Ra-01SCH-P

Usage Tutorial

Usage Tutorial of Ra-01SCH-P

Introduction

LoRa is a low-power wide-area network (LPWAN) wireless standard developed by Semtech. Its name "LoRa" stands for Long Range Radio.

Its greatest characteristic is that under the same power consumption conditions, it can achieve longer transmission distances than other wireless methods, realizing the unification of low power consumption and long range. Under the same power consumption, it can extend the communication distance of traditional wireless RF communication by 3-5 times.

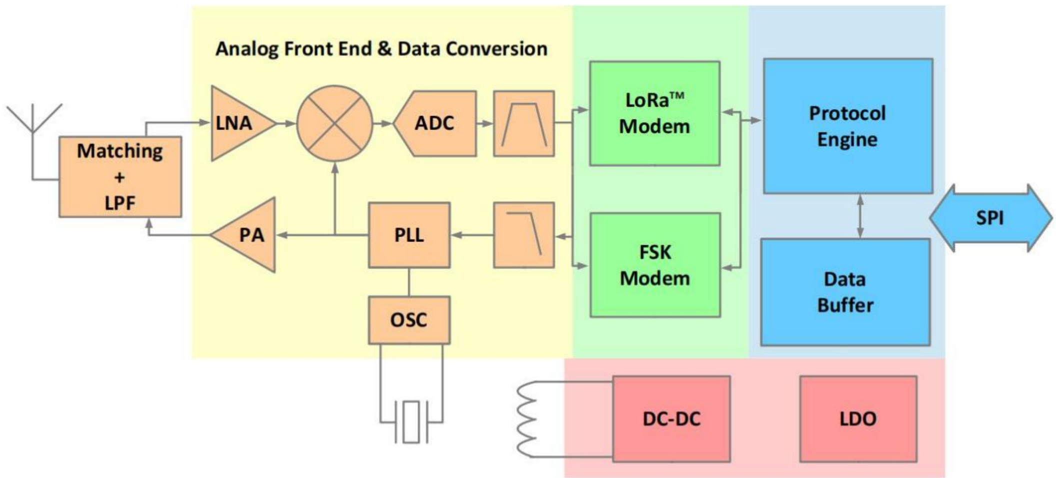
Product Overview

The Ra-01SCH-P is a LoRa series module designed and developed by Ai-Thinker. This module is used for ultra-long-distance spread spectrum communication.

Its RF chip LLCC68+ primarily uses the LoRa™ long-range modem for ultra-long-distance spread spectrum communication, with strong anti-interference capability and minimal current consumption. Leveraging Semtech's patented LoRa™ modulation technology, the module integrates a Power Amplifier (PA) and Low Noise Amplifier (LNA), achieving high sensitivity exceeding -137dBm and +29dBm transmit power for long transmission distances and high reliability. Meanwhile, compared with traditional modulation technology, LoRa™ modulation technology also has obvious advantages in anti-blocking and selectivity, solving the problem that traditional design solutions cannot simultaneously balance distance, anti-interference, and power consumption.

Application Scenarios

Can be widely used in automatic meter reading, home and building automation, security systems, remote irrigation systems, etc.



Features

- Supports FSK, GFSK, LoRa® modulation methods;
- Supports frequency band 803MHz~930MHz;
- Maximum transmit power is optional

① Default configuration: internal PA uses 3.3V supply voltage; in this state, Tx Power can reach up to +29dBm with an operating current of 750mA;

② Optional configuration: internal PA uses 5V supply voltage; in this state, Tx Power can reach up to +31dBm with an operating current of 1A;

- High sensitivity: as low as -137dBm@SF10 125KHz;
- Extremely small size 17163.2(±0.2)MM, dual-row stamp hole patch package;
- Supports spreading factors SF5/SF6/SF7/SF8/SF9/SF10/SF11;
- Low power consumption in receiving state, with receive current as low as 16mA;
- Module uses SPI interface, half-duplex communication, with CRC, up to 256-byte packet engine;
- Supports multiple antenna installation methods, compatible with half-hole pads/through-hole pads/IPEX connector;

I. Software and Hardware Introduction

1. Code Download and Analysis

Obtain the relevant demo through the following link:

https://aithinker-static.oss-cn-shenzhen.aliyuncs.com/docs/example/LLCC68-Ra-01SC-P_Ra-01SCH-P_Driver-V1.0.1.zip

The function for initializing hardware is shown below:

```c
UART:
  USB to TTL      STM32
  Tx              PA_9
  Rx              PA_10
*/

//硬件初始化
void SysInit(void){
  GPIO_InitTypeDef  GPIO_InitStructure;

  NVIC_PriorityGroupConfig(NVIC_PriorityGroup_4); //NVIC(中断优先级管理)分组配置,注意:这个分组整个程序只能有一次,配置后不要修改,否则会出现很多问题 这里直接用4,0~15优先级
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);      //JTAG复用为GPIO需要使用 RCC_APB2Periph_AFIO 时钟
  GPIO_PinRemapConfig(GPIO_Remap_SWJ_JTAGDisable,ENABLE); //关闭JTAG功能(需要打开 RCC_APB2Periph_AFIO 时钟)

  //led指示灯
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_12;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;   //推挽输出
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;  //IO口速度为50MHz
  GPIO_Init(GPIOB, &GPIO_InitStructure);
  GPIO_SetBits(GPIOB,GPIO_Pin_12);       //PB.12 输出高

  HALUart1Init();
  SysTick_Config(SystemCoreClock/1000);
}
```

初始化硬件函数

The NVIC_PriorityGroupConfig() function configures the interrupt priority grouping to 4, allocating all 4 bits for preemptive priority; PB12 pin is configured as a light control pin to indicate when the program sends or receives data; the clock is configured to execute the interrupt function every 1ms, with the interrupt function content shown below:

```c
void PendSV_Handler(void)
{
}

void SysTick_Handler(void)
{
    systickCount++;
}

uint32_t Get_SysTick(void)
{
```

The main function is shown below:

```c
int main(void){
  SysInit();  //硬件初始化

  printf("SysInit OK, version:%s\r\n",SOFT_VERSION);

  //测试demo, 一个程序只能打开一条测试demo, 进入测试demo后将进入死循环, 不会返回了
  ExampleLLCC68ReciveDemo();  //循环接收demo

//  ExampleLLCC68SendDemo();  //定时发送demo

  //开启测试demo后代码就执行不到这里了
  while(1){
    printf("systick=%d\r\n",Get_SysTick());
    GPIO_ResetBits(GPIOB,GPIO_Pin_12);
    delay_ms(500);
    GPIO_SetBits(GPIOB,GPIO_Pin_12);
    delay_ms(500);
  }
}
```

Where the functions ExampleLLCC68ReciveDemo() and ExampleLLCC68SendDemo() are used to set the Ra-01SCH-P module to select whether to use the module for receiving or sending;

1) ExampleLLCC68ReciveDemo() Function

The content of the ExampleLLCC68ReciveDemo() function is shown below:

```c
//开启一个定时发送任务，每隔1s发送一条数据
void ExampleLLCC68ReciveDemo(void){
  uint32_t u32_count=0;
  uint8_t OCP_Value = 0;
  printf("start %s() example\r\n",__func__);

  LLCC68RadioEvents.TxDone = LLCC68OnTxDone;
  LLCC68RadioEvents.RxDone = LLCC68OnRxDone;
  LLCC68RadioEvents.TxTimeout = LLCC68OnTxTimeout;
  LLCC68RadioEvents.RxTimeout = LLCC68OnRxTimeout;
  LLCC68RadioEvents.RxError = LLCC68OnRxError;

  Radio.Init( &LLCC68RadioEvents );
  Radio.SetChannel(LORA_FRE);
  Radio.SetTxConfig( MODEM_LORA, LORA_TX_OUTPUT_POWER, 0, LORA_BANDWIDTH,
                     LORA_SPREADING_FACTOR, LORA_CODINGRATE,
                     LORA_PREAMBLE_LENGTH, LORA_FIX_LENGTH_PAYLOAD_ON,
                     true, 0, 0, LORA_IQ_INVERSION_ON, 3000 );//参数：lora模式,发射功率,fsk用的lora设置为0就可以，带宽，纠错编码率，前导码长度，固定长度数据包(一般是不固定的所以选false)，crc校验，

  OCP_Value = Radio.Read(REG_OCP);
  printf("[%s()-%d]read OCP register value:0x%04X\r\n",__func__,__LINE__,OCP_Value);

  Radio.SetRxConfig( MODEM_LORA, LORA_BANDWIDTH, LORA_SPREADING_FACTOR,
                     LORA_CODINGRATE, 0, LORA_PREAMBLE_LENGTH,
                     LORA_LLCC68_SYMBOL_TIMEOUT, LORA_FIX_LENGTH_PAYLOAD_ON,
                     0, true, 0, 0, LORA_IQ_INVERSION_ON, false );

  printf("start enter rx mode\r\n");
  Radio.Rx( 0 );  //进入接收模式
  printf("all setting\r\n");
  printf("freq: %d\r\n Tx power: %d\r\n band width: %d\r\n FS: %d\r\n CODINGRATE: %d\r\n PREAMBLE_LENGTH: %d\r\n",LORA_FRE,LORA_TX_OUTPUT_POWER,LORA_BANDWIDTH,LORA_SPREADING_FACTOR,LORA_CODINGRATE
  while(1){
    Radio.IrqProcess( ); // Process Radio IRQ
    u32_count++;
    delay_ms(1);
  }
}
```

Five callback functions are registered:

- LLCC68OnTxDone(): Callback function executed when data transmission is completed;
- LLCC68OnRxDone(): Callback function executed when data reception is completed;
- LLCC68OnTxTimeout(): Callback function executed after data transmission timeout;
- LLCC68OnRxTimeout(): Callback function executed after data reception timeout;
- LLCC68OnRxError(): Callback function executed after data reception error;

The Radio.Init(&LLCC68RadioEvents) function registers the above five callback functions. The implementation of this function is shown below;

```c
void RadioInit( RadioEvents_t *events )
{
    RadioEvents = events;
    LLCC68IoInit();

    LLCC68Init( RadioOnDioIrq );
    LLCC68SetStandby( STDBY_RC );
    LLCC68SetRegulatorMode( USE_DCDC );

    LLCC68SetBufferBaseAddress( 0x00, 0x00 );
    LLCC68SetTxParams( 0, RADIO_RAMP_200_US );
    LLCC68SetDioIrqParams( IRQ_RADIO_ALL, IRQ_RADIO_ALL, IRQ_RADIO_NONE, IRQ_RADIO_NONE );

    LLCC68TimerInit();

    IrqFired = false;
}
```

The Radio.SetChannel(LORA_FRE) function is used to set the RF frequency of the Ra-01SCH-P module. The function implementation is shown below;

```c
void RadioSetChannel( uint32_t freq )
{
    LLCC68SetRfFrequency( freq );
}
```

The Radio.SetTxConfig(MODEM_LORA, LORA_TX_OUTPUT_POWER, 0, LORA_BANDWIDTH, LORA_SPREADING_FACTOR, LORA_CODINGRATE, LORA_PREAMBLE_LENGTH, LORA_FIX_LENGTH_PAYLOAD_ON, true, 0, 0, LORA_IQ_INVERSION_ON, 3000) function is used to set the TX mode parameters of the Ra-01SCH-P module.

Function parameters:

LoRa mode, transmit power, FSK mode parameter (set to 0 for LoRa mode), bandwidth, spreading factor, error correction coding rate, preamble length, fixed-length packet (default false), CRC check, 0 means frequency hopping disabled, number of symbols between frequency hops (meaningless when frequency hopping is disabled). This function implementation is shown below:

```c
void RadioSetRxConfig( RadioModems_t modem, uint32_t bandwidth,
                         uint32_t datarate, uint8_t coderate,
                         uint32_t bandwidthAfc, uint16_t preambleLen,
                         uint16_t symbTimeout, bool fixLen,
                         uint8_t payloadLen,
                         bool crcOn, bool freqHopOn, uint8_t hopPeriod,
                         bool iqInverted, bool rxContinuous )
{
    uint8_t syncWordBuf[]={ 0xC1, 0x94, 0xC1, 0x00, 0x00, 0x00, 0x00, 0x00 };

    RxContinuous = rxContinuous;
    if( rxContinuous == true )
    {
        symbTimeout = 0;
    }
    if( fixLen == true )
    {
        MaxPayloadLength = payloadLen;
    }
    else
    {
        MaxPayloadLength = 0xFF;
    }

    switch( modem )
    {
        case MODEM_FSK:
            LLCC68SetStopRxTimerOnPreambleDetect( false );
            LLCC68.ModulationParams.PacketType = PACKET_TYPE_GFSK;

            LLCC68.ModulationParams.Params.Gfsk.BitRate = datarate;
            LLCC68.ModulationParams.Params.Gfsk.ModulationShaping = MOD_SHAPING_G_BT_1;
            LLCC68.ModulationParams.Params.Gfsk.Bandwidth = RadioGetFskBandwidthRegValue( bandwidth << 1 ); // LLCC68 badwidth is double sided

            LLCC68.PacketParams.PacketType = PACKET_TYPE_GFSK;
            LLCC68.PacketParams.Params.Gfsk.PreambleLength = ( preambleLen << 3 ); // convert byte into bit
```

OCP_Value = Radio.Read(REG_OCP) reads the current maximum value of the over-current protection setting;

The Radio.SetRxConfig(MODEM_LORA, LORA_BANDWIDTH, LORA_SPREADING_FACTOR, LORA_CODINGRATE, 0, LORA_PREAMBLE_LENGTH, LORA_LLCC68_SYMBOL_TIMEOUT, LORA_FIX_LENGTH_PAYLOAD_ON, 0, true, 0, 0, LORA_IQ_INVERSION_ON, false) function is used to set the RX mode parameters of the Ra-01SCH-P module.

Function parameters:

LoRa mode, bandwidth, spreading factor, coding rate, automatic frequency bandwidth control, preamble length, symbol timeout time (maximum time the receiver waits for the next symbol to arrive), whether packet length is fixed, payload length, whether CRC check is enabled, whether frequency hopping is enabled, frequency hopping period (requires frequency hopping to be enabled, otherwise invalid), whether to invert I/Q components, whether to enable continuous reception;

Radio.Rx(0) enters receive mode. The function implementation is shown below:

```
void RadioRx( uint32_t timeout )
{
    LLCC68SetDioIrqParams( IRQ_RADIO_ALL, //IRQ_RX_DONE | IRQ_RX_TX_TIMEOUT,
                           IRQ_RADIO_ALL, //IRQ_RX_DONE | IRQ_RX_TX_TIMEOUT,
                           IRQ_RADIO_NONE,
                           IRQ_RADIO_NONE );

    if( timeout != 0 )
    {
        LLCC68SetRxTimerValue(timeout);
        LLCC68RxTimerStart();
    }

    if( RxContinuous == true )
    {
        LLCC68SetRx( 0xFFFFFF ); // Rx Continuous
    }
    else
    {
        LLCC68SetRx( RxTimeout << 6 );
    }
}
```

Radio.IrqProcess( ) is the event processing function that determines whether events have occurred. It can judge events: TX_DONE, RX_DONE, CRC_ERROR, CAD_DONE, RX_TX_TIMEOUT, PREAMBLE_DETECTED, SYNCWORD_VALID, HEADER_VALID, HEADER_ERROR. The function implementation is shown below:

```
}
void RadioIrqProcess( void )
{
    uint16_t irqRegs = 0;
    if( IrqFired == true )
    {
        CRITICAL_SECTION_BEGIN( );
        // Clear IRQ flag
        IrqFired = false;
        CRITICAL_SECTION_END( );

        irqRegs = LLCC68GetIrqStatus( );
        LLCC68ClearIrqStatus( irqRegs );

        if( ( irqRegs & IRQ_TX_DONE ) == IRQ_TX_DONE )
        {
            LLCC68TxTimerStop();
            //!< Update operating mode state to a value lower than \ref MODE_STDBY_XOSC
            LLCC68SetOperatingMode( MODE_STDBY_RC );
            if( ( RadioEvents != NULL ) && ( RadioEvents->TxDone != NULL ) )
            {
                RadioEvents->TxDone( );
            }
        }

        if( ( irqRegs & IRQ_RX_DONE ) == IRQ_RX_DONE )
        {
            if( ( irqRegs & IRQ_CRC_ERROR ) == IRQ_CRC_ERROR )
            {
                if( RxContinuous == false )
                {
                    //!< Update operating mode state to a value lower than \ref MODE_STDBY_XOSC
                    LLCC68SetOperatingMode( MODE_STDBY_RC );
                }
                if( ( RadioEvents != NULL ) && ( RadioEvents->RxError ) )
                {
```

delay_ms(1) delays 1ms to check once whether an event has occurred;

2) ExampleLLCC68SendDemo() Function

The ExampleLLCC68SendDemo() function is a timed transmission function.

The function implementation is shown below:

```
void ExampleLLCC68SendDemo(void){
  uint32_t u32_count=0;
  uint8_t OCP_Value = 0;

  printf("start %s() example\r\n",__func__);

  LLCC68RadioEvents.TxDone = LLCC68OnTxDone;
  LLCC68RadioEvents.RxDone = LLCC68OnRxDone;
  LLCC68RadioEvents.TxTimeout = LLCC68OnTxTimeout;
  LLCC68RadioEvents.RxTimeout = LLCC68OnRxTimeout;
  LLCC68RadioEvents.RxError = LLCC68OnRxError;

  Radio.Init( &LLCC68RadioEvents );
  Radio.SetChannel(LORA_FRE);
  Radio.SetTxConfig( MODEM_LORA, LORA_TX_OUTPUT_POWER, 0, LORA_BANDWIDTH,
                     LORA_SPREADING_FACTOR, LORA_CODINGRATE,
                     LORA_PREAMBLE_LENGTH, LORA_FIX_LENGTH_PAYLOAD_ON,
                     true, 0, 0, LORA_IQ_INVERSION_ON, 3000 );//参数：lora模式,发射功率,fsk用的lora设置为0就可以，带宽，纠错编码率，前导码长度，固定长度数据包(一般是不固定的所以选false)，crc校验，

  OCP_Value = Radio.Read(REG_OCP);
  printf("[%s()-%d]read OCP register value:0x%04X\r\n",__func__,__LINE__,OCP_Value);

  Radio.SetRxConfig( MODEM_LORA, LORA_BANDWIDTH, LORA_SPREADING_FACTOR,
                     LORA_CODINGRATE, 0, LORA_PREAMBLE_LENGTH,
                     LORA_LLCC68_SYMBOL_TIMEOUT, LORA_FIX_LENGTH_PAYLOAD_ON,
                     0, true, 0, 0, LORA_IQ_INVERSION_ON, false );

  printf("all setting\r\n");
  printf("freq: %d\r\n Tx power: %d\r\n band width: %d\r\n FS: %d\r\n CODINGRATE: %d\r\n PREAMBLE_LENGTH: %d\r\n",LORA_FRE,LORA_TX_OUTPUT_POWER,LORA_BANDWIDTH,LORA_SPREADING_FACTOR,LORA_CODINGRATE
  while(1){
    Radio.IrqProcess( ); // Process Radio IRQ

    if(0==u32_count%1000){
      printf("systick=%d ,send u32 data:%d\r\n", Get_SysTick(),u32_count);
      if(0==u32_count%2000){
        Radio.Send((uint8_t *)&u32_count,4);
      }else
        Radio.Send((uint8_t *)sendData,(strlen(sendData)+1));
    }
    u32_count++;
    delay_ms(1);
  }
}
```

The difference from the ExampleLLCC68ReciveDemo() function is inside the while() loop. Get_Systicket() is used to get the current program running time (ms).

```
if(0==u32_count%1000){
    printf("systick=%d ,send u32 data:%d\r\n", Get_SysTick(),u32_count);
    if(0==u32_count%2000){
```

```
        Radio.Send((uint8_t *)&u32_count,4);

    }else{

        Radio.Send((uint8_t *)sendData,(strlen(sendData)+1));

    }}

u32_count++;delay_ms(1);
```

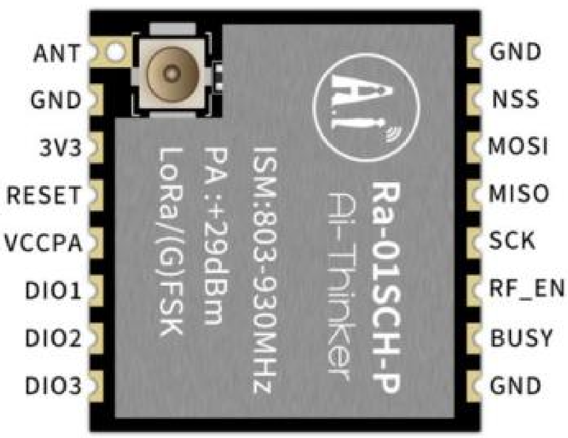
The above code sends data every 2 seconds, sending a count value every 2 seconds. The send function implementation is shown below:

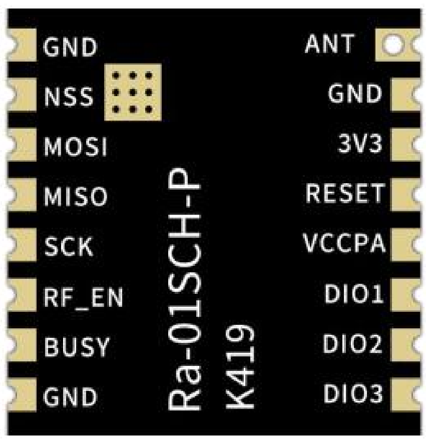
```
void RadioSend( uint8_t *buffer, uint8_t size )
{
    LLCC68SetDioIrqParams( IRQ_TX_DONE | IRQ_RX_TX_TIMEOUT,
                           IRQ_TX_DONE | IRQ_RX_TX_TIMEOUT,
                           IRQ_RADIO_NONE,
                           IRQ_RADIO_NONE );

    if( LLCC68GetPacketType( ) == PACKET_TYPE_LORA )
    {
        LLCC68.PacketParams.Params.LoRa.PayloadLength = size;
    }
    else
    {
        LLCC68.PacketParams.Params.Gfsk.PayloadLength = size;
    }
    LLCC68SetPacketParams( &LLCC68.PacketParams );

    LLCC68SendPayload( buffer, size, 0 );
    LLCC68SetTxTimerValue(TxTimeout);
    LLCC68TxTimerStart();
}
```

2. Hardware Introduction





No.	Name	Function
1	ANT	Connect antenna
2	GND	Ground
3	3V3	Typical value 3.3V power supply
4	RESET	Reset pin
5	VCCPA	FEM chip power supply pin, input 5V maximum power can reach +31dBm.
6	DIO1	Digital IO1 software configuration
7	DIO2	Digital IO2 software configuration
8	DIO3	Digital IO3 software configuration
9	GND	Ground
10	BUSY	Status indication pin
11	RF_EN	FEM chip enable pin, high level is effective, the module is pulled up by default; high level is in working state, low level is in sleep state
12	SCK	SPI clock input
13	MISO	SPI data output
14	MOSI	SPI data input
15	NSS	SPI chip select input
16	GND	Ground
EPA	GND	Ground, reliable grounding is required to facilitate heat dissipation

The general IO pins of LLCC68+ are all available in LoRa™ mode. Their mapping relationship depends on the configuration of the RegDioMapping1 and RegDioMapping2 registers.

The IO pin function mapping table is shown below:

Operation Mode	DIOx Mapping	DIO3	DIO2	DIO1
All	00	CadDone	Fhss Change Channel	RxRimeout
	01	Valid Header	Fhss Change Channel	Fhss Change Channel
	10	PayloadCrc Error	Fhss Change Channel	CadDetected
	11	-	-	-

1) Special Pin Description

About VCCPA Pin

The internal PA of the module supports multiple supply voltages. With 3.3V supply, the module's maximum transmit power can reach +29dBm; with 5V supply, the module's maximum transmit power can reach +31dBm;

The module's default BOM configuration uses the module's 3V3 pin for PA power supply. Simply leave the module's VCCPA pin floating;

If the PA uses 5V power supply, please contact Ai-Thinker to modify the module BOM and provide 5V power supply to the module's VCCPA pin;

About RF_EN Pin

RF_EN is the enable pin for the module's built-in PA chip. When this pin is high, the module's RF is in normal transceiver state; when this pin is low, the module's RF function is turned off, which can reduce the module's power consumption.

The module's default BOM has an internal 10K pull-up resistor (i.e., default is normal transceiver state). For low-power application scenarios, please use an external MCU to control this pin to low level. In low-level state, the pin's default pull-up resistor may have leakage current. If the internal pull-up resistor is not needed, please contact Ai-Thinker to modify the BOM.

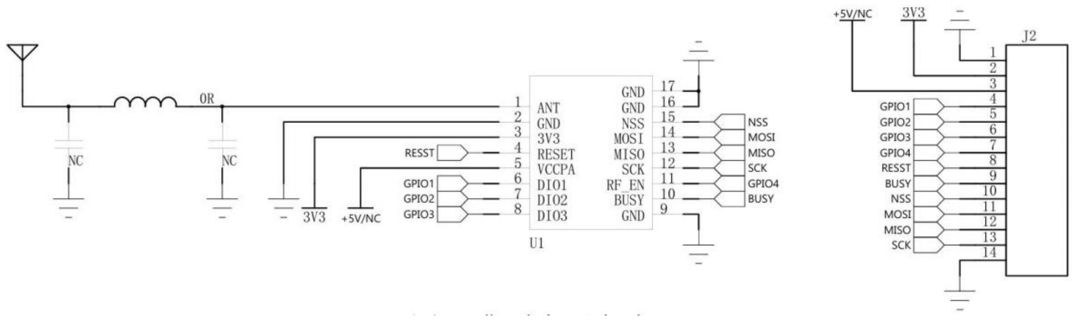
Table 6 RF switch truth table

Mode	RF_EN
FEM power off	0
FEM working	1

In summary, the module has four BOM configurations:

- Configuration 1: VCCPA floating, RF_EN with internal 1M pull-up resistor (default BOM configuration)
- Configuration 2: VCCPA floating, RF_EN without internal pull-up resistor
- Configuration 3: VCCPA connected, RF_EN with internal 1M pull-up resistor
- Configuration 4: VCCPA connected, RF_EN without internal pull-up resistor 1M

2) Typical Application Circuit



The module's VCCPA is floating by default. If you need 5V power supply to achieve higher transmit power, please contact Ai-Thinker to modify the BOM;

It is recommended that the external MCU's IO pin control the module's RF_EN to achieve low-power application scenarios;

3) Other Notes

In addition to the SPI interface for communication with the main control MCU, BUSY/DIO1 should also be connected to the main control MCU's IO pins;

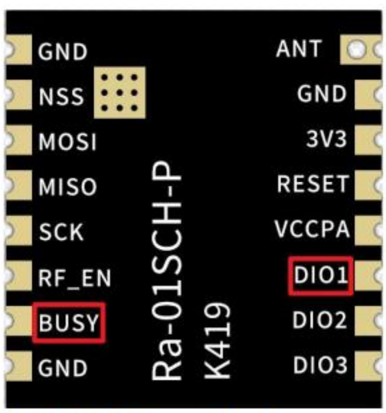


Figure 8 BUSY/DIO1 application considerations

When the antenna is soldered on the main control board, it is recommended to reserve a pi-type matching circuit at the antenna interface;

3. Antenna Installation

The Ra-01SCH-P needs to be used with an external antenna. The module has half-hole pads that can be routed to the main board;

For optimal antenna performance, the antenna installation position should be kept away from metal parts;

The antenna installation structure has a significant impact on module performance. Ensure the antenna is exposed, preferably vertically upward. When the module is installed inside a housing, a high-quality antenna extension cable can be used to extend the antenna outside the housing;

The antenna must not be installed inside a metal shell, as this will greatly weaken the transmission distance;

4. Power Supply

Recommended 3.3V voltage with peak current above 1A;

If using DC-DC, it is recommended to keep ripple within 100mV;

DC-DC power supply circuit is recommended to reserve positions for dynamic response capacitors to optimize output ripple when load changes significantly;

It is recommended to add ESD devices at the 3.3V power interface;

When designing the power supply circuit for the module, it is recommended to retain more than 30% current margin, which is beneficial for long-term stable operation of the whole device;

Please pay attention to correct connection of power positive and negative poles. Reverse connection may cause permanent damage to the module;

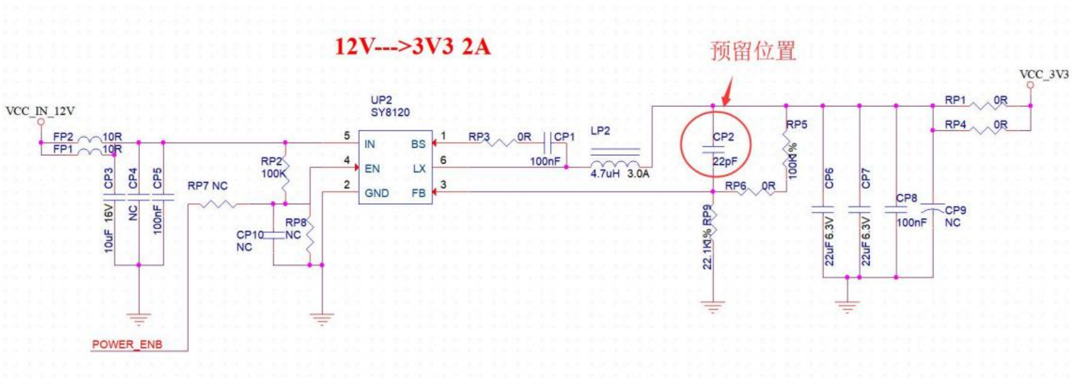


图 10 DC-DC 降压电路

5. Software Notes

The FEM chip's maximum input power must not exceed +15dBm, otherwise the FEM chip may be damaged. Users must strictly configure the LLCC68's output power, with 0dBm-3dBm recommended;

This module is LLCC68+peripheral circuits. Users can operate it completely according to the LLCC68 chip manual;

DIO1/DIO2 are general-purpose IO pins that can be configured into multiple functions;

The RF switch TX/RX control can be controlled by external MCU, or jointly controlled by external MCU and LLCC68's DIO2;

Differences between LLCC68 and SX1262/SX1268:

- SX1262/SX1268 support spreading factors SF5, SF6, SF7, SF8, SF9, SF10, SF11, SF12; SX1262/SX1268; Configurable spreading factors and receive bandwidth; LoRa@ Rx/Tx, BW = 7.8 - 500 kHz; SF5 TO SF12, BR=0.018 - 62.5 Kb/S;
- LLCC68 supports spreading factors SF5, SF6, SF7, SF8, SF9, SF10, SF11; LLCC68 configurable spreading factors and receive bandwidth; LoRa@ Rx/Tx, BW = 125 - 250 - 500 kHz; LoRa@, SF=5-6-7-8-9 for BW=125kHz; LoRa@, SF=5-6-7-8-9-10 for BW =250 kHz; LoRa@, SF=5-6-7-8-9-10-11 for BW=500 kHz;

III. Usage Introduction

1. Preparation

Two Ra-01SCH-P modules;

Multiple DuPont wires;

Two USB-to-TTL tools;

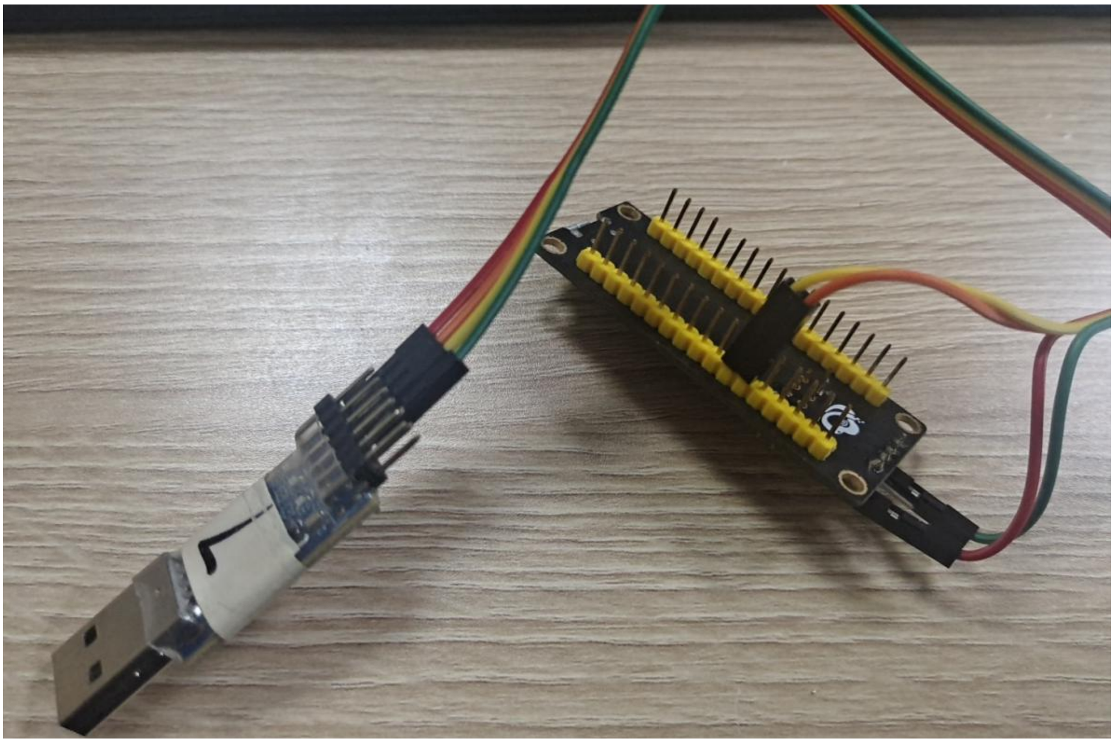
Ra-01SCH-P Demo;

ST-Link programming tool;

STM32F103C8T6 development board * 2;

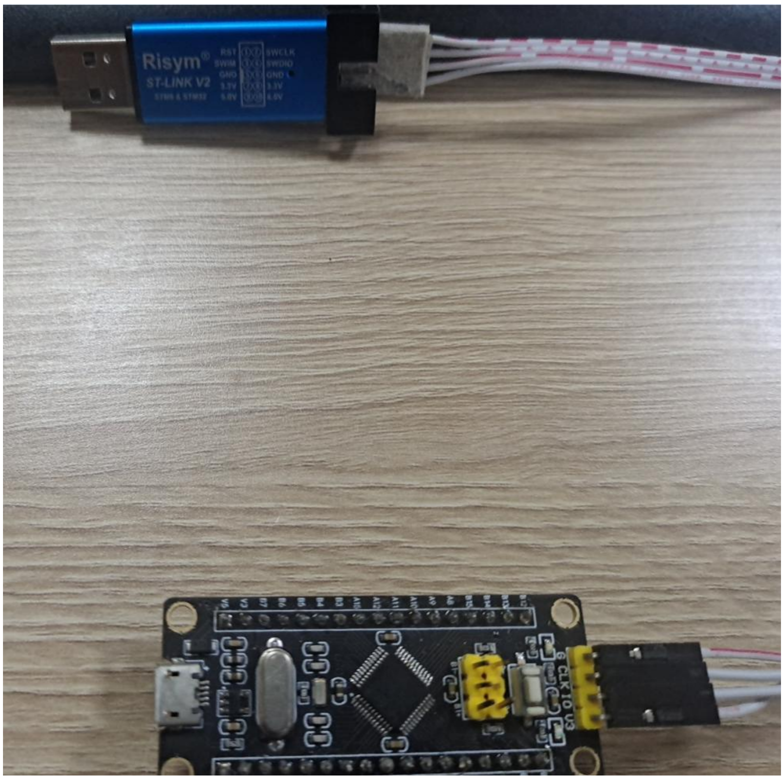
The wiring between USB-to-TTL tool and STM32 development board is shown below:

3V3	3V3
GND	GND
RX	PA9
TX	PA10



The programming wiring is shown below:

3V3	3V3
GND	GND
SWCLK	CLK
SWIO	IO

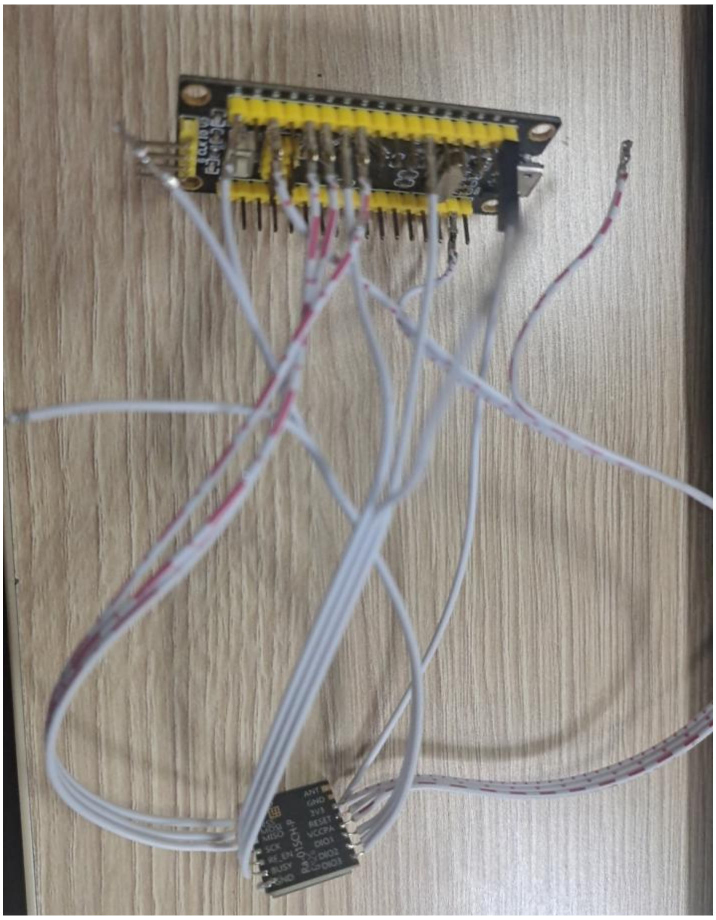


The wiring between Ra-01SCH-P module and STM32 development board is shown below:

STM32F103	RA-01SCH-P
3V3	3V3
GND	GND
PA9(TX)	
PA10(RX)	
PA4	NSS
PA7	MOSI
PA6	MISO
PA5	SCK
PB1	RST
PB11	DIO1
PA0	BUSY

Note: If higher transmit power is needed, use 5V to power the PA by providing 5V voltage to the VCCPA pin; (The module uses internal 3.3V for PA power supply by default)

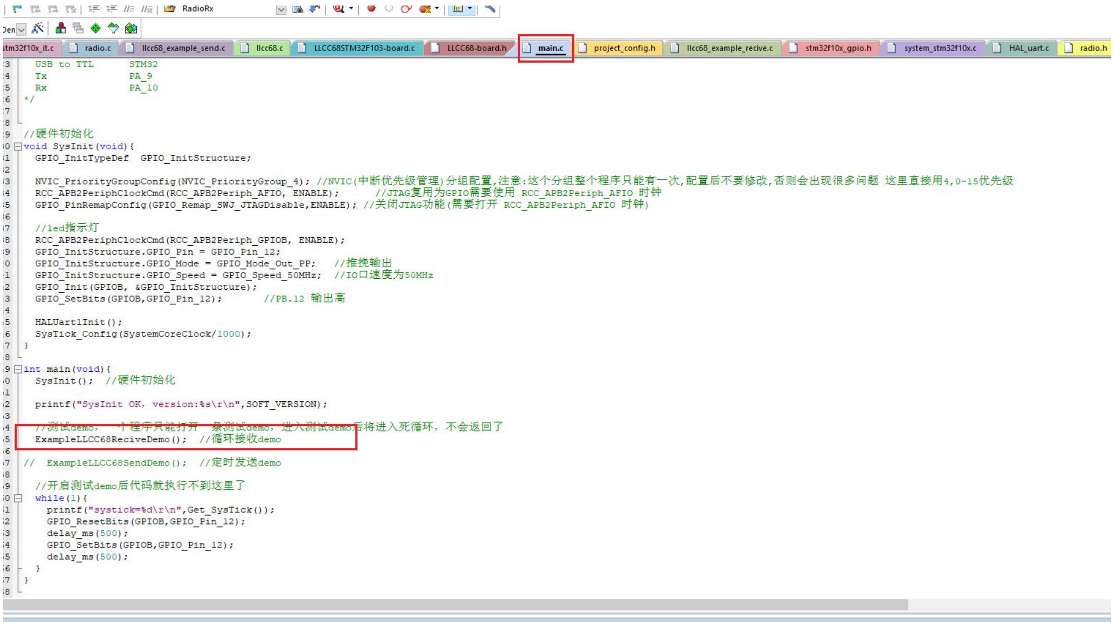
The wiring is shown below:



2. Program Flashing

1) Flash Receiver Program

The main function is shown below:



Then connect ST-Link to the STM32 development board, compile and flash;

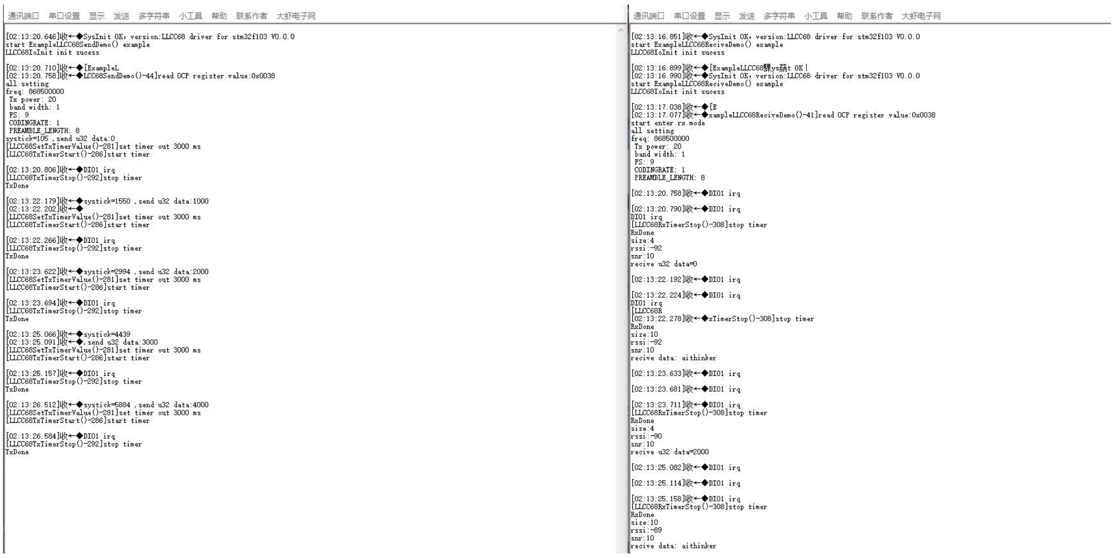
2) Flash Transmitter Program

The main function is shown below:

Then connect ST-Link to the STM32 development board, compile and flash;

IV. Demonstration

1. Serial Port Debugger Communication Log Information



Left side is the module in transmit mode, right side is the module in receive mode;

2. Module Communication LED Indicator During Operation

Video: [LED Demonstration]