杭州菲迪科技有限公司

ERC 实验室

RF905-FE

开

发

手

册

杭州菲迪科技有限公司ERC实验室

一． 模块介绍

(1) 433Mhz 开放 ISM 频段免许可证使用

(2) 高效 GFSK 调制，抗干扰能力强，适合工控场合

(3) 125 频道，满足多点通信和跳频通信需要

(4) 内置硬件 CRC 检错和多点通信地址匹配机制

(5) 接收数据包有中断管脚提示机制，便于嵌入式运用

(6) 工作电压 1.9 - 3.6V

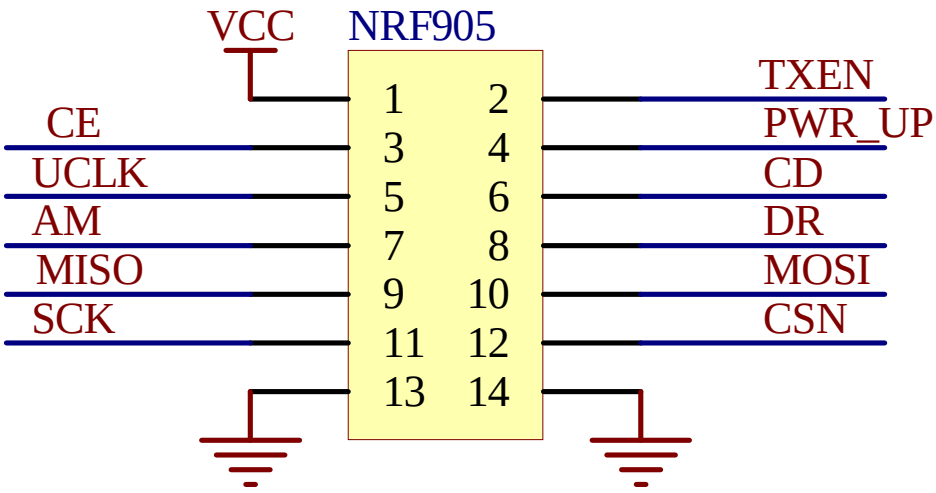
(7) TX Mode（+10dBm） 电流为 30mA

RX Mode 电流为 12.2mA

POWERDOWM Mode 电流为 2.5uA

(8) 收发模式切换时间 < 650us

二. 接口电路和引脚说明



(1) VCC 管脚接直流电源，电压范围 1.9V-3.6V，推荐 3.3V

(2) 数字输入管脚，用于控制 905 模块工作状态切换

由 MCU 配置：“0”发射模式 “1”接收模式

(3) 数字输入管脚，用于使能 905 模块发送和接收

由 MCU 配置：“0”禁止 “1”使能

(4) 数字输入管脚，用于控制 905 模块的内部上电

由 MCU 配置：“0”掉电 “1”上电

(5) 晶振时钟输出，用于作为其他数字芯片时钟，一般不使用

(6) 数字输出管脚，用于外部载波检测，一般不使用

由 MCU 读入：“0”没有载波信号 “1”检测到载波信号

(7) 数字输出管脚，用于提示地址匹配成功，一般不使用

由 MCU 读入：“0”匹配失败 “1”匹配成功

(8) 数字输出管脚，用于提示数据包接收成功

可以被用作软件查询或作为中断触发信号

由 MCU 读入：“0”没有数据包 “1”接收到数据包

(9) 数字输出管脚，SPI 口线标准的数据输出管脚

由 MCU 读入，MCU 以 SPI 标准时序读取管脚上的数据和状态

(10) 数字输入管脚，SPI 口线标准的数据输入管脚

由 MCU 写入，MCU 以 SPI 标准时序将命令和数据写到该管脚

(11) 数字输入管脚，SPI 口线标准的时钟信号管脚

由 MCU 以 SPI 标准时序给予时钟脉冲

(12) 数字输入管脚，用于使能 905 模块的 SPI 总线

由 MCU 配置，“0”使能 “1”禁止

(13) 接地

(14) 接地

(15) 实际使用我们模块时无需对以上口线特别了解，我们有专门的傻瓜驱动提供给用户，用户只需调用三个函数即可实现简单运用。如有特殊运用可参考 Nordic 公司的 nRF905 数据手册或者联系我们。

(16) 在选择 MCU 时，尽量使用电压为 3.3V 的型号，推荐 AVR 系列。如所选 MCU 工作电压不是 3.3V 的，则可按照 MCU 的 IO 管脚的驱动能力加上合适阻值的上拉或缓冲电阻以保护口线不被烧毁。

(17) 对接口顺序或者模块有任何特殊要求的可以联系我们

三. 技术参数

RF905-FE模块使用Nordic公司的nRF905芯片开发。芯片将无线收发工作完全集成，用户无需去考虑其中的细节过程。芯片的ShockBurst技术使无线收发的效率大大提高，即使低速的MCU也能有很高的瞬间传输速率。这使得无线冲撞概率和系统功耗等都得以很好的降低。

快速参考数据:

参数	数值	单位
最低工作电压	1.9	V
最高工作电压	3.6	V
最大发射功率	10	dBm
最小发射功率	-10	dBm
最大传输率曼切斯特编码	50	kbps
输出功率 10 dBm工作电流	30	mA
输出功率-10 dBm工作电流	9	mA
接收模式时工作电流	12.5	mA
Standby模式时工作电流	32	uA
POWERDOWN 模式时工作电流	2.5	uA
温度范围	-40 to +85	°C
典型灵敏度	-100	dBm

四. 工作方式

RF905-FE 模块一共有 4 种模式，分为两种活动模式和两种节电模式。

活动模式：

ShockBurst 接收模式

ShockBurst 发送模式

节电模式：

掉电 SPI 编程模式

空闲 SPI 编程模式

nRF905工作模式由TRX_CE、TX_EN、PWR_UP 的管脚配置来设定。

PWR_UP	TRX_CE	TX_EN	工作模式
0	X	X	掉电SPI编程
1	0	X	空闲SPI编程
1	1	0	ShockBurst接收
1	1	1	ShockBurst发送

在设定好工作模式后必须按照 905 芯片内部定义的命令数据规范进行 SPI 口线操作来运作 RF905-FE 模块。这其中还得牵涉到 905 模式切换的控制管脚的时序规范，SPI 口线的时序规范，SPI 口线上的命令类型和数据格式规范等等诸多因素，给用户带来很大不便。

为解决这一问题我们向用户提供一份开源的傻瓜驱动，用户无需了解 905 内部工作原理和寄存器组，只需在接对 IO 口线的前提下调用三个函数即可完成无线收发功能。具体代码实例请看下文的示范代码区。

五. 示范代码

在使用我们 ERC 实验室的 RF905-FE 模块时你只需要一点 C 语言的基础知识和数字电路的常识即可简单的运用。以下范例适合入门者和简单运用，对于高级用户和特殊运用只要稍加改动我们提供的傻瓜驱动也能实现卓越的性能。后者可以参考 Nordic 公司的 nRF905 数据手册或者联系我们。我们有强大的嵌入式射频运用人才可为您提供技术帮助和方案参谋。

我们的驱动中定义的函数有以下所列：

```
/********************函数申明*********************/
void nRF905Write8Bit(unsigned char byte);
unsigned char nRF905Read8Bit(void);
void nRF905ConfigRegister(unsigned char* p);
void nRF905WriteTxAddress(
  unsigned char addr0 , unsigned char addr1 ,
  unsigned char addr2 , unsigned char addr3 );
void nRF905WriteRxAddress(
  unsigned char addr0 , unsigned char addr1 ,
  unsigned char addr2 , unsigned char addr3 );
void nRF905ReadRxPayload(unsigned char* p,unsigned char total);
void nRF905WriteTxPayload(unsigned char* p,unsigned char total);
void nRF905Initializtion();
unsigned char nRF905Scanf(unsigned char* p);
void nRF905Printf(unsigned char* p);
```

一般的用户只需使用其中的最后三个函数即可。以下介绍这三个函数：

```
void nRF905Initializtion();
```

905 初始化函数，在调用另外两个函数前必须调用一次，推荐在程序开始时调用。

```
unsigned char nRF905Scanf(unsigned char* p);
```

905 接收函数，需要一个 32 字节的数组作为数据包传递。每次调用该函数，该函数都会试图去读取数据包。如果模块收到数据包则数据包会被写入该 32 字节的数组中而函数返回值为 1，否则不对数组进行操作而函数返回值为 0。

```
void nRF905Printf(unsigned char* p);
```

905 发送函数，需要一个 32 字节的数组作为数据包传送。每次调用该函数时该函数都会把该 32 字节数组中的数据写到 905 中并发送。

代码实例：

```
/*发送范例*/
unsigned char tx_buf[32];
unsigned char rx_buf[32];
void main()
{
  nRF905Initializtion();
  tx_buf[0] = 0;
  tx_buf[1] = 1;//在发送数组中写入所需信息
  while(1)
  {
    nRF905Printf(tx_buf);
    //用发送函数调用发送数组，将其打包发送
  }
}

/*接收范例*/
unsigned char tx_buf[32];
unsigned char rx_buf[32];
void main()
{
  nRF905Initializtion();
  while(1)
  {
    while(nRF905Scanf(rx_buf)!=1);
    //当没有接收到数据包时候无限循环，一旦接收到数据包则将其写入所指定的 32 字节接收数组 rx_buf 中
```

```
    PORTB = rx_buf[0];
    //将接收到的数据包的第一个字节从 B 口输出
  }
}

/*转发包范例*/
unsigned char buf[32];
void main()
{
  nRF905Initializtion();
  while(1)
  {
    while(nRF905Scanf(buf)!=1);//等待收到数据包
    nRF905Printf(buf);//将收到的数据包原样发送
    //注释:nRF905Printf() nRF905Scanf() 两个函数调用顺序完全随意不用考虑两者间的
内在影响
  }
}
```

其他扩展运用可以调用另外的接口函数，比如：

```
void nRF905WriteTxAddress(
  unsigned char addr0 , unsigned char addr1 ,
  unsigned char addr2 , unsigned char addr3 );
void nRF905WriteRxAddress(
  unsigned char addr0 , unsigned char addr1 ,
  unsigned char addr2 , unsigned char addr3 );
```

两个函数分别用于设定数据包的发送地址和本机的接收地址，简单易用。

再来讲解一下驱动的高级运用和移植问题，驱动的高级运用主要在于驱动的 nRF905_pub.h 文档中，里面有寄存器的配置信息和控制模式切换的宏函数。

比如掉电模式宏函数：

```
#define nRF905_POWER_DOWN_MODE() \
nRF905_PWR_UP = 0     //掉电模式
```

可以像调用一般函数一样直接调用 nRF905_POWER_DOWN_MODE(); 来进

入 905 的掉电模式，但这些运用需要用户对 905 内部原理有一定的了解，请慎用！

再如 905 的寄存器配置信息区：

```
#define CH_NO      76          //freq=422.4+76/10=430MHz
```

用户可以将 76 改成其他值如 1，可以使 905 工作在 freq=422.4+1/10=422.5MHz 的频率上。该 h 文档中已经加入很多中文的注释，用户可以根据这些注释来实现更多的高级运用。在移植驱动的时候主要是关于 IO 口线的配置，主要取决于所选用的 MCU 的类型和 IDE 环境。这些配置也主要在 nRF905_pub.h 文档中，文档中主要有以下配置与移植相关：

```
#include "iom48.h"     //IO 寄存器定义文件
```

iom48.h 是 IAR 环境中的 ATmega48 这一 MCU 的寄存器定义文件，在实际移植时应按实际情况来调用相关的寄存器定义文件。

```
/*****************IO 定义(根据具体情况重新定义)*****************/
#define nRF905_TX_EN_DDR   DDRB_Bit6
#define nRF905_TRX_CE_DDR  DDRB_Bit7
#define nRF905_PWR_UP_DDR  DDRD_Bit5
#define nRF905_DR_DDR      DDRB_Bit0
#define nRF905_MISO_DDR    DDRB_Bit1
#define nRF905_MOSI_DDR    DDRB_Bit2
#define nRF905_SCK_DDR     DDRB_Bit3
#define nRF905_CSN_DDR     DDRB_Bit4
//IO 方向寄存器
#define nRF905_TX_EN       PORTB_Bit6
#define nRF905_TRX_CE      PORTB_Bit7
#define nRF905_PWR_UP      PORTD_Bit5
#define nRF905_DR          PORTB_Bit0
#define nRF905_MISO        PORTB_Bit1
```

```
#define nRF905_MOSI         PORTB_Bit2
#define nRF905_SCK          PORTB_Bit3
#define nRF905_CSN          PORTB_Bit4
//IO 数据输出寄存器
#define nRF905_DR_PIN       PINB_Bit0
#define nRF905_MISO_PIN     PINB_Bit1
//IO 读管脚寄存器
```

该些管脚配置在移植时特别要小心，也比较容易出错。由于牵涉到三种寄存器（IO 方向，IO 输出数据，IO 读引脚）推荐移植时选用的 MCU 的 IO 控制寄存器尽量和 AVR 相似，否则需要较大的改动（比如 51 类型的）。

对于模块的驱动移植问题和模块的特殊及高级运用可以直接和我们联系，根据具体情况我们可以为您设计更适合实际系统或者有特殊高级运用的可靠驱动，更欢迎和我们直接进行技术合作和开发。我们强大的技术实力经验和确实以客户着想的理念将是您成功的有力保障！