AN1901B ATK-LORA-02 无线串口模块

使用说明

本应用文档（AN1901B）将教大家如何在 ALIENTEK MiniV3 STM32F103 开发板上使用 ATK-LORA-02 无线串口模块。

本文档分为如下几部分：

1， ATK-LORA-02 模块简介

2， 硬件连接

3， 软件实现

4， 验证

1、ATK-LORA-02 模块简介

ATK-LORA-02_V1.5(V1.5 是版本号，型号是 ATK-LORA-02，下面均以 ATK-LORA-02 表示该产品)是 ALIENTEK 推出的一款采用 SMD 封装，体积小、微功率、低功耗、高性能远距离 LORA 无线串口模块。模块设计是采用高效的 ISM 频段射频 SX1278 扩频芯片，模块的工作频率 410Mhz~441Mhz，以 1Mhz 频率为步进信道，共 32 个信道，可通过 AT 指令在线修改串口速率，发射功率，空中速率、工作模式等各种参数，并且支持固件升级功能。

ATK-LORA-02 模块具有：体积小、灵敏度高、支持低功耗省电，特点包括：

1、工业频段：433Mhz 免申请频段

2、多种功率等级（最大 20dBm，最大 100mW）

3、多种串口波特率、空中速率、工作模式

4、支持空中唤醒功能，低接收功耗

5、双 512 环形 FIFO

6、频率 410-441Mhz，提供 32 个信道

7、接收灵敏度达-136dBm，传输距离 3000 米

8、自动分包传输，保证数据包的完整性

模块电器参数如表 1.1 所示。

项目	说明
封装	SMD-15
模块尺寸	24*17mm
工作频段	410-441Mhz（共 32 个通道），1Mhz，出厂默认 433Mhz
调制方式	LoRa 扩频
通信距离	约 3000 米（测试条件：晴朗、空旷，最大功率 20dbm，空中速率 2.4Kbps，天线增益 3dbi）
发射功率	最大 20dBm（约 100mW），4 级可调（0-3），每一级增减约 3dBm
空中速率	6 级可调（0.3、1.2、2.4、4.8、9.6、19.2Kbps）
工作电压	3.3~5V
发射电流	118ma（20dbm 100mw 电压 5V）
接收电流	17ma（模式 0、模式 1），最低约 2.3uA（模式 2+2S 唤醒）
通信接口	UART 串口，8N1、8E1、8O1，从 1200-115200 共 8 种波特率（默认 9600、8N1）
发射长度	内部环形 FIFO 缓存 512 字节，内部自动分包发送。某些空速与波特率组合，可发送无限长度数据包。

接收长度	内部环形 FIFO 缓存 512 字节，内部自动分包发送。某些空速与波特率组合可发送无限长度数据包。
模块地址	可配置 65536 个地址（便于组网支持广播和定向传输）
接收灵敏度	-136dBm@0.3Kbps（接收灵敏度和串口波特率、延迟时间无关）
天线形式	引脚引出
工作温度	-40~+85℃
存储温度	-40~+125℃

表 1.1 ATK-LORA-02 无线串口模块电器参数

1.1 硬件简介

ATK-LORA-02 无线串口模块外观如图 1.1-1 正面图和图 1.1-2 背面图所示：

图 1.1-1 ATK-LORA-02 无线串口模块实物图正面

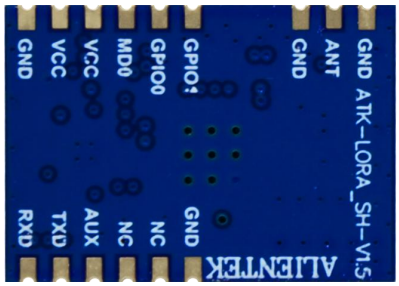

图 1.1-2 ATK-LORA-02 无线串口模块实物图背面

模块通过邮票孔与外部电路连接，各引脚的详细描述如表 1.1.2 所示：

引脚	名称	方向	说明
1、3、9、15	GND		地线
2	ANT		天线
4	GPIO1		未用
5	GPIO0		未用
6	MD0	输入	1、配置进入参数设置 2、上电时与 AUX 引脚配合进入固件升级模式
7、8	VCC		3.3V~5V 电源输入
10	RXD	输入	TTL 串口输入，连接到外部 TXD 输出引脚
11	TXD	输出	TTL 串口输出，连接到外部 RXD 输入引脚
12	AUX	1、输出 2、输入	1、用于指示模块工作状态，用户唤醒外部 MCU 2、上电时与 MD0 引脚配合进入固件升级模式
13、14	NC		未用

表 1.1.2 ATK-LORA-02 无线串口模块引脚说明

关于模块固件升级功能模式的详细说明，请看“**ATK-LORA-02 模块固件升级操作说明**

_V1.0.pdf”文档。

1.2 模块功能介绍

模块根据 MD0 的配置与 AUX 引脚的状态会进入不同的功能，如表 1.2.1 所示：

功能	介绍	进入方法
配置功能	模块参数配置（AT 指令）	上电后，AUX 空闲状态（AUX=0）,MD0=1
通信功能	无线通信	上电后，AUX 空闲状态（AUX=0）,MD0=0
固件升级功能	固件升级	上电后，AUX=1 且 MD0=1（持续 1 秒时间，电平不变）

表 1.2.1 功能介绍

其中通信功能下，包含 4 种工作模式，如表 1.2.2 所示：

模式（0-2）	介绍	备注
0 一般模式	无线透明、定向数据传输	接收方必须是模式 0、1
1 唤醒模式	和模式 0 唯一区别：数据包发射前，自动增加唤醒码，这样才能唤醒工作在模式 2 的接收方	接收方可以是模式 0、1、2
2 省电模式	串口接收关闭，无线处于空中唤醒模式，收到无线数据后打开串口发出数据	发射方必须是模式 1 该模式下串口接收关闭，不能无线发射
3 信号强度模式	查看通讯双方的信号强度	接收方必须是模式 0、1

表 1.2.2 工作模式

注意：工作模式需要模块进入配置功能发送 AT 指令才能切换。

1.3 快速了解

（1）**透明传输**：即透传数据，例如：A 设备发 5 字节数据 AA BB CC DD EE 到 B 设备，B 设备就可以收到数据 AA BB CC DD EE。（**透明传输，针对设备相同地址、相同的通信信道，用户数据可以是字符或 16 进制数据形式**）

（2）**定向传输**：即定点传输，例如：A 设备（地址为：0x1400，信道为 0x17（23 信道、433Mhz））需要向 B 设备（地址为：0x1234，信道为 0x10（16 信道、426Mhz））发送数据 AA BB CC，其通信格式为：12 34 10 AA BB CC，其中 1234 为模块 B 的地址，10 为信道，则模块 B 可以收到 AA BB CC。同理，如果 B 设备需要向 A 设备发送数据 AA BB CC，其通信格式为：14 00 17 AA BB CC，则 A 设备可以收到 AA BB CC。（**定向传输，针对设备间地址和通信信道不同，数据格式为 16 进制，发送格式：高位地址+低位地址+信道+用户数据**）

（3）**广播与数据监听**：将模块地址设置为 0xFFFF（即 65535），可以监听相同信道上的所有模块的数据传输，发送的数据，可以被相同信道上任意地址的模块收到，从而起到广播和监听的作用。

以上是模块的简单说明，关于模块通信功能及更多详细说明及应用，请查看 **ATK-LORA-02 无线串口模块用户手册_V1.0.pdf**。

2、硬件连接

2.1 功能介绍

本实验功能简介：本实验用于测试 ATK-LORA-02 无线串口模块，工作流程如下：

上电后，先检测模块是否存在（通过配置进入配置功能，发送 AT 指令），存在则进入主菜单，主菜单会显示模块需要配置的参数以及进入通信，通过开发板按键 KEY0、KEY1、KEY_UP 进行功能的操作。按下按键 KEY1、或 KEY0，上下箭头选择要配置的参数或者选择进入通信。若选中的是“配置参数”：当按下 KEY_UP 按键，会选中该项（下方显示下划线），这时通过按下 KEY1 或 KEY0 可以对参数进行配置，配置完后再次按下 KEY_UP 按键，可退出该项的选中（下滑线取消）。

若选择“进入通信”选项。按下 KEY_UP 按键，则会进入无线通信测试界面，模块会根据主菜单中用户设置的参数进行配置（在配置功能（串口波特率：115200，8 位数据位、1 位停止位、无校验位）发送 AT 指令，配置完后需重新切换回无线通信下串口配置），参数配置结束后，DS1 绿灯会闪烁，屏幕左上方会显示模块的当前配置：模块地址、通信信道、空中速率、发射功率、工作模式以及发送状态。屏幕右上方则提示，KEY_UP 返回主菜单、KEY1 发送数据。

若发送状态设置的是：“定向传输”，则会显示多一个选项：“KEY0 设置参数”。设置参数目的主要是设置发送目标设备的“地址”和“信道”。按下 KEY0 后，会显示输入框，提示用户输入目标地址，地址最大值为 65535，输入结束后按下“确定”，会提示继续输入目标信道，信道最大值为 31。再次按下“确定”则返回无线通信测试界面。按下 KEY1 则对目标设备发送数据，DS0 红灯会指示数据发送或接收的状态。

红灯亮：表示数据开始发送或数据开始接收。红灯灭：则表示数据发送完毕或者数据接收完成。发送和接收的数据会显示在屏幕上，同时接收到的数据会串口输出。按下 KEY_UP，则返回主菜单界面。

注意：

（1）进入配置功能，串口需设置：“波特率：115200，8 位数据位、1 位停止位、无校验位”，退出配置功能，返回无线通信需设置通信下的串口参数，以免无线通信下工作不正常。

（2）AUX 指示：模块输出数据给 MCU，AUX 引脚会有上升沿电平，提示数据开始输出，当 AUX 引脚下降沿电平，表示数据输出完毕。MCU 发送数据给模块，AUX 引脚上升沿电平表示数据开始发送，当 AUX 下降沿电平，则表示 MCU 发送的数据已发送完毕。

2.2 硬件准备资源

本实验所需要的硬件资源如下

1，ALIENTEK MiniV3 STM32 开发板 1 个
2，TFTLCD 模块
3，ATK-LORA-02 无线串口模块 1 个
4，USB 线一条（用于供电和模块与电脑串口调试助手通信）
5，SD 卡（若需要更新字库，则将模块资料下 SD 卡根目录文件里对应开发板的 SYSTEM 文件内容复制到 SD 卡中）

注意：模块在发射时，瞬间的工作电流会比较大，用 USB 线供电给板子，液晶屏有可能会出现闪屏现象。

2.3 模块与开发板连接

MiniV3 STM32 开发板与 ATK-LORA-02 模块的连接关系如表 2.3.1 所示：

ATK-LORA-02 无线串口模块与开发板连接关系						
ATK-LORA-02 模块	VCC	GND	TXD	RXD	AUX	MD0
Mini V3 STM32 开发板	5V	GND	PA3	PA2	PA4	PA11

表 2.3.1ATK-LORA-02 模块与 MiniV3 STM32 开发板连接关系图

3、软件实现

本实验在 MiniV3 开发板的 ATK-SIM800C 模块扩展实验基础上进行修改，在 HARDWARE 文件夹内新建了 LORA 文件夹，并在工程中添加 LORA 分组，新建 lora_ui.c、lora_ui.h、lora_app.c、lora_app.h、lora_cfg.h 共五个文件，存放在 LORA 文件夹内。将 lora_app.c、lora_ui.app、lora_cfg.h 加入 LORA 分组，并添加 LORA 文件夹到头文件包含路径。

最终的工程如图 3.1 所示：

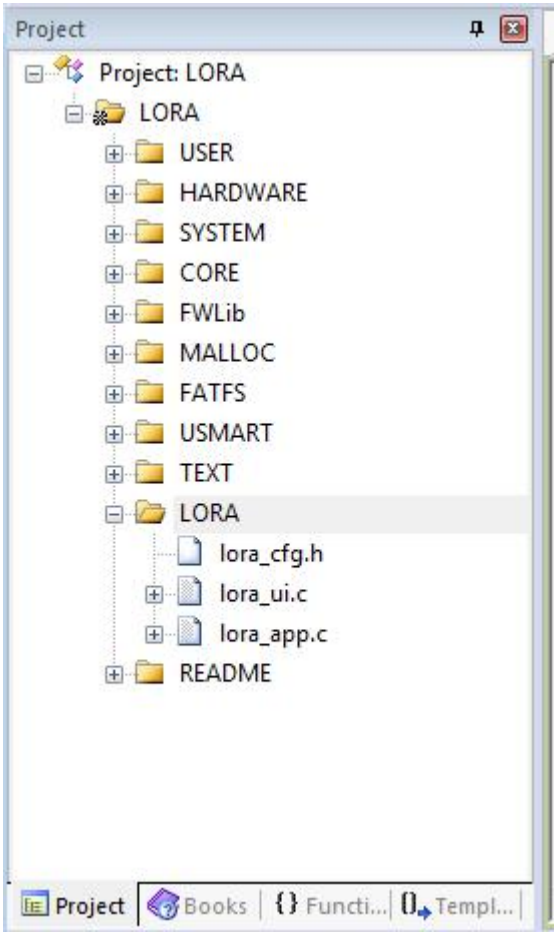

图 3.1 ATK-LORA-02 模块测试实验工程截图

本例程代码量不多，我们仅对部分重要代码讲解（lora_app.c、lora_ui.c、lora_cfg.h），以及 main 函数进行讲解。

Lora_ui.c 文件是液晶显示 UI 和底层驱动功能的一些函数，这里我们只列出部分代码，代码如下：

```
//lora 发送命令后,检测接收到的应答
//str:期待的应答结果
```

```
//返回值:0,没有得到期待的应答结果
//其他,期待应答结果的位置(str 的位置)
u8* lora_check_cmd(u8 *str)
{
	char *strx=0;
	if(USART2_RX_STA&0X8000)//接收到一次数据了
	{
		USART2_RX_BUF[USART2_RX_STA&0X7FFF]=0;//添加结束符
		strx=strstr((const char*)USART2_RX_BUF,(const char*)str);
	}
	return (u8*)strx;
}
//lora 发送命令
//cmd:发送的命令字符串(不需要添加回车了),当 cmd<0XFF 的时候,发送数字(比如发送
0X1A),大于的时候发送字符串.
//ack:期待的应答结果,如果为空,则表示不需要等待应答
//waittime:等待时间(单位:10ms)
//返回值:0,发送成功(得到了期待的应答结果)
//		1,发送失败
u8 lora_send_cmd(u8 *cmd,u8 *ack,u16 waittime)
{
	u8 res=0;
	USART2_RX_STA=0;
	if((u32)cmd<=0XFF)
	{
		while(DMA1_Channel7->CNDTR!=0);	//等待通道 7 传输完成
		USART2->DR=(u32)cmd;
	}else u2_printf("%s\r\n",cmd);//发送命令
	if(ack&&waittime)		//需要等待应答
	{
	    while(--waittime)	//等待倒计时
	    {
	        delay_ms(10);
	        if(USART2_RX_STA&0X8000)//接收到期待的应答结果
	        {
	            if(lora_check_cmd(ack))
	            {
	                USART2_RX_STA=0;
	                break;//得到有效数据
	            }
	        }
	    }
	    if(waittime==0)res=1;
	}
```

```
    return res;
}
```

u8* lora_check_cmd(u8 *str)是检测模块应答函数，该函数用于检测 ATK-LORA-01 模块发送回来的应答/数据，其中 str 为期待应答字符串，返回值如果是 0，则表示没有收到期待应答字符串，否则为期待应答字符串所在的位置。

u8 lora_send_cmd(u8 *cmd,u8 *ack,u16 waittime)，该函数用于向 ATK-LORA-01 模块发送命令。cmd 为命令字符串，当 cmd<=0XFF 的时候，则直接发送 cmd。Ack 为期待应答字符串，waittime 为等待时间（单位：10ms）。在 usart2.c 的串口接收函数中，模块在配置功能下，发送命令时，串口两个字符接收间隔以 10ms 为标准的，超过 10ms 为一次接收完成。（具体的请查看 usart2.c 中 TIM4_IRQHandler()定时器 4 的中断服务函数和 USART2_IRQHandler 串口 2 的中断服务函数）。

lora_ui.c 我们就介绍到这里，我们在来看看 lora_cfg.h 文件，lora_cfg.h 代码如下：

```
//设备参数定义
typedef struct
{
    u16 addr;//设备地址
    u8 chn;//信道
    u8 power;//发射功率
    u8 wlrate;//空中速率
    u8 wltime;//休眠时间
    u8 mode;//工作模式
    u8 mode_sta;//发送状态
    u8 bps;//串口波特率
    u8 parity;//校验位
}_LoRa_CFG;

//空中速率(单位:Kbps)
#define   LORA_RATE_0K3    0 //0.3
#define   LORA_RATE_1K2    1 //1.2
#define   LORA_RATE_2K4    2 //2.4
#define   LORA_RATE_4K8    3 //4.8
#define   LORA_RATE_9K6    4 //9.6
#define   LORA_RATE_19K2   5 //19.2

//休眠时间(单位:秒)
#define LORA_WLTIME_1S   0   //1 秒
#define LORA_WLTIME_2S   1   //2 秒

//工作模式
#define LORA_MODE_GEN     0      //一般模式
#define LORA_MODE_WK      1     //唤醒模式
#define LORA_MODE_SLEEP   2     //省电模式
```

```
//发射功率
#define LORA_PW_11dBm    0     //11dBm
#define LORA_PW_14Bbm    1     //14dBm
#define LORA_PW_17Bbm    2     //17dBm
#define LORA_PW_20Bbm    3     //20dBm

//发送状态
#define LORA_STA_Tran    0   //透明传输
#define LORA_STA_Dire    1   //定向传输

//串口波特率(单位:bps)
#define LORA_TTLBPS_1200       0   //1200
#define LORA_TTLBPS_2400       1   //2400
#define LORA_TTLBPS_4800       2   //4800
#define LORA_TTLBPS_9600       3   //9600
#define LORA_TTLBPS_19200      4   //19200
#define LORA_TTLBPS_38400      5   //38400
#define LORA_TTLBPS_57600      6   //57600
#define LORA_TTLBPS_115200     7   //115200

//串口数据校验
#define LORA_TTLPAR_8N1    0 //8 位数据
#define LORA_TTLPAR_8E1    1 //8 位数据+1 位偶校验
#define LORA_TTLPAR_8O1    2 //8 位数据+1 位奇校验

//设备出厂默认参数
#define LORA_ADDR          0                       //设备地址
#define LORA_CHN           23                      //通信信道
#define LORA_POWER         LORA_PW_20Bbm     //发射功率
#define LORA_RATE          LORA_RATE_19K2       //空中速率
#define LORA_WLTIME        LORA_WLTIME_1S       //休眠时间
#define LORA_MODE          LORA_MODE_GEN     //工作模式
#define LORA_STA           LORA_STA_Tran         //发送状态
#define LORA_TTLBPS        LORA_TTLBPS_9600    //波特率
#define LORA_TTLPAR        LORA_TTLPAR_8N1     //校验位
```

该文件主要是模块参数的一些宏定义。在下面可以看到列出的设备出厂默认参数,这些参数到时会在配置模块的时候用到。

接下来我们说下 lora_app.c 文件，该文件比较重要，包含模块配置参数和发送接收过程。我们先说下设备配置的参数结构体，代码如下：

```
//设备参数初始化(具体设备参数见 lora_cfg.h 定义)
_LoRa_CFG LoRa_CFG=
{
    .addr = LORA_ADDR,          //设备地址
    .power = LORA_POWER,     //发射功率
```

```
    .chn = LORA_CHN,            //信道
    .wlrate = LORA_RATE,        //空中速率
    .wltime = LORA_WLTIME,      //睡眠时间
    .mode = LORA_MODE,          //工作模式
    .mode_sta = LORA_STA,       //发送状态
    .bps = LORA_TTLBPS ,        //波特率设置
    .parity = LORA_TTLPAR       //校验位设置
};
```

该结构体成员参数赋予了设备出厂的默认参数，在初始化配置时和 UI 的显示会用到该结构体。接下来我们看下 Aux_Int 中断设置函数和 EXTI4_IRQHandler 中断服务函数，代码如下：

```
//全局参数
EXTI_InitTypeDef EXTI_InitStructure;
NVIC_InitTypeDef NVIC_InitStructure;

//设备工作模式(用于记录设备状态)
u8 Lora_mode=0;//0:配置模式  1:接收模式  2:发送模式
//记录中断状态
static u8 Int_mode=0;//0:关闭  1:上升沿  2:下降沿

//AUX 中断设置
//mode:配置的模式  0:关闭  1:上升沿  2:下降沿
void Aux_Int(u8 mode)
{
    if(!mode)
    {
        EXTI_InitStructure.EXTI_LineCmd = DISABLE;//关闭中断
        NVIC_InitStructure.NVIC_IRQChannelCmd = DISABLE;
    }else
    {
        if(mode==1)
            EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Rising; //上升沿
        else if(mode==2)
            EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;//下降沿

        EXTI_InitStructure.EXTI_LineCmd = ENABLE;
        NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    }
    Int_mode = mode;//记录中断模式
    EXTI_Init(&EXTI_InitStructure);
    NVIC_Init(&NVIC_InitStructure);
}
```

```
//LORA_AUX 中断服务函数
void EXTI4_IRQHandler(void)
{
    if(EXTI_GetITStatus(EXTI_Line4))
    {
        if(Int_mode==1)//上升沿(发送:开始发送数据  接收:数据开始输出)
        {
            if(Lora_mode==1)//接收模式
            {
                USART3_RX_STA=0;//数据计数清 0
            }
            Int_mode=2;//设置下降沿触发
            LED0=0;//DS0 亮
        }
        else if(Int_mode==2)//下降沿(发送:数据已发送完  接收:数据输出结束)
        {
            if(Lora_mode==1)//接收模式
            {
                USART3_RX_STA|=1<<15;//数据计数标记完成
            }else if(Lora_mode==2)//发送模式(串口数据发送完毕)
            {
                Lora_mode=1;//进入接收模式
            }
            Int_mode=1;//设置上升沿触发
            LED0=1;//DS0 灭
        }
        Aux_Int(Int_mode);//重新设置中断边沿
        EXTI_ClearITPendingBit(EXTI_Line4); //清除 LINE4 上的中断标志位
    }
}
```

我们先说下 lora_mode 和 Int_mode 变量，lora_mode 变量用于记录模块工作模式状态的情况（配置模式、接收模式、发送模式）。Int_mode 变量用于记录中断配置情况（关闭、上升沿、下降沿）。

Aux_int 中断设置函数，该函数用于配置 MCU 外部中断触发方式，有三种方式：关闭中断、上升沿中断、下降沿中断。它可用于获取 Aux 引脚的中断情况，若 MCU 要接收模块发送的数据，则可配置上升沿触发，以提示数据要来了，MCU 请做好准备。当 MCU 想知道模块的数据是否已全部发送给 MCU，则可配置下降沿中断，提示数据已经接收完了。在配置中断后，我们使用 Int_mode 变量来记录中断配置的情况。

接下来看下 EXTI4_IRQHandler()中断服务函数，在函数中可以看到，当是上升沿中断触发，先判断是否为接收模式，是则将串口接收数据计数清零，然后设置下降沿触发，同时点亮 LED0，以表示模块开始发送数据或开始输出数据。当下次下降沿中断来临，若是接收模式模式则标记串口接收数据已接收完成，若当前是发送模式状态的话，则知道数据已发送完毕，重新标记为接收模式，同时 LED0 灯灭，然后再调用 Aux_Int 函数配置下次触发的边沿中断，再继续实现一次发送或接收。

LoRa_Init()函数为模块的初始化函数，代码如下：

```
//LoRa 模块初始化
//返回值:0,检测成功
//       1,检测失败
u8 LoRa_Init(void)
{
    u8 retry=0;
    u8 temp=1;

    GPIO_InitTypeDef    GPIO_InitStructure;

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);  //使能 PA 端口时钟

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_11;              //LORA_MD0
    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(GPIOA, &GPIO_InitStructure);          //推挽输出 ，IO 口速度为 50MHz

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_4;               //LORA_AUX
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPD;         //下拉输入
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;   //IO 口速度为 50MHz
    GPIO_Init(GPIOA, &GPIO_InitStructure);             //根据设定参数初始化 GPIOA.4

    GPIO_EXTILineConfig(GPIO_PortSourceGPIOA,GPIO_PinSource4);
    EXTI_InitStructure.EXTI_Line=EXTI_Line4;
    EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
    EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Rising;    //上升沿触发
    EXTI_InitStructure.EXTI_LineCmd = DISABLE;                //中断线关闭
    EXTI_Init(&EXTI_InitStructure);
   //根据 EXTI_InitStruct 中指定的参数初始化外设 EXTI 寄存器

    NVIC_InitStructure.NVIC_IRQChannel = EXTI4_IRQn;         //LORA_AUX
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0x02;//抢占优先级 2，
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0x03;  //子优先级 3
    NVIC_InitStructure.NVIC_IRQChannelCmd = DISABLE;       //关闭外部中断通道
    NVIC_Init(&NVIC_InitStructure);

    LORA_MD0=0;
    LORA_AUX=0;

    while(LORA_AUX)//确保 LORA 模块在空闲状态下(LORA_AUX=0)
    {
        Show_Str(40+30,50+20,200,16,"模块正忙,请稍等!!",16,0);
        delay_ms(500);
```

```
            Show_Str(40+30,50+20,200,16,"                    ",16,0);
            delay_ms(100);
        }
        usart2_init(115200);                //初始化串口 2

        LORA_MD0=1;//进入 AT 模式
        delay_ms(40);
        retry=3;
        while(retry--)
        {
            if(!lora_send_cmd("AT","OK",70))
            {
                temp=0;//检测成功
                USART2_RX_STA=0;
                break;
            }
        }
        if(retry==0) temp=1;//检测失败
        return temp;
    }
```

函数中对 MD0 控制引脚和 AUX 状态引脚初始化，然后确保模块在空闲状态下，配置通信串口接口（波特率 115200、8 位数据、1 位停止、无校验），接着将模块配置进入配置功能 AT 指令模式，发送 AT 指令等待模块的应答，应答成功则模块初始化成功。

LoRa_Set()函数，该函数实现对模块参数进行配置，具体代码如下:

```
    //Lora 模块参数配置
    void LoRa_Set(void)
    {
        u8 sendbuf[20];
        u8 lora_addrh,lora_addrl=0;

        usart2_set(LORA_TTLBPS_115200,LORA_TTLPAR_8N1);
    //进入配置模式前设置通信波特率和校验位(115200 8 位数据  1 位停止  无数据校验）
        usart2_rx(1);//开启串口 2 接收

        while(LORA_AUX);//等待模块空闲
        LORA_MD0=1; //进入配置模式
        delay_ms(40);
        Lora_mode=0;//标记"配置模式"

        lora_addrh =    (LoRa_CFG.addr>>8)&0xff;
        lora_addrl = LoRa_CFG.addr&0xff;
        lora_send_cmd("AT","OK",70);
        sprintf((char*)sendbuf,"AT+ADDR=%02x,%02x",lora_addrh,lora_addrl);//设置设备地址
        lora_send_cmd(sendbuf,"OK",50);
```

```
        sprintf((char*)sendbuf,"AT+WLRATE=%d,%d",LoRa_CFG.chn,LoRa_CFG.wlrate);
        //设置信道和空中速率
        lora_send_cmd(sendbuf,"OK",50);
        sprintf((char*)sendbuf,"AT+TPOWER=%d",LoRa_CFG.power);//设置发射功率
        lora_send_cmd(sendbuf,"OK",50);
        sprintf((char*)sendbuf,"AT+CWMODE=%d",LoRa_CFG.mode);//设置工作模式
        lora_send_cmd(sendbuf,"OK",50);
        sprintf((char*)sendbuf,"AT+TMODE=%d",LoRa_CFG.mode_sta);//设置发送状态
        lora_send_cmd(sendbuf,"OK",50);
        sprintf((char*)sendbuf,"AT+WLTIME=%d",LoRa_CFG.wltime);//设置睡眠时间
        lora_send_cmd(sendbuf,"OK",50);
        sprintf((char*)sendbuf,"AT+UART=%d,%d",LoRa_CFG.bps,LoRa_CFG.parity);
        //设置串口波特率、数据校验位
        lora_send_cmd(sendbuf,"OK",50);

        LORA_MD0=0;//退出配置,进入通信
        delay_ms(40);
        while(LORA_AUX);//判断是否空闲(模块会重新配置参数)
        USART2_RX_STA=0;
        Lora_mode=1;//标记"接收模式"
        usart2_set(LoRa_CFG.bps,LoRa_CFG.parity);
        //返回通信,更新通信串口配置(波特率、数据校验位)
        Aux_Int(1);//设置 LORA_AUX 上升沿中断

    }
```

函数中，先对通信串口设置波特率 115200、无校验位，然后进入配置功能模式，发送 AT 指令配置设备地址、信道、空中速率等参数，配置完成后推出配置功能模式，用 Lora_mode 变量标记为接收模式，同时通信串口重新设置回通信下的设置，最后设置 AUX 中断为上升沿中断，等待数据接收或数据的发送。

LoRa_SendData()函数，该函数实现模块发送数据，具体代码如下：

```
u8 Dire_Date[]={0x11,0x22,0x33,0x44,0x55};//定向传输数据
u8 date[30]={0};//定向数组
u8 Tran_Data[30]={0};//透传数组

#define Dire_DateLen sizeof(Dire_Date)/sizeof(Dire_Date[0])
extern u32 obj_addr;//记录用户输入目标地址
extern u8 obj_chn;//记录用户输入目标信道

u8 wlcd_buff[10]={0}; //LCD 显示字符串缓冲区
//Lora 模块发送数据
void LoRa_SendData(void)
{
        static u8 num=0;
        u16 addr;
```

```
    u8 chn;
    u16 i=0;

  if(LoRa_CFG.mode_sta == LORA_STA_Tran)//透明传输
  {
      sprintf((char*)Tran_Data,"ATK-LORA-02 TEST %d",num);
      u2_printf("%s\r\n",Tran_Data);
      LCD_Fill(0,195,240,220,WHITE); //清除显示
      Show_Str_Mid(10,195,Tran_Data,16,240);//显示发送的数据

      num++;
      if(num==255) num=0;

  }else if(LoRa_CFG.mode_sta == LORA_STA_Dire)//定向传输
  {
      addr = (u16)obj_addr;//目标地址
      chn = obj_chn;//目标信道

      date[i++] =(addr>>8)&0xff;//高位地址
      date[i++] = addr&0xff;//低位地址
      date[i] = chn;    //无线信道
      for(i=0;i<Dire_DateLen;i++)//数据写到发送 BUFF
      {
          date[3+i] = Dire_Date[i];
      }
      for(i=0;i<(Dire_DateLen+3);i++)
      {
          while(USART_GetFlagStatus(USART2,USART_FLAG_TC)==RESET);
          //循环发送,直到发送完毕
          USART_SendData(USART2,date[i]);
      }
      //将十六进制的数据转化为字符串打印在 lcd_buff 数组
      sprintf((char*)wlcd_buff,"%x %x %x %x %x %x %x %x",
              date[0],date[1],date[2],date[3],date[4],date[5],date[6],date[7]);

      LCD_Fill(0,200,240,230,WHITE);//清除显示
      Show_Str_Mid(10,200,wlcd_buff,16,240);//显示发送的数据

      Dire_Date[4]++;//Dire_Date[4]数据更新
  }
}
```

当设备是“透明传输”或“定向传输”时，分别调用不同的数据发送，其中 obj_addr 和 obj_chn 为全局变量，是用户输入的目标地址和信道（具体的请看 lora_ui.c 文件下 Dire_Set 函数），这两个参数只在定向传输才用到。在定向传输，发送的数据前三个字节为“高位地

址、低位地址、目标信道”，然后后面才是用户真正的数据。

LoRa_ReceData()函数，该函数实现模块接收数据，具体代码如下：

```
u8 rlcd_buff[10]={0}; //LCD 显示字符串缓冲区
//Lora 模块接收数据
void LoRa_ReceData(void)
{
    u16 i=0;
    u16 len=0;

    //有数据来了
    if(USART2_RX_STA&0x8000)
    {
        len = USART2_RX_STA&0X7FFF;
        USART2_RX_BUF[len]=0;//添加结束符
        USART2_RX_STA=0;

        for(i=0;i<len;i++)
        {
            while(USART_GetFlagStatus(USART1,USART_FLAG_TC)==RESET);
            //循环发送,直到发送完毕
            USART_SendData(USART1,USART2_RX_BUF[i]);
        }
         LCD_Fill(10,260,240,320,WHITE);
        if(LoRa_CFG.mode_sta==LORA_STA_Tran)//透明传输
        {
            Show_Str_Mid(10,270,USART2_RX_BUF,16,240);//显示接收到的数据

        }else if(LoRa_CFG.mode_sta==LORA_STA_Dire)//定向传输
        {
            //将十六进制的数据转化为字符串打印在 lcd_buff 数组
            sprintf((char*)rlcd_buff,"%x %x %x %x %x",
            USART2_RX_BUF[0],USART2_RX_BUF[1],USART2_RX_BUF[2]
            USART2_RX_BUF[3],USART2_RX_BUF[4]);
            Show_Str_Mid(10,270,rlcd_buff,16,240);//显示接收到的数据
        }
        memset((char*)USART2_RX_BUF,0x00,len);//串口接收缓冲区清 0
    }
}
```

根据前面的了解，我们知道在数据接收完毕后会标记接收完成，然后我们就可以根据标记来读取数据，并且将数据显示在屏幕上。

LoRa_Process()函数，该函数实现发送和接收处理函数，具体代码如下:

```
void LoRa_Process(void)
{
    u8 key=0;
```

```
    u8 t=0;
DATA:
    Process_ui();//界面显示
    LoRa_Set();//LoRa 配置(进入配置需设置串口波特率为 115200)
    while(1)
    {
        key = KEY_Scan(0);
        if(key==KEY0_PRES)
        {
            if(LoRa_CFG.mode_sta==LORA_STA_Dire)
            //若是定向传输,则进入配置目标地址和信道界面
            {
                usart2_rx(0);//关闭串口接收
                Aux_Int(0);//关闭中断
                Dire_Set();//进入设置目标地址和信道
                goto DATA;
            }
        }else if(key==WKUP_PRES)//返回主菜单页面
        {
            LORA_MD0=1; //进入配置模式
            delay_ms(40);
            usart2_rx(0);//关闭串口接收
            Aux_Int(0);//关闭中断
            break;
        }
        else if(key==KEY1_PRES)//发送数据
        {
           if(!LORA_AUX&&(LoRa_CFG.mode!=LORA_MODE_SLEEP))
            //空闲且非省电模式
              {
                  Lora_mode=2;//标记"发送状态"
                  LoRa_SendData();//发送数据
              }
        }
        //数据接收
        LoRa_ReceData();

        t++;
        if(t==20)
        {
            t=0;
            LED1=~LED1;
        }
        delay_ms(10);
```

```
    }

}
```

该函数调用了刚刚我们提到的 LoRa_SendData()发送和 LoRa_ReceDate()接收函数，在发送数据时需判断 AUX 是否为空闲才能进行发送。我们在定向传输配置目标设备地址和信道或在返回主菜单参数页面前，会先把串口接收和 AUX 中给关闭，然后再进行下一步的操作。

Lora_Test()函数，该函数为模块的测试函数，会调用前面我们讲解的函数，具体代码如下：

```
//主测试函数
void Lora_Test(void)
{
    u8 t=0;
    u8 key=0;
    u8 netpro=0;

    LCD_Clear(WHITE);
    POINT_COLOR=RED;
    Show_Str_Mid(0,30,"ATK-LORA-02  测试程序",16,240);

    while(LoRa_Init())//初始化 ATK-LORA-02 模块
    {
        Show_Str(40+30,50+20,200,16,"未检测到模块!!!",16,0);
        delay_ms(300);
        Show_Str(40+30,50+20,200,16,"                    ",16,0);
    }
    Show_Str(40+30,50+20,200,16,"检测到模块!!!",16,0);
    delay_ms(500);
    Menu_ui();//菜单

    while(1)
    {

        key = KEY_Scan(0);
        if(key)
        {
            Show_Str(30+10,95+45+netpro*25,200,16,"    ",16,0);//清空之前的显示

              if(key==KEY0_PRES)//KEY0 按下
            {
                if(netpro<6)netpro++;
                else netpro=0;
            }else if(key==KEY1_PRES)//KEY1 按下
            {
                if(netpro>0)netpro--;
```

```
                    else netpro=6;
                }else if(key==WKUP_PRES)//KEY_UP 按下
                {
                    if(netpro==0)//进入通信选项
                    {
                       LoRa_Process();//开始数据测试
                       netpro=0;//索引返回第 0
                       Menu_ui();

                    }else
                    {
                        Show_Str(30+40,95+45+netpro*25+2,200,16,"________",16,1);
                        //显示下划线,表示选中
                        Show_Str(30+10,95+45+netpro*25,200,16,"→",16,0);
                        //指向新条目
                        Menu_cfg(netpro);//参数配置
                        LCD_Fill(30+40,95+45+netpro*25+2+15,30+40+100,
                        95+45+netpro*25+2+18,WHITE);//清除下划线显示
                    }
                }
                Show_Str(30+10,95+45+netpro*25,200,16,"→",16,0);//指向新条目
            }

            t++;
            if(t==30)
            {
                t=0;
                LED1=~LED1;
            }
            delay_ms(10);
        }

    }
```

该函数中先对模块进行初始化，初始化成功进入主菜单页面，通过按键配置参数或进入测试功能，代码比较简单。

以上就是 lora_app.c 的全部代码，最后我们说下 maic.c，该文件里面就 1 个 main 函数 main 函数具体代码如下：

```
    //主函数
     int main(void)
     {
         u8 key,fontok=0;
         NVIC_Configuration();
         delay_init();                    //延时函数初始化
         uart_init(115200);               //串口初始化为 9600
```

```
usmart_dev.init(72);                    //初始化 USMART
LCD_Init();                             //初始化液晶
LED_Init();                             //LED 初始化
KEY_Init();                             //按键初始化
tp_dev.init();                          //触摸屏初始化
mem_init();                             //初始化内存池
exfuns_init();                          //为 fatfs 相关变量申请内存
f_mount(fs[0],"0:",1);                  //挂载 SD 卡
f_mount(fs[1],"1:",1);                  //挂载 FLASH.
key=KEY_Scan(0);
if(key==KEY0_PRES)                      //强制校准
{
    LCD_Clear(WHITE);                   //清屏
    tp_dev.adjust();                    //屏幕校准
    LCD_Clear(WHITE);                   //清屏
}
fontok=font_init();                     //检查字库是否 OK
if(fontok||key==KEY1_PRES)              //需要更新字库（字库不存在/KEY1 按下）
{
     LCD_Clear(WHITE);                  //清屏
     POINT_COLOR=RED;                   //设置字体为红色
     LCD_ShowString(60,50,200,16,16,"ALIENTEK STM32");
     while(SD_Initialize())             //检测 SD 卡
     {
          LCD_ShowString(60,70,200,16,16,"SD Card Failed!");
          delay_ms(200);
          LCD_Fill(60,70,200+60,70+16,WHITE);
          delay_ms(200);
     }
     LCD_ShowString(60,70,200,16,16,"SD Card OK");
     LCD_ShowString(60,90,200,16,16,"Font Updating...");
     key=update_font(20,110,16); //更新字库
     while(key)                         //更新失败
     {
          LCD_ShowString(60,110,200,16,16,"Font Update Failed!");
          delay_ms(200);
          LCD_Fill(20,110,200+20,110+16,WHITE);
          delay_ms(200);
     }
     LCD_ShowString(60,110,200,16,16,"Font Update Success!");
     delay_ms(1500);
     LCD_Clear(WHITE);                  //清屏
}
Lora_Test();//主测试
```

```
}
```

此部分代码比较简单，先初始化一系列外设，然后判断触摸屏是否校准，不存在着触摸屏校准，接着判断字库是否存在，不存在则进行字库更新（在启动的时候，按下 KEY1 可以强制进行字库的更新），最后调用 Lora_Test 函数对模块进行测试。

至此，软件实现部分介绍完了，我们接下来看代码验证。

4、验证

首先，请先确保硬件都已经连接好了：

1，ATK-LORA-02 模块与 ALIENTEK MiniV3 STM32 开发板连接（连接方式见 2.3 小节）

2，ALIENTEK MiniV3 STM32 开发板插上 TFTCLD 液晶。

3，給 ALIENTEK MiniV3 STM32 开发板供电。

代码编译成功之后，我们将代码下载到 STM32 开发板上。LCD 界面显示如图 4.1 所示：

图 4.1 本测试实验界面

可以看到，初始成功后 LCD 屏幕显示参数的配置和选项栏，同时显示了 KEY_UP/KEY1/KEY0 测试功能选项，KEY_UP 为确定或取消功能、KEY1 为上一个选项功能、KEY0 为下一个选项功能。箭头选中参数配置栏，然后按下 KEY_UP，会选中该项，并且会显示下划线，然后按下 KEY0 或 KEY1 可以对该栏设置参数，如图 4.2 所示：

图 4.2 设置参数

4.1 透明传输

在主菜单界面，发送状态为“透明传输”，选择“进入通信”然后按 KEY_UP，则可进入测试，屏幕左上方会显示模块配置的参数，在右上方则会提示 KEY_UP 返回主菜单、KEY1 发送数据，如图 4.1.1 所示：

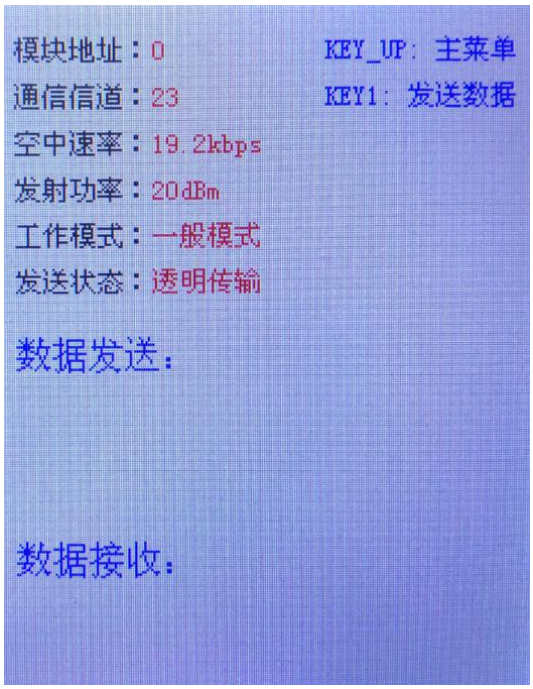

图 4.1.1 透明传输页面

按下 KEY1 则液晶屏会显示当前发送的数据，DS0 红灯会先亮后灭，表示数据已发送完毕，若接收方为同一地址和信道就会收到这数据，同样 DS0 红灯也会先亮后灭，表示数据接收完毕。按下 KEY_UP 返回主菜单配置参数页面，数据发送和接收如图 4.1.2 所示：

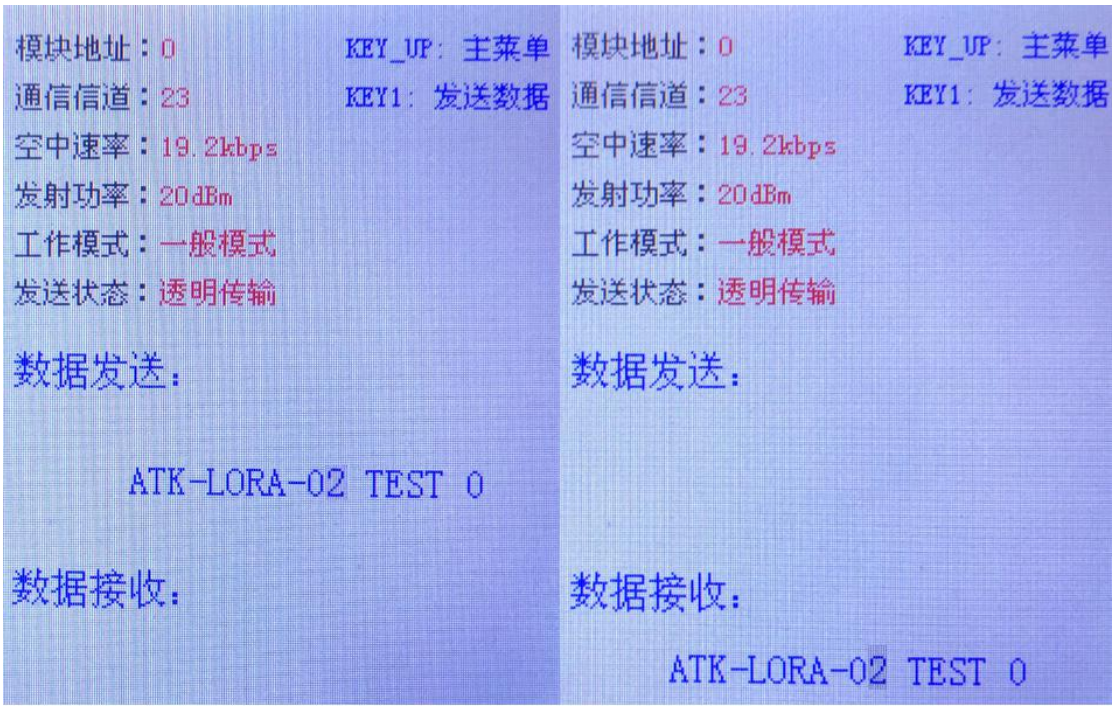

图 4.1.2 数据显示页面

4.2 定向传输

在主菜单界面，发送状态为“定向传输”，选择“进入通信”然后按 KEY_UP，则可进入测试，屏幕左上方会显示模块配置的参数，在右上方则会提示 KEY_UP 按键返回主菜单、KEY1 发送数据，KEY0 设置参数，如图 4.2.1 所示：

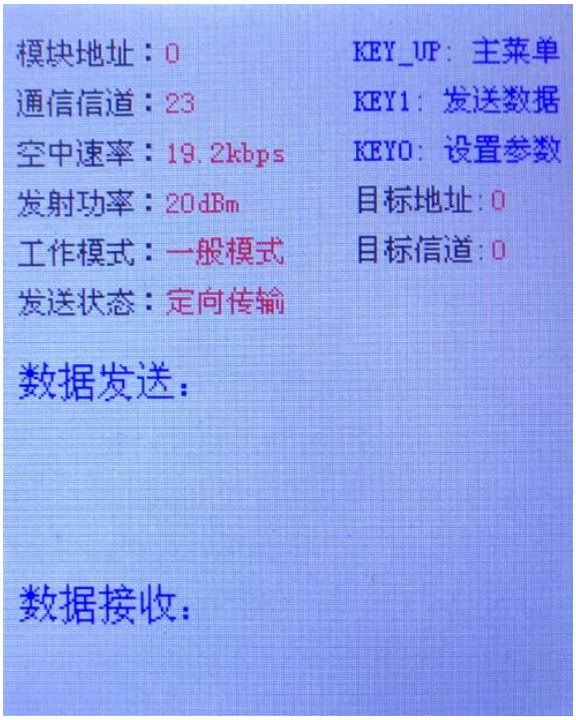

图 4.2.1 定向传输页面

按下 KEY0 会进入配置发送目标地址和目标信道的界面，设置地址范围为 0~65535，信道范围：0~31，设置界面如图 4.2.2 所示：

图 4.2.2 设置目标地址和信道界面

配置完成后返回测试界面，按下 KEY1 则液晶屏会当前发送的数据，DS0 红灯会先亮后灭，表示数据已发送完毕，若接收方为同一地址和信道就会收到这数据，同样 DS0 红灯也会先亮后灭，表示数据接收完毕。按下 KEY_UP 返回主菜单配置参数页面，数据发送和接收会显示在屏幕上（发送数据前三个字节为目标高位地址、目标低位地址、目标信道）如图 4.2.3 所示：

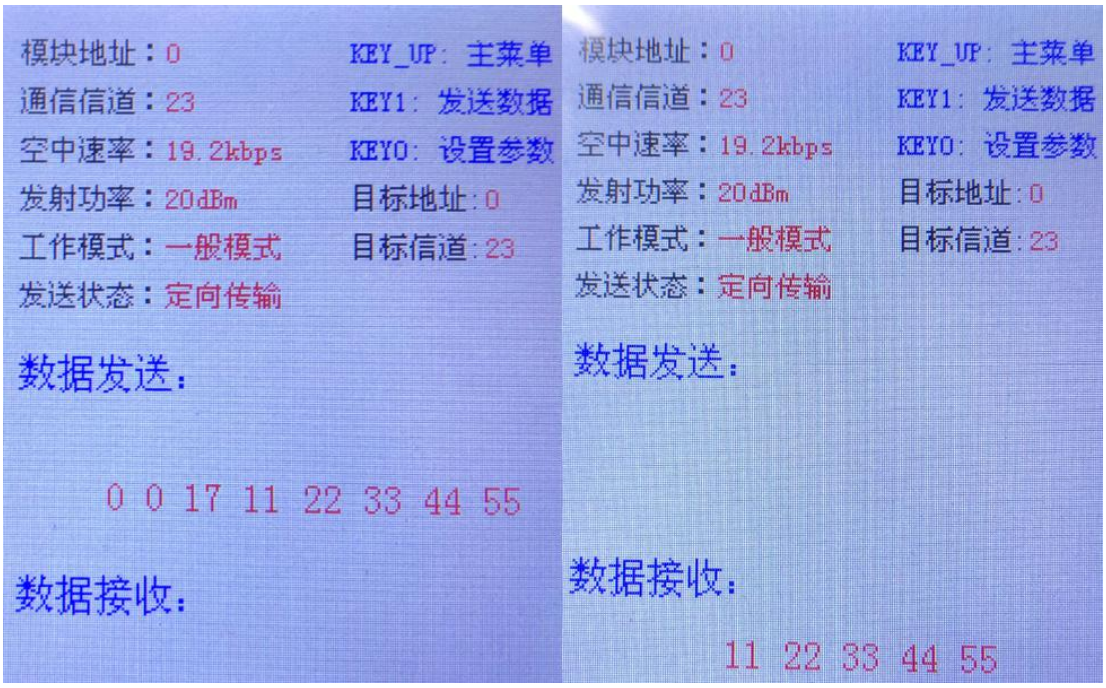

图 4.2.3 数据显示页面

注意：由于历程中当设置为透明传输时，液晶显示的是以字符串显示，而定向传输时，显示以 16 进制数据显示，若发送端和接收端设置的传输不一样，会导致接收端液晶显示数据乱码，所以测试的时候最好设置两者的传输方式一样。

至此，关于 ATK-LORA-02 模块的使用介绍，我们就讲完了，本文档介绍了 ATK-LORA-02

模块的使用，有助于大家快速学会 ATK-LORA-02 模块的使用。

正点原子@ALIENTEK
2019-5-6
公司网址：www.alientek.com
技术论坛：www.openedv.com
电话：020-38271790
传真：020-36773971