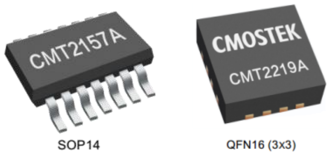
CMT2157A & CMT2219A Communication Example (1920coding)

This chapter will guide the user to carry out the communication experiment on a pair of single transmitting and single receiving chips CMT2157A and CMT2219A. CMT2157A is the FSK/OOK modulated single transmitting chip with coding function belonging to HopeRF's CMOSTEK wireless product line below. CMT2219A is the FSK/OOK modulated single receiving chip, all support Sub-1G applications.



1. Tools and software needed to be prepared

- Arduino IDE version 1.0.5
- HopeDuino board

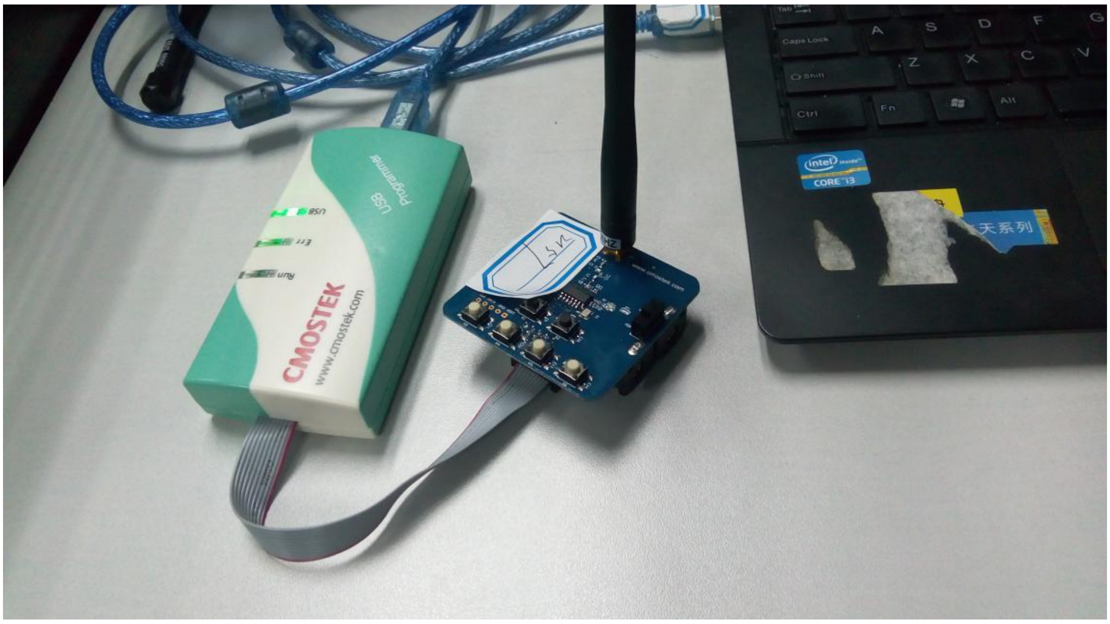
  (If you have not used the HopeDuino board, please refer to the 《AN0002-HopeDuino Platform Construction Guideline》)
- USB cable(Type A to Type B)
- CMT2157A-EM board（Or product based on CMT2157A chip design）
- CMOSTEK USB Programmer
- CMOSTEK RFPDK V1.38（Pay attention to using the latest version. The latest version is V1.38 in the paper）
- Module RFM219S (Based on chip CMT2219A) and the matching conversion board

RFM219S

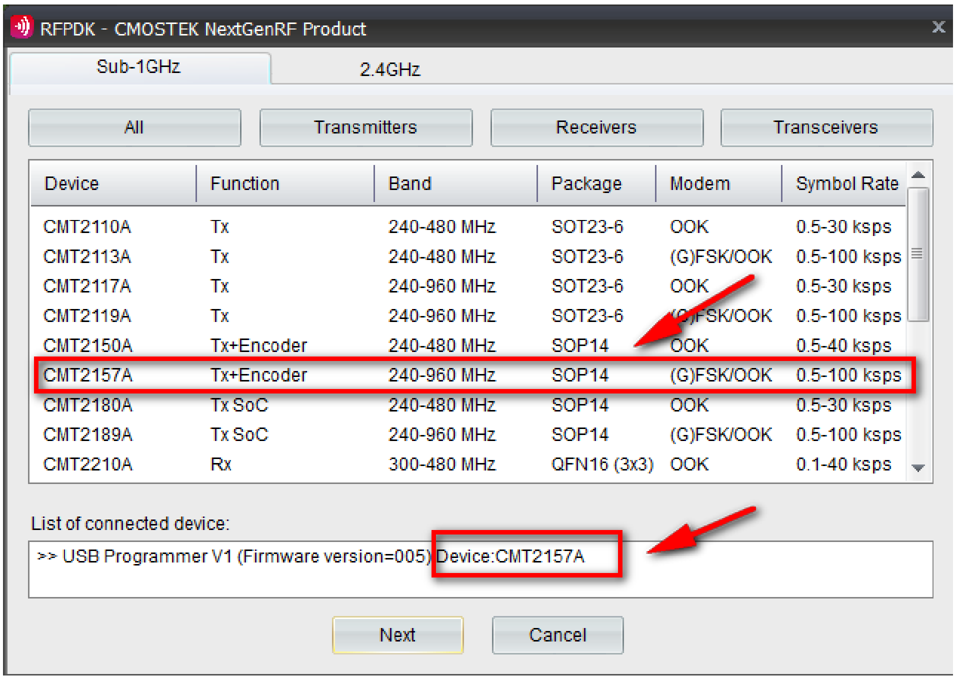
CMT2157A - EM

2. CMT2157A parameter configuration and burning

- CMT2157A configuration details refer to CMOSTEK《AN112 CMT2150A Configuration Guideline》
- This paper focuses on the experiment for the purpose, configuring parameters and demonstrating effect simply.
  - Connect CMT2157A-EM to PC with USB Programmer



- Open CMOSTEK RFPDK interface, select“CMT2157A”as below, click and enter:

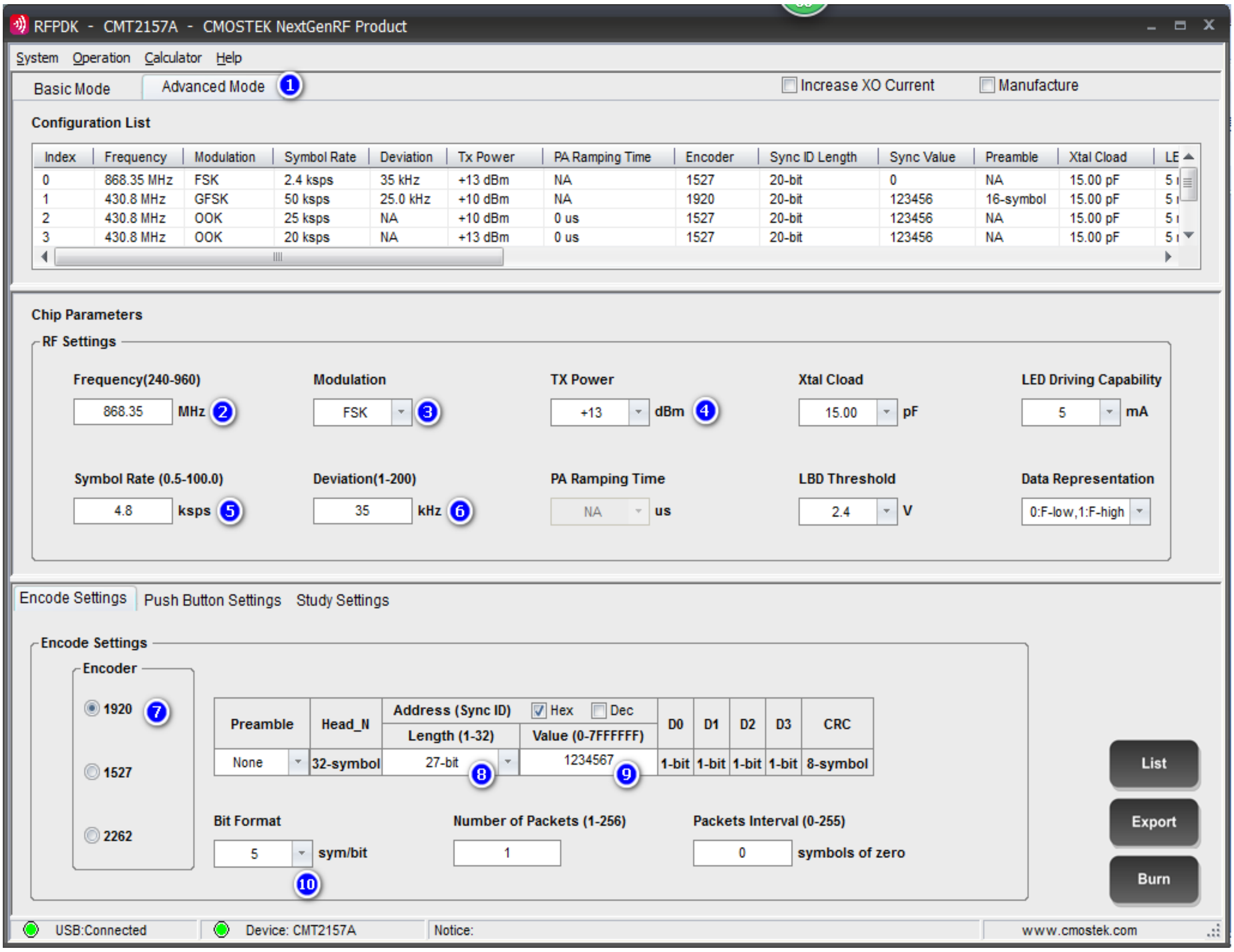


Device	Function	Band	Package	Modem	Symbol Rate
CMT2110A	Tx	240-480 MHz	SOT23-6	OOK	0.5-30 ksps
CMT2113A	Tx	240-480 MHz	SOT23-6	(G)FSK/OOK	0.5-100 ksps
CMT2117A	Tx	240-960 MHz	SOT23-6	OOK	0.5-30 ksps
CMT2119A	Tx	240-960 MHz	SOT23-6	(G)FSK/OOK	0.5-100 ksps
CMT2150A	Tx+Encoder	240-480 MHz	SOP14	OOK	0.5-40 ksps
CMT2157A	Tx+Encoder	240-960 MHz	SOP14	(G)FSK/OOK	0.5-100 ksps
CMT2180A	Tx SoC	240-480 MHz	SOP14	OOK	0.5-30 ksps
CMT2189A	Tx SoC	240-960 MHz	SOP14	(G)FSK/OOK	0.5-100 ksps
CMT2210A	Rx	300-480 MHz	QFN16 (3x3)	OOK	0.1-40 ksps

List of connected device:

>> USB Programmer V1 (Firmware version=005) Device:CMT2157A

- Configuring parameters and burning



RFPDK - CMT2157A - CMOSTEK NextGenRF Product
System Operation Calculator Help
Basic Mode
Advanced Mode
1
2
3
4
5
6
7
8
9
10
List
Export
Burn
www.cmostek.com

□ Increase XO Current □ Manufacture

Configuration List

Index	Frequency	Modulation	Symbol Rate	Deviation	Tx Power	PA Ramping Time	Encoder	Sync ID Length	Sync Value	Preamble	Xtal Cload	LE
0	868.35 MHz	FSK	2.4 ksps	35 kHz	+13 dBm	NA	1527	20-bit	0	NA	15.00 pF	5
1	430.8 MHz	GFSK	50 ksps	25.0 kHz	+10 dBm	NA	1920	20-bit	123456	16-symbol	15.00 pF	5
2	430.8 MHz	OOK	25 ksps	NA	+10 dBm	0 us	1527	20-bit	123456	NA	15.00 pF	5
3	430.8 MHz	OOK	20 ksps	NA	+13 dBm	0 us	1527	20-bit	123456	NA	15.00 pF	5

Chip Parameters

RF Settings

Frequency(240-960): 868.35 MHz
Modulation: FSK
TX Power: +13 dBm
Xtal Cload: 15.00 pF
LED Driving Capability: 5 mA
Symbol Rate (0.5-100.0): 4.8 ksps
Deviation(1-200): 35 kHz
PA Ramping Time: NA us
LBD Threshold: 2.4 V
Data Representation: 0:F-low,1:F-high

Encode Settings | Push Button Settings | Study Settings

Encode Settings

Encoder: ● 1920 ○ 1527 ○ 2262

Preamble	Head_N	Address (Sync ID) ☑ Hex □ Dec		D0	D1	D2	D3	CRC
		Length (1-32)	Value (0-7FFFFFF)					
None	32-symbol	27-bit	1234567	1-bit	1-bit	1-bit	1-bit	8-symbol

Bit Format: 5 sym/bit
Number of Packets (1-256): 1
Packets Interval (0-255): 0 symbols of zero

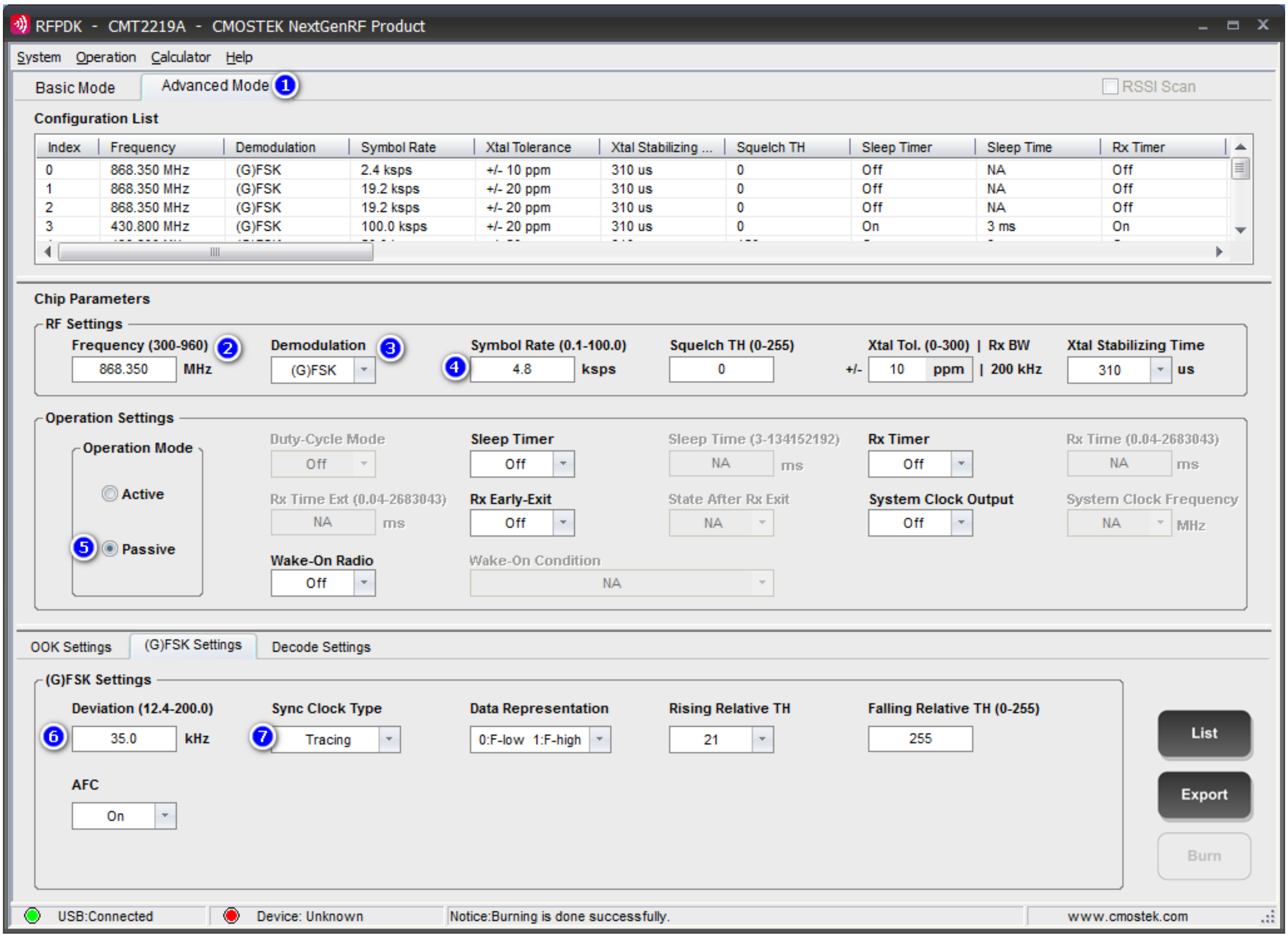
USB:Connected Device: CMT2157A Notice:

1. Select AdvancedMode (This mode has a lot of special features on CMT2157A, please refer to AN112 instructions)
2. Configure work frequency. Here we select 868.35MHz. Note RFM219S also need to select the 868 band matching test.
3. Configure modulation mode, this experiment uses FSK modulation.
4. Configure CMT2157A transmit power, output power is +13dBm (20mW).
5. Configure wireless rate, here we select 4.8Ksps. Note RFM219S rate is also corresponding to it.
6. Configure transmit frequency deviation. The parameter must be configured in the FSK mode. This time we choose 35KHz as the frequency deviation.
7. Select 1920 coding format
8. Configure ID Bits. This is unique to the 1920 encoding format. This time we choose a more special length——27 Bits.
9. Configure the ID value in Hex format. For simple convenience, the ID value is defined as 0x1234567
10. In BitFormat, we also choose a special 5Symbol that represents a logical bit encoding format.
11. Other parameters are default and no adjustments. (If you select “Normal”key mode， K1~K4 is effective in the CMT2150A-EM .)
12. Click “Burn” button, burn parameters.

- Pull out CMT2150A-EM from USB Programmer and toggle the switch to “VBAT” after burning (Prior to this, please install 2 AA batteries). At this point, press any button of K1～K4, LED on the CMT2150A-EM will be lit, it indicates the key transmitting is effective.

3. CMT2219A parameter configuration and burning

- ➢ CMT2219A configuration details refer to CMOSTEK《AN138 CMT2219A Configuration Guideline》
- ➢ Default parameters of module RFM219S are as below:


RFPDK - CMT2219A - CMOSTEK NextGenRF Product
System Operation Calculator Help
Basic Mode
Advanced Mode
1
☐ RSSI Scan
2
3
4
5
6
7
List
Export
Burn

Configuration List

Index	Frequency	Demodulation	Symbol Rate	Xtal Tolerance	Xtal Stabilizing ...	Squelch TH	Sleep Timer	Sleep Time	Rx Timer
0	868.350 MHz	(G)FSK	2.4 ksps	+/- 10 ppm	310 us	0	Off	NA	Off
1	868.350 MHz	(G)FSK	19.2 ksps	+/- 20 ppm	310 us	0	Off	NA	Off
2	868.350 MHz	(G)FSK	19.2 ksps	+/- 20 ppm	310 us	0	Off	NA	Off
3	430.800 MHz	(G)FSK	100.0 ksps	+/- 20 ppm	310 us	0	On	3 ms	On

Chip Parameters

RF Settings

Frequency (300-960): 868.350 MHz
Demodulation: (G)FSK
Symbol Rate (0.1-100.0): 4.8 ksps
Squelch TH (0-255): 0
Xtal Tol. (0-300) | Rx BW: +/- 10 ppm | 200 kHz
Xtal Stabilizing Time: 310 us

Operation Settings

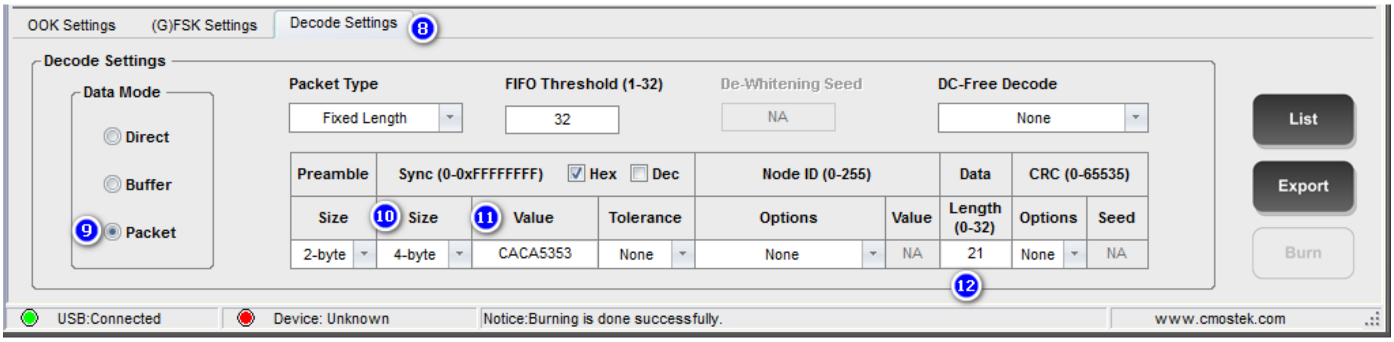
Operation Mode: ○ Active ● Passive
Duty-Cycle Mode: Off
Sleep Timer: Off
Sleep Time (3-134152192): NA ms
Rx Timer: Off
Rx Time (0.04-2683043): NA ms
Rx Time Ext (0.04-2683043): NA ms
Rx Early-Exit: Off
State After Rx Exit: NA
System Clock Output: Off
System Clock Frequency: NA MHz
Wake-On Radio: Off
Wake-On Condition: NA

OOK Settings | (G)FSK Settings | Decode Settings

(G)FSK Settings

Deviation (12.4-200.0): 35.0 kHz
Sync Clock Type: Tracing
Data Representation: 0:F-low 1:F-high
Rising Relative TH: 21
Falling Relative TH (0-255): 255
AFC: On

USB:Connected | Device: Unknown | Notice:Burning is done successfully. | www.cmostek.com



OOK Settings | (G)FSK Settings | Decode Settings

Decode Settings

Data Mode: ○ Direct ○ Buffer ● Packet
Packet Type: Fixed Length
FIFO Threshold (1-32): 32
De-Whitening Seed: NA
DC-Free Decode: None

Preamble	Sync (0-0xFFFFFFFF) ☑ Hex ☐ Dec			Node ID (0-255)		Data	CRC (0-65535)	
Size	Size	Value	Tolerance	Options	Value	Length (0-32)	Options	Seed
2-byte	4-byte	CACA5353	None	None	NA	21	None	NA

USB:Connected | Device: Unknown | Notice:Burning is done successfully. | www.cmostek.com

a) Select the Advanced Mode tab.

b) Corresponding to the CMT2157A, the configuration frequency is 868.35MHz.

c) Modulation mode is FSK.

d) The corresponding rate is 4.8Ksps.

e) Select Passive mode. That CMT2219A is working in a passive control mode that must have MCU participation (the default is Passive).

f) The corresponding configuration frequency deviation is 35KHz.

g) Select Tracing decoding mechanism (Tracing will have better sensitivity of experience, of course is the premise of the transmitter encoding accurately, while CMT2157A can guarantee this point. if the transmitter coding is error or larger, it is recommended to switch to Counting).

h) Select data packet processing configuration " Decode Settings ".

i) Use the data packet hardware processing mechanism, namely Packet mode.

j) Packet synchronization bit is 4 bytes

k) Synchronization value is 0xCACA5353, this is based on the 1920 encoding format. See 《AN112 CMT2150A Configuration Guideline》 in detail.

l) Packet length is 21Byte. The principle of this calculation is as follows:

PayloadLength = [(ID_Length + Key_Length) * Bit_Format + CRC-8] / 8

You need rounding plus 1 if the results are non integer.

Including: PayloadLength is the message length for CMT2219A.

ID_Length is the ID bit for CMT2157A.

Key_Length is the key bit for CMT2157A. Use the Normal way according to this case, that is 4bits.

Bit_Format is the 1920 coded symbol format for CMT2157A. There are 3, 4, 5, 6, a total of 4 kinds.

CRC-8 is the 8 bits CRC check information for CMT2157A coding chip. They are 8Symbol.

For example:

ID_Length = 27, Key_Length = 4, Bit_Format = 5, CRC-8 = 8，

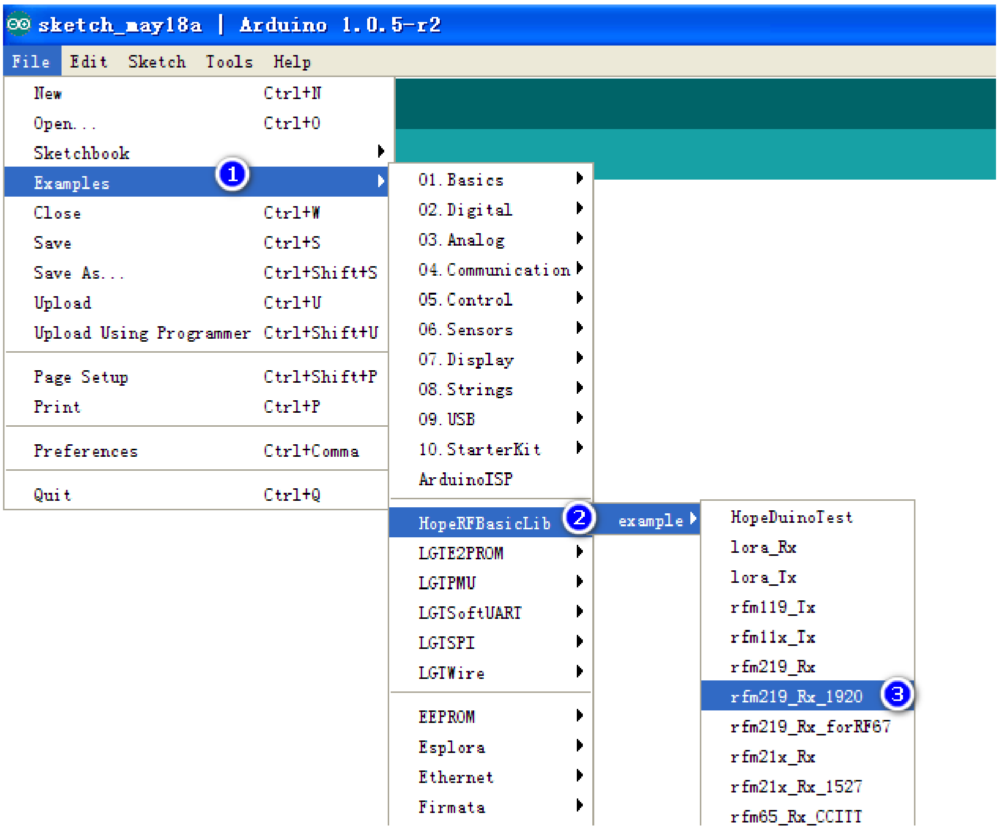
Therefore,

PayloadLength = ((27+4) * 5 + 8) / 8 = 20.375 ≈ 21

4. Hands-on Experiment

- Insert module RFM219S（with conversion board）into HopeDuino board
- Connect the HopeDuino boards to PC with USB cable.
- Open Arduino IDE interface, Click 【Files】->【Examples】->【HopeRFBasicLib】->【example】->【rfm219_Rx_1920】, as shown below.

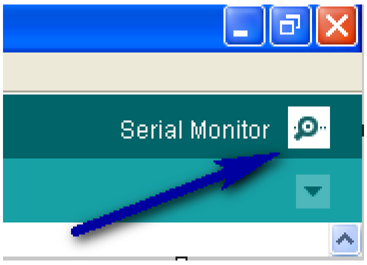
⚠ Notice: You couldn't find [HopeRFBasicLib] in [Examples] because you didn't install the HSP provided by HopeRF. Please refer to 《AN0002-HopeDuino Platform Construction Guideline》.



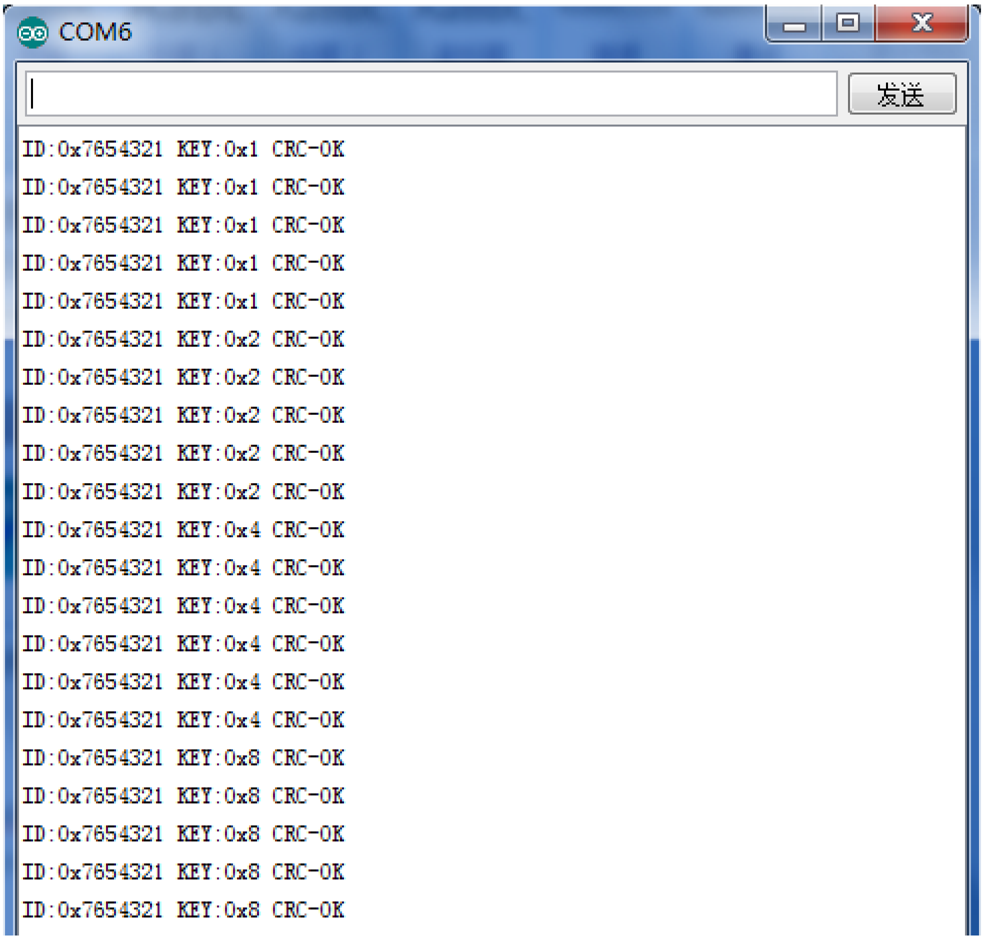
- ➢ At this time the program has been opened, please compile the download program according to the corresponding COM port.

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

- ➢ After the programs are downloaded, the Tx board will transmit a packet of data by pressing any key of K1～ K4 on the CMT2157A-EM board. The Rx board will receive a packet of data through module RFM219S periodically, and analyze the packet data, 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.



```
ID:0x7654321 KEY:0x1 CRC-OK
ID:0x7654321 KEY:0x1 CRC-OK
ID:0x7654321 KEY:0x1 CRC-OK
ID:0x7654321 KEY:0x1 CRC-OK
ID:0x7654321 KEY:0x1 CRC-OK
ID:0x7654321 KEY:0x2 CRC-OK
ID:0x7654321 KEY:0x2 CRC-OK
ID:0x7654321 KEY:0x2 CRC-OK
ID:0x7654321 KEY:0x2 CRC-OK
ID:0x7654321 KEY:0x2 CRC-OK
ID:0x7654321 KEY:0x4 CRC-OK
ID:0x7654321 KEY:0x4 CRC-OK
ID:0x7654321 KEY:0x4 CRC-OK
ID:0x7654321 KEY:0x4 CRC-OK
ID:0x7654321 KEY:0x4 CRC-OK
ID:0x7654321 KEY:0x4 CRC-OK
ID:0x7654321 KEY:0x8 CRC-OK
ID:0x7654321 KEY:0x8 CRC-OK
ID:0x7654321 KEY:0x8 CRC-OK
ID:0x7654321 KEY:0x8 CRC-OK
ID:0x7654321 KEY:0x8 CRC-OK
```

⚠ Notice:

a) 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.

b) ID is 0x1234567 configured by RFPDK. On the receiving side the order is reversed, the ID is 0x7654321.

c) "CRC-OK" means the data is verified and successful. If the data cannot be verified, the same will show the data, but the follow-up is “CRC-Fail"

5. Program Explanation

➢ Rfm219_Rx_1920.ino Case explanation

```
#include <HopeDuino_CMT2219A.h>      // Call the corresponding library file.
                                      //Calling UART is added because of using UART.
#include <HopeDuino_UART.h>
cmt2219aClass radio;                  // Define variable radio for CMT2119A
uartClass uart;                       // Define variable uart for UART

byte CfgTbl[62] = {
                    0x72,      //  Mode             = Advanced
                    0x42,      //  Part Number      = CMT2219A
                    0x44,      //  Frequency        = 868.350 MHz
                    0x15,      //  Demodulation     = (G)FSK
                    0x0D,      //  Symbol Rate      = 4.8 ksps
```

```
                0x63,           //  Xtal Tolerance          = +/- 10 ppm
                0x9A,           //  Xtal Stabilizing Time   = 310 us
                0x80,           //  Squelch TH              = 0
                0xC6,           //  Sleep Timer             = Off
                0xA9,           //  Sleep Time              = NA
                0x00,           //  Rx Timer                = Off
                0x00,           //  Rx Time                 = NA
                0x62,           //  Rx Time Ext             = NA
                0x2E,           //  Rx Early Exit           = Off
                0x00,           //  State After Rx Exit     = NA
                0x90,           //  System Clock Output     = Off
                0x84,           //  System Clock Frequency = NA
                0x14,           //  Wake-On Radio           = Off
                0xE0,           //  Wake-On Condition       = NA
                0x00,           //  Demod Method            = NA
                0x27,           //  Fixed Demod TH          = NA
                0x9F,           //  Peak Drop               = NA
                0x53,           //  Peak Drop Step          = NA
                0x53,           //  Peak Drop Rate          = NA
                0xCA,           //  Deviation               = 35.0 kHz
                0xCA,           //  Sync Clock Type         = Tracing
                0x00,           //  Data Representation     = 0:F-low   1:F-high
                0x3C,           //  Rising Relative TH      = 21
                0x01,           //  Falling Relative TH     = 255
                0x01,           //  AFC                     = On
                0x55, //Length  //  Data Mode               = Packet
                0x21,           //  Packet Type             = Fixed Length
                0x07,           //  FIFO Threshold          = 32
                0x84,           //  De-Whitening Seed       = NA
                0x00,           //  DC-Free Decode          = None
                0x00,           //  FILE CRC                = AAAD
                0x19,
                0x00,
                0x00,   0x00,   0xAC,   0xAE,   0x53,   0xD4,   0x40,   0x49,   0xFF,   0x1B,
                0x12,   0x00,   0x90,   0xFA,   0x00,   0x00,   0x40,   0xC0,   0x00,   0x00,
                0x20,   0xCA,   0x07,   0x00
                };

byte getstr[32];                // Define pending data buffer
byte hexstr[5];             //Analyze pending data buffer
byte sendstr[32];               // Define sending message buffer
byte length;                    //Define the message length
byte keycode;               //Define the extracting key value
byte crc_rx;                    //Define the extracting CRC value
byte crc_calc;                     //CRC calculation value

byte BitFormat;                         //Configuration range is [3-6] for 1920 coding format
byte IDBitLength;                       //Configuration range is [1-32] for 1920 coding ID length

void setup()
{
BitFormat       = 5;               // Configure 5 symbol encoding formats for CMT2157
IDBitLength    = 27;               // ID bit length is set to 27 bits

radio.CrcDisable          = true;       //Disable checking CRC
radio.FixedPktLength    = true;         // Fixed length message format
radio.NodeDisable        = true;      //Disable Node ID
radio.PktLength            = 21;   //CMT2157A-1920: [((ID-Length+4)*BitFormat)+8]/8      "4" for KeyValue
                                        // For example:    ID-Length=27bits, BitFormat=5Symbol/bit，
                                        //therefore:      ((27+4)*5+8)/8 = 21Byte


radio.vInit(CfgTbl);               // Execute initializing CMT2219A.
                                        // Input is the configuration table above.
radio.vGpioFuncCfg(GPIO1_INT1|GPIO2_DCLK|GPIO3_CLK|GPIO4_Dout);
                                        //configure GPIO, GPIO1 is INT1,
                                            //GPIO2 is DCLK (Demodulation data synchronous clock output),
                                            //GPIO3 is CLK (clock division),
                                            //GPIO4 is DATA (Demodulation data stream output).
radio.vIntSourcCfg((FIFO_WBYTE+OFFSET), 0);         // INT1 selecting configuration is WBYTE interrupt,
                                                  // each time a byte is received to generate an interrupt signal.

radio.vEnableIntSource(0xFF);           //Enable full interrupt source
radio.vGoRx();                     // Enter receiving state
uart.vUartInit(9600, _8N1);             //Initialize UART, parameters are 9600 baud rate and 8N1 format.
}

void loop()
{
byte i;
if(radio.bGetMessage(getstr)!=0)   //Check radio whether to receive data function,
                                        //analyze data received.
     {
     for(i=0; i<5; i++)                 //Clear buffer
          hexstr[i] = 0x00;
     keycode = 0;
     crc_rx = bAnalysisMsg(getstr, BitFormat, IDBitLength, &keycode, hexstr);
                                                                              //analyze message and extract CRC
```

```
        if(bCalcCrc8(getstr, BitFormat, IDBitLength, crc_rx))
            vAssembleSendMsg(hexstr, IDBitLength, keycode, true, sendstr);
                                                //According to CRC comparison results show different messages
        else
            vAssembleSendMsg(hexstr, IDBitLength, keycode, false, sendstr);
        uart.vUartPutNByte(sendstr, length);                // Output the received data to PC via UART
        uart.vUartNewLine();                                    //UART newline is easy to read.
        }
}
```

The following focuses on explaining the specific functions in the program.

- **bAnalysisMsg**

  Type: Function

  Input: inptr[ ]， pointer, the pending data entrance.

  bit_format， byte, the 1920 coding format for CMT2157A.

  id_length， byte, the ID bit length for CMT2157A.

  *key， pointer, return the extracting key value from inptr.

  outptr[ ], pointer, the storage entrance of the extracting data according to 1920 coding format from inptr.

  Output: return the extracting CRC-8 value from inptr

  Function: From batch data that CMT2219A received from CMT2157A, ID, key value and CRC-8 information are extracted respectively according to 1920 coding characteristics.

- **vAssembleSendMsg**

  Type: Function

  Input: inptr[ ]， pointer, the pending data entrance

  id_length， byte, the ID bit length for CMT2157A

  key, byte, the key information for CMT2157A

  crc_ok， bool type, true represents data verification is passed, false represents data verification is not passed.

  outptr[ ]， pointer, the storage entrance of message data from the serial port.

  Output: None

  Function: Analyze the pending data; send message data from the serial port.

- **bChangeToAscii**

  Type: Function

  Input: ch, unsigned char, character to be converted

  Output: Convert ch into ASCII code

  Function: Convert ch into ASCII code and return. Call it for vAnalysisMsg in the program.

  The purpose is converting the received 1527 value into the corresponding ASCII code.

- **bCalcCrc8**

  Type: Function

  Input: inptr[ ]， pointer, the pending data entrance

  bit_format， byte, the 1920 coding format for CMT2157A

  id_length, byte, the ID bit length for CMT2157A crc_check， byte, the extracting CRC-8 value from message data that CMT2219A received from CMT2157A

Output: bool type, true represents crc verification is passed; false represents crc verification is not passed.

Function: Calculate CRC-8 value for chip CMT215xA

6. CMT2219A Library Function Description

“CMT221xA.h”and“CMT221xA.cpp”library files are stored in the Arduino IDE file\ libraries \ HopeRFLib.

- **FixedPktLength**

  Type: bool type

  Content: true, the message length is fixed (Corresponding need to set the message length, which is PktLength)

- **PktLength**

  Type: unsigned char

  Content: the received message length, it is applicable that FixedPktLength is true.

- **vInit**

  Type: Function

  Output: cfg[ ]， pointer, the array entrance to be configured.

  Output: None

  Function: Initialize module RFM219S (CMT2219A), call it at the beginning of program. The initialization is mainly for the IO configuration of MCU and writing configuration parameters to CMT2219A. Need to pay attention to, if the program call vSoftReset function to reset the chip, then you need re configure the parameter to CMT2219A after reset.

- **vGoRx**

  Type: Function

  Input: None

  Output: None

  Function: Let chip CMT2219A into the receiving state

- **vGoSleep**

  Type: Function

  Input: None

  Output: None

  Function: Let chip CMT2219A into sleep state

- **vGoStandby**

  Type: Function

  Input: None

  Output: None

  Function: Let chip CMT2219A into the standby state, and keep the crystal in the state of oscillation.

- **vSoftReset**

  Type: Function

  Input: None

Output: None

Function: Reset chip CMT2219A, this is software reset operation.

- **vClearFIFO**

  Type: Function

  Input: None

  Output: None

  Function: Clear FIFO content for CMT2219A, commonly used to operate after receiving and reading data.

- **bReadStatus**

  Type: Function

  Input: None

  Output: Return state, unsigned char， effective in grade three

  0x00-------Reset state（PUP state）

  0x20-------Sleep state（default）

  0x40------- Standby state（Standby/STBY）

  0x60------- Frequency synthesis state（Tune state）

  0x80------- Receiving state（Rx）

  0xA0-------EEPROM mode（EEPROM read and write mode in CMT2219A）

  Other undefined

  Function: Read the current state of the CMT2219A

- **bReadRssi**

  Type: Function

  Input: None

  Output: Signal strength value, unsigned char, range 0～255, the stronger the signal, the greater the value.

  Function: Read the current signal strength value

- **bReadIngFlag**

  Type: Function

  Input: None

  Output: Return the interrupt flag

  Function: Read the interrupt flag register, return the interrupt flag.

- **vClearIntFlag**

  Type: Function

  Input: None

  Output: None

  Function: Clear all interrupt flags

- **vGpioFuncCfg**

  Type: Function

  Input: io_cfg， unsigned char， configuring GPIO auxiliary functions for CMT2219A.

  Output: None

Function: Configuring four GPIO auxiliary functions for CMT2219A. See details of the CMT2219A specifications "Table18. IO_SEL Register"

- **vIntSourcCfg**

  Type: Function

  Input: int_1 & int_2， unsigned char， configuring interrupt source

  Output: None

  Function: Select the appropriate interrupt source for INT1 and INT2. See details of the CMT2219A specifications "Table19"

- **vEnableIntSource**

  Type: Function

  Input: en_int， unsigned char， enabling interrupt source

  Output: None

  Function: Enable interrupt source

- **bGetMessage**

  Type: Function

  Input: msg[ ]， pointer, array entrance to be received

  Output: Received data length

  Function: Receive a packet of data. Real-time call requirements is not high because querying interrupt source is used

7. Pin Assignment Table

HopeDuino	MCU	CMT2219A
13	PB5	SCL
12	PB4	FCSB
11	PB3	SDA
10	PB2	CSB
9	PB1	
8	PB0	GPO1
7	PD7	GPO2（jumper）
6	PD6	GPO4（jumper）
5	PD5	GPO3（jumper）
4	PD4	

8. Version Records

Version	Revised Contents	Date
1.0	Initial version	2016-03-29
1.1	Revise text bug, add watermarks, program explanations and descriptions	2016-04-06