CRC 通讯范例

本章将指引用户通过 HopeDuino 进行 RFM67 和 RFM65 两个产品的收发通讯范例，展示 RF 芯片常用的数据报文 CRC 方式。

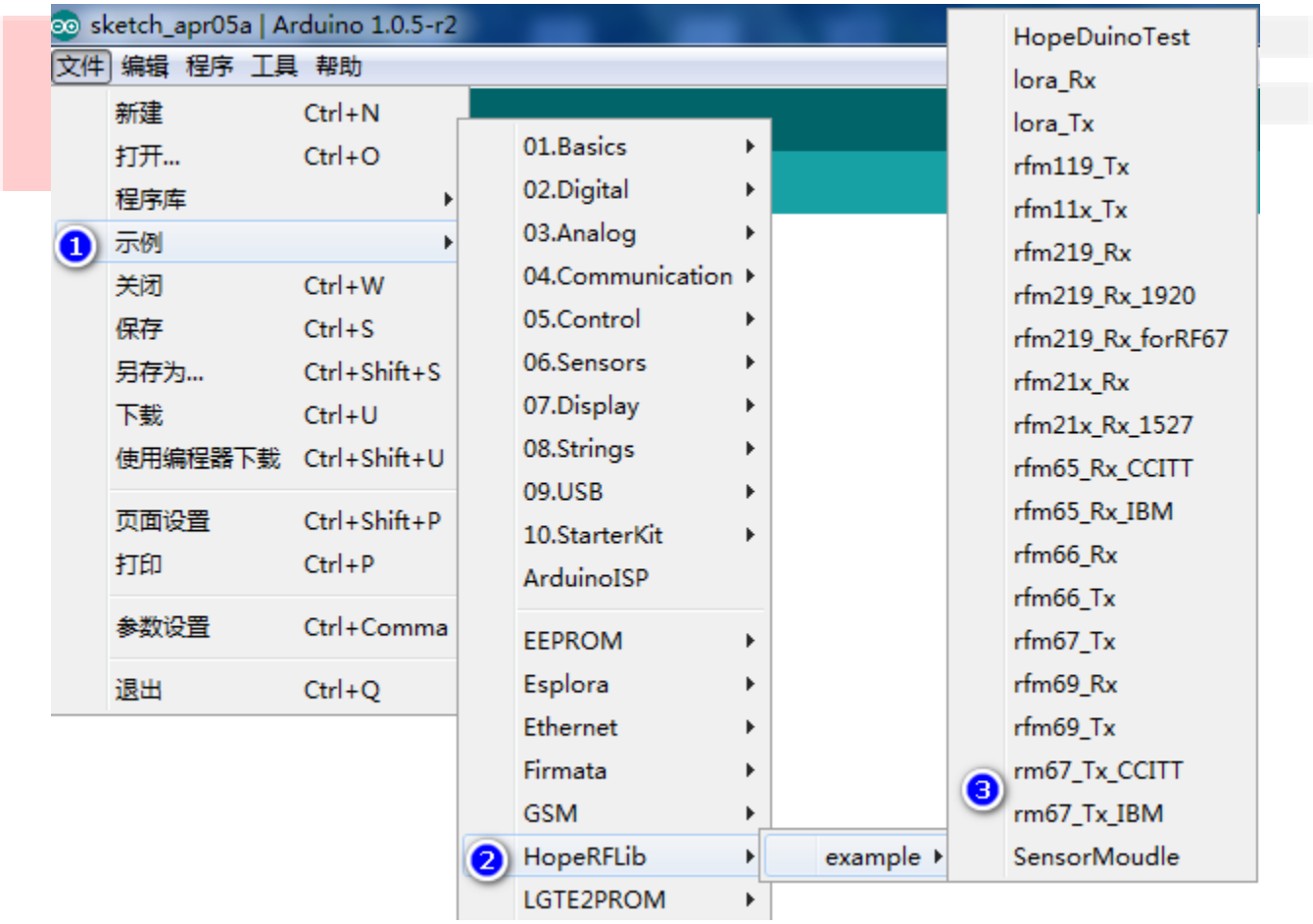
一、需要准备的工具和软件

- Arduino 1.0.5 版本 IDE
- HopeDuino 板（2 个）
  （如果此前没有使用过 HopeDuino 板，请参见《AN0002-HopDuino 平台搭建指引》）
- USB 连接线（A 口转 B 口）
- RFM67 模块（或基于 RF67 设计的模块）
- RFM65 模块（或基于 RF65 设计的模块）

二、动手实验

- 把 RFM67 和 RFM65 两个模块（带转换板）分别插入到各自对应的 HopeDuino 板上；
- 把两个 HopeDuino 板通过 USB 线连接到 PC；
- 打开 Arduino IDE 界面，点击【文件】→【示例】→【HopeRFLib】→【rfm67_Tx_CCITT】或【rfm67_Tx_IBM】，如下图操作；

⚠注意：如果没有在【示例】中找到【HopeRFLib】，是因为没有安装好 HopeRF 提供的 HSP，具体操作请参见《AN0002-HopDduino 平台搭建指引》



- 打开 Arduino IDE 界面，点击【文件】→【示例】→【HopeRFLib】→【rfm65_Rx_CCITT】或【rfm65_Rx_IBM】，

操作方式同上。

注意：收和发两个文件需要对应，即选择 rfm67_Tx_CCITT 的，对应选择 rfm65_Rx_CCITT；反之一样。

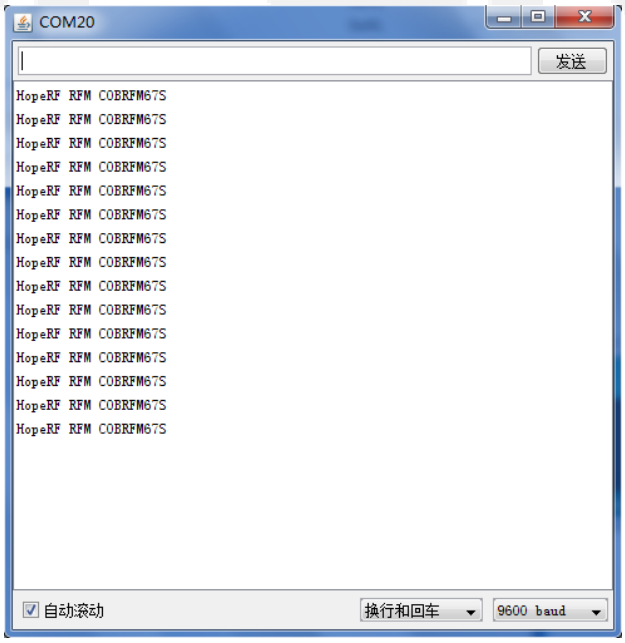
- 此时已经打开一个 Tx 的例程和一个 Rx 的例程，请根据对应的 COM 口进行编译下载程序；

注意：

1. 不清楚如何编译下载代码，请参见《AN0002-HopDuino 平台搭建指引》
2. HopeDuino 可以支持多个连接同一台 PC，需要使手动修改，所以每次下载程序需要特别留意当前界面下载是那个 COM 口的 HopeDuino。COM 口在 Arduino 界面的右下角，如下图所示：

- 两个例程下载完成后，Tx 端的 HopeDuino 就会周期性通过 RFM67 模块发射一包数据；Rx 端的 HopeDuino 就会相应周期性通过 RFM65 模块接收到一包数据，并同时把这包数据通过 UART（USB）上传到 PC。此时，可以把 Arduino IDE 的 COM 口设置为连接到 Rx 的 HopeDuino 板，并打开“串口监视器”，如下图所示：

- 点击“串口监视器”后，就会弹出串口助手界面，如下图所示，窗口就会显示出收到数据报文。



注意：

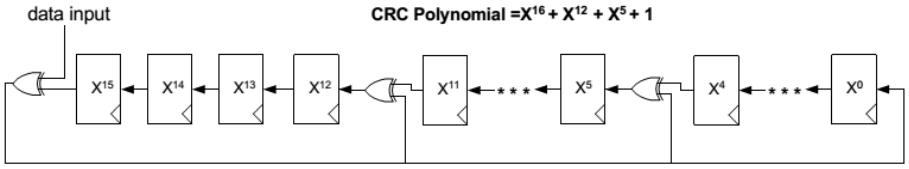
1. 接收例程启用 UART 库函数，关于 UART 库函数描述，请直接参见“HopeDuino_UART”库文件，它同样存储在 HopeRFLib 之中。

三、RF 芯片常见的 CRC16 简述

- 常见 RF 芯片数据报文内嵌一般采用 16 位 CRC 校验，主要有两种 CRC-16 方式——IBM 和 CCITT，以 RF65/69 为例，如下图表所示：

CRC 方式	多项式	计算初值	结果取反
CCITT	$X^{16}+X^{12}+X^5+1$	0x1D0F	是
IBM	$X^{16}+X^{15}+X^2+1$	0xFFFF	否

- 以 CCITT 多项式 $X^{16}+X^{12}+X^5+1$，移位框图如下：



- 需要注意“计算初值”和“结果取反”，因为“计算初值”不同，也会导致 CRC 结果不同；而“结果取反”也会直接影响 CRC 结果。“计算初值”在本例程序变量名为 CrcSeed，“结果取反”在本例程序变量名为 CrcInverse。

- 如前面提及，由于 CRC 是接收端芯片用于确认数据报文是否有误，所以采用那种 CRC 方式、计算初值、结果是否取反，都由接收端芯片决定。因此本例程的接收芯片 RF65 来决定，而 RF67 作为发射端而言，芯片内部不带有 CRC 功能，只能通过程序实现，这些程序存放在“～\HopeRFLib\ HopeDuino_RFM67.cpp”和“～\HopeRFLib\ HopeDuino_RFM67.h”库文件中，有需要的用户请自行阅读并移植。

四、范例解释

- rfm67_Tx_CCITT 范例解释（rfm67_Tx_IBM 基本相同）

```
#include <HopeDuino_RFM67.h>              //调用对应库文件
rf67Class radio;                          //定义一个 rf67 的类变量 radio
                                          //待发报文数据
byte str[21] = {'H','o','p','e','R','F',' ','R','F','M',' ','C','O','B','R','F','M','6','7','S'};

void setup()
{
 radio.Modulation      = FSK;             // FSK 调制
 radio.Frequency       = 434000;          //频率为 434MHz
 radio.OutputPower     = 10+13;           //10dBm 输出功率
 radio.PreambleLength  = 16;              //16Byte Preamble
 radio.FixedPktLength  = false;           //可变长度报文结构
 radio.PayloadLength   = 21;              //报文内容 21 字节

 radio.CrcDisable      = false;           //使能 CRC 校验
 radio.CrcInverse      = true;            // CRC 计算结果需要取反，针对 RF65 接收需求
 radio.CrcMode         = false;           // CRC 采用 CCITT 多项式，需要采用 IBM 的用户，此处改为 true，
                                          //但需要注意 CRC 初始值和是否需要取反结果
 radio.CrcSeed         = 0x1D0F;          // CRC 初始值为 0x1D0F（十六进制），与 RF65 对应
```

```
  radio.NodeDisable    = true;          //不采用节点模式
 //radio.NodeBefore    = false;         //
 //radio.NodeAddr      = 0x01;

 radio.SymbolTime     = 416000;        //2.4Kbps 空中速率
 radio.Devation       = 35;            //35KHz 发射频偏
 radio.SyncLength     = 3;             //同步字为 3 字节长度
 radio.SyncWord[0]    = 0xAA;          //同步字为 0xAA2DD4
 radio.SyncWord[1]    = 0x2D;
 radio.SyncWord[2]    = 0xD4;
 radio.vInitialize();                  //执行 radio 初始化功能
 radio.vGoStandby();                   //进入待机模式
}

void loop()
{
 radio.bSendMessage(str, 21);          //每秒钟发射一报文，发完之后进入睡眠（用户可以根据需要改
 //radio.vGoStandby();                 //为进入待机模式）
 radio.vGoSleep();
 delay(1000);
}
```

➢ rfm65_Rx_CCITT.ino 范例解释（rfm65_Rx_IBM.ino 范例基本相同）

```
#include <HopeDuino_RFM69.h>          //调用两个库文件，RFM69（RFM65 接收部分与 RF69 同）和 UART
#include <HopeDuino_UART.h>

rf69Class radio;                      //定义 RF69 的类变量 radio
uartClass uart;                       //定义 UART 的类变量 uart
byte getstr[21];                      //待接收数据缓冲数组

void setup()
{
 radio.Modulation     = FSK;           //FSK 调制
 radio.COB            = RFM65;         //模块类型为 RFM65
 radio.Frequency      = 434000;        //频率为 434MHz
 radio.OutputPower    = 10+18;         //此项配置没意义，RF65 不具备发射功能
 radio.PreambleLength = 16;            //16Byte Preamble
 radio.FixedPktLength = false;         //可变长度报文结构
 radio.PayloadLength  = 21;            //报文内容 21 字节
 radio.CrcDisable     = false;         //CRC 使能
 radio.CrcMode        = false;         //CRC 采用 CCITT 模式，用户需要 IBM，可以改为 true
 radio.AesOn          = false;         //不采用 AES 加密功能

 radio.SymbolTime     = 416000;        //2.4Kbps 空中速率
```

```
    radio.Devation          = 35;           //35KHz 频偏（此项配置也没有意义，频偏是针对发射的参数）
    radio.BandWidth         = 100;          //100K 接收带宽
    radio.SyncLength        = 3;            //同步字长度为 3 字节
    radio.SyncWord[0]       = 0xAA;         //同步字值为 0xAA2DD4
    radio.SyncWord[1]       = 0x2D;
    radio.SyncWord[2]       = 0xD4;
    radio.vInitialize();                    //初始化参数
    radio.vGoRx();                          //进入接收模式
    uart.vUartInit(9600, _8N1);             //执行 UART 配置初始化，初始化为 9600 波特率、8N1 格
}

void loop()
{
 if(radio.bGetMessage(getstr)!=0)           //执行 radio 查询是否收到数据函数，收到就解析数据
    {
    uart.vUartPutNByte(getstr, 21);         //把收到的数据通过 UART 打印到 PC 上
    uart.vUartNewLine();                    //UART 换行，便于阅读
    }
}
```

五、RF67 库函数说明

“RF67.h”和“RF67.cpp”库文件存放路径在 Arduino IDE 文件夹\ libraries \ HopeRFLib；

- **FreqStruct**

  类型： Union 类型

  功能： 针对 LoRa 芯片（模块）频率寄存器定义

  内容： Freq，长整型，4 字节，频率值；
  FreqL，字节，针对 Freq 拆分的频率值低 8 位[0:7]；
  FreqM，字节，针对 Freq 拆分的频率值中 8 位[8:15]；
  FreqH，字节，针对 Freq 拆分的频率值高 8 位[16:23]；
  FreqX，字节，冗余，凑整 4 字节，没有意义。

- **modulationType**

  类型： 枚举类型

  功能： 选择调制解调制式

  ⚠ **内容：** OOK、FSK、GFSK
  OOK——On-Off-Key 属于 ASK 调制，是 ASK 调制一种特例；
  FSK——Frequency-Shift-Key，跳频键控 ，相对 ASK 有更强抗干扰效果；但同等功率情况下，比 ASK 电流大；
  GFSK——带高斯滤波的 FSK 调制；

- **Modulation**

  类型： modulationType 类型

  功能： 定义调制解调制式，在 OOK、FSK、GFSK 三者中选择其一。

- **Frequency**
  类型：lword 型（unsigned long）
  功能：目标工作频率，单位 KHz，例如：Frequency = 433920，代表 433.92MHz。

- **SymbolTime**
  类型：lword 型（unsigned long）
  功能：目标工作速率，单位 ns，例如：SymbolTime = 416000，代表每个符号 416us，即 2.4kbps。

- **Devation**
  类型：lword 型（unsigned long）
  功能：目标工作频偏，针对 FSK 和 GFSK 发射需要定义，单位 KHz，例如：Devation = 45，代表工作频率 45KHz。

- **OutputPower**
  类型：unsigned char
  功能：目标输出功率，针对发射需要定义，范围-13dBm～+17dBm，例如：设置为 13+10，表示 10dBm。

- **PreambleLength**
  类型：word 型（unsigned int）
  功能：数据包的 Preamble 长度设置，针对发射需要配置，单位字节，范围 0～31。

- **CrcDisable**
  类型：bool 类型
  功能：选择数据包功能中是否带 CRC，设置 true 代表禁止 CRC 功能；设置 false 代表开启 CRC 功能。

- **CrcMode**
  类型：bool 类型
  功能：选择数据包功能中 CRC-16 的多项式模式，设置 true 代表采用 IBM 多项式（$X^{16}+X^{15}+X^2+1$）；设置 false 代表采用 CCITT（$X^{16}+X^{12}+X^5+1$）。

- **CrcSeed**
  类型：unsigned int 类型
  功能：CRC-16 计算的起始值。

- **CrcInverse**
  类型：bool 类型
  功能：CRC-16 运算完毕后是否需要取反结果，设置 true 代表需要把结果取反；设置 false 代表不需要把结果取反。

- **NodeDisable**
  类型：bool 类型
  功能：选择数据包功能中是否带 NodeID 功能，设置 true 代表禁止 NodeID 功能；设置 false 代表开启 NodeID 功能。

- **NodeAddr**
  类型：unsigned char 类型
  功能：NodeID 的具体值。

- **NodeBefore**
  类型：bool 类型
  功能：选择数据包格式，设置 true，数据包格式为：
  Preamble + SyncWord + NodeID + Length + Message + CRC;
  设置 false，数据包格式为：
  Preamble + SyncWord + Length + NodeID + Message + CRC；
  其中 CRC 和 Length 可选；

- **FixedPktLength**
  类型：bool 类型
  功能：定义数据包是固定包长度，还是变长数据包，设置 true 代表固定包长度；设置 false 代表变长数据包格式。

- **SyncLength**
  类型：字节
  功能：无线数据包格式中，同步字长度，设置范围是 1～8 字节；不能设置为 0 字节。

- **SyncWord[8]**
  类型：字节数组
  功能：设置数据包格式中，同步字的内容，需要与 SyncLength 设置符合（长度）。

- **PayloadLength**
  类型：字节
  功能：在固定包长模式中，定义固定包长度。

- **vInitialize**
  类型：函数
  入参：无
  出参：无
  功能：初始化模块（芯片），适用于 RFM67 模块，在程序最开始调用；调用前，需要把上述关联的变量设置完整。初始化函数配置完毕后（内含调用 vConfig 函数），让模块（芯片）出于 Standby 状态，即非发、非睡眠。

- **vConfig**
  类型：函数
  入参：无
  出参：无
  功能：配置参数到模块（芯片）中，适用于程序工作过程中需要重新配置参数的场合。调用前同样需要把关联变量设置完整。如果此前关联变量设置完整，后续无需改动，仅想重新配置一次参数，可以直接调用；如果需要在工作过程中，切换频点工作等，需要重新修改相关参数，然后在调用。调用完毕后，需要用到工作模式切

换函数，以便让芯片准确到具体工作模式中，模式切换函数有：vGoFs、vGoTx、vGoStandby、vGoSleep。

- **vGoTx**
  类型：函数
  入参：无
  出参：无
  功能：配置模块（芯片）进入发射模式。

- **vGoFs**
  类型：函数
  入参：无
  出参：无
  功能：配置模块（芯片）进入频率合成模式（非发射，仅保持晶振起振，锁相环锁定）。

- **vGoStandby**
  类型：函数
  入参：无
  出参：无
  功能：配置模块（芯片）进入待机模式。

- **vGoSleep**
  类型：函数
  入参：无
  出参：无
  功能：配置模块（芯片）进入睡眠模式。

- **bSendMessage**
  类型：函数
  入参：msg[ ]，unsigned char 类型指针，待发射数据数组调用入口（指针）；
  length，unsigned char 类型，待发射数据长度，单位是 byte；
  出参：返回 bool 类型，true 表示发射成功；false 表示发射失败，诸如：推送超时等情况；
  功能：把待发射数据发送出去，仅发送一次（一帧）；发送完毕后自动返回待机状态（Standby 模式）。

- **vChangeFreq**
  类型：函数
  入参：freq，unsigned long 类型指针，待切换目标频率值；
  出参：无
  功能：切换到目标频率。

六、RF65 库函数说明

RF65 函数与 RF69 同，库文件“RFM69.h”和“RFM69.cpp”存放路径在 Arduino IDE 文件夹\ libraries \ HopeRFLib；

- **FreqStruct**
  类型：Union 类型

功能：针对 RF69 频率寄存器定义
内容：Freq，长整型，4 字节，频率值；
FreqL，字节，针对 Freq 拆分的频率值低 8 位[0:7]；
FreqM，字节，针对 Freq 拆分的频率值中 8 位[8:15]；
FreqH，字节，针对 Freq 拆分的频率值高 8 位[16:23]；
FreqX，字节，冗余，凑整 4 字节，没有意义。

- **modulationType**

类型：枚举类型
功能：选择调制解调制式
⚠**内容：**OOK、FSK、GFSK
OOK——On-Off-Key 属于 ASK 调制，是 ASK 调制一种特例；
FSK——Frequency-Shift-Key，跳频键控 ，相对 ASK 有更强抗干扰效果；但同等功率情况下，比 ASK 电流大；
GFSK——带高斯滤波的 FSK 调制。

- **moduleType**

类型：枚举类型
功能：选择模块型号
内容：RFM65, RFM65C, RFM69, RFM69C, RFM69H, RFM69HC

- **Modulation**

类型：modulationType 类型
功能：定义调制解调制式，在 OOK、FSK、GFSK 三者中选择其一。

- **COB**

类型：moduleType 类型
功能：定义模块型号，COB 表示 Chip-On-Borad，可以在 RFM65, RFM65C, RFM69, RFM69C, RFM69H, RFM69HC 中择其一。

- **Frequency**

类型：lword 型（unsigned long）
功能：目标工作频率，单位 KHz，例如：Frequency = 433920，代表 433.92MHz。

- **SymbolTime**

类型：lword 型（unsigned long）
功能：目标工作速率，单位 ns，例如：SymbolTime = 416000，代表每个符号 416us，即 2.4kbps。

- **Devation**

类型：lword 型（unsigned long）
功能：目标工作频偏，针对 FSK 和 GFSK 发射需要定义，单位 KHz，例如：Devation = 45，代表工作频率 45KHz。

- **BandWidth**
  类型：word 型（unsigned int）
  功能：目标工作接收带宽，针对接收需要定义，单位 KHz，例如：BandWidth = 100，代表接收带宽 100KHz。

- **OutputPower**
  类型：unsigned char
  功能：目标输出功率，针对发射需要定义，单位 0.5dBm，其中
  针对 RFM69/RFM69C，设置范围 0-31，代表 -18dBm～+13dBm，
  针对 RFM69H/RFM69HC，设置范围 0-31，代表-11dBm～+20dBm。

- **PreambleLength**
  类型：word 型（unsigned int）
  功能：数据包的 Preamble 长度设置，针对发射需要配置，单位字节。

- **CrcDisable**
  类型：bool 类型
  功能：选择数据包功能中是否带 CRC，设置 true 代表禁止 CRC 功能；设置 false 代表开启 CRC 功能。

- **FixedPktLength**
  类型：bool 类型
  功能：定义数据包是固定包长度，还是变长数据包，设置 true 代表固定包长度；设置 false 代表变长数据包格式。

- **AesOn**
  类型：bool 类型
  功能：定义数据包是否需要 AES128 加密，设置 true 代表数据收发均经过 AES128 处理；设置 false 代表取消 AES128 处理。

- **AfcOn**
  类型：bool 类型
  功能：定义是否启用 AFC 功能，针对接收需要配置。

- **SyncLength**
  类型：字节
  功能：无线数据包格式中，同步字长度，设置范围是 1～8 字节；不能设置为 0 字节。

- **SyncWord[8]**
  类型：字节数组
  功能：设置数据包格式中，同步字的内容，需要与 SyncLength 设置符合（长度）。

- **PayloadLength**
  类型：字节
  功能：在固定包长模式中，定义固定包长度。

- **AesKey[16]**
  类型：字节数组
  功能：开启 AES128 加密模式后，定义 AES128 的密钥内容，密钥长度 128 位，即 16 字节。

- **vInitialize**
  类型：函数
  入参：无
  出参：无
  功能：初始化模块（芯片），适用于 RFM69/C、RFM69H/HC 模块，在程序最开始调用；调用前，需要把上述关联的变量设置完整。初始化函数配置完毕后（内含调用 vConfig 函数），让模块（芯片）出于 Standby 状态，即非发、非收、非睡眠。

- **vConfig**
  类型：函数
  入参：无
  出参：无
  功能：配置参数到模块（芯片）中，适用于程序工作过程中需要重新配置参数的场合。调用前同样需要把关联变量设置完整。如果此前关联变量设置完整，后续无需改动，仅想重新配置一次参数，可以直接调用；如果需要在工作过程中，切换频点工作等，需要重新修改相关参数，然后在调用。调用完毕后，需要用到工作模式切换函数，以便让芯片准确到具体工作模式中，模式切换函数有：vGoRx、vGoStandby、vGoSleep 等。

- **vGoRx**
  类型：函数
  入参：无
  出参：无
  功能：配置模块（芯片）进入接收模式。

- **vGoStandby**
  类型：函数
  入参：无
  出参：无
  功能：配置模块（芯片）进入待机模式。

- **vGoSleep**
  类型：函数
  入参：无
  出参：无
  功能：配置模块（芯片）进入睡眠模式。

- **bSendMessage**
  类型：函数
  入参：**msg[ ]**，unsigned char 类型指针，待发射数据数组调用入口（指针）；
  length，unsigned char 类型，待发射数据长度，单位是 byte；
  出参：返回 bool 类型，true 表示发射成功；false 表示发射失败，诸如：推送超时等情况；

功能：把待发射数据发送出去，仅发送一次（一帧）；发送完毕后自动返回待机状态（Standby 模式）。

- **bGetMessage**

  类型：函数

  入参：msg[ ]，unsigned char 类型指针，待接收数据数组调用入口（指针）；

  出参：返回接收数据的长度值，如果返回 0 表示没有收到数据；

  功能：查询是否接收到数据，查询对象是芯片输出的 IO 状态，如果没有收到数据，返回 0；如果收到数据，返回收到的数据长度，收取完毕后，模块（芯片）仍旧处于接收状态。

- **vRF69SetAesKey**

  类型：函数

  入参：无

  出参：无

  功能：把 AesKey[16]设置到模块（芯片）的相关寄存器中。

- **vTrigAfc**

  类型：函数

  入参：无

  出参：无

  功能：手动做一次 AFC 调整，适用于 AFC_ON 状态下。

- **vDirectRx**

  类型：函数

  入参：无

  出参：无

  功能：在当前参数配置下，让模块（芯片）处于 DirectMode，通过 DIO2 管脚输出接收到的数据流；一般用于灵敏度测试中。

- **vChangeFreq**

  类型：函数

  入参：freq，长整型，待切换的目标频率值；

  出参：无

  功能：通过这个函数，可以快速切换工作频率（其它参数不变，如速率、带宽等，仅调整频率），切换完成后，继续处于接收模式，适用于跳频机制应用。

- **bReadRssi**

  类型：函数

  入参：无

  出参：返回当前检测到 RSSI 的值

  功能：手动做一次 RSSI 检测，并把值返回出来。

七、引脚分配表

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（跳线）	DIO1（跳线）
6	PD6	DATA（跳线）	DIO2（跳线）
5	PD5	PLL_LOCK（跳线）	DIO3（跳线）
4	PD4		DIO4（跳线）

八、版本记录

版本	修改内容	日期
1.0	初始版本	2016-04-05
1.1	增加范例解释	2016-04-07