

nRF905 433mhz 接收发射芯片中文资料（附 c 程序）

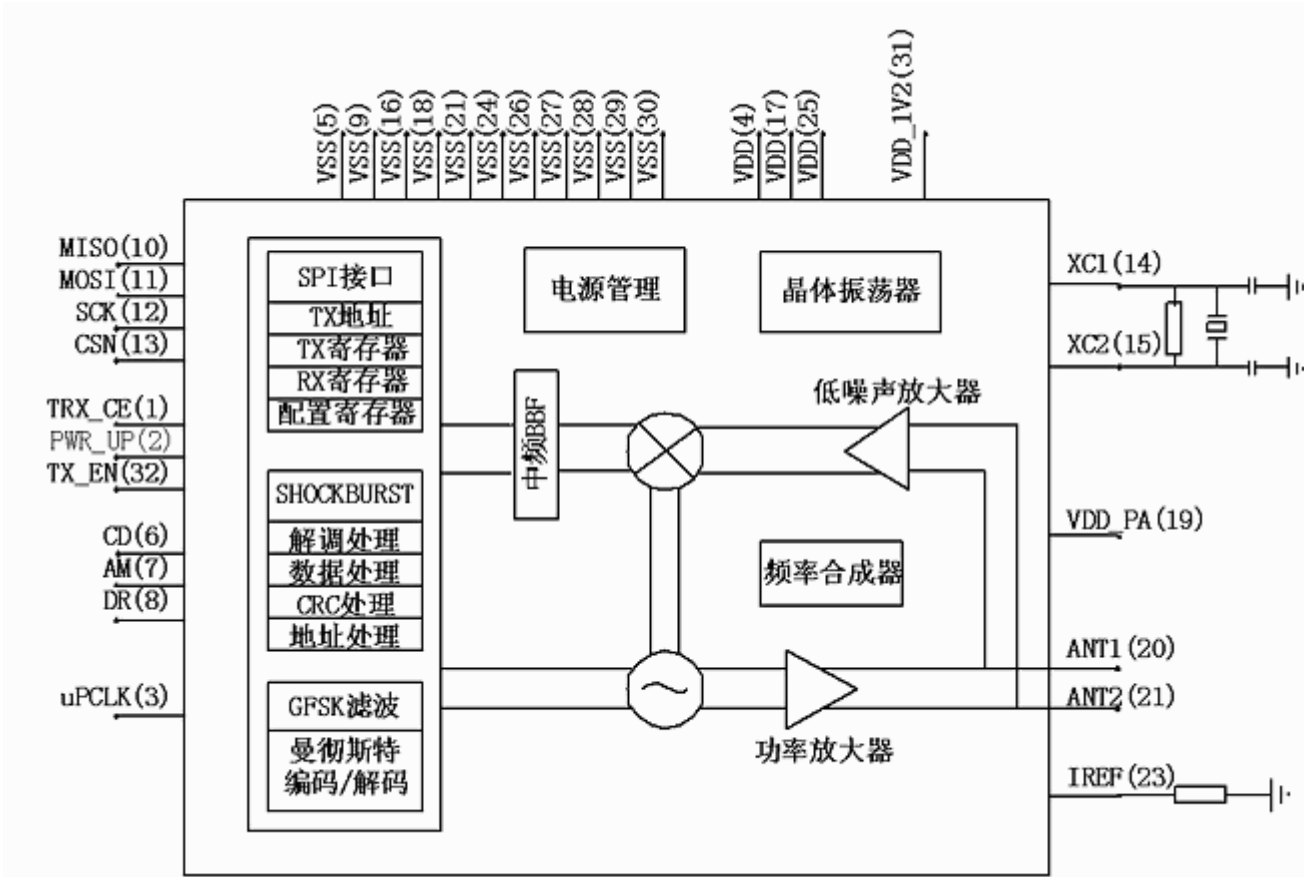
由于还没有 加附件的权限，中文的 pdf 还不能上传
现在发中文简要资料，附带程序，请大家谅解，待可加附件补上

1. 引言

nRF905 是挪威 Nordic VLSI 公司推出的单片射频收发器，工作电压为 1.9~3.6V，32 引脚 QFN 封装 (5×5mm)，工作于 433/868/915MHz 三个 ISM(工业、科学和医学)频道，频道之间的转换时间小于 650us。**nRF905** 由频率合成器、接收解调器、功率放大器、晶体振荡器和调制器组成，不需外加声表滤波器，ShockBurst™ 工作模式，自动处理字头和 CRC(循环冗余码校验)，使用 SPI 接口与微控制器通信，配置非常方便。此外，其功耗非常低，以 -10dBm 的输出功率发射时电流只有 11mA，工作于接收模式时的电流为 12.5mA，内建空闲模式与关机模式，易于实现节能。**nRF905** 适用于无线数据通信、无线报警及安全系统、无线开锁、无线监测、家庭自动化和玩具等诸多领域。

2. 芯片结构、引脚介绍及工作模式

2.1 芯片结构[1]



nRF905 片内集成了电源管理、晶体振荡器、低噪声放大器、频率合成器功率放大器等模块，曼彻斯特编码/解码由片内硬件完成，无需用户对数据进行曼彻斯特编码，因此使用非常方便。**nRF905** 的详细结构如图 1 所示。

2.2 引脚介绍

表 1: **nRF905** 引脚

引脚	名称	引脚功能	描述
1	TRX_CE	数字输入	使 nRF905 工作于接收或发送状态
2	PWR_UP	数字输入	工作状态选择
3	uPCLK	时钟输出	输出时钟
4	VDD	电源	电源正端
5	VSS	电源	电源地
6	CD	数字输出	载波检测
7	AM	数字输出	地址匹配
8	DR	数字输出	数据准备好
9	VSS	电源	电源地
10	MISO	SPI 输出	SPI 输出
11	MOSI	SPI 输入	SPI 输入
12	SCK	SPI 时钟	SPI 时钟
13	CSN	SPI 片选	SPI 片选, 低有效
14	XC1	模拟输入	晶振输入引脚 1
15	XC2	模拟输出	晶振输出引脚 2
16	VSS	电源	电源地
17	VDD	电源	电源正端
18	VSS	电源	电源地
19	VDD_PA	输出电源	给功率放大器提供 1.8V 的电压
20	ANT1	射频	天线接口 1
21	ANT2	射频	天线接口 2
22	VSS	电源	电源地
23	IREF	模拟输入	参考输入
24	VSS	电源	电源地
25	VDD	电源	电源正端
26	VSS	电源	电源地
27	VSS	电源	电源地
28	VSS	电源	电源地
29	VSS	电源	电源地
30	VSS	电源	电源地
31	DVDD_1V2	电源	低电压正数字输出
32	TX_EN	数字输入	等于 1, 发送模式; 等于 0, 接收模式

2.3 工作模式

表 2: **nRF905** 工作模式

PWR_UP	TRX_CE	TX_EN	工作模式
0	×	×	关机模式
1	0	×	空闲模式
1	1	0	射频接收模式
1	1	1	射频发送模式

nRF905 有两种工作模式和两种节能模式。两种工作模式分别是 ShockBurst™ 接收模式和 ShockBurst™ 发送模式, 两种节能模式分别是关机模式和空闲模式。**nRF905** 的工作模式由 TRX_CE、TX_EN 和 PWR_UP 三个引脚决定, 详见表 2。

2.3.1 ShockBurst™ 模式

与射频数据包有关的高速信号处理都在 **nRF905** 片内进行，数据速率由微控制器配置的 SPI 接口决定，数据在微控制器中低速处理，但在 **nRF905** 中高速发送，因此中间有很长时间的空闲，这很有利于节能。由于 **nRF905** 工作于 ShockBurstTM 模式，因此使用低速的微控制器也能得到很高的射频数据发射速率。在 ShockBurstTM 接收模式下，当一个包含正确地址和数据的数据包被接收到后，地址匹配(AM)和数据准备好(DR)两引脚通知微控制器。在 ShockBurstTM 发送模式，**nRF905** 自动产生字头和 CRC 校验码，当发送过程完成后，数据准备好引脚通知微处理器数据发射完毕。由以上分析可知，**nRF905** 的 ShockBurstTM 收发模式有利于节约存储器和微控制器资源，同时也减小了编写程序的时间。下面具体详细分析 **nRF905** 的发送流程和接收流程。

2.3.1.1 发送流程

典型的 **nRF905** 发送流程分以下几步：

- A. 当微控制器有数据要发送时，通过 SPI 接口，按时序把接收机的地址和要发送的数据送给 **nRF905**，SPI 接口的速率在通信协议和器件配置时确定；
- B. 微控制器置高 TRX_CE 和 TX_EN，激发 **nRF905** 的 ShockBurstTM 发送模式；
- C. **nRF905** 的 ShockBurstTM 发送：
 - l 射频寄存器自动开启；
 - l 数据打包(加字头和 CRC 校验码)；
 - l 发送数据包；
 - l 当数据发送完成，数据准备好引脚被置高；
- D. AUTO_RETRAN 被置高，**nRF905** 不断重发，直到 TRX_CE 被置低；
- E. 当 TRX_CE 被置低，**nRF905** 发送过程完成，自动进入空闲模式。

ShockBurstTM 工作模式保证，一旦发送数据的过程开始，无论 TRX_EN 和 TX_EN 引脚是高或低，发送过程都会被处理完。只有在前一个数据包被发送完毕，**nRF905** 才能接受下一个发送数据包。

2.3.1.2 接收流程

- A. 当 TRX_CE 为高、TX_EN 为低时，**nRF905** 进入 ShockBurstTM 接收模式；
- B. 650us 后，**nRF905** 不断监测，等待接收数据；
- C. 当 **nRF905** 检测到同一频段的载波时，载波检测引脚被置高；
- D. 当接收到一个相匹配的地址，地址匹配引脚被置高；
- E. 当一个正确的数据包接收完毕，**nRF905** 自动移去字头、地址和 CRC 校验位，然后把数据准备好引脚置高
- F. 微控制器把 TRX_CE 置低，**nRF905** 进入空闲模式；
- G. 微控制器通过 SPI 口，以一定的速率把数据移到微控制器内；
- H. 当所有的数据接收完毕，**nRF905** 把数据准备好引脚和地址匹配引脚置低；
- I. **nRF905** 此时可以进入 ShockBurstTM 接收模式、ShockBurstTM 发送模式或关机模式。

当正在接收一个数据包时，TRX_CE 或 TX_EN 引脚的状态发生改变，**nRF905** 立即把其工作模式改变，数据包则丢失。当微处理器接到地址匹配引脚的信号之后，其就知道 **nRF905** 正在接收数据包，其可以决定是让 **nRF905** 继续接收该数据包还是进入另一个工作模式。

2.3.2 节能模式

nRF905 的节能模式包括关机模式和节能模式。

在关机模式，**nRF905** 的工作电流最小，一般为 2.5uA。进入关机模式后，**nRF905** 保持配置字中的内容，但不会接收或发送任何数据。

空闲模式有利于减小工作电流，其从空闲模式到发送模式或接收模式的启动时间也比较短。在空闲模式下，**nRF905** 内部的部分晶体振荡器处于工作状态。**nRF905** 在空闲模式下的工作电流跟外部晶体振荡器的频率有关。

3. 器件配置

所有配置字都是通过 SPI 接口送给 **nRF905**。SIP 接口的工作方式可通过 SPI 指令进行设置。当 **nRF905** 处于空闲模式或关机模式时，SPI 接口可以保持在工作状态。

3.1 SPI 接口配置

SPI 接口由状态寄存器、射频配置寄存器、发送地址寄存器、发送数据寄存器和接收数据寄存器 5 个寄存器组成。状态寄存器包含数据准备好引脚状态信息和地址匹配引脚状态信息；射频配置寄存器包含收发器配置信息，如频率和输出功能等；发送地址寄存器包含接收机的地址和数据的字节数；发送数据寄存器包含待发送的数据包的信息，如字节数等；接收数据寄存器包含要接收的数据的字节数等信息。

3.2 射频配置

射频配置寄存器和内容如表 3 所示：

表 3：射频配置寄存器

名称	位宽	描述
CH_NO	9	和 HFREQ_PLL 一起进行频率设置(默认值为 001101100 ₉ =108 ₉)， $f_{RF} = (422.4 + CH_NO/10) * (1 + HFREQ_PLL/10) \text{MHz}$
HFREQ_PLL	1	使 PLL 工作于 433 或 868/915MHz(默认值为 0) '0' - 工作于 433MHz 频段； '1' - 工作于 868/915MHz 频段
PA_PWR	2	输出功率(默认值为 00)， '00' - 10dBm；'01' - 2dBm；'00' + 6dBm；'00' + 10dBm
RX_RED_PWR	1	接收方式节能端，该位为高时，接收工作电流为 1.6mA，但同时灵敏度也降低
AUTO_RETRAN	1	自动重发位，只有当 TRX_CE 和 TXEN 为高时才有效
RX_AFW	3	接收地址宽度(默认值为 100)， '001' - 1byte RX 地址；'100' - 4byte RX 地址
TR_AFW	3	发送地址宽度(默认值为 100)， '001' - 1byte TX 地址；'100' - 4byte TX 地址
RX_PW	6	发送数据宽度(默认值为 100000) '000001' - 1byte 发送数据宽度； '000010' - 2byte 发送数据宽度； ~~~~~ '100000' - 32byte 发送数据宽度
TX_PW	6	接收数据宽度(默认值为 100000) '000001' - 1byte 接收数据宽度； '000010' - 2byte 接收数据宽度； ~~~~~ '100000' - 32byte 接收数据宽度
RX_ADDRESS	32	发送地址标识(默认值为 E7E7E7E7)
UP_CLK_FREQ	2	输出时钟频率(默认值为 11) '00' - 4MHz；'01' - 2MHz；'10' - 1MHz；'11' - 500kHz
UP_CLK_EN	1	输出时钟使能
XOF	3	晶振频率端，必须与外部晶振频率相对应(默认值为 100) '000' - 4MHz；'001' - 8MHz；'010' - 12MHz； '011' - 16MHz；'100' - 20MHz；
CRC_EN	1	CRC 校验使能端，高为使能，默认值为高
CRC_MODE	1	CRC 方式选择端，高为 16 位，低为 8 位，默认值为高

射频寄存器的各位的长度是固定的。然而，在 ShockBurst™ 收发过程中，TX_PAYLOAD、RX_PAYLOAD、TX_ADDRESS 和 RX_ADDRESS 4 个寄存器使用字节数由配置字决定。**nRF905** 进入关机模式或空闲模式时，寄存器中的内容保持不变。

4. 应用电路


```
sbit LCD    =    P3^2;
```

```
//RF 寄存器配置//
```

```
unsigned char idata RFConf[11]=
```

```
{
    0x00,      //配置命令//
    0x6C,      //CH_NO,配置频段在 433.2MHZ
    0x0E,      //输出功率为 10db,不重发，节电为正常模式
    0x44,      //地址宽度设置，为 4 字节
    0x03,0x03, //接收发送有效数据长度为 3 字节
    0xE7,0xE7,0xE7,0xE7, //接收地址
    0xDE,      //CRC 充许，16 位 CRC 校验，外部时钟信号使能，16M 晶振
};
```

```
uchar TxRxBuffer[5];
```

```
bit lcdbit;
```

```
//////////延时//////////
```

```
void Delay(uint x)
```

```
{
    uint i;
    for(i=0;i<x;i++){
        _nop_();
    }
}
```

```
//////////用 SPI 口写数据至 NRF905 内//////////
```

```
void SpiWrite(unsigned char b)
```

```
{
    unsigned char i=8;
    while (i--)
    {
        Delay(10);
        SCK=0;
        MOSI=(bit)(b&0x80);
        b<<=1 ;
        Delay(10);
        SCK=1;
        Delay(10);
        SCK=0;
    }
    SCK=0;
}
```

```
//////////from 905 read data//////////
```

```
unsigned char SpiRead(void)
```

```

{
    register unsigned char i=8;
    unsigned char ddata=0;
    while (i--)
    {
        ddata<<=1 ;
        SCK=0;
        _nop();_nop();
        ddata|=MISO;
        SCK=1 ;
        _nop();_nop();
    }
    SCK=0;
    return ddata;
}

//////////接收数据包//////////
void RxPacket(void)
{
    uchar i;
    i=0;
    while(DR)
    {
        TxRxBuffer[i] = SpiRead();
        i++;
    }
}

/*
;写发射数据命令:20H
;读发射数据命令:21H
;写发射地址命令:22H
;读发射地址命令:23H
;读接收数据命令:24H
*/
void TxPacket(void)
{
    TXEN=1;
    CSN=0;
    SpiWrite(0x22);    //写发送地址,后面跟 4 字节地址//
    SpiWrite(0xE7);
    SpiWrite(0xE7);
    SpiWrite(0xE7);
    SpiWrite(0xE7);
    CSN=1;

```



```

_nop();_nop();
CSN=0;
SpiWrite(0x20);    //写发送数据命令,后面跟三字节数据//
SpiWrite(0x01);
SpiWrite(0x02);
SpiWrite(0x03);
CSN=1;
_nop();_nop();
TRX_CE=1;          //使能发射模式//
Delay(50);         //等带发送完成//
TRX_CE=0;
while(!DR);
}
//////////等待接收数据包//////////
uchar temp;
void Wait_Rec_Packet(void)
{
    TXEN=0;
    TRX_CE=1;
    while(1)
    {
        if(DR)
        {
            TRX_CE=0;    //如果数据准备好，则进入待机模式，以便 SPI 口操作
            CSN=0;
            SpiWrite(0x24);
            RxPacket();
            CSN=1;
            temp=TxRxBuffer[0]+TxRxBuffer[1]+TxRxBuffer[2];
            if(temp==0x06){
                lcdbit=!lcdbit;
                LCD=lcdbit;    //如果接收的数据正确
            }
            break;
        }
    }
}
//////////初始化配置寄存器//////////
void Ini_System(void)
{
    uchar i;
    LCD=0;
    Delay(10000);
    LCD=1;

```

```

    lcdbit=1;
    CSN=1;
    SCK=0;
    PWR=1;
    TRX_CE=0;
    TXEN=0;
    _nop_();
    CSN=0;
    for(i=0;i<11;i++){
        SpiWrite(RFConf[i]);
    }
    CSN=1;
    PWR=1;
    TRX_CE=1;
    TXEN=0;
    Delay(1000);
}

void main(void)
{
    uchar i;
    Ini_System();
    PWR=0;
    while(1)
    {
        Wait_Rec_Packet();    //等待接收完成
        // for(i=0;i<2;i++)
        // Delay(65530);
        TxPacket();
    }
}

#include <reg52.h>
#include <ABSACC.h>
#include <intrins.h>
#include <stdio.h>
#define uint unsigned int
#define uchar unsigned char
//配置口定义//
sbit TXEN    =    P1^7;
sbit TRX_CE =    P1^6;
sbit PWR     =    P1^5;
//SPI 口定义//
sbit MISO    =    P1^2;

```

```

sbit MOSI   =   P1^3;
sbit SCK    =   P1^1;
sbit CSN    =   P1^0;
sbit P2_0   = P2^0;
//状态输出//
sbit DR      =   P1^4;
sbit LCD     =   P3^2;

//RF 寄存器配置//
unsigned char idata RFConf[11]=
{
    0x00,      //配置命令//
    0x6C,      //CH_NO,配置频段在 433.2MHZ
    0x0E,      //输出功率为 10db,不重发, 节电为正常模式
    0x44,      //地址宽度设置, 为 4 字节
    0x03,0x03, //接收发送有效数据长度为 3 字节
    0xE7,0xE7,0xE7,0xE7, //接收地址
    0xDE,      //CRC 充许, 16 位 CRC 校验, 外部时钟信号使能, 16M 晶振
};

uchar TxRxBuffer[5];
bit lcdbit;
//////////延时//////////
void Delay(uint x)
{
    uint i;
    for(i=0;i<x;i++){
        _nop_();
    }
}

//////////用 SPI 口写数据至 NRF905 内//////////
void SpiWrite(unsigned char b)
{
    unsigned char i=8;
    while (i-->0)
    {
        Delay(10);
        SCK=0;
        MOSI=(bit)(b&0x80);
        b<<=1 ;
        Delay(10);
        SCK=1;
        Delay(10);
    }
}

```


```

        SCK=0;
    }
    SCK=0;
}
//////////from 905 read data//////////
unsigned char SpiRead(void)
{
    register unsigned char i=8;
    unsigned char ddata=0;
    while (i--)
    {
        ddata<<=1 ;
        SCK=0;
        _nop();_nop();
        ddata|=MISO;
        SCK=1 ;
        _nop();_nop();
    }
    SCK=0;
    return ddata;
}
//////////接收数据包//////////
void RxPacket(void)
{
    uchar i;
    i=0;
    while(DR)
    {
        TxRxBuffer[i] = SpiRead();
        i++;
    }
}

/*
;写发射数据命令:20H
;读发射数据命令:21H
;写发射地址命令:22H
;读发射地址命令:23H
;读接收数据命令:24H
*/
void TxPacket(void)
{
    TXEN=1;
    CSN=0;

```

```

SpiWrite(0x22);    //写发送地址,后面跟 4 字节地址//
SpiWrite(0xE7);
SpiWrite(0xE7);
SpiWrite(0xE7);
SpiWrite(0xE7);
CSN=1;
_nop();_nop();
CSN=0;
SpiWrite(0x20);    //写发送数据命令,后面跟三字节数据//
SpiWrite(0x01);
SpiWrite(0x02);
SpiWrite(0x03);
CSN=1;
_nop();_nop();
TRX_CE=1;          //使能发射模式//
Delay(50);          //等待发送完成//
TRX_CE=0;
while(!DR);
}
//////////等待接收数据包//////////
uchar temp;
void Wait_Rec_Packet(void)
{
    TXEN=0;
    TRX_CE=1;
    while(1)
    {
        if(DR)
        {
            TRX_CE=0;    //如果数据准备好,则进入待机模式,以便 SPI 口操作
            CSN=0;
            SpiWrite(0x24); 
            RxPacket();
            CSN=1;
            temp=TxRxBuffer[0]+TxRxBuffer[1]+TxRxBuffer[2];
            if(temp==0x06){
                lcdbit=!lcdbit;
                LCD=lcdbit;    //如果接收的数据正确
            }
            break;
        }
    }
}
}
//////////初始化配置寄存器//////////

```

```

void Ini_System(void)
{
    uchar i;
    LCD=0;
    Delay(10000);
    LCD=1;
    lcdbit=1;
    CSN=1;
    SCK=0;
    PWR=1;
    TRX_CE=0;
    TXEN=0;
    _nop_();
    CSN=0;
    for(i=0;i<11;i++){
        SpiWrite(RFConf[i]);
    }
    CSN=1;
    PWR=1;
    TRX_CE=1;
    TXEN=0;
    Delay(1000);
}

```

```

void main(void)
{
    uchar i;
    Ini_System();
    PWR=0;
    while(1)
    {
        Wait_Rec_Packet();    //等待接收完成
        // for(i=0;i<2;i++)
        // Delay(65530);
        TxPacket();
    }
}

```

nrf905 配置寄存器宏定义

nrf905 配置寄存器(10Byte)

工作频率 $f=(422.4+CH_NO/10)*(1+HFREQ_PLL)$ MHz

```

#define RX_ADDRESS  0x00000000    //接收有效地址(本方)
#define TX_ADDRESS  0x02345678    //发送有效地址(对方)

```

```

#define CH_NO_FREQ_422_4MHz 0x000 //工作频率 422.4MHz(433MHz 频段最低频率)
#define CH_NO_FREQ_422_5MHz 0x001 //工作频率 422.5MHz
#define CH_NO_FREQ_425_0MHz 0x01a //工作频率 425.0MHz
#define CH_NO_FREQ_427_5MHz 0x033 //工作频率 427.5MHz

#define CH_NO_FREQ_430_0MHz 0x04c //工作频率 430.0MHz
#define CH_NO_FREQ_433_0MHz 0x06a //工作频率 433.0MHz(433MHz 频段基准频率)
#define CH_NO_FREQ_433_1MHz 0x06b //工作频率 433.1MHz
#define CH_NO_FREQ_433_2MHz 0x06c //工作频率 433.2MHz
#define CH_NO_FREQ_434_7MHz 0x07b //工作频率 434.7MHz
#define CH_NO_FREQ_473_5MHz 0x1ff //工作频率 473.5MHz(433MHz 频段最高频率)

#define CH_NO_FREQ_844_8MHz 0x000 //工作频率 844.8MHz(868MHz 频段最低频率)

#define CH_NO_FREQ_862_0MHz 0x056 //工作频率 862.0MHz
#define CH_NO_FREQ_868_0MHz 0x074 //工作频率 868.0MHz(868MHz 频段基准频率)
#define CH_NO_FREQ_868_2MHz 0x075 //工作频率 868.2MHz
#define CH_NO_FREQ_868_4MHz 0x076 //工作频率 868.4MHz
#define CH_NO_FREQ_869_8MHz 0x07d //工作频率 869.8MHz
#define CH_NO_FREQ_895_8MHz 0x0ff //工作频率 895.8MHz
#define CH_NO_FREQ_896_0MHz 0x100 //工作频率 896.0MHz
#define CH_NO_FREQ_900_0MHz 0x114 //工作频率 900.0MHz
#define CH_NO_FREQ_902_2MHz 0x11f //工作频率 902.2MHz
#define CH_NO_FREQ_902_4MHz 0x120 //工作频率 902.4MHz
#define CH_NO_FREQ_915_0MHz 0x15f //工作频率 915.0MHz(915MHz 频段基准频率)
#define CH_NO_FREQ_927_8MHz 0x19f //工作频率 927.8MHz

#define CH_NO_FREQ_947_0MHz 0x1ff //工作频率 947.0MHz(915MHz 频段最高频率)

#define CH_NO_FREQ CH_NO_FREQ_430_0MHz //工作频率 433.0MHz

#define CH_NO_BYTE CH_NO_FREQ & 0xff //工作频率低 8 位 Byte0 01101100

#define AUTO_RETRAN 0x20 //重发数据包 Byte1.5 0
#define RX_RED_PWR 0x10 //接收低功耗模式 Byte1.4 0
#define PA_PWR__10dBm 0x00 //输出功率-10dBm Byte1.3~2 00
#define PA_PWR_2dBm 0x04 //输出功率+2dBm Byte1.3~2
#define PA_PWR_6dBm 0x08 //输出功率+6dBm Byte1.3~2
#define PA_PWR_10dBm 0x0c //输出功率+10dBm Byte1.3~2
#define HFREQ_PLL_433MHz 0x00 //工作在 433MHz 频段 Byte1.1 0
#define HFREQ_PLL_868MHz 0x02 //工作在 868MHz 频段 Byte1.1
#define HFREQ_PLL_915MHz 0x02 //工作在 915MHz 频段 Byte1.1
#define CH_NO_BIT8 CH_NO_FREQ >> 8 //工作频率第 9 位 Byte1.0 0

```

```

#define TX_AFW_1BYTE    1 * 16    //发送地址宽度 1 字节  Byte2.7~4
#define TX_AFW_2BYTE    2 * 16    //发送地址宽度 2 字节  Byte2.7~4
#define TX_AFW_3BYTE    3 * 16    //发送地址宽度 3 字节  Byte2.7~4
#define TX_AFW_4BYTE    4 * 16    //发送地址宽度 4 字节  Byte2.7~4  100
#define RX_AFW_1BYTE    1          //接收地址宽度 1 字节  Byte2.3~0
#define RX_AFW_2BYTE    2          //接收地址宽度 2 字节  Byte2.3~0
#define RX_AFW_3BYTE    3          //接收地址宽度 3 字节  Byte2.3~0
#define RX_AFW_4BYTE    4          //接收地址宽度 4 字节  Byte2.3~0  100
#define RX_PW_1BYTE     1          //接收数据宽度 1 字节  Byte3.5~0
#define RX_PW_32BYTE    32         //接收数据宽度 32 字节 Byte3.5~0  00100000
#define TX_PW_1BYTE     1          //发送数据宽度 1 字节  Byte4.5~0
#define TX_PW_32BYTE    32         //发送数据宽度 32 字节 Byte4.5~0  00100000
#define RX_ADDRESS_0    RX_ADDRESS >> 24    //接收有效地址第 1 字节 Byte5      11100111
#define RX_ADDRESS_1    (RX_ADDRESS >> 16) & 0xff //接收有效地址第 2 字节 Byte6
11100111#define RX_ADDRESS_2    (RX_ADDRESS >> 8) & 0xff //接收有效地址第 3 字节 Byte7
11100111
#define RX_ADDRESS_3    RX_ADDRESS & 0xff      //接收有效地址第 4 字节 Byte8 11100111

#define CRC_MODE_16BIT   0x80    //CRC16 模式      Byte9.7  1
#define CRC_MODE_8BIT    0x00    //CRC8 模式      Byte9.7
#define CRC_EN           0x40    //CRC 使能      Byte9.6  1
#define CRC16_EN         0xc0    //CRC16 模式使能 Byte9.7~6  11
#define CRC8_EN          0x40    //CRC8 模式使能 Byte9.7~6
#define XOF_20MHz        0x20    //晶体振荡器频率 20MHz Byte9.5~3
#define XOF_16MHz        0x18    //晶体振荡器频率 16MHz Byte9.5~3  100
#define XOF_12MHz        0x10    //晶体振荡器频率 12MHz Byte9.5~3
#define XOF_8MHz         0x08    //晶体振荡器频率 8MHz  Byte9.5~3
#define XOF_4MHz         0x00    //晶体振荡器频率 4MHz  Byte9.5~3
#define UP_CLK_EN        0x40    //输出时钟使能   Byte9.2  1
#define UP_CLK_FREQ_500kHz 0x03    //输出时钟频率 500kHz Byte9.1~0  11
#define UP_CLK_FREQ_1MHz  0x02    //输出时钟频率 1MHz   Byte9.1~0
#define UP_CLK_FREQ_2MHz  0x01    //输出时钟频率 2MHz   Byte9.1~0
#define UP_CLK_FREQ_4MHz  0x00    //输出时钟频率 4MHz   Byte9.1~0

#define UP_CLK_EN_500kHz  0x43    //输出时钟频率 500kHz Byte9.2~0  111
#define UP_CLK_EN_1MHz    0x42    //输出时钟频率 1MHz   Byte9.2~0
#define UP_CLK_EN_2MHz    0x41    //输出时钟频率 2MHz   Byte9.2~0
#define UP_CLK_EN_4MHz    0x40    //输出时钟频率 4MHz   Byte9.2~0

#define TX_ADDRESS_0     TX_ADDRESS >> 24    //发送有效地址第 1 字节
#define TX_ADDRESS_1     (TX_ADDRESS >> 16) & 0xff //发送有效地址第 2 字节
#define TX_ADDRESS_2     (TX_ADDRESS >> 8) & 0xff //发送有效地址第 3 字节
#define TX_ADDRESS_3     TX_ADDRESS & 0xff      //发送有效地址第 4 字节

```


基于 nRF905 的无线数据传输设备设计

引言

无线通信在机动性要求较强的设备中或人们不方便随时到达现场的环境下得到了越来越广泛的应用，如无线数据采集、无线设备管理和监控、汽车仪表数据的无线读取等都是其典型应用。微功率短距离无线通信技术作为无线通信实用技术，一般使用单片射频收发芯片，加上微控制器和少量外围器件构成专用或通用无线通信模块，通常射频芯片采用 GFSK(高斯频移键控)调制方式，工作于 ISM(工业、科学、医疗)频段，通信模块包含简单透明的数据传输协议或使用简单的加密协议，用户不必对无线通信原理和工作机制有较深的了解，只要依据命令字进行操作即可实现基本的数据无线传输功能，因其功率小、开发简单快速而在工业、民用等领域应用广泛。本文介绍利用 ATmega16单片机和无线数据收发芯片 nRF905构成的短距离无线数据传输设备，给出了硬件和软件设计方案。

1 系统硬件设计

1.1 系统结构

无线数据传输系统结构如图1所示。该系统由外部数据设备和无线数据传输模块组成，外部数据设备为 PC 机或数据采集等设备，我们设计的主要是无线数据传输模块。无线数据传输模块基于微功耗单片射频收发器 nRF905 设计，采用 Atmel 公司的高性能、低功耗8位处理器 ATmega16 为主处理芯片，完成数据的处理和

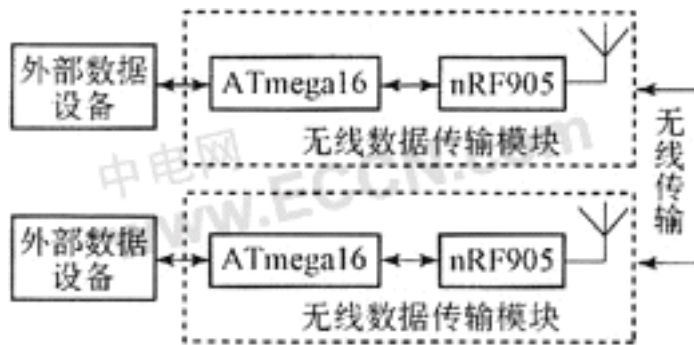


图 1 无线数据传输系统框图

1.2 ATmega16和 nRF905

Atmel 公司的 ATmega16 单片机具有先进的 RISC(精简指令集计算机)结构、非易失性程序和数据存储器，16 kB 可编程 Flash 存储器、512 B 的 EEPROM 和 1 kB 片内 SRAM，具有丰富的外设接口，其 USART(通用同步和异步接收器和转发器)是一个高度灵活的串行通信设备，SPI(串行外设接口)允许 ATmega16 与外设或其他 AVR 器件进行高速的同步数据传输。

nRF905 是挪威 Nordic VLSI 公司推出的单片射频收发器，工作电压为 1.9 V~3.6 V，工作于 433 / 868 / 915 MHz 这 3 个 ISM 频段，频道转换时间小于 650 μs，最大数据速率为 100 kbit / s。nRF905 由频率合成器、接收解调器、功率放大器、晶体振荡器和 GFSK 调制器组成，不需外加声表面滤波器，ShockBurst™ 工作模式，自动处理字头和 CRC(循环冗余检验)，使用 SPI 接口与微控制器通信，配置非常方便。此外，其功耗很低，以 -10 dBm 输出功率发射时电流只有 11 mA，工作于接收模式时的电流为 12.5 mA，具有窄闲模式与关机模式，易于实现功率管理。

1.3 硬件电路

硬件电路主要由电源与复位电路、外部数据设备接口电路、单片机系统和 nRF905 应用电路等几部分组成。硬件电路如图 2 所示。

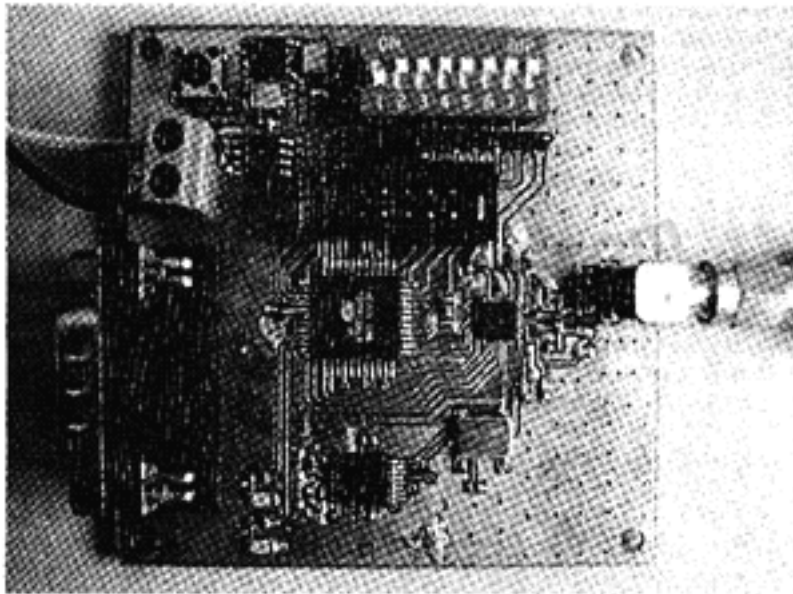


图2 无线数据传输模块硬件电路

1.3.1 电源与复位电路

nRF905和单片的典型工作电压为+3.3 V，而系统供电电源为+5V，所以采用低压差线性稳压器 TPS7333 实现+5 V~+3.3 V的线性稳压。为了实现稳定、可靠的复位，使用低电压工作的复位芯片 TPS70733产生复位信号。

1.3.2 外部数据设备接口

无线数据传输模块与外部数据设备之间采用 RS-232接口，ATmega16的 PD0-PD1用于连接 RS-232串口。通常，PC 机与单片机用两根线方式进行全双工异步通信。由于 AVR 单片机输入输出为 TTL 电平，PC 机配置的是 RS-232标准串行接口，二者电气规范不一致，因此，使用 ICL3221收发芯片实现串口电平转换。

数据传输速率在板可设置或通过外部数据设备设置。在板波特率利用 ATmega16的 PA7、PA6两位设置，可设置为9.6kbit / s、19.2 kbit / s、38.4kbit / s、115.2 kbit / s。利用外部数据设备设置波特率时，单片机的初始数据传输速率为9.6 kbit / s，PA7、PA6置为00状态，当单片机收到波特率设置命令后，数据传输速率调整为设定值。

1.3.3 单片机与 nRF905接口电路

单片机与 nRF905的接口电路很最要。nRF905内部有5个寄存器：状态寄存器、配置寄存器、发射地址寄存器、发射数据寄存器和接收数据寄存器。除了对寄存器读写外，还需对 nRF905工作模式的切换进行控制。单片机与 nRF905的信号连接见图3。

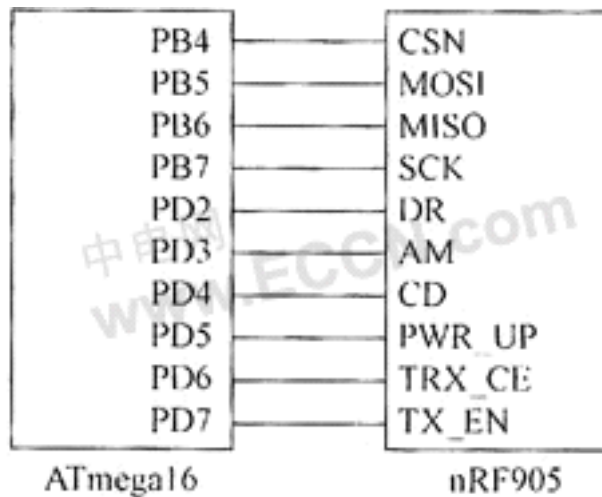


图 3 单片机与 nRF905 的信号

ATmega16与 nRF905之间的双向数据传输使用 SPI 接口，单片机的 PB7-PB4连接 nRF905的 SPI 接口，PD2-PD7连接 nRF905的控制信号和检测信号，用于 nRF905的模式切换以及通信过程中必须的信号指示接口。

2 系统软件设计

2.1 数据传输过程

PC 机(或其他外部设备)有数据传输或需设置设备参数时，通过串口将数据发送给单片机，单片机接收数据后，将需发送的数据(这里包括目标设备地址和所要发送的数据)通过 SPI 接口发送给 nRF905，nRF905将数据加前导码和 CRC 码，将数据包发送。

当 nRF905接收到有效数据后，DR 置高，单片机检测到 DR 为高电平后，复位 TRX_CE 引脚，使 nRF905进入空闲模式，通过 SPI 接口从 nRF905中读出接收数据，然后通过 USART 传送给 PC 机或其他外部设备。软件功能模块由 CPU 寄存器初始化、串行口初始化、串口收发送程序、SPI 初始化、SPI 收发送程序、I/O 口初始化、nRF905配置寄存器操作、nRF905接收程序、发送程序、主程序模块组成。下面简要介绍主要的软件功能模块。

2.2 USART 串口软件设计

AVR USART 与 AVR UART 在寄存器位定义、波特率发生器、发送器操作、发送缓冲器的功能以及接收器操作等方面完全兼容，此外，接收缓冲器进行了两方面改进：增加了一个缓冲器；接收移位寄存器可以作为第3级缓冲。

2.2.1 串口数据帧格式

外部数据设备与无线数据传输设备间的双向数据传输使用相同的帧格式，帧格式由帧头、帧长、帧标志和数据组成。帧头为数据帧开始标志，固定为0FF81H，长度2字节。帧长指从帧标志开始至本帧结束的所有数据的字节数，不包括帧头、帧长本身，单位为字节，帧长占1字节。帧标志用以指示本帧数据的内容属性，长度为1字节。不同类型帧的数据长度和帧标志具体定义如表1所示。

表1 数据长度和帧标志定义

方 向	内 容	数据长度/字节	帧标志数值
外部数据设备 至无线设备	波特率设置	1	01H
	设备地址设置	4	02H
	发射功率	1	03H
	工作频率	1	04H
	发送数据	不超过 254	05H
无线设备至 外部数据设备	波特率已设置	1	12H
	设备地址已设置	4	13H
	发射功率已设置	1	14H
	工作频率已设置	1	15H
	接收数据	不超过 254	16H

数据指所传输的业务等内容，数据长度见表1，数据内容定义如下：

- a) 波特率设置：01H~0AH 对应波特率(单位为 kbit / s)为2.4、4.8、9.6、14.4、19.2、28.8、38.4、57.6、76.8、115.2。
- b) 设备地址设置：设备地址为00000000H~FFFFFFFFH。
- c) 发射功率：00H 为低功率；01H 为高功率。
- d) 工作频率：433 MHz 频段，信道间隔100 kHz。
- e) 发送数据：发送数据长度不定，最长不超过254字节。

2.2.2 USART 初始化

初始化 USART 操作包括波特率设置、数据格式和 UCSRB 寄存器设置。USART 的波特率寄存器 UBRR 和降序计数器相连接，一起构成可编程的预分频器或波特率发生器。UBRR 值的计算由该公式完成： $UBRR = f_{osc} / (16baud) - 1$ ，其中 Baud 为波特率， f_{osc} 为系统时钟频率。通过设置 UCSRC 寄存器，设置数据格式为8位数据位和1位停止位。通过设置 UCSRB 寄存器，使能串口发送和接收，并响应接收完成中断。

2.2.3 数据发送和接收

数据发送采用查询方式。置位 UCSRB 寄存器的发送允许位 TXEN 将使能 USART 的数据发送，将需要发送的数据加载到发送缓冲区将启动数据发送，加载过程为 CPU 对 UDR 寄存器的写操作。发送数据时，按照帧格式在所需发送的数据前加上帧头、帧长、帧标志组帧发送。

数据接收采用中断方式。置位 UCSRB 寄存器的接收允许位 RXEN 将启动 USART 的数据接收器，通过读取 UDR 寄存器就可以获得接收缓冲区的内容。接收数据时，帧标志有效才能开始接收一帧数据，并根据读出的帧长信息完成接收规定长度的数据。

2.3 SPI 接口软件设计

本设计中 SPI 配置为主机模式，nRF905 为从设备。SPI 波特率最高可设置为 1 / 2 系统时钟，系统时钟为 8 MHz，因此 SPI 速率可达 4 MHz。此外，正确选择 SPI 的工作模式对 SPI 数据传输非常重要，ATmega16 的 SCK 的相位和极性有 4 种组合，SPI 工作模式由 CPOL、CPHA 设置，根据 nRF905 的 SPI 读写时序，ATmega16 的 SPI 工作模式应设置为模式 0。

ATmega16 与 nRF905 同时进行双向数据传输。ATmega16 配置为 SPI 主机时，SPI 接口不自动控制 SS 引脚，由用户软件来控制。ATmega16 通过将从机的 CSN 引脚置低实现与从机的同步。SPI 时钟由写入到 SPI 发送缓冲寄存器的数据启动，SPI MOSI 引脚上的数据发送次序由寄存器 SPCR 的 DORD 位控制，置位时数据的 LSB(最低位)首先发送，否则数据的 MSB(最高位)首先发送。我们选择先发送 MSB，同时接收到的数据传送到接收缓冲寄存器，CPU 进行右对齐从接收缓冲器中读取接收到的数据。应该注意，当需要从 nRF905

中读取多个数据时，即使 nRF905 并不需要 ATmega16 串行输出的数据，每读取一个数据前都要向 SPI 发送缓冲器写一个数据以启动 SPI 接口时钟。由于 SPI 系统的发送方向只有 1 个缓冲器，而在接收方向有 2 个缓冲器，所以在发送时一定要等到移位过程全部结束后，才能对 SPI 数据寄存器执行写操作；而在接收数据时，需要在下一个字节移位过程结束之前通过访问 SPI 数据寄存器读取当前接收到的数据，否则第 1 个数据丢失。

2.4 nRF905 配置及收发流程

对 nRF905 寄存器的操作是一个很关键的问题，nRF905 的所有配置都是通过 SPI 接口进行的。nRF905 的 SPI 接口只有在掉电模式和 standby 模式是激活的。当 CSN 为低时，SPI 接口开始等待一条指令，任何一条新指令均由 CSN 由高到低的转换开始。

nRF905 发送模式工作过程如下：

- a) 当 ATmega16 发送数据时，将接收设备地址和所要发送的数据通过 SPI 接口写入 nRF905，SPI 传输速率由初始化设置。
- b) 置位 TRX_CE、TX_EN，激活 nRF905 发送模式。
- c) nRF905 自动完成数据打包(加入前导码和 CRC)，包经过 GFSK 调制以 100 kbit / s 发送，当传输完毕 DR 置位。
- d) 如果将 AUTO_RETRAN 位置高，nRF905 将连续发送数据包，直至将 TRX_CE 引脚复位。
- e) 当 TRX_CE 引脚被设置为低时，nRF905 结束发送模式，并进入 standby 模式。

nRF905 接收模式工作过程如下：

- a) 将 TRX_CE 置位，TX_EN 复位后 650μs，nRF905 进入接收模式等待数据到来。
- b) 当 nRF905 在接收信号检测到载波，则 CD(carrier detect)引脚置位；然后，如果接收到有效地址则 AM(address match)置位，最后将接收到的有效数据包去掉前导码、地址，CRC 正确后，将 DR(data ready)引脚置位。
- c) CPU 复位 TRX_CE 引脚，使 nRF905 进入空闲模式，然后通过 SPI 接口读取数据。
- d) 数据接收完毕后，nRF905 DR 和 AM 引脚复位并准备进入下一个工作模式。

应该注意的是，在数据发送过程中无论将 TRX_CE、TX_EN 怎样设置，nRF905 都会完成此次发送而不受影响，此后，进入所设置的工作模式。而在接收数据包的过程中 TRX_CE 或 TX_EN 状态改变，则 nRF905 会立即改变工作模式，丢失数据。

2.5 主程序流程

主程序流程图如图 4 所示。

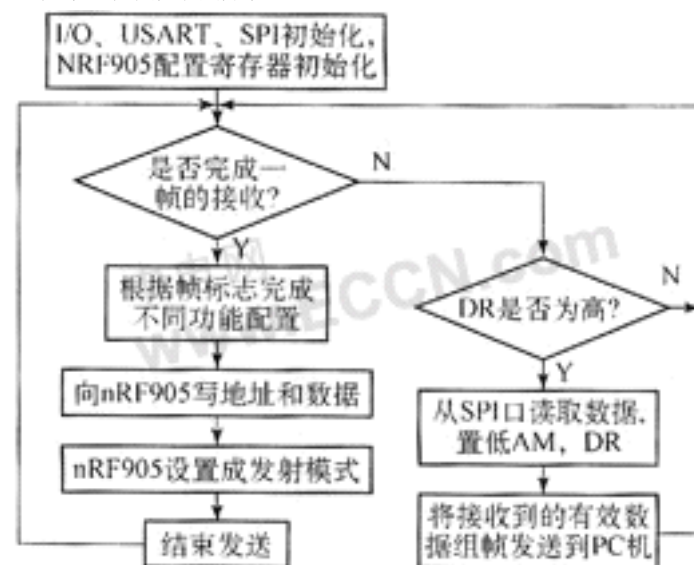


图 4 主程序流程

3 结束语

我们采用 nRF905 射频收发芯片和 ATmega16 微控制器设计了短距离无线数据传输设备,完成硬件电路和系统软件调试后,进行了无线数据收发实验。实验结果表明,在 300 m 通信距离,该无线传输设备工作稳定,能实现数据的高速有效传输,具有低功耗、抗干扰能力强等优点。

nRF905 的相关资料

nRF905 是 Nordic VLSI 公司推出的一款无线收发芯片。32 脚封装(32L QFN 5 x 5mm),供电电压为 1.9~3.6V,工作于 433/868/915MHz 三个 ISM(工业、科学和医学)频道。可自动处理字头和 CRC(循环冗余码校验)。微处理器可以通过 SPI 接口及相关指令访问 nRF905 的寄存器。功耗低,高抗干扰 GFSK 调制,可跳频,载波检测输出,地址匹配输出以及数据就绪输出。nRF905 适用于遥感、遥测、无线抄表、工业数据采集以及家庭自动化等领域。

由于 nRF905 具有 ShockBurst™ 功能,使得 nRF905 不需要使用昂贵的高速微控制处理器(MCU)对数据处理/时钟恢复,也能达到较高的数据率。通过在芯片上将所有的高速信号处理变为射频通信协议,nRF905 芯片提供了一个具有微控制器能力的 SPI 接口,数据率由具有微控制器功能的接口速率自行设定。收发电路的数字部分是一个低速率电路,而收发电路的射频链接却是一个处于最高速率的电路,整个电路要通过变速才能解决速率上的差异。nRF905 芯片的 ShockBurst™ 模式减少了在这一过程中的平均电流消耗。在 ShockBurst™ RX 模式中,当一个有效地址的数据包被接收时,能够通过 AM 和 DR 两个信号外送给 MCU。在 ShockBurst™ TX 模式中,nRF905 芯片自动地完成报头的生成和 CRC 校验,当发送过程完成后,能够通过 DR 信号外送给 MCU,发送工作已经完成。这样可以降低 MCU 对内存的要求,使得 MCU 实现了低成本,同时也缩短了软件的开发周期。

单片机选用 ATMEL 公司生产的 AT89LV51 单片机。它具有低功耗、低电压(与 nRF905 共用同一电压)的特性,它既适合结构比较简单的应用系统,也适合于比较复杂的实时系统。单片机主要完成两个方面的工作,一方面完成对射频芯片通信过程的控制,另一方面通过 RS-232 总线与上位机相连。由于 AT89LV51 内部没有集成 SPI 接口,因此通过软件模拟的方法来实现与 nRF905 的 SPI 通信。硬件连接上,由 P2 口、P3.2、P3.3、P3.5 连接到 nRF905 模块的连接器相应的引脚上。电路如图 1 所示。

【<http://wenku.baidu.com/view/a5c235661ed9ad51f01df2f0.html>

nRF905 无线收发芯片原理及设计实现】

<http://wenku.baidu.com/view/b9315ee2524de518964b7dee.html>

nRF905 中文手册