

杭州菲迪科技有限公司

ERC 实验室

RF2401-FE

开
发
手
册

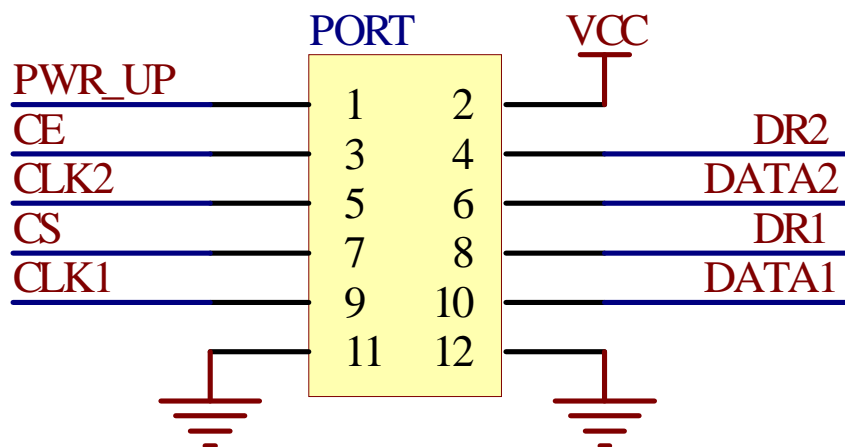


一. 模块介绍



- (1) 2.4—2.5Ghz 开放 ISM 频段免许可证使用
- (2) 高效 GFSK 调制 (1Mbps), 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 (0dBm) 电流为 13mA
RX Mode 电流为 18mA
POWERDOWN Mode 电流为 0.4uA
- (8) 收发模式切换时间 200us

二. 接口电路



- (1) 数字输入管脚，用于控制 2401 模块的内部上电
由 MCU 配置：“0”掉电 “1”上电
- (2) 数字输入管脚，用于使能 2401 模块进入收发模式
由 MCU 配置：“0”其他模式 “1”收发模式
- (3) 数字输入管脚，2401 模块的通道 2 的类 SPI 口线的时钟信号管脚，为读取数据提供时钟源，一般不用
由 MCU 以 2401 的类 SPI 时序给予时钟脉冲
- (4) 数字输入管脚，用于使能 2401 模块进入配置模式
由 MCU 配置：“0”其他模式 “1”配置模式
- (5) 数字输入管脚，2401 模块的通道 1 的类 SPI 口线的时钟信号管脚，为读取和写入数据提供时钟源
由 MCU 以 2401 的类 SPI 时序给予时钟脉冲
- (6) 接地
- (7) VCC 管脚接直流电源，电压范围 1.9V-3.6V，推荐 3.3V

- (8) 数字输出管脚, 用于提示通道 2 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, 一般不用
由 MCU 读入: “0” 没有数据包 “1” 接收到数据包
- (9) 数字输出管脚, 2401 模块通道 2 的类 SPI 口线的数据输出管脚, 一般不用
由 MCU 读入, MCU 以 2401 的类 SPI 时序读取管脚上的数据
- (10) 数字输出管脚, 用于提示通道 1 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
- (11) 数字输出输入管脚, 2401 模块通道 1 的类 SPI 口线的数据传输管脚, 通道 1 的收发数据包数据都由该管脚读取和写入
由 MCU 读写, MCU 以 2401 的类 SPI 时序在管脚上传输数据
- (12) 接地
- a) 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 nRF2401 数据手册或者联系我们。
- b) 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 (实际使用发现 2401 的 IO 承受能力很强, 很多情况可以和 5V 的 MCU 的管脚直接相连)。

c) 对接口顺序或者模块有任何特殊要求的可以联系我们，如果量大我们可以专门定做。

三. 技术参数

nRF2401-FE模块使用Nordic公司的nRF2401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。模块有专门的数字管脚控制其进入掉电模式，从掉电模式唤醒后寄存器信息仍旧保存。该模块很适合高数据量短距离的无线传输，以及低功耗的运用。

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	0	dBm
最小发射功率	-20	dBm
最大传输率	1000	kbps
输出功率 0 dBm工作电流	13	mA
输出功率-20 dBm工作电流	8.8	mA
接收模式时工作电流	18	mA
空闲模式时工作电流	12	uA
掉电模式时工作电流	0.4	uA
温度范围	-40 to +85	°C
典型灵敏度	-93	dBm

四. 工作方式

nRF2401-FE 模块一共有 4 种模式:

收发模式 配置模式 空闲模式 掉电模式

nRF2401-FE 工作模式由 PER_UP、CE、CS 的管脚配置来设定。

PWR_UP	CE	CS	工作模式
1	1	1	收发模式
1	0	1	配置模式
1	0	0	空闲模式
0	X	X	掉电模式

在设定好工作模式后必须按照 2401 芯片内部定义的操作规范进行口线操作来运作 nRF2401-FE 模块。这其中还得牵涉到 2401 模式切换的控制管脚的时序规范，类 SPI 口线的时序规范，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 2401 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 nRF2401-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 nRF2401 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 nRF2401AWrite8Bit(unsigned char byte);  
unsigned char nRF2401ARead8BitChannel_1(void);  
void nRF2401AConfigRegisters(unsigned char* p);  
void nRF2401AWriteTxPayload(unsigned char* p, unsigned char total);  
void nRF2401AReadRxPayload(unsigned char* p, unsigned char total);  
void nRF2401AInitializtion();  
unsigned char nRF2401AScanf(unsigned char* p);  
void nRF2401APrintf(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 nRF2401AInitializtion();
```

2401 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 nRF2401AScanf(unsigned char* p);
```

2401 接收函数，需要一个 28 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数据包会被写入该 28 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 nRF2401APrintf(unsigned char* p);
```

2401 发送函数，需要一个 28 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 28 字节数组中的数据写到 905 中并发送。2401 芯片本身支持地址匹配机制，但考虑到用户使用方便，我们默认将发包地址和本机地址都设定为 0xcc, 0xcc。如有需要可以适当修改驱动以作调整。

代码实例:

```
/*发送范例*/
unsigned char tx_buf[28];
unsigned char rx_buf[28];
void main()
{
    nRF2401AInitialization();
    tx_buf[0] = 0;
    tx_buf[1] = 1; //在发送数组中写入所需信息
    while(1)
    {
        nRF2401APrintf(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[28];
unsigned char rx_buf[28];
void main()
{
    nRF2401AInitialization();
    while(1)
    {
        while(nRF2401AScanf(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
        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[28];
void main()
{
    nRF2401AInitialization();
    while(1)
    {
        while(nRF2401AScanf(buf)!=1); //等待收到数据包
        nRF2401APrintf(buf); //将收到的数据包原样发送
        //注释: nRF2401APrintf() nRF2401AScanf() 两个函数调用顺序完全随意不用考虑两者间的内在影响
    }
}
菲迪 ERC 科技实验室
销售电话: 0571-85355020      技术咨询: 0571-81300937      技术支持 QQ: 117492552
公司网站: http://www.hzfeidi.com.cn      http://www.erc-lab.com
```

```
}
```

其他扩展运用主要，比如本机地址发送地址设定主要修改以下函数和预处理命令设定：

```
void nRF2401AWriteTxPayload(unsigned char* p, unsigned char total)
{
    unsigned char temp;

    nRF2401A_ACTIVE_MODE();
    nRF2401AWrite8Bit(0xcc); //发送地址高 8 位默认设定为 0xcc
    nRF2401AWrite8Bit(0xcc); //发送地址低 8 位默认设定为 0xcc
    for(temp=0; temp<total; temp++)
    {
        nRF2401AWrite8Bit(p[temp]);
    }
    nRF2401A_STANDBY_MODE();
}
```

这里可以将发包地址重新设定为其他的数值（该段函数在 nRF2401A_pub.c 文件中）。

```
#define ADDR1_Byte4 0x0
#define ADDR1_Byte3 0xcc
#define ADDR1_Byte2 0xcc
#define ADDR1_Byte1 0xcc
#define ADDR1_Byte0 0xcc //第一通道接收地址
```

这里可以修改本机接收地址（该段预处理命令在 nRF2401A_pub.h 文件中）。

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 nRF2401_pub.h 文档中，里面有寄存器和 I/O 口线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 nRF2401A_POWER_DOWN_MODE() \
nRF2401A_PWR_UP = 0
//掉电模式
```

可以像调用一般函数一样直接调用 nRF2401A_POWER_DOWN_MODE(); 来进

入 2401 的掉电模式，但这些运用需要用户对 2401 内部原理有一定的了解，请慎用！

再如 2401 的寄存器配置信息区：

```
#define RF_CH    2    //载波频率 freq=2400+RF_CH*1=2400+2*1=2402MHz
```

用户可以将 2 改成其他值如 10，可以使 2401 工作在 $\text{freq}=2400+\text{RF_CH}*1=2400+10*1=2410\text{MHz}$ 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 nRF2401_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 型号的寄存器定义文件，该文件在移植时应按实际情况做相应修改。

```
/******IO 定义(根据实际情况重新定义)*****/  
#define nRF2401A_CE_DDR    DDRD_Bi t5  
#define nRF2401A_CS_DDR    DDRB_Bi t1  
#define nRF2401A_DR1_DDR    DDRB_Bi t2  
#define nRF2401A_CLK1_DDR    DDRB_Bi t3  
#define nRF2401A_DATA_DDR    DDRB_Bi t4  
#define nRF2401A_PWR_UP_DDR    DDRB_Bi t7  
//  
#define nRF2401A_CE    PORTD_Bi t5  
#define nRF2401A_CS    PORTB_Bi t1  
#define nRF2401A_DR1    PORTB_Bi t2  
#define nRF2401A_CLK1    PORTB_Bi t3  
#define nRF2401A_DATA    PORTB_Bi t4  
#define nRF2401A_PWR_UP    PORTB_Bi t7  
//
```

```
#define nRF2401A_DR1_PIN    PINB_Bi t2  
#define nRF2401A_DATA_PIN  PINB_Bi t4
```

这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！