
Simple frequency hopping with the nRF9E5 and nRF24E1

1. Introduction

This paper presents a simple frequency-hopping protocol for the nRF9E5 and nRF24E1. A demo application is made for each device showing how to use the protocol.

The demo on the nRF9E5 is a simple temperature measurement application using the LM35 temperature sensor on the evaluation board. Two EVBOARDS are needed for the demo; one local connected to your PC via the serial port and one remote communicating with the local board using the frequency hopping protocol described below. A simple Visual Basic application is used to display the temperature on each board.

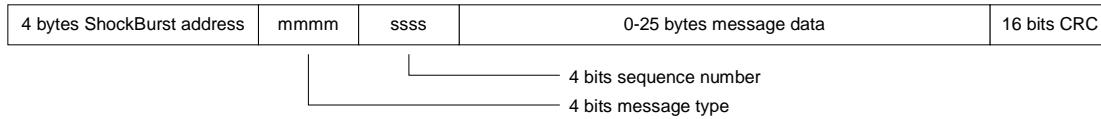
To demonstrate the protocol on the nRF24E1 a simple file transfer application is implemented on top of the frequency-hopping protocol for easy demonstration with a PC with two serial ports interfacing the serial ports on two EVBOARDS.

2. Protocol description

The frequency hopping code hops between 64 channels (channel 2 to channel 65) pseudo-randomly distributed in a 256-bytes constant table embedded in the code. The hopping functions as follows.

Each time the transmitter sends a packet on a channel it starts waiting for an acknowledge packet (ACK) on the same channel. If the ACK is received within a predefined time-out (3ms in this example) the transmitter selects the next channel in the hopping table and sends the next packet on the newly selected channel. If an ACK is not received within the time-out period the packet is re-transmitted on the same channel. If this does not result in a valid ACK the transmitter hops to the next channel and repeats the procedure above. This process is repeated until an ACK is received or a time-out (3 seconds in the demo applications) is reached and if the time-out is reached the function returns with an error. A simple float diagram of the **TransmitPacket** function can be found in the appendix.

The functionality on the receiver mirrors that of the transmitter. Each time a packet is received an ACK is sent before changing to the next channel in the hopping table. In addition to the address and CRC, each packet consists of 0-25 bytes of data and one byte of control information as shown in the figure below.



At present there are two message types defined: data message (type=0001b) and ACK (type=0010b). The ACK has 0 bytes message data in this application. The sequence bits contain a four bits counter that increments after each packet is sent, is used by the receiver to keep track of which packet it has received.

3. Application Programming Interface (API)

The frequency hopping code is completely self-contained and can easily be adapted to your own application. The hopping code is stored in the file **radio.c** and the API is declared in the file **radio.h**. A 1ms timer is also required. In the example code Timer0 is used and the timer routines can be found in the file **util.c**.

The API consists of the following functions:

```
void InitReceiver(unsigned char n, unsigned char *pAddr)
void InitTransmitter(unsigned char n, unsigned char *pAddr)
void ReceiverTimeout(unsigned tOut);
void TransmitterTimeout(unsigned tOut);
unsigned char TransmitPacket(unsigned char *pBuf)
unsigned char ReceivePacket(unsigned char *pBuf)
```

Here, **n** is the number of bytes (0-25) in the packet described above, **pAddr** is a pointer to a buffer containing four address bytes, **tOut** is the time-out in ms and **pBuf** is a pointer to a buffer of at least **n** unsigned characters.

The **ReceiverTimeout** function set how long in ms the receiver should wait for a packet before returning to the caller with an error and the **TransmitterTimeout** function sets how long the transmitter should try to get in contact with the receiver. The default time-out for both receiver and transmitter is 3000ms.

To use the hopping protocol, first call **InitReceiver** and **InitTransmitter** with the number of bytes (must be the same number for both receiver and transmitter) and a pointer to a four bytes array of address bytes, and then call **TransmitPacket** and **ReceivePacket** as appropriate. If you call **InitReceiver** and **InitTransmitter** with **pAddr=0** a default address will be used. Please note that the present code is half-duplex.

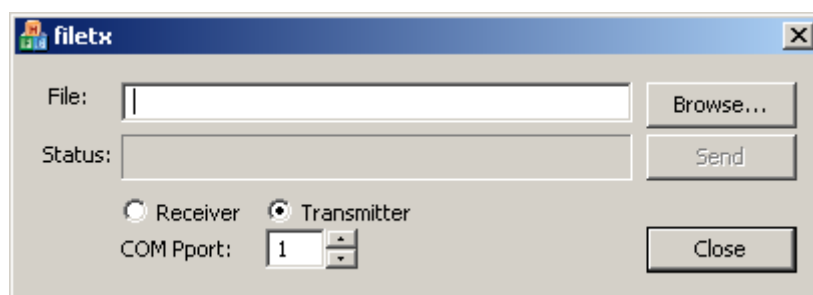
4. File transfer application with the nRF24E1

A simple file transfer application has been built on top of the hopping protocol. A packet length of maximum is used; 24 bytes of file data and one byte with the number of bytes used in the packet. A packet with the number of bytes set to zero signals end of file. The application functions as follows.

After initializing the UART and timers both evaluation boards start waiting for a character on the serial port. If the character is 'T' the **InitTransmitter** function of the hopping code is called with the number of bytes set to 25 and a file transfer is initiated. If the character is 'R' the **InitReceiver** function is called with the same number of bytes and a file reception is initiated. After this packets are exchanged with the PC on both sides with the following format: The character 'P', number of bytes **n** and **n** data bytes. A packet with **n** set to zero signifies end of file.

The file transfer application is called **filetx.exe** and can be found in the zip-file accompanying this document.

To test the file transfer application first program two nRF24E1 evaluation boards with the file **hoptest.hex** and connect the two evaluation boards to the serial port(s) on one or two PC's. Then start one instance of **filetx.exe** on each PC or two instances if you are using one PC with two serial ports. The main window of the **filetx** application is shown in the following figure.



Select the serial port(s) you are using then click the Receiver button in one of the windows and the Transmitter button in the other and hit the browse button in each instance and select a file to send and a filename for the file to receive. Finally hit the Receive button on the receiver (it is important to start the receiver first) then the Send button on the transmitter and the file transfer will start.

5. Temperature measurement with the nRF9E5

A simple temperature measurement application has been built on top of the hopping protocol. A packet length of 2 bytes is used. There is one receiver connected to the PC via the serial port and one remote transmitter. The receiver is selected by shorting P0.6 and P0.4 and the transmitter by shorting P0.6 and P0.5. The application functions as follows.

After initializing the UART the receiver starts waiting for a character on the serial port. If the character is 'L' the temperature sensor on the local EVBOARD is read and the result written to the serial port as four ASCII-hex digits. If the character is 'R' the receiver starts waiting for a radio packet containing the temperature from the remote EVBOARD. Again the result is written to the serial port as four ASCII-hex digits.

After initializing the transmitter runs in an endless loop reading the temperature sensor and sending the result in a RF packet to the receiver.

A simple Visual Basic client is included for testing the temperature measurement application. You need the VB6 runtime installed on your computer if you want to run the application. The VB6 can be downloaded from the following link:

<http://download.microsoft.com/download/vb60pro/Redist/sp5/WIN98Me/EN-US/vbrun60sp5.exe>

WHITE PAPER



APPENDIX

TransmitPacket

