低功耗扩频无线模块YL-1278RF使用手册

---扩频编码，超远距离---

版本号：V2.2

深圳捷迅易联科技有限公司

电话：0755-26031631　　传真：0755-26521631
邮箱：yl-link@rf-module.cn　　网站：www.rf-module.cn
地址：深圳市南山区科技园中区科智西路1号科苑西工业区南23栋6楼

目录

一、产品概述

YL-1278RF是一款高性能、低功耗、远距离的微功率无线扩频编码模块，内部自动扩频计算和硬件校验处理，用户不需要了解太复杂的射频知识，和硬件调，只是需要调试底层SPI通信，和理解好函数的意义。就可以轻松的应用YL_1278RF扩频模块。YL_1278RF模块非常适合远距离，低数据量和低功耗等应用场合。

射频模块的射频芯片基于扩频跳频技术，在稳定性、抗干扰能力以及接收灵敏度上都超越现有的GFSK射频模块。

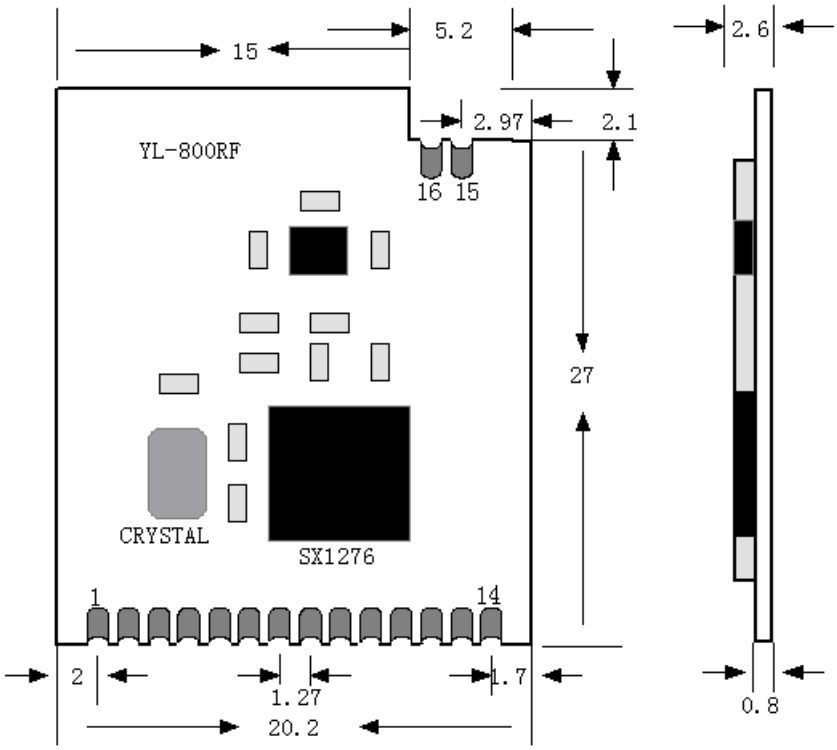
二、产品特点

- 基于 LoRa™ 扩频调制技术。
- 半双工通讯，标准 SPI 通信控制。
- 420~450MHz 免申请频段，其他频段可定制。
- 生产免调试，2.1-3.6V 宽电压范围。
- 微功率发射，标准 100mW，设置功率寄存器。
- 接收灵敏度高达-148dBm，最大发射功率+20dBm。
- 硬件检验，和硬件扩频编码，可以自定义调频机制。
- 接收，发射，CAD 检测，休眠等多种模式任意却换。
- 贴片封装，方便客户嵌入自己的 PCB。
- C 语言函数封装，直接调入函数接口。
- 嵌套式屏蔽盖保护，增加抗干扰性能。

三、应用领域

- ✓ 智能家居、智能交通、传感网络；
- ✓ 工业自动化、农业现代化、建筑智能化；
- ✓ 水、电、气、暖等计量表自动集中抄表系统；
- ✓ 水利、油田、矿井、气象等设备信息采集；
- ✓ 路灯控制、电网监测、风光互补系统；
- ✓ 工业设备数据无线传输以及工业环境监测；
- ✓ 掌机数据采集，和嵌入式设备数据传输；
- ✓ 其他一切需要无线代替有线通讯的情况。

四、尺寸结构



五、引脚定义

引脚符号	引脚功能	引脚说明
GND	电源地	GND
TX	射频开关脚控制	发射接收时用来控制射频开关，用法见后面详细解说。
RX	射频开关脚控制	发射接收时用来控制射频开关，用法见后面详细解说。
REST	模块复位脚	用以复位模块，和初始化寄存器。
GPIO0	模块普通IO口	用户可自定义使用，用法见后面详细解说。
GPIO1	模块普通IO口	用户可自定义使用，用法见后面详细解说。
GPIO2	模块普通IO口	用户可自定义使用，用法见后面详细解说。
GPIO3	模块普通IO口	用户可自定义使用，用法见后面详细解说。
GPIO5	模块普通IO口	用户可自定义使用，用法见后面详细解说。
SCK	SPI时钟输入	SPI通信，用来接收MCU的时钟。
MISO	SPI数据输出	SPI通信，模块发射数据给MCU。
MOSI	SPI数据输入	SPI通信，模块接收MCU的数据。

NSS	SPI使能控制	SPI通信，使能模块的SPI接口。
VCC	供电电源	电源范围2.1V-3.6V

注：

管脚TX，RX,在后面第9章有详细介绍，REST管教在10章有图文介绍，GPIO管教在11章有表格介绍，SPI 在第七章有时序介绍。

六、技术参数：

调制方式	LoRa™扩频
工作频率：	420~450MHz（可定制）
发射功率：	20dBm
接收灵敏度：	-148dBm
工作电压：	2.1～3.6V（输出20dBm）
发射电流：	≤120mA（发射功率20dBm）
接收电流：	≤9.9mA
休眠模式：	≤1uA
工作温度：	-40～+80℃ （工业级别）
工作湿度：	10%～90%相对湿度，无冷凝

注：输入的电源纹波系数要控制在50mV以内,并可提供瞬间脉冲电流300MA以上，脉冲宽度大于800MS。

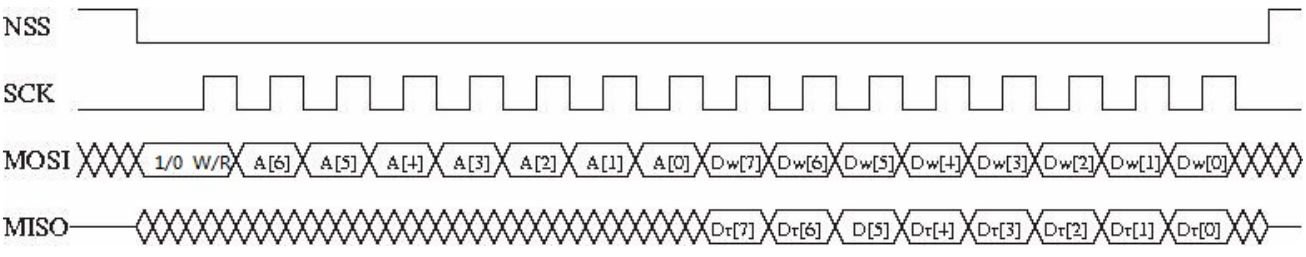
七、SPI 时序说明

YL_1278RF 无线扩频模块是标准的 4 线 SPI 接口，客户可以用 MCU 的 IO 口模拟，也可以使用 MCU 自带的 SPI 接口来进行通信。如果用 IO 口模拟，在高速 MCU 上面注意延时。模块 SPI 提供 3 种读写方式。

1：一个地址后面跟一个数据，NSS 从写地址到(写/读)数据都为低电平。直到数据完成。

2：一个地址后面跟 N 个数据，在数据写入后地址也跟着增加，直到对应最后一个数据。NSS 从地址操作到数据完成都为低电平。

3：FIFO 地址操作，写入 FIFO 地址后，数据写入或读取后地址不增加，只是在 FIFO 地址里面存储或输出。



此图是 SPI 单地址时序图

八、应用电路

无线扩频模块和用户 MCU 连接方式如图 8.1 所示，需要注意共地连接，否则模块可能无法正常

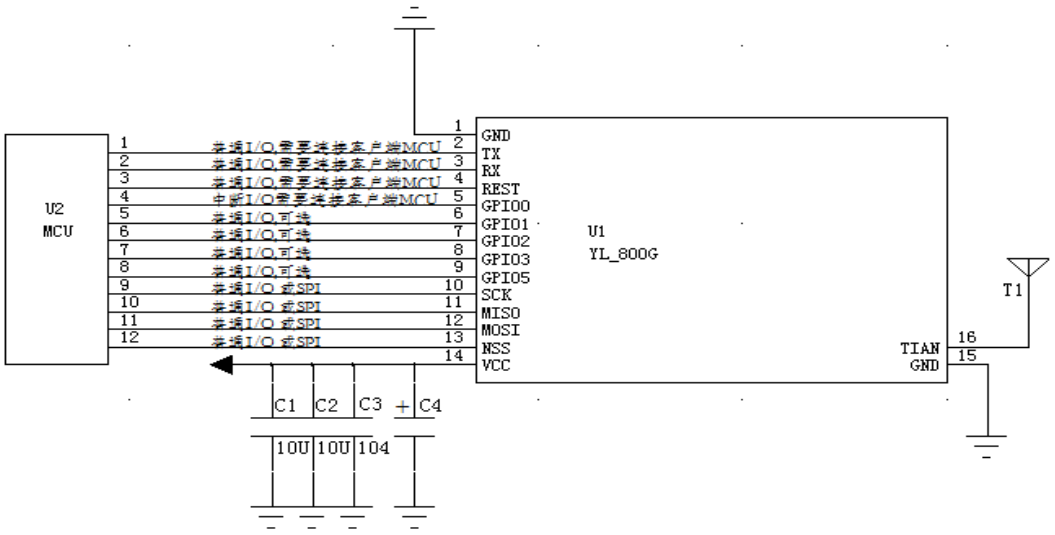


图 8.1　无线模块和用户 MCU 链接图

注：在画原理图时要注意，TX,RX,REST,GPIO0,SCK,MISO,MOSI,NSS 这些脚必须连接到客户 MCU 上面，并且 GPIO0 连接的最好是中断脚。

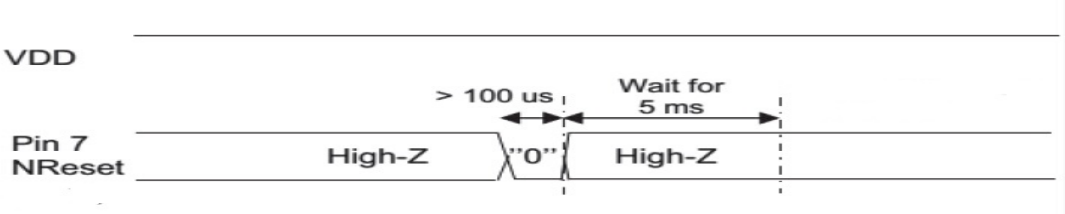
九、TX,RX 控制

TX, RX 管脚，主要是用来切换射频开关管，在发射模式和接收模式切换射频开关管。根据 YL-1278RF 无线扩频模块的硬件原理，在发射和接收模式下控制如下表格。

发射模式	TX管脚	RX管脚	接收模式	TX管脚	RX管脚
	H(高电平)	L(低电平)		L(低电平)	H(高电平)

十、REST 管脚

REST管脚主要是复位YL-1278RF无线扩频模块，低电平有效，高电平运行。注意这个管脚一般是在初始化的时候进行操作，初始化操作成功后就严禁使用此管脚，要保持REST管脚的高电平。



REST管脚操作时序图

十一、GPIO 管脚

YL-1278RF 无线扩频模块有 5 个普通的 GPIO，这些 IO 口的功能可以通过模块的寄存器来设置功能。

寄存器控制管脚对应表格

地址	bit	控制管脚	地址	bit	控制管脚
0X40	7~6	GPIO0	0X41	保留	
	5~4	GPIO1		5~4	GPIO5
	3~2	GPIO2		保留	
	1~0	GPIO3		保留	

管脚功能对应表适应所有模式

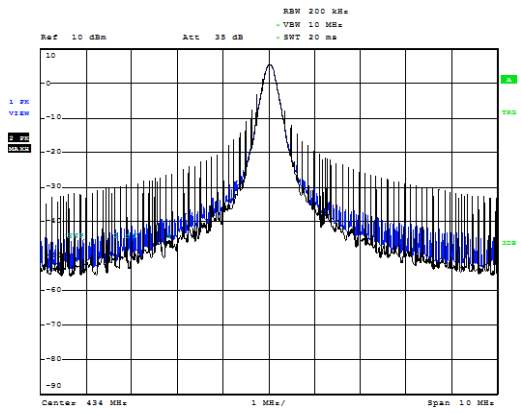
寄存器值	GPIO0	GPIO1	GPIO2	GPIO3	GPIO5
00（bit）	RxDone	RxTimeout	FhssChangeChannel	CadDone	ModeReady
01（bit）	TxDone	FhssChangeChannel	FhssChangeChannel	ValidHeader	ClkOut
10（bit）	CadDone	CadDetected	FhssChangeChannel	PayloadCrcError	ClkOut
11（bit）	保留	保留	保留	保留	保留

十二、扩频参数解说

扩频模块几个基本的参数，和传统的 GFSK 的模块有不同的理解。下面就专门介绍下模块的几个基本的参数。

1）载波频率

以这个频率基准进行扩频载频，如果无数据发送，那么就是出一个载波信号。



载波频谱图

注：在设置载波频率的时候要避开 32M 的倍数频率，如果频率设置为 32M 的倍数则模块的接收灵敏度就会很低，会影响距离（此为厂家芯片特性,请避开这些频率）。

2）扩频因子

扩频因子是码分多址的基本组成部分，码片速率=符号速率*扩频因子，扩频因子的使用使得 TD 中的信道的符号速率选择性更大，为 QOS 保证提供了强有力的支持，扩频因子也决定了可接入终端的数量。扩频因子的大小决定了一个用户的实际数据数率的大小（注意，这里说的是实际数据，例如大家都传输 11111111 这个数据，A 用 11 表示 1，那么他的实际数据是 1111，而 B 用 1111 表示 1，那么他的实际数据为 11，这样 B 的出错概率就比 A 小，但他的数据数率也比 A 小）但是因为正交码的存在，从基站上看，提高扩频因子，对某一用户的实际数据数率降低了，但同时的可用用户数多了（扩频码）整体的实际数据数率却没变。

3）扩频带宽

扩频带宽，简单的说就是你的信号是在以基频为基准多宽的频率下进行调制。下图是 125K 和 250K 的扩频带宽图（紫线是保持线，黄线是调制信号线）。扩频带宽的设置也取决于晶体精度是否支持，我们推荐最低的扩频带宽是 125K。

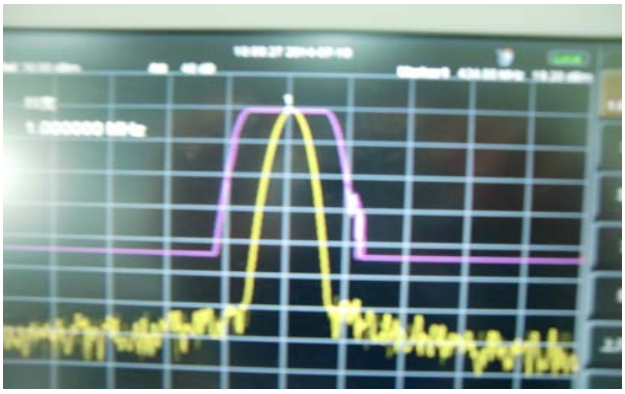


图 1 125K 扩频带宽图

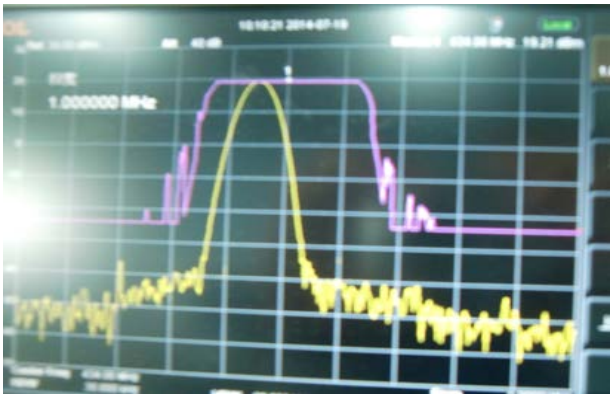


图 2 250K 扩频带宽图

注：在调试模块的通信时，如果两模块要进行通信，一定要保证模块的载波频率，扩频因子，扩频带宽这三个参数相同。

十三、扩频参数设置

1）载波频率

载波频率是通过三寄存器来控制，在软件那章有个专门的数组和函数来体现。这里指讲述下怎么计算三寄存器的值。

如：载波频率是 433M. 433000000/61.035=0X6C4012。这样就算出寄存器的值为 0X6C,0X40，0X12.

2）扩频因子

扩频因子比较好计算，在软件有个专门的变量和函数来体现，这里就不做过多解释。

3）扩频带宽

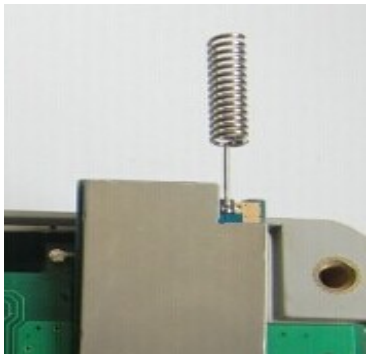
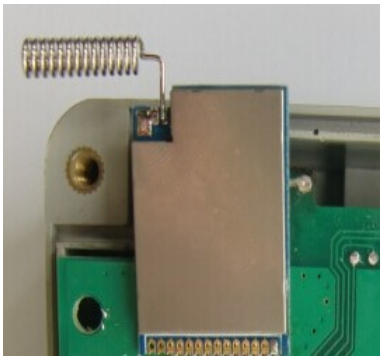
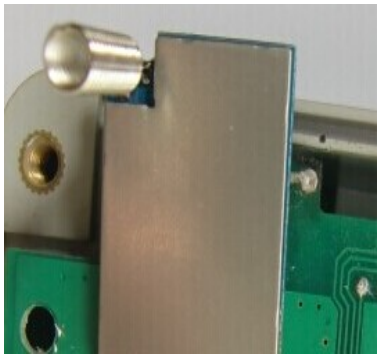
扩频带宽比较好计算，在软件有个专门的变量和函数来体现，这里就不做过多解释。

十四、硬件设计

YL-1278RF 无线扩频模块是贴片式封装，方便嵌入到客户设备底板上，所以客户在设计电路板时就应当把模块当一个元件单元设计。所以在 PCB 布局和走线方面就有很多讲究的地方。

1）PCB 布局

在 PCB 布局时，在符合模具结构的前提下，无线模块应当远离功率器件，场器件，如：喇叭，蜂鸣器，开关电源电感 等一些可以产生场干扰的器件和发热器件。在贴模块区域内，PCB 背面严禁摆放器件。如用内置弹簧天线，那么天线的摆放不可以和模块重叠放置，要么垂直 PCB 板子，或平行模块板边。下面有几个内置天线摆放图片，供客户参考。

如果天线焊接在客户的 PCB 板子上面，那么天线的焊接点不可以离模块太远，射频线不应走的过长。在电源接口处尽量多放点快速响应的电容器件，以保障电源的瞬间脉冲。

2）PCB 走线

数据线的连接最好平行，在同一个面上，线尽量等长。贴模块的区域内部应当走线，尽量保持铜皮的完整性。但天线下面禁止有铺地铜皮，最好是掏空电路板子。

注：输入的电源纹波系数要控制在50mV以内,并可提供瞬间脉冲电流300MA以上，脉冲宽度大于800MS。

十五、参考软件代码

无线模块参考代码在 STM8 单片机上完成的，如果客户用其他的单片机，需要对管脚驱动进行修改。以下是程序的解说，可以提供.C 和.H 文件

1）程序解说

```
#include "stm8l15x.h"
#include "Sx1276.h"
uint8_t  Frequency[3]={0x6c,0x80,0x00};  //频率计算值
uint8_t  SpreadingFactor=11;     //7-12     扩频因子参数
uint8_t  Bw_Frequency=7;         //6-9      扩频带宽参数
uint8_t  powerValue=7;           //0-7      发射功率参数
uint8_t power_data[8]={0X80,0X80,0X80,0X83,0X86,0x89,0x8c,0x8f};
/*
TX,RX 管脚的控制
```

```
*/
  void PA_TXD_OUT(void)
  { RF_RXC_L;
    RF_TXC_H; }
  void PA_RXD_OUT(void)
  {RF_TXC_L;
   RF_RXC_H;}
  void PA_SEELP_OUT(void)
  {RF_TXC_L;
   RF_RXC_L;}
/*
底层 IO 口模拟 SPI
*/

 void RF_SPI_INIT(void)
    {
       RF_CKL_L;
       RF_CE_H;
       RF_SDI_H;
    }
    void RF_SPI_MasterIO(uint8_t out)
    {
       uint8_t i;
       for (i=0;i<8;i++)
       {
         if ((out & 0x80)== 0x80)          /* check if MSB is high */
         RF_SDI_H;
         else
         RF_SDI_L;                       /* if not, set to low */

         RF_CKL_H;                      /* toggle clock high */
         out = (out << 1);                   /* shift 1 place for next bit */
         RF_CKL_L;                       /* toggle clock low */
       }
    }
    uint8_t RF_SPI_READ_BYTE()
    {

       uint8_t j;
       uint8_t i;
       j=0;
         for (i = 0; i < 8; i++)
```

```
        {    RF_CKL_H;
            j = (j << 1);
        if(GPIO_ReadInputDataBit(GPIOB,GPIO_Pin_4) )
              j = j | 0x01;
                 RF_CKL_L;
        }
        return j;
    }
  /*
  寄存器读写函数
  */
    void SX1276WriteBuffer( uint8_t addr, uint8_t buffer)
    {
        RF_CE_L; //NSS = 0;
        RF_SPI_MasterIO( addr | 0x80 );
        RF_SPI_MasterIO( buffer);
        RF_CE_H; //NSS = 1;
    }
    uint8_t SX1276ReadBuffer( uint8_t addr)
    {
        uint8_t i;
        RF_CE_L; //NSS = 0;
        RF_SPI_MasterIO( addr & 0x7F );
        i = RF_SPI_READ_BYTE();
        RF_CE_H; //NSS = 1;
        return i;
    }
void SX1276Reset( void )
    {
        RF_REST_L;
        Delay1s(200);
        RF_REST_H;
        Delay1s(500);
    }
  void SX1276LoRaSetOpMode( RFMode_SET opMode )
      {
                uint8_t opModePrev;
            opModePrev=SX1276ReadBuffer(REG_LR_OPMODE);
            opModePrev &=0xf8;
                   opModePrev |= (uint8_t)opMode;
            SX1276WriteBuffer( REG_LR_OPMODE, opModePrev);
      }
```

```
    void SX1276LoRaFsk( Debugging_fsk_ook opMode )
   {
           uint8_t opModePrev;
           opModePrev=SX1276ReadBuffer(REG_LR_OPMODE);
           opModePrev &=0x7F;
           opModePrev |= (uint8_t)opMode;
           SX1276WriteBuffer( REG_LR_OPMODE, opModePrev);
   }
void SX1276LoRaSetRFFrequency(uint8_t FrequencyADD )
    {
       SX1276WriteBuffer( REG_LR_FRFMSB, Frequency[0]);
       SX1276WriteBuffer( REG_LR_FRFMID, Frequency[1]);
       SX1276WriteBuffer( REG_LR_FRFLSB, Frequency[2]);
    }
void SX1276LoRaSetRFPower( uint8_t power )
{
    SX1276WriteBuffer( REG_LR_PADAC, 0x87);
    SX1276WriteBuffer( REG_LR_PACONFIG,  power_data[power] );
}


void SX1276LoRaSetSpreadingFactor( uint8_t factor )
{
    uint8_t RECVER_DAT;
    SX1276LoRaSetNbTrigPeaks( 3 );
    RECVER_DAT=SX1276ReadBuffer( REG_LR_MODEMCONFIG2);
    RECVER_DAT = ( RECVER_DAT & RFLR_MODEMCONFIG2_SF_MASK ) | ( factor << 4 );
    SX1276WriteBuffer( REG_LR_MODEMCONFIG2, RECVER_DAT );
}
void SX1276LoRaSetNbTrigPeaks( uint8_t value )
{
   uint8_t RECVER_DAT;
   RECVER_DAT = SX1276ReadBuffer( 0x31);
   RECVER_DAT = ( RECVER_DAT & 0xF8 ) | value;
   SX1276WriteBuffer( 0x31, RECVER_DAT );
}
void SX1276LoRaSetErrorCoding( uint8_t value )
{  uint8_t RECVER_DAT;
    RECVER_DAT=SX1276ReadBuffer( REG_LR_MODEMCONFIG1);
    RECVER_DAT = ( RECVER_DAT & RFLR_MODEMCONFIG1_CODINGRATE_MASK ) | ( value <<
1 );
    SX1276WriteBuffer( REG_LR_MODEMCONFIG1, RECVER_DAT);
```

```
       // LoRaSettings.ErrorCoding = value;
    }
    void SX1276LoRaSetPacketCrcOn( bool enable )
    {  uint8_t RECVER_DAT;
RECVER_DAT= SX1276ReadBuffer( REG_LR_MODEMCONFIG2);
RECVER_DAT = ( RECVER_DAT & RFLR_MODEMCONFIG2_RXPAYLOADCRC_MASK ) | ( enable << 2 );
       SX1276WriteBuffer( REG_LR_MODEMCONFIG2, RECVER_DAT );
    }
    void SX1276LoRaSetSignalBandwidth( uint8_t bw )
    {
       uint8_t RECVER_DAT;
       RECVER_DAT=SX1276ReadBuffer( REG_LR_MODEMCONFIG1);
       RECVER_DAT = ( RECVER_DAT & RFLR_MODEMCONFIG1_BW_MASK ) | ( bw << 4 );
       SX1276WriteBuffer( REG_LR_MODEMCONFIG1, RECVER_DAT );
       // LoRaSettings.SignalBw = bw;
    }
    void SX1276LoRaSetImplicitHeaderOn( bool enable )
    {
       uint8_t RECVER_DAT;
       RECVER_DAT=SX1276ReadBuffer( REG_LR_MODEMCONFIG1 );
       RECVER_DAT = ( RECVER_DAT & RFLR_MODEMCONFIG1_IMPLICITHEADER_MASK ) |
( enable );
       SX1276WriteBuffer( REG_LR_MODEMCONFIG1, RECVER_DAT );
    }
    void SX1276LoRaSetSymbTimeout( uint16_t value )
    {
       uint8_t RECVER_DAT[2];
       RECVER_DAT[0]=SX1276ReadBuffer( REG_LR_MODEMCONFIG2 );
       RECVER_DAT[1]=SX1276ReadBuffer( REG_LR_SYMBTIMEOUTLSB );
       RECVER_DAT[0] = ( RECVER_DAT[0] & RFLR_MODEMCONFIG2_SYMBTIMEOUTMSB_MASK ) |
( ( value >> 8 ) & ~RFLR_MODEMCONFIG2_SYMBTIMEOUTMSB_MASK );
       RECVER_DAT[1] = value & 0xFF;
       SX1276WriteBuffer( REG_LR_MODEMCONFIG2, RECVER_DAT[0]);
       SX1276WriteBuffer( REG_LR_SYMBTIMEOUTLSB, RECVER_DAT[1]);
    }
    void SX1276LoRaSetPayloadLength( uint8_t value )
    {
       SX1276WriteBuffer( REG_LR_PAYLOADLENGTH, value );
    }
    void SX1276LoRaSetPreamLength( uint16_t value )
    {
      uint8_t a[2];
```

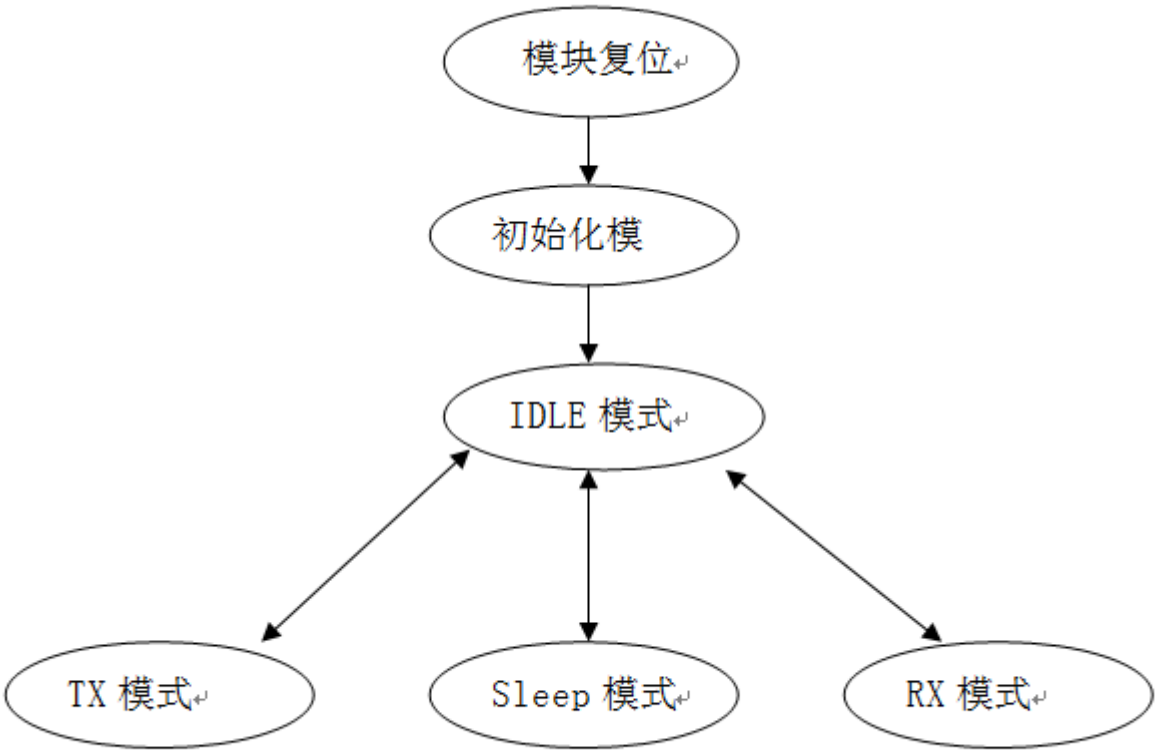
```
    a[0]=(value&0xff00)>>8;
    a[1]=value&0xff;
     SX1276WriteBuffer( REG_LR_PREAMBLEMSB, a[0] );
     SX1276WriteBuffer( REG_LR_PREAMBLELSB, a[1] );
}
void SX1276LoRaSetMobileNode( bool enable )
{   uint8_t RECVER_DAT;
     RECVER_DAT=SX1276ReadBuffer( REG_LR_MODEMCONFIG3 );
     RECVER_DAT = ( RECVER_DAT & RFLR_MODEMCONFIG3_MOBILE_NODE_MASK ) | ( enable
<< 3 );
     SX1276WriteBuffer( REG_LR_MODEMCONFIG3, RECVER_DAT );
}
void SX1276LORA_INT(void)
{
     SX1276Reset();
     RF_SPI_INIT();
     SX1276LoRaSetOpMode(Sleep_mode);
     SX1276LoRaFsk(LORA_mode);
     SX1276LoRaSetOpMode(Stdby_mode);
     SX1276WriteBuffer( REG_LR_DIOMAPPING1,GPIO_VARE_1);
     SX1276WriteBuffer( REG_LR_DIOMAPPING2,GPIO_VARE_2);
     SX1276LoRaSetRFFrequency(0);
     SX1276LoRaSetRFPower(powerValue);
     SX1276LoRaSetSpreadingFactor(SpreadingFactor);           //
     SX1276LoRaSetErrorCoding(CodingRate);
     SX1276LoRaSetPacketCrcOn( TRUE);
     SX1276LoRaSetSignalBandwidth( Bw_Frequency );
     SX1276LoRaSetImplicitHeaderOn( FALSE);
     SX1276LoRaSetPayloadLength( 0xff);
     SX1276LoRaSetSymbTimeout( 0x3FF );
     SX1276LoRaSetMobileNode(TRUE );
}
void RF_RECEIVE (void)
{

  SX1276LoRaSetOpMode( Stdby_mode );
  SX1276WriteBuffer(REG_LR_IRQFLAGSMASK,IRQN_RXD_Value);  //打开发送中断
  SX1276WriteBuffer(REG_LR_HOPPERIOD,  PACKET_MIAX_Value );
  SX1276WriteBuffer( REG_LR_DIOMAPPING1, 0X00 );
  SX1276WriteBuffer( REG_LR_DIOMAPPING2, 0X00 );
  SX1276LoRaSetOpMode( Receiver_mode );
  PA_RXD_OUT();
```

```
    }
    void RF_SEELP(void)
    {
          SX1276LoRaSetOpMode( Stdby_mode );
          SX1276WriteBuffer(REG_LR_IRQFLAGSMASK,  IRQN_SEELP_Value);
SX1276WriteBuffer( REG_LR_DIOMAPPING1, 0X00 );
          SX1276WriteBuffer( REG_LR_DIOMAPPING2, 0X00 );
          SX1276LoRaSetOpMode( Sleep_mode );
          PA_SEELP_OUT();
    }
```

2）程序流程

模块的程序流程比较简单，但有些地方也要注意。在这里大概的讲下程序流程，首先模块上电复位，初使化参数,打开各项功能.在这些工作全部做好后 就可以设置成发射模式来发射数据，和设置接收模式接收数据。值得注意的是在接收模式和发射模式下，不可以直接转换，一定要先转换到 IDIE 模式下才可以进行发射和接收模式的转换。



十六、天线选择

天线是通信系统的重要组成部分，其性能的好坏直接影响通信系统的指标，用户在选择天线时必须首先注重其性能。一般有两个方面：

（1）天线类型——天线的电波覆盖是否符合系统设计要求；

（2）电气性能——天线的频率带宽、增益、阻抗、额定功率等是否符合系统设计要求，一般要求天线的阻抗为 50 欧，驻波比小于 1.4。

我司提供多种天线方案，用户根据实际情况选择，以便达到最佳传输效果。

弹簧天线

胶棒天线

折叠天线

小吸盘天线

十七、使用须知

考虑到空中传输的复杂性，无线数据传输方式固有的一些特点，应注意以下几个问题。

1）数据延迟

由于无线通信发射端是从终端设备接收到一定数量的数据后，或等待一定的时间没有新的数据才开始发射，无线通信发射端到无线通信接收端存在着几到几十毫秒延迟（具体延迟是、空中速率以及数据包的大小决定），另外从无线通信接收端到终端设备也需要一定的时间，但同样的条件下延迟时间是固定的。

2）流量控制

为了确保数据完整性，请尽量控制单次发送的数据包大小，FIFO是64个字节，避免因缓存不足而造成数据溢出，减少丢包的概率。

3）差错控制

YL-1278RF模块虽具有很强的抗干扰能力，但在极端恶劣的条件下时，难免出现接收不佳或丢包的状况。此时客户可增加对系统的链路层协议的开发，如增加丢包重发功能，可提高无线网络的可靠性和灵活性。

十八、注意事项

（1）安装模块时，天线的位置不要过于靠近您产品的MCU，防止干扰；

（2）电源供电时，请确认模块的地线与您产品的地线相连共地；

（3）正常工作时，请勿触摸模块及天线部分，以便达到最佳传输效果。

十玖、故障排除

故障现象	故障原因	解决方法
传输距离不远	环境复杂，障碍物多。	在空旷环境使用，架高天线或引到室外。
	天气恶劣，如雾霾、雨雪、沙尘等	避免在恶劣天气使用，或更换高功率模块。
	天线不匹配，天线增益小。	选择匹配的天线，尽量用高增益天线。
	传输速率过快	降低通信速率，包括串口速率和空中速率。
	可能存在同频或强磁或电源干扰	更换信道或远离干扰源
无法正常通讯	接线不正确	参照说明书接线图正确接线
	接触不良	重新接好电源线、信号线，尽可能焊死
	MCU 的 SPI 时钟过快	调整好 SIP 时序
	收发模块之间的参数不匹配	重新配置参数，频率、信道、空中速率等
	数据吞吐量太大	分包传输，或更换性能更高的模块
	模块主体已损坏	更换新的模块
误码率太高	附近有同频信号干扰	远离干扰源或者修改频率、信道避开
	天馈系统匹配不好	更换良好的天馈系统
	SPI 上时钟波形不标准	检查 SPI 线上是否有干扰
	通讯速率过大	尽可能低速通讯，特别是空中速率
	电源纹波大	更换稳定的电源
	接口电缆线过长	更换好的电缆线或者缩短电缆长度

声明：本公司保留未经通知随时更新本产品使用手册的最终解释权和修改权！