

# KL-NRF24L01 开发指南

V1.8

因为专业，所以卓越

杭州金龙电子有限公司

联系方式:

QQ: 363423117

TEL: 15258818037

地址: 浙江省杭州市拱墅区拱墅工业园祥茂路18号

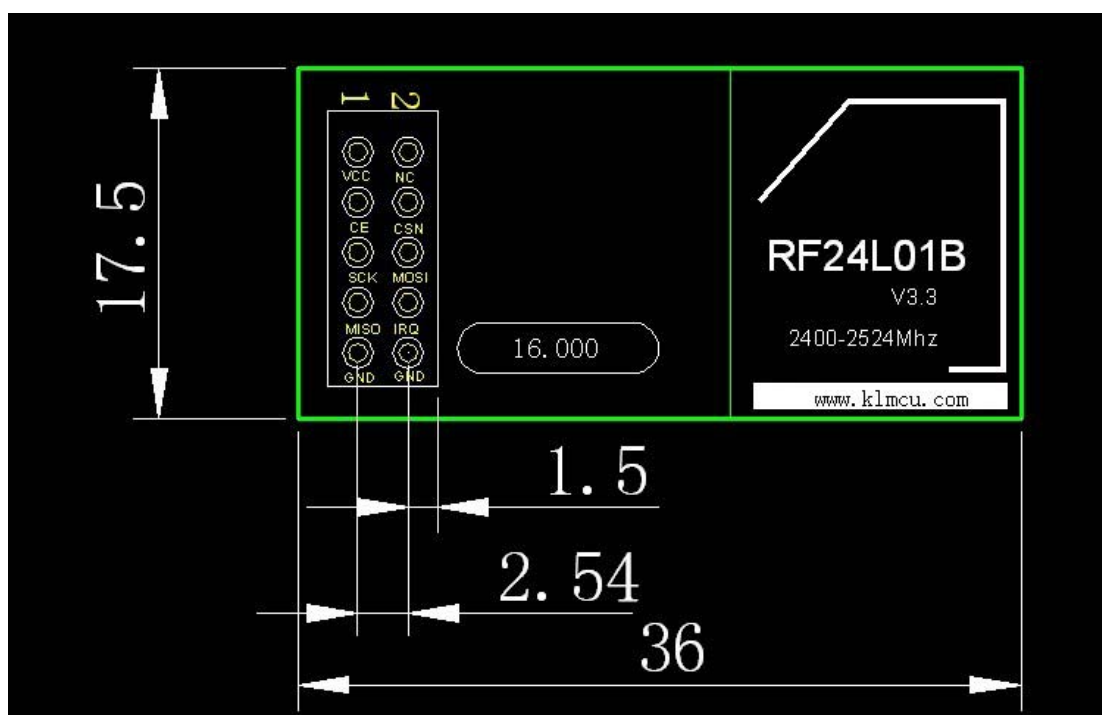
## 目录

一、模块介绍及配合测试小板操作方式 .....	<u>3</u>
二、接口电路以 .....	<u>7</u>
三、模块结构和引脚说明 .....	<u>8</u>
四、工作方式 .....	<u>10</u>
五、配置KL-NRF24L01模块 .....	<u>13</u>
六、参考源代码 .....	<u>15</u>
七、51 系列-测试小板原理图 .....	<u>23</u>
八、备注 .....	<u>24</u>
九、联系方式 .....	<u>25</u>

## 一、 模块介绍



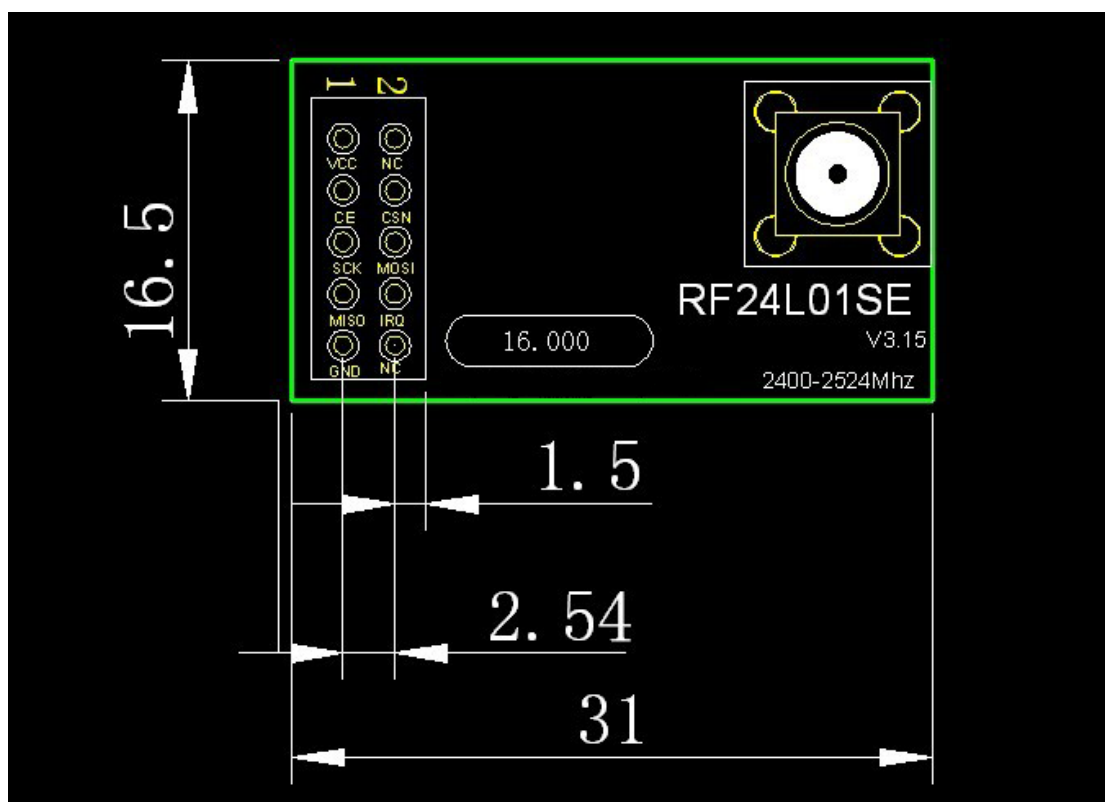
KL-NRF24L01B实物图



KL-NRF24L01尺寸图 (单位mm)



KL-NRF24L01SE模块实物图（SMA接口）



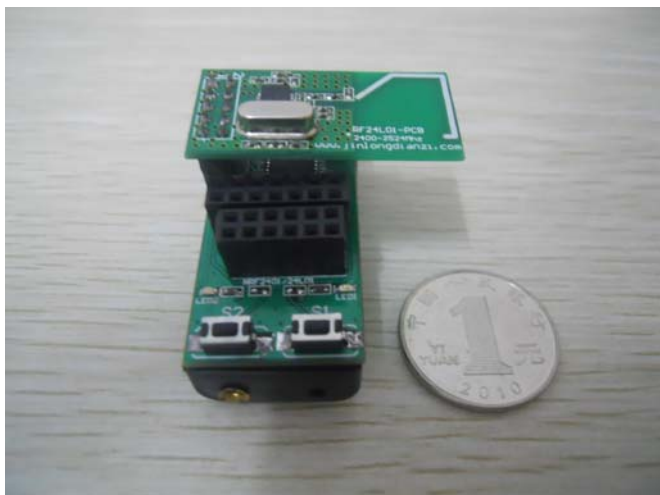
KL-NRF24L01SE模块尺寸图（单位mm）

## 基本特性

- (1) 2.4Ghz 全球开放ISM 频段免许可证使用
- (2) 最高工作速率2Mbps, 高效GFSK调制, 抗干扰能力强, 特别适合工业控制场合
- (3) 126 频道, 满足多点通信和跳频通信需要
- (4) 内置硬件CRC 检错和点对多点通信地址控制
- (5) 低功耗1.9 - 3.6V 工作, 待机模式下状态为22uA; 掉电模式下为900nA
- (6) 内置2.4Ghz 天线, 体积小巧
- (7) 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), 可直接接各种单片机使用, 软件编程非常方便
- (8) 内置专门稳压电路, 使用各种电源包括DC/DC 开关电源均有很好的通信效果
- (9) 标准DIP间距接口, 便于嵌入式应用
- (10) 工作于 Enhanced ShockBurst 具有 Automatic packet handling, Auto packet transaction handling, 具有可选的内置包应答机制, 极大的降低丢包率。
- (11) 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。
- (12) 其他系列的单片机, 如果是5V的, 请参考该系列单片机I/O口输出电流大小, 如果超过10mA, 需要串联电阻分压, 否则容易

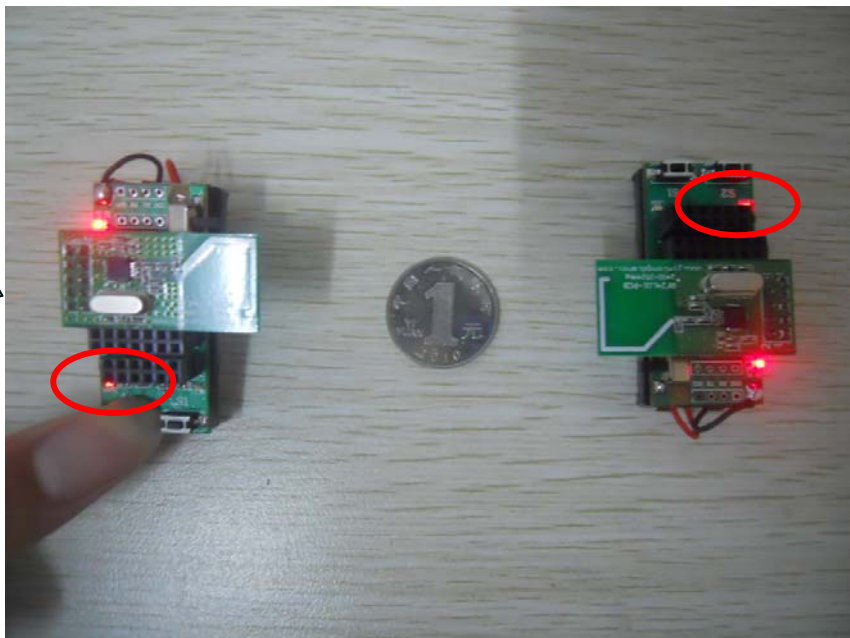
烧毁模块！如果是3.3V的，可以直接和RF2401模块的 IO口线连接。比如AVR系列单片机如果是5V的，一般串接2K的电阻。

## 与测试小板结合使用，操作实物图片



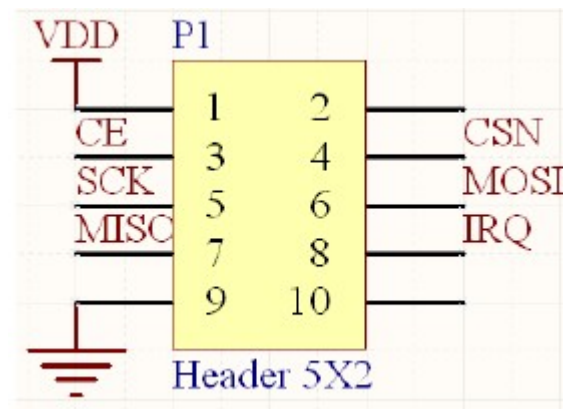
请注意模块插接的方向与接头座子的连接方式

通信成功！





## 二、接口电路

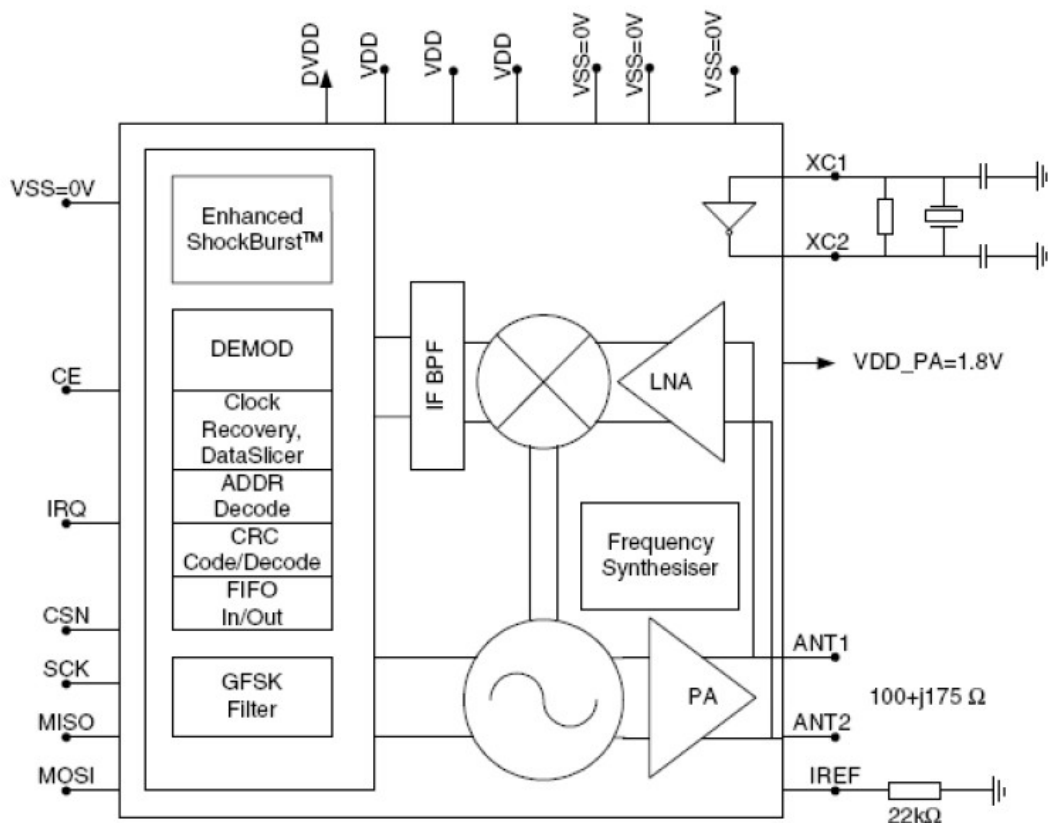


接口说明:

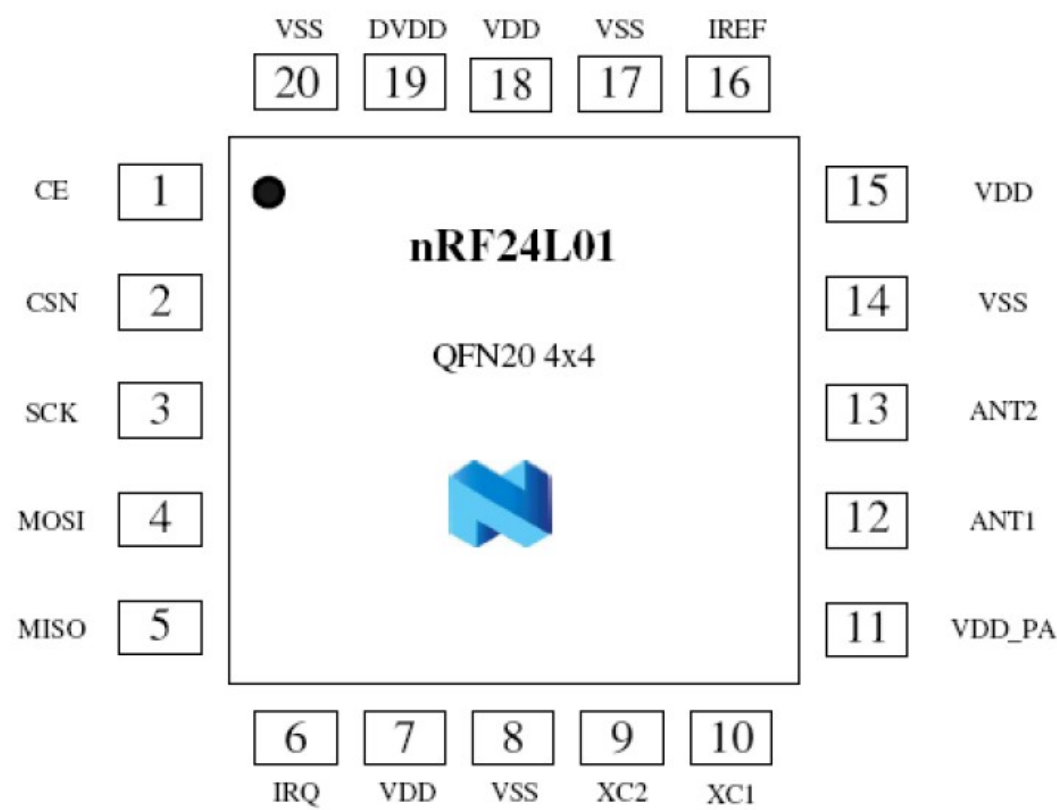
- (1) VCC脚接电压范围为 1.9V~3.6V之间, 不能在这个区间之外, 超过3.6V将会烧毁模块。推荐电压3.3V左右。
- (2) 除电源VCC和接地端, 其余脚都可以直接和普通的5V单片机IO口直接相连, 无需电平转换。当然对3V左右的单片机更加适用了。
- (3) 硬件上面没有SPI的单片机也可以控制本模块, 用普通单片机IO口模拟SPI不需要单片机真正的串口介入, 只需要普通的单片机IO口就可以了, 当然用串口也可以了。
- (4) 9脚接地脚, 需要和母板的逻辑地连接起来; 2脚和9脚悬空。
- (5) 排针间距为2.54mm, 标准DIP插针, 如果需要其他封装接口, 比如密脚插针, 或者其他形式的接口, 可以联系我们定做。

### 三、模块结构和引脚说明

KL-NRF24L01 模块使用 Nordic 公司的 nRF24L01 芯片开发而成。







Pin	Name	Pin function	Description
1	CE	Digital Input	Chip Enable Activates RX or TX mode
2	CSN	Digital Input	SPI Chip Select
3	SCK	Digital Input	SPI Clock
4	MOSI	Digital Input	SPI Slave Data Input
5	MISO	Digital Output	SPI Slave Data Output, with tri-state option
6	IRQ	Digital Output	Maskable interrupt pin
7	VDD	Power	Power Supply (+3V DC)
8	VSS	Power	Ground (0V)
9	XC2	Analog Output	Crystal Pin 2
10	XC1	Analog Input	Crystal Pin 1
11	VDD_PA	Power Output	Power Supply (+1.8V) to Power Amplifier
12	ANT1	RF	Antenna interface 1
13	ANT2	RF	Antenna interface 2
14	VSS	Power	Ground (0V)
15	VDD	Power	Power Supply (+3V DC)
16	IREF	Analog Input	Reference current
17	VSS	Power	Ground (0V)
18	VDD	Power	Power Supply (+3V DC)
19	DVDD	Power Output	Positive Digital Supply output for de-coupling purposes
20	VSS	Power	Ground (0V)

## 四、工作方式

KL-NRF2401有工作模式有四种:

收发模式

配置模式

空闲模式

关机模式

工作模式由 PWR\_UP register 、 PRIM\_RX register 和 CE 决定, 详见下表。

Mode	PWR_UP register	PRIM_RX register	CE	FIFO state
RX mode	1	1	1	-
TX mode	1	0	1	Data in TX FIFO
TX mode	1	0	1→0	Stays in TX mode until packet transmission is finished
Standby-II	1	0	1	TX FIFO empty
Standby-I	1	-	0	No ongoing packet transmission
Power Down	0	-	-	-

### 4.1 收发模式

收发模式有Enhanced ShockBurst™收发模式、 ShockBurst™收发模式和直接收发模式三种, 收发模式由器件配置字决定, 具体配置将在器件配置部分详细介绍。

#### 4.1.1 Enhanced ShockBurst™收发模式

Enhanced ShockBurst™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(1Mbps)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。Enhanced ShockBurst™技术同时也减小了整个系统的平均工作电流。

在Enhanced ShockBurst™收发模式下, KL-NRF24L01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, 在发送模式下, 置CE为高, 至少10us, 将发送过程完成后。

#### 4.1.1.1 Enhanced ShockBurst™发射流程

- A. 把接收机的地址和要发送的数据按时序送入KL-NRF24L01;
- B. 配置CONFIG寄存器, 使之进入发送模式。 C. 微控制器把CE置高 (至少10us), 激发KL-NRF24L01进行Enhanced ShockBurst™发射;
- D. KL-NRF24L01的Enhanced ShockBurst™发射 (1) 给射频前端供电; (2) 射频数据打包 (加字头、CRC校验码); (3) 高速发射数据包; (4) 发射完成, KL-NRF24L01进入空闲状态。

4.1.1.2 Enhanced ShockBurst™接收流程 A. 配置本机地址和要接收的数据包大小; B. 配置CONFIG寄存器, 使之进入接收模式, 把CE置高。

- C. 130us 后, KL-NRF24L01 进入监视状态, 等待数据包的到来;
- D. 当接收到正确的数据包 (正确的地址和CRC校验码), KL-NRF24L01自动把字头、地址和CRC校验位移去;
- E. KL-NRF24L01通过把STATUS寄存器的RX-DR置位 (STATUS一般引起微控制器中断) 通知微控制器; F. 微控制器把数据从KL-NRF24L01

读出; G. 所有数据读取完毕后, 可以清除STATUS寄存器。

KL-NRF24L01可以进入四种主要的模式之一。

#### 4.1.2 ShockBurst™收发模式

ShockBurst™收发模式可以与Nrf2401a, 02, E1及E2兼容, 具体表述前看本公司的NewMsg-RF2401文档。

#### 4.2 空闲模式

KL-NRF24L01的空闲模式是为了减小平均工作电流而设计, 其最大的优点是, 实现节能的同时, 缩短芯片的起动时间。在空闲模式下, 部分片内晶振仍在工作, 此时的工作电流跟外部晶振的频率有关。

#### 4.4 关机模式

在关机模式下, 为了得到最小的工作电流, 一般此时的工作电流为900nA左右。关机模式下, 配置字的内容也会被保持在KL-NRF24L01片内, 这是该模式与断电状态最大的区别。

## 五、配置KL-NRF24L01模块

KL-NRF24L01的所有配置工作都是通过SPI完成，共有30字节的配置字。

我们推荐KL-NRF24L01工作于Enhanced ShockBurst™ 收发模式，这种工作模式下，系统的程序编制会更加简单，并且稳定性也会更高，因此，下文着重介绍把KL-NRF24L01配置为Enhanced ShockBurst™ 收发模式的器件配置方法。

ShockBurst™的配置字使KL-NRF24L01能够处理射频协议，在配置完成后，在KL-NRF24L01工作的过程中，只需改变其最低一个字节中的内容，以实现接收模式和发送模式之间切换。

ShockBurst™的配置字可以分为以下四个部分：

数据宽度：声明射频数据包中数据占用的位数。这使得KL-NRF24L01能够区分接收数据包中的数据 and CRC校验码；

地址宽度：声明射频数据包中地址占用的位数。这使得KL-NRF24L01能够区分地址和数据；

地址：接收数据的地址，有通道0到通道5的地址；

CRC：使 KL-NRF24L01 能够生成 CRC 校验码和解码。

当使用KL-NRF24L01片内的CRC技术时，要确保在配置字 (CONFIG的EN\_CRC) 中CRC校验被使能，并且发送和接收使用相同的协议。

KL-NRF24L01 配置字的 CONFIG 寄存器的位描述如下表所示。

Address (Hex)	Mnemonic	Bit	Reset Value	Type	Description
00	CONFIG				Configuration Register
	Reserved	7	0	R/W	Only '0' allowed
	MASK_RX_DR	6	0	R/W	Mask interrupt caused by RX_DR 1: Interrupt not reflected on the IRQ pin 0: Reflect RX_DR as active low interrupt on the IRQ pin
	MASK_TX_DS	5	0	R/W	Mask interrupt caused by TX_DS 1: Interrupt not reflected on the IRQ pin 0: Reflect TX_DS as active low interrupt on the IRQ pin
	MASK_MAX_RT	4	0	R/W	Mask interrupt caused by MAX_RT 1: Interrupt not reflected on the IRQ pin 0: Reflect MAX_RT as active low interrupt on the IRQ pin
	EN_CRC	3	1	R/W	Enable CRC. Forced high if one of the bits in the EN_AA is high
	CRCO	2	0	R/W	CRC encoding scheme '0' - 1 byte '1' - 2 bytes
	PWR_UP	1	0	R/W	1: POWER UP, 0:POWER DOWN
	PRIM_RX	0	0	R/W	1: PRX, 0: PTX



## 六、参考源代码

```
/*
Email: 363423117@qq.com
官方网址: http://www.klmcu.com
*/

#include <reg52.h>
#include <intrins.h>

typedef unsigned char uchar;
typedef unsigned char uint;

//*****NRF24L01 端口定义*****

sbit MISO    =P1^5;
sbit MOSI    =P1^4;
sbit SCK     =P1^3;
sbit CE      =P1^1;
sbit CSN     =P1^2;
sbit IRQ     =P1^6;

//*****按键*****

sbit KEY1=P2^6;
sbit KEY2=P2^5;

//*****数码管位选*****

sbit led1=P2^4;
sbit led2=P3^5;

//*****NRF24L01*****

#define TX_ADR_WIDTH    5    // 5 uints TX address width
#define RX_ADR_WIDTH    5    // 5 uints RX address width
#define TX_PLOAD_WIDTH  20   // 20 uints TX payload
#define RX_PLOAD_WIDTH  20   // 20 uints TX payload
uint const TX_ADDRESS[TX_ADR_WIDTH]= {0x34,0x43,0x10,0x10,0x01}; //本地地址
uint const RX_ADDRESS[RX_ADR_WIDTH]= {0x34,0x43,0x10,0x10,0x01}; //接收地址

//*****NRF24L01 寄存器指令*****

#define READ_REG        0x00  // 读寄存器指令
#define WRITE_REG       0x20  // 写寄存器指令
#define RD_RX_PLOAD     0x61  // 读取接收数据指令
#define WR_TX_PLOAD     0xA0   // 写待发数据指令
#define FLUSH_TX        0xE1  // 冲洗发送 FIFO 指令
#define FLUSH_RX        0xE2  // 冲洗接收 FIFO 指令
#define REUSE_TX_PL     0xE3  // 定义重复装载数据指令
#define NOP             0xFF   // 保留

//*****SPI(nRF24L01)寄存器地址*****
```

```
#define CONFIG          0x00 // 配置收发状态, CRC 校验模式以及收发状态响应方式
#define EN_AA           0x01 // 自动应答功能设置
#define EN_RXADDR        0x02 // 可用信道设置
#define SETUP_AW         0x03 // 收发地址宽度设置
#define SETUP_RETR       0x04 // 自动重发功能设置
#define RF_CH            0x05 // 工作频率设置
#define RF_SETUP         0x06 // 发射速率、功耗功能设置
#define STATUS           0x07 // 状态寄存器
#define OBSERVE_TX       0x08 // 发送监测功能
#define CD               0x09 // 地址检测
#define RX_ADDR_P0       0x0A // 频道 0 接收数据地址
#define RX_ADDR_P1       0x0B // 频道 1 接收数据地址
#define RX_ADDR_P2       0x0C // 频道 2 接收数据地址
#define RX_ADDR_P3       0x0D // 频道 3 接收数据地址
#define RX_ADDR_P4       0x0E // 频道 4 接收数据地址
#define RX_ADDR_P5       0x0F // 频道 5 接收数据地址
#define TX_ADDR          0x10 // 发送地址寄存器
#define RX_PW_P0         0x11 // 接收频道 0 接收数据长度
#define RX_PW_P1         0x12 // 接收频道 0 接收数据长度
#define RX_PW_P2         0x13 // 接收频道 0 接收数据长度
#define RX_PW_P3         0x14 // 接收频道 0 接收数据长度
#define RX_PW_P4         0x15 // 接收频道 0 接收数据长度
#define RX_PW_P5         0x16 // 接收频道 0 接收数据长度
#define FIFO_STATUS      0x17 // FIFO 栈入栈出状态寄存器设置
```

```
//*****
```

```
void Delay(unsigned int s);
void inerDelay_us(unsigned char n);
void init_NRF24L01(void);
uint SPI_RW(uint uchar);
uchar SPI_Read(uchar reg);
void SetRX_Mode(void);
uint SPI_RW_Reg(uchar reg, uchar value);
uint SPI_Read_Buf(uchar reg, uchar *pBuf, uchar uchars);
uint SPI_Write_Buf(uchar reg, uchar *pBuf, uchar uchars);
unsigned char nRF24L01_RxPacket(unsigned char* rx_buf);
void nRF24L01_TxPacket(unsigned char * tx_buf);
```

```
//*****长延时*****
```

```
void Delay(unsigned int s)
{
    unsigned int i;
    for(i=0; i<s; i++);
}
```

```
for(i=0; i<s; i++);
}

//*****

uint    bdata sta;    //状态标志
sbit  RX_DR  =sta^6;
sbit  TX_DS  =sta^5;
sbit  MAX_RT  =sta^4;

//*****

/*延时函数
//*****/

void inerDelay_us(unsigned char n)
{
    for(;n>0;n--)
        _nop_();
}

//*****

/*NRF24L01 初始化
//*****/

void init_NRF24L01(void)
{
    inerDelay_us(100);
    CE=0;    // chip enable
    CSN=1;    // Spi  disable
    SCK=0;    //
    SPI_Write_Buf(WRITE_REG + TX_ADDR, TX_ADDRESS, TX_ADR_WIDTH);    //
    写本地地址
    SPI_Write_Buf(WRITE_REG + RX_ADDR_P0, RX_ADDRESS, RX_ADR_WIDTH);    //
    写接收端地址
    SPI_RW_Reg(WRITE_REG + EN_AA, 0x01);    // 频道 0 自动  ACK  应答 允许

    SPI_RW_Reg(WRITE_REG + EN_RXADDR, 0x01);    //允许接收地址只有频道 0, 如果
    需要多频道可以参考 Page21
    SPI_RW_Reg(WRITE_REG + RF_CH, 0);    // 设置信道工作为 2.4GHZ, 收发必须一致
    SPI_RW_Reg(WRITE_REG + RX_PW_P0, RX_PLOAD_WIDTH);    //设置接收数据长度,
    本次设置为 32 字节
    SPI_RW_Reg(WRITE_REG + RF_SETUP, 0x07);    //设置发射速率为 1MHZ, 发射
    功率为最大值 0dB
}

//*****
```

```
*****

/*函数: uint SPI_RW(uint uchar)
/*功能: NRF24L01 的 SPI 写时序
/*****
*****/

uint SPI_RW(uint uchar)
{
    uint bit_ctr;
    for(bit_ctr=0;bit_ctr<8;bit_ctr++) // output 8-bit
    {
        MOSI = (uchar & 0x80);          // output 'uchar', MSB to MOSI
        uchar = (uchar << 1);           // shift next bit into MSB..
        SCK = 1;                        // Set SCK high..
        uchar |= MISO;                  // capture current MISO bit
        SCK = 0;                        // ..then set SCK low again
    }
    return(uchar);                     // return read uchar
}

/*****
*****/

/*函数: uchar SPI_Read(uchar reg)
/*功能: NRF24L01 的 SPI 时序
/*****
*****/

uchar SPI_Read(uchar reg)
{
    uchar reg_val;

    CSN = 0;                          // CSN low, initialize SPI communication...
    SPI_RW(reg);                      // Select register to read from..
    reg_val = SPI_RW(0);              // ..then read registervalue
    CSN = 1;                          // CSN high, terminate SPI communication

    return(reg_val);                  // return register value
}

/*****
/*功能: NRF24L01 读写寄存器函数
/*****

uint SPI_RW_Reg(uchar reg, uchar value)
{
    uint status;
```

```

    CSN = 0;                // CSN low, init SPI transaction
    status = SPI_RW(reg);    // select register
    SPI_RW(value);           // ..and write value to it..
    CSN = 1;                // CSN high again

    return(status);          // return nRF24L01 status uchar
}

/*****
*****/
/*函数: uint SPI_Read_Buf(uchar reg, uchar *pBuf, uchar uchars)
/*功能: 用于读数据, reg: 为寄存器地址, pBuf: 为待读出数据地址, uchars: 读出数据的
个数
*****/
uint SPI_Read_Buf(uchar reg, uchar *pBuf, uchar uchars)
{
    uint status, uchar_ctr;

    CSN = 0;                // Set CSN low, init SPI transaction
    status = SPI_RW(reg);    // Select register to write to and read status uchar

    for(uchar_ctr=0; uchar_ctr<chars; uchar_ctr++)
        pBuf[uchar_ctr] = SPI_RW(0);    //

    CSN = 1;

    return(status);          // return nRF24L01 status uchar
}

/*****
*****/
/*函数: uint SPI_Write_Buf(uchar reg, uchar *pBuf, uchar uchars)
/*功能: 用于写数据: 为寄存器地址, pBuf: 为待写入数据地址, uchars: 写入数据的个数
*****/
uint SPI_Write_Buf(uchar reg, uchar *pBuf, uchar uchars)
{
    uint status, uchar_ctr;

    CSN = 0;                //SPI 使能
    status = SPI_RW(reg);

```

```
for(uchar_ctr=0; uchar_ctr<uchars; uchar_ctr++) //
    SPI_RW(*pBuf++);
CSN = 1;          //关闭 SPI
return(status);   //
}

/*****
*****/

/*函数: void SetRX_Mode(void)
/*功能: 数据接收配置
/*****
*****/

void SetRX_Mode(void)
{
    CE=0;
    SPI_RW_Reg(WRITE_REG + CONFIG, 0x0f);          // IRQ 收发完成中断响应, 16
    位 CRC , 主接收
    CE = 1;
    inerDelay_us(130);
}

/*****
*****/

/*函数: unsigned char nRF24L01_RxPacket(unsigned char* rx_buf)
/*功能: 数据读取后放如 rx_buf 接收缓冲区中
/*****
*****/

unsigned char nRF24L01_RxPacket(unsigned char* rx_buf)
{
    unsigned char revale=0;
    sta=SPI_Read(STATUS);    // 读取状态寄存其来判断数据接收状况
    if(RX_DR)                // 判断是否接收到数据
    {
        CE = 0;              //SPI 使能
        SPI_Read_Buf(RD_RX_PLOAD,rx_buf,TX_PLOAD_WIDTH);    // read receive
                                                                Payload from
                                                                RX_FIFO buffer
        revale =1;            //读取数据完成标志
    }
    SPI_RW_Reg(WRITE_REG+STATUS,sta);    //接收到数据后 RX_DR,TX_DS,MAX_PT
                                         都置高为 1, 通过写 1 来清楚中断标志
    return revale;
}
```



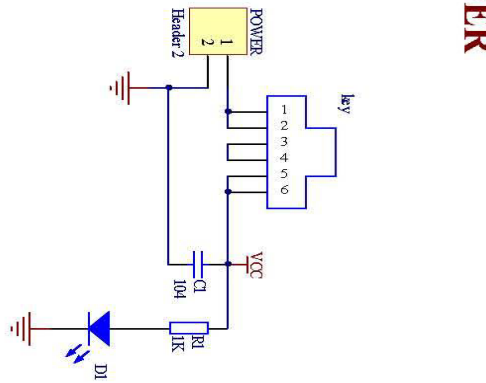
```
/******  
*****  
/*函数: void nRF24L01_TxPacket(unsigned char * tx_buf)  
/*功能: 发送 tx_buf 中数据  
/******  
*****/  
void nRF24L01_TxPacket(unsigned char * tx_buf)  
{  
    CE=0;          //StandBy I 模式  
    SPI_Write_Buf(WRITE_REG + RX_ADDR_P0, TX_ADDRESS, TX_ADR_WIDTH); //  
    装载接收端地址  
    SPI_Write_Buf(WR_TX_PLOAD, tx_buf, TX_PLOAD_WIDTH);    // 装载数据  
    SPI_RW_Reg(WRITE_REG + CONFIG, 0x0e);                  // IRQ 收发完成中断响应,  
                                                             16 位 CRC, 主发送  
    CE=1;          //置高 CE, 激发数据发送  
    inerDelay_us(10);  
}  
  
/******主函数*****  
void main(void)  
{  
    unsigned char tf =0;  
    unsigned char TxBuf[20]={0};    //  
    unsigned char RxBuf[20]={0};  
    init_NRF24L01();  
    led1=1;led2=1;  
    P0=0x00;  
    TxBuf[1] = 1 ;  
    TxBuf[2] = 1 ;  
    nRF24L01_TxPacket(TxBuf);    // Transmit Tx buffer data  
    Delay(6000);  
    P0=0xBF;  
    while(1)  
    {  
        if(KEY1 ==0 )  
        {  
            TxBuf[1] = 1 ;  
            tf = 1 ;  
            led1=0;  
            Delay(120);  
            led1=1;  
            Delay(120);  
        }  
    }  
}
```

```
    }
    if(KEY2 ==0 )
    {
        TxBuf[2] =1 ;
        tf = 1 ;
        led2=0;
        Delay(120);
        led2=1;
        Delay(120);
    }
    if (tf==1)
    {
        nRF24L01_TxPacket(TxBuf);    // Transmit Tx buffer data
        TxBuf[1] = 0x00;
        TxBuf[2] = 0x00;
        tf=0;
        Delay(1000);
    }
    SetRX_Mode();
    RxBuf[1] = 0x00;
    RxBuf[2] = 0x00;
    Delay(1000);
    nRF24L01_RxPacket(RxBuf);
    if(RxBuf[1]|RxBuf[2])
    {
        if( RxBuf[1]==1)
        {
            led1=0;
        }
        if( RxBuf[2]==1)
        {
            led2=0;
        }
        Delay(6000);    //old is '1000'
    }

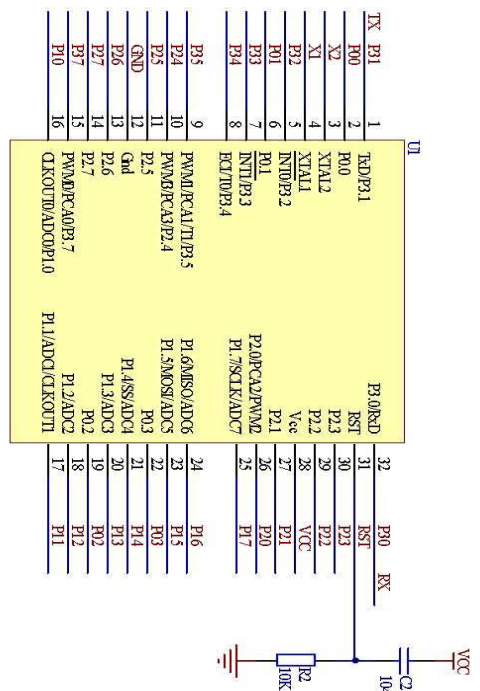
    RxBuf[1] = 0x00;
    RxBuf[2] = 0x00;
    led1=1;
    led2=1;
}
}
```

## 七、51 系列-测试小板原理图

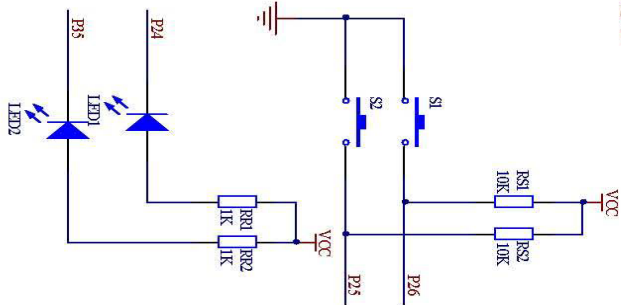
### POWER



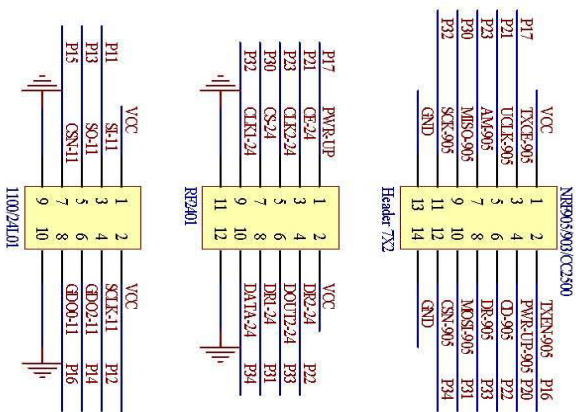
### IC



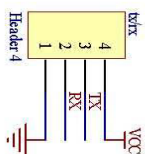
### TEST



### RF PORT



### DWONLOAD



## 八、备注

杭州金龙电子有限公司主要经营范围:

- ☒ 433M/915M/2.4G 频段无线数传模块 ;
- ☒ 常用 MCU 开发平台 (51, AVR, MSP430, PIC 等);
- ☒ ARM7、ARM9 开发平台;
- ☒ 有源 RFID 系统, 有源电子标签;
- ☒ 开关电源以及镍氢电池保护板 、 锂离子电池保护板、
- ☒ 锂铁磷电池保护板;
- ☒ 电池电量显示板;
- ☒ LED 显示板;
- ☒ FFC (柔性扁平电缆线);
- ☒ LVDS (液晶屏线;)

## 九、联系方式

公司名称: 杭州金龙电子有限公司

电话: 15258818037

Email: [363423117@qq.com](mailto:363423117@qq.com)

QQ: 363423117