NRF905 无线数传模块

开发文档

V1.8

携手共赢

杭州金龙电子有限公司

目　录

一、JL-RF905 模块介绍 ····· （3）
二、JL-RF905 接口电路介绍 ····· （5）
三、模块引脚和电气参数说明 ····· （7）
四、工作方式 ····· （8）
五、配置JL-RF905模块 ····· （11）
六、JL-RF905 编程指南以及代码 ····· （17）
七、51 系列测试小板原理图 ····· （38）
八、备注 ····· （39）
九、联系方式 ····· （40）

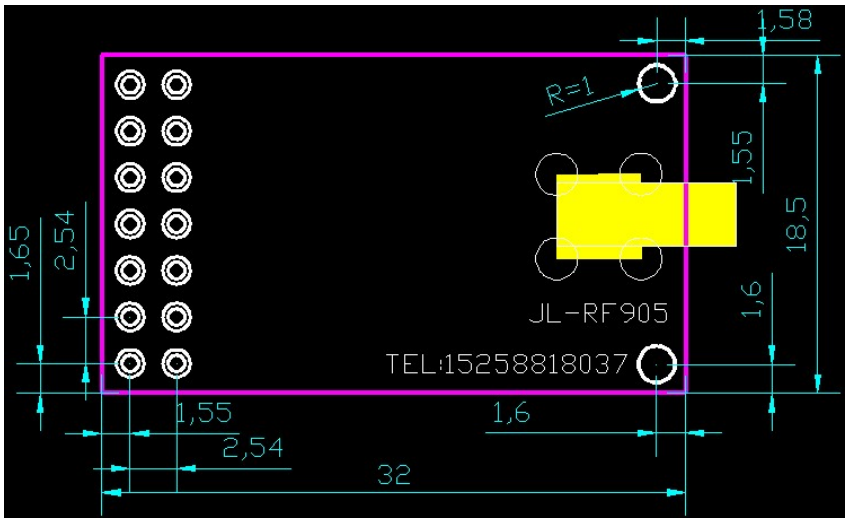
一、JL-RF905 模块介绍

JL_RF905 实物图（柱状天线）

天线水平 180 度

JL_RF905 实物图（柱状天线）

天线垂直 90 度



JL_RF905 机械尺寸

（单位：mm）

【特点】

(1) 433/868/915Mhz 开放ISM 频段免许可证使用 （亚洲地区客户使用433Mhz，具体频段选择请参详下面这张图片）；

(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，传输距离： 开阔地传输500米,距离传输视具体环境而定 ；

(3)接收灵敏度：-100dBm，最大发射功率：10mW (+10dBm)，瞬间最大工作电流:<30mA，125 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 ；

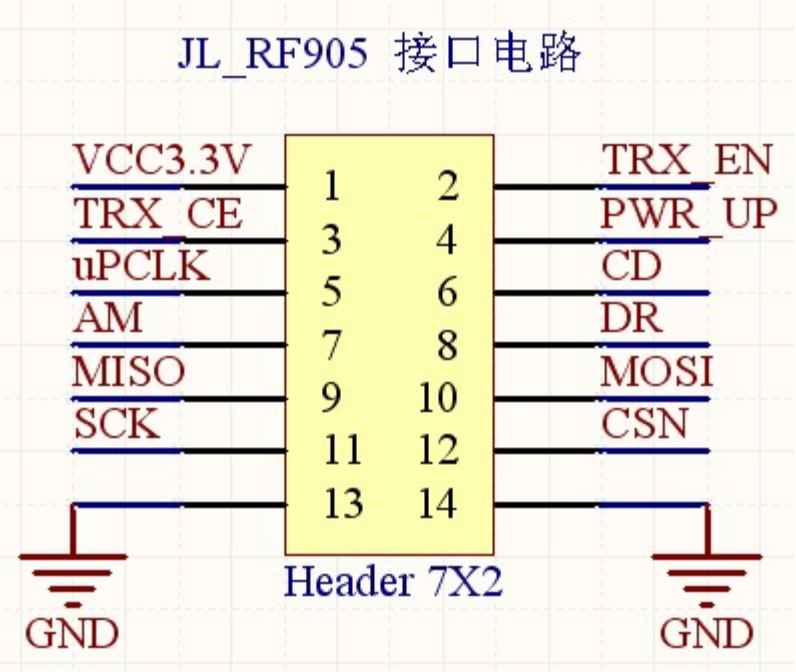
(5) 低功耗1.9 - 3.6V 工作，一般情况下，请使用3.3V供电待机模式下状态仅为2.5uA ，收发模式切换时间 < 650us ；

(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30mA; RX Mode：12.2mA ；

(8) 标准DIP间距接口，便于嵌入式应用 ；

二、JL-RF905 接口电路介绍



管脚	名称	管脚功能	说明
1	VCC	电源	电源+1.9-3.6V DC
2	TX_EN	数字输入	TX_EN= 1 TX 模式 TX_EN= 0 RX 模式
3	TRX_CE	数字输入	使能芯片发射或接收
4	PWR_UP	数字输入	芯片上电
5	uCLK	时钟输出	本模块该脚废弃不用，向后兼容
6	CD	数字输出	载波检测
7	AM	数字输出	地址匹配
8	DR	数字输出	接收或发射数据完成
9	MISO	SPI 接口	SPI 输出
10	MOSI	SPI 接口	SPI 输入
11	SCK	SPI 时钟	SPI 时钟
12	CSN	SPI 使能	SPI 使能
13	GND	地	接地
14	GND	地	接地

说明:

(1) VCC脚接电压范围为 3 V¯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，间距2.54mm,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RF905模块的IO口线连接;

三、模块引脚和电气参数说明

JL-RF905模块使用Nordic公司的nRF905芯片开发而成。

JL RF905 单片无线收发器工作在 433/868/915MHZ 的 ISM 频段由一个完全集成的频率调制器一个带解调器的接收器一个功率放大器一个晶体震荡器和一个调节器组成ShockBurst 工作模式的特点是自动产生前导码 和 CRC 可以很容易通过 SPI 接口进行编程配置电流消耗很低在发射功率为＋10dBm 时发射电流为 30mA 接收电流为 12.5mA. 进入 POWERDOWN 模式可以很容易实现节电.

JL-RF905SE 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	9	mA
接收模式时工作电流	12.5	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

JL-RF905SE 模块工作电压与最大发射增益参考数据

工作电压(模块VCC供电电压)	模块最大发射增益(dBm)
+3.3V	+7.3dBm
+3.6V	+10dBm

四、工作方式

nRF905 工作模式由TRX_CE、TX_EN、PWR_UP 的设置来设定。

PWR_UP	TRX_CE	TX_EN	工作模式
0	X	X	掉电和SPI 编程
1	0	X	Standby 和SPI 编程
1	1	0	ShockBurst RX
1	1	1	ShockBurst TX

JL-RF905一共有四种工作模式，其中有两种活动RX/TX 模式和两种节电模式。

活动模式 ：

ShockBurst RX

ShockBurst TX

节电模式 ：

掉电 和 SPI编程

STANDBY 和 SPI 编程

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， RF905 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RF905发送流程分以下几步：

A. 当微控制器有数据要发送时，通过SPI接口，按时序把接收机的地址和要发送的数据送传给RF905，SPI接口的速率在通信协议和器件配置时确定；

B. 微控制器置高TRX_CE和TX_EN，激发RF905的ShockBurstTM发送模式；

C. RF905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；

D. AUTO_RETRAN被置高，RF905不断重发，直到TRX_CE被置低；

E. 当TRX_CE被置低，RF905发送过程完成，自动进入空闲模式。 注意：ShockBurstTM工作模式保证，一旦发送数据的过程开始，无论TRX_EN和TX_EN引脚是高或低，发送过程都会被处理完。只有在前一个数据包被发送完毕，RF905才能接受下一个发送数据包。

4.1.2 ShockBurst RX 接收流程

接收流程

A. 当TRX_CE为高、TX_EN为低时，RF905进入ShockBurstTM接收模式；

B. 650us后，RF905不断监测，等待接收数据；

C. 当 RF905 检测到同一频段的载波时，载波检测引脚被置高；

D. 当接收到一个相匹配的地址，AM引脚被置高；

E. 当一个正确的数据包接收完毕， RF905自动移去字头、地址和CRC校验位，然后把DR引脚置高

F. 微控制器把 TRX_CE 置低，nRF905 进入空闲模式；

G. 微控制器通过SPI口，以一定的速率把数据移到微控制器内；

H. 当所有的数据接收完毕，nRF905把DR引脚和AM引脚置低；

I. nRF905此时可以进入ShockBurstTM接收模式、ShockBurstTM发送模式或关机模式。

当正在接收一个数据包时，TRX_CE或TX_EN引脚的状态发生改变，RF905立即把其工作模式改变，数据包则丢失。当微处理器接到AM引脚的信号之后， 其就知道RF905正在接收数据包，其可以决定是让RF905继续接收该数据包还是进入另一个工作模式。

4.1.3 节能模式

RF905的节能模式包括关机模式和节能模式。

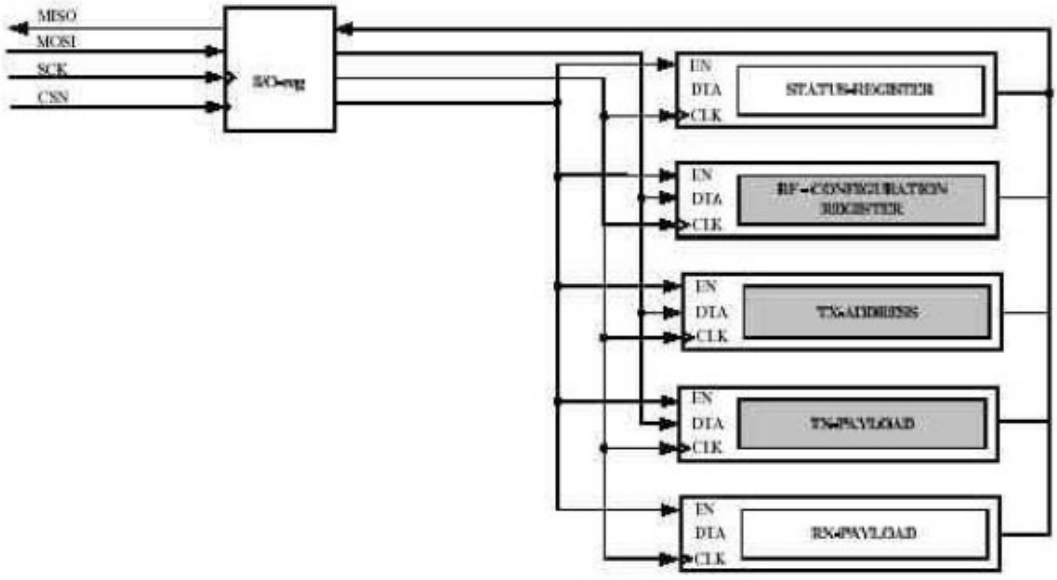
在关机模式，RF905 的工作电流最小，一般为 2.5uA。进入关机模式后，RF905 保持配置字中的内容，但不会接收或发送任何数据。 空闲模式有利于减小工作电流，其从空闲模式到发送模式或接收模式的启动时间也比较短。在空闲模式下，RF905 内部的部分晶体振荡器处于工作状态。

五、配置JL-RF905模块

所有配置字都是通过SPI接口送给RF905。SIP接口的工作方式可通过SPI指令进行设置。当RF905处于空闲模式或关机模式时，SPI接口可以保持在工作状态。

5.1 SPI 接口寄存器配置

SPI接口由状态寄存器、射频配置寄存器、发送地址寄存器、发送数据寄存器和接收数据寄存器5个寄存器组成。状态寄存器包含数据准备好引脚状态信息和地址匹配引脚状态信息；射频配置寄存器包含收发器配置信息，如频率和输出功能等；发送地址寄存器包含接收机的地址和数据的字节数；发送数据寄存器包含待发送的数据包的信息，如字节数等；接收数据寄存器包含要接收的数据的字节数等信息。SPI 接口由5 个内部寄存器组成执行寄存器的回读模式来确认寄存器的内容



SPI 接口和 5 个内部寄存器

状态寄存器Status-Register

寄存器包含数据就绪DR 和地址匹配AM 状态

RF 配置寄存器RF-Configuration Register

寄存器包含收发器的频率,输出功率等配置信息

发送地址TX-Address

寄存器包含目标器件地址字节长度由配置寄存器设置

发送有效数据TX-Payload

寄存器包含发送的有效ShockBurst 数据包数据字节长度由配置寄存器设置

接收有效数据TX-Payload

寄存器包含接收到的有效 ShockBurst 数据包数据字节长度由配置寄存器设置在寄存器中的有效数据由数据准备就绪 DR 指示

5.2 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:

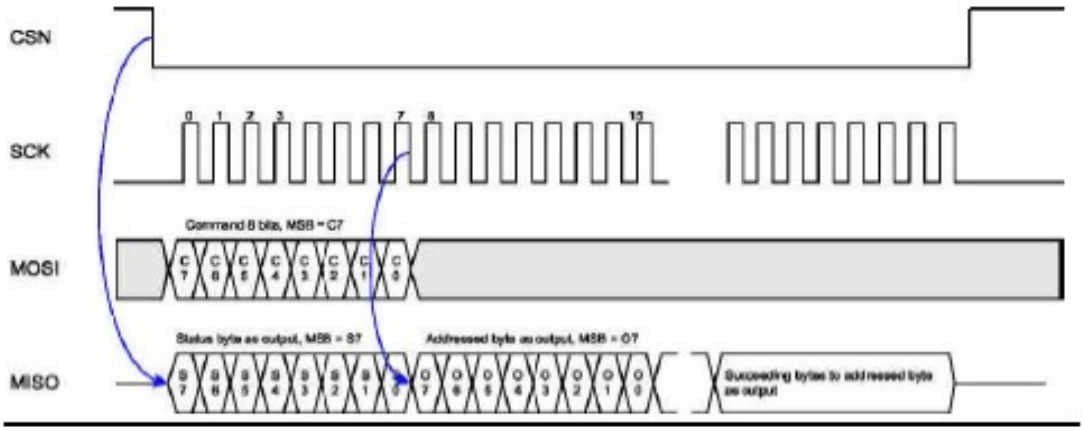
注意：请认真阅读一下一系列表格内容介绍，是编程关键

SPI 串行接口指令设置

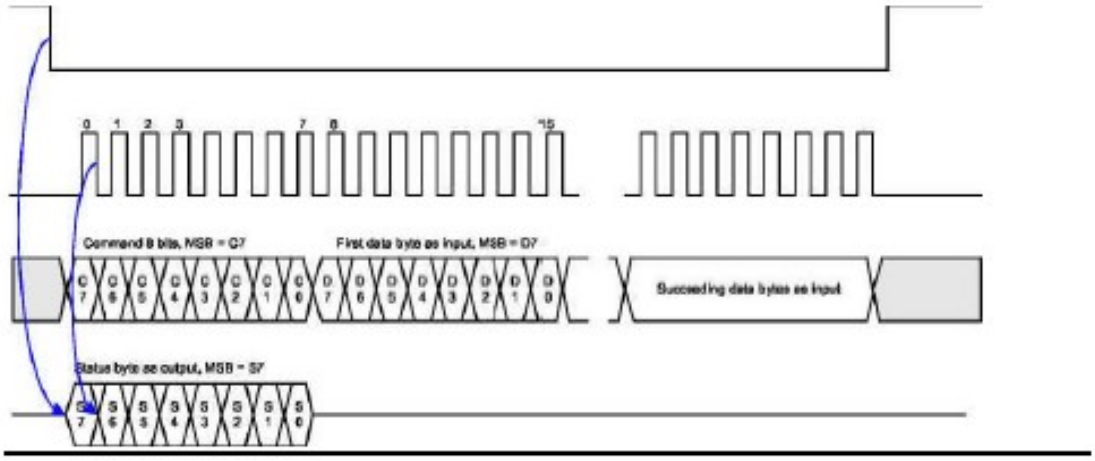
SPI 串行接口指令		
指令名称	指令格式	操作
W_CONFIG (WC)	0000AAAA	写配置寄存器AAAA 指出写操作的开始字节字节数量取决于AAAA 指出的开始地址
R_CONFIG (RC)	0001AAAA	读配置寄存器AAAA 指出读操作的开始字节字节数量取决于AAAA 指出的开始地址
W_TX_PAYLOAD (WTP)	00100000	写TX 有效数据1-32 字节写操作全部从字节0 开始
R_TX_PAYLOAD (RTP)	00100001	读TX 有效数据1-32 字节读操作全部从字节0 开始
W_TX_ADDRESS (WTA)	00100010	写TX 地址1-4 字节写操作全部从字节0 开始
R_TX_ADDRESS (RTA)	00100011	读TX 地址1-4 字节读操作全部从字节0 开始
R_RX_PAYLOAD (RRP)	00100100	读RX 有效数据1-32 字节读操作全部从字节0 开始
CHANNEL_CONFIG (CC)	1000pphccccccccc	快速设置配置寄存器中CH_NO HFREQ_PLL 和PA_PWR 的专用命令CH_NO=ccccccccc HFREQ_PLL=h PA_PWR=pp

5.3 SPI 时序

SPI 读操作



SPI 写操作



5.4 配置寄存器RF-Configuration-Register 说明

参数	位宽	说明
CH_NO	9	同HFREQ_PLL 一起设置中心频率默认值=001101100b=108d FRF= 422.4+ CH_NOd/10 *(1+ HFREQ_PLLd)MHZ
HFREQ_ PLL	1	设置PLL 在433 或868/915MHZ 模式默认值=0 0 -器件工作在433MHZ 频段1 -器件工作在868/915MHZ 频段
PA_PWR	2	输出功率默认值=00 00 -10dBm 01 -2dBm 10 +6dBm 11 +10dBm
RX_RED_ PWR	1	降低接收模式电流消耗至1.6mA 灵敏度降低默认值=0 0 -正常模式1 -低功耗模式
AUTO_ RETRAN	1	重发数据如果TX 寄存器的TRX_CE 和TX_EN 被设置为高默认值=0 0 -不重发数据1 -重发数据包
RX_AWF	3	RX 地址宽度默认值=100 001 -1 字节RX 地址宽度100 -4 字节RX 地址宽度
TX_AWF	3	TX 地址宽度默认值=100 001 -1 字节TX 地址宽度100 -4 字节TX 地址宽度
RX_PW	6	RX 接收有效数据宽度默认值=100000 000001 -1 字节RX 有效数据宽度000010 -2 字节RX 有效数据宽度100000 -32 字节RX 有效数据宽度
TX_PW	6	TX 有效数据宽度默认值=100000 000001 -1 字节TX 有效数据宽度000010 -2 字节TX 有效数据宽度100000 -32 字节TX 有效数据宽度
RX_ ADDRESS	32	RX 地址使用字节依赖于RX_AFW 默认值=E7E7E7E7h
UP_CLK_ FREQ	2	输出时钟频率默认值=11 00 -4MHZ 01 -2MHZ 10 -1MHZ 11 -500KHZ

UP_CLK_EN	1	输出时钟使能默认值=1 0 -没有外部时钟1 -外部时钟信号使能
XOF	3	晶体振荡器频率必须依据外部晶体的标称频率设置默认值=100 000 -4MHZ 001 -8MHZ 010 -12MHZ 011 -16MHZ 100 -20MHZ
CRC_EN	1	CRC 校验允许默认值=1 0 -不允许1 -允许
CRC_ MODE	1	CRC 模式默认值=1 0 -8 位CRC 校验位1 -16 位CRC 校验位

5.5 配置寄存器内容

RF-Configuration-Register(R/W)		
字节#	内容位[7 0] MSB=BIT[7]	初始化值
0	Bit[7 0]	0110_1100
1	Bit[7:6]没用AUTO_RETRAN RX_RED_PWR PA_PWR[1:0] HFREQ_PLL CH_NO[8]	0000_0000
2	Bit[7] 没用TX_AFW[2:0] Bit[3] 没用 RX_AFW[2:0]	0100_0100
3	Bit[7:6]没用RX_PWR[5:0]	0010_0000
4	Bit[7:6]没用TX_PWR[5:0]	0010_0000
5	RX 地址0 字节	E7
6	RX 地址1 字节	E7
7	RX 地址2 字节	E7
8	RX 地址3 字节	E7
9	CRC_ 模式 CRC 校验允许X OF[2:0] UP_CLK_EN UP_CLK_FREQ[1:0]	1110_0111

TX_PAYLOAD(R/W)		
字节#	内容位[7 0] MSB=BIT[7]	初始化值
0	TX_PAYLOAD[7:0]	X
1	TX_PAYLOAD[15:8]	X
		X
		X
30	TX_PAYLOAD[247:240]	X
31	TX_PAYLOAD[255:248]	X

TX_ADDRESS(R/W)		
字节#	内容位[7 0] MSB=BIT[7]	初始化值
0	TX_ADDRESS[7:0]	E7
1	TX_ ADDRESS [15:8]	E7
2	TX_ ADDRESS [23:16]	E7
3	TX_ ADDRESS [31:24]	E7

RX_PAYLOAD(R)		
字节#	内容位[7 0] MSB=BIT[7]	初始化值
0	RX_PAYLOAD[7:0]	X
1	RX_PAYLOAD[15:8]	X
		X
		X
30	RX_PAYLOAD[247:240]	X
31	RX_PAYLOAD[255:248]	X

STATUS_REGISTER(R)		
字节#	内容位[7 0] MSB=BIT[7]	初始化值
0	AM bit[6] 没用DR bit[4:0] 没用	E7

注意：射频寄存器的各位的长度是固定的。然而，在 ShockBurstTM 收发过程中，TX_PAYLOAD、RX_PAYLOAD、TX_ADDRESS 和 RX_ADDRESS 4 个寄存器使用字节数由配置字决定。RF905 进入关机模式或空闲模式时，寄存器中的内容保持不变。

六、JL-RF905 编程指南以及代码

使用 JL-RF905 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，请您直接联系客服 QQ：363423117，我们会尽快回复您。

RF-Configuration-Register(R/W)		
字节#	内容位[7：0]，MSB=BIT[7]	初始化值
0	Bit[7：0] CH NO[7:0]	0110_1100
1	Bit[7:6]没用，AUTO_RETRAN，RX_RED_PWR，PA_PWR[1:0]，HFREQ_PLL，CH_NO[8]	0000_0000
2	Bit[7] 没用，TX_AFW[2:0]， Bit[3] 没用， RX_AFW[2:0]	0100_0100
3	Bit[7:6]没用，RX_PWR[5:0]	0010_0000
4	Bit[7:6]没用，TX_PWR[5:0]	0010_0000
5	RX 地址 0 字节	E7
6	RX 地址 1 字节	E7
7	RX 地址 2 字节	E7
8	RX 地址 3 字节	E7
9	CRC_模式， CRC 校验允许，XOF[2:0]， UP_CLK_EN，UP_CLK_FREQ[1:0]	1110_0111

字节0:

[7:0] CH_NO[7:0]:

连同字节1的CH_NO[8]和HFREQ_PLL控制905的载波频段

参考设置:

Operating frequency HFREQ_PLL CH_NO

430.0 MHz [0] [001001100]

433.1 MHz [0] [001101011]

433.2 MHz [0] [001101100]

434.7 MHz [0] [001111011]

862.0 MHz [1] [001010110]

868.2 MHz [1] [001110101]

868.4 MHz [1] [001110110]

869.8 MHz [1] [001111101]

902.2 MHz [1] [100011111]

902.4 MHz [1] [100100000]

927.8 MHz [1] [110011111]

载波频率的计算公式:

$$f_{OP} = (422.4 + (CH\_NO/10)) \cdot (1 + HFREQ\_PLL)\ MHz$$

字节1:

[0] CH_NO [8] : 参见字节0

[1] HFREQ_PLL :

0 - 器件工作在433MHZ频段

1 - 期间工作在868/915MHZ频段

[3:2] PA_PWR :

输出功率

00 -10dBm （默认）

01 -2dBm

10 +6dBm

11 +10dBm

[4] RX_RED_PWR :

降低接收模式电流消耗至1.6mA，灵敏度降低。

0 - 正常模式 （默认）

1 - 低功耗模式

[5] AUTO_RETRAN:

自动重发TX寄存器中的数据包，如果TRX_CE和TX_EN被设置为高。

0 – 不重发数据包 （默认）

1 – 自动重发数据包

[7:6] 保留

字节2

[2:0] RX_AWF [2:0] :

RX 地址宽度

001 – 1字节RX地址宽度 （默认）

100 – 4字节RX地址宽度

[3] 保留

[6:4] TX_AWF [2:0] :

TX地址宽度

001 – 1字节TX地址宽度

100 – 4 字节TX地址宽度

[7] 保留

字节3

[5:0] RX_PW [5:0] :

RX接收有效数据宽度

000001 – 1字节RX有效数据宽度

000010 – 2字节RX有效数据宽度

……

100000 – 32字节RX有效数据宽度

[7:6] 保留

字节4

[5:0] TX_PW [5:0] :

TX发送有效数据宽度

000001 – 1字节TX有效数据宽度

000010 – 2 字节 TX 有效数据宽度

……

100000 – 32字节TX有效数据宽度

[7:6] 保留

字节5 : RX地址0字节

字节6 : RX地址1字节

字节7 : RX地址2字节

字节8 : RX地址3字节

字节 9

[1:0] UP_CLK_FREQ [1:0]:

输出时钟频率

00 – 4MHZ

01 – 2MHZ

10 – 1MHZ

11 – 500KHZ

[2] UP_CLK_EN :

输出时钟使能

0 - 没有外部时钟

1 - 外部时钟信号使能 （默认）

[5:3] XOF [2:0] :

晶体振荡器频率，必须依据外部晶体的标称频率设置

(无线模块上 905 芯片外接晶振的频率)

000 - 4MHZ

001 - 8MHZ

010 - 12MHZ

011 - 16MHZ（模块标配为16MHZ）

100 - 20MHZ

[6] CRC_EN :

CRC校验允许

0 - 部允许

1 - 允许 （默认）

[7] CRC_MODE :

CRC模式

0 - 8位CRC校验位

1 -16位CRC校验位 （默认）

范例程序中的相关代码段:

```
/*nRF905寄存器配置参数*/
typedef struct RFConfig
```

```
{
  uchar n;
  uchar buf[10];
}RFConfig;
code RFConfig RxTxConf =
{
  10,
  0x4c, 0x0c, 0x44, 0x20, 0x20, 0xcc, 0xcc, 0xcc,0xcc, 0x58
};
```

//buf[10] 中数据对应 字节0 ~ 字节9 ，具体内容可参考上文寄存器配置章节

注：对于频段设置参数CH_NO，在我们提供的范例程序中CH_NO[7:0]的值为0x4c。我们不建议各位用户使用其他数值，因为我们的模块在硬件上只适应430MHz左右的频率，为了达到最好的效果，软件参数上应当与硬件匹配，否则会影响通讯距离。

6.2 [通过SPI接口向nRF905 配置寄存器读写配置信息]

nRF905通过SPI接口与单片机通讯，因此必须首先了解SPI接口。

[SPI概念] SPI外围串行接口由四条线构成:

MOSI主机输出从机输入 （主机写操作）

MISO主机输入从机输出 （主机读操作）

SCK 串行时钟信号，由主机控制

CSN 片选信号，低电平有效

```
//<SPI写操作 代码>
void SpiWrite(uchar byte)
{
    uchar i;
    DATA_BUF=byte; // 将需要发送的数据写入缓存
    for (i=0;i<8;i++) // 循环8次发送一个字节的数据
    {
        if (flag) // flag = DATA_BUF^7;
        MOSI=1;
        else
        MOSI=0;
        SCK=1; // SCK 高电平
        DATA_BUF=DATA_BUF<<1; // 左移一位,为下一位的发送做准备
        SCK=0; // SCK 低电平
    }
}
```

步骤一: MOSI线准备好需要发送的数据位

步骤二: SCK置高, 器件读取MOSI线上的数据

步骤三: SCK置低, 准备发送数据的下一位

以上步骤循环执行8次, 通过SPI向器件发送数据完成!

注意: 数据的传输时, 高位在前, 低位在后。

```
//<SPI读操作 代码>
```

```
uchar SpiRead(void)
{
    uchar i;
    for (i=0;i<8;i++) //循环8次发送一个字节的数据
    {
        DATA_BUF=DATA_BUF<<1; //左移一位,准备接收下一位数据
        SCK=1; // SCK 高电平
        if (MISO)
        flag1=1; // flag1 = DATA_BUF^0;
        else
        flag1=0;
        SCK=0; // SCK低电平
    }
    return DATA_BUF; // DATA_BUF 为接收到的完整数据
}
```

步骤一：MISO线准备好需要发送的数据位

步骤二：SCK置高，主机读取MISO线上的数据

步骤三：SCK置低，准备接收数据的下一位

以上步骤循环执行8次，通过SPI从器件上读数据完成！

注意：数据的传输时，高位在前，低位在后。

```
//<主机通过SPI接口向905配置寄存器写入信息>
void Config905(void)
```

```
{
    uchar i;
    CSN=0; // CSN片选信号，SPI使能
    SpiWrite(WC); // 向905芯片写配置命令
    for (i=0;i<RxTxConf.n;i++) // 循环写入配置信息
    {
        SpiWrite(RxTxConf.buf[i]); //RxTxConf保存预先设置
                                    好的配置信息
    }
    CSN=1; // 结束SPI数据传输
}
```

步骤一：CSN置低电平，SPI接口开始等待第一条指令

步骤二：调用SpiWrite函数，向器件发送WC信号，准备写入配置信息（SpiWrite函数在上文讲解）

步骤三：反复调用SpiWrite函数，向器件配置寄存器写入配置信息

步骤四：CSN置高电平，结束SPI通讯。

nRF905 配置完成!

代码中nRF905 SPI接口指令的宏定义

//（以下操作全部从对应寄存器的字节0开始）

```
#define WC 0x00
// 写配置寄存器（RF-Configuration Register）
#define RC 0x10
// 读配置寄存器（RF-Configuration Register）
#define WTP 0x20
// 向TX-Payload寄存器写入发送有效数据
#define RTP 0x21
// 从TX-Payload寄存器读取发送有效数据
#define WTA 0x22 // 向TX-Address寄存器写入发送地址
#define RTA 0x23 // 从TX-Address寄存器读取发送地址
#define RRP 0x24
// 从RX-Payload寄存器读取接收到的有效数据
//使用nRF905发送数据
void TxPacket(void)
{
   uchar i;
   CSN=0;
   SpiWrite(WTP); // Write payload command
   for (i=0;i<32;i++)
   {
   SpiWrite(TxBuf[i]); // 写入32直接发送数据
   }
```

```
    CSN=1; // 关闭SPI,保存写入的数据
    Delay(1);
    CSN=0; // SPI使能,准备写入地址信息
    SpiWrite(WTA); // 写数据至地址寄存器
    for (i=0;i<4;i++) // 写入4字节地址
    {
        SpiWrite(RxTxConf.buf[i+5]);
    }
    CSN=1; // 关闭SPI
    TRX_CE=1; // 进入发送模式,启动射频发送
    Delay(1); // 进入ShockBurst发送模式后,芯片保证数据发送完
    成后返回STANDBY模式
    TRX_CE=0;
}
```

步骤一：通过SpiWrite 函数发送WTP命令，准备写入TX有效数据

步骤二：循环调用SpiWrite向TX-Payload寄存器写入TX有效数据

（中间夹有CSN电平变化）

步骤三：延时

步骤四：通过SpiWrite函数发送WTA命令，准备写入TX地址

步骤五：循环调用SpiWrite向TX-Address寄存器写入TX地址

步骤六：TRX_CE=1；开始发送数据

延时，nRF905数据发送完成

当nRF905接收到一条完成的信息时，会将DR引脚置高。

//这段代码和范例中提供的有所不同，做了较大的简化，只留下必要的部分

```
void RxPacket(void)
{
  uchar i;
  TRX_CE=0; // 设置905进入待机模式
  CSN=0; // 使能SPI
  SpiWrite(RRP); // 准备读取接收到的数据
  for (i=0;i<32;i++)
  {
      RxBuf[i]=SpiRead(); // 通过SPI接口从905芯片读取数据
  }
  CSN=1; // 禁用SPI
  while(DR||AM);
  TRX_CE=1;
}
```

步骤一：TRX_CE=0; 必须将此引脚置低，使905进入standby模式

步骤二：发送RRP指令

步骤三：循环调用SpiRead函数，读取接收到的数据

步骤四：等待DR和AM引脚复位为低电平

（中间夹有CSN电平变化）

数据包接收完成!

注:

AM 地址匹配，接收到有效地址，被置高

DR 接收到有效数据包，并解码后，被置高，当所有有效数据被读取后，nRF905将AM和DR置低

最后需要注意的是，必须首先设置器件的发送/接收模式才能保证有效的数据发送接收

```
//<设置器件为发送模式>
void SetTxMode(void) //<设置器件为接收模式>
{
    TX_EN=1;
    TRX_CE=0;
    Delay(1); // delay for mode change(>=650us)
}
void SetRxMode(void)
{
    TX_EN=0;
    TRX_CE=1;
    Delay(1); // delay for mode change(>=650us)
}
```

6.3[SPI 接口相关数据]

PARAMETER	SYMBOL	MIN	MAX	UNITS
Data to SCK Setup	Tdc	5		ns
SCK to Data Hold	Tdh	5		ns
CSN to Data Valid	Tcsd		45	ns
SCK to Data Valid	Tcd		45	ns
SCK Low Time	Tcl	40		ns
SCK High Time	Tch	40		ns
SCK Frequency	Tsck	DC	10	MHz
SCK Rise and Fall	Tr,Tf		100	ns
CSN to SCK Setup	Tcc	5		ns
SCK to CSN Hold	Tcch	5		ns
CSN Inactive time	Tcwh	500		ns
CSN to Output High Z	Tcdz		45	ns

Table 8 SPI timing parameters (C_{load} = 10pF).

在使用高性能的单片机作为nRF905的主机时需要考虑这个表格中的相关数据。

6.4[器件模式切换时间]

nRF905 timing	Max.
PWR_DWN ➔ ST_BY mode	3 ms
STBY ➔ TX ShockBurst™	650 µs
STBY ➔ RX ShockBurst™	650 µs
RX ShockBurst™ ➔ TX ShockBurst™	550 [1]µs
TX ShockBurst™ ➔ RX ShockBurst™	550 [1]µs

以下是以 RF51 测试小板为基础的 NRF905 例程：

```
/**************************************************************************
1. 将改程序下载到 2 个开发板
2. 然后按 KEY1-KEY2 会显示对应信息，而且两个开发板都可以做为发送和接收，一个做
   为发送，对应另一即为接收
**************************************************************************/
#include <reg52.h>
#include <ABSACC.h>
#include <intrins.h>
#include <stdio.h>
//-----------------------------------------------------------------------------------------------------------
#define uint unsigned int
#define uchar unsigned char
//-----------------------------------------------------------------------------------------------------------
#define BYTE_BIT0      0x01
#define BYTE_BIT1      0x02
#define BYTE_BIT2      0x04
#define BYTE_BIT3      0x08
#define BYTE_BIT4      0x10
#define BYTE_BIT5      0x20
#define BYTE_BIT6      0x40
#define BYTE_BIT7      0x80
//-----------------------------------------------------------------------------------------------------------
bdata unsigned    char DATA_BUF;
#define DATA7      ((DATA_BUF&BYTE_BIT7) != 0)
#define DATA0      ((DATA_BUF&BYTE_BIT0) != 0)
sbit  flag  =DATA_BUF^7;
sbit  flag1       =DATA_BUF^0;
//----------------------------------发送数据缓冲区-------------------------------------------
#define TxRxBuf_Len 4
unsigned char TxRxBuf[TxRxBuf_Len]=
{
0x29,0x30,0x31,0x32,
};
//-------------------------------------------------配置口定义-------------------------------------------
//配置口定义//
//配置口定义//
sbit  TXEN=P1^6;
sbit  TRX_CE=P1^7;
sbit  PWR=P2^0;
sbit  MISO=P3^0;
sbit  MOSI=P3^1;
```

```
sbit  SCK=P3^2;
sbit  CSN=P3^4;
////////////////////////////////////////////////////
sbit  AM=P2^3;
sbit  DR=P3^3;
sbit  CD=P2^2;
//-----------------------------------------------------按键-----------------------------------------------

sbit      KEY0      =P3^0;
sbit      KEY1      =P3^1;
//---------------------------------------数码管位控制-----------------------------------------------------
sbit      led0      =P1^0;
sbit      led1      =P1^1;
//-------------------------------------------------------------------------------------------------------
uchar seg[10]={0xC0,0xCF,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90};          //0~~9段码
//-----------------------------------------------nrf905 控制指令-----------------------------------------
#define WC          0x00
#define RC          0x10
#define WTP         0x20
#define RTP         0x21
#define WTA         0x22
#define RTA         0x23
#define RRP         0x24
//-----------------------------------------NRF905 寄存器配置-----------------------------------------------
unsigned char idata RFConf[11]=
{
   0x00,                                  //配置命令//
   0x4c,                                  //CH_NO,配置频段在 430MHZ
   0x0c,                                  //输出功率为 10db,不重发，节电为正常模式
   0x44,                                  //地址宽度设置，为 4 字节
   0x04,0x04,                             //接收发送有效数据长度为 32 字节
   0xCC,0xCC,0xCC,0xCC,                   //接收地址
   0x58,                                  //CRC 充许，8 位 CRC 校验，外部时钟信号不使能，16M
晶振
};
code TxAddress[4]={0xcc,0xcc,0xcc,0xcc};
char tf;
//-------------------------------------------延时-----------------------------------------------------
static void Delay(uchar n)
{
      uint i;
      while(n--)
      for(i=0;i<80;i++);
}
```

```
//-----------------------------------------------SPI 读函数-----------------------------------------------------
unsigned char SpiRead(void)
{
    unsigned char j;
    for (j=0;j<8;j++)
    {
        DATA_BUF=DATA_BUF<<1;
        SCK=1;
        if (MISO)                   //读取最高位，保存至最末尾，通过左移位完成整个字节
        {
            DATA_BUF|=BYTE_BIT0;
        }
        else
        {
            DATA_BUF&=~BYTE_BIT0;
        }
        SCK=0;
     }
     return DATA_BUF;
}
//-----------------------------------------------SPI 写函数-----------------------------------------------------
void SpiWrite(unsigned char send)
{
    unsigned char i;
    DATA_BUF=send;
    for (i=0;i<8;i++)
    {
        if (DATA7)                  //总是发送最高位
        {
            MOSI=1;
        }
        else
        {
            MOSI=0;
        }
        SCK=1;
        DATA_BUF=DATA_BUF<<1;
        SCK=0;
    }
}
//-----------------------------------------------初始化 nRF905---------------------------------------------
void nRF905Init(void)
{
    CSN=1;                          // Spi    disable
```

```
    SCK=0;                      // Spi clock line init low
    DR=0;                       // Init DR for input
    AM=0;                       // Init AM for input
    CD=0;                       // Init CD for input
    PWR=1;                      // nRF905 power on
    TRX_CE=0;                   // Set nRF905 in standby mode
    TXEN=0;                     // set radio in Rx mode
}
//-----------------------------------------------------初始化寄存器-----------------------------------------------
void Config905(void)
{
    uchar i;
    CSN=0;                      // Spi enable for write a spi command
    //SpiWrite(WC);             // Write config command 写放配置命令
    for (i=0;i<11;i++)          // Write configration words   写放配置字
    {
        SpiWrite(RFConf[i]);
    }
    CSN=1;                      // Disable Spi
}
//-------------------------------------------发送数据打包-------------------------------------------------
void TxPacket(uchar *TxRxBuf)
{
    uchar i;
    //Config905();
    CSN=0;
    SpiWrite(WTP);              // Write payload command
    for (i=0;i<4;i++)
    {
        SpiWrite(TxRxBuf[i]);   // Write 32 bytes Tx data
    }// Spi enable for write a spi command
    CSN=1;
    Delay(1);                   // Spi disable
    CSN=0;                      // Spi enable for write a spi command
    SpiWrite(WTA);              // Write address command
    for (i=0;i<4;i++)           // Write 4 bytes address
    {
        SpiWrite(TxAddress[i]);
    }
    CSN=1;                      // Spi disable
    TRX_CE=1;                   // Set TRX_CE high,start Tx data transmission
    Delay(1);                   // while (DR!=1);
    TRX_CE=0;                   // Set TRX_CE low
}
```

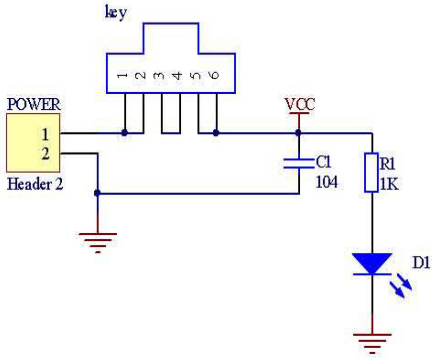
```
//-----------------------------------------------设置发送状态-----------------------------------------
void SetTxMode(void)
{
    TRX_CE=0;
    TXEN=1;
    Delay(1);                       // delay for mode change(>=650us)
}
//-----------------------------------------------设置发送状态--------------------------
void SetRxMode(void)
{
    TXEN=0;
    TRX_CE=1;
    Delay(1);                       // delay for mode change(>=650us)
}
//---------------------------------------判断数据接收状态--------------------------------
unsigned char CheckDR(void)         //检查是否有新数据传入  Data Ready
{
    if (DR=1&&TRX_CE==1 && TXEN==0)
    {
       // Delay(50)    ;
        return 1;
    }
    else
    {
        return 0;
    }
}
//-----------------------------读 NRF905 接收数据-----------------------------------------------
void RxPacket(void)
{
    uchar i;
    Delay(1);
//  TRX_CE=0;                       // Set nRF905 in standby mode
    Delay(100);
    TRX_CE=0;
    CSN=0;                          // Spi enable for write a spi command
    Delay(1);
    SpiWrite(RRP);
    for (i = 0 ;i < 4 ;i++)
    {
        TxRxBuf[i]=SpiRead();       // Read data and save to buffer
    }
    CSN=1;
    Delay(10);
```

```
    TRX_CE=1;
}
//-----------------------------------------------------数据接收-----------------------------------------------
void    RX(void)
{
            SetRxMode();                // Set nRF905 in Rx mode
            while (CheckDR()==0);
            Delay(10);
            RxPacket();
            if(TxRxBuf[0]==0x29)
            {
                    led0=0;
                    led1=1;

            }
            if(TxRxBuf[1]==0x29)
            {
                    led0=1;
                    led1=0;

            }
}
//-------------------------------------------------------------------------------------------------------------
void main(void)
{
        nRF905Init();
        Config905();
        led0=0;
        led1=0;

        while(1)
        {
            RX();
          if(KEY0 ==0 )
            {
                tf = 1 ;
                TxRxBuf[0]=0x29;
                TxRxBuf[1]=0x30;

                led0=0;
                led1=1;

                  }
          if(KEY1 ==0 )
```

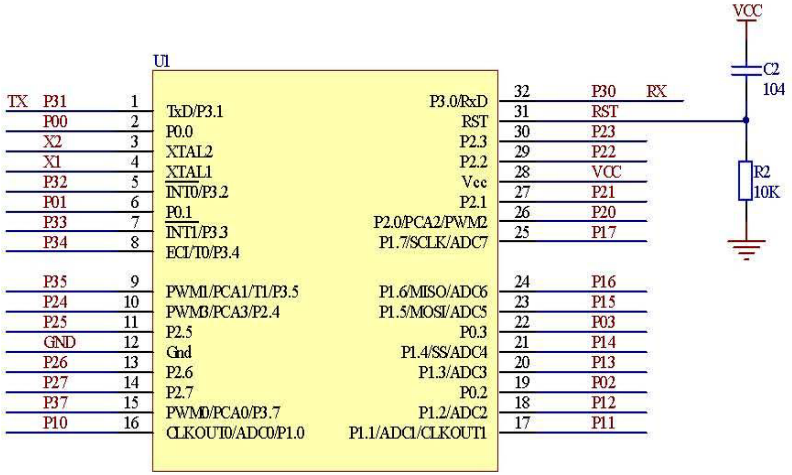
```
                {   tf = 1 ;
                    TxRxBuf[0]=0x30;
                    TxRxBuf[1]=0x29;

                    led0=1;
                    led1=0;

                    }
              if (tf==1)
               {
                    SetTxMode();// Set nRF905 in Tx mode
                    TxPacket(TxRxBuf);// Send data by nRF905
                    tf = 0;
                    Delay(100);
               }
          Delay(100);
          led1=1;
          led0=1;
          Delay(100);
     }
}
```

七、测试小板原理图

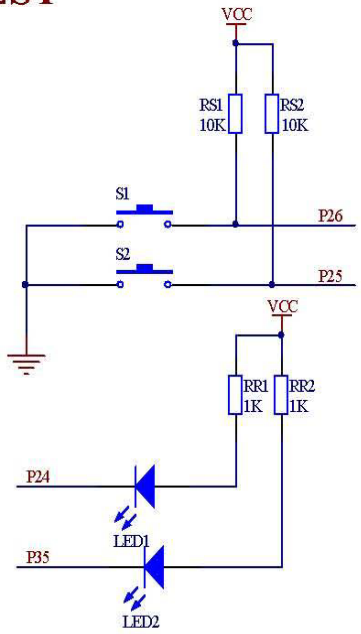
POWER



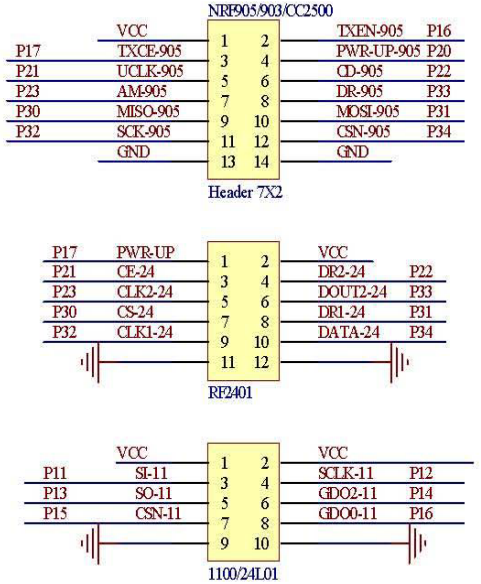
IC



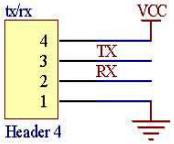
TEST



RF PORT



DWONLOAD



八、备注:

杭州金龙电子有限公司主要经营范围:

☑ 300M-2.4G 无线数传模块研发以及生产 ;

☑ 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、锂铁鳞电池保护板;

☑ 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

QQ: 363423117