CMT2119A & CMT2219A Communication Example

This chapter will guide the user to carry out the communication experiment on a pair of single transmitting and single receiving chips CMT2119A and CMT2219A. CMT2119A is the FSK/OOK modulated single transmitting chip belonging to HopeRF's CMOSTEK wireless product line below. CMT2219A is the corresponding FSK/OOK modulated single receiving chip, supports Sub-1G applications.

1. Tools and software needed to be prepared

SOT23-6

QFN16 (3x3)

- Arduino IDE version 1.0.5
- HopeDuino board(2 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)
- CMOSTEK USB Programmer（Option）
- CMOSTEK RFPDK V1.38（Pay attention to using the latest version. The latest version is V1.38 in the paper）
- Module RFM119 (Based on chip CMT2119A) , module RFM219S (Based on chip CMT2219A) and the matching conversion board

RFM219S

RFM119

2. CMT2119A and CMT2219A parameter configuration and recording

- Chip CMT2119A and chip CMT2219A parameters are configured through CMOSTEK RFPDK software. You choose any one of the following ways of recording parameters:

（1） USB Programmer burns parameters inside the chip EEPROM. This way is the general way supported by CMT211xA and CMT221xA. Chip parameters stored in the on-chip EEPROM never lost. The parameters are automatically loaded when the power is on. Chip work according to the preset parameters.

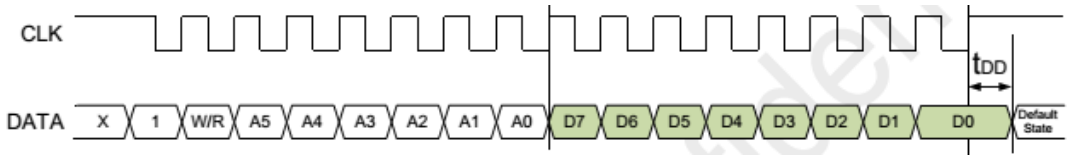
（2） The second way is to configure the parameters to the chip internal registers through the bus. It needs MCU intervention because of slave mode. In this way, only CMT2119A and CMT2219A are available in all chips CMT211xA and CMT221xA. CMT2119A parameters are configured by bus interface TWI(Two Wire Interface). They are composed of DAT and CLK ports. CMT2219A parameters are configured by bus interface SPI. They are composed of CSB、SDA、SCK and FCSB ports.

Notice:

1. The two ways have their own advantages and disadvantages. Using USB Programmer burning, easy to use, and the working parameters is fixed, can not change. By using MCU configuration mode, the MCU can be controlled flexibly by the program, so that more applications can be realized by configuring different working parameters. Such as: frequency hopping transmission (That needs the dynamic transmission of several frequency points of the data during a transmission process). This requires the user to choose according to specific needs.
2. Special attention should be paid when selecting the MCU configuration parameter. The MCU control command

through bus configuration will reset the chip in the use of reset commands (such as: SoftRst, etc.). During the reset operation, the working parameters stored in the register are lost. You need to reconfigure these parameters. Users need to pay special attention to it. If the working parameters are not configured after resetting, the chip will automatically load the default parameters of the EEPROM. (The specific default parameters see chip specifications)

- **TWI configuration instructions for CMT2119A**
  - Open CMOSTEK RFPDK software, select chip CMT2119A, configure the parameters needed for the job, such as: modulation mode(FSK/OOK), working frequency and transmitting power etc.
  - Click the "Export" button on the RFPDK interface; generate a parameter file with the suffix exp. This file can be used for USB Programmer burning parameters. The same is also used as a parameter for the MCU configuration. The exp file can be opened with a writing board. The main storage parameters are 16 bit data of hexadecimal.
  - If you configure the chip CMT2119A through the TWI, you need to understand the basic timing of the TWI, as shown below:



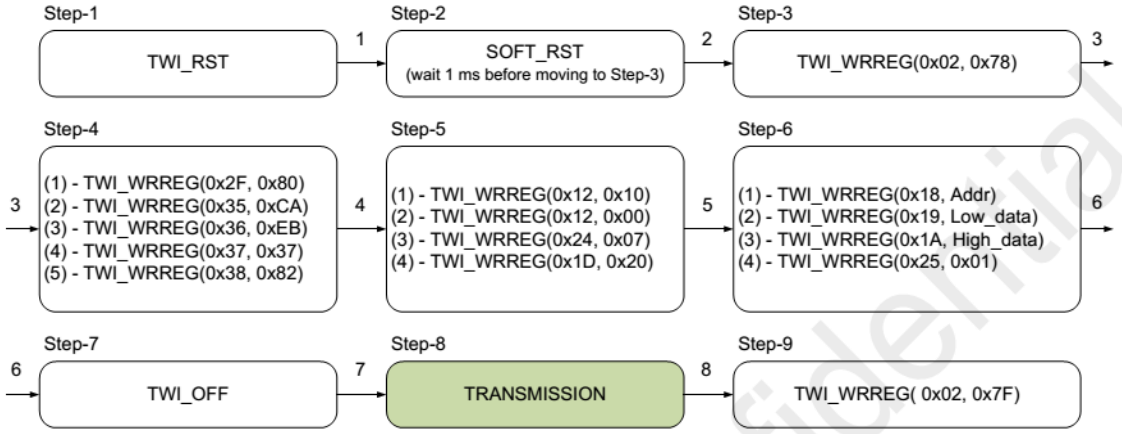
Notice: W/R = 1 represents reading operation; W/R = 0 represents writing operation.

- TWI two basic operations——reading and writing(TWI_WRREG and TWI_RDREG), as shown below:

Command	Description
TWI_WRREG	TWI write command. TWI_WRREG(XX, YY) means clocking in 16b'10xx xxxx yyyy yyyy, which xx xxxx is the register address to be written, ranging from 0x00 to 0x3F; yyyy yyyy is the register content to be written ranging from 0x00 to 0xFF. For example, TWI_WRREG(0x12, 0xAA) means clocking in 0x92AA.
TWI_RDREG	TWI read command, TWI_RDREG(XX, ZZ) means clocking in 8b'11xx xxxx and read out zzzz zzzz, which xx xxxx is the register address to be written, ranging from 0x00 to 0x3F; zzzz zzzz is the read out value from the register, ranging from 0x00 to 0xFF For example, TWI_RDREG(0x2A, DAT), means clocking in 0xEA, and read out DAT which is an 8-bit value.

Notice: They are the basic reading and writing operations of the TWI bus. You couldn't directly use the two operations to write the parameter of the exp file to the register. In fact, writing the parameters of the exp file to CMT2119A is a relatively complex operation process. You need to read further in this article.

- CMT2119A Configuration Flow



Above is the configuration flow chart of CMT2119A. From Step1 to Step7 is a complete configuration process. Step1 ~ Step5 is a fixed operation process. Only Step6 is the formal configuration of the parameters of the exp file to the CMT2119A. But Step1 ~ Step5 as a fixed process must be completed before operating Step6.

So we focus on how the Step6 configures the parameters of the exp file to CMT2119A. Step6 includes 4 TWI_WRREG operations. They are specified as follows:

（1）TWI_WRREG（0x18， Addr） // Write the parameter (16 bit) to the 0x18

（2）TWI_WRREG（0x19， Low_data）// Write the low 8 bit data of the parameter (16 bit) to the 0x19

（3）TWI_WRREG（0x1A，High_data）// Write the high 8 bit data of the parameter (16 bit) to the 0x1A

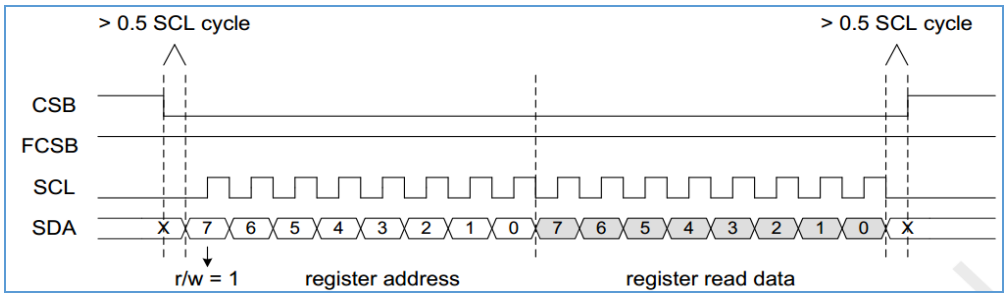
（4）TWI_WRREG（0x25，0x01） // Fixed operation indicates the completion of the above task

Notice:

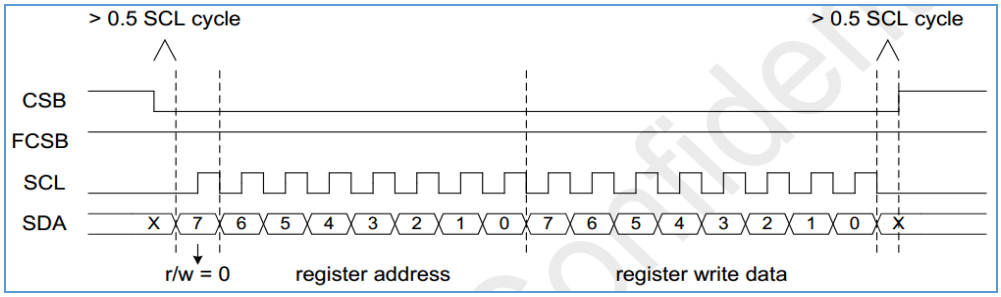
1. Step6 is only for one target object. If you want to complete entire parameters configuration of the exp file, you need a cycle of operation.
2. CMT2119A configuration parameters process is like a two mapping, not writing data simply based on the address. Each parameter must be configured through an indirect process.
3. Note that the address value is hidden in the file without parameters address description within exp file. The first parameter address is 0x00. The address range is 0x00 ~ 0x14. Parameters are 21 Data.

➢ **SPI configuration instructions for CMT2219A**

- Open CMOSTEK RFPDK software, Select chip CMT2219A, Configure the parameters needed for the job, such as: modulation mode (FSK/OOK), working frequency, communication rate and packet format etc.
- Click the "Export" button on the RFPDK interface. Generate two parameter files with the suffix exp. One of them can be used for USB Programmer burning parameters. Another suffix is the same, but the name contains "REG". It is used as a parameter for the MCU configuration. Attentive users may find that the exp file identifies the "Addr" and "Value", distinguishes address and parameter values.
- Users need to configure chip CMT2219A through the SPI-3（3 wire system SPI，CSB、SDA、SCK）. The basic writing and reading operation sequence is as shown below.



SPI-3 读取时序



SPI-3 写入时序

- CMT2219A configuration process is relatively simple to CMT2119A. You can input the parameters of exp file according to the "Addr" and "Value" directly through the above SPI-3 writing operation. The configuration is completed.

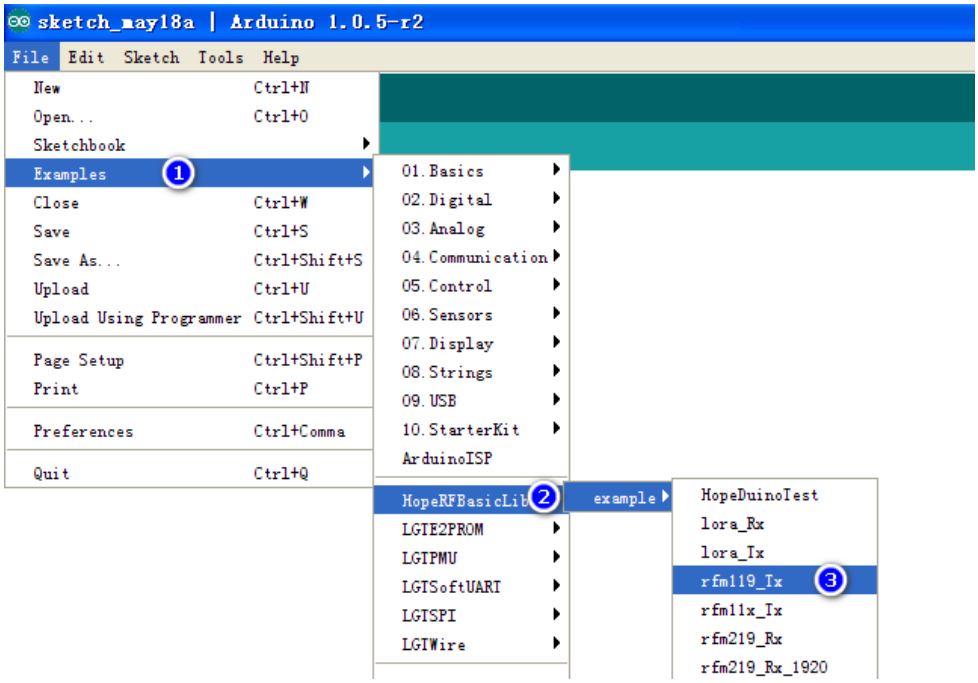
⚠ Notice:

1. This paper focuses on the communication example use of CMT2119A and CMT2219A in the HopeDuino, not for the specific use of CMT2119A and CMT2219A. If you need to understand the details of two models, please refer to the specifications.
2. If you want to know the configuration details of CMT2119A and CMT2219A in the RFPDK interface, please refer to 《AN122 CMT2113-19A Configuration Guideline》and《AN138 CMT2219A Configuration Guideline》of CMOSTEK。

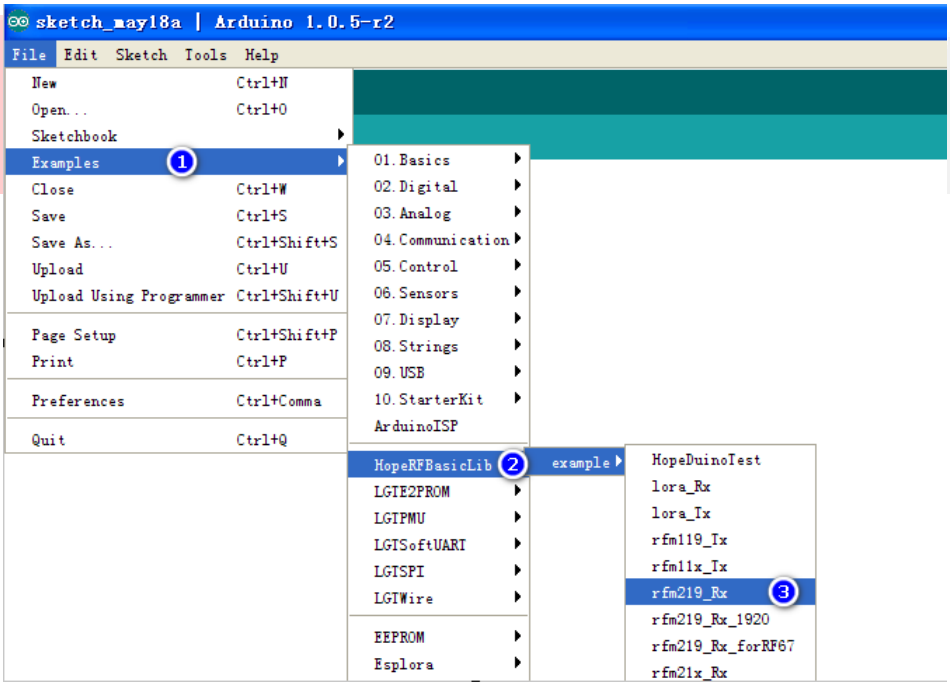
3. Hands-on Experiment

- Module RFM119S and module RFM219 (with conversion board) are inserted into the corresponding HopeDuino board respectively.
- Connect the HopeDuino boards to PC with USB cable;
- Open Arduino IDE interface, Click 【File】→【Examples】→【HopeRFBasicLib】→【example】→【rfm119_Tx】,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》.



➢ Open Arduino IDE interface, Click 【File】→【Examples】→【HopeRFBasicLib】→【example】→【rfm219_Rx】,as shown below.



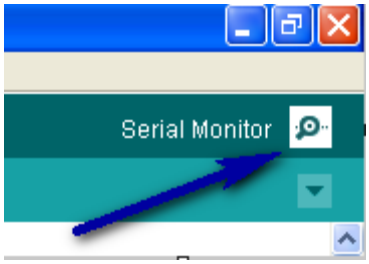
➢ At this time you have opened a Tx program and a Rx program, please compile the download programs according to the corresponding COM port.

⚠ Notice:

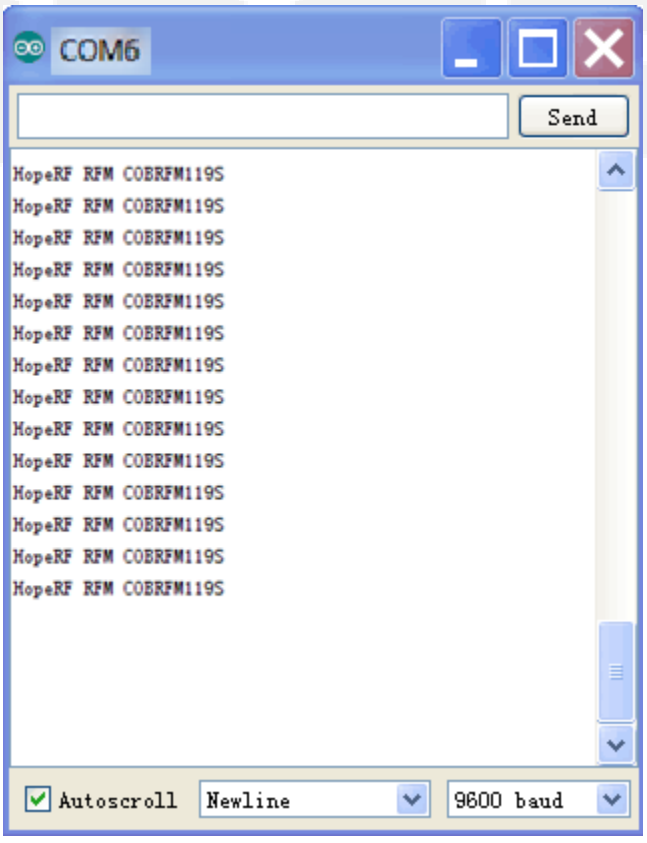
1. Do not know how to compile the download code, please refer to 《AN0002-HopeDuino Platform Construction Guideline》

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 RFM119. The Rx board will receive a packet of data through module RFM219S 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.



⚠ 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.

2. In the rfm119_Tx and rfm219_Rx programs, all parameters are derived from the exp file of RFPDK parameters as the configuration data table. In the rfm219_Rx program, the array omits the address value (addr) because the address is basic continuous. Users can set up their own parameters through the RFPDK, and then import them to the chip to do the appropriate experiment. Of course, it is recommended to read code, understand its function and then do the adjustment.

4. Program Explanation

- rfm119_Tx.ino Case explanation

```
#include <HopeDuino_CMT211xA.h>              //Call corresponding library file
cmt211xaClass radio;                                    // Define variable radio for CMT2119A
byte    str[31] = {                                     // Array to be transmitted
                0xAA, 0xAA, 0xAA, 0xAA, 0xAA, 0xAA, 0xAA, 0xAA, 0x2D, 0xD4,
                 'H',   'o',   'p',   'e',   'R',   'F',   ' ',   'R',   'F',   'M', ' ',   'C',   'O',   'B',   'R',   'F',   'M',   '1',
                 '1',   '9',   'S',
                };
word CfgTbl[21] = {             // Configuration parameter table generated by RFPDK CMOSTEK software
                        0x007F,             // Mode                 = Advanced
                        0x1400,             // Part Number          = CMT2119A
                        0x0000,             // Frequency            = 434 MHz
                        0x0000,             // Modulation            = FSK
                        0x0000,             // Symbol Rate          = 0.5-100.0 ksps
                        0xF000,             // Tx Power             = +14 dBm
                        0x0000,             // Deviation             = 35.0 kHz
                        0xC4EC,             // PA Ramping Time    = NA
                        0x4208,             // Xtal Cload            = 15.00 pF
                        0x0160,             // Data Representation = 0:F-low,1:F-high
                        0x2400,             // Tx Start by           = DATA Pin Falling Edge
                        0x0081,             // Tx Stop by            = DATA Pin Holding Low For 20 ms
                        0x8000,             // Increase XO Current    = No
                        0x0000,             // FILE CRC              = D150
                        0xFFFF,
                        0x0020,
                        0x5FE8,
                        0xA2D6,
                        0x0E13,
                        0x0019,
                        0x0000
                        };

void setup()                        // Initialization function “setup” is required in the Arduino software architecture
{
radio.Chipset           = CMT2119A;       //Define chip as CMT2119A
```

```
radio.SymbolTime        = 416;              // Symbol time is 2.4Ksps (that is 416us)
 radio.vCMT2119AInit(CfgTbl, 21);           //Initialize CMT2119A, input is the configuration table above.
radio.vEncode(str, 31, ENRZ);              // Execute radio encoding function,
                                              //the source data is srt, the length is 7 bytes.
                                              //Encoding uses NRZ(Non return to zero) encoding format.
}


void loop()                                 //Loop function is required in the Arduino software architecture

{
radio.vTxPacket();                           // Transmit a packet of data
   _delay_ms(100);                           //Delay 100ms.
                                            //That means it will transmit a packet of data automatically every 100ms.
}
```

➢ rfm219_Rx.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 getstr[21];                            //Define pending data buffer
byte CfgTbl[62] = {              // Configuration parameter table generated by RFPDK CMOSTEK software
                    0x72,              // Mode                    = Advanced
                    0x21,              // Part Number             = CMT2219A
                    0x9B,              // Frequency               = 434.000 MHz
                    0x7F,              // Demodulation            = (G)FSK
                    0x06,              // Symbol Rate             = 2.4 ksps
                    0x63,              // Xtal Tolerance          = +/- 20 ppm
                    0x9A,              // Xtal Stabilizing Time   = 310 us
                    0x80,              // Squelch TH              = 0
                    0xC6,              // Sleep Timer             = Off
                    0x53,              // Sleep Time              = NA
                    0x01,              // Rx Timer                = Off
                    0x00,              // Rx Time                 = NA
                    0x62,              // Rx Time Ext             = NA
                    0x1E,              // Rx Early Exit           = Off
                    0x00,              // State After Rx Exit     = STBY
                    0x10,              // 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
                0x00,           // Peak Drop Step          = NA
                0xD4,           // Peak Drop Rate          = NA
                0x2D,           // Deviation               = 35.0 kHz
                0xAA,           // Sync Clock Type         = Counting
                0x00,           // Data Representation     = 0:F-low  1:F-high
                0x38,           // Rising Relative TH      = 21
                0x00,           // Falling Relative TH     = 255
                0x01,           // AFC                     = Off
                0x55,           // 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                = DB2F
                0x19,
                0x00,
                0x00,  0x00,  0xAC,  0x56,  0x53,  0xD4,  0x40,  0x49,  0xFF,
                0x5D,  0x12,  0x00,  0x90,  0xFA,  0x00,  0x00,  0x40,  0xC0,
                0x00,  0x00,  0x20,  0xEB,  0x07,  0x00
        };

void setup()
{
radio.CrcDisable        = true;         //Disable CRC for CMT2219A
radio.FixedPktLength = true;            // Define CM2219, the length of the received message is fixed.
radio.NodeDisable     = true;           //Define CMT2219A, received message is not with NodeID function
radio.PktLength         = 21;           //Define CMT2219A, received message length is 21 bytes.
radio.vInit(CfgTbl);                    // Initialize 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()
```

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

⚠ Notice: Users can make their own attempt to modify the rate and data content in order to deepen understanding of this case. You pay attention to the correspondence between transmitting and receiving. At the same time, because the configuration parameters of CMT2119A and CMT2219A are generated by the RFPDK software, the configuration table of the program needs to be updated.

5. CMT2119A Library Function Description

CMT2119A library functions are"CMT211xA.h"and"CMT211xA.cpp", shared with CMT211xA. Files are stored in IDE Arduino files \ libraries \ HopeRFLib.

- **chipsetType**

  Type: Enumeration type

  Function: Select chip model

  Contents: CMT2110A、CMT2113A、CMT2117A、CMT2119A

- **encodeType**

  Type: Enumeration type

  Function: Select encoding type

  Contents: ENRZ、E527、E201

  ENRZ——Represents NRZ（Non-Return-Zero）encoding format. Simple explanation is not coding. "1" is the high level. "0" is the low level.

  E527——Represents 1527/527 encoding format. Each bit consists of 4 unit time, shown as below:

  E201——Represents a code consisting of 3 unit time, shown as below:

- **Chipset**

  Type: chipsetType, **enumeration type**

  Function: Define chip model

- **SymbolTime**

  Type: unsigned int

  Function: Define rate, which is symbol time (SYM for short). The unit is microsecond (us). The setting range is

10~4000.

- **vCMT211xAInit**

  Type: Function

  Input: None

  Output: None

  Function: The initialization is suitable for module CMT2110A, CMT2113A and CMT2117A, call it at the beginning of program. The initialization is mainly for the IO configuration and reset operation for MCU. It cannot configure the working parameters of the CMT211xA. These parameters must be made by RFPDK software through USB Programmer.

- **vCMT2119AInit**

  Type: Function

  Input: para[ ], array pointer, the array entrance.

  length, unsigned char, the data length of the array to be configured.

  Output: None

  Function: Initialize CMT2119A, call it at the beginning of the program. Parameters are derived from the RFPDK software.

- **vCMT211xASleep**

  Type: Function

  Input: None

  Output: None

  Function: Configure chip to enter low power consumption state for CMT2110A, CMT2113A and CMT2117A. It is suitable for the situation with reducing electricity consumption and getting the chip (module) into hibernation.

- **vCMT2119ASleep**

  Type: Function

  Input: None

  Output: None

  Function: Let chip CMT2119A to enter low power consumption state. It is suitable for the situation with reducing electricity consumption and getting the chip (module) into hibernation.

- **vEncode**

  Type: Function

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

  Length, unsigned char, the data length to be encoded, the unit is byte**.**

  Etype, encodeType, enumeration type, select encoding format（ENRZ、E1527、E201）.

  Output: None

  Function: Encoding the data to be transmitted according to the specified encoding format.

- **vTxPacket**

  Type: Function

Input: None

Output: None

Function: Transmit the coded data. You must call vEncode before calling the function. Two functions must be used successively. Call once, and transmit a packet of data. And then return to sleep. If you need to transmit multiple times, you need to repeat the cycle call.

⚠ Notice:

1. If you do not want to configure CMT2119A, you can configure it according to AN1001 document mode (CMT211xA control mode).
2. In this program, we consider the hardware decoding function in the CMT2219A, so we use the NRZ coding format. If users change the encoding format to other types (such as E527, etc.), the receiver will have the confusion data due to the different coding format.

6. CMT2219A Library Function Description

"CMT2219A.h"and"CMT2219A.cpp"library files are stored in the Arduino IDE file\ libraries \ HopeRFLib

- **FixedPktLength**

  Type: bool type

  Content: ture, indicates that the received message is the fixed length (PktLength is the target message length).

- **PktLength**

  Type: unsigned char

  Content: received message length. It is suitable for the condition that FixedPktLength is true.

- **vInit**

  Type: Function

  Input: cfg[ ], pointer, the entrance of array

  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 the 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，configure 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，enable 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

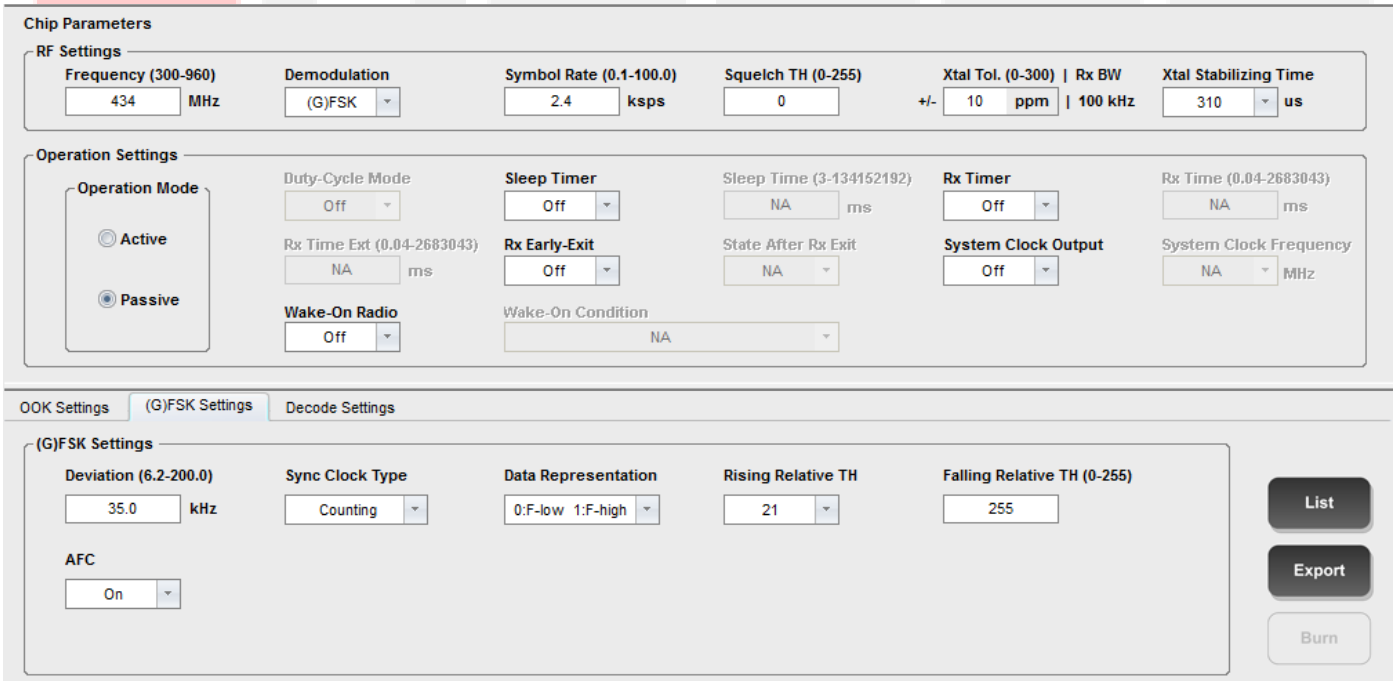
7. Pin Assignment Table

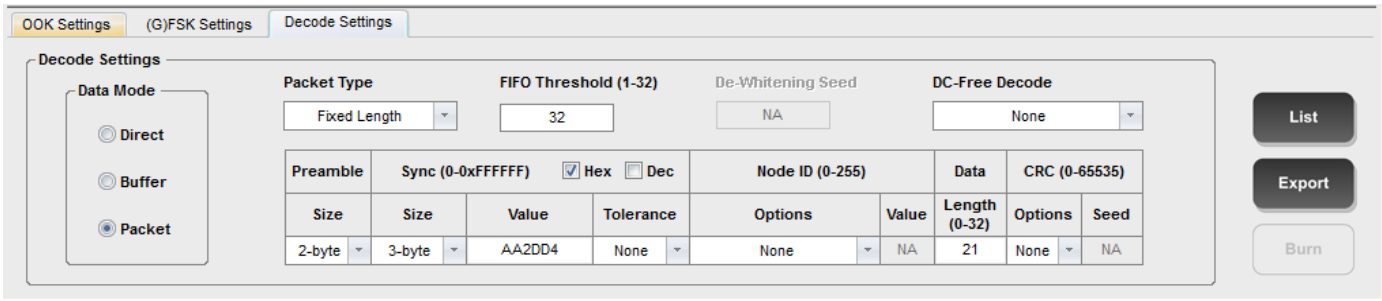
HopeDuino	MCU	CMT2119A	CMT2219A
13	PB5		SCL
12	PB4	CLK	FCSB
11	PB3		SDA
10	PB2		CSB
9	PB1		
8	PB0		GPO1
7	PD7		GPO2
6	PD6	DAT	GPO4
5	PD5		GPO3
4	PD4		

8. Version Records

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

Appendix I、CMT2219A configuration screenshot





Appendix II、CMT2119A configuration screenshot