UTC-903 开发指南（V3.1）

一、模块介绍

UTC-903（电路板尺寸：38mm X 22mm 板厚：1mm）

Outstanding Features:

配 4.5cm 棒状天线，开阔地通信距离可以达到 600 米；配我们公司专门的高增益吸盘天线，开阔地距离可以达到 800 米！

特点：

(1) 433Mhz 开放 ISM 频段免许可证使用

(2) 最高工作速率 50kbps，高效 GFSK 调制，抗干扰能力强，特别适合工业控制场合

(3) 内置硬件 CRC 检错和点对多点通信地址控制

(4) 低功耗 3-3.6V 工作，待机模式下状态仅为 2.5uA

(5) 收发模式切换时间 < **650us**

(6) 模块可软件设地址，只有收到本机地址时才会输出数据（提供中断指示），可直接连接各种单片机使用，软件编程非常方便

(7) TX Mode: 在+10dBm 情况下，电流为 40mA; RX Mode: 14mA

(8) 标准 DIP 间距接口，7X2pin，硬件管脚兼容 RF905SE,无需修改底板，只需要升级软件即可

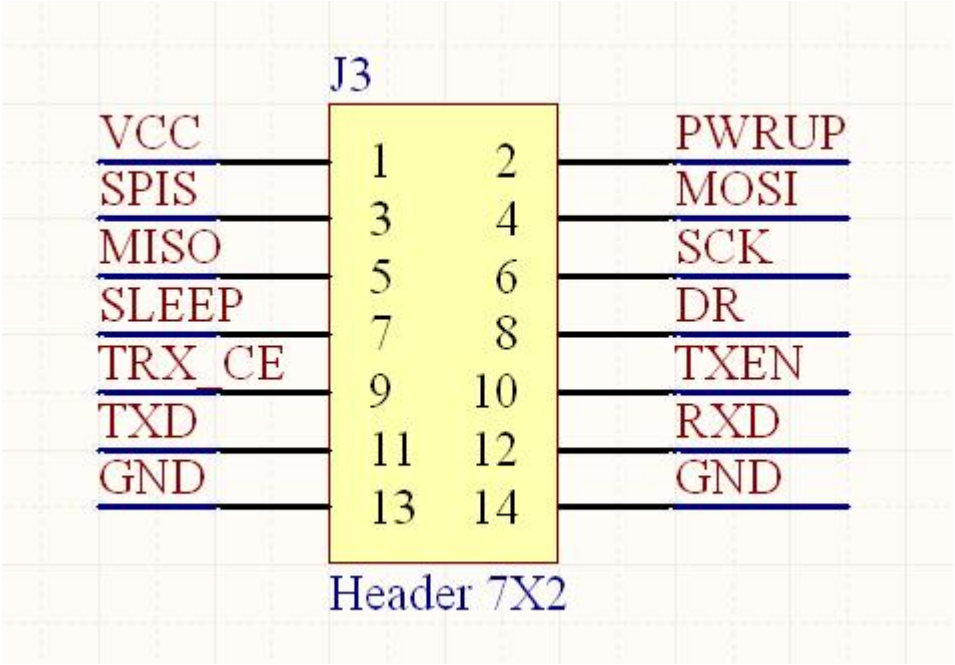
(9)RFModule-Quick-DEV 快速开发系统，含开发板

(10)增加了电源切断模式，可以实现硬件冷启动功能!

(11)SPI 接口--功能强大、编程简单，与 RF905SE 编程接口类似。

(12) 增加了 RSSI 功能，通过 SPI 接口可以获取当前接收到的信号强度(0-254)，可以供当前设备做出决策，比如低于某个数值可以报警，提示用户当前信号质量比较低等。

二、接口电路管脚说明



管脚	名称	管脚功能	说明
1	VCC	电源	电源+3¯3.6V DC
2	PWRUP	数字输入	PWRUP=1 模块上电; PWRUP=0 切断模块电源
3	SPIS	SPI 使能	SPI 使能
4	MOSI	SPI 接口	SPI 输入
5	MISO	SPI 接口	SPI 输出
6	SCK	SPI 时钟	SPI 时钟
7	SLEEP	数字输入	SLEEP=0模块进入睡眠状态
8	DR	数字输出	接收或发射数据完成
9	TRX_CE	数字输入	使能芯片发射或接收
10	TXEN	数字输入	TX_EN= 1 发送模式 TX_EN= 0 接收模式
11	TXD	串口	该功能当前版本无效
12	RXD	串口	该功能当前版本无效
13	GND	地	接地
14	GND	地	接地

说明:

(1) VCC 脚接电压范围为 3V¯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 5V 单片机 IO 口相连，需要加 1K-2K 的限流电阻。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 SPI 模块介入，只需添加代码模拟 SPI 时序即可。

(4) 13 脚、14 脚为接地脚,需要和母板的逻辑地连接起来

(5) 排针间距为 100mil,标准 DIP 插针，如果需要其他封装接口，比如密脚插针，或者其他形式的接口，可以联系我们定做。

(6) 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。

(7) 其他系列的单片机，如果是 5V 的，请参考该系列单片机 IO 口输出电流大小，如果超过 10mA，需要串联电阻分压，否则容易烧毁模块！如果是 3.3V 的，可以直接和 UTC-903 模块的 IO 口线连接。

三、模块引脚和电气参数说明

UTC-903模块性能参考数据

参数	数值	单位
最低工作电压	3.0	V
最大发射功率	10	dBm
最大数据传输率曼切斯特编码	50	kbps
输出功率为+10 dBm 时工作电流	40	mA
接收模式时工作电流	14.5	mA
温度范围	-40 to +85	
典型灵敏度	-100	dBm
POWERDOWN 模式时工作电流	2.5	uA

四、 UTC-903 工作方式

UTC-903 一共有四种工作模式，RX/TX 模式和节能睡眠模式及配置模式。

活动模式

ShockBurst RX

ShockBurst TX

节电模式

节能睡眠模式

关机模式　　PWRUP=0 关机模式

UTC-903工作模式由SLEEP、TRX_CE、TX_EN、PWRUP 的设置来设定。

SLEEP	TRX_CE	TX_EN	工作模式
0	X	X	节能睡眠模式
1	0	0	配置模式
1	1	0	ShockBurst RX
1	1	1	ShockBurst TX

注意：为了提高性能，建议不要频繁改变 SLEEP 引脚电平。

另外，UTC-903 可以通过 PWRUP 控制电源的通断。PWRUP=1 表示对 UTC-903 供电，PWRUP=0 表示对 UTC-903 处于断电状态。在用户无需无线数据传输的时候，PWRUP=0 可以最大限度的降低功耗。

4.1 ShockBurst 模式

ShockBurstTM 收发模式下，使用片内的先入先出堆栈区，数据低速从微控制器送入，但高速发射，这样可以尽量节能，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。与射频协议相关的所有高速信号处理都在片内进行，这种做法有三大好处：尽量节能;低的系统费用(低速微处理器也能进行高速射频发射);数据在空中停留时间短，抗干扰性高。ShockBurstTM 技术同时也减小了整个系统的平均工作电流。

在 ShockBurstTM 收发模式下， UTC-903 自动处理字头和 CRC 校验码。在接收数据时，自动把字头和 CRC 校验码移去。在发送数据时，自动加上字头和 CRC 校验码，当发送过程完成后，DR 引脚通知微处理器数据发射完毕。

4.1.1 ShockBurst TX 发送流程

典型的 UTC-903 发送流程分以下几步:

A. 当微控制器有数据要发送时，首先置 TRX_CE 和 TX_EN 为高电平，然后通过 SPI 接口，按时序把要发送的数据送传给 UTC-903;

B. 接收完 16（固定为 16 字节）字节后，激发 UTC-903 的 ShockBurstTM 发送模式;

C. UTC-903 的 ShockBurstTM 发送:

(1) 射频寄存器自动开启;

(2) 数据打包(加字头和 CRC 校验码);

(3) 发送数据包;

(4) 当数据发送完成，数据准备好引脚被置高;

4.1.2 ShockBurst RX 接收流程

接收流程

A. 当 TRX_CE 为高、TX_EN 为低时，UTC-903 进入 ShockBurstTM 接收模式;

B. 650us 后，UTC-903 不断监测，等待接收数据;

C. 当一个正确的数据包接收完毕(相匹配的地址,CRC 正确)，UTC-903 自动移去字头、地址和 CRC 校验位，然后把 DR 引脚置高 ;

D. 微控制器通过 SPI 口，以一定的速率把数据移到微控制器内（此时需要保持 TRX_CE 为高电平、TX_EN 为低 电平);

E. 当所有的数据接收完毕，UTC-903 把 DR 引脚置低;

F. 之后 UTC-903 自动进入 Rx 模式，接收到数据通过 DR 脚中断外部 mcu(外部 mcu 检测到中断，读取数据的时候，必须先置 TRX_CE 为高、TX_EN 为低时，然后通过 spi 读取数据，读取的数据为有效数据和一字节的 RSSI 值);。

当正在通过 SPI 读取一个数据包时，TRX_CE 或 TX_EN 引脚的状态发生改变，数据包数据将出错。

4.1.3 节能模式

UTC-903 的节能模式包括关机模式和节能模式。

在关机模式，UTC-903 的工作电流最小，一般为 2.5uA。进入关机模式后，UTC-903 保持配置寄存器的内容，但不会接收或发送任何数据，也不保存 TX ADSRESS 和 RX ADDRESS，上电后都将恢复 为默认

的{0x54,0x75,0xC5,0x2A}。 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，UTC-903 寄存器内容和 TX ADSRESS，RX ADDRESS 都保存。

五、配置 UTC-903 模块

所有配置字都是通过 SPI 接口送给 UTC-903。SIP 接口的工作方式可通过 SPI 指令进行设置。

5.1 SPI 指令设置

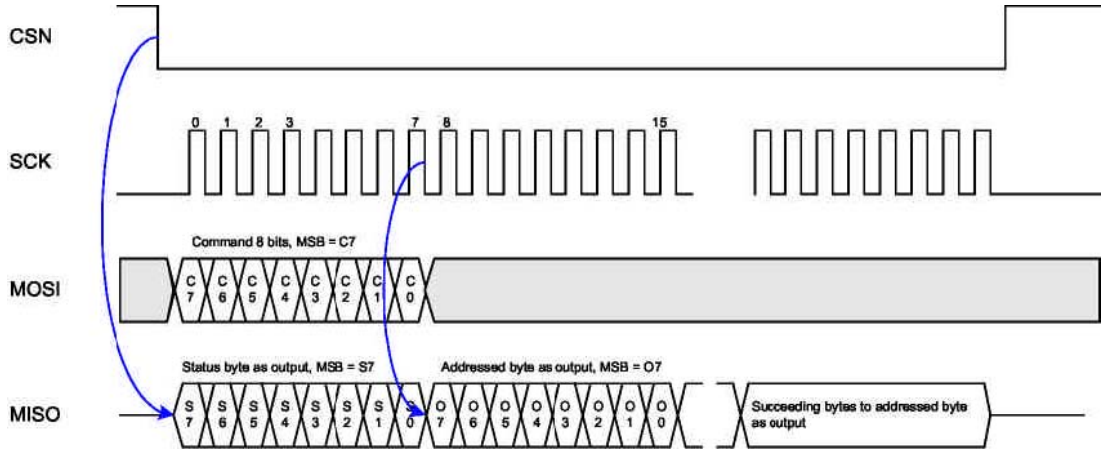
当CSN 为低时，SPI接口开始等待一条指令。任何一条新指令均由CSN 的由高到低的转换开始。用于SPI 接口的有用命令见下表:

SPI 串行接口指令设置

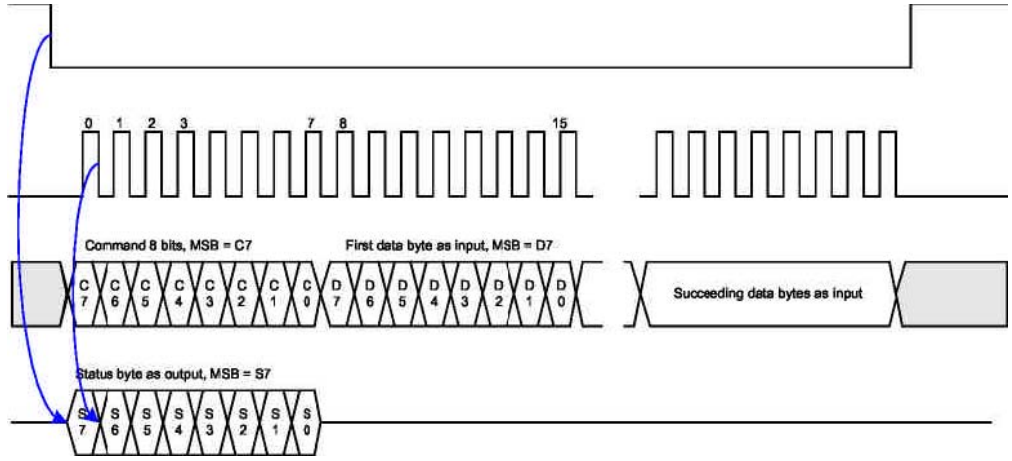
SPI 串行接口指令		
指令名称	指令格式	操作
W_RX_ADDRES S (WTA)	00010000	写RX 地址4 字节写操作全部从字节0 开始
R_RX_ADDRES S (WTA)	00010001	读RX 地址4 字节写操作全部从字节0 开始（暂时没有提供）
W_TX_PAYLOA D (WTP)	00100000	写TX 有效数据（默认32字节长，可配置为1-64字节） 字节写操作全部从字节0 开始
W_TX_ADDRES S (WTA)	00100010	写TX 地址4 字节写操作全部从字节0 开始
R_TX_ADDRES S (RTA)	00100011	读TX 地址1-4 字节读操作全部从字节0 开始（暂时没有提供）
R_RX_PAYLOA D (RRP)	00100100	读RX 有效数据（默认32字节长，可配置为1-64字节） 字节读操作全部从字节0 开始

5.2 SPI 时序

SPI 读操作



SPI 写操作



六、 UTC-903 编程指南

使用 UTC-903 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。

UTC-903 模块配置简洁，用户只需配置 TX ADDRESS, RX ADDRESS（默认发送接收地址一致）,数据包长度（固定 16 字节），然后控制 SLEEP,TRX_CE,TX_EN 引脚，通过 spi 进行数据的发送,接收，RSSI 读取；进入低功耗模式；电源通断等。

范例程序中的相关代码段（1）:（代码使用 mega48 mcu）

```
//<SPI 命令>
#define WRA       0x10
#define RRA       0x11
#define WTP       0x20
#define RTP       0x21
#define WTA       0x22
#define RTA       0x23
#define RRP       0x24
```

```
// IO 线模拟 spi 读写
unsigned char nRF903WriteRead8Bit(unsigned char byte)
```

```
{
  unsigned char temp;
  nRF903_SCK = 0;
  for(temp=8;temp>0;temp--)
  {
    nRF903_MOSI = (byte&0x80)>>7;
    byte = byte<<1;
    nRF903_SCK = 1;
    _NOP_ (4); // 延时函数
    byte = byte|(nRF903_MISO_PIN&0x01);
    nRF903_SCK = 0;
     _NOP_(4);
  }
  return byte;
}



// 更改发送地址
void SetTxAddress(unsigned char* buf)
{
  unsigned char i=0;
```

```
  nRF903_TRX_CE=0;
  nRF903_TX_EN=0;
  nRF903_SS=0;
  nRF903WriteRead8Bit(WTA);
  for(i=0;i<4;i++)
  {
       nRF903WriteRead8Bit(buf[i]);
  }
  nRF903_SS=1;
  _NOP_ms(1);
}

// 更改接收地址
void SetRxAddress(unsigned char* buf)
{
  unsigned char i=0;
  nRF903_TRX_CE=0;
  nRF903_TX_EN=0;
  nRF903_SS=0;
  nRF903WriteRead8Bit(WRA);
  for(i=0;i<4;i++)
  {
```

```
      nRF903WriteRead8Bit(buf[i]);
  }
  nRF903_SS=1;
  _NOP_ms(1);
}
// 发送一个数据包
void SendPacket(unsigned char* buf)
{
    unsigned char i=0;
    nRF903_TRX_CE=1;
    nRF903_TX_EN=1;
    nRF903_SS=0;
    nRF903WriteRead8Bit(WTP);
    for(i=0;i<16;i++) //数据包长度为固定 16 字节
    {
      nRF903WriteRead8Bit(buf[i]);
    }
    nRF903_SS=1;
    while(nRF903_DR_PIN==0);
    _NOP_nop(1);
}
// 接收一个数据包 (此函数最好在中断函数中调用，中断设置为上
```

升沿触发)，注意，RSSI 的值在接收数据包的最后一个字节中，比如数据包长度为 32 字节，那么[0-31]字节为 32 字节的数据，第 32 字节为 RSSI 值，所以每次读取数据包的时候，接收缓冲区要 32+1 字节（数据包长度加一个字节 RSSI ）。

```
unsigned char ReceivePacket(unsigned char* buf)
{
    unsigned char i=0;
    if(nRF903_DR_PIN==0)
      return 0;
    nRF903_TRX_CE=1;
    nRF903_TX_EN=0;
    nRF903_SS=0;
    nRF903WriteRead8Bit(RRP);
    for(i=0;i<16;i++) //数据包长度为固定 16 字节
    {
      buf[i]=nRF903WriteRead8Bit(RRP);
    }
     Buf[16] = nRF903WriteRead8Bit(RRP); // RSSI 值（每次读取数据包的时候必须一起读取 rssi 值，如果 RSSI 值为 255，代码本次 RSSI 值无效）
    nRF903_SS=1;
    while(nRF903_DR_PIN ==1); // 确认读取完成
```

```
    return 1;
}

// 进入睡眠模式/离开睡眠模式 s=1 进入睡眠模式；否则离开睡眠模式
void Enter_Sleep(unsigned char s)
{
  if(s)
  {
    nRF903_SLEEP=1;//高到低
    nRF903_SLEEP=0;
  }
  else
  {
    nRF903_SLEEP=0; // 低到高
    nRF903_SLEEP=1;
  }
  _NOP_us(15);
}
```

范例程序中的相关代码段（2）:（代码使用 AT89S52 mcu）

```
// def.h
#ifndef _DEF_H_
```

```
#define _DEF_H_

#include <at89x52.h>
#include <intrins.h>
sbit   nRF903_PWR_UP =P1^0;
sbit   nRF903_SS     =P3^4;
sbit   nRF903_MOSI    =P1^1;
sbit    nRF903_MISO   =P0^5;
sbit   nRF903_TRX_CE   =P1^6;
sbit   nRF903_TX_EN   =P1^5;
sbit   nRF903_SLEEP   =P1^4;
sbit   nRF903_DR      =P3^2;
sbit   nRF903_SCK     =P1^2;
sbit    KEY1          =P3^6 ;
sbit    KEY2          =P3^7;
sbit   led1             =P2^1;
sbit   led0=P2^0;
sbit   led2=P2^2;
sbit   led3=P2^3;

#endif //_DEF_H_




// main.c

#include "def.h"

//<SPI Command>
#define WRA      0x10
#define RRA      0x11
#define WTP      0x20
#define RTP      0x21
#define WTA      0x22
#define RTA      0x23
#define RRP      0x24


void Delay(unsigned char n)
{
    unsigned char i;
    while(n--)
    for(i=0;i<80;i++);
```

```
}


// spi write and read
unsigned char nRF903WriteRead8Bit(unsigned char byte)
{
  unsigned char temp;
  nRF903_SCK = 0;
  for(temp=8;temp>0;temp--)
  {
    nRF903_MOSI = (byte&0x80)>>7;
    byte = byte<<1;
    nRF903_SCK = 1;
    _nop_();
    byte = byte|(nRF903_MISO&0x01);
    nRF903_SCK = 0;
     _nop_();
  }
  return byte;
}

// set TX address
void SetTxAddress(unsigned char* buf)
{
  unsigned char i=0;
  nRF903_TRX_CE=0;
  nRF903_TX_EN=0;
  nRF903_SS=0;
  nRF903WriteRead8Bit(WTA);
  for(i=0;i<4;i++)
  {
     nRF903WriteRead8Bit(buf[i]);
  }
  nRF903_SS=1;
  Delay(100);
}


// set RX address
void SetRxAddress(unsigned char* buf)
{
  unsigned char i=0;
  nRF903_TRX_CE=0;
  nRF903_TX_EN=0;
```

```
  nRF903_SS=0;
  nRF903WriteRead8Bit(WRA);
  for( i=0;i<4;i++)
  {
  nRF903WriteRead8Bit(buf[i]);
  }
  nRF903_SS=1;
  Delay(100);
}

// send one packet,
void SendPacket(unsigned char* buf)
{
    unsigned char i=0;
    nRF903_TRX_CE=1;
    nRF903_TX_EN=1;
    nRF903_SS=0;
    nRF903WriteRead8Bit(WTP);
    for( i=0;i<32;i++) // 32 is the packet length
    {
      nRF903WriteRead8Bit(buf[i]);
    }
    nRF903_SS=1;
    //nRF903_DR=1;
    while(nRF903_DR==0);
    Delay(50);
}

//receive packet (buf 必须能容内有效数据包长度和 RSSI 的值)
unsigned char ReceivePacket(unsigned char* buf)
{
    unsigned char i=0;
    if(nRF903_DR==0)
      return 0;

    nRF903_TRX_CE=1;
    nRF903_TX_EN=0;

    nRF903_SS=0;
    nRF903WriteRead8Bit(RRP);
    for(i=0;i<16;i++)
    {
      buf[i]=nRF903WriteRead8Bit(RRP);
    }
```

```
    buf[i]=nRF903WriteRead8Bit(RRP); //read RSSI
    nRF903_SS=1;
    //while(nRF903_DR_PIN ==1);
    Delay(10);
    return 1;
}


// 进入睡眠模式/离开睡眠模式， s=1 进入睡眠；否则离开睡眠模式
void Enter_Sleep(unsigned char s)
{
  if(s)
  {
    nRF903_SLEEP=1;
    nRF903_SLEEP=0;
  }
  else
  {
    nRF903_SLEEP=0;
    nRF903_SLEEP=1;
  }
  Delay(1);
}


void IO_init()
{
  nRF903_PWR_UP=1;

  nRF903_SS=1;
  nRF903_MOSI=0;
  nRF903_MISO=1;
  nRF903_SLEEP=1;
  nRF903_DR=1;
  nRF903_SCK=0;
  nRF903_TRX_CE=0;
  nRF903_TX_EN=0;
}

// 串口初始化
void StartUART( void )
{                          //9600 bps
     SCON = 0x50;
     TMOD = 0x20;
```

```
    TH1 = 0xFD;
    TL1 = 0xFD;
    PCON = 0x00;
    TR1 = 1;
}

// 串口发送一字节
void R_S_Byte(unsigned char R_Byte)
{
    SBUF = R_Byte;
    while( TI == 0 );
    TI = 0;
}
```

```
// 全局变量
unsigned char RxTxBuf[17] ; //接收发送缓冲区, 有效数据加一字节 RSSI 为
17 字节
unsigned char buf[4]={0x54,0x75,0xC5,0x2A}; //接收&发送地址
```

```
// 简单演示收发数据
void main()
{
  IO_init();

   SetRxAddress(buf);
   Delay(1);
   SetTxAddress(buf);
   Delay(1);
   StartUART();
   while(1)
  {
      if(KEY1 ==0 )
      {
         RxTxBuf[0]=1;
         SendPacket(RxTxBuf); // Send  packet
        RxTxBuf[0]=0;
        Delay(1);
      }
```

```
        if(KEY2 ==0 )
        {
          RxTxBuf[0]=2;
          SendPacket(RxTxBuf);
         RxTxBuf[0]=0;
         Delay(1);
        }

        if(ReceivePacket(RxTxBuf)==1)  // receive  packet
        {
           R_S_Byte(RxTxBuf[0]);       // send the first byte to com
          //R_S_Byte(RxTxBuf[16]);    // rssi value
        }
    }
}
```

七 常见问题问答

1，我已经把 UTC-903 的 PWRUP 置低，为什么电流还是没有下降，并且还能正常发送接受？

答: 因为虽然把 UTC-903 的 PWRUP 置低，VCC 对 RF 芯片不在供电，可是 mcu 的其他管脚还是提供了电流供 UTC-903，所以外面 mcu 必须把和 UTC-903 连接的全部 I/O 脚设置为不会对外供电，比如高阻态。