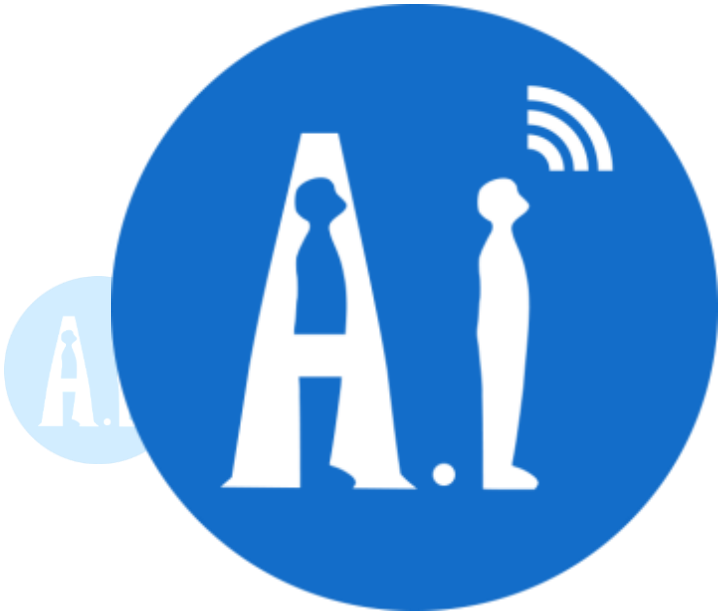
2.4G 系列模组常见问题



版本 V1.0

关于本文档：

1. 本文档收集安信可 2.4G 系列模组的常见软硬件问题。
2. 本文档为初版，后期持续更新。

版本日志：

版本	日期	制定/修订内容	制定	核准
V1.0	2019.10.29	首次发布	junx	徐宏

目录

1. 硬件篇

1.1. 问:NF-01-S、NF-01-N、NF-02-SM、NF-02-PA、NF-02-PE、NF-03、NF-04、NF-04-MI 分别用的什么芯片?

答:2.4G 模组系列采用了 nRF24L01+、SI24R1 和 BK2425 三种芯片方案。

1)NF-01-N、NF-02-PA 和 NF-02-PE 采用的是 nRF24L01+挪威芯片。

2)NF-01-S、NF-02-SM 和 NF-03 采用的是 SI24R1 台产芯片。

3)NF-04、NF-04-MI 采用的是 BK2425 博通芯片。

1.2. 问:同时使用了 2 种芯片的 2.4G 模组,他们软件是否兼容?

答:不同的芯片驱动有所不同,大致的配置内容是相同的。

1.3. 问:常见 2.4G 模组发热损坏的原因?

答:主要原因如下:

1)供电电源超过 3.6V。

2)电源纹波过大,不稳定。

3)静电。

1.4. 问:如何排查 2.4G 通讯距离过短的问题?

答:基本上从以下方面进行排查:

1)需考虑环境中的障碍物。

2)需考虑测试环境同频段干扰源。

3)如果是板载天线,要考虑在整机内位置摆放。
如果是外接天线,要考虑天线的性能是否合格。

4)软件上设置发射功率是否过低。

5)软件上设置发射速率是否过大。

6)供电电压是否过低,不能低于 1.8V。

1.5. 问:请问这种 3.3V 供电的 2.4G 模块,能否与 5V 供电的单片机串口信号线直接连接?

答:不能,需要使用 5V 转 3.3V 的 IO 电平转换电路。

1.6. 问:Si24R1 可以和 nRF24L01+互相通信且软件上兼容吗?

答:Si24R1 的工作参数与 nRF24L01+基本一致,仅在一个寄存器配置部分存在差别(发射功率配置),是可以互相通信的,但是要保证发射接收数据宽度相同(最大 32 个字节)、发射接收地址相同(5 个 8 位地址)、发射接收频道相同(0~125)、发射接收速率相同(2M 1M 250K)。

1.7. 问:产品外壳是金属的,如何选择 2.4G 模组?

答:考虑到金属外壳,建议采用外接天线,可以选用 NF-02-PE、NF-02-PA、NF-02-SM。

1.8. 问:需要远距离传输数据,贵司的哪些 2.4G 模组带有 PA?

答:NF-02-PA 等。

1.9. 问:进入 SI24R1 芯片的模块的低功耗(关断)模式后,为什么功耗偏大?

答:由于芯片采用 CMOS 工艺,当芯片处于关断模式时,芯片的数字输入引脚,CE,CSN,SCK,MOSI,必须为低电平,不能为高阻状态或高电平。否则由于输入端累积电荷,会导致内部电路不能关断,而使得功耗增加。

1.10. 问:把 SI24R1 芯片的模块设置为 7dbm 的最大发射功率的时候，距离好像没有增加太多？

答:当芯片发射功率大于 0dbm 以后，芯片底部的金属焊盘会有很多白噪声到地，电源处需要多加一个大电容去滤波。

1.11. 问:NF-01-S、NF-02-SM 和 NF-03 这三种模块工作频率范围是多少，有多少信道和多大的带宽？

答:这三个模块都使用的是 SI24R1 芯片，工作频率范围为 2400MHz-2525MHz，共有 126 个 1MHz 带宽的信道。

1.12. 问:2.4G 无线模组可以直接和串口连接并进行通讯吗？

答:不可以的，2.4G 无线模块不提供串行接口。如果需要与 PC 通过串口连接和通讯，请使用 MCU 作为主控，接收 PC 串口发送的数据并对将其传送至 2.4G 无线模块中并进行发送工作。另一端的 2.4G 无线模块接收数据后，由 MCU 读出数据并发送给计 PC 串口。

2. 软件篇

2.1. 问:NF-03 模组采用了什么数字调制与解调技术？

答:支持 GFSK/FSK 调制解调技术。

2.2. 问:NF-01-S、NF-02-SM 和 NF-03 模组的支持哪些数据传输速率？

答:支持 2Mbps. 1Mbps, 250Kbps 三种数据速率。

2.3. 问:NF-03 模组的工作模式是怎样的？

答: NF-03 模组可配置为关断模式、待机模式、发射空闲模式、发射模式和接收模式 5 种。

1） 关断模式：射频收发功能关闭，芯片停止工作，消耗电流最小，但所有内部寄存器值和 FIFO 值保持不变，仍可通过 SPI 实现对寄存器的读写。设置 CONFIG 寄存器的 PWR_UP 位的值为 0，芯片立即返回到关断工作模式。

2） 待机模式:只有晶体振荡器电路工作,保证了芯片在消耗较少电流的同时能够快速启动。设置 CONFIG 寄存器下的 PWR_UP 位的值为 1，芯片待时钟稳定后进入待机模式。芯片的时钟稳定时间一般为 1.5~2ms，与晶振的性能有关。当引脚 CE=1 时，芯片将由待机模式进入到发射空闲或接收模式，当 CE=0 时，芯片将由发射空闲、发射或接收模式返回到待机模式。

3） 发射空闲模式：晶体振荡器电路及时钟电路工作。相比于待机模式，芯片消耗更多的电流。当发送端 TX FIFO 寄存器为空，并且引脚 CE=1 时，芯片进入到发射空闲模式。在该模式下，如果有新的数据包被送到 TX FIFO 中，芯片内部的电路将立即启动，切换到发射模式将数据包发送。在待机和发射空闲工作模式下,所有内部寄存器值和 FIFO 值保持不变,仍可通过 SPI 实现对寄存器的读写。

4） 发射模式：当需要发送数据时，需要切换到发射工作模式。芯片进入到发射工作模式的条件为： TX FIFO 中有数据，CONFIG 寄存器的 PWR_UP 位的值为 1， PRIM_RX 位的值为 0，同时要求引脚 CE 上有一个至少持续 10us 的高脉冲。芯片不会直接由待机模式直接切换到发射模式，而是先立即切换到发射空闲模式，再由发射空闲模式自动切换到发射模式。发射空闲模式切换到发射模式的时间为 120us~130us 之间，但不会超过 130us。单包数据发送完成后，如果 CE=1，则由 TX FIFO 的状态来决定芯片所处的工作模式，当 TX FIFO 还有数据，芯片继续保持在发射工作模式，并发送下一包数据；当 TX FIFO 没有数据，芯片发射空闲模式；如果 CE=0，立即返回 Standby 模式。数据发射完成后，芯片产生数据发射完成中断。

5） 接收模式：当需要接收数据时，需要切换到接收工作模式。芯片进入到接收工作模式的条件为：设置寄存器 CONFIG 的 PWR_UP 位的值为 1， PRIM_RX 位的值为 1，并且引脚 CE=1。芯片由待机模

式切换到接收模式的时间为 120~130us。当接收到数据包的地址与芯片的地址相同，并且 CRC 检查正确时，数据会自动存入 RX FIFO，并产生数据接收中断。芯片最多可以同时存三个有效数据包，当 FIFO 已满，接收到的数据包被自动丢掉。在接收模式下，可以通过 RSSI 寄存器检测接收信号功率。当接收到的信号强度大于-60dBm 时， RSSI 寄存器的 RSSI 位的值将被设置为 1。否则， RSSI=0。RSSI 寄存器的更新方法有两种：当接收到有效的数据包后，RSSI 会自动更新，此外，将芯片从接收模式换到待机模式时 RSSI 也会自动更新。RSSI 的值会随温度的变化而变化，范围在± 5dBm 以内。

2.4. 问:SI24R1 芯片系列的 2.4G 模组的数据包处理协议是怎样的?

答:SI24R1 基于包通信，支持停等式 ARQ 协议。芯片内部 ARQ 协议基带处理引擎,可以不需要外部 MCU 干预下,自动实现 ACK 和 NO_ACK 数据包的处理。ARQ 协议基带处理单元支持 1 到 32 字节动态数据长度，数据长度在数据包内。也可以采用固定数据长度，通过寄存器指定；基带处理单元完成数据的自动解包、打包、自动回复 ACK 确认信号以及自动重发。该处理单元内部有 6 个通信管道，可以直接支持 1：6 星型网络。

2.5. 问:2.4G 系列模组的 SPI 接口是怎样实现的？

答:是用的标准四线 SPI 接口,读写速度最大值为 10Mb/s。外部 MCU 通过 SPI 接口对模组进行配置。只能在关断、待机和发射空闲的模式下才能对模组的寄存器操作。

2.6. 问:2.4G 系列模组主要的寄存器有哪些？

答:以 SI24R1 芯片系列的 2.4G 模组为例：

CONFIG(配置寄存器)：主要对发射和接收、关机和开机、CRC 长度和收发中断屏蔽控制等进行配置。
EN_AA(使能自动确认寄存器)：主要使能数据管道自动确认。
EN_RXADDR(使能接收数据管道寄存器)：主要使能数据管道接收。
SETUP_AW(地址宽度配置寄存器)：主要配置发射和接收的地址宽度。
SETUP_RETR(自动重发配置寄存器)：主要配置自动重发的延时（250~4000us）和最大次数（1-15）。
RF_CH(射频信道寄存器)：主要设置工作信道，1~125 个信道，间隔为 1MHz。
RF_SETUP(射频配置寄存器)：
1） 设置射频数据数率，250Kbps\1Mbps\2Mbps 三档。
2） 设置发送发射功率，-12~7dBm。
STATUS(状态寄存器)：SPI 操作开始，状态寄存器值通过 MISO 串行输出。
OBSERVE_TX(发射结果统计寄存器):统计丢包和重发次数。
RSSI(接收信号强度寄存器)：读取 RSSI 值。
FIFO_STATUS(FIFO 状态寄存器)：判断收发 FIFO 的状态。
DYNPD（使能动态负载寄存器）：使能接收管道的动态负载长度。
FEATURE(特征寄存器)：主要使能动态负载长度和 ACK 负载。

2.7. 问:NO ACK 配置方案是怎样的？

答: 用 W_TX_PAYLOAD 命令对发送端写 TX PAYLOAD 时，数据包中的 NO_ACK 标志位置位，发送端发送完一包数据后，立即产生 TX_DS 中断，且准备发送下一包数据。接收端收到数据后判断 NO_ACK 标志位复位，且数据有效，则产生 RX_DR 中断，通信完成不回复 ACK 信号。

1)发射方配置：

```
SPI_WRITE_BUF(TX_ADDR,TX_ADDRESS,5); // 写入发送地址
SPI_RW_REG(FEATURE,0x01); // 使能 W_TX_PAYLOAD_NOACK 命令
SPI_WRITE_BUF(W_TX_PAYLOAD_NOACK,buf,TX_PLOAD_WIDTH); // 写 FIFO
SPI_RW_REG(SETUP_AW,0x03); // 5 byte Address width
```

```
SPI_RW_REG(RF_CH,0x40); // 选择射频通道 0x40
SPI_RW_REG(RF_SETUP,0x08); // 数据传输率 2Mbps
SPI_RW_REG(CONFIG,0x0e); //配置为发射模式、 CRC 为 2Bytes
CE = 1;
```

2)接收方配置：

```
SPI_WRITE_BUF(RX_ADDR_P0,TX_ADDRESS,5); // 接收地址
SPI_RW_REG(EN_RXADDR,0x01); // 使能接收通道 0Preliminary Si24R1
SPI_RW_REG(RF_CH,0x40); // 选择射频信道
SPI_RW_REG(RX_PW_P0,TX_PLOAD_WIDTH); //设置接收通道 0 负载数据宽度
SPI_RW_REG(RF_SETUP,0x08); // 数据传输率 2Mbps,-18dbm TX power
SPI_RW_REG(CONFIG,0x0f); // 配置为接收方、 CRC 为 2Bytes
```

2.8. 问:2.4 模组收发采用的是 ACK 通信方式，为什么发送端没有收到 ACK 信号？

答:发送端需要设置接收管道地址与自身发送的地址相同。

2.9. 问:ACK 配置方案是怎样的？

答:用 W_TX_PAYLOAD 命令对发送端 TX FIFO 写数据时，打包数据后，数据包中的控制段 NO_ACK 标志位复位。接收端收到一帧有效数据后，产生 RX_DR 中断自动发送一帧 ACK 信号，发送端接收到 ACK 信号后，自动清除 TX FIF 数据并产生 TX_DS 发射中断，如图：

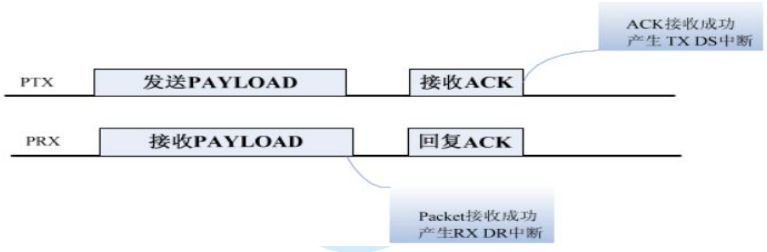


图 2.8.1 ACK 通信方式

1)发射方配置：

```
SPI_RW_REG(SETUP_AW,0x03); // 设置地址宽度为 5bytes
SPI_WRITE_BUF(TX_ADDR, TX_ADDRESS, 5); // 写入发送地址， 5 字节
SPI_WRITE_BUF(RX_ADDR_P0,TX_ADDRESS,5); //接收通道 0 地址和发射地址相同
SPI_WRITE_BUF(W_TX_PAYLOAD,buf,TX_PLOAD_WIDTH); // 写 TX FIFO
SPI_RW_REG(FEATURE,0x04); //使能动态负载长度
SPI_RW_REG(DYNPD,0x01); //开启 DPL_P0
SPI_RW_REG(SETUP_RETR,0x15); //自动重发延时等待 500us,自动重发 5 次
SPI_RW_REG(RF_CH,0x40); // 选择射频信道
SPI_RW_REG(RF_SETUP,0x0e); // 数据传输率 2Mbps 及功率
SPI_RW_REG(CONFIG,0x0e); //配置为发射模式、 CRC、可屏蔽中断
CE = 1;
```

2)接收方配置：

```
SPI_WRITE_BUF(RX_ADDR_P0,TX_ADDRESS,5); //接收通道 0 地址和发射地址相同
SPI_RW_REG(EN_RXADDR,0x01); // 使能接收通道 0
SPI_RW_REG(RF_CH,0x40); // 选择射频信道
SPI_RW_REG(RX_PW_P0,TX_PLOAD_WIDTH); //设置负载长度，使用 PIPE0 接收
SPI_RW_REG(SETUP_AW,0x03); // 设置地址宽度为 5bytes
SPI_RW_REG(FEATURE,0x04); //使能动态负载
```

```
SPI_RW_REG(DYNPD,0x01); //开启 DPL_P0
SPI_RW_REG(RF_SETUP,0x0e); // 数据传输率 2Mbps 及功率
SPI_RW_REG(CONFIG,0x0f); //配置为发射模式、 CRC、可屏蔽中断
CE = 1;
```

2.10. 问:实现 2.4G 模组一发多收，收发程序该怎样写？

答:发射端按下按键触发发送数据，接收端使用 LED 灯作为信号接收指示灯.当为多个接收时，只需对应改写接收端地址即可。

1）发射端程序

```
#define SI_KEY1    PAin(1)          // 按键1
#define SI_KEY2    PAin(2)          // 按键2
#define SI_KEY3    PAin(3)          // 按键3
#define SI_KEY4    PAin(4)          // 按键4
#define SI_KEY5    PAin(5)          // 按键5
#define SI_KEY6    PAin(6)          // 按键6
#define SI_LED     PCout(13)        // LED灯
//发射端程序
const u8 TX_ADDRESS1[TX_ADR_WIDTH]={0x0A,0x01,0x07,0x0E,0x01};  // 发送地址1
const u8 TX_ADDRESS2[TX_ADR_WIDTH]={0x0A,0x01,0x07,0x0E,0x02};  // 发送地址2
const u8 TX_ADDRESS3[TX_ADR_WIDTH]={0x0A,0x01,0x07,0x0E,0x03};  // 发送地址3
const u8 TX_ADDRESS4[TX_ADR_WIDTH]={0x0A,0x01,0x07,0x0E,0x04};  // 发送地址4
const u8 TX_ADDRESS5[TX_ADR_WIDTH]={0x0A,0x01,0x07,0x0E,0x05};  // 发送地址5
const u8 TX_ADDRESS6[TX_ADR_WIDTH]={0x0A,0x01,0x07,0x0E,0x06};  // 发送地址6
int main(void)
{
  u8 buf[32] = {0};//要发得到数据包
  u8 KeyNum = 0;//按键的键值
  LED_Init();
  KEY_Init();
  delay_init();
  SI24R1_Init();
  while(1)
  {
    if(!SI_KEY1 || !SI_KEY2 || !SI_KEY3 || !SI_KEY4 || !SI_KEY5 || !SI_KEY6)
    {
      //判断是哪一个按键按下啦
      KeyNum=Keynum_return();
      //根据按键的值进行发包
      switch(KeyNum)
      {
        case 1:buf[0] = 0xAA;SendData((u8 *)TX_ADDRESS1,(u8 *)buf); break;
        case 2:buf[0] = 0xAA;SendData((u8 *)TX_ADDRESS2,(u8 *)buf); break;
        case 3:buf[0] = 0xAA;SendData((u8 *)TX_ADDRESS3,(u8 *)buf); break;
        case 4:buf[0] = 0xAA;SendData((u8 *)TX_ADDRESS4,(u8 *)buf); break;
        case 5:buf[0] = 0xAA;SendData((u8 *)TX_ADDRESS5,(u8 *)buf); break;
        case 6:buf[0] = 0xAA;SendData((u8 *)TX_ADDRESS6,(u8 *)buf); break;
        default:break;
      }
    }
      delay_ms(100);
  }
}
```

2）接收端程序

```
#define SI_KEY1    PAin(1)          // 按键1
#define SI_KEY2    PAin(2)          // 按键2
#define SI_KEY3    PAin(3)          // 按键3
#define SI_KEY4    PAin(4)          // 按键4
#define SI_KEY5    PAin(5)          // 按键5
#define SI_KEY6    PAin(6)          // 按键6

#define SI_LED     PCout(13)        // LED灯

//接收端程序
const u8 RX_ADDRESS1[TX_ADR_WIDTH]={0x0A,0x01,0x07,0x0E,0x01};  // 接受地址1
const u8 RX_ADDRESS2[TX_ADR_WIDTH]={0x0A,0x01,0x07,0x0E,0x02};  // 接受地址2
const u8 RX_ADDRESS3[TX_ADR_WIDTH]={0x0A,0x01,0x07,0x0E,0x03};  // 接受地址3
const u8 RX_ADDRESS4[TX_ADR_WIDTH]={0x0A,0x01,0x07,0x0E,0x04};  // 接受地址4
const u8 RX_ADDRESS5[TX_ADR_WIDTH]={0x0A,0x01,0x07,0x0E,0x05};  // 接受地址5
const u8 RX_ADDRESS6[TX_ADR_WIDTH]={0x0A,0x01,0x07,0x0E,0x06};  // 接受地址6
int main(void)
{
  LED_Init();
  KEY_Init();
  delay_init();
  SI24R1_Init();
  SI24R1_RX_Mode((u8 *)RX_ADDRESS1);//此处可修改接收地址
  while(1)
  {
    u8 buf[32] = {0};
    if(!SI24R1_RxPacket(buf))
    {
      switch(buf[0])
      {
        case 0xAA:
          SI_LED = 0;
          delay_ms(100);
          SI_LED = 1;
          delay_ms(100);
          break;
        default:
          break;
      }
      buf[0] = 0;
    }
    delay_ms(200);
  }
}
```

免责申明和版权公告

本文中的信息，包括供参考的 URL 地址，如有变更，恕不另行通知。

文档“按现状”提供，不负任何担保责任，包括对适销性、适用于特定用途或非侵权性的任何担保，和任何提案、规格或样品在他处提到的任何担保。本文档不负任何责任，包括使用本文档内信息产生的侵犯任何专利权行为的责任。本文档在此未以禁止反言或其他方式授予任何知识产权使用许可，不管是明示许可还是暗示许可。

文中所得测试数据均为安信可实验室测试所得，实际结果可能略有差异。

Wi-Fi 联盟成员标志归 Wi-Fi 联盟所有。

文中提到的所有商标名称、商标和注册商标均属其各自所有者的财产，特此声明。

由于产品版本升级或其他原因，本手册内容有可能变更。深圳市安信可科技有限公司保留在没有任何通知或者提示的情况下对本手册的内容进行修改的权利。本手册仅作为使用指导，深圳市安信可科技有限公司尽全力在本手册中提供准确的信息，但是深圳市安信可科技有限公司并不确保手册内容完全没有错误，本手册中的所有陈述、信息和建议也不构成任何明示或暗示的担保。

最终解释权归深圳市安信可科技有限公司所有。

联系我们

官方官网：https://www.ai-thinker.com
开发 DOCS：http://docs.aithinker.com
官方论坛：http://bbs.ai-thinker.com
样品购买：https://anxinke.taobao.com
商务合作：sales@aithinker.com
技术支持：support@aithinker.com
公司地址:深圳市宝安区西乡固戍华丰智慧创新港 C 栋 410
联系电话：0755-29162996