基于单片机 MSP430和 nRF905的无线通信模块

高章飞,朱善安
(浙江大学 电气工程学院,浙江 杭州 310027)

摘 要:介绍了一种基于 MSP430和 nRF905的无线通信模块,以及相关的 MSP430SPI驱动、nRF905驱动及接口实现。

关键词: 单片机;通信
中图分类号: TN923 **文 献标识码**: A **文章编号**: 1001 - 4551(2006)02 - 0039 - 05

A Wireless Transceiver Based on MSP430 and nRF905

GAO Zhang-fei, ZHU Shan-an
(College of Electrical Engineering, Zhejiang University, Hangzhou 310027, China)

Abstract: This article presents a wireless transceiver based on MSP430 and nRF905, first the realization of the hardware is put forward, then the driver for SPI of MSP430 is introduced, finally the driver for nRF905 is given.

Key words: signle chip microcomputer; communication

系统采用通用的低功耗单片机 MSP430作为主芯片, nRF905作为发射模块,利用 SPI口实现双向通信, SPI支持高速数据传输,从而满足了射频带宽的要求。

nRF905提供了强大的跳频机制以及大量的频道支持,可以用在许多特殊的场合,而且即使利用无增益的 PCB天线,其传输距离也可达 200m。如果需要传输更远距离,也可以改成带增益的天线,传输距离即可扩大到 1km以上,可满足不同客户的需求。

1 系统硬件实现

无线通信模块的实现框图如图 1所示。除了 MSP430和 nRF905外,系统还留有 MAX232接口可以实现与 PC机的通信,MAX485接口满足一些通用仪器仪表的要求,并提供了按键和液晶等人机交互界面。

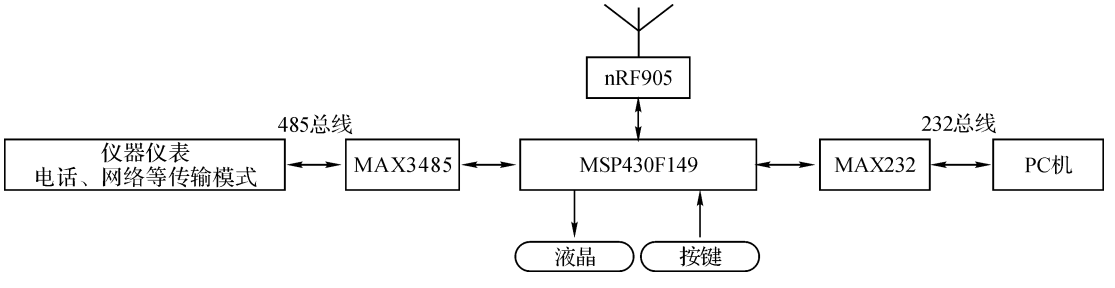

图 1 系统的硬件结构

nRF905与 MSP430接口如图 2所示,其中 MOSI、MISO、SCK分别与主机 SPI口对应, CSN、TRX_CE、PWR_UP、TX_EN接通用 I/O口,而 CD、AM、DR接中断口, MSP430的 P2口都是复用的中断口,这样收到数据可以以中断方式及时通知 MSP430。

2 驱动实现

2.1 MSP430的 SPI驱动

MSP430用标准 SPI口和 nRF905进行通信,标准接口包括两根数据线: MOSI(主发从收)和 MISO

收稿日期: 2005 - 04 - 15 **修订日期**: 2005 - 06 - 27
作者简介: 高章飞(1982 -),男,安徽芜湖人,硕士研究生,研究方向:嵌入式系统。

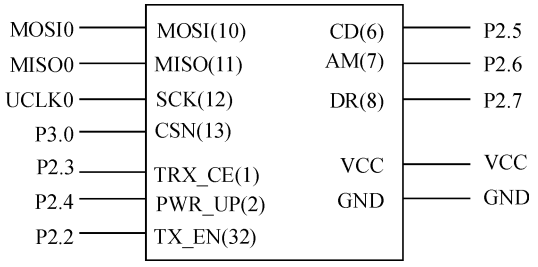

图 2 接口连接

(从发主收),还有时钟线 CLK,主机用 CLK与从机时钟同步。

SPI可以理解成双工方式,因为在发送数据的同时也可以接受数据。SPI分成主模式和从模式,从模式完全被动,数据的发送完全由主机掌握。实际上参与工作的都有 4个寄存器,主机将数据写入发送缓存 UTXBUF,数据并行存入发送移位寄存器,数据一旦写入 UTXBUF,立即从 MOSI线移位到从机的接受移位缓存,而从机移位缓存中的数据又将其发送移位寄存器中的数据通过 MISO移位到主机的接受移位寄存器,再并行读入接受缓存中。所以,利用 SPI既可以读数据,也可以写数据。读写完毕后可以选用中断通知 MSP430,也可以采用轮询方式。为了降低系统复杂性,同时考虑到 nRF905最多一次只操作 32B,所需时间不长,所以采用轮询方式。

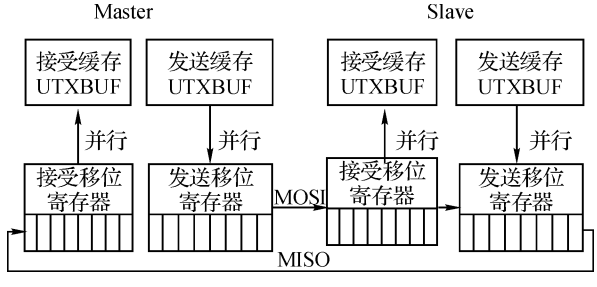

图 3 430SPI示意图

部分代码如下:

```
//实现 SPI读写功能,写入 data的同时也会 nRF905指定地址中的值
Char SpiReadWrite(char data)
{
    //禁止 SPI中断
    IE1 &= ~UTXIE0;
    IE1 &= ~URXIE0;
    TRXBUF0 = data;
    //等待数据发送完和接受完
    While(((IFG1 &UTXIFG0) == =0)||((IFG1&URXIFG0) == =0));
    //返回读入的值,不需要可以舍弃
    Return RXBUF0;
}
```

2.2 nRF905的驱动

nRF905共有 32个引脚,其中有 10个引脚尤其需要注意,如与主 MCU通信的 SPI接口的 4个引脚:数据线 MOSI、MISO;时钟线 SCK;使能线 CSN。其中 CSN可以接到一个 IO口,来模拟时序,而其它 3个脚则接到主 MCU的 SPI接口;主 MCU的控制线有 3个引脚:(1)控制低功耗的 PWR_UP;(2)控制正常工作的 TX_EN;(3)选择发送还是接受方式的 TRX_CE。这几个引脚都接到主 MCU的通用 IO口;nRF905的反馈线有 3根,检测到频道正被使用的 CD(carrier detected),通知接受地址正确的 AM(address matched),告诉 MCU数据接受正确的 DR(data received),这几个引脚需要接到主 MCU的中断引脚上,当接收数据正确时以中断方式通知主 MCU。

2.2.1 寄存器操作

2.2.1.1 寄存器介绍

nRF905提供了数据线和控制线,却没有提供地址线。为能对其内部地址进行操作,nRF905提供了特殊的命令字来直接选定内部的寄存器,这样就屏蔽了复杂的地址运算,使 nRF905的使用更加简单。

主要的寄存器有 4个,分别如下:

(1)配置寄存器。包括频道、频段、功率、CRC校验等设置,其中还包括了自己节点的地址配置,即 RX_ADDRESS,相当于给节点配置了一个 ID,在网络中可以通过对接受数据的目标 ID和自己 ID比较来判断该数据是否属于自己。如果是,则 AM返回高电平通知 MCU;如果不是,则 nRF905直接舍弃该数据。如果不对 RX_ADDRESS设置,则会默认为广播地址,这样会接受所有广播信息。

(2)发送地址寄存器。即 TX_ADDRESS,通过对该寄存器操作可以确定要发送的数据要发送到的节点。当发送数据时,该信息会自动加在包头。如果没有设置,则会缺省为广播地址,那么发出的信息就是广播消息了。

数据帧格式:

ADDR + PAYLOAD

其中,ADDR就是欲建立连接的地址,就是写在 TX_ADDRESS中的信息,所以每次发送数据前都要对该

寄存器进行设置,否则发送的就是广播消息。

发送包格式:

Preamble + ADDR + PAYLOAD + CRC

其中,Preamble自动加入的,nRF905也会自动处理,而 CRC也由芯片硬件电路自行计算并附在包尾,用户可以对配置寄存器进行操作来选择是否要 CRC校验,以及选择 8位还是 16位。

(3)发送数据寄存器,即 TX_PAYLOAD。该寄存器一共有 32B,故 nRF905一次只能发送 32B,每次用户将欲发送的数据写入该寄存器,再由控制引脚进行发送操作。

(4)接受数据寄存器,即 RX_PAYLOAD。该寄存器也有 32B,即一次只能接受 32B,当接受数据正确时,DR会通知 MCU读取寄存器中的数据。

2.2.1.2 操作寄存器

nRF905留有 4个数据线接口,包括 3个标准 SPI口和一个使能端 CSN:其下降沿使能寄存器,如果希望对某个寄存器进行操作的时候,首先需要将 CSN引脚置低,而操作完后则要将 CSN恢复成高电平,便于下次操作。905提供了特殊的命令字支持对寄存器的操作,如,写配置寄存器的命令字为(WC)0000 XXXX,读配置寄存器的命令字为(RC)0001XXXX,其中 XXXX为起始地址。当需要操作某个寄存器时,先写入该寄存器的命令字,这样就相当于选中了该寄存器的基地址,然后依次从基地址读写就可以了。但要注意每个寄存器都有相应的大小,不能越界,否则会对其它寄存器造成影响而出错。实现配置寄存器中发送和接受字节数的函数:

```
//配置寄存器起始地址
#define WC 0x00
//nTx是要发送的字节数
//nRx是要接受的字节数
InitLength(char nRx, char nTx)
{
    //使能配置寄存器
    CSN = LOW;

    //写入命令字,选择接受字节数寄存器
    SpiReadWrite(WC | 0x03);

    //写入这次操作可以接受的字节数
    SpiReadWrite(nRx);

    //下一个寄存器即是发送字节数寄存器
    //写入这次操作可以发送的字节数
    SpiReadWrite(nTx);

    //配置完毕,恢复使能位,为下次操作做准备
    CSN = HIGH;
}
```

2.2.2 设置频道和频段

nRF905最吸引人的一个特点就是提供跳频支持,以及拥有大量的频道可使用。nRF905可以在 433/868/915频段进行通信,其实 868和 915属于同一频段,即主要分两大频段,而每一频段又有 2^9 个频道可以使用,但实际上针对不同的天线,只有一个频段可以让芯片发挥最好的功能,所以一种天线有 2^9 频道使用。当在某个频道上遇到干扰时,可以跳频来继续通信,确保数据完整性。配置寄存器提供了 CH_NO和 HFREQ_PLL来设置频道,公式为:

$$f_{op} = (422.4 + (CH\_NO/10)) \times (1 + HFREQ\_PLL)\,MHz$$

式中,HFREQ_PLL为 1位寄存器;0为 433频段;1则是 868/915频段;而 CH_NO为 9位寄存器,来选择具体的频道。

同时还有一个 2位寄存器 PA_PWR可以设置发送功率,缺省为 0,对应 -10dBm,1对应 -2 dBm,2对应 +6 dBm,最大 3对应 +10dBm。

nRF905提供了一个专门的命令字来支持快速跳频,其代码实现如下:

```
//快速配置频道,功率命令字
#define CC 0x80
//power输出功率 2位
//hfreq主频段选择:0为 433频段 ,1为 866/915频段
//channel频道选择:9位
Void SwitchChannel(char power, char hfreq , short channel)
{
    CSN = LO;
    //POWER在第二位,HFREQ_PLL在第一位
    SpiReadWrite(CC | (power << 2) | (hfreq << 1));
    SpiReadWrite(channel);
    CSN = HIGH
}
```

2.2.3 发送数据流程

设置好配置寄存器后,就可以发送数据了,先给出具体的时序图,如图 4所示。再解释具体流程。

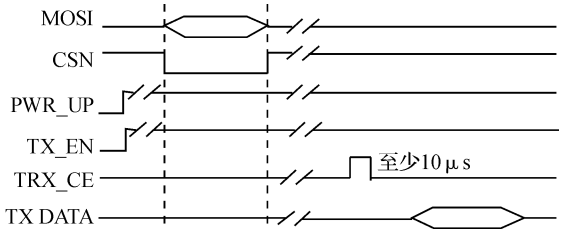

图 4 发送时序图

(1)主 MCU将 PWR_UP置高,使 nRF905进入工作模式,再将 TX_EN置高进入发送数据模式。

(2)将发送地址通过 SPI口写入发送地址寄存器 TX_ADDRESS,再将数据写入发送数据寄存器 TX_PAYLOAD, SPI口的速度由主 MCU设置。

(3)主 MCU置高 TRX_CE, nRF905自动将数据帧格式补齐,加入包头 preamble,并根据寄存器设置计算 CRC校验填入包尾,然后 nRF905将整个数据以 100B/s的速度,采用曼彻斯特编码,以 GFSK形式发送出去。发送完毕, DR会置高,通知主 MCU可以继续下次发送。

(4)如果配置成自动重发模式, nRF905会自动重发,直到 TRX_CE置低。

(5)发送完后可以将 TRX_CE置低,这样就进入 standby模式,实际操作时可以直接将 TRX_CE产生脉冲,持续时间不少于 10μs,就可以发送完数据。

发送数据流程图如图 5所示。

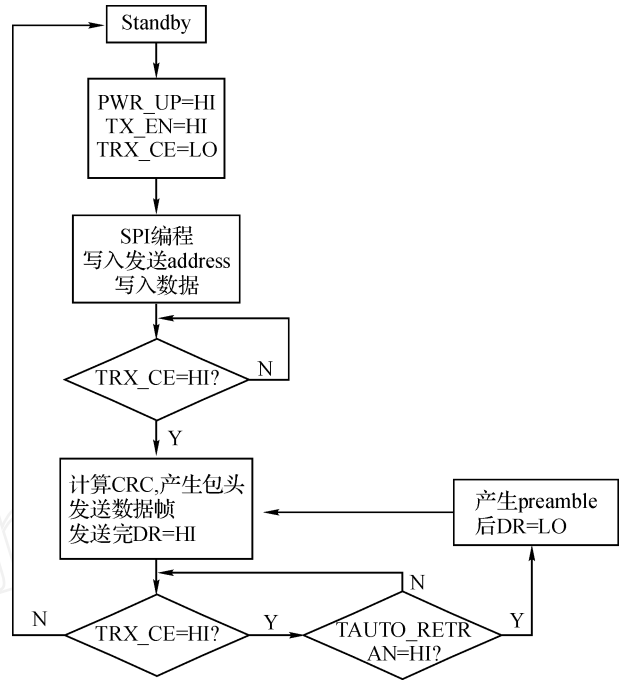

图 5 发送数据流程图

部分实现代码:

```
//写发送数据寄存器命令字
#define WTP 0x20

//发送一个单元, nRF905最多一次发送 32B
//buf:发送数据的投指针
//txnum:一次发送的字节数,不大于 32
//rxnum:接受到的确认数,发送完直接转到接受状态等待确认信息
Void Transmit(char * buf, char txum, char rxnum)
{
    If(txnum > 32) txnum = 32;
    InitLength( rxnum, txmun );
    TXEN = HIGH;

    //写入发送地址及数据
    CSN = LOW;
    SpiReadWrite(WTP);
    For(i = 0; i < txnum; i++)
    SpiReadWrite(puf[i]);
    CSN = HIGH;

    //产生脉冲,发送数据
    TRX_CE = HIGH;
    Delay();
    TRX_CE = LOW;

    //等待发送完毕
    While(DR == HIGH);
}
```

2.2.4 接受数据流程

接受数据应先给出时序图,如图 6所示,再解释流程。

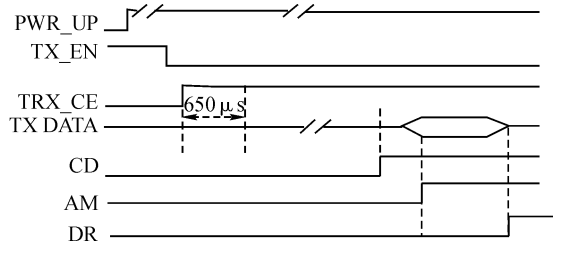

图 6 接收数据时序图

(1) 主 MCU将 TX_EN置高、TRX_CE置低,过 650μs后进入接受模式。

(2) nRF905监控频道使用状况,如果发现频道被占用,则将 CD置高,可以利用该特性采取一些冲突避免检测机制,发送数据前如果检测到 CD信号,则可以随机延迟一段时间再发送数据,该特性可以有效地避免数据冲突。

(3) 当接收到的数据发送地址和自己地址匹配时,则 AM置高,通知该数据是发给自己的。

(4) 对数据的 CRC进行校验,如果正确,则去除包头和 CRC段,将数据保存在接受数据寄存器 RX_PAYLOAD,同时 DR信号置高,通知主 MCU读取数据。

(5)主 MCU将 TRX_CE置低,进入 standby(省电)模式再通过 SPI口将数据读出来,当数据都读完后,nRF905将 AM和 DR重新置低,为下次接受数据做准备。

接受数据流程图如图 7所示。

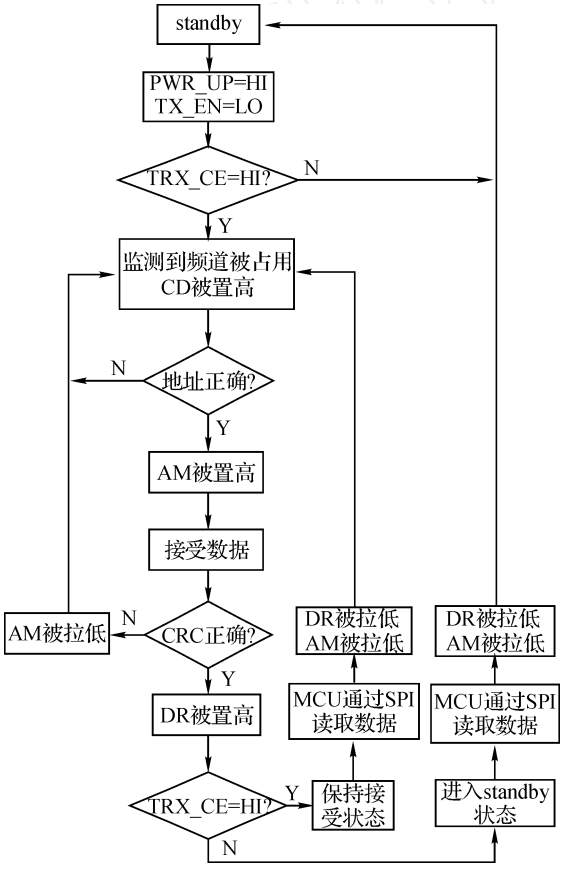

图 7 接受数据流程图

部分代码实现:

```
//读接受数据寄存器命令字
#define RRP 0x24

//接受数据函数,一次最多能接受 32B
//buf存放接受数据缓存
//rxbuf接受数据字节数
//txbuf发送确认字节数
void Receive(char * buf, char rxnum, char txnum)
{
    int i=0;
    If(rxnum>32) rxnm=32;
    InitLength(rxnum, txmun);
    TXEN = LOW;
    TRX_CE = HIGH;
    While(DR==LOW);
    TRX_CE = LOW;

    //读取数据
    CSN = LOW;
    SpiReadWrite(RRP);
    While(DR==HIGH)
    {
      Buf(i++)=SpiReadWrite(puf[i]);
      if(i==rxnum) break;
    }
    CSN = HIGH;
}
```

如果需要将驱动移植到其它平台如 ARM上,只要将接口的重新定义、SpiReadWrite()函数做相应调整,移植起来非常简单。

3 结 论

该无线通信模块,一方面留有与 PC机通信的 232接口,可以用来检验 MAC协议的有效性;另一方面留有与采集仪器仪表数据的 485接口,可以用于无线抄表系统。

利用此通信模块,同时物理层采用 802.11MAC算法,网络层采用鱼眼路由算法,可以很好的组建一个小型的 Ad Hoc网络,该网络中各节点定时发送广播,更新自己路由信息表,另一方面新节点加入时也发送广播通知,从而实现了无中心,自组织和动态建网等特性。

参考文献:

[1] Single chip 433/868/915 MHz Transceiver nRF905 datasheet, 2004.7.

[2] 胡大可. FLASH型超低功耗 16位单片机[M]. 北京:北京航空航天大学出版社, 2002.