal(String decimal);

2.49.3 Returns

true : success;

false : fail;

2.49.4 Parameters

String decimal : [in] , decimal string to be judge.

2.50 LockGenCode ()

2.50.1 brief

Create lock code.

2.50.2 Prototype

public boolean LockGenCode(byte bkill, byte baccess, byte buii, byte btid, byte buser, byte[] LockCode);

2.50.3 Returns

true : success;

false : fail,

2.50.4 Parameters

byte bkill : [in] , kill password, the value represents as the followings:

0 : keep status;

1 : lock;

2 : open;

3 : permanent lock;

4 : permanent open;

byte baccess : [in] , access password, the value is the same as bkill;

byte buii : [in] , UII bank, the value is the same as bkill;

byte btid : [in] , TID bank, the value is the same as bkill;

byte buser : [in] , USER bank, the value is the same as bkill;

byte[] LockCode : [out] , lock code has been created（the length of the array is 3）;

Appendix A : Srecord definition

```
public static final class Srecord{
        public byte Sindex;
        public byte Slen;
        public byte Target;
        public byte Action;
        public byte Bank;
        public byte[] Ptr = new byte[2];
        public byte Len;
        public byte[] Mask= new byte[32];
        public byte Trancate;
    }
```

Srecord class definition:

field name	data type	description
Sindex	byte	The index of record, value range from 0 to 15
Slen	byte	How many bytes in the record including Sindex, Slen, Target, Action, Bank, Ptr, Len, Mask, Truncate.
Target	byte	The value must be 0.
Action	byte	The vaule must be 4.
bank	byte	Bank of tag , the value represents as the followings: 0 : RESERVED; 1 : UII; 2 : TID; 3 : USER;

Ptr	byte[]	Bit address of mask, address ⩾ 0, it means that mask start from which bit.
		if (address > 127)
		{
		byte uPtr[2];
		uPtr[0] = ((address >> 7) | 0x80);
		uPtr[1] = ((address >> 7) & 0x7F);
		}
		else
		{
		byte uPtr[1];
		uPtr[0] = address & 0x7F;
		}

Len	byte	How many bits in the mask ,for example,if mask is “0800”, then there are 16 bits in mask.
Mask	byte[]	mask, such as “0800”.
Truncate	byte	The value must be 0.

The sample code:

```
Srecord[] pSrecord = new Srecord[1];
byte[] bStatus = new byte[1];

pSrecord[0].Sindex = 1;
pSrecord[0].Slen = 12;
pSrecord[0].Target = 0;
pSrecord[0].Action = 4;
pSrecord[0].Bank = 3;
pSrecord[0].Ptr[0] = 16;
pSrecord[0].Len = 32;
pSrecord[0].Mask[0] = 12;
pSrecord[0].Mask[1] = 34;
pSrecord[0].Mask[2] = 56;
pSrecord[0].Mask[3] = 78;
pSrecord[0].Truncate = 0;
if(moduleControl.UhfAddFilter(pSrecord ,    bStatus ,    flagCrc))
{
      //success
}else{
      //fail
}
```

Appendix B : Instructions of the parameters about tag operation

field name	data type	description
bAccessPwd	byte[]	Access password stored in reserved area, length of the array is 4 , for example : byte[] bAccessPwd = {0x12 , 0x34 , 0x56 , 0x78};
bKillPwd	byte[]	Kill password stored in reserved area, length of the array is 4 , for example : byte[] bKillPwd = {0x12 , 0x34 , 0x56 , 0x78};
bBank	byte	Bank of tag , the value represents as the followings: 0x00 : RESERVED; 0x01 : UII; 0x02 : TID; 0x03 : USER;

bPtr	byte[]	Address where you start to read or write (Address≥0), it means which word you start to read or write from.
		if (address > 127)
		{
		byte[] bPtr = new byte[2];
		bPtr[0] = ((address >> 7) | 0x80);
		bPtr[1] = ((address >> 7) & 0x7F);
		}
		else
		{
		byte[] bPtr = new byte[1];
		bPtr[0] = address & 0x7F;
		}

bCnt	byte	How many words you want to read or write.
bLockData	byte[]	Lock code , the length of the array is 3.

Appendix C : Description of the frequency parameters

field name	data type	description
bFreMode	byte[](the length of the array is 1) /byte	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;
bFreBase	byte[](the length of the array is 1)	Frequency base, the value represents as the followings: 0 : 50MHz; 1 : 125MHz;
bBaseFre	byte[](the length of the array is 2) /byte	Starting frequency, range from 840 to 960;
bChannNum	byte[](the length of the array is 1) /byte	The number of channels, range from 1 to 16;
bChannSpc	byte[](the length of the array is 1) /byte	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;
bFreHop	byte[](the length of the array is 1) /byte	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 920.625-924.375MHz
bFreBase = 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"
bBaseFre[0] = (byte)(11 1001 1000 >> 3); --->0x73
bBaseFre[1] = (byte)(11 1001 1000 << 5 | (0x05 & 0x1F)); --->0x05

Ending frequency - Starting frequency = (The number of channels - 1) * The base of channel's bandwidth * 0.125

924.375 -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 = 30

If the number of channels is equal to 16 , then the base of channel's bandwidth should be 2.

bFreMode = 4;
bFreBase = 1;
bFreBase = {0x73 , 0x05};
bChannNum = 16;
bChannSpc = 2;
bFreHop = 0;

Appendix D : Function category

name of function	which class the function in	RLM100	RLM200	RLM300
UhfReaderConnect	com.raylinks.ModuleControl	√	√	√
UhfReaderDisconnect	com.raylinks.ModuleControl	√	√	√
UhfGetPaStatus	com.raylinks.ModuleControl	√	√	√
UhfGetPower	com.raylinks.ModuleControl	√	√	√
UhfSetPower	com.raylinks.ModuleControl	√	√	√
UhfGetFrequency	com.raylinks.ModuleControl	√	√	√
UhfSetFrequency	com.raylinks.ModuleControl	√	√	√
UhfGetVersion	com.raylinks.ModuleControl	√	√	√
UhfStartInventory	com.raylinks.ModuleControl	√	√	√
UhfInventorySingleTag	com.raylinks.ModuleControl	√	√	√
UhfReadInventory	com.raylinks.ModuleControl	√	√	√
UhfStopOperation	com.raylinks.ModuleControl	√	√	√
UhfReadDataByEPC	com.raylinks.ModuleControl	√	√	√
UhfWriteDataByEPC	com.raylinks.ModuleControl	√	√	√
UhfEraseDataByEPC	com.raylinks.ModuleControl	√	√	√
UhfLockMemByEPC	com.raylinks.ModuleControl	√	√	√
UhfKillTagByEPC	com.raylinks.ModuleControl	√	√	√
UhfBlockWriteDataByEPC	com.raylinks.ModuleControl	√	√	√
UhfReadDataFromSingleTag	com.raylinks.ModuleControl	√	√	√
UhfWriteDataToSingleTag	com.raylinks.ModuleControl	√	√	√
UhfEraseDataFromSingleTag	com.raylinks.ModuleControl	√	√	√
UhfLockMemFromSingleTag	com.raylinks.ModuleControl	√	√	√
UhfKillSingleTag	com.raylinks.ModuleControl	√	√	√
UhfBlockWriteDataToSingleTag	com.raylinks.ModuleControl	√	√	√
UhfReadMaxDataByEPC	com.raylinks.ModuleControl	√	√	√
UhfReadMaxDataFromSingleTag	com.raylinks.ModuleControl	√	√	√
UhfEnterSleepMode	com.raylinks.ModuleControl	√	√	√
UhfGetReaderUID	com.raylinks.ModuleControl		√	√
UhfStartReadDataFromMultiTag	com.raylinks.ModuleControl		√	√
UhfGetDataFromMultiTag	com.raylinks.ModuleControl		√	√
UhfGetRegister	com.raylinks.ModuleControl		√	√
UhfSetRegister	com.raylinks.ModuleControl		√	√
UhfResetRegister	com.raylinks.ModuleControl		√	√
UhfSaveRegister	com.raylinks.ModuleControl		√	√
UhfAddFilter	com.raylinks.ModuleControl		√	√
UhfDeleteFilterByIndex	com.raylinks.ModuleControl		√	√
UhfStartGetFilterByIndex	com.raylinks.ModuleControl		√	√
UhfReadFilterByIndex	com.raylinks.ModuleControl		√	√
UhfSelectFilterByIndex	com.raylinks.ModuleControl		√	√

UhfUpdateInit	com.raylinks.ModuleControl		√	√
UhfUpdateSendRN32	com.raylinks.ModuleControl		√	√
UhfUpdateSendSize	com.raylinks.ModuleControl		√	√
UhfUpdateSendData	com.raylinks.ModuleControl		√	√
UhfUpdateCommit	com.raylinks.ModuleControl		√	√
bytesToHexString	com.raylinks.Function	√	√	√
byteToHexString	com.raylinks.Function	√	√	√
HexStringToBytes	com.raylinks.Function	√	√	√
isHex	com.raylinks.Function	√	√	√
isDecimal	com.raylinks.Function	√	√	√
LockGenCode	com.raylinks.Function	√	√	√