CRC Communication Example

This chapter will guide the user to carry out RFM67 and RFM65 transmitting and receiving communication experiment through HopeDuino, show RF chip commonly used data message CRC structure.

1. Tools and software needed to be prepared

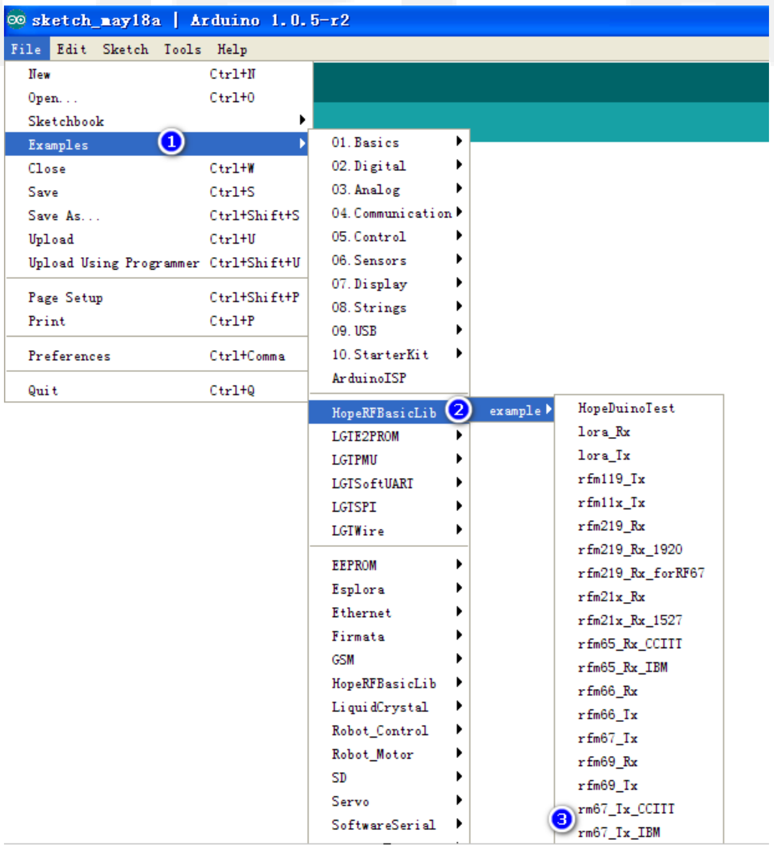
- Arduino IDE version 1.0.5
- HopeDuino board (two pieces)

  (If you have not used the HopeDuino board, please refer to the 《AN0002-HopeDuino Platform Construction Guideline》)
- USB cable（Type A to Type B）
- RFM67 module (or module based on RF67 chip design)
- RFM65 module (or module based on RF65 chip design)

2. Hands-on experiment

- Insert RFM67 and RFM65 modules (with conversion board) into the corresponding HopeDuino board.
- Connect the two HopeDuino boards to PC with USB cable.
- Open Arduino IDE interface, Click 【File】→【Examples】→【HopeRFBasicLib】→【example】→【rfm67_Tx_CCITT】 or 【rfm67_Tx_IBM】, as shown below.

⚠ Notice: You couldn't find [HopeRFLib] in [example] because you didn't install the HSP provided by HopeRF. Please refer to 《AN0002-HopeDuino Platform Construction Guideline》.



➢ Open Arduino IDE interface, Click 【File】→【Examples】→【HopeRFBasicLib】→【example】→【rfm65_Rx_CCITT】 or【rfm65_Rx_IBM】, as shown above.

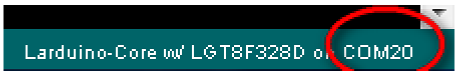
⚠ Notice: Transmitted and received files are corresponding, that is the choice of rfm67_Tx_CCITT, corresponding to the choice of rfm65_Rx_CCITT, and vice versa.

➢ At this time the Tx program and Rx program have been opened, please compile the download programs according to the corresponding COM port.

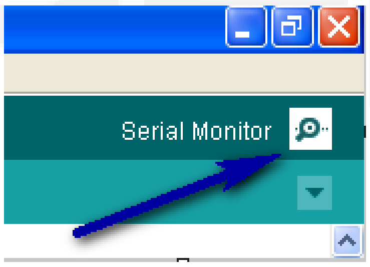
⚠ Notice:

1. Do not know how to compile the download code, please refer to 《AN0002-HopeDuino Platform Construction Guideline》

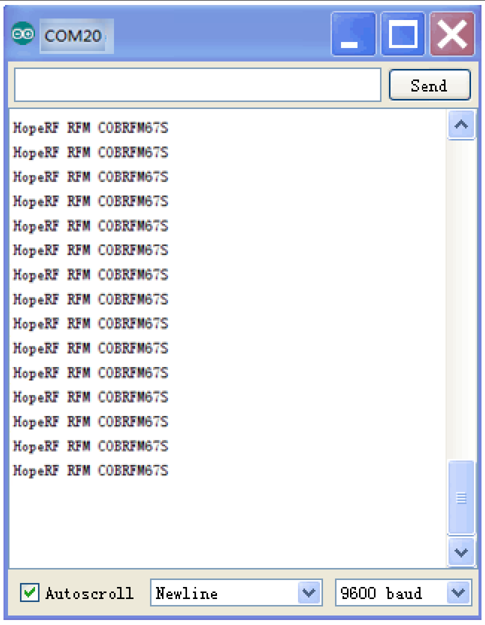
2. HopeDuino platform support multiple boards connected to the same PC. But you need to modify the parameters manually. So you need to pay special attention to which COM port when you download the program every time. COM port is in the lower right corner of Arduino interface, as shown below.



➢ After the two programs are downloaded, the Tx board will transmit a packet of data through module RFM67. The Rx board will receive a packet of data through module RFM65 periodically and upload the data to PC through UART (USB). At this point, you can set the COM of Arduino IDE as the port connected with Rx board. Open the "Serial Monitor" , as shown below.



➢ Click the "Serial Monitor"; pop up the serial port assistant interface, as shown below. Window will display the received data message.



```
HopeRF RFM COBRFM67S
HopeRF RFM COBRFM67S
HopeRF RFM COBRFM67S
HopeRF RFM COBRFM67S
HopeRF RFM COBRFM67S
HopeRF RFM COBRFM67S
HopeRF RFM COBRFM67S
HopeRF RFM COBRFM67S
HopeRF RFM COBRFM67S
HopeRF RFM COBRFM67S
HopeRF RFM COBRFM67S
HopeRF RFM COBRFM67S
HopeRF RFM COBRFM67S
HopeRF RFM COBRFM67S
HopeRF RFM COBRFM67S
```

Notice:

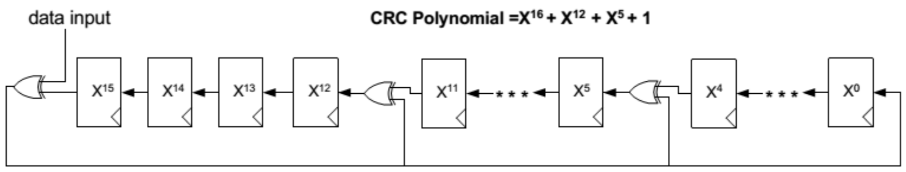
1. The receiving program enables UART library function. On the description of library function UART, please refer to the "HopeDuino_UART" library file. It is also stored in the HopeRFLib.

3. CRC16 brief introduction of common RF chip

Common RF chip data message generally uses of 16 bit CRC check. There are mainly two kinds of CRC-16 methods——IBM and CCITT, for example: RF65/69, as shown below:

CRC mode	Polynomials	Seed	Result inversed
CCITT	$X^{16}+X^{12}+X^5+1$	0x1D0F	true
IBM	$X^{16}+X^{15}+X^2+1$	0xFFFF	false

➢ For example: CCITT polynomial $X^{16}+X^{12}+X^5+1$, the block diagram is as follows.



➢ Pay attention to "seed calculated" and "result inversed". Because the seed calculated is different, it will lead to different CRC result. “Seed calculated” variable is CrcSeed in this program, “Result inversed” variable is CrcInverse in this program.

➢ As mentioned earlier CRC is the receiver chip used to confirm whether the data message is wrong. So the CRC method, the seed calculated and the result inversed, are determined by the receiver chip, which is RF65 in this program. And RF67 is a transmitter chip without CRC functions inside, only can be achieved by the program into MCU. These programs are stored in the “～\HopeRFLib\ HopeDuino_RFM67.cpp”and “～\HopeRFLib\ HopeDuino_RFM67.h”library files. There is a need for users to read and transplant.

3. Program Explanation

- rfm67_Tx_CCITT case explanation (rfm67_Tx_IBM is the same basically)

```
#include <HopeDuino_RFM67.h>                // Call the corresponding library file
rf67Class radio;                            // Define variable radio for rf67
byte str[21] = {'H','o','p','e','R','F',' ','R','F','M',' ','C','O','B','R','F','M','6','7','S'};
                                               //Message to be transmitted

void setup()
{
 radio.Modulation        = FSK;             //Modulation mode is FSK
 radio.Frequency         = 434000;              //Target frequency is 434MHz
 radio.OutputPower       = 10+13;            //Output power is 10dBm
 radio.PreambleLength = 16;                 //Preamble length is 16 bytes
 radio.FixedPktLength = false;             //Message length is variable
 radio.PayloadLength    = 21;               //Payload length is 21 bytes

 radio.CrcDisable        = false;            //Enable CRC
 radio.CrcInverse        = true;         // CRC calculation result is inversed for RF65 receiving demand.
 radio.CrcMode              = false;        // CRC uses CCITT polynomials.
                                               //Users need to use IBM will change here to true.
                                            // But need to pay attention to the initial value of CRC and
                                            //whether it needs to inverse the results.
 radio.CrcSeed              = 0x1D0F;    //CRC initial value is 0x1D0F (hexadecimal), corresponding to RF65
   radio.NodeDisable       = true;          //Disable Node mode
 //radio.NodeBefore      = false;
 //radio.NodeAddr     = 0x01;

 radio.SymbolTime        = 416000;           //Symbol time is 2.4Kbps.
 radio.Deviation           = 35;            // Frequency deviation is 35 KHz
 radio.SyncLength        = 3;               // Sync length is 3 bytes
 radio.SyncWord[0]       = 0xAA;            //Sync word is 0xAA2DD4
 radio.SyncWord[1]       = 0x2D;
 radio.SyncWord[2]       = 0xD4;
 radio.vInitialize();                    // Initialize radio
 radio.vGoStandby();                         // Enter standby mode
}

void loop()
{
 radio.bSendMessage(str, 21);               // Transmit a packet of message every one second,
 //radio.vGoStandby();                        //enter sleep mode after transmitting
 radio.vGoSleep();                            // (users can change to standby mode according to the need)
delay(1000);
}
```

- rfm65_Rx_CCITT.ino case explanation (rfm65_Rx_IBM.ino is the same basically)

```
#include <HopeDuino_RFM69.h>              //Call the two library files,
#include <HopeDuino_UART.h>       // RFM69 (the RFM65 receiving part is the same as the RF69) and UART

rf69Class radio;                          //Define variable radio for RF69
uartClass uart;                           //Define variable uart for UART
byte getstr[21];                          // Define pending data buffer

void setup()
{
  radio.Modulation        = FSK;          //Modulation mode is FSK
radio.COB                 = RFM65;        //Module type is RFM65
radio.Frequency          = 434000;        //Target frequency is 434MHz
radio.OutputPower        = 10+18;          // This configuration has no meaning
                                             //because RF65 has no transmitting function.
  radio.PreambleLength = 16;              //Preamble length is 16 bytes
  radio.FixedPktLength = false;         //Message length is variable
radio.PayloadLength     = 21;             //Payload length is 21 bytes
radio.CrcDisable        = false;        //Enable CRC
radio.CrcMode             = false;        //CRC uses CCITT polynomials,
                                             //Users need to use IBM， change here to true.
  radio.AesOn              = false;       //Do not use the AES encryption function.

  radio.SymbolTime         = 416000;        //Symbol time is 2.4Kbps.
radio.Deviation            = 35;           // Frequency deviation is 35 KHz
                    //(This configuration has no meaning because the frequency deviation is for the transmitting).
  radio.BandWidth          = 100;           //Received bandwidth is 100 KHz
  radio.SyncLength         = 3;            // Sync length is 3 bytes
radio.SyncWord[0]        = 0xAA;          //Sync value is 0xAA2DD4
radio.SyncWord[1]        = 0x2D;
  radio.SyncWord[2]        = 0xD4;
  radio.vInitialize();                  // Initialize radio
  radio.vGoRx();                             // Enter receiving mode
uart.vUartInit(9600, _8N1);             // Initialize UART, parameters are 9600 baud rate and 8N1 format.
}

void loop()
{
  if(radio.bGetMessage(getstr)!=0)      //Check radio whether to receive data function,
                                          //analyze data received.
      {
      uart.vUartPutNByte(getstr, 21);         // Output the received data to PC via UART
      uart.vUartNewLine();                    //UART newline is easy to read.
```

```
        }
    }
```

4. RF67 Library Function Description

"RF67.h"and"RF67.cpp"library files are stored in Arduino IDE files \ libraries \ HopeRFLib.

- **FreqStruct**

  Type: union type

  Function: Define frequency register for LoRa chip (module).

  Contents: Freq, long integer, 4 bytes, frequency value;

  FreqL， byte, low 8 bit from splitting Freq value is [0:7]

  FreqM， byte, mid 8 bit from splitting Freq value is [8:15]

  FreqH， byte, high 8 bit from splitting Freq value is [16:23]

  FreqX， byte, redundancy, rounding up 4 bytes, no meaning.

- **modulationType**

  Type: Enumeration type

  Function: Select modulation mode

  Contents: OOK、FSK、GFSK Contents: OOK、FSK、GFSK

  OOK——On-Off-Key is ASK modulation, a special case of ASK modulation

  FSK——Frequency-Shift-Key, relative to the ASK has a stronger anti interference effect. But the current is larger than the ASK under the same power.

  GFSK——FSK modulation with Gauss filter.

- **Modulation**

  Type: modulationType

  Function: Define modulation mode, select one of OOK, FSK and GFSK

- **Frequency**

  Type: lword type (unsigned long)

  Function: working frequency, the unit is KHz, for example: Frequency = 433920， indicates 433.92MHz.

- **SymbolTime**

  Type: lword type (unsigned long)

  Function: working rate, the unit is ns, for example: SymbolTime = 416000，indicates each symbol is 416us, that is 2.4kbps.

- **Deviation**

  Type: lword type (unsigned long)

  Function: frequency deviation for FSK and GFSK transmitting, the unit is KHz, for example: Deviation=45, indicates the frequency deviation is 45 KHz.

- **OutputPower**

Type: unsigned char

Function: output power for transmitting, the range is -13dBm～+17dBm, for example: Set is 13+10, on behalf of 10 dBm.

- **PreambleLength**

  Type: word type (unsigned int)

  Function: preamble length for transmitting, the unit is byte, the range is 0~31.

- **CrcDisable**

  Type: bool type

  Function: Select whether the data package has CRC function, set true to disable CRC function, set false to enable CRC function.

- **CrcMode**

  Type: bool type

  Function: Select CRC-16 polynomial mode, set true to use IBM polynomial ($X^{16}+X^{15}+X^2+1$); set false to use CCITT ($X^{16}+X^{12}+X^5+1$).

- **CrcSeed**

  Type: unsigned int

  Function: the initial value of CRC-16 calculation.

- **CrcInverse**

  Type: bool type

  Function: Select whether the CRC-16 needs to inverse the results after calculation completion, set true to inverse the results; set false not to inverse the results.

- **NodeDisable**

  Type: bool type

  Function: Select whether the packet has NodeID function, set true to disable NodeID function, set false to enable NodeID function.

- **NodeAddr**

  Type: unsigned char type

  Function: Specific value of NodeID

- **NodeBefore**

  Type: bool type

  Function: Select the packet format, set true, the packet format is:

  Preamble + SyncWord + NodeID + Length + Message + CRC;

  Set false, the packet format is:

  Preamble + SyncWord + Length + NodeID + Message + CRC；

  where CRC and Length are optional.

- **FixedPktLength**

Type: bool type

Function: Define whether the packet is fixed length or variable length, set true to represent the fixed length, set false to represent the variable length.

- **SyncLength**

  Type: byte

  Function: In wireless packet format, synchronous word length, the setting range is 1~8 bytes. Don't set to 0 bytes.

- **SyncWord[8]**

  Type: byte array

  Function: In setting packet format, synchronous word contents needs to be consistent with SyncLength settings.

- **PayloadLength**

  Type: byte

  Function: In fixed packet length mode, defines the length of the fixed packet.

- **vInitialize**

  Type: function

  Input: none

  Output: none

  Function: Initialize module (chip), applicable to RFM67 module, call at the start of the program. Before the call, the above related variables are set to complete. After the initialization function configuration is completed (containing call vConfig function), let the module (chip) into the Standby state, that is, non - transmitting, non - sleeping.

- **vConfig**

  Type: function

  Input: none

  Output: none

Function: Configure parameters to the module (chip), suitable for the occasion needs to re configure the parameters in the working process. The same need to complete the associated variables before the call. If the associated variables are set up, follow-up without modification, only to re configure the parameter, you can call it directly. If you need to switch frequency etc. in the working process, need to re modify the relevant parameters， and then call them again. After the call is completed, you need to use the mode switching function, so that let the chip work accurately in the specific mode. The mode switching functions are vGoFs、vGoTx、vGoStandby and vGoSleep.

- **vGoTx**

  Type: function

  Input: none

  Output: none

  Function: Configure module (chip) into the transmitting mode

- **vGoFs**

Type: function

Input: none

Output: none

Function: Configure module (chip) into the frequency synthesis mode (No transmitting, only keep the crystal oscillator starting, the PLL is locked).

- **vGoStandby**

  Type: function

  Input: none

  Output: none

  Function: Configure module (chip) into standby mode

- **vGoSleep**

  Type: function

  Input: none

  Output: none

  Function: Configure module (chip) into the sleep mode.

- **bSendMessage**

  Type: function

  Input: **msg[ ]**， unsigned char, the calling entrance (pointer) of array to be transmitted.
  length， unsigned char, the length of array to be transmitted, the unit is byte.

  Output: bool type, true indicates the transmitting is successful, false indicates the transmitting is failure, such as: push over time, etc.

  Function: transmit the data only once (one frame), return to standby mode automatically after completion of the transmission.

- **vChangeFreq**

  Type: function

  Input: freq, unsigned long type, target frequency to be switched.

  Output: none

  Function: switch to target frequency.

5. RF65 Library Function Description

The RF69 function is the same as the RF65 function,“RFM69.h”and“RFM69.cpp”library files are stored in Arduino IDE files \ libraries \ HopeRFLib.

- **FreqStruct**

  Type: union type

  Function: Define frequency register for RF69 chip.

  Contents: Freq, long integer, 4 bytes, frequency value;
  FreqL， byte, low 8 bit from splitting Freq value is [0:7];
  FreqM， byte, mid 8 bit from splitting Freq value is [8:15];
  FreqH， byte, high 8 bit from splitting Freq value is [16:23];
  FreqX， byte, redundancy, rounding up 4 bytes, no meaning.

- **modulationType**

  Type: Enumeration type

  Function: Select modulation mode

  Contents: OOK、FSK、GFSK

  OOK——On-Off-Key is ASK modulation, a special case of ASK modulation

  FSK——Frequency-Shift-Key, relative to the ASK has a stronger anti interference effect. But the current is larger than the ASK under the same power.

  GFSK——FSK modulation with Gauss filter.

- **moduleType**

  Type: enumeration type

  Function: select module mode

  Contents: RFM65, RFM65C, RFM69, RFM69C, RFM69H, RFM69HC

- **Modulation**

  Type: modulation type

  Function: Define modulation mode, select one of OOK, FSK and GFSK

- **COB**

  Type: module type

  Function: Define the module type, COB represents Chip-On-Board, select one of RFM65, RFM65C, RFM69, RFM69C, RFM69H, RFM69HC.

- **Frequency**

  Type: lword type (unsigned long)

  Function: working frequency, the unit is KHz, for example: Frequency = 433920， indicates 433.92MHz.

- **SymbolTime**

  Type: lword type (unsigned long)

  Function: working rate, the unit is ns, for example: SymbolTime = 416000，indicates each symbol is 416us, that is 2.4kbps.

- **Deviation**

  Type: lword type (unsigned long)

  Function: frequency deviation for FSK and GFSK transmitting, the unit is KHz, for example: Deviation=45, indicates the frequency deviation is 45 KHz.

- **BandWidth**

  Type: word type (unsigned int)

  Function: frequency bandwidth, the unit is KHz, for example: BandWidth = 100，indicates the receiving frequency bandwidth is 100KHz.

- **OutputPower**

Type: unsigned char

Function: output power for transmitting, the unit is 0.5dBm,

For RFM69/RFM69C, the range is 0~31, represents -18dBm～+13dBm.

For RFM69H/RFM69HC, the range is 0~31, represents -11dBm～+20dBm.

- **PreambleLength**

  Type: word type (unsigned int)

  Function: preamble length for transmitting, the unit is byte.

- **CrcDisable**

  Type: bool type

  Function: Select whether the data package has CRC function, set true to disable CRC function, set false to enable CRC function.

- **FixedPktLength**

  Type: bool type

  Function: Define whether the packet is fixed length or variable length, set true to represent the fixed length, set false to represent the variable length.

- **AesOn**

  Type: bool type

  Function: Define whether the packet requires AES128 encryption, set true to represent the packet is processed by AES128, set false to represent the packet is not processed by AES128.

- **AfcOn**

  Type: bool type

  Function: Define whether to enable AFC function for receiving configuration.

- **SyncLength**

  Type: byte

  Function: In wireless packet format, synchronous word length, the setting range is 1~8 bytes. Don't set to 0 bytes.

- **SyncWord[8]**

  Type: byte array

  Function: In setting packet format, synchronous word contents needs to be consistent with SyncLength settings.

- **PayloadLength**

  Type: byte

  Function: In fixed packet length mode, defines the length of the fixed packet.

- **AesKey[16]**

  Type: byte array

  Function: After opening the AES128 encryption mode, the key content of the AES128 is defined, and the key length is 128 bits, which is 16 bytes.

- **vInitialize**
  Type: function
  Input: none
  Output: none
  Function: Initialize module (chip), applicable to RFM69/C and RFM69H/HC modules, call them at the start of the program. Before the call, the above related variables are set to complete. After the initialization function configuration is completed (containing call vConfig function), let the module (chip) into the Standby state, that is, non - transmitting, non-receiving, non - sleeping.

- **vConfig**
  Type: function
  Input: none
  Output: none
  Function: Configure parameters to the module (chip), suitable for the occasion needs to re configure the parameters in the working process. The same need to complete the associated variables before the call. If the associated variables are set up, follow-up without modification, only to re configure the parameter, you can call it directly. If you need to switch frequency etc. in the working process, need to re modify the relevant parameters， and then call them again. After the call is completed, you need to use the mode switching function, so that let the chip work accurately in the specific mode. The mode switching functions are vGoRx、vGoStandby and vGoSleep.

- **vGoRx**
  Type: function
  Input: none
  Output: none
  Function: Configure module (chip) into the receiving mode

- **vGoStandby**
  Type: function
  Input: none
  Output: none
  Function: Configure module (chip) into standby mode

- **vGoSleep**
  Type: function
  Input: none
  Output: none
  Function: Configure module (chip) into the sleep mode.

- **bSendMessage**
  Type: function
  Input: **msg[ ]**， unsigned char, the calling entrance (pointer) of array to be transmitted.
  length， unsigned char, the length of array to be transmitted, the unit is byte.

Output: bool type, true indicates the transmitting is successful, false indicates the transmitting is failure, such as: push over time, etc.

Function: transmit the data only once (one frame), return to standby mode automatically after completion of the transmission.

- **bGetMessage**

  Type: function

  Input: **msg[ ]**，unsigned char, the calling entrance (pointer) of array to be received.

  Output: Returns the length of the received data, 0 indicates that the data is not received;

  Function: check whether to receive data. The object is the IO state of the chip output. If you do not receive the data, return 0; if you receive the data, return the received data length. After the completion of receiving, the module (chip) is still in the receiving state.

- **vRF69SetAesKey**

  Type: function

  Input: none

  Function: Set the AesKey[16] to the relevant registers in the module (chip).

- **vTrigAfc**

  Type: function

  Input: none

  Output: none

  Function: Adjust a AFC manually, suitable for AFC_ON state.

- **vDirectRx**

  Type: function

  Input: none

  Output: none

  Function: Let the module (chip) in DirectMode in the current configuration; output the received data stream through the DIO2 pin, generally used in sensitivity tests.

- **vChangeFreq**

  Type: function

  Input: freq, long integer type, target frequency to be switched.

  Output: none

  Function: By this function, you can quickly switch the operating frequency (Other parameters unchanged, such as speed, bandwidth, etc., only adjust the frequency). After the switch is completed, the system continues to be in the receiving mode, which is suitable for the application of frequency hopping mechanism.

- **bReadRssi**

  Type: function

  Input: none

  Output: Return the RSSI value detected currently.

  Function: Manually do a RSSI test, and return the value.

6. Pin Assignment Table:

HopeDuino	MCU	RF67	RF65
13	PB5	SCK	SCK
12	PB4	MISO	MISO
11	PB3	MOSI	MOSI
10	PB2	nCS	nCS
9	PB1	DCLK	POR
8	PB0	POR	DIO0
7	PD7	EOL（jumper）	DIO1（jumper）
6	PD6	DATA（jumper）	DIO2（jumper）
5	PD5	PLL_LOCK（jumper）	DIO3（jumper）
4	PD4		DIO4（jumper）

7. Version Records:

Version	Revised Contents	Date
1.0	Initial version	2016-04-5
1.1	Add example explanations	2016-04-07