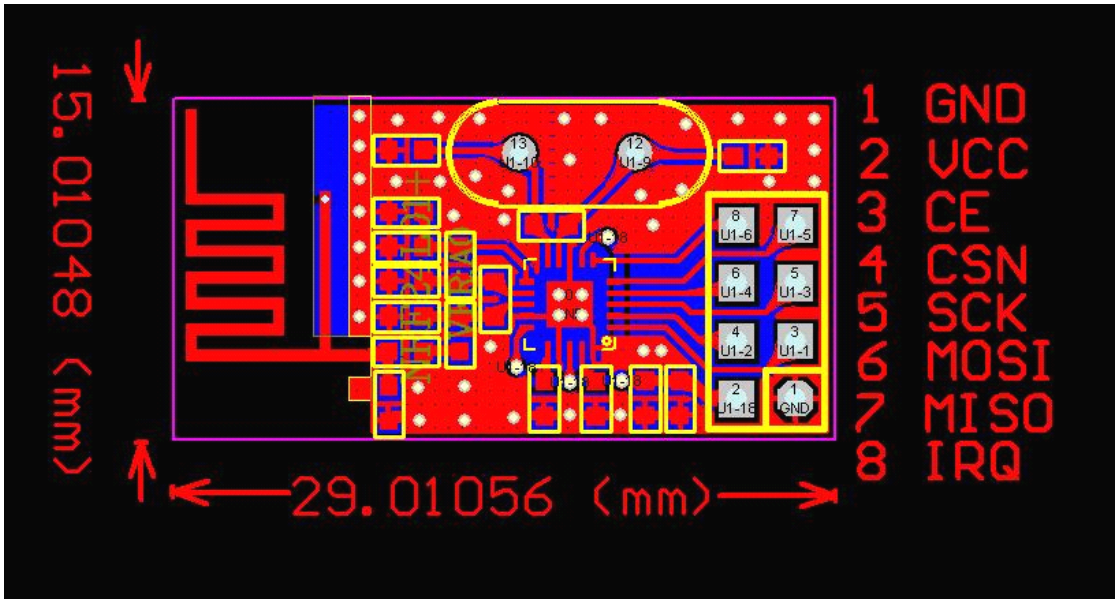
1. Module description

1. 2.4Ghz ISM band open license and use for free
2. The maximum working rate is 2Mbps, with efficient GFSK modulation, strong anti-interfering capability
3. 126 channels, meet the requirement of multi-point communication and frequency hopping communication
4. Built-in hardware CRC error detection and one-to-multipoint communication control
5. Low power supply1.9 - 3.6V; 22uA current in the standby mode; 900nA in power-down mode
6. Built-in 2.4Ghz antenna, size of 5mm x 29mm
7. Module's address can be set via software. Data will be output only when receiving the its own address (interrupt instruction). It can be directly controlled by various SCM, and programming is very easy
8. Built-in special regulator circuit, which guarantees good communication even using a variety of power supplies such as DC / DC switching power supply
9. 2.54MM DIP interfaces,
10. It works in the Enhanced ShockBurst with Automatic packet handling, Auto packet transaction handling, and optional built-in package response mechanism, which reduces packet loss rate.
11. While connecting with P0 port of 51 MCU series, 10K pull-up resistor is needed. This is not the case for other ports.
12. While working with other series of MCU, if the voltage is 5V, please refer to the output current of MCU IO port. If more than 10mA, it requires resistance as voltage divider. If it is 3.3V, MCU can be directly connected with RF24l01 module's IO port. For example, AVR is 5V, resistance of 2K should be connected in series.

2. Interface circuit

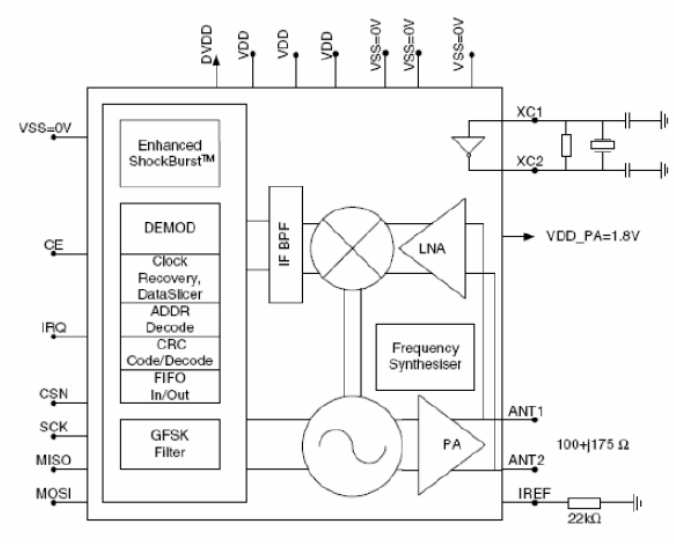


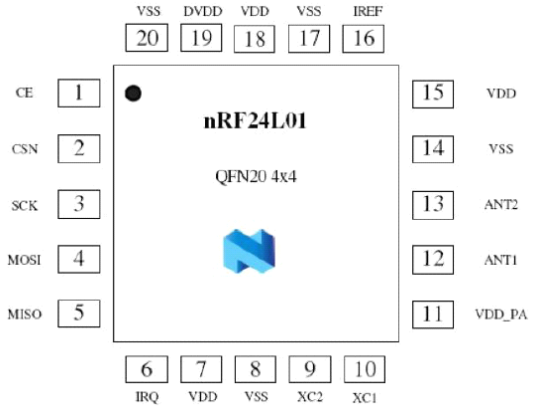
Description:

1. VCC pin voltage is 1.9V ~ 3.6V. If more than 3.6V, the module will be destroyed. Recommended voltage is about 3.3V.
2. Except VCC and GND, other pins can be directly connected with IO port of 5V MCU, without voltage conversion. Of course, 3V is more suitable.
3. MCU without SPI can also control the module. MCU's ordinary IO port can simulate SPI. You can use serial port or ordinary IO port to do it.
4. If other kinds of interfaces are needed, such as intensive DIP, you can contact us to custom it.

3. Module structure and pin description

NRF24L01 module uses Nordic's nRF24L01 chip.





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 Pin2
10	XC1	Analog Input	Crystal Pin1
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 DC-coupling pruposes
20	VSS	Power	Ground (0V)

4. Work mode

NRF2401 There are four models to work:

- Transceiver Mode
- Configuration Mode
- Idle mode
- Shutdown mode

Working model depends on PWR_UP register, PRIM_RX register and CE , as the table below:

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 done
Standby-II	1	0	1	TX FIFO empty
Standby-I	1	-	0	No ongoing packet transmission
Power Down	0	-	-	-

4.1 Transceiver mode

Transceiver mode Enhanced ShockBurstTM receiving and transmission mode, ShockBurstTM receiving and transmission mode and direct receiving and transmission mode, which depends on the setting register. More detail in device configuration part of this manual.

4.1.1 Enhanced ShockBurstTM transceiver mode

In Enhanced ShockBurstTM mode, the FIFO stack in the chip is used. Data is send to MCU in low speed, but is send out from MCU at high-speed (1Mbps), which can save energy as much as possible. So data transmission rate can be very high even working with low-speed MCU. All high-speed signal processing associated with RF protocol is done within the chip. In this way, it has three advantages:

- Minimize energy use;
- Low system cost (low-speed microprocessor can carry out high-speed RF transmission);
- Data transmission in the air is done in a short time, with high anti-interference ability.

Enhanced ShockBurstTM technology also reduces the average operating current of the system.

In Enhanced ShockBurstTM mode, NRF24L01 automatically processes the prefix and CRC code. While receiving data, it automatically removes CRC check code and the prefix. And while sending data, it automatically add prefix and CRC checksum code. In sending mode, set CE high, at least 10us.

4.1.1.1 Enhanced ShockBurstTM sending process

A. Receiver's address and data is sent into NRF24L01 by time series;

B. Configure CONFIG register to transmission mode.

C. MCU set CE high (at least for 10us), to make NRF24L01 do Enhanced ShockBurstTM emission;

D. Enhanced ShockBurstTM emission:

- power RF front-end;
- process RF data package (plus prefix, CRC check code);

- transmit data packets at high speed;
- NRF24L01 stays in idle state after transmission is done.

4.1.1.2 Enhanced ShockBurstTM receiving process

A. Configure the machine address and receiving packet's size;

B. Configuration CONFIG register to receive mode, and set CE high.

C. 130us later, NRF24L01 comes into surveillance state, waiting for the arrival of packet;

D. Upon receiving correct data package (the correct address and CRC code), NRF2401 automatically removes the prefix, address and CRC code;

E. NRF 24L01 inform MCU by setting RX_DR bit in STATUS register (STATUS generate disruption to MCU) ;

F. MCU read data from NewMsg_RF2401;

G. After reading all the data, clear the STATUS register. NRF2401 comes into one of 4 main modes.

4.1.2 ShockBurstTM transmission and receiving mode

ShockBurstTM mode is compatible with Nrf2401a, 02, E1 and E2. Please refer to the N-RF2401 document.

4.2 Idle Mode

NRF24L01 idle mode can reduce the average operating current. It can achieve energy efficiency while reducing the chip start-up time. In idle mode, most crystal oscillators are still working. And the operating current has no relation with the external crystal frequency.

4.3 Shutdown Mode

In shutdown mode, in order to get operating current minimum, the current is 900nA. the configuration content will be stored in NRF2401 chip, which is the main difference with the power-down situation.

5. Configuring NRF24L01 module

all the configuration work of NRF2401 is completed by SPI. It has 30-byte configuration registers. We recommend to make NRF24L01 working in Enhanced ShockBurstTM mode, because in this mode, programming will be easier, and working will be more stable.

ShockBurstTM configuration register makes NRF24L01 be able to handle RF protocol. The receiving mode and transmission mode switch process can be done by only changing a one-byte register.

ShockBurstTM configuration register can be divided into four parts:

- Data width: specify bits occupied by RF data, which make NRF24L01 distinguish received packet data and CRC check code;
- Address Width: specify bits occupied by the address in RF data packet, which makes NRF24L01 distinguish address and data;
- Address: address of receive data. There are addresses from channels 0 to channel 5;
- CRC: it makes NRF24L01 generate CRC code or decode CRC.

When using the CRC technology of NRF24L01 chip, make sure that in configuration register (EN_CRC of CONFIG) the CRC check is enabled, and sending and receiving use the same protocol.

Descriptions of NRF24L01 CONFIG register configuration bit is as follows.

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_DS as active low interrupt on the IRQ pin
	MASK_RX_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

6. Reference code

```
#include <reg51.h>
//<nRF2401_Pins >
sbit MISO =P1^3;
sbit MOSI =P1^4;
sbit SCK =P1^5;
sbit CE =P1^6;
sbit CSN =P3^7;
sbit IRQ =P1^2;
sbit LED2 =P3^5;
sbit LED1 =P3^4;
sbit KEY1 =P3^0;
sbit KEY2 =P3^1;
// SPI(nRF24L01) commands
#define READ_REG 0x00 // Define read command to register
#define WRITE_REG 0x20 // Define write command to register
#define RD_RX_PLOAD 0x61 // Define RX payload register address
#define WR_TX_PLOAD 0xA0 // Define TX payload register address
#define FLUSH_TX 0xE1 // Define flush TX register command
#define FLUSH_RX 0xE2 // Define flush RX register command
#define REUSE_TX_PL 0xE3 // Define reuse TX payload register command
#define NOP 0xFF // Define No Operation, might be used to read status register
//*************************************************//
// SPI(nRF24L01) registers(addresses)
#define CONFIG 0x00 // 'Config' register address
#define EN_AA 0x01 // 'Enable Auto Acknowledgment' register address
#define EN_RXADDR 0x02 // 'Enabled RX addresses' register address
#define SETUP_AW0x03 // 'Setup address width' register address
#define SETUP_RETR 0x04 // 'Setup Auto. Retrans' register address
#define RF_CH 0x05 // 'RF channel' register address
#define RF_SETUP 0x06 // 'RF setup' register address
```

```
#define STATUS 0x07 // 'Status' register address
#define OBSERVE_TX 0x08 // 'Observe TX' register address
#define CD 0x09 // 'Carrier Detect' register address
#define RX_ADDR_P0 0x0A // 'RX address pipe0' register address
#define RX_ADDR_P1 0x0B // 'RX address pipe1' register address
#define RX_ADDR_P2 0x0C // 'RX address pipe2' register address
#define RX_ADDR_P3 0x0D // 'RX address pipe3' register address
#define RX_ADDR_P4 0x0E // 'RX address pipe4' register address
#define RX_ADDR_P5 0x0F // 'RX address pipe5' register address
#define TX_ADDR 0x10 // 'TX address' register address
#define RX_PW_P0 0x11 // 'RX payload width, pipe0' register address
#define RX_PW_P1 0x12 // 'RX payload width, pipe1' register address
#define RX_PW_P2 0x13 // 'RX payload width, pipe2' register address
#define RX_PW_P3 0x14 // 'RX payload width, pipe3' register address
#define RX_PW_P4 0x15 // 'RX payload width, pipe4' register address
#define RX_PW_P5 0x16 // 'RX payload width, pipe5' register address
#define FIFO_STATUS 0x17 // 'FIFO Status Register' register address
//----------------------------------------------------------------
// write a byte to 24L01 and read one byte
uchar SPI_RW(uchar byte)
{
uchar bit_ctr;
for(bit_ctr=0;bit_ctr<8;bit_ctr++) // output 8-bit
{
MOSI = (byte & 0x80); // output 'byte', MSB to MOSI
byte = (byte << 1); // shift next bit into MSB..
SCK = 1; // Set SCK high..
byte |= MISO; // capture current MISO bit
SCK = 0; // ..then set SCK low again
}
return(byte); // return read byte
}
// write one byte to register reg and return status byte
uchar SPI_RW_Reg(BYTE reg, BYTE value)
{
uchar 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 byte
}
// read data
uchar SPI_Read_Buf(BYTE reg, BYTE *pBuf, BYTE bytes)
```

```
{
uchar status,byte_ctr;
CSN = 0; // Set CSN low, init SPI tranaction
status = SPI_RW(reg); // Select register to write to and read status byte
for(byte_ctr=0;byte_ctr<bytes;byte_ctr++)
pBuf[byte_ctr] = SPI_RW(0); //
CSN = 1;
return(status); // return nRF24L01 status byte
}
// write data
uchar SPI_Write_Buf(BYTE reg, BYTE *pBuf, BYTE bytes)
{
uchar status,byte_ctr;
CSN = 0;
status = SPI_RW(reg);
for(byte_ctr=0; byte_ctr<bytes; byte_ctr++) //
SPI_RW(*pBuf++);
CSN = 1; // Set CSN high again
return(status); //
}
// receiving function, return 1 while receiving is done
unsigned char nRF24L01_RxPacket(unsigned char* rx_buf)
{
unsigned char revale=0;
// set in RX mode
SPI_RW_Reg(WRITE_REG + CONFIG, 0x0f); // Set PWR_UP bit, enable CRC(2 bytes) &
Prim:RX. RX_DR enabled..
CE = 1; // Set CE pin high to enable RX device
dalay130us();
sta=SPI_Read(STATUS); // read register STATUS's value
if(RX_DR) // if receive data ready (RX_DR) interrupt
{
CE = 0; // stand by mode
SPI_Read_Buf(RD_RX_PLOAD,rx_buf,TX_PLOAD_WIDTH);// read receive payload from
RX_FIFO buffer
revale =1;
}
SPI_RW_Reg(WRITE_REG+STATUS,sta);// clear RX_DR or TX_DS or MAX_RT interrupt
flag
return revale;
}
// sending function
void nRF24L01_TxPacket(unsigned char * tx_buf)
{
```

```
CE=0;
//SPI_Write_Buf(WRITE_REG + TX_ADDR, TX_ADDRESS, TX_ADR_WIDTH); // Writes
TX_Address to nRF24L01
//SPI_Write_Buf(WRITE_REG + RX_ADDR_P0, TX_ADDRESS, TX_ADR_WIDTH); //
RX_Addr0 same as TX_Adr for Auto.Ack
SPI_Write_Buf(WR_TX_PLOAD, tx_buf, TX_PLOAD_WIDTH); // Writes data to TX payload
SPI_RW_Reg(WRITE_REG + CONFIG, 0x0e); // Set PWR_UP bit, enable CRC(2 bytes) &
Prim:TX. MAX_RT & TX_DS enabled..
CE=1;
dalay10us();
CE=0;
}
// setting function
void nRF24L01_Config(void)
{
//initial io
CE=0; // chip enable
CSN=1; // Spi disable
SCK=0; // Spi clock line init high
CE=0;
SPI_RW_Reg(WRITE_REG + CONFIG, 0x0f); // Set PWR_UP bit, enable CRC(2 bytes) &
Prim:RX. RX_DR enabled..
SPI_RW_Reg(WRITE_REG + EN_AA, 0x01);
SPI_RW_Reg(WRITE_REG + EN_RXADDR, 0x01); // Enable Pipe0
SPI_RW_Reg(WRITE_REG + SETUP_AW, 0x02); // Setup address width=5 bytes
SPI_RW_Reg(WRITE_REG + SETUP_RETR, 0x1a); // 500us + 86us, 10 retrans...
SPI_RW_Reg(WRITE_REG + RF_CH, 0);
SPI_RW_Reg(WRITE_REG + RF_SETUP, 0x07); // TX_PWR:0dBm, Datarate:1Mbps,
LNA:HCURR
SPI_RW_Reg(WRITE_REG + RX_PW_P0, RX_PLOAD_WIDTH);
SPI_Write_Buf(WRITE_REG + TX_ADDR, TX_ADDRESS, TX_ADR_WIDTH);
SPI_Write_Buf(WRITE_REG + RX_ADDR_P0, TX_ADDRESS, TX_ADR_WIDTH); CE=1; //
}
```