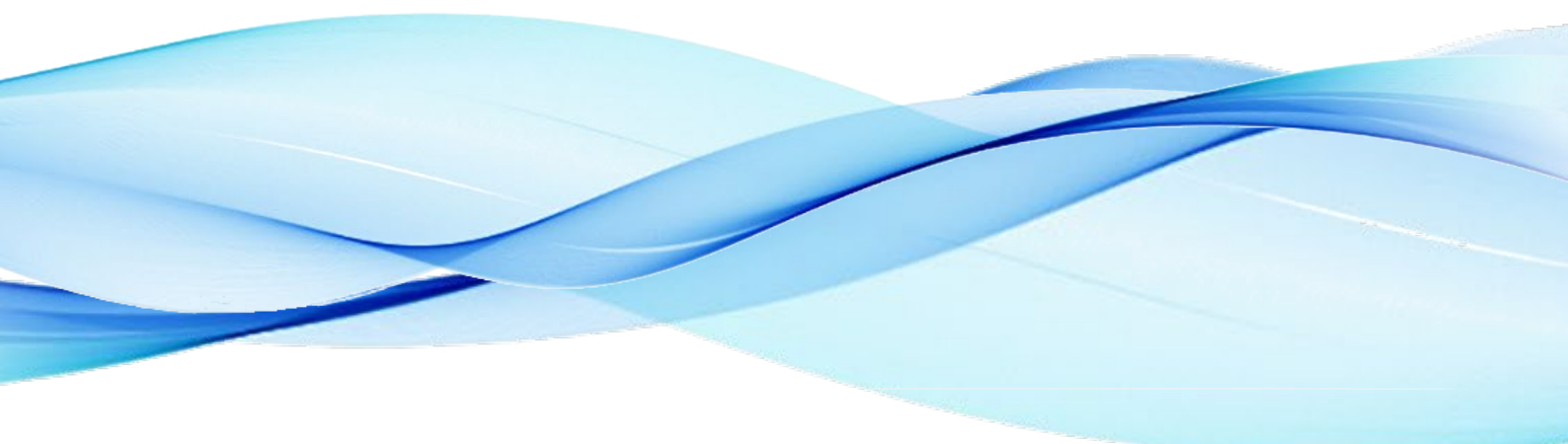




# Si24R1\_Stc89c52

## Demo 程序说明

---



官网: [www.ashining.com](http://www.ashining.com)

邮箱: [support@ashining.com](mailto:support@ashining.com)

地址: 四川省·成都市·高新西区百草路898号  
智能信息产业园2层、5层

## Si24R1\_Stc89c52 的 demo 程序说明

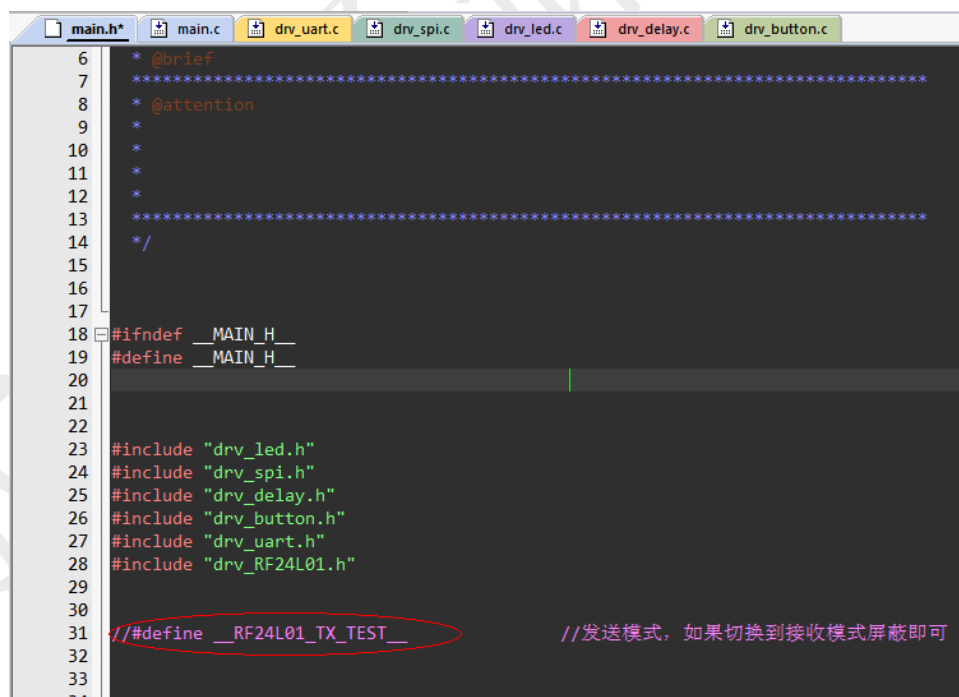
本 demo 程序是基于 Stc89c52 单片机和 Si24R1 开发设计的。本程序包含了主函数文件 main.c, SPI 文件 drv\_spi.c, 串口文件 drv\_uart.c, 指示灯相关函数的文件 drv\_led.c, 按键相关函数文件 drv\_button.c, 延时函数文件 drv\_delay.c 以及 Si24R1 的驱动文件 drv\_RF24L01.c 等。

本程序实现的功能是使用 Si24R1 进行透明传输的功能, 但只支持单独的接收或单独的发送。若想实现收发一体的功能需要用户自行修改程序。发送功能中分为了 2 种模式, 固定发送模式和自由发送模式, 由按键控制。自由发送模式是发送串口收到的数据。

### 1. 切换发送功能或接收功能的进行编译

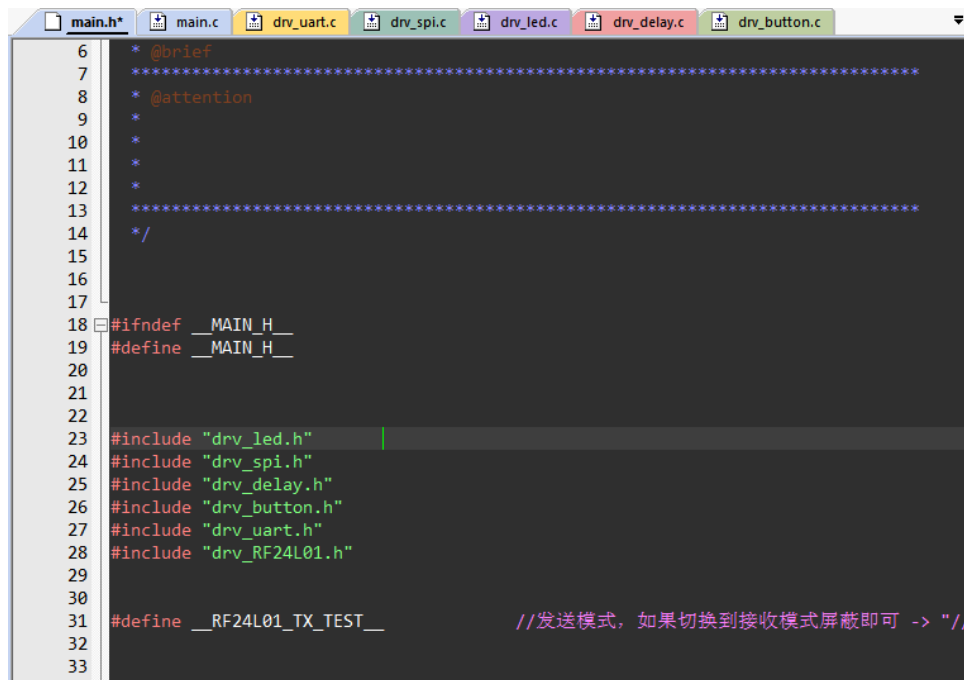
打开工程文件后, 在 main.c 文件中有 \_\_RF24L01\_TX\_TEST\_\_ 的一个宏定义, 若该参数未被定义的话发送功能则未被编译, 若就当前状态进行编译下载, 则该模块有了接收功能。要想编译下载发送功能的程序, 需要点开 main.h 文件, 将 #define \_\_RF24L01\_TX\_TEST\_\_ 释放出来即可。如图:

打开 main.h 文件



```
6  * @brief
7  *
8  * @attention
9  *
10 *
11 *
12 *
13 *
14 */
15
16
17
18 #ifndef __MAIN_H__
19 #define __MAIN_H__
20
21
22
23 #include "drv_led.h"
24 #include "drv_spi.h"
25 #include "drv_delay.h"
26 #include "drv_button.h"
27 #include "drv_uart.h"
28 #include "drv_RF24L01.h"
29
30
31 // #define __RF24L01_TX_TEST__ // 发送模式, 如果切换到接收模式屏蔽即可
32
33
34
```

释放掉圈出部分

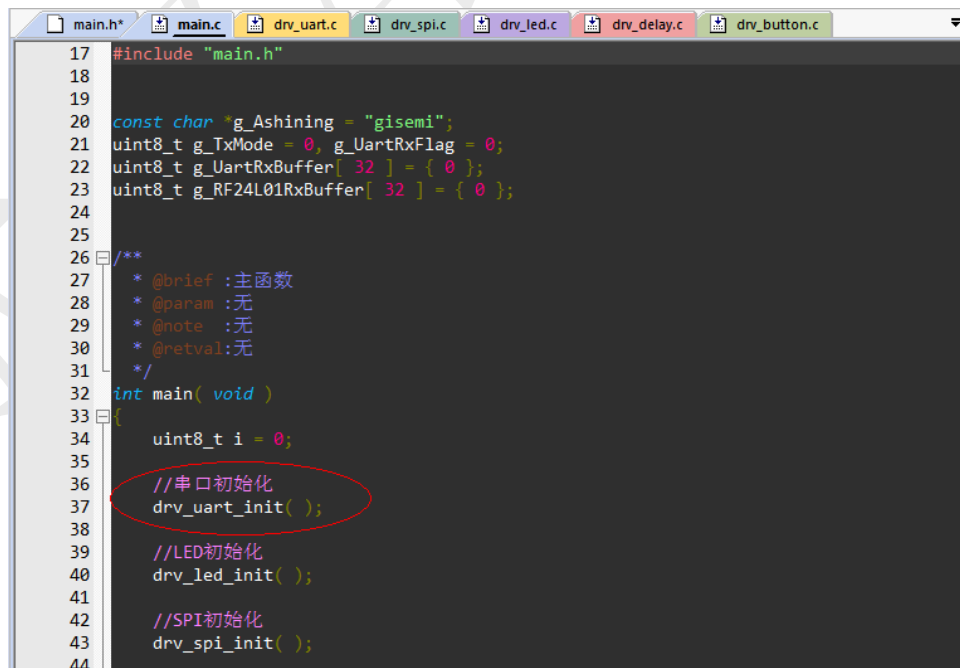


```
6  * @brief
7  *****
8  * @attention
9  *
10 *
11 *
12 *
13 *****
14 */
15
16
17
18 #ifndef MAIN_H_
19 #define MAIN_H_
20
21
22
23 #include "drv_led.h"
24 #include "drv_spi.h"
25 #include "drv_delay.h"
26 #include "drv_button.h"
27 #include "drv_uart.h"
28 #include "drv_RF24L01.h"
29
30
31 #define __RF24L01_TX_TEST__ //发送模式，如果切换到接收模式屏蔽即可 -> "/*
32
33
```

现在的主程序，发送功能可编译，接受功能部分不可编译

## 2. 更改串口波特率

本程序默认的串口波特率是 9600，我们可以通过更改 drv\_uart\_init() 中 TH 和 TL 的值来更改串口波特率。drv\_uart\_init() 在 drv\_uart.c 文件中如图：



```
17 #include "main.h"
18
19
20 const char *g_Ashining = "gisemi";
21 uint8_t g_TxMode = 0, g_UartRxFlag = 0;
22 uint8_t g_UartRxBuffer[ 32 ] = { 0 };
23 uint8_t g_RF24L01RxBuffer[ 32 ] = { 0 };
24
25
26 /**
27  * @brief :主函数
28  * @param :无
29  * @note :无
30  * @retval:无
31  */
32 int main( void )
33 {
34     uint8_t i = 0;
35
36     //串口初始化
37     drv_uart_init( );
38
39     //LED初始化
40     drv_led_init( );
41
42     //SPI初始化
43     drv_spi_init( );
44
```

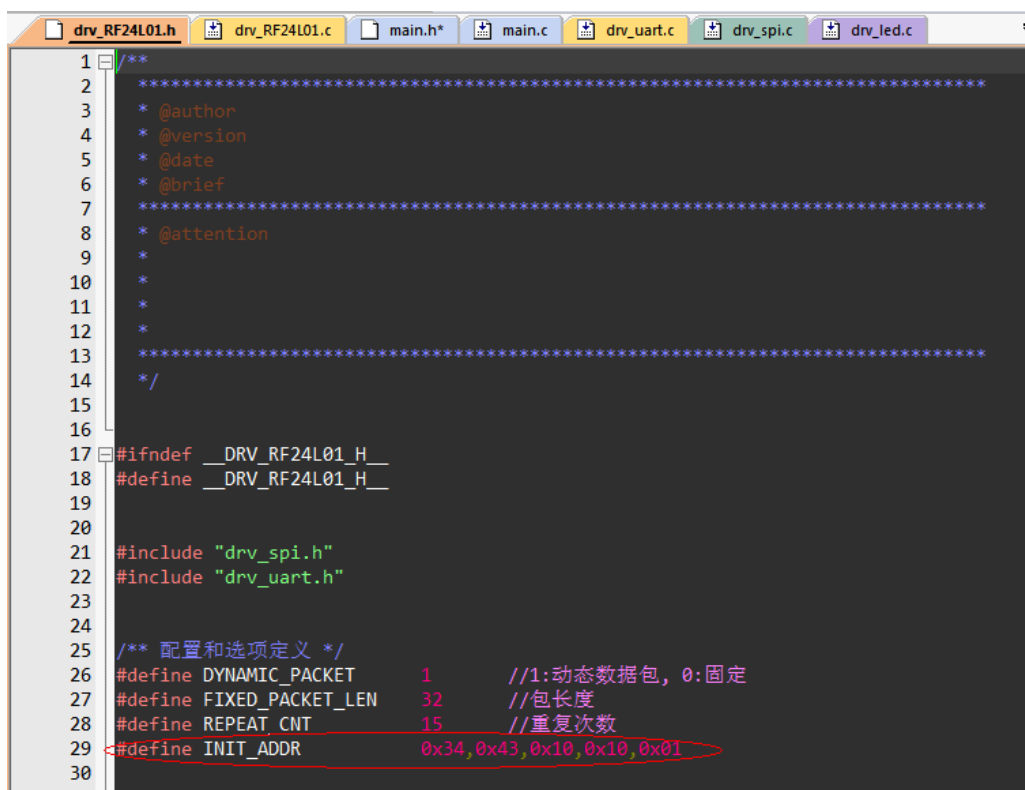
```
18
19
20
21 /**
22  * @brief :串口初始化
23  * @param :无
24  * @note :无
25  * @retval:无
26  */
27 void drv_uart_init( )
28 {
29     //引脚配置 部分51单片机不需要
30     //TX配置为输出 RX配置为输入
31     UART_TX_PxM0 |= IO_OUT_PUT_PP_M0 << UART_TX_PIN_BIT;
32     UART_TX_PxM1 |= IO_OUT_PUT_PP_M1 << UART_TX_PIN_BIT;
33     UART_RX_PxM0 |= IO_IN_PUT_ONLY_M0 << UART_RX_PIN_BIT;
34     UART_RX_PxM1 |= IO_IN_PUT_ONLY_M1 << UART_RX_PIN_BIT;
35     UART_TX = 1;
36
37     //串口配置
38     SCON &= (uint8_t)((uint8_t)( ~( UART_MODE | UART_RX ))); //清SM0 SM1 REN
39     SCON |= (uint8_t)( UART_8BAUDRATE_VOLATILE | UART_RX );
40
41     //TIM1配置
42     TMOD &= (uint8_t)((uint8_t)( ~TIM1_MODE ));
43     TMOD |= TIM1_MODE_2; //8位自动重装
44     PCON = 0x00;
45     TH1 = 0xFD; //波特率默认配置为9600
46     TL1 = 0xFD;
47
48     TI = 1; //清发送标志
49     TR1 = 1; //使能定时器
50 }
```

本程序默认波特率为 9600 则在 11.0592Mhz 晶振且 SMOD=0 时配置为 TH1=0xFD,TL1=0xFD。例如需要配置成

相同条件需要配置成 2400 则需要配置为 TH1=0xF4,TL1=0xF4。

### 3. 更改 Si24R1 的通信地址

若需更改 Si24R1 的通信地址，我们需要打开 drv\_RF24L01.h 文件，更改 INIT\_ADDR 的宏定义参数。值得注意的是发送模块和接收模块的通信地址要一致才能通信。

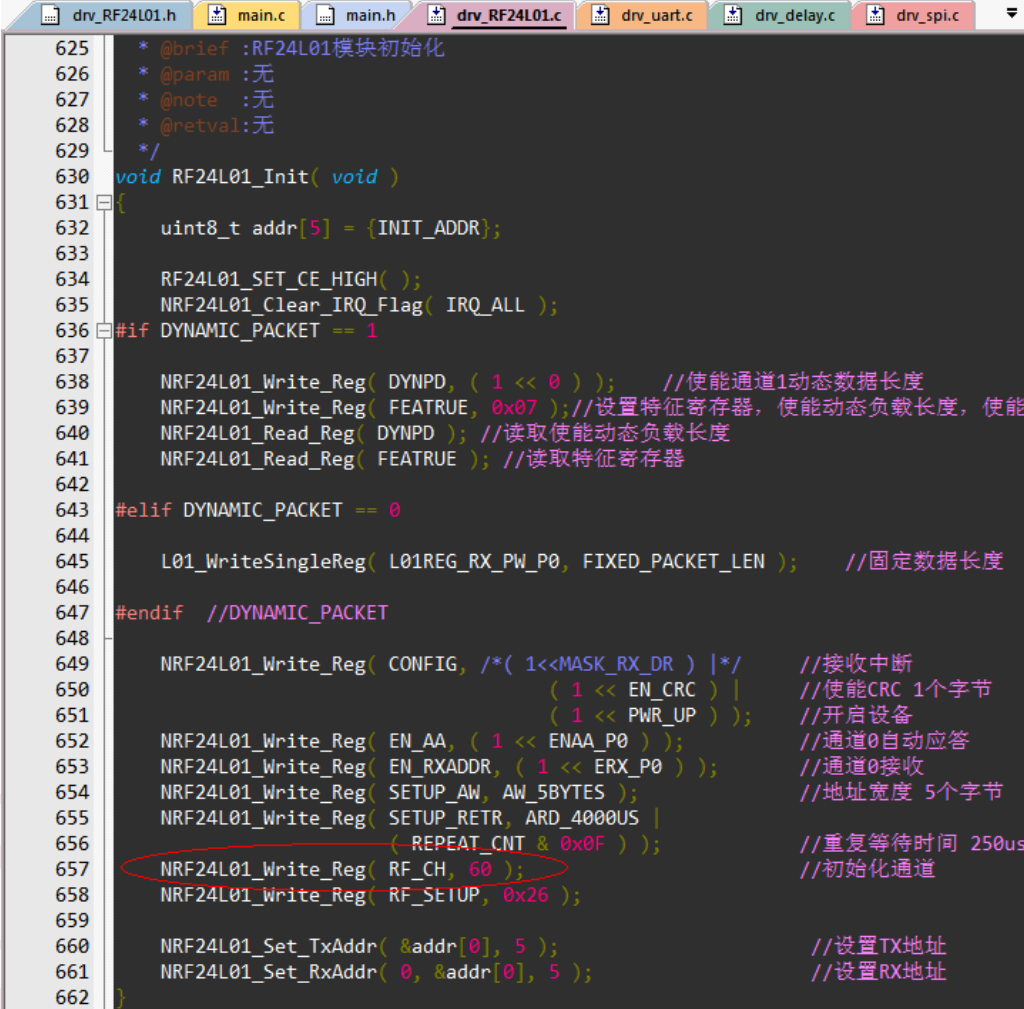


```
1  /**
2  ****
3  * @author
4  * @version
5  * @date
6  * @brief
7  ****
8  * @attention
9  *
10 *
11 *
12 *
13 ****
14 */
15
16
17 #ifndef __DRV_RF24L01_H__
18 #define __DRV_RF24L01_H__
19
20
21 #include "drv_spi.h"
22 #include "drv_uart.h"
23
24
25 /** 配置和选项定义 */
26 #define DYNAMIC_PACKET 1 //1:动态数据包, 0:固定
27 #define FIXED_PACKET_LEN 32 //包长度
28 #define REPEAT_CNT 15 //重复次数
29 #define INIT_ADDR 0x34, 0x43, 0x10, 0x10, 0x01
30
31
```

## 4. 更改 Si24R1 的通信配置

若需更改 Si24R1 的通信配置则需要在 drv\_rf24l1.c 文件中的 RF24L01\_Init()函数修改对应参数。其中通信配置中最重要的是信道，空速，发射功率等。需注意的是发送方与接收方的信道，空速，发射功率都需一致。

a. 若需要更改信道则更改 RF\_CH 寄存器的参数。



```
625  * @brief :RF24L01模块初始化
626  * @param :无
627  * @note :无
628  * @retval:无
629  */
630 void RF24L01_Init( void )
631 {
632     uint8_t addr[5] = {INIT_ADDR};
633
634     RF24L01_SET_CE_HIGH();
635     NRF24L01_Clear_IRQ_Flag( IRQ_ALL );
636     #if DYNAMIC_PACKET == 1
637
638     NRF24L01_Write_Reg( DYNPD, ( 1 << 0 ) ); //使能通道1动态数据长度
639     NRF24L01_Write_Reg( FEATRUE, 0x07 ); //设置特征寄存器，使能动态负载长度，使能
640     NRF24L01_Read_Reg( DYNPD ); //读取使能动态负载长度
641     NRF24L01_Read_Reg( FEATRUE ); //读取特征寄存器
642
643     #elif DYNAMIC_PACKET == 0
644
645     L01_WriteSingleReg( L01REG_RX_PW_P0, FIXED_PACKET_LEN ); //固定数据长度
646
647     #endif //DYNAMIC_PACKET
648
649     NRF24L01_Write_Reg( CONFIG, /*( 1<<MASK_RX_DR ) |*/ //接收中断
650                        ( 1 << EN_CRC ) | //使能CRC 1个字节
651                        ( 1 << PWR_UP ) ); //开启设备
652     NRF24L01_Write_Reg( EN_AA, ( 1 << ENAA_P0 ) ); //通道0自动应答
653     NRF24L01_Write_Reg( EN_RXADDR, ( 1 << ERX_P0 ) ); //通道0接收
654     NRF24L01_Write_Reg( SETUP_AW, AW_5BYTES ); //地址宽度 5个字节
655     NRF24L01_Write_Reg( SETUP_RETR, ARD_4000US |
656                        ( REPEAT_CNT & 0x0F ) ); //重复等待时间 250us
657     NRF24L01_Write_Reg( RF_CH, 60 ); //初始化通道
658     NRF24L01_Write_Reg( RF_SETUP, 0x26 );
659
660     NRF24L01_Set_TxAddr( &addr[0], 5 ); //设置TX地址
661     NRF24L01_Set_RxAddr( 0, &addr[0], 5 ); //设置RX地址
662 }
```

寄存器参数设置：(2.4GHz---2.525GHz)

05	RF_CH				RF Channel
	Reserved	7	0	R/W	Only '0' allowed
	RF_CH	6:0	0000010	R/W	Sets the frequency channel nRF24L01+ operates on

## b. 若需更改空速和发射功率则更改 RF\_SETUP 寄存器的参数。

```
625  * @brief :RF24L01模块初始化
626  * @param :无
627  * @note :无
628  * @retval:无
629  */
630 void RF24L01_Init( void )
631 {
632     uint8_t addr[5] = {INIT_ADDR};
633
634     RF24L01_SET_CE_HIGH( );
635     NRF24L01_Clear_IRQ_Flag( IRQ_ALL );
636 #if DYNAMIC_PACKET == 1
637     NRF24L01_Write_Reg( DYNPD, ( 1 << 0 ) ); //使能通道1动态数据长度
638     NRF24L01_Write_Reg( FEATRE, 0x07 ); //设置特征寄存器,使能动态负载长度,使能
639     NRF24L01_Read_Reg( DYNPD ); //读取使能动态负载长度
640     NRF24L01_Read_Reg( FEATRE ); //读取特征寄存器
641
642 #elif DYNAMIC_PACKET == 0
643     L01_WriteSingleReg( L01REG_RX_PW_P0, FIXED_PACKET_LEN ); //固定数据长度
644 #endif //DYNAMIC_PACKET
645
646     NRF24L01_Write_Reg( CONFIG, /*( 1 << MASK_RX_DR ) | */ //接收中断
647                         ( 1 << EN_CRC ) | //使能CRC 1个字节
648                         ( 1 << PWR_UP ) ); //开启设备
649     NRF24L01_Write_Reg( EN_AA, ( 1 << ENAA_P0 ) ); //通道0自动应答
650     NRF24L01_Write_Reg( EN_RXADDR, ( 1 << ERX_P0 ) ); //通道0接收
651     NRF24L01_Write_Reg( SETUP_AW, AW_5BYTES ); //地址宽度 5个字节
652     NRF24L01_Write_Reg( SETUP_RETR, ARD_4000US |
653                         ( REPEAT_CNT & 0x0F ) ); //重复等待时间 250us
654     NRF24L01_Write_Reg( RF_CH, 60 ); //初始化通道
655     NRF24L01_Write_Reg( RF_SETUP, 0x26 );
656
657     NRF24L01_Set_TxAddr( &addr[0], 5 ); //设置TX地址
658     NRF24L01_Set_RxAddr( 0, &addr[0], 5 ); //设置RX地址
659 }
660
661
662
663
```

寄存器参数设置：

06	RF_SETUP				RF Setup Register
	CONT_WAVE	7	0	R/W	Enables continuous carrier transmit when high.
	Reserved	6	0	R/W	Only '0' allowed
	RF_DR_LOW	5	0	R/W	Set RF Data Rate to 250kbps. See RF_DR_HIGH for encoding.
	PLL_LOCK	4	0	R/W	Force PLL lock signal. Only used in test
	RF_DR_HIGH	3	1	R/W	Select between the high speed data rates. This bit is don't care if RF_DR_LOW is set. Encoding: [RF_DR_LOW, RF_DR_HIGH]: '00' - 1Mbps '01' - 2Mbps '10' - 250kbps '11' - Reserved
	RF_PWR	2:1	11	R/W	Set RF output power in TX mode '00' - -18dBm '01' - -12dBm '10' - -6dBm '11' - 0dBm
	Obsolete	0			Don't care

例如参数为 0x26:空速为 250kbps, 发射功率为 0dBm。