**UP-RFID-RT型综合教学平台**

**900MHz模块应用开发**

**API说明**

**博创智联科技有限公司**

**2016-08-30**

**900MHz模块应用开发API说明**

## **900MHz模块API相关说明**

这部分主要介绍900MHz模块API的使用范围，通信接口等。

## **API的开发环境**

本API由Qt 5.5.0 Mingw492\_32开发，如果用户的Qt版本比较旧的话，则有些API函数功能可能不能使用。

本API函数是针对Qt开发环境来实现的，如果用户需要支持其他平台的API，建议结合协议文档，参照本API去改写。

## **通信接口**

900MHz模块与其他主从模块通过串口通信（232电平），其波特率为57600、数据位为8位、停止位为1位。

## **API文件说明**

本次提供的API分为两个目录，lib为库，inc为头文件，使用时必须添加头文件（xxxx.h，xxxx\_global.h），lib下有xxx.a的文件为静态库，有xxx.dll的文件为动态库，根据实际需求可以只引入一个库。

## **API函数的定义**

## **UHF\_Connect ()**

1. **功能简介**

用于获取连接命令帧。

1. **函数原型**

uint8\* UHF\_Connect(void);

1. **返回值**

一个包含连接命令帧的uint8型指针，第一个字节是该帧的总长度，从第二个字节开始到结束为连接命令帧实体。

1. **输入参数**

无

1. **调用示例**

uint8 \*p = m900-> UHF\_Connect();

serialPort->write((char \* ) ( p + 1),p[0]);

说明：m900为M900Dll的实体，serialPort为串口实体。

## **UHF\_GetPower()**

1. **功能简介**

用于构造获取功率的命令帧。

1. **函数原型**

uint8\* UHF\_GetPower(void);

1. **返回值**

一个包含获取功率命令帧的uint8型指针，第一个字节是该帧的总长度，从第二个字节开始到结束为获取功率命令帧实体。

1. **输入参数**

无

1. **调用示例**

uint8 \*p = m900-> UHF\_GetPower();

serialPort->write((char \* ) ( p + 1),p[0]);

说明：m900为M900Dll的实体，serialPort为串口实体。

## **UHF\_SetPower()**

1. **功能简介**

用于构造设置功率命令帧。

1. **函数原型**

uint8\* UHF\_SetPower(uint8 opt, uint8 pwr);

1. **返回值**

一个包含连接命令帧的uint8型指针，第一个字节是该帧的总长度，从第二个字节开始到结束为设置功率命令帧实体。

1. **输入参数**

参数1：OPTION字段，一般输入0x01，详见协议文档

参数2：待设置的功率值加0x80

1. **调用示例**

uint8 \*p = m900-> UHF\_SetPower(0x01,0x8A);

serialPort->write((char \* ) ( p + 1),p[0]);

说明：m900为M900Dll的实体，serialPort为串口实体，OPTION参数设置为0x01，功率设置10dBm。

## **UHF\_GetFrequency()**

1. **功能简介**

用于构造获取频率命令帧。

1. **函数原型**

uint8\* UHF\_GetFrequency(void);

1. **返回值**

一个包含获取频率命令帧的uint8型指针，第一个字节是该帧的总长度，从第二个字节开始到结束为获取频率命令帧实体。

1. **输入参数**

无

1. **调用示例**

uint8 \*p = m900->UHF\_GetFrequency();

serialPort->write((char \* ) ( p + 1),p[0]);

说明：m900为M900Dll的实体，serialPort为串口实体。

## **UHF\_SetFrequency()**

1. **功能简介**

用于构造设置频率命令。

1. **函数原型**

uint8\* UHF\_SetFrequency(UHF\_ReqSetFre\_t \*data);

1. **返回值**

一个包含设置频率命令帧的uint8型指针，第一个字节是该帧的总长度，从第二个字节开始到结束为设置频率命令帧实体。

1. **输入参数**

参数1：UHF\_ReqSetFre\_t结构体指针，该结构体内包含了模式，频率基数等信息，请参照协议文档构造该结构体。

1. **调用示例**

uint8 \*p = m900-> UHF\_SetFrequency(data);

serialPort->write((char \* ) ( p + 1),p[0]);

说明：m900为M900Dll的实体，serialPort为串口实体，data为UHF\_ReqSetFre\_t类型指针，UHF\_ReqSetFre\_t请参照附录。

## **UHF\_Inventory()**

1. **功能简介**

用于构造单步（循环）识别标签的命令帧。

1. **函数原型**

uint8\* UHF\_Inventory(void);

1. **返回值**

一个包含单步识别命令帧的uint8型指针，第一个字节是该帧的总长度，从第二个字节开始到结束为单步识别命令帧实体。

1. **输入参数**

无

1. **调用示例**

uint8 \*p = m900-> UHF\_Inventory();

serialPort->write((char \* ) ( p + 1),p[0]);

说明：m900为M900Dll的实体，serialPort为串口实体。

## **UHF\_Inventory(Q)**

1. **功能简介**

用于构造防碰撞识别命令帧。

1. **函数原型**

uint8\* UHF\_Inventory(uint8 Q);

1. **返回值**

一个包含防碰撞识别命令帧的uint8型指针，第一个字节是该帧的总长度，从第二个字节开始到结束为防碰撞识别命令帧实体。

1. **输入参数**

参数1：uint8型值，用于间接指明防碰撞识别标签数目。Q值的选择请参照对应协议文档的附录。

1. **调用示例**

uint8 \*p = m900->UHF\_Inventory(4);

serialPort->write((char \* ) ( p + 1),p[0]);

说明：m900为M900Dll的实体，serialPort为串口实体，Q=4，以最佳的方式一直识别两张卡。

## **UHF\_InventorySingle()**

1. **功能简介**

用于构造单步（非循环）识别UII命令帧。

1. **函数原型**

uint8\* UHF\_InventorySingle(void);

1. **返回值**

一个包含单步识别（非循环）命令帧的uint8型指针，第一个字节是该帧的总长度，从第二个字节开始到结束为单步识别命令帧实体。

1. **输入参数**

无

1. **调用示例**

uint8 \*p = m900-> UHF\_InventorySingle ();

serialPort->write((char \* ) ( p + 1),p[0]);

说明：m900为M900Dll的实体，serialPort为串口实体。

## **UHF\_StopGet**

1. **功能简介**

用于构造终止当前任务命令帧。

1. **函数原型**

uint8\* UHF\_StopGet(void);

1. **返回值**

一个包含终止命令帧的uint8型指针，第一个字节是该帧的总长度，从第二个字节开始到结束为终止命令帧实体。

1. **输入参数**

无

1. **调用示例**

uint8 \*p = m900->UHF\_StopGet();

serialPort->write((char \* ) ( p + 1),p[0]);

说明：m900为M900Dll的实体，serialPort为串口实体。

## **UHF\_ReadData()**

1. **功能简介**

用于构造指定UII的方式读取标签数据命令帧。

1. **函数原型**

uint8\* UHF\_ReadData(UHF\_ReqReadData\_t \*data, uint8\* uii, uint8 uiiLen);

1. **返回值**

一个包含读取标签内存信息命令帧的uint8型指针，第一个字节是该帧的总长度，从第二个字节开始到结束为命令帧实体。

1. **输入参数**

参数1：UHF\_ReqReadData\_t类型的指针，请参照附录。

参数2：UII，用于指定是对哪张卡进行读取

参数3：UII的长度，UII包含PCBits，长度可变

1. **调用示例**

uint8 \*p = m900-> UHF\_ReadData(data，uii，len);

serialPort->write((char \* ) ( p + 1),p[0]);

说明：m900为M900Dll的实体，serialPort为串口实体，data中包含密码等信息，uii为卡号字符串，len为uii的长度。

## **UHF\_ReadDataSingle**

1. **功能简介**

用于构造不指定UII的方式读取标签数据的命令帧。

1. **函数原型**

uint8\* UHF\_ReadDataSingle(UHF\_ReqReadData\_t \*data);

1. **返回值**

一个不包含UII信息的读取标签内存数据命令帧的uint8型指针，第一个字节是该帧的总长度，从第二个字节开始到结束为该命令帧实体。

1. **输入参数**

参数1：UHF\_ReqReadData\_t类型的指针，请参照附录。

1. **调用示例**

uint8 \*p = m900-> UHF\_ReadDataSingle(data);

serialPort->write((char \* ) ( p + 1),p[0]);

说明：m900为M900Dll的实体，serialPort为串口实体，data中包含密码等信息。

## **UHF\_WriteData()**

1. **功能简介**

用于构造指定UII的方式写入数据命令帧。

1. **函数原型**

uint8\* UHF\_WriteData(UHF\_ReqWriteData\_t \*data, uint8 \*uii, uint8 uiiLen);

1. **返回值**

一个包含指定UII方式写入数据命令帧的uint8型指针，第一个字节是该帧的总长度，从第二个字节开始到结束为该命令帧实体。

1. **输入参数**

参数1：UHF\_ReqWriteData\_t类型的指针，请参照附录。

参数2：UII，用于指定是对哪张卡进行写入

参数3：UII的长度，UII包含PCBits，长度可变

1. **调用示例**

uint8 \*p = m900-> UHF\_WriteData(data,uii，uiilen);

serialPort->write((char \* ) ( p + 1),p[0]);

说明：m900为M900Dll的实体，serialPort为串口实体，data包含密码、分区、地址、长度信息，uii为操作的卡号，uiilen对应于卡号的长度。

## **UHF\_WriteDataSingle()**

1. **功能简介**

用于构造不指定UII方式写入数据到标签的命令帧。

1. **函数原型**

uint8\* UHF\_WriteDataSingle(UHF\_ReqWriteData\_t \*data);

1. **返回值**

一个包含不指定UII方式写入数据到标签命令帧的uint8型指针，第一个字节是该帧的总长度，从第二个字节开始到结束为该命令帧实体。

1. **输入参数**

参数1：UHF\_ReqWriteData\_t类型的指针，请参照附录。

1. **调用示例**

uint8 \*p = m900-> UHF\_WriteDataSingle(data);

serialPort->write((char \* ) ( p + 1),p[0]);

说明：m900为M900Dll的实体，serialPort为串口实体，data包含密码、分区、地址、长度信息。

## **UHF\_EraseData()**

1. **功能简介**

用于构造擦除数据块命令帧。

1. **函数原型**

uint8\* UHF\_EraseData(UHF\_ReqEraseData\_t \*data, uint8 \*uii, uint8 uiiLen);

1. **返回值**

一个包含擦除数据块命令帧的uint8型指针，第一个字节是该帧的总长度，从第二个字节开始到结束为擦除命令帧实体。

1. **输入参数**

参数1：UHF\_ReqEraseData\_t类型的指针，请参照附录。

参数2：UII，用于指定是对哪张卡进行擦除

参数3：UII的长度，UII包含PCBits，长度可变

1. **调用示例**

uint8 \*p = m900-> UHF\_EraseData(data,uii，uiilen);

serialPort->write((char \* ) ( p + 1),p[0]);

说明：m900为M900Dll的实体，serialPort为串口实体，data包含密码、分区、地址、长度信息，uii为操作的卡号，uiilen对应于卡号的长度。

## **UHF\_LockMen()**

1. **功能简介**

用于构造锁定标签命令帧。

1. **函数原型**

uint8\* UHF\_LockMen(UHF\_ReqLockMem\_t \*data);

1. **返回值**

一个包含锁定命令帧的uint8型指针，第一个字节是该帧的总长度，从第二个字节开始到结束为锁定命令帧实体。

1. **输入参数**

参数1：UHF\_ReqLockMem\_t类型的指针，请参照附录。

1. **调用示例**

uint8 \*p = m900-> UHF\_ LockMen(data);

serialPort->write((char \* ) ( p + 1),p[0]);

说明：m900为M900Dll的实体，serialPort为串口实体，data指针中包含密码、锁定负载、UII信息。

## **UHF\_KillTag()**

1. **功能简介**

用于构造销毁标签命令帧。

1. **函数原型**

uint8\* UHF\_KillTag(UHF\_ReqKillTag\_t \*data);

1. **返回值**

一个包含销毁标签命令帧的uint8型指针，第一个字节是该帧的总长度，从第二个字节开始到结束为销毁命令帧实体。

1. **输入参数**

参数1：UHF\_ReqKillTag\_t类型的指针，请参照附录。

1. **调用示例**

uint8 \*p = m900-> UHF\_KillTag(data);

serialPort->write((char \* ) ( p + 1),p[0]);

说明：m900为M900Dll的实体，serialPort为串口实体，data中包含执行销毁操作的密码、UII信息。

## **UHF\_GetVersion()**

1. **功能简介**

用于构造获取版本信息命令帧。

1. **函数原型**

uint8\* UHF\_GetVersion(void);

1. **返回值**

一个包含获取版本信息命令帧的uint8型指针，第一个字节是该帧的总长度，从第二个字节开始到结束为获取版本信息命令帧实体。

1. **输入参数**

无

1. **调用示例**

uint8 \*p = m900-> UHF\_GetVersion();

serialPort->write((char \* ) ( p + 1),p[0]);

说明：m900为M900Dll的实体，serialPort为串口实体。

## **UHF\_RequestFrameConstructor()**

1. **功能简介**

用于构造一个请求帧。

1. **函数原型**

UHF915M\_ReqFrame\_t UHF\_RequestFrameConstructor(uint8 \*data,int len);

1. **返回值**

一个包含SOF、Len、CMD等所有字段的请求帧结构体。返回类型请参照附录。

1. **输入参数**

参数1：uint8类型的指针，指向要从串口实际发送的数据。

参数1：数据长度。

1. **调用示例**

uint8 \*p = m900->UHF\_RequestFrameConstructor(uint8 \*data,int len);

## **UHF\_RsponseFrameConstructor()**

1. **功能简介**

用于构造一个响应帧。

1. **函数原型**

UHF915M\_RspFrame\_t UHF\_RsponseFrameConstructor(uint8 \*data , int len);

1. **返回值**

UHF915M\_RspFrame\_t类型的结构体。UHF915M\_RspFrame\_t类型请参照附录。

1. **输入参数**

参数1：uint8类型的指针，指向要从串口读取出来的数据。

参数1：数据长度。

1. **调用示例**

uint8 \*p = m900-> UHF\_RsponseFrameConstructor(uint8 \*data , int len);

serialPort->write((char \* ) ( p + 1),p[0]);

说明：m900为M900Dll的实体，serialPort为串口实体。

**附录**

1. **数据结构的定义**
2. **UHF915M\_ReqFrame\_t：**

typedef struct

{

uint8 sof;

uint8 len;

uint8 cmd; // bit-7:(0=没有crc-16, 1=带有crc-16) bit<6-0>:(cmd-id)

uint8 \*data; // <254

uint8 datalen;

uint16 crc;

uint8 crclen;

uint16 eof;

}UHF915M\_ReqFrame\_t;

1. **UHF915M\_RspFrame\_t：**

typedef struct

{

uint8 sof;

uint8 len;

uint8 cmd;

uint8 sta; // 执行结果

uint8 \*data; // <253

uint8 datalen;

uint16 crc;

uint8 crclen;

uint8 eof;

}UHF915M\_RspFrame\_t;

1. **UHF\_ReqSetFre\_t：**

typedef struct

{

uint8 freMode; // H4-Bit: NC L4-Bit:(0000:中国标准<920~925MHz> 0001:中国标准<840~845MHz> 0010:ETSI标准 0100:用户自定义 其他:中国标准（920~925MHz))

uint8 freBase; // H7-Bit: NC L1-Bit:(0: 50KHz 1: 125KHz)

uint16 bf; // (15)-Bit: NC (14~5)-Bit:起始频率（整数部分） (4~0)-Bit:起始频率尾数积数（起始频率小数部分）

uint8 cn; // 频道数

uint8 spc; // H4-Bit: NC L4-Bit:频道带宽积数

uint8 freHop; // H6-Bit: NC L2-Bit:跳频顺序方式（00:随机跳频 01:从高往低顺序跳频 10:从低往高顺序跳频 其它:随机跳频）

}UHF\_ReqSetFre\_t;

1. **UHF\_RspGetFre\_t：**

typedef struct

{

uint8 freMode;

uint16 bf;

uint8 cn;

uint8 spc;

uint8 freHop;

}UHF\_RspGetFre\_t;

1. **UHF\_ReqReadData\_t：**

typedef struct

{

uint8 apwd[4];

uint8 bank;

uint8 ptr;

uint8 cnt;

}UHF\_ReqReadData\_t;

1. **UHF\_ReqWriteData\_t：**

typedef struct

{

uint8 apwd[4];

uint8 bank;

uint8 ptr;

uint8 cnt;

uint8 data[2];

}UHF\_ReqWriteData\_t;

1. **UHF\_ReqEraseData\_t：**

typedef struct

{

uint8 apwd[4];

uint8 bank;

uint8 ptr;

uint8 cnt;

}UHF\_ReqEraseData\_t;

1. **UHF\_ReqLockMem\_t：**

typedef struct

{

uint32 apwd;

uint8 lockData[3]; // H4-Bit: NC L20-Bit: Lock-CommandPayload

uint8 uii[26];

}UHF\_ReqLockMem\_t;

1. **类型的定义：**

typedef signed char int8;

typedef unsigned char uint8;

typedef signed short int16;

typedef unsigned short uint16;

typedef signed long int32;

typedef unsigned long uint32;

1. **标识符的定义：**

#define UHF\_SOF 0xAA

#define UHF\_EOF 0x55

enum UHF\_CmdTable

{

UHFCMD\_GET\_STATUS = 0x00, // 询问状态

UHFCMD\_GET\_POWER, // 读取功率

UHFCMD\_SET\_POWER, // 设置功率

UHFCMD\_GET\_FRE = 0x05, // 读取频率

UHFCMD\_SET\_FRE, // 设置频率

UHFCMD\_GET\_VERSION, // 读取版本信息

UHFCMD\_INVENTORY = 0x10, // 识别标签（单标签识别）

UHFCMD\_INVENTORY\_ANTI, // 识别标签（防碰撞识别）

UHFCMD\_STOP\_GET, // 停止操作

UHFCMD\_READ\_DATA, // 读取标签数据

UHFCMD\_WRITE\_DATA, // 写入标签数据

UHFCMD\_ERASE\_DATA, // 擦除标签数据

UHFCMD\_LOCK\_MEM, // 锁定标签

UHFCMD\_KILL\_TAG, // 销毁标签

UHFCMD\_INVENTORY\_SINGLE, // 识别标签（单步识别）

UHFCMD\_WIEGAND\_INVENTORY, // 韦根识别

UHFCMD\_SINGLE\_READ\_DATA = 0x20, // 读取标签数据（不指定UII）

UHFCMD\_SINGLE\_WRITE\_DATA // 写入标签数据（不指定UII）

};

enum UHF\_FrequenceMode

{

UHF\_FREMODE\_CHINA\_920\_925MHZ,

UHF\_FREMODE\_CHINA\_840\_845MHZ,

UHF\_FREMODE\_ETSI,

UHF\_FREMODE\_915MHZ,

UHF\_FREMODE\_USER,

UHF\_FREMODE\_OTHER

};

enum UHF\_FrequenceBase

{

UHF\_FREBASE\_50KHZ,

UHF\_FREBASE\_125KHZ

};

enum UHF\_FrequenceHopMode

{

UHF\_FREHOP\_RANDOM,

UHF\_FREHOP\_SEQUENCE\_HL,

UHF\_FREHOP\_SEQUENCE\_LH

};

#define BUILD\_FRE\_BF(integer, decimal) ((uint16)((integer<<5)&(decimal&0x1f)))