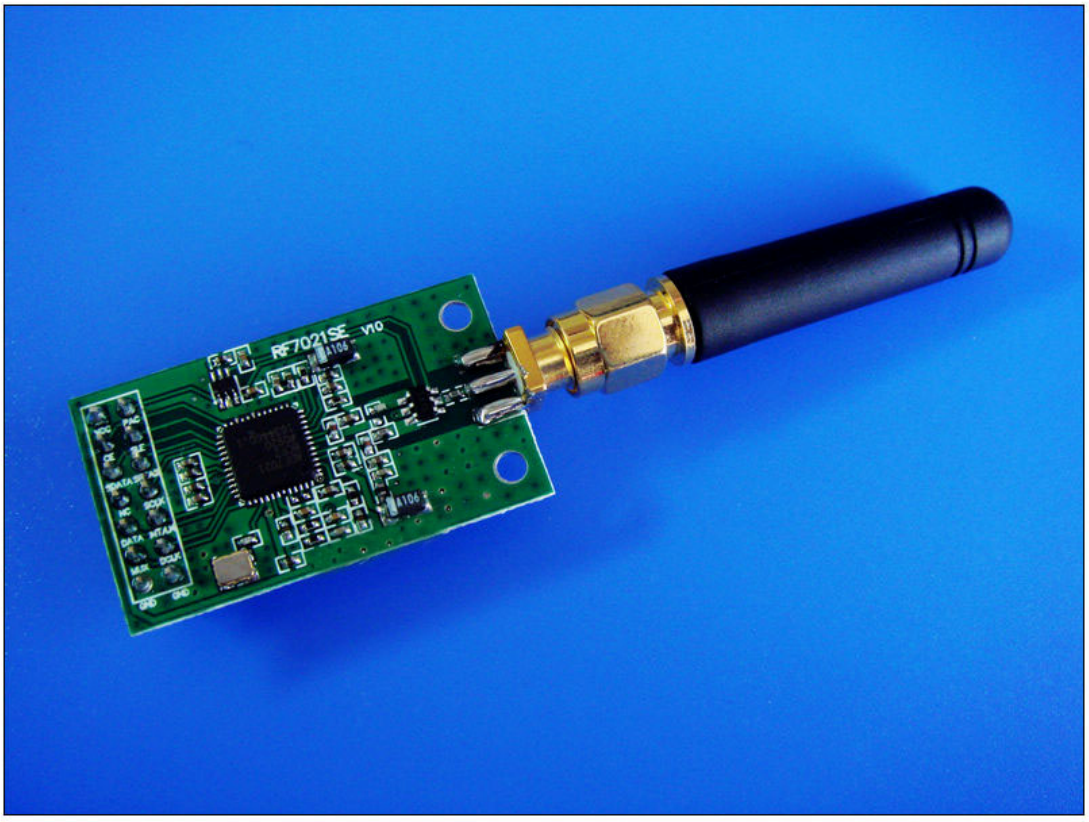
ADF7021 无线模块

用户手册



ADF7021 模块（尺寸：37mm X 21mm　板厚：1mm ，SMA 天线接口）

目录

产品介绍

ADF7021 是 ADI（美国模拟器件）公司推出的一款低功耗、高集成度 2FSK/3FSK/4FSK 窄带收发器芯片，在 1 kbps (868/915MHz) 工作条件下达到-123 dBm 最佳接收器灵敏度，具有内置收发器(TRX)开关、压控振荡器(VCO)谐振回路以及射频和中频(RF/IF)滤波器、全自动的自动频率控制(AFC)和内置模数转换器(ADC)。它同时提供高斯滤波器和升余弦滤波器选项以便为窄带应用提高频谱效率。发射部分包含一个压控振荡器 (VCO)和一个输出分辨率小于 1 ppm、低噪声、小数 N 分频的锁相环(PLL)。这种频率捷变 PLL 允许 ADF7021 用于跳频扩频(FHSS)系统。VCO 工作于两倍基频以减少杂散发射和频率牵引问题。发射器输出功率具有自动功率斜波控制以防止频谱溅射，有助于满足管理标准。当它不使用时可以进入待机状态以降低功耗。由于 ADF7021 具有高接收灵敏度，因此非常适合高可靠性应用，例如无线自动抄表(AMR)、工业控制与自动化和远程安全系统。

基本特性

(1) 频率范围为 80Mhz - 650Mhz 频段

(2) 2.3-3.6V 供电电压，适合电池供电

(3) 可编程输出功率,从-16dBm 到+13dBm,步进 0.3dBm

(4) 功耗 ：发送模式（10dBm） 29.2mA;接收模式 26.4mA

(5) 接收灵敏度非常高

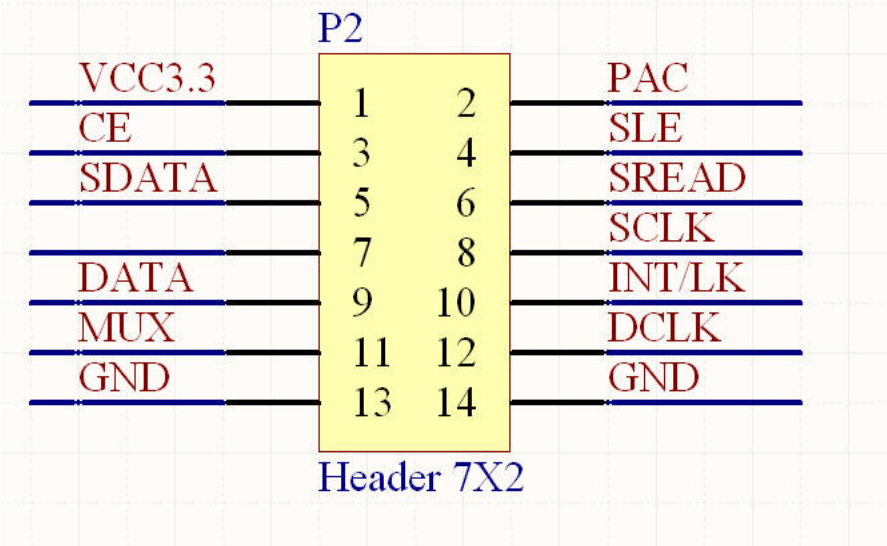
-125dBm (1kps,2FSK)

-113dBm (25kps,2FSK)

(6) 标准 2.54 DIP 间距接口，便于嵌入式应用

(7) 传输速率支持 0.05kbps - 32.8 kbps，低速，通信时能量集中，距离远是最大优势

模块接口说明



引脚功能

管脚	名称	管脚功能	说明
1	VCC3.3	电源	3.3V电源
2	PAC	数字输入	收发电路切换，置高则切换到发送
3	CE	数字输入	芯片片选，高有效
4	SLE	数字输入	四线串行口，使能信号
5	SDATA	数字输入	四线串行口，输入数据线
6	SREAD	数据输出	四线串行口，输出数据线
7	NC	无	悬空
8	SCLK	数字输入	四线串行口，时钟信号
9	DATA	数字输入输出	数据收发，数据线
10	INT/LK	数字输出	前导接收指示

11	MUX	数字输出	未启用
12	DCLK	数字输出	数据收发，时钟线
13	GND	地	接地
14	GND	地	接地

备注

1.VCC 引脚的电压范围为2.3-3.6V 之间，不能在这个区间之外，如超过 3.6V 将会烧毁模块。推荐电压 3.3V 左右；

2.硬件没有集成SPI功能的单片机也可以控制本模块，用普通单片IO口模拟 SPI 时序进行读写操作即可；

3.模块接口采用标准2.54mmDIP插针，13 脚、14 脚为接地脚，需要和系统电路的逻辑地连接起来；

4.与 51 系列单片机 P0 口连接时候，需要加 10K 的上拉电阻，与其余口连接不需要。其他系列的5V单片机，如AVR、PIC，请参考该系列单片机 IO 口输出电流大小，如果超过 10mA，需要串联2-5K电阻分压，否则容易烧毁模块！如果是 3.3V 的MCU，可以直接和模块的IO口线连接。

ADF7021 模块电气参数

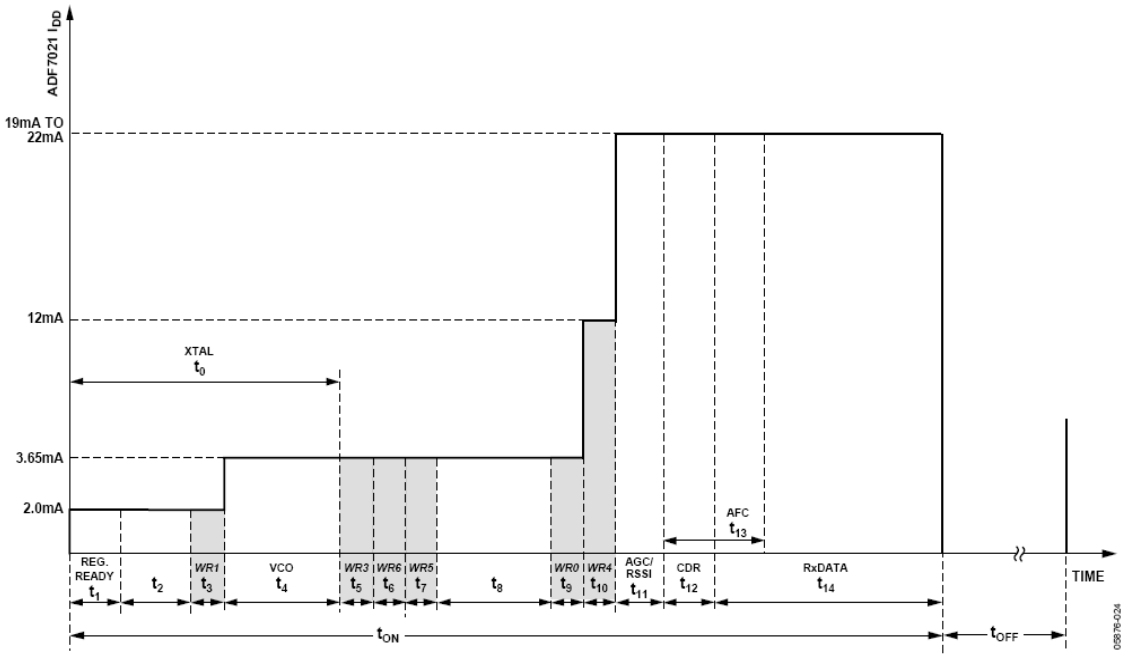
ADF7021模块性能参考数据

参数	数值	单位
工作电压	2.3-3.6	V
最大发射功率	13	dBm
最大数据传输率	32.8	kbps
输出功率为+10 dBm 时工作电流	29.2	mA
接收模式时工作电流	26.4	mA
温度范围	-40 to +85	℃
典型灵敏度	-122	dBm

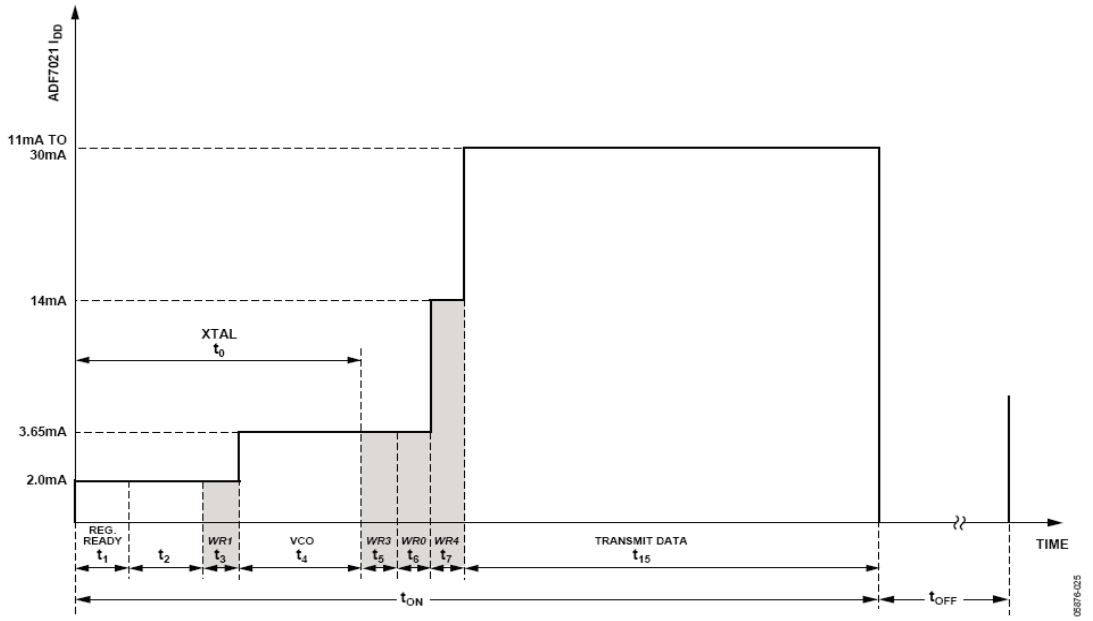
工作流程

模块采用 ADI 的 ADF7021 芯片开发而成。ADF7021 单片无线收发器工作在 80Mhz-650Mhz 频段，集成了位同步器。通过四线 SPI 串口，可对芯片进行配置。设置处在接收模式，当接收到正确的前导同步码时，INT/LK 引脚产生中断信号。同时，DCLK 引脚上产生同步时钟信号，解调的数据通过 DATA 输出。设置处在发送模式时，DCLK 引脚上产生同步时钟信号，要发送的数据通过 DATA 输入。ADF7021 本身硬件不含 CRC 校验，如需数据包编址和 CRC 校验功能，可通过软件协议实现。

为便于用户开发，我们提供配套评估套件，为产品开发保驾护航，使无线应用开发大大加速，并避免不必要的误区。



数据发送时序图



数据接收时序图

ADF7021 编程指南

ADF7021 模块无需掌握任何专业无线或高频方面的理论，读者只需要具备一定的 C 语言程序基础即可。本文档没有涉及到的问题，读者可以参考 ADF7021 官方手册或向我们寻求技术支持。

ADF7021 寄存器配置

```
typedef struct  _ADF7021_RF_SETTINGS_
{
    uint32 N_REG_RX;
    uint32 TRANSMIT_MODULATION_REG_RX;
    uint32 N_REG_TX;
    uint32 TRANSMIT_MODULATION_REG_TX;
    uint32 VCO_OSCILLATOR_REG;
```

```
    uint32 TRANSMIT_RECEIVER_CLOCK_REG;
    uint32 DEMODULATOR_SETUP_REG;
    uint32 IF_FILTER_SETUP_REG;
    uint32 IF_FINE_CAL_SETUP_REG;
    uint32 AGC_REG;
    uint32 AFC_REG;
    uint32 SYNC_BYTE_REG;
    uint32 SWD_THRESHOLD_SETUP_REG;
    uint32 _3FSK_4FSK_DEMO_REG;
    }ADF7021_RF_SETTINGS;
```

SPI 写寄存器操作

```
void ADF7021_WriteReg(uint32 value32)
{
    uint8 i ;
    SLE_OFF();
    SCLK_OFF();
    for (i = 0;i < 32;i++)
    {
        if (value32&0x80000000) //总是发送最高位
        {
            SDATA_ON();
        }
        else
        {
            SDATA_OFF();
        }
        SCLK_ON();
        value32 = value32<<1;
        SCLK_OFF();
    }
    SLE_ON();
    SDATA_OFF();
    SLE_OFF();

}
```

SPI 读寄存器操作

```
uint32 ADF7021_ReadReg(uint32 readback)
{
    uint8 i ;
    uint32 rxdata;
    SLE_OFF();
    SCLK_OFF();
    for (i = 0;i < 32;i++)
    {
        if (readback&0x80000000) //总是发送最高位
        {
            SDATA_ON();
        }
        else
        {
            SDATA_OFF();
        }
        SCLK_ON();
        readback = readback<<1;
        SCLK_OFF();
    }
    SLE_ON();
    SDATA_OFF();
    SCLK_ON(); //输出第一个 bit，废弃
    rxdata = 0;
    SCLK_OFF();
    for (i = 0;i < 16;i++)
    {
        SCLK_ON();
        rxdata = rxdata<<1;
        SCLK_OFF();
        if (SREAD_IN)
        {
            rxdata |= 0x00000001;
        }
     }
    SCLK_ON();
    SCLK_OFF();
    SLE_OFF();
    return rxdata;
}
```

ADF7021 初始化函数

```
void ADF7021_Init(void)
{
    CE_ON()  ;
    PAC_OFF();
    delay_ms(1);
    ADF7021_WriteReg(ADF7021_Config.VCO_OSCILLATOR_REG);
    delay_ms(1);
    ADF7021_WriteReg(ADF7021_Config.TRANSMIT_RECEIVER_CLOCK_REG);
    ADF7021_WriteReg(ADF7021_Config.IF_FINE_CAL_SETUP_REG);
    ADF7021_WriteReg(ADF7021_Config.IF_FILTER_SETUP_REG);
    delay_ms(6);
    ADF7021_WriteReg(ADF7021_Config.SYNC_BYTE_REG);
    ADF7021_WriteReg(ADF7021_Config.SWD_THRESHOLD_SETUP_REG);
    ADF7021_WriteReg(ADF7021_Config._3FSK_4FSK_DEMO_REG);
    ADF7021_WriteReg(ADF7021_Config.TRANSMIT_MODULATION_REG_RX);
   ADF7021_WriteReg(ADF7021_Config.N_REG_RX);
    delay_ms(1);
    ADF7021_WriteReg(ADF7021_Config.DEMODULATOR_SETUP_REG);
    ADF7021_WriteReg(ADF7021_Config.AGC_REG);
    ADF7021_WriteReg(ADF7021_Config.AFC_REG);
    delay_ms(1);
    ADF7021_TxRxBuf[0] = 0xAA;
    ADF7021_TxRxBuf[1] = 0xAA;
    ADF7021_TxRxBuf[2] = 0xAA;
    ADF7021_TxRxBuf[3] = 0xAA;
    ADF7021_TxRxBuf[4] = 0xAA;
    ADF7021_TxRxBuf[5] = 0x12;
    ADF7021_TxRxBuf[6] = 0x34;
    ADF7021_TxRxBuf[7] = 0x56;
    RX_Flag = FALSE;
    ADF7021_Status = IDLE_MODE;
}
```

ADF7021 发送数据流程

数据包发送函数:完成数据包前导、地址、CRC 的计算，最后开启发送中断，并设置模块进入发送模式

```
void ADF7021_TxPacket(void)
{
```

```
    uint16 CRC;
    ADF7021_TxRxBuf[ADF7021_Preamble_Len] = ADF7021_TxAddr[0];
    ADF7021_TxRxBuf[ADF7021_Preamble_Len+1]= ADF7021_TxAddr[1];
    CRC                                                                    =
CRC16(ADF7021_TxRxBuf+ADF7021_Preamble_Len,ADF7021_Address_Len+ADF702
1_TxRxBuf_Len);
    ADF7021_TxRxBuf[ADF7021_Preamble_Len+ADF7021_Address_Len+ADF7021_
TxRxBuf_Len] = CRC&0x00FF;
    ADF7021_TxRxBuf[ADF7021_Preamble_Len+ADF7021_Address_Len+ADF7021_
TxRxBuf_Len+1] = CRC>>8;
    TxRxdata  = ADF7021_TxRxBuf[0]; ;
    TxRxdata_i = 0;
    TxRxdata_index = 0;
    ADF7021_Status = TX_MODE;
    DATA_DDR_OUT();
    INT_MASK_OFF();
    ADF7021_WriteReg(ADF7021_Config.TRANSMIT_MODULATION_REG_TX);
    ADF7021_WriteReg(ADF7021_Config.N_REG_TX);
    delay_ms(1);
    PAC_ON();
    DCLK_MASK_ON();
    PCIE2_FLAG_CLEAR() ;
    PCIE2_ON();
}
```

ADF7021 接收数据流程

开启接收中断，并设置模块进入接收模式

```
void ADF7021_SetRxMode(void)
{
    ADF7021_WriteReg(ADF7021_Config.TRANSMIT_MODULATION_REG_RX);
     ADF7021_WriteReg(ADF7021_Config.N_REG_RX);
    delay_ms(1);
    PAC_OFF();
    TxRxdata = 0 ;
    TxRxdata_i = 0;
    TxRxdata_index = ADF7021_Preamble_Len;
    ADF7021_Status = RX_MODE;
    DCLK_MASK_OFF();
    INT_MASK_ON();
    PCIE2_FLAG_CLEAR() ;
    PCIE2_ON();
}
```

数据包接收判断函数:根据标志位判断是否接收到数据，然后计算 CRC 值和地址是否匹配。

```
uint8 ADF7021_RxPacket(void)
{
    if(RX_Flag)
    {
        uint16 CRC;
        CRC=
CRC16(ADF7021_TxRxBuf+ADF7021_Preamble_Len,ADF7021_Address_Len+ADF702
1_TxRxBuf_Len);

    if(  ADF7021_TxRxBuf[ADF7021_Preamble_Len+ADF7021_Address_Len+ADF7
021_TxRxBuf_Len] == (CRC&0x00FF))
        {

    if(ADF7021_TxRxBuf[ADF7021_Preamble_Len+ADF7021_Address_Len+ADF70
21_TxRxBuf_Len+1] == (CRC>>8))
            {
                if((ADF7021_RxAddr[0]                                    ==
ADF7021_TxRxBuf[ADF7021_Preamble_Len])&&     (ADF7021_RxAddr[1]     ==
ADF7021_TxRxBuf[ADF7021_Preamble_Len+1]))
                {
                    RX_Flag = FALSE;
                    return TRUE;
                }
            }
        }
        RX_Flag = FALSE;
    }
    return FALSE;
}
```

中断方式数据处理

接收和发送中断:数据收发最终在中断里完成

```
#pragma vector=PCINT2_vect__interrupt void PCINT2_ISR(void)
{
    if(INT_RISING_FLAG)
    {
```

```
        DATA_DDR_IN(); INT_MASK_OFF();DCLK_MASK_ON();
        PCIE2_FLAG_CLEAR() ;
    }
    else if(DCLK_RISING_FLAG)
    {
            switch(ADF7021_Status)
            {
                case RX_MODE:
                    TxRxdata <<= 1;
                    TxRxdata |= DATA_IN;
                    TxRxdata_i++;
                        if(8 == TxRxdata_i)
                    {
                        if(TxRxdata_index                                    <
(ADF7021_Preamble_Len+ADF7021_Address_Len+ADF7021_TxRxBuf_Len+ADF7021
_CRC16_Len))
                        {
                            ADF7021_TxRxBuf[TxRxdata_index++] = TxRxdata;
                        }
                        else
                        {
                            RX_Flag = TRUE;
                            ADF7021_Status = IDLE_MODE;
                            DCLK_MASK_OFF();PCIE2_OFF();
                        }
                        TxRxdata_i = 0;
                        }
                    break;
                default:
                    break;
            }
         }
    else if(DCLK_FALLING_FLAG)
    {
```

```
            switch(ADF7021_Status)
            {
                case TX_MODE:
                    if(TxRxdata&0x80)
                    {
                        DATA_ON();
                    }
                    else
                    {
                        DATA_OFF();
                    }
                    TxRxdata <<= 1;
                    TxRxdata_i++;
                        if(8 == TxRxdata_i)
                    {
                        if(TxRxdata_index                                    <
(ADF7021_Preamble_Len+ADF7021_Address_Len+ADF7021_TxRxBuf_Len+ADF7021
_CRC16_Len+1))//多发一字节，防止最后一位发送错误
                        {
                            TxRxdata = ADF7021_TxRxBuf[TxRxdata_index++];
                        }
                        else
                        {
                            ADF7021_Status = IDLE_MODE;
                            DCLK_MASK_OFF();PCIE2_OFF();
                        }
                        TxRxdata_i = 0;
                    }
                    break;
                default:
                    break;
            }
    }
}
```

无线应用注意事项

(1) 无线模块的 VCC 电压范围为 1.8V-3.6V 之间，不能在这个区间之外，超过 3.6V 将会烧毁模块。推荐电压 3.3V 左右。

(2) 除电源 VCC 和接地端，其余脚都可以直接和普通的 51 单片机 IO 口直接相连，无需电平转换。当然对 3V 左右的单片机更加适用了。

(3) 硬件上面没有 SPI 的单片机也可以控制本模块，用普通单片机 IO 口模拟 SPI 不需要单片机真正的串口介入，只需要普通的单片机 IO 口就可以了，当然用串口也可以了。模块按照接口提示和母板的逻辑地连接起来

(4) 标准 DIP 插针，如需要其他封装接口，或其他形式的接口，可联系我们定做。

(5) 任何单片机都可实现对无线模块的数据收发控制，并可根据我们提供的程序，然后结合自己擅长的单片机型号进行移植;

(6) 频道的间隔的说明：实际要想 2 个模块同时发射不相互干扰，两者频道间隔应该至少相差 1MHZ，这在组网时必须注意,否则同频比干扰。

(7) 实际用户可能会应用其他自己熟悉的单片机做为主控芯片，所以，建议大家在移植时注意以下 4 点：

A:确保 IO 是输入输出方式，且必须设置成数字 IO;

B:注意与使用的 IO 相关的寄存器设置，尤其是带外部中断、

带 AD 功能的 IO，相关寄存器一定要设置好;

C:调试时先写配置字，然后控制数据收发

D:注意工作模式切换时间

我们的承诺

最后，欢迎您使用我们的产品，在应用中有技术问题请及时向我们联系，我们会予以技术知道，同时运输中出现产品问题我们会全面责任并予以更换。

愿与您一起走向成功