TA0032 Instructions for RF Transceiver Module

1.Introduction

The nRF905 is a radio transceiver IC similar to the well-known nRF24L01, but operates at 433/898/915MHz instead of 2.4GHz, has a much longer range and a few extra IO pins. However, the nRF905 data rate is only 50Kbps compared to nRF24L01’s 2Mbps.

This library offers quite a bit of flexibility: Optional use of interrupts, 2 of the connections to the module are optional since their states can also be accessed by the ICs status register, and supports basic collision avoidance.

2. Pin Instructions

|  |  |  |
| --- | --- | --- |
| nRF905 Pin name | Arduino Uno pin connected | Pin Description |
| VCC | 3.3V | Power(3.3V) |
| CE | 7 | Standy-High=TX/RX mode,Low=standby |
| TXE | 9 | TX or Rx mode-High=TX,Low=RX |
| PWR | 8 | Power up – High = on, Low = off |
| CD | 2 | Carrier detect – High when a signal is detected, for collision avoidance |
| AM | - | Address Match – High when receiving a packet that has the same address as the one set for this device, optional since state is stored in register, not used by this library |
| DR | 3 | Data Ready – High when finished transmitting/High when new data received, optional since state is stored in register, if interrupts are used this pin must be connected **NOTE:** On Arduino Mega change INTERRUPT\_NUM to 5 in nRF905\_config.h and for Arduino Yun it should be changed to 0. |
| SO | 12 | SPI MISO |
| SI | 11 | SPI MOSI |
| SCK | 13 | SPI SCK |
| CSN | 10 | SPI SS |
| GND | GND | Ground |

3. Example

This is a example code about how use this module to communicate with wireless serial.

And other example or Library file are attached.

\*\*\*\*\*\*\*\*\*\*Code Begin\*\*\*\*\*\*\*\*\*

/\*

\* Wireless serial link

\*

\* 7 -> CE

\* 8 -> PWR

\* 9 -> TXE

\* 2 -> CD

\* 3 -> DR

\* 10 -> CSN

\* 12 -> SO

\* 11 -> SI

\* 13 -> SCK

\*/

#include <nRF905.h>

#include <SPI.h>

#define PACKET\_TYPE\_DATA 0

#define PACKET\_TYPE\_ACK 1

#define MAX\_PACKET\_SIZE (NRF905\_MAX\_PAYLOAD - 2)

typedef struct {

byte dstAddress[NRF905\_ADDR\_SIZE];

byte type;

byte len;

byte data[MAX\_PACKET\_SIZE];

} packet\_s;

void setup()

{

// Start up

nRF905\_init();

// Put into receive mode

nRF905\_receive();

Serial.begin(9600);

Serial.println(F("Ready"));

}

void loop()

{

packet\_s packet;

// Send serial data

byte dataSize;

while((dataSize = Serial.available()))

{

// Make sure we don't try to send more than max packet size

if(dataSize > MAX\_PACKET\_SIZE)

dataSize = MAX\_PACKET\_SIZE;

packet.type = PACKET\_TYPE\_DATA;

packet.len = dataSize;

// Copy data from serial to packet buffer

for(byte i=0;i<dataSize;i++)

packet.data[i] = Serial.read();

// Send packet

sendPacket(&packet);

// Receive mode

nRF905\_receive();

// Wait for ACK packet

byte startTime = millis();

while(1)

{

bool timeout = false;

while(1)

{

if(getPacket(&packet)) // Get new packet

break;

else if((byte)(millis() - startTime) > 50) // 50ms timeout

{

timeout = true;

break;

}

}

if(timeout) // Timed out

{

Serial.println(F("TO"));

break;

}

else if(packet.type == PACKET\_TYPE\_ACK) // Is packet type ACK?

break;

}

}

// Put into receive mode

nRF905\_receive();

// Wait for data

while(1)

{

if(getPacket(&packet) && packet.type == PACKET\_TYPE\_DATA) // Got a packet and is it a data packet?

{

// Print data

Serial.write(packet.data, packet.len);

// Reply with ACK

packet.type = PACKET\_TYPE\_ACK;

packet.len = 0;

sendPacket(&packet);

// Put into receive mode

nRF905\_receive();

}

else if(Serial.available()) // We've got some serial data, need to send it

break;

}

}

// Send a packet

static void sendPacket(void\* \_packet)

{

// Void pointer to packet\_s pointer hack

// Arduino puts all the function defs at the top of the file before packet\_s being declared :/

packet\_s\* packet = (packet\_s\*)\_packet;

// Convert packet data to plain byte array

byte totalLength = packet->len + 2;

byte tmpBuff[totalLength];

tmpBuff[0] = packet->type;

tmpBuff[1] = packet->len;

memcpy(&tmpBuff[2], packet->data, packet->len);

// Set address of device to send to

//nRF905\_setTXAddress(packet->dstAddress);

// Set payload data

nRF905\_setData(tmpBuff, totalLength);

// Send payload (send fails if other transmissions are going on, keep trying until success)

while(!nRF905\_send());

}

// Get a packet

static bool getPacket(void\* \_packet)

{

// Void pointer to packet\_s pointer hack

// Arduino puts all the function defs at the top of the file before packet\_s being declared :/

packet\_s\* packet = (packet\_s\*)\_packet;

byte buffer[NRF905\_MAX\_PAYLOAD];

// See if any data available

if(!nRF905\_getData(buffer, sizeof(buffer)))

return false;

// Convert byte array to packet

packet->type = buffer[0];

packet->len = buffer[1];

// Sanity check

if(packet->len > MAX\_PACKET\_SIZE)

packet->len = MAX\_PACKET\_SIZE;

memcpy(packet->data, &buffer[2], packet->len);

return true;

}

\*\*\*\*\*\*\*\*Code End\*\*\*\*\*\*\*