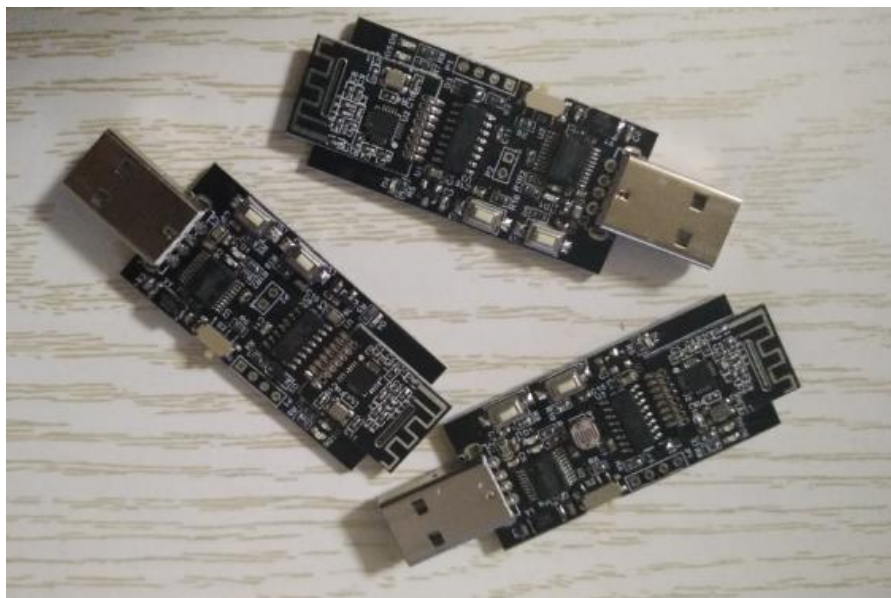


## STC15Wxx 测试板使用指南



nRF24L01+（或 nRF24L01P）是一款工作在 2.4~2.5GHz 世界通用 ISM 频段的单片无线收发器芯片。无线收发器包括：频率发生器、增强型 ShockBurst 模式控制器、功率放大器、晶体振荡器调制器、解调器。输出功率频道选择和协议的设置可以通过 SPI 接口进行设置。

极低的电流消耗，当工作在发射模式下发射功率为 0dBm 时电流消耗为 11.3mA，接收模式时为 13.5mA，掉电模式和待机模式下电流消耗更低。

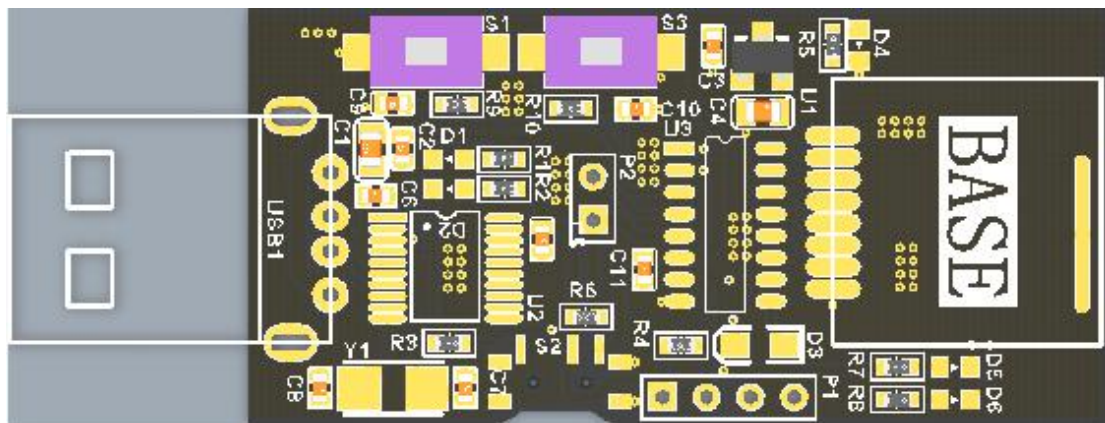
因为在无线通讯应用中经常会遇到远距离通讯的要求，目前有一些 nRF24L01+无线模块在原设计上增加了 PA（功率放大器）和 LNA（低噪声放大器）的型号，如“nRF24L01+PA”等。在发射部分通过 PA 电路将 nRF24L01+最大 0dBm 的输出功率放大到+22dBm 左右，同时在接收部分通过 LNA 电路增加接收信号的强度。通过这种方式可以有效的增加 nRF24L01+无线模块的通讯距离，在空旷环境下最高可增加到 2km。

## 目录

一、 产品介绍.....	1
二、 开发环境搭建.....	2
三、 硬件调试方法.....	10
四、 SPI 编程说明.....	11

## 一、产品介绍

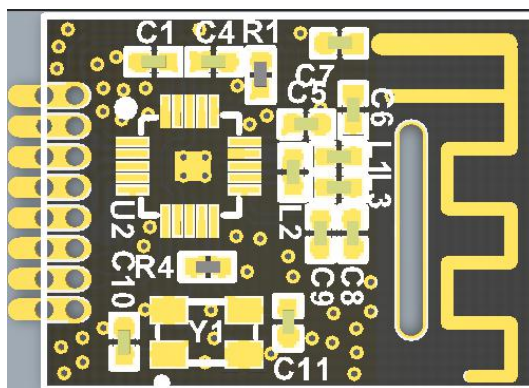
STC15Wxx 测试板全套硬件包括：以 STC15xx 系列单片机为核心的开发板、以 NRF24L01+为核心的 2.4G 无线模块。



1.1 STC15Wxx 测试板

**STC15Wxx 测试板：**

1. 板载两个独立按键
2. 黑色陶瓷晶振，2 毛钱一颗的廉价晶振不想用
3. 硬件支持光敏电阻、热敏电阻、温湿度传感器等
4. 板载单片机为 STC15Wxx 系列，最高频率可达 33.1776M
5. 板载 CH340 串口芯片，无需另外购买下载器，插上电脑就能做程序开发



1.2 NRF 无线模块

**NRF24L01+模块：**

1. 使用 NRF24L01+芯片
2. 板载 PCB 天线，有效距离 10~30m（不扯淡，板载天线也就这种强度了，网上动不动上百上千距离的模块，一般是加了 PA 电路，或使用外接天线的）

## 二、开发环境搭建

### 1. 配套资料

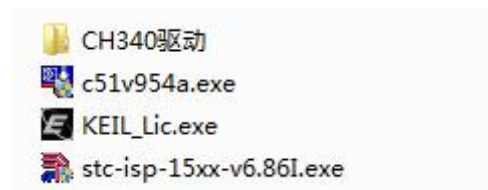
考虑到使用这个模块的人，有可能是刚接触无线通信，甚至刚接触单片机的纯小白，这里简单介绍下开发环境的搭建。

在模块配套的资料包里面，有一个名为开发软件的文件夹。本产品会用到的软件都会在里面。不过考虑到后期的产品升级，文件夹内也许会添加更多的文件，所以截图不为最终版本资料。



#### 2.1 产品配套资料

打开名为开发软件的文件夹，里面包括了程序开发所使用到所有软件。



#### 2.2 开发软件内容

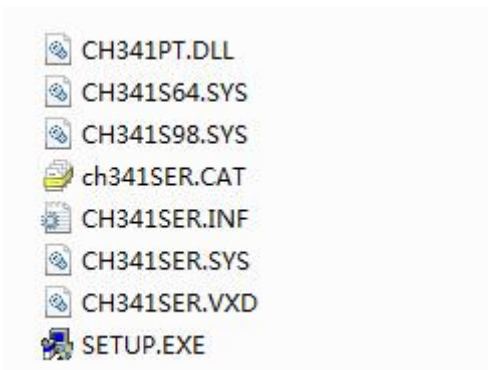
### 2. CH340 驱动的安装

直接打开 CH340 驱动文件夹，根据自己的需要选择对应的文件夹。



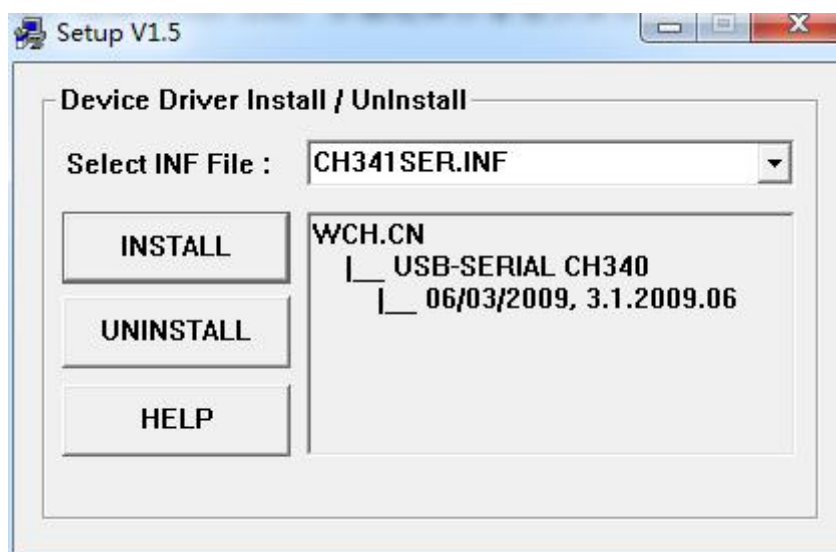
#### 2.3 两种平台的驱动

我是 WIN7 的系统，那么以 WIN7 的为例说一下安装工程。打开文件夹，最终看到如图内容，选择 SETUP.EXE，右键选择以管理员身份打开该软件。



## 2.4 CH340 驱动安装文件

打开界面如下所示，点击 INSTALL 选项。



## 2.4 CH340 安装界面

安装成功之后，在电脑插上一个 STC15Wxx 测试模块，打开电脑的设备管理器，可以看到自己的驱动安装好了，我这里的括号内显示端口是 COM5，每个人的电脑都不一样，不一定是 COM5，但是括号外的是一样的，如果设备管理器这里显示带有黄色感叹号，那么说明驱动安装失败了，这个时候，可以使用驱动人生等软件辅助安装驱动。



## 2.5 设备管理器内容

打开 STC-ISP 软件，这个软件直接复制到桌面打开就是了，不需要安装，在配套资料里面也有



## 2.6 STC-ISP 软件

如果驱动安装正常，软件会自动找到设备。



2.7 STC-ISP 检测到串口设备

### 3. KEIL 软件的安装及注册

下面文字来源于网络，软件版本有差异，但是过程是一致的：

<https://jingyan.baidu.com/article/c85b7a6447bf9d003bac9500.html>

即将安装软件如下，一个 keil 安装程序，一个注册机（右侧）



打开 C51V901.EXE 安装程序，点击 Next



I agree all the tems of……选中

点击 Next





设置安装目录，根据自己的情况选中安装目录，重新设置点击 Browse，这里默认 C 盘，设置好安装目录，点击 Next



输入相关信息（随便输入），输入完毕后点击 Next

The screenshot shows the 'Setup Keil C51 Version 9.01' window with the 'Customer Information' tab selected. The window has a blue title bar and a yellow background. The 'KEIL' logo is in the top right corner. The text 'Please enter your information.' is displayed. Below it, a message says 'Please enter your name, the name of the company for whom you work, and your E-mail address.' There are four input fields: 'First Name' with '好友电子', 'Last Name' with '好友电子', 'Company Name' with '好友电子', and 'E-mail' with 'haoyoudianzi'. At the bottom, there are buttons for '<< Back', 'Next >>', and 'Cancel'. A small 'Keil μVision4 Setup' label is in the bottom left corner.

Setup Keil C51 Version 9.01

**Customer Information**

Please enter your information.

KEIL<sup>TM</sup>  
An ARM<sup>®</sup> Company

Please enter your name, the name of the company for whom you work, and your E-mail address.

First Name: 好友电子

Last Name: 好友电子

Company Name: 好友电子

E-mail: haoyoudianzi

Keil μVision4 Setup

<< Back Next >> Cancel

开始安装，安装过程中……等待安装完成

The screenshot shows the 'Setup Keil C51 Version 9.01' window with the 'Setup Status' tab selected. The window has a blue title bar and a yellow background. The 'KEIL' logo is in the top right corner. The text 'μVision Setup is performing the requested operations.' is displayed. Below it, the progress is shown as 'Install Files ...' and 'Installing MAIN.C.'. A progress bar with five blue segments is visible. At the bottom, there are buttons for '<< Back', 'Next >>', and 'Cancel'. A small 'Keil μVision4 Setup' label is in the bottom left corner.

Setup Keil C51 Version 9.01

**Setup Status**

μVision Setup is performing the requested operations.

Install Files ...

Installing MAIN.C.

Keil μVision4 Setup

<< Back Next >> Cancel

安装完成，点击 Finish 即可。



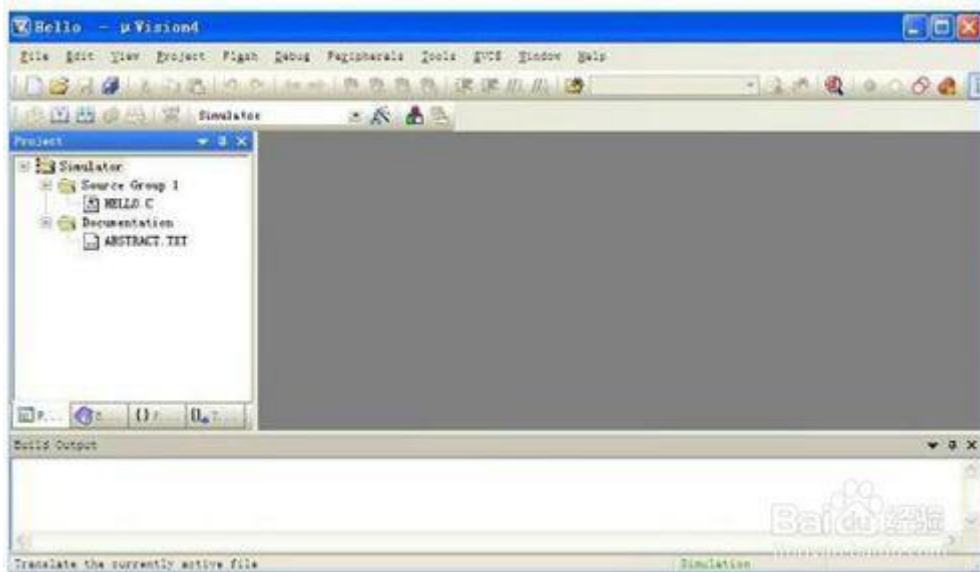


接下来破解软件，打开注册机软件

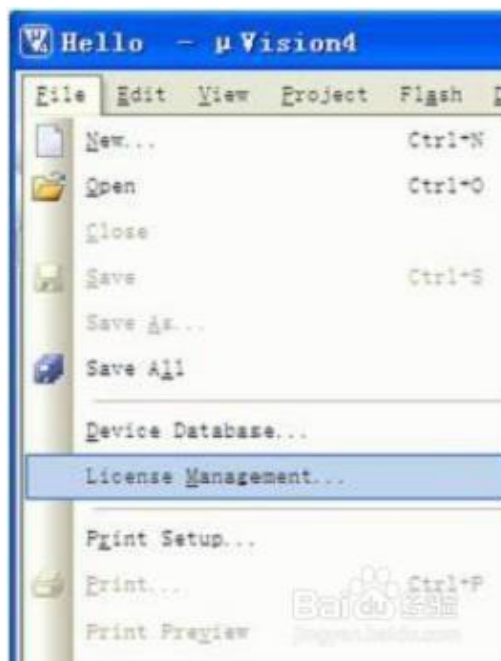


Target: 选择 C51

打开刚刚安装好的 Keil 软件



点击 File, 选择 License Management

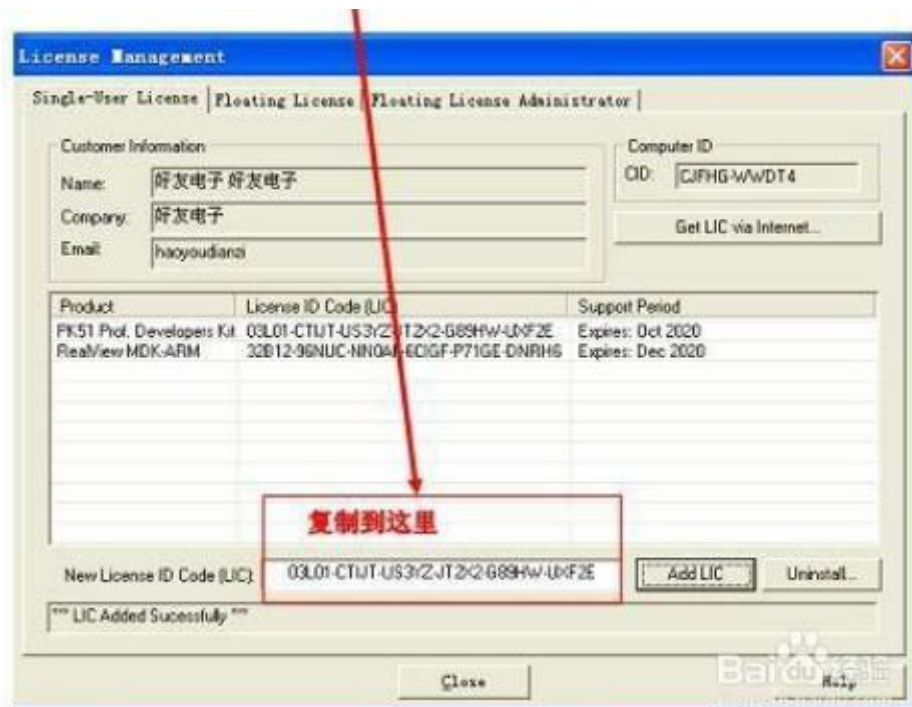


现在还没有破解, 复制到 CID 码到注册机中

现在没有破解



复制完成后点击右侧 Add LIC，即可完成破解



提示如下



### 三、硬件调试方法

如果用户没有使用 STC15Wxx 测试板，只用了一个 NRF 模块。一定要按照如下步骤，检测硬件，否则有可能会驱动不亮 NRF 模块。

1. 确定电源，NRF 模块的供电是 3.3V，一定不能太高或者太低，否则可能会烧毁芯片。但这是电源的第一步，还有一步至关重要，那就是电源纹波，一定要使用示波器，查看 3.3V 的电源纹波，很多刚接触这个模块的人，在硬件设计上都是输在了这一步，几十 mV 左右的纹波是可以接受的，当然纹波还是越小越好，如果纹波上百了，那么很有可能读写寄存器成功了，但是无法正常通信。

2. 先去将单片机控制 NRF 的 IO 做电平翻转测试，确定 IO 口是正常工作的。

3. 如果是自己设计电路，那么要注意，在 NRF 模块周围不要摆放高频的元器件以及电源，最好的办法是，NRF 模块下给一个单独的地，天线下方做镂空处理。

## 四、SPI 编程说明

模块配套资料中，有几个测试代码。



### 4.1 NRF 测试程序

打开上图所示的工程，NRF 驱动都是一样的，只是初始化不同，这里我打开接收的测试代码，说一下 NRF 的编程方法。如果要做程序移植，直接把 NRF24L01.c 以及 NRF24L01.h 复制出去使用就行了，当然几点必要的改动还是要有的。

首先是系统时钟的设置，在 system.h 里面有对时钟的设置。烧录的时候记得设置一样的时钟，因为 STC-ISP 软件打开就是 11.0592Mhz 的时钟，避免客户有时忘记改，我将时钟配置为 11.0592Mhz，有的程序里面注释可能是 33.1776Mhz，请直接忽略。

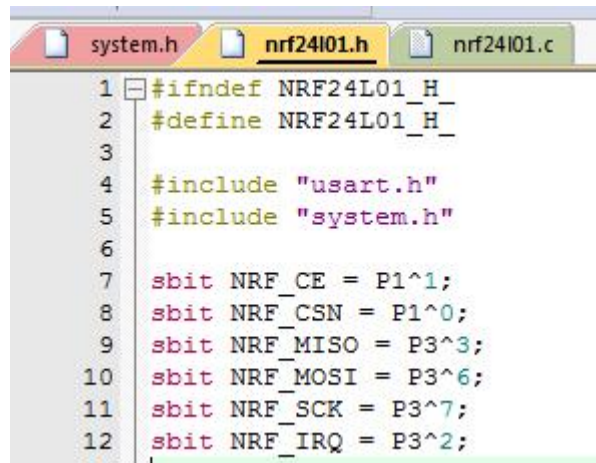


### 4.2 NRF 程序时钟配置

配置 NRF 的控制端口，模块的原理图在附送资料里面也有，方便客户参考，



使用的是模拟 SPI，使用其他平台请自行设置。



```
1 #ifndef NRF24L01_H_
2 #define NRF24L01_H_
3
4 #include "usart.h"
5 #include "system.h"
6
7 sbit NRF_CE = P1^1;
8 sbit NRF_CSN = P1^0;
9 sbit NRF_MISO = P3^3;
10 sbit NRF_MOSI = P3^6;
11 sbit NRF_SCK = P3^7;
12 sbit NRF_IRQ = P3^2;
```

#### 4.3 NRF 模块端口设置

程序里面，端口设置之后，就是寄存器的定义，这一部分是根据收据手册二来，移植不需要改动。

```
/****** NRF24L01寄存器操作命令 *****/
#define READ_REG      0x00 //读配置寄存器,低5位
#define WRITE_REG     0x20 //写配置寄存器,低5位
#define RD_RX_PLOAD   0x61 //读RX有效数据,1~32字节
#define WR_TX_PLOAD   0xA0 //写TX有效数据,1~32字节
#define FLUSH_TX      0xE1 //清除TX FIFO寄存器.
#define FLUSH_RX      0xE2 //清除RX FIFO寄存器.
#define REUSE_TX_PL   0xE3 //重新使用上一包数据
#define NOP           0xFF //空操作,可以用来读写
/****** NRF24L01寄存器地址 *****/
#define CONFIG        0x00 //配置寄存器地址
#define EN_AA         0x01 //使能自动应答功能
#define EN_RXADDR     0x02 //接收地址允许
#define SETUP_AW      0x03 //设置地址宽度(所有寄存器)
#define SETUP_RETR    0x04 //建立自动重发
#define RF_CH         0x05 //RF通道
#define RF_SETUP       0x06 //RF寄存器
#define STATUS        0x07 //状态寄存器
#define OBSERVE_TX    0x08 // 发送检测寄存器
#define CD            0x09 // 载波检测寄存器
#define RX_ADDR_P0    0x0A // 数据通道0接收地址
#define RX_ADDR_P1    0x0B // 数据通道1接收地址
#define RX_ADDR_P2    0x0C // 数据通道2接收地址
#define RX_ADDR_P3    0x0D // 数据通道3接收地址
#define RX_ADDR_P4    0x0E // 数据通道4接收地址
#define RX_ADDR_P5    0x0F // 数据通道5接收地址
#define TX_ADDR       0x10 // 发送地址寄存器
```

#### 4.3 NRF 模块寄存器

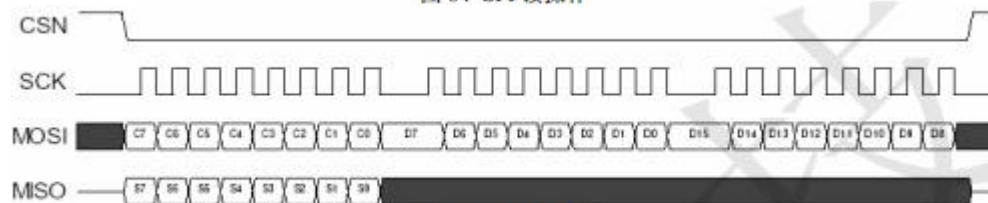
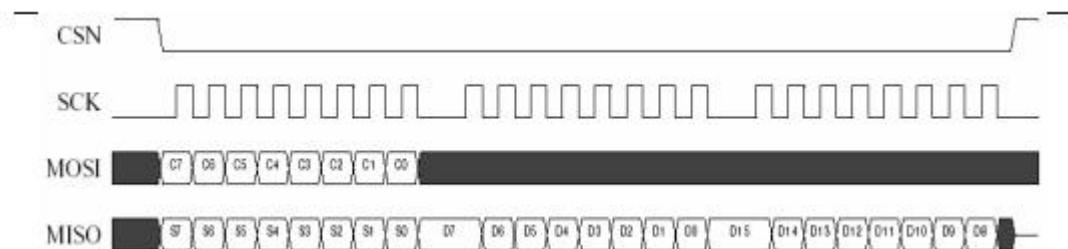
做 NRF 的程序开发，首先是要做好 SPI 通信协议。关于 SPI 协议的通信时序，请自行百度搜索。移植的时候，这一个函数，可能会根据所使用的开发平台，做相关的改动。

```

/*****
/* SPI数据收发函数 */
/*****
u8 SPI_RW(u8 tr_data)
{
    u16 bit_ctr;
    for(bit_ctr=0;bit_ctr<8;bit_ctr++) // output
    {
        NRF_MOSI = (tr_data & 0x80); // output
        tr_data = (tr_data << 1); // shift
        NRF_SCK = 1; // Set SCK
        tr_data |= NRF_MISO; // capture
        NRF_SCK = 0; // ..then set
    }
    return(tr_data); // return read
}

```

### 4.3 SPI 读写函数



### 4.4 SPI 读写时序(来源于 NRF24L01 数据手册)

然后是读写寄存器内容，时序逻辑在附送资料里面有。这里是移植的重点，不同的开发平台，如果主频比较高，要考虑每步操作之间的间隔时间。

```

/*****
/* 函数功能：给24L01的寄存器写值（一个字节） */
/* 入口参数：reg 要写的寄存器地址 */
/*             value 给寄存器写的值 */
/* 出口参数：status 状态值 */
*****/
u8 NRF24L01_RW_Reg(u8 reg,u8 value)
{
    u16 status;

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

    return(status);              // return nRF24L01 status uchar
}
/*****
/* 函数功能：读24L01的寄存器值（一个字节） */
/* 入口参数：reg 要读的寄存器地址 */
/* 出口参数：value 读出寄存器的值 */
*****/
u8 NRF24L01_Read_Reg(u8 reg)
{
    u8 reg_val;

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

    return(reg_val);             // return register value
}

```

#### 4.4 寄存器读写函数

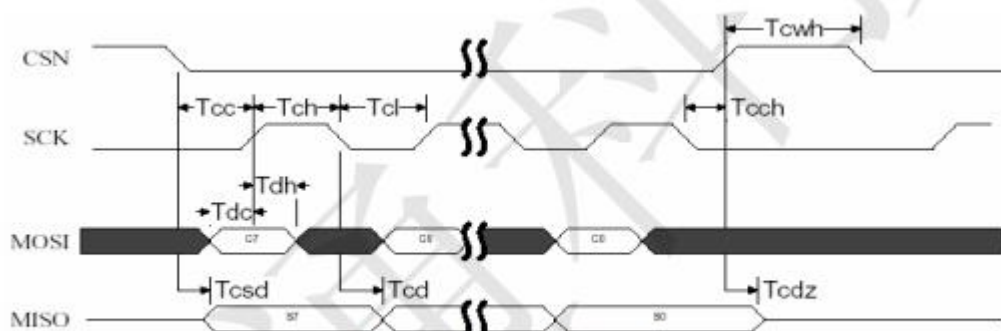


图 10、SPI NOP 操作时序图



PARAMETER	SYMBOL	MIN	MAX	UNITS
Data to SCK Setup	Tdc	2		ns
SCK to Data Hold	Tdh	2		ns
CSN to Data Valid	Tcsd		38	ns
SCK to Data Valid	Tcd		55	ns
SCK Low Time	Tel	40		ns
SCK High Time	Teh	40		ns
SCK Frequency	Fsck	0	8	MHz
SCK Rise and Fall	Tr,Tf		100	ns
CSN to SCK Setup	Tcc	2		ns
SCK to CSN Hold	Tcch	2		ns
CSN Inactive time	Tcwh	50		ns
CSN to Output High Z	Tcdz		38	ns

表 9、SPI 参考时间 (C<sub>load</sub>=5pF)

PARAMETER	SYMBOL	MIN	MAX	UNITS
Data to SCK Setup	Tdc	2		ns
SCK to Data Hold	Tdh	2		ns
CSN to Data Valid	Tcsd		42	ns
SCK to Data Valid	Tcd		58	ns
SCK Low Time	Tel	40		ns
SCK High Time	Teh	40		ns
SCK Frequency	Fsck	0	8	MHz
SCK Rise and Fall	Tr,Tf		100	ns
CSN to SCK Setup	Tcc	2		ns
SCK to CSN Hold	Tcch	2		ns
CSN Inactive time	Tcwh	50		ns
CSN to Output High Z	Tcdz		42	ns

表 10、SPI 参考时间 (C<sub>load</sub>=10pF)

#### 4.5 寄存器读写时序

后面读写数组的就不说了，移植不需要改动，说一下，校验函数。

```

/*****/
u8 NRF24L01_Check(void)
{
    u8 check_in_buf[5]={0x11,0x22,0x33,0x44,0x55};
    u8 check_out_buf[20]={0x00};

    NRF_CE=0;

    NRF24L01_Write_Buf(WRITE_REG+TX_ADDR, check_in_buf, 5);
    NRF24L01_Read_Buf(READ_REG+TX_ADDR, check_out_buf, 5);

    if((check_out_buf[0] == 0x11)&&\
        (check_out_buf[1] == 0x22)&&\
        (check_out_buf[2] == 0x33)&&\
        (check_out_buf[3] == 0x44)&&\
        (check_out_buf[4] == 0x55))return 0;

    return 1;
}

```

#### 4.6 硬件检测函数

这个函数实现的功能是给 NRF 的寄存器写数据，然后再读取出来，如果读写正确返回 0，读写有问题就返回 1. 在主函数的调用里面会将这个函数做一个 while 循环里面，可以用来检测硬件是否正常工作。

然后是初始化函数，就是一个对寄存器的配置过程。

```
void NRF24L01_RT_Init(void)
{
    NRF_CE=0;    // chip enable
    NRF24L01_Write_Buf(WRITE_REG + RX_ADDR_P0, TX_ADDRESS);
    NRF24L01_RW_Reg(WRITE_REG+EN_AA,0x01);    //使能通道0
    NRF24L01_RW_Reg(WRITE_REG+EN_RXADDR,0x01); //使能通道0
    NRF24L01_RW_Reg(WRITE_REG+RF_CH,0);        //设置RF通道为125
    NRF24L01_RW_Reg(WRITE_REG+RX_PW_P0,TX_PLOAD_WIDTH); //设置通道0的接收地址
    NRF24L01_RW_Reg(WRITE_REG+RF_SETUP,0x07);  //7db增益,250kbps
    NRF24L01_RW_Reg(WRITE_REG+CONFIG,0x0f);    //配置基本工作模式的参数
    NRF_CE=1; //CE置高
}
```

#### 4.7 对接收设备的配置

```
void NRF24L01_RT_Init(void)
{
    NRF_CE=0;    // chip enable
    NRF24L01_Write_Buf(WRITE_REG+TX_ADDR, (u8*)TX_ADDRESS, TX_ADR_WIDTH);
    NRF24L01_Write_Buf(WRITE_REG+RX_ADDR_P0, (u8*)TX_ADDRESS, TX_ADR_WIDTH);
    NRF24L01_RW_Reg(WRITE_REG+EN_AA,0x01);    //使能通道0的自动应答
    NRF24L01_RW_Reg(WRITE_REG+EN_RXADDR,0x01); //使能通道0的接收地址
    NRF24L01_RW_Reg(WRITE_REG+SETUP_RETR,0x1a); //设置自动重发间隔时间
    NRF24L01_RW_Reg(WRITE_REG+RF_CH,0);        //设置RF通道为125
    NRF24L01_RW_Reg(WRITE_REG+RF_SETUP,0x07);  //7db增益,250kbps
    NRF24L01_RW_Reg(WRITE_REG+CONFIG,0x0e);    //配置基本工作模式的参数
    NRF_CE=1; //CE置高
}
```

#### 4.8 对发送模式的配置

过程都是一样的，只是有个别的数据有差异。然后还有一个要注意的是收发的地址，必须保证主机和从机是一致的。

```
const u8 TX_ADDRESS[TX_ADR_WIDTH]={0x34,0x43,0x10,0x10,0x01};
const u8 RX_ADDRESS[RX_ADR_WIDTH]={0x34,0x43,0x10,0x10,0x01};
```

#### 4.9 收发地址设置