2.4G Wireless nRF24L01p

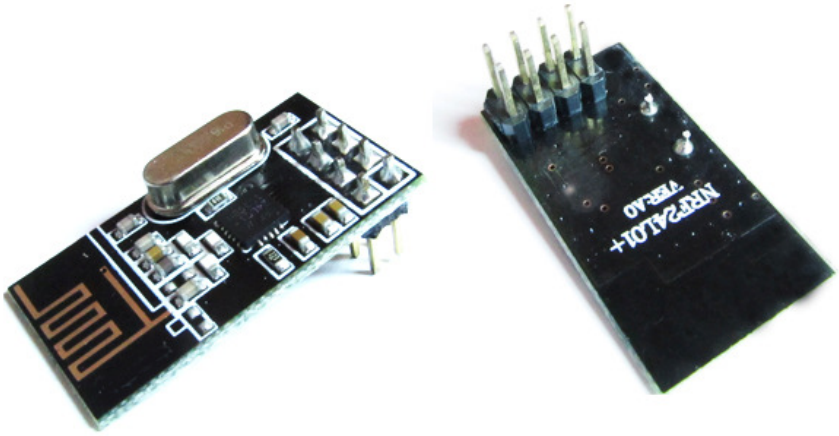
From Wiki

Contents

Introduction

The nRF24L01+(nRF24L01p) is a single chip 2.4GHz transceiver with an embedded baseband protocol engine (Enhanced ShockBurst™), suitable for ultra low power wireless applications. The nRF24L01+ is designed for operation in the world wide ISM frequency band at 2.400 - 2.4835GHz.
To design a radio system with the nRF24L01+, you simply need an MCU (microcontroller) and a few exter-nal passive components. The high air data rate combined with two power saving modes make the nRF24L01+ very suitable for ultra low power designs. nRF24L01+ is drop-in compatible with nRF24L01 and on-air compatible with nRF2401A, nRF2402, nRF24E1 and nRF24E2. Intermodulation and wideband blocking values in nRF24L01+ are much improved in comparison to the nRF24L01 and the addition of internal filtering to nRF24L01+ has improved the margins for meeting RF regulatory standards.

Model: RFM04 (http://www.elecfreaks.com/store/24g-wireless-nrf24l01p-p-118.html)

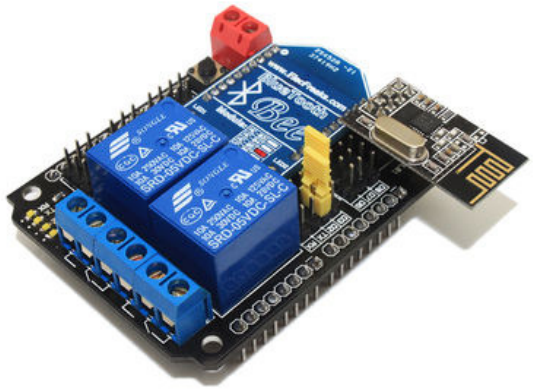


Feature

- Worldwide 2.4GHz ISM band operation， Free license to use.
- 126 RF channels.
- High air data rate: 250kbps, 1 and 2Mbps.
- Transmitter: 11.3mA at 0dBm output power.
- Receiver: Fast AGC for improved dynamic range.
- Receiver: Integrated channel filters.
- Enhanced ShockBurst™:1 to 32 bytes dynamic payload length,6 data pipe MultiCeiver™ for 1:6 star networks.
- Host Interface:4-pin hardware SPI,3 separate 32 bytes TX and RX FIFOs.
- Low Power Management:1.9 to 3.6V supply range.
- GFSK modulation.
- Auto packet transaction handling.
- Easy for designed.
- Small size:15mm*29mm.

Application Ideas

- Wireless PC Peripherals
- Mouse, keyboards and remotes
- 3-in-1 desktop bundles
- Advanced Media center remote controls
- VoIP headsets
- Game controllers
- Sports watches and sensors
- RF remote controls for consumer electronics
- Home and commercial automation
- Ultra low power sensor networks
- Active RFID
- Asset tracking systems
- Toys
- 2 Channel Relay Sheild (http://www.elecfreaks.com/wiki/index.php?title=2_channel_Relay_Shield_For_Arduino_(With_XBee/BTBee_interface))

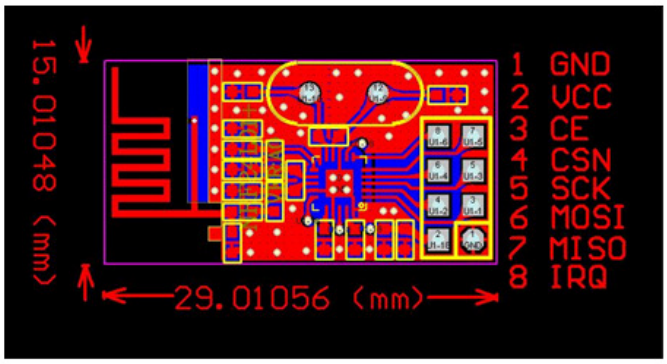


Cautions

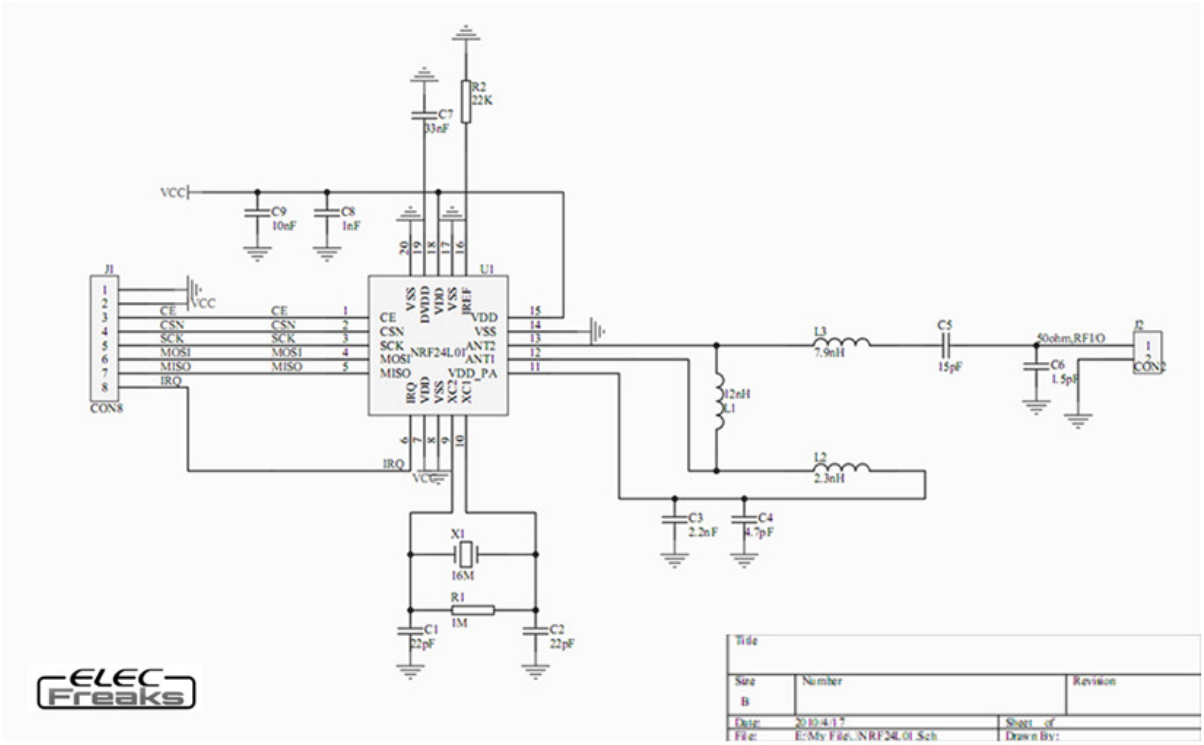
- **VCC is range of 1.9V~3.6V, do not exceed this range, otherwise it by destroyed the module.**
- **Except of VCC and GND，the other pins can be direct connected to 5V Microprocessor's IO, needless level converter. Of course 3V Microprocessor's IO more applied.**
- **If the Microprocessor isn't hardware SPI interface,could be simulated with ordinary IO.**
- **If connected to 51 series MCU's P0, there is a 10K Pull-up resistor need，others Port needless.**
- **If Microprocessor IO more than 10mA，need connect 2K resistive divider. It could direct connected to AVR Series.**

Schematic

Board Schematic



Schematic Diagram



Specification

Specification	Value
PCB Size	15mm*29mm*0.8mm
Power supply	1.9V~3.6V
Working current	13.5mA at 2Mbps / 11.3mA at 0dBm output power
IO counts	8
Sensitivity	-85dBm at 1Mbps
Emission distance	70~100 meter at 256kbps
Data rate	256kbps / 1Mbps / 2Mbps
Commuinitcation mode	Enhanced ShockBurst TM / ShockBurst TM
Working mode	Power Down Mode / Standby Mode / RX Mode / TX Mode
Temperatures	Operating:-40°C ~ 85°C / Storage:-40°C ~ 125°C

Pin definition and Rating

1 GND	2 VCC
3 CE	4 CSN
5 SCK	6 MOSI
7 MISO	9 IRQ

Mechanic Dimensions

Usage

- Enhanced ShockBurst™

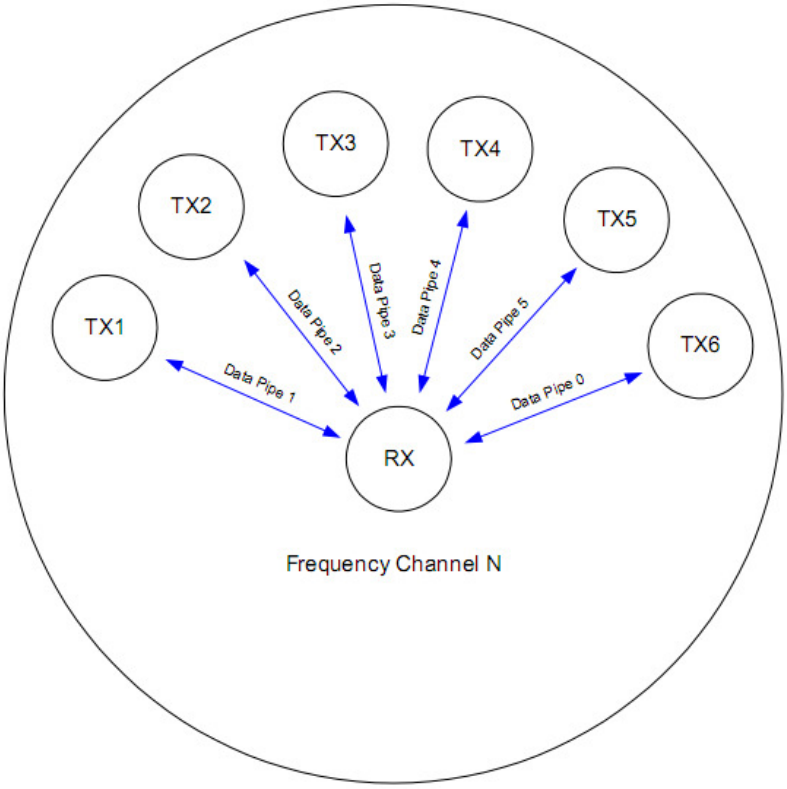
The Enhanced ShockBurst™ features enable significant improvements of power efficiency for bi-directional and uni-directional systems, without adding complexity on the host controller side. In Enhanced ShockBurst™ it is possible to configure parameters such as the

maximum number of retrans- mits and the delay from one transmission to the next retransmission. All automatic handling is done without the involvement of the MCU. it could be for wireless mouse wireless keyboard.

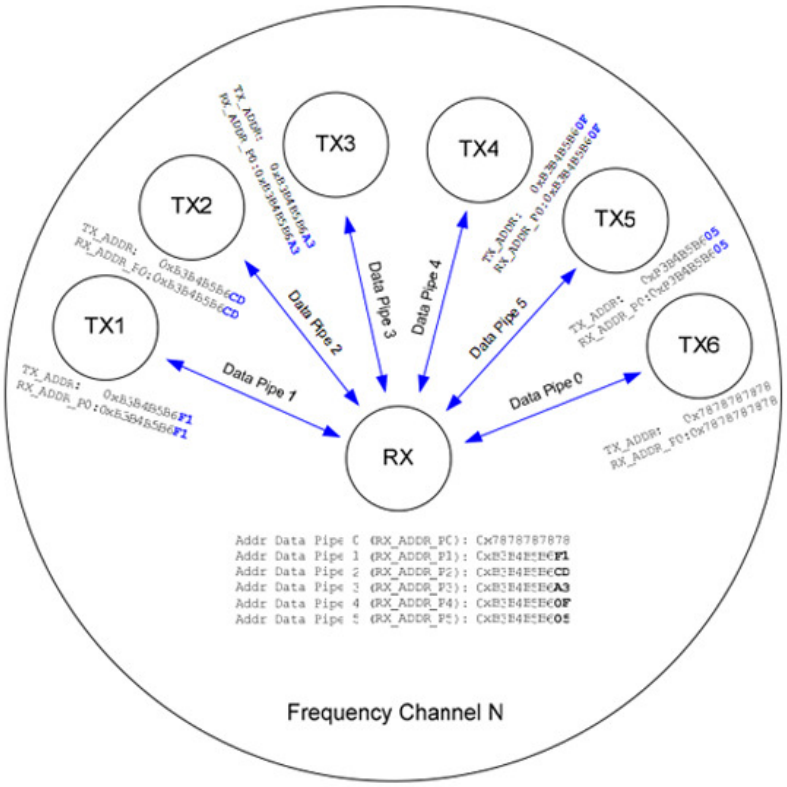
- ShockBurst™

ShockBurst™ makes it possible to use the high data rate offered by nRF24L01 without the need of a costly, high-speed microcontroller (MCU) for data processing/clock recovery.In ShockBurst™ TX, nRF24L01 automatically generates preamble and CRC, it compatable with nRF2401A, nRF24E1, nRF2402 and nRF24E2 communication.

If you want use 6 data pipe MultiCeiver™ for 1:6 star networks, selection Enhanced ShockBurst™



nRF24L01 in a star network configuration.
An nRF24L01 configured as primary RX (PRX) will be able to receive data trough 6 different data pipes, see Figure 4. A data pipe will have a unique address but share the same frequency channel. This means that up to 6 different nRF24L01 configured as primary TX (PTX) can communicate with one nRF24L01 configured as PRX, and the nRF24L01 configured as PRX will be able to distinguish between them. Data pipe 0 has a unique 40 bit configurable address. Each of data pipe 1-5 has an 8 bit unique address and shares the 32 most significant address bits. All data pipes can perform full Enhanced ShockBurst™ functionality. nRF24L01 will use the data pipe address when acknowledging a received packet. This means that nRF24L01 will transmit ACK with the same address as it receives payload at. In the PTX device data pipe 0 is used to received the acknowledge, and therefore the receive address for data pipe 0 has to be equal to the transmit address to be able to receive the acknowledge. See Figure 5 for addressing example.

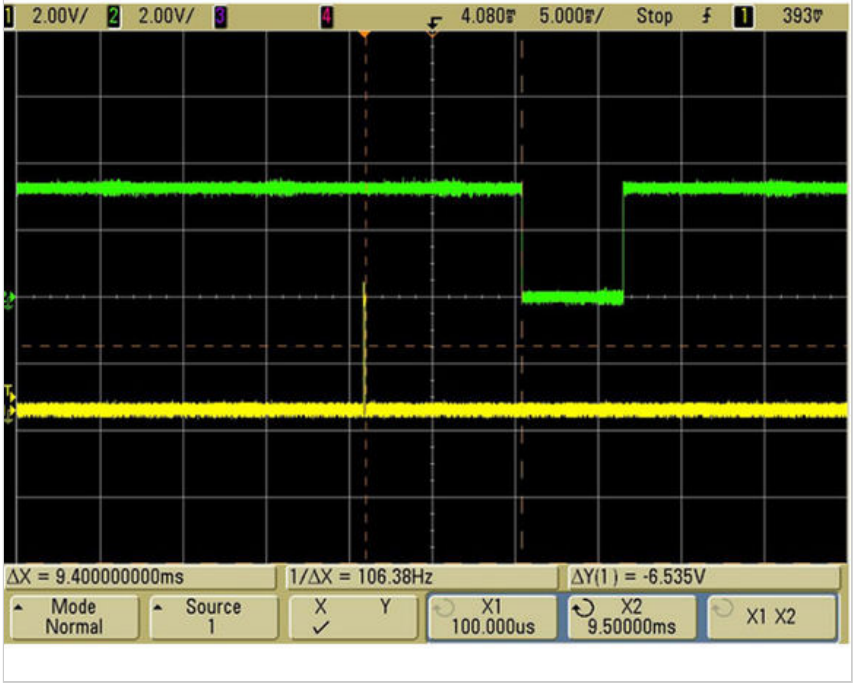


Example on how the acknowledgement addressing is done.
An nRF24L01 configured as PTX with Enhanced ShockBurst™ enabled, will use the ShockBurst™ feature to send a packet whenever the microcontroller wants to. After the packet has been transmitted, nRF24L01 will switch on its receiver and expect an acknowledgement to arrive from the terminating part. If this acknowledgement fails to arrive, nRF24L01 will retransmit the same packet until it receives an acknowledgement or the number of retries exceeds the number of allowed retries given in the SETUP_RETR_ARC register. If the number of retries exceeds the number of allowed retries, this will show in the STATUS register bit MAX_RT and gives an interrupt. Whenever an acknowledgement is received by an nRF24L01 it will consider the last transmitted packet as delivered. It will then be cleared from the TX FIFO, and the TX_DS IRQ source will be set high. With Enhanced ShockBurst™ nRF24L01 offers the following benefits:

- Highly reduced current consumption due to short time on air and sharp timing when operating with acknowledgement traffic
- Lower system cost. Since the nRF24L01 handles all the high-speed link layer operations, like re-transmission of lost packet and generating acknowledgement to received packets, it is no need for hardware SPI on the systemmicrocontroller to interface the nRF24L01. The interface can be done by using general purpose IO pins on a low cost microcontroller where the SPI is emulated in firmware. With the nRF24L01 this will be sufficient speed even when running a bi-directional link.
- Greatly reduced risk of “on-air” collisions due to short time on air
- Easier firmware development since the link layer is integrated on chip

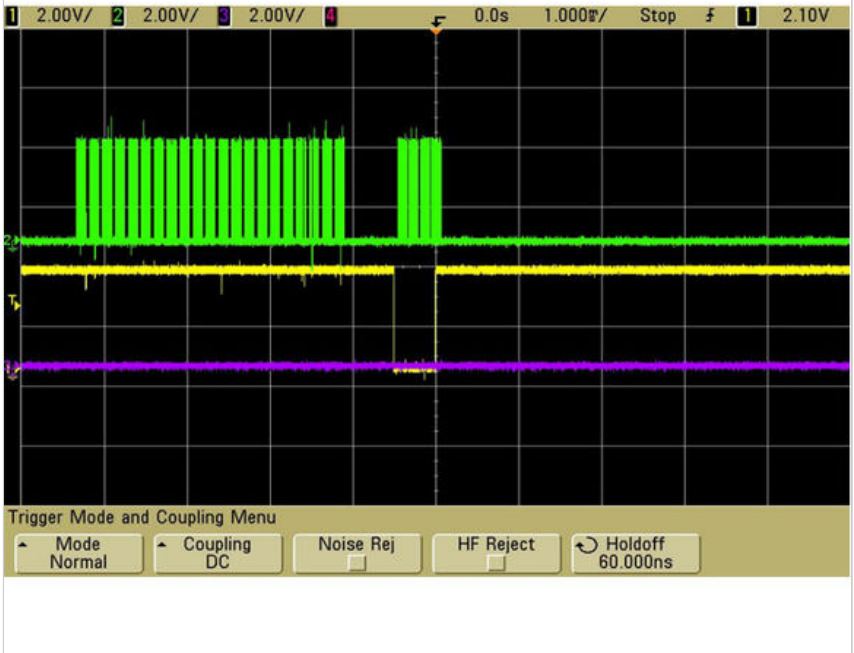
The actual communication process oscilloscope diagram

For nRF24L01 programming, mainly through the command (WRITE_REG, READ_REG, etc). Control line CE, CSN and the interrupt signal IRQ completed. Be Transmitte node, enable ACK and IRQ, after communicate successfull(received ACK from Accept node)IRQ to Low. Be Accpet node, disable ACK and IRQ,after communicate successfull(Based on Enhanced ShockBurst agreement that the successful receipt of a valid data width data) IRQ to Low. For more detail, we captured oscilloscope schematic:



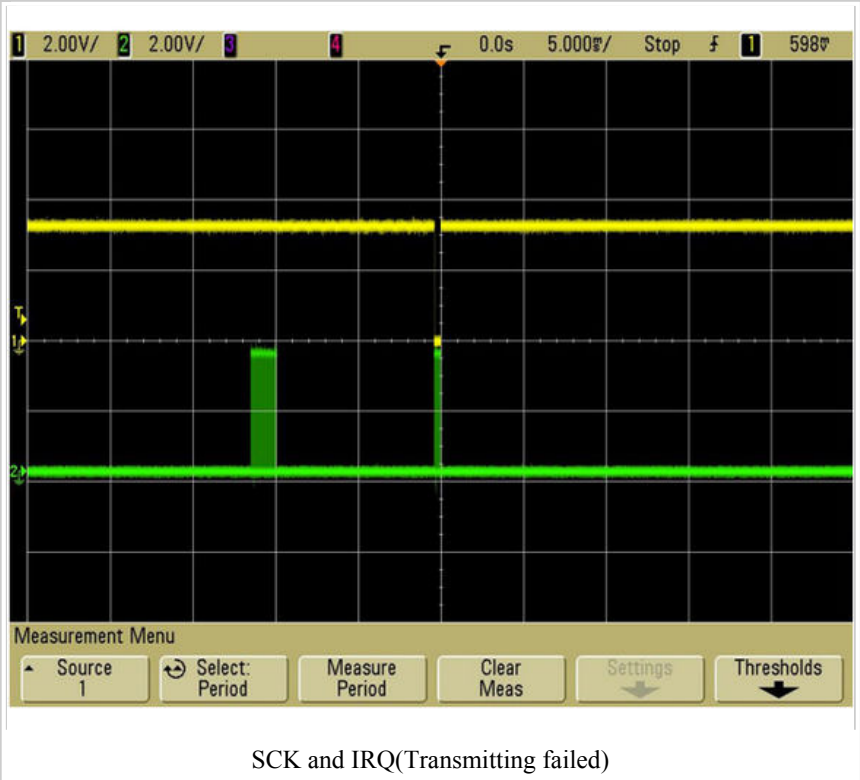
Transmitting node CE and IRQ

The Yellow sigal is CE and Green sigal is IRQ. After setting module to Transmitting node. Then send data by SPI_Write_Buf (WRITE_REG + RX_ADDR_P0, TX_ADDRESS, TX_ADR_WIDTH) to FIFO. CE keep 10us， the data be seed through wireless. If enable IQR(TX_DS,RX_DS,MAX_RT), when the Transmitting node receive the ACK from Accpeting Node or reach MAX transmitte count. IRQ to Low and the same time the CONFIG flag(TX_DS,RX_DS,MAX_RT) to 1, clear the flag SPI_RW_Reg (WRITE_REG+STATUS,status); // clear RX_DR or TX_DS or MAX_RT interrupt flag and the IQR will to High level. As the figure show, after CE set High Level 10ms, the IQR to Low level. The reason is reach to Max transimitte count(MAX_RT=1). So there are not Accpeting Node or they are different address.



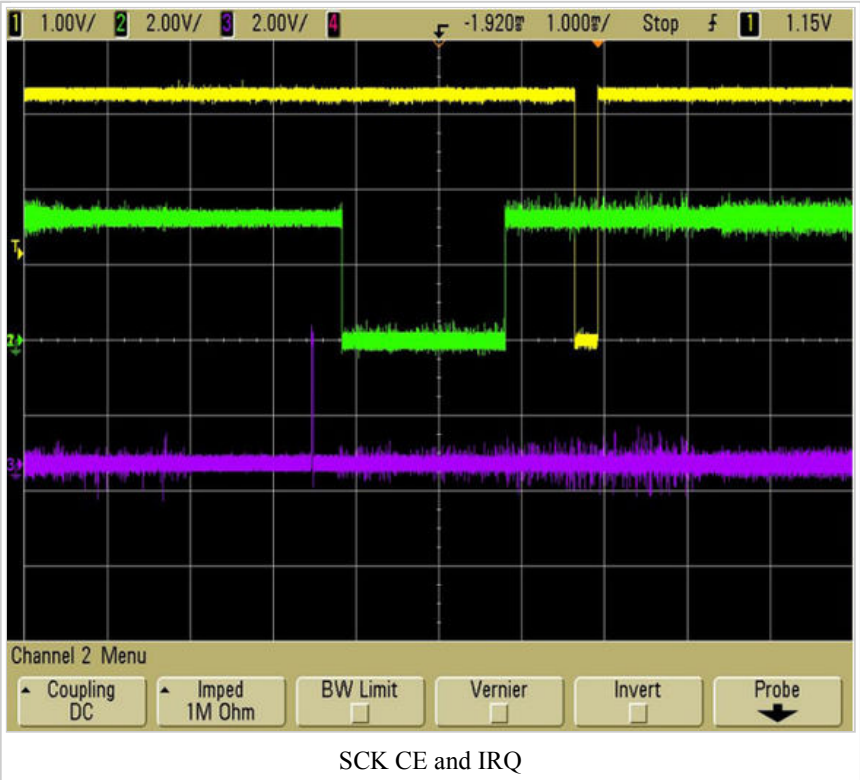
SCK and IRQ(Transmitting successfull)

The Green sigal is SCK, the Purple signal is CE, and the Yellow sigal is IRQ.The first part SCK is paired, when frist IRQ reach about 1ms. It is successfull paired and communication.



SCK and IRQ(Transmitting failed)

The Green sigal is SCK and the Yellow sigal is IRQ.The first part SCK is paired, when frist IRQ reach about 10ms. It is failed paired and communication.



SCK CE and IRQ

The Purple signal is CE, the Green sigal is Accpeting node's IRQ and the Yellow sigal is Transmitting node's IRQ. The figure means: After paired, CE is High level, and FIFO data will be send, then Accpeting node received the data, IRQ to Low level(Purple and green signal from the time interval between signals can determine the success of communication),then Accept node auto send ACK(enable ACK),then Transmitting node received ACK and set IQR Low level.Different communication environment may cause the Transmitting node and Accpeting node phase different.This is mainly due to the different communication environment receiver sends ACK signal to be re-issued several times in order to be received by the sender.

Programming

Includes important code snippet. Demo code like :
The Demo pins to Arduino as below:

GND – GND, VCC – 3.3V, CS – D8, CSN – D9, SCK – D10, MOSI – D11, MISO – D12, IRQ – D13

Download the code below into the **TX Arduino** (transmit) — This code will drive the nRF24L01 module to send out data form 0×00 to 0xFF .

Note: between the write TX_FIFO and clear RX_DR or TX_DS or MAX_RT interrupt flag, would better not serial print anything, which maybe case ACK failed.

```
void setup()
{
   SPI_DIR = ( CE + SCK + CSN + MOSI);
   SPI_DIR &=~ ( IRQ + MISO);
   //  attachInterrupt(1, _ISR, LOW);// interrupt enable
   Serial.begin(9600);
   init_io();                          // Initialize IO port
   unsigned char status=SPI_Read(STATUS);
   Serial.print("status = ");
   Serial.println(status,HEX);      // read the mode's status register, the default value shou
   Serial.println("********************TX_Mode Start*****************************");
   TX_Mode();                          // set TX mode
}
void loop()
{
   int k = 0;
   for(;;)
   {
     for(int i=0; i<32; i++)
         tx_buf[i] = k++;
     unsigned char status = SPI_Read(STATUS);                  // read register STATUS's value
     if(status&TX_DS)                                          // if receive data ready (TX_DS)
     {
       SPI_RW_Reg(FLUSH_TX,0);
       SPI_Write_Buf(WR_TX_PLOAD,tx_buf,TX_PLOAD_WIDTH);      // write playload to TX_FIFO
     }
     if(status&MAX_RT)                                         // this is retransmit than  SETUE
     {
       SPI_RW_Reg(FLUSH_TX,0);
       SPI_Write_Buf(WR_TX_PLOAD,tx_buf,TX_PLOAD_WIDTH);      // disable standy-mode
     }
     SPI_RW_Reg(WRITE_REG+STATUS,status);                      // clear RX_DR or TX_DS or MAX_RT
     delay(1000);
   }
}
```

Download the code below into the **RX Arduino** (receive) – This code will drive the nFR24L01 module to receive the data that transmit form the TX module and print it to serial port.

Note: clear RX_FIFO must bellow Read_FIFO

```
void setup()
{
   SPI_DIR = ( CE + SCK + CSN + MOSI);
   SPI_DIR &=~ ( IRQ + MISO);
   //  attachInterrupt(1, _ISR, LOW); // interrupt enable
   Serial.begin(9600);
   init_io();                          // Initialize IO port
   unsigned char status=SPI_Read(STATUS);
   Serial.print("status = ");
   Serial.println(status,HEX);      // read the mode's status register, the default value shou
   Serial.println("*****************RX_Mode start*****************************R");
   RX_Mode();                          // set RX mode
}
void loop()
{
   for(;;)
   {
     unsigned char status = SPI_Read(STATUS);                // read register STATUS's value
     if(status&RX_DR)                                        // if receive data ready (TX_DS)
     {
       SPI_Read_Buf(RD_RX_PLOAD, rx_buf, TX_PLOAD_WIDTH);    // read playload to rx_buf
       SPI_RW_Reg(FLUSH_RX,0);                               // clear RX_FIFO
       for(int i=0; i<32; i++)
       {
           Serial.print(" ");
           Serial.print(rx_buf[i],HEX);                      // print rx_buf
       }
       Serial.println(" ");
     }
     SPI_RW_Reg(WRITE_REG+STATUS,status);                    // clear RX_DR/TX_DS/MAX_RT inter
     delay(1000);
   }
}
```

- nRF24L01_Demo_For_Arduino (http://elecfreaks.com/store/download/nRF24L01_Demo_For_Arduino.zip)
- SPI_Demo_rf24L01 (http://elecfreaks.com/store/download/datasheet/rf/rf24l01/SPI_rf24L01.zip)

Example

The projects and application examples. These file are a sample code for your reference.

- STC2052 demo (http://elecfreaks.com/store/download/datasheet/rf/rf24l01/STC2052.zip)
- PIC 24L01 demo (http://elecfreaks.com/store/download/datasheet/rf/rf24l01/PIC_24L01.zip)
- MSP430F149-RF24L01 demo (http://elecfreaks.com/store/download/datasheet/rf/rf24l01/MSP430F149-RF24L01.zip)
- NRF24L01 one to six demo (http://elecfreaks.com/store/download/datasheet/rf/rf24l01/NRF24L01_one_to_six.zip)
- C51 for nrf24l01 (http://elecfreaks.com/store/download/datasheet/rf/rf24l01/C51_for_nrf24l01.zip)
- AVR48 demo (http://elecfreaks.com/store/download/datasheet/rf/rf24l01/AVR48.zip)
- AT89S52 source code demo (http://elecfreaks.com/store/download/datasheet/rf/rf24l01/AT89S52_source_code.zip)
- Assembly C51 for nRF24L01 demo (http://elecfreaks.com/store/download/datasheet/rf/rf24l01/Assembly_C51_for_nRF24L01.zip)

Bill of Materials (BOM) /parts list

All the components used to produce the product.

FAQ

Please list your question here:

Support

If you have questions or other better design ideas,

Version Tracker

Revision	Descriptions	Release
v1.1	Initial public release	date

Bug Tracker

Bug Tracker is the place you can publish any bugs you think you might have found during use. Please write down what you have to say, your answers will help us improve our

products.

Additional Idea

The Additional Idea is the place to write your project ideas about this product, or other usages you've found. Or you can write them on Projects page.

Resources

- Download nRF24L01P Datasheet. (http://elecfreaks.com/store/download/datasheet/rf/rf24l01_PA_LAN/nRF24L01P.PDF)
- nRF24L01 with Arduinio's SPI Library. (http://elecfreaks.com/store/download/datasheet/rf/rf24l01/SPI_rf24L01.zip)
- nRF24L01 Demo For Arduino (http://www.elecfreaks.com/203.html)
- Download the zip include all the file below here. (http://elecfreaks.com/store/download/NRF24L01_module.zip)

```
*nRF24L01_sch.pdf                          Schematic of the nRL24L01 module
*nRF24L01_Specification_v2_0.pdf           Datasheet of the nRL2401 chip
*NRF24L01.JPG                              Pin map of the nRF24L01 module
*Demo_80S52.rar                            Demo code for nRF24L01 module on 80S52
*acceptavrnrf24l01.rar                     Accept date demo for nRF24L01 module on AVR
*sendavrnrf24l01.rar                       Send data demo for nRF24L01 module on AVR
```

How to buy

Click here to buy: http://www.elecfreaks.com/store/24g-wireless-nrf24l01p-p-118.html

See Also

Other related products and resources.

Licensing

External Links

Links to external webpages which provide more application ideas, documents/datasheet or software libraries