基于 SX127x 的 LoRa 模组认证操作说明文档

一、文档概述

1.1 适用范围

本文档适用于 c 的 LoRa 模组做认证的操作，需要通过 单片机（如 STM32、ESP32 等）经 SPI 接口驱动 LoRa 模组，实现射频参数配置，以完成认证测试的场景。文档详细说明认证设备的准备，包括：模组硬件连接、认证固件功能、参数配置参考协议及认证流程。设备需支持调制方式切换、频率配置、功率配置以及带宽配置等核心功能。配置操作可通过串口 / USB 接口实现。本文档将以串口配置方式为例说明操作过程。

支持本文的内容适用于以下模组：

Ra-01/Ra-01H/Ra-02

二、硬件连接配置

2.1 SPI 接口接线（MCU 与模组）

LoRa 模组采用 SPI 接口与 MCU 通信，引脚接线如下：

模组引脚	功能描述	MCU 对应引脚	备注
VCC	电源输入	3.3V 输出	禁止接 5V，需稳压供电
GND	地	GND	必须共地，保证信号稳定
SCK	SPI 时钟线	SPI_SCK	时钟信号同步
MISO	SPI 主机输入	SPI_MISO	模组→MCU 数据传输
MOSI	SPI 主机输出	SPI_MOSI	MCU→模组数据传输
NSS	片选信号	通用 GPIO（如 PA4）	低电平有效，选中模组
DIO0	中断信号提示引脚	通用 GPIO（如 PA5）	无中断信号低电平，有中断信号高电平。
RST	复位信号	通用 GPIO（如 PA6）	低电平复位，高电平正常工作

2.2 配置接口接线（串口）

参数配置方式，以串口实现参数配置为例，接线规范如下：

MCU 引脚	串口模块引脚	配置要求
RXD	TXD	模组接收端接串口发送端
TXD	RXD	模组发送端接串口接收端
GND	GND	共地，避免电平漂移

三、认证功能固件说明

3.1 固件核心功能

认证专用固件支持三大核心配置功能，均通过 SPI 驱动实现模组参数写入，具体如下：

1. **调制方式切换：**支持 FSK 与 LoRa 模式切换
2. **频率配置：**默认频率以实际应用频率为准（部分地区认证有具体的频率要求，如 CE 要求测试 869.525MHz, 868.3MHz 以及 868.5MHz）
3. **功率配置**：支持 - 9dBm~+22dBm 可调
4. **带宽配置**：支持 0~2 可调（0: 125 kHz,1: 250 kHz,2: 500 kHz）

3.2 驱动配置流程（以安信可官网参考驱动 (sx127xled 点对点控制 demo) 为例）

单载波信号配置流程如下：

```
void OnTxDone( void )
{
    RadioSleepPending = true;
    State = TX;
}

void OnRxDone( uint8_t *payload, uint16_t size, int16_t rssi, int8_t snr , int16_t rssiAvr)
{
    int rssiValue = 0;
    RadioSleepPending = true;
    BufferSize = size;
    memcpy( Buffer, payload, BufferSize );
    RssiValue = rssi;
    SnrValue = snr;
    State = RX;
}

void OnTxTimeout( void )
{
    RadioSleepPending = true;
    State = TX_TIMEOUT;
}

void OnRxTimeout( void )
{
    RadioSleepPending = true;
    State = RX_TIMEOUT;
}
```

```
void OnRxError( void )
{
    RadioSleepPending = true;
    State = RX_ERROR;
}
main(void){
    RadioEvents_t RadioEvents;
    uint32_t freq = 868300000;
    int8_t power = 22;
    RadioEvents.TxDone = OnTxDone;
    RadioEvents.RxDone = OnRxDone;
    RadioEvents.TxTimeout = OnTxTimeout;
    RadioEvents.RxTimeout = OnRxTimeout;
    RadioEvents.RxError = OnRxError;
    Radio.Init(&RadioEvents);
    Radio.SetChannel(freq );
    Radio.SetTxConfig( MODEM_FSK, power , FSK_FDEV, 0,
                       FSK_DATARATE, 0,
                       FSK_PREAMBLE_LENGTH, FSK_FIX_LENGTH_PAYLOAD_ON,
                       true, 0, 0, 0, 3000 );
    SX1276Write( REG_FEILSB, ( SX1276Read( REG_FEILSB ) | (1<<3) ) );
    Radio.Send( Buffer, BufferSize );
    while(1){
    }
}
```

调制波信号配置流程如下：

```
void OnTxDone( void )
{
    RadioSleepPending = true;
    State = TX;
}

void OnRxDone( uint8_t *payload, uint16_t size, int16_t rssi, int8_t snr , int16_t rssiAvr)
{
    int rssiValue = 0;
    RadioSleepPending = true;
    BufferSize = size;
    memcpy( Buffer, payload, BufferSize );
    RssiValue = rssi;
    SnrValue = snr;
    State = RX;
}

void OnTxTimeout( void )
{
    RadioSleepPending = true;
    State = TX_TIMEOUT;
}

void OnRxTimeout( void )
{
    RadioSleepPending = true;
    State = RX_TIMEOUT;
}

void OnRxError( void )
```

```
{
    RadioSleepPending = true;
    State = RX_ERROR;
}
main(void){
    RadioEvents_t RadioEvents;
    uint32_t freq = 868300000;
    int8_t power = 22;
    RadioEvents.TxDone = OnTxDone;
    RadioEvents.RxDone = OnRxDone;
    RadioEvents.TxTimeout = OnTxTimeout;
    RadioEvents.RxTimeout = OnRxTimeout;
    RadioEvents.RxError = OnRxError;
    Radio.Init(&RadioEvents);
    Radio.SetChannel(freq );
    Radio.SetTxConfig( MODEM_LORA, 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 ); Radio.Send("\0",1);
    Radio.Send( Buffer, BufferSize );
    while(1){
        switch( State ){
            case TX:
                myPrintf(LEVEL_DEBUG,"TX_Done\r\n");
                Radio.Sleep( );
                delay_ms( 1000 );
                Radio.Send( Buffer, BufferSize );
                State = LOWPOWER;
                break;
            case TX_TIMEOUT:
                myPrintf(LEVEL_DEBUG,"TX_TIMEOUT\r\n");
```

```
                    delay_ms( 500 );
                    Radio.Send( Buffer, BufferSize );
                    State = LOWPOWER;
                    break;
                case RX:
                    myPrintf(LEVEL_DEBUG,"RX\r\n");
                    Radio.Rx( RX_TIMEOUT_VALUE );
                    State = LOWPOWER;
                    delay_ms( 10 );
                    break;
                case RX_TIMEOUT:
                    myPrintf(LEVEL_DEBUG,"RX_TIMEOUT\r\n");
                    Radio.Rx( RX_TIMEOUT_VALUE );
                    State = LOWPOWER;
                    delay_ms( 10 );
                    break;
                case RX_ERROR:
                    myPrintf(LEVEL_DEBUG,"RX_ERROR\r\n");
                    Radio.Rx( RX_TIMEOUT_VALUE );
                    State = LOWPOWER;
                    delay_ms( 10 );
                    break;
                default:
                    delay_ms( 1 );
                    break;
            }
        }
}
```

四、参数配置协议规范

4.1 参数配置指令格式

参数配置指令用户可自行设计，以下是配置参考示例。

4.2 具体配置指令

4.2.1 调制方式切换指令

- 指令格式：AT+MODE=<mode>
- 参数数据 (mode)：0 表示 FSK 模式；1 表示 LoRa 模式；
- 示例（切换为 LoRa 调制）：

帧数据：AT+MODE=1

4.2.2 频率、功率以及带宽配置指令

- 指令格式：AT+LORACONFIG=<freq>,<pwr>,<bw>
- 参数数据 说明：
  - freq（频率）：范围 150MHz ~ 960MHz
  - pwr：-9~22dBm
  - bw：0: 125 kHz,1: 250 kHz,2: 500 kHz
- 示例（配置 868.3MHz，功率 22dBm，带宽 125kHz）：

帧数据：AT+LORACONFIG=868300000,22,0

4.3 响应帧格式

模组接收指令后，会返回响应帧确认配置结果，响应帧格式为：

运行成功回复 OK，失败则回复 ERROR

五、常见问题排查

5.1 SPI 驱动失败

- 问题现象：MCU 正确驱动 LoRa 模组
- 排查步骤：
  a. 检查 VCC 电压是否稳定在 3.3V，纹波是否超过 50mV
  b. 确认 SPI 引脚接线正确，NSS 引脚是否拉低选中模组
  c. 排查 MCU SPI 时序是否符合要求（详情见芯片规格书，8.2.1 SPI Timing When the Transceiver is in Active Mode 章节内容）

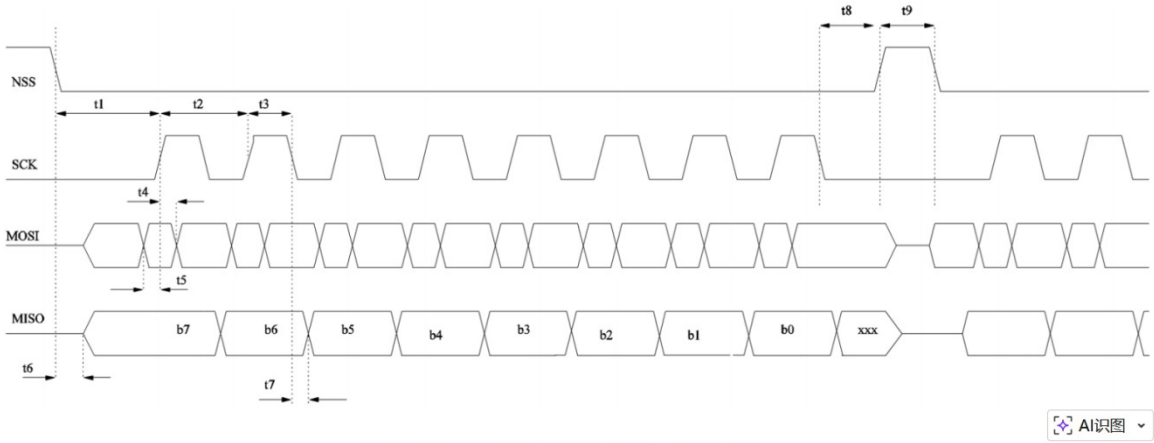

Figure 8-1: SPI Timing Diagram

  d. 检查模组 RST 引脚是否为高电平（正常工作状态）
  e. DIO1 中断识别功能是否有效
  f. BUSY 识别功能是否正确

5.2 串口配置异常（用户自行排查串口配置问题）

5.3 功率 / 频率配置异常

- 问题现象：配置后测试值与设定值偏差过大
- 排查步骤：
  g. 检查指令参数是否正确（如频率转换是否有误）
  h. 验证模组供电电流是否满足要求（如 + 22dBm 需≥100mA）
  i. 检查天线是否匹配（辐射测试），射频头连接是否牢固（传导测试）
  j. 确认配置频率是否为指定频率（实验室定义或者用户自行定义）