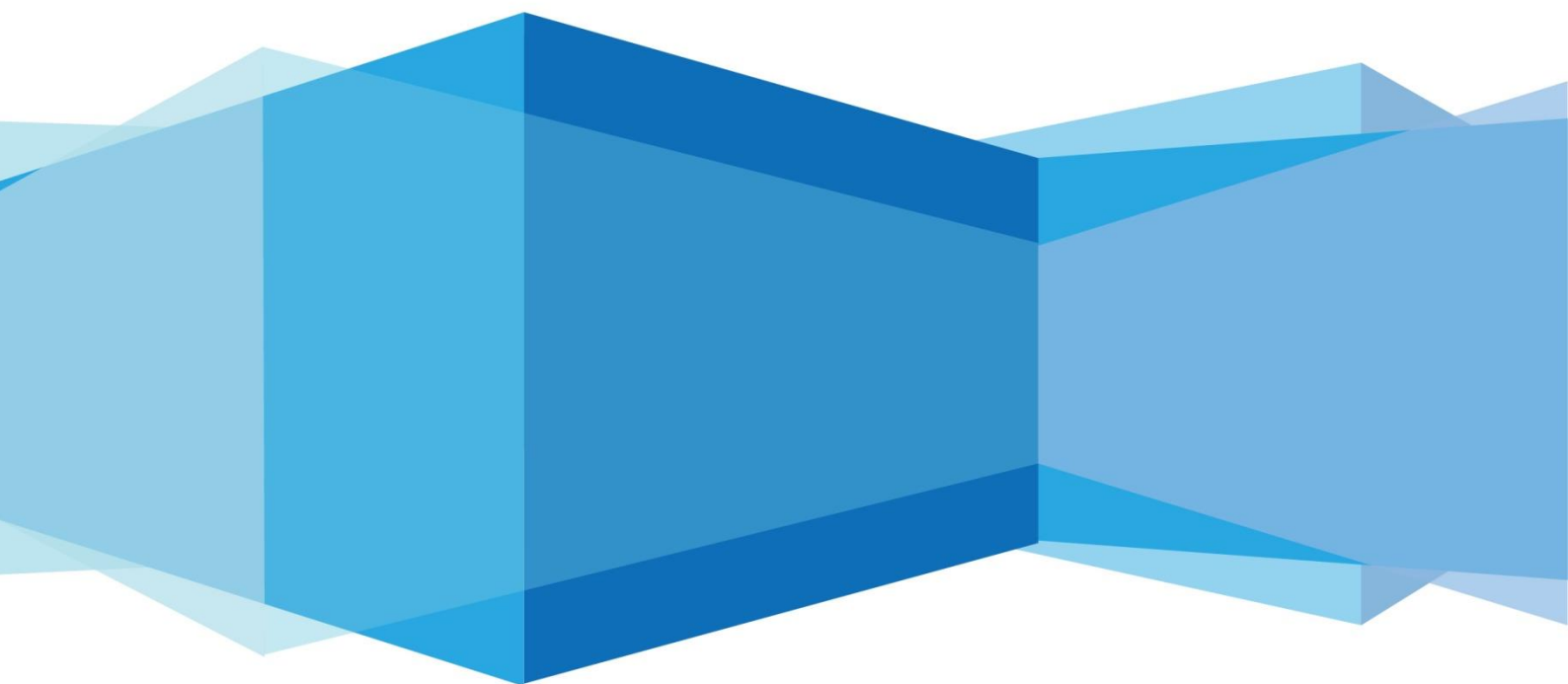


# Lierda NB86-G

## OpenCPU\_UART 应用笔记

版本：Rev1.0

日期：2019-01-11



## 法律声明

若接收浙江利尔达物联网技术有限公司（以下称为“利尔达”）的此份文档，即表示您已经同意以下条款。若不同意以下条款，请停止使用本文档。

本文档版权所有浙江利尔达物联网技术有限公司，保留任何未在本文档中明示授予的权利。文档中涉及利尔达的专有信息。未经利尔达事先书面许可，任何单位和个人不得复制、传递、分发、使用和泄漏该文档以及该文档包含的任何图片、表格、数据及其他信息。

本产品符合有关环境保护和人身安全方面的设计要求，产品的存放、使用和弃置应遵照产品手册、相关合同或者相关法律、法规的要求进行。

本公司保留在不预先通知的情况下，对此手册中描述的产品进行修改和改进的权利；同时保留随时修订或收回本手册的权利。

## 文件修订历史

版本	修订日期	修订日志
1.0	2018-12-05	新建文档

--

## 安全须知

用户有责任遵循其他国家关于无线通信模块及设备的相关规定和具体的使用环境法规。通过遵循以下安全原则，可确保个人安全并有助于保护产品和工作环境免遭潜在损坏。我司不承担因客户未能遵循这些规定导致的相关损失。



道路行驶安全第一！当您开车时，请勿使用手持移动终端设备，除非其有免提功能。请停车，再打电话！



登机前请关闭移动终端设备。移动终端的无线功能在飞机上禁止开启以防止对飞机通讯系统的干扰。忽略该提示项可能会导致飞行安全，甚至触犯法律。



当在医院或健康看护场所，注意是否有移动终端设备使用限制。RF 干扰会导致医疗设备运行失常，因此可能需要关闭移动终端设备。



移动终端设备并不保障任何情况下都能进行有效连接，例如在移动终端设备没有花费或 SIM 无效。当您在紧急情况下遇见以上情况，请记住使用紧急呼叫，同时保证您的设备开机并且处于信号强度足够的区域。



您的移动终端设备在开机时会接收和发射射频信号，当靠近电视，收音机电脑或其它电子设备时都会产生射频干扰。



请将移动终端设备远离易燃气体。当您靠近加油站，油库，化工厂或爆炸作业场所，请关闭移动终端设备。在任何有潜在爆炸危险场所操作电子设备都有安全隐患。

## 目 录

法律声明.....	2
文件修订历史.....	3
安全须知.....	4
目 录.....	5
1. 引言.....	7
2. NB86-G 引脚说明.....	7
3. AT 串口（低功耗串口）使用说明.....	8
3.1. Low-Power UART 特性.....	8
3.2. 硬件参考设计.....	8
3.3. 软件参考设计.....	9
3.3.1. 初始化.....	9
3.3.2. 串口接收.....	9
3.3.3. 串口发送.....	9
3.4. 样例测试.....	10
3.4.1. 样例概述.....	10
3.4.2. 串口初始化.....	11
3.4.3. 串口数据接收.....	11
3.4.4. 串口数据发送.....	12
3.4.5. 样例测试结果.....	13
3.5. 设计要点.....	14
4. 第三路标准串口（非低功耗串口）使用说明.....	14
4.1. UART 特性.....	14
4.2. 硬件参考设计.....	15
4.3. 软件参考设计.....	15
4.3.1. 串口初始化.....	15
4.3.2. 串口接收.....	16
4.3.3. 串口发送.....	16
4.3.4. 关闭串口.....	16

4.4.	测试样例.....	17
4.4.1.	样例概述.....	17
4.4.2.	串口初始化.....	17
4.4.3.	串口数据接收.....	18
4.4.4.	串口数据发送.....	20
4.4.5.	样例测试结果.....	21
4.4.6.	设计要点.....	23
5.	相关文档及术语缩写.....	23

## 1. 引言

本文旨在帮助基于使用 Lierda NB86-G 模组进行 OpenCPU 开发的用户，让其能快速使用模组本身的各种硬件资源（I2C、GPIO、UART）和 LiteOS 操作系统（创建、删除、挂起、恢复线程。创建、删除、启动、停止软件定时器），文章概述了低功耗串口（AT 串口）和 openCPU 映射的第三路标准 UART 的特性、软硬件参考设计、样例说明以及设计要点。

## 2. NB86-G 引脚说明

VDD\_IO\_R1、VDD\_IO\_R2、VDD\_IO\_L1、VDD\_IO\_L2 四个电源域的 DC 特性都相同，其中 [PIO0~PIO9] 从属 VDD\_IO\_R1 电源域，[PIO10~PIO21] 从属 VDD\_IO\_R2 电源域，[PIO22~PIO33] 从属 VDD\_IO\_L1 电源域，[PIO34~PIO39] 从属 VDD\_IO\_L2 电源域，在选择 PIO 时注意电源域的选择，如图 2-1。

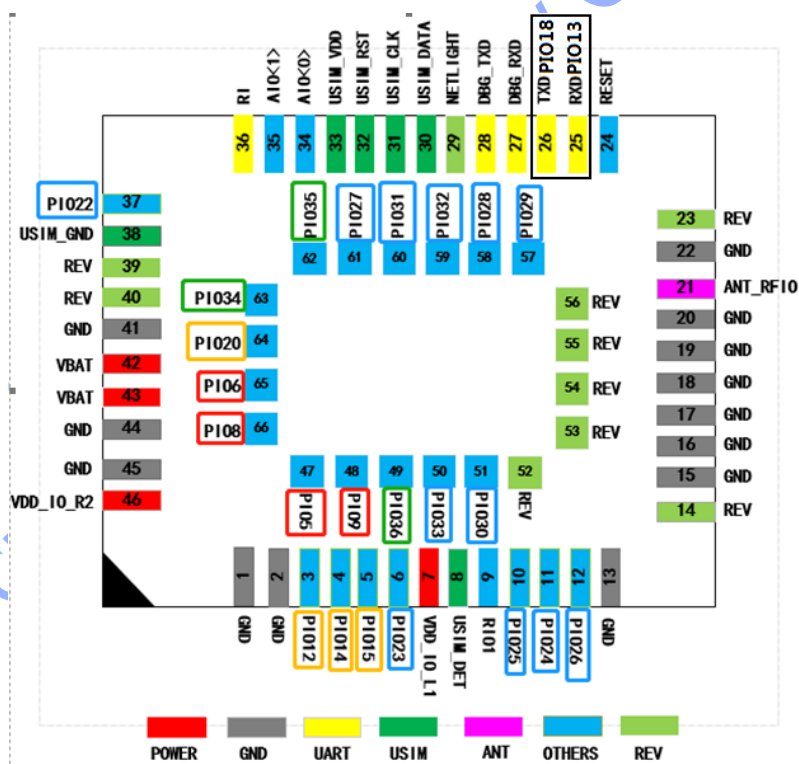


图 2-1 NB86-G 电源域图

说明：

红色框起来的 PIO 属于 VDD\_IO\_R1 电源域；黄色框起来的 PIO 属于 VDD\_IO\_R2 电源域；蓝色框起来的 PIO 属于 VDD\_IO\_L1 电源域；绿色框起来的 PIO 属于 VDD\_IO\_L2。

VDD\_IO\_R1 电源域：1.8V 电平；

VDD\_IO\_L2 电源域：由于 USIM 卡的 IO 是从属于 VDD\_IO\_L2 电源域的，所以该电源域的电平取决于外接 SIM 卡的电平。

VDD\_IO\_R2、VDD\_IO\_L1 电源域：默认值 3.0V。

**NOTE：**AT 串口 TX（图 2-1 中标号 26 的 TXD）的引脚号为：PIO18；AT 串口 RX（图 2-1 中标号 25 的 RXD）的引脚号为：PIO13。

## 3. AT 串口（低功耗串口）使用说明

### 3.1. Low-Power UART 特性

低功耗 UART 可在深度睡眠模式下接收 UART 数据，也可以在活动和待机模式下访问。LPUART 不支持硬件流控制（CTS / RTS）。使用低功耗串口收发数据不会引起 PSM 功耗异常的现象，发送/接收完数据后功耗会降至 PSM 正常功耗。具体的参数配置如下表：

表 3-1 3.1. Low-Power UART 参数配置

参数名	取值范围	备注
波特率	4800 ~ 256000	
数据位	5、6、7、8	
停止位	1、2	
奇偶校验	无校验、奇校验、偶校验	
数据收发长度	<1024 字节	
Tx&Rx 引脚定义	Tx: PIO18 Rx: PIO13	

### 3.2. 硬件参考设计

终端串口电平不匹配时需要增加电平转换电路，可参考图 3-1 设计。串口引脚需要选择同一个电源域的通用 GPIO。Terminal\_VCC 上拉电平需要和终端串口电平保持一致。

电平转换 NB 模组端的上拉电平一定要选择 R2 电源做上拉。



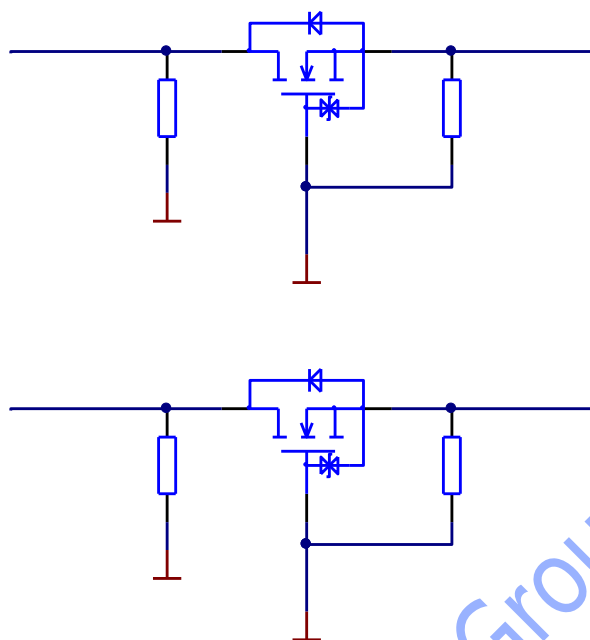


图 3-1 硬件参考电路

### 3.3. 软件参考设计

#### 3.3.1. 初始化

初始化函数：`void lierdaUARTInit(UART_HandleTypeDef *huart)`

串口参数配置：波特率、数据位、停止位、奇偶校验位参考 3.1 小节中特性概述配置即可，收发引脚定义：Tx: PIO18 Rx: PIO13。

#### 3.3.2. 串口接收

串口接收函数：`lierdaUARTReceive(UART_HandleTypeDef *huart, uint8 *UserDataPtr, uint16 *UserDataLen, uint32 WiatTimeOut)`

串口接收函数的调用会引起任务的阻塞，建议在使用第三路串口时单独建立一个任务来处理串口数据。

#### 3.3.3. 串口发送

串口发送函数：`lierdaUARTSend(UART_HandleTypeDef *huart, const uint8 *buffer, uint32 length)`串口发送调用发送函数即可，数据发送的最大长度：1024 字节。

## 3.4. 样例测试

### 3.4.1. 样例概述

以华大北斗 GPS 数据为例（数据长度：600 字节左右），线程 2（串口数据接收）接收到数据后做包头包尾处理，当接收到完整一包的数据后发送接收完成消息给线程 1；线程 1（业务线程）在收到队列发来的消息后做判断，若消息是串口数据接收完成则将接收到的 GPS 数据发送给 AT 串口。

GPS 测试数据：

```
$GNGGA,053929.000,3016.71670,N,11959.30658,E,1,12,0.74,88.4,M,6.8,M,,*72
$GPGSA,A,3,193,15,13,05,24,194,21,29,02,195,20,30,1.04,0.74,0.74,1*2E
$BDGSA,A,3,,,,,,,,,,1.04,0.74,0.74,4*0E
$GPGSV,3,1,12,193,71,52,32,15,70,304,24,13,52,32,38,5,38,77,25*47
$GPGSV,3,2,12,24,37,174,23,194,36,138,27,21,35,314,21,29,26,234,17*40
$GPGSV,3,3,12,2,18,149,21,195,16,174,21,20,11,299,20,30,5,38,29*71
$BDGSV,2,1,05,3,53,193,,1,38,129,,4,36,124,,2,35,240,*64
$BDGSV,2,2,05,5,10,261,*6C
$GNRMC,053929.000,A,3016.71670,N,11959.30658,E,0.000,50.63,021118,,A*71
$GNZDA,053929.000,02,11,2018,,*45
$GPTIZ,E,8.0*33
$GNZDA,053929.000,02,11,2018,,*45
$GPTIZ,E,8.0*33
```

串口配置：波特率：9600 数据位：8 校验位：无 停止位：1

NOTE：打开样例后，使能 AT 串口测试如图 3-1 AT 串口使能。

```
#define USERATUART 1 //使能AT串口
```

```
#define GPS_SIZE 700
```

图 3-2 AT 串口使能

### 3.4.2. 串口初始化

测试代码如下：

```
/******  
 * @函数名      串口初始化函数  
 * @参数        无  
 * @返回值      无  
*****/  
  
static void Uart_init(void)  
{  
    UARHandle.baudrate = 9600;  
    UARHandle.data_bits = UART_DATA_BITS_8;  
    UARHandle.parity = UART_PARITY_NONE;  
    UARHandle.stopbits = UART_STOP_BITS_1;  
    UARHandle.rx_pin = PIN_13;  
    UARHandle.tx_pin = PIN_18;  
    lierdaUARTInit(&UARHandle); //串口初始化  
}
```

### 3.4.3. 串口数据接收

AT 串口在接收数据不会产生分包接收，示例代码如下：

```
/******  
 * @函数名  AT串口数据接收任务  
 * @参数 param : 空参数  
 * @返回值  无  
*****/  
  
static void lierda_UART_task(void *param)  
{  
    uint8 uart_flag=1;  
    UNUSED(param);  
    osDelay(500); //等待模组初始化完成  
    lierdaLog("DBG_INFO:串口接收数据线程"); //通过AT指令串口打印Log  
    Uart_init(); //串口初始化  
    Uart_dataclear();  
    for(;;)  
{
```

```

    lierdaUARTReceive(&UARTHandle, Gps_usrt_buff, &GPS_DATA_len, 0xFFFFFFFF);
//串口数据接收
    if (GPS_DATA_len > 0)
    {
        if(strstr((const char *)Gps_usrt_buff, "$GNGGA")) //判断是不是数据的包头
        {
            if(strstr((const char *) Gps_usrt_buff, "$GPTIZ"))//判断是否接收完成
            {
                if(mess_QueueId!=NULL)
                    osMessageQueuePut (mess_QueueId,&uart_flag, 0, osNoWait); //发送串口接收完成标志
            }
            else
            {
                Uart_dataclear();//GPS buff清空
            }
        }
        else
        {
            Uart_dataclear();//GPS buff清空
        }
    }
    osDelay(1);//用于任务切换
}
}

```

### 3.4.4. 串口数据发送

示例代码如下：

```

/*****
* @函数名  AT串口主线程
* @参数 param：空参数
* @返回值 无
*****/
static void lierda_App_task(void *param)
{
    uint8 temp=0;
    UNUSED(param);
    osDelay(500); //等待模组初始化完成
    lierdaLog("DBG_INFO:主线程"); //通过AT指令串口打印Log
    for(;;)
    {
        if (mess_QueueId != NULL)
        {

```

```

    if (osMessageQueueGet(mess_QueueId, &temp, NULL, 0xffffffff)== osOK) //
    队列接收串口接收完消息标志
    {
        if (temp == 1)
        {
            lierdaUARTSend(&UARTHandle,Gps_usrt_buff, GPS_DATA_len); //发送
            接收到的数据
            Uart_dataclear();
        }
    }
    osDelay(1); //用于任务切换
}
}

```

### 3.4.5. 样例测试结果

- 1、 AT 串口收到数据后 GPS 数据后会通过 AT 串口把收到的数据发送出去。串口调试助手数据如图 3-2 AT 串口测试数据。

```

[17:17:36.023]发->◇$GNGGA,053929.000,3016.71670,N,11959.30658,E,1,12,0.74,88.4,M,6.8,M,,*72
$GPGSA,A,3,193,15,13,05,24,194,21,29,02,195,20,30,1.04,0.74,0.74,1*2E
$BDGSA,A,3,,,,,,,,,,,,,1.04,0.74,0.74,4*0E
$GPGSV,3,1,12,193,71,52,32,15,70,304,24,13,52,32,38,5,38,77,25*
[13:39:27.235]收<-◆47
$GPGSV,3,2,12,24,37,174,23,194,36,138,27,21,35,314,21,29,26,234,17*40
$GPGSV,3,3,12,2,18,149,21,195,16,174,21,20,11,299,20,30,5,38,29*71
$BDGSV,2,1,05,3,53,193,,1,38,129,,4,36,124,,2,35,240,*64
$BDGSV,2,2,05,5,10,261,*6C
$GNRMC,053929.000,A,3016.71670,N,11959.30658,E,0.000,50.63,021118,,A*71
$GNZDA,053929.000,02,11,2018,,*45
$GPTIZ,E,8.0*33
□
[17:17:36.222]收<-◆$GNGGA,053929.000,3016.71670,N,11959.30658,E,1,12,0.74,88.4,M,6.8,M,,*72
$GPGSA,A,3,193,15,13,05,24,194,21,29,02,195,20,30,1.04,0.74,0.74,1*2E
$BDGSA,A,3,,,,,,,,,,,,,1.04,0.74,0.74,4*0E
$GPGSV,3,1,12,193,71,52,32,15,70,304,24,13,52,32,38,5,38,77,25*
[13:39:27.235]收<-◆47
$GPGSV,3,2,12,24,37,174,23,194,36,138,27,21,35,314,21,29,26,234,17*40
$GPGSV,3,3,12,2,18,149,21,195,16,174,21,20,11,299,20,30,5,38,29*71
$BDGSV,2,1,05,3,53,193,,1,38,129,,4,36,124,,2,35,240,*64
$BDGSV,2,2,05,5,10,261,*6C
$GNRMC,053929.000,A,3016.71670,N,11959.30658,E,0.000,50.63,021118,,A*71
$GNZDA,053929.000,02,11,2018,,*45
$GPTIZ,E,8.0*33

```

图 3-3 AT 串口测试数据

- 2、 AT 串口功耗测试结果：PSM 模式下能收发数据，收发数据后功耗回复 PSM 正常功耗 3μA 左右，发送/接收数据的平均功耗：3.29mA。测试结果如图 3-3 AT 串口功耗测试图

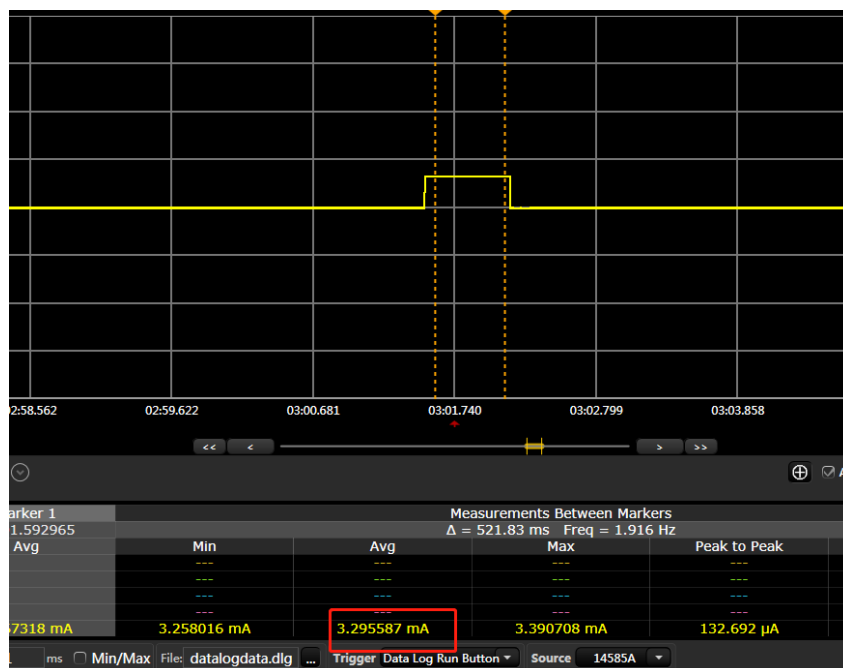


图 3-4 AT 串口功耗测试图

3、 AT 串口 Tx 发送数据波形分析如图 3-4 AT 串口 Tx 数据波形。

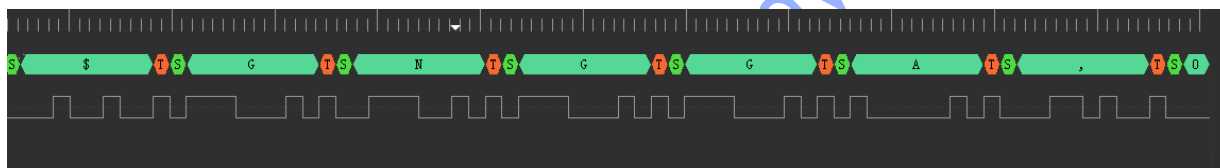


图 3-5 AT 串口 Tx 数据波形

### 3.5. 设计要点

- TX/RX 引脚配置: Tx: PIO18 Rx: PIO13

## 4. 第三路标准串口（非低功耗串口）使用说明

### 4.1. UART 特性

标准 UART 具有可编程波特率，可在深度睡眠模式下接收 UART 数据，也可以在活动和待机模式下访问。可以将 UART 接口分配给任何通用 GPIO。支持的配置参数如下表：

表 4-1 UART 参数配置

参数名	取值范围	备注
波特率	4800 ~ 256000	
数据位	5、6、7、8	
停止位	1、2	
奇偶校验	无校验、奇校验、偶校验	

数据收发长度	<1024 字节	
Tx&Rx 引脚定义	同一个电源域的通用 GPIO	参考 NB-IoT 模块硬件应用手册 NB86 型手册

## 4.2. 硬件参考设计

建议使用 R2 电源域所在的 IO 口，其它电源域所在 IO 不建议使用。

终端串口电平不匹配时需要增加电平转换电路，可参考图 4-1 设计。串口引脚需要选择同一个电源域的通用 GPIO。Terminal\_VCC 上拉电平需要和终端串口电平保持一致。

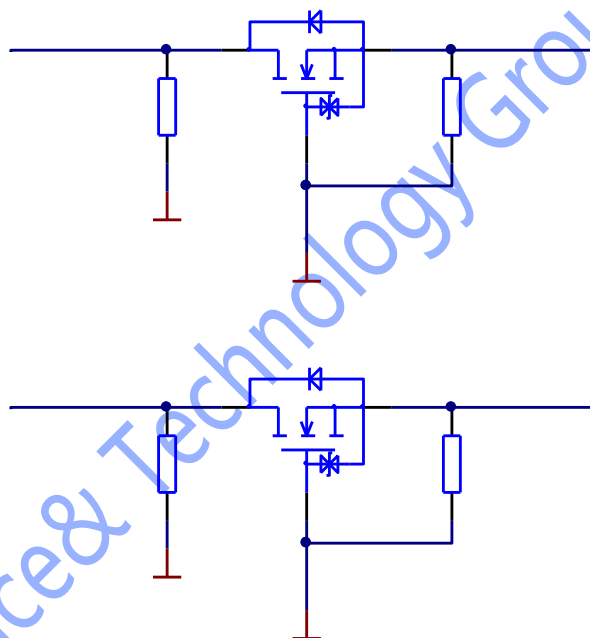


图 4-1 电平转换电路

## 4.3. 软件参考设计

### 4.3.1. 串口初始化

初始化函数：void lierdaUARTInit(UART\_HandleTypeDef \*huart)

串口参数配置：波特率、数据位、停止位、奇偶校验位参考小节 2 中特性概述配置即可，接收和发送引脚必须选择同一个电源域的通用 GPIO。

### 4.3.2. 串口接收

串口接收函数：lierdaUARTReceive(UART\_HandleTypeDef \*huart,uint8 \*UserDataPtr, uint16 \*UserDataLen, uint32 WiatTimeOut)

串口接收函数的调用会引起任务的阻塞，建议在使用第三路串口时单独建立一个任务来处理串口数据；发送给第三路串口的一包数据过大时，第三路串口会分包接收，在使用过程中注意接收数据的完整性，参考接收流程参考如图 4-1。

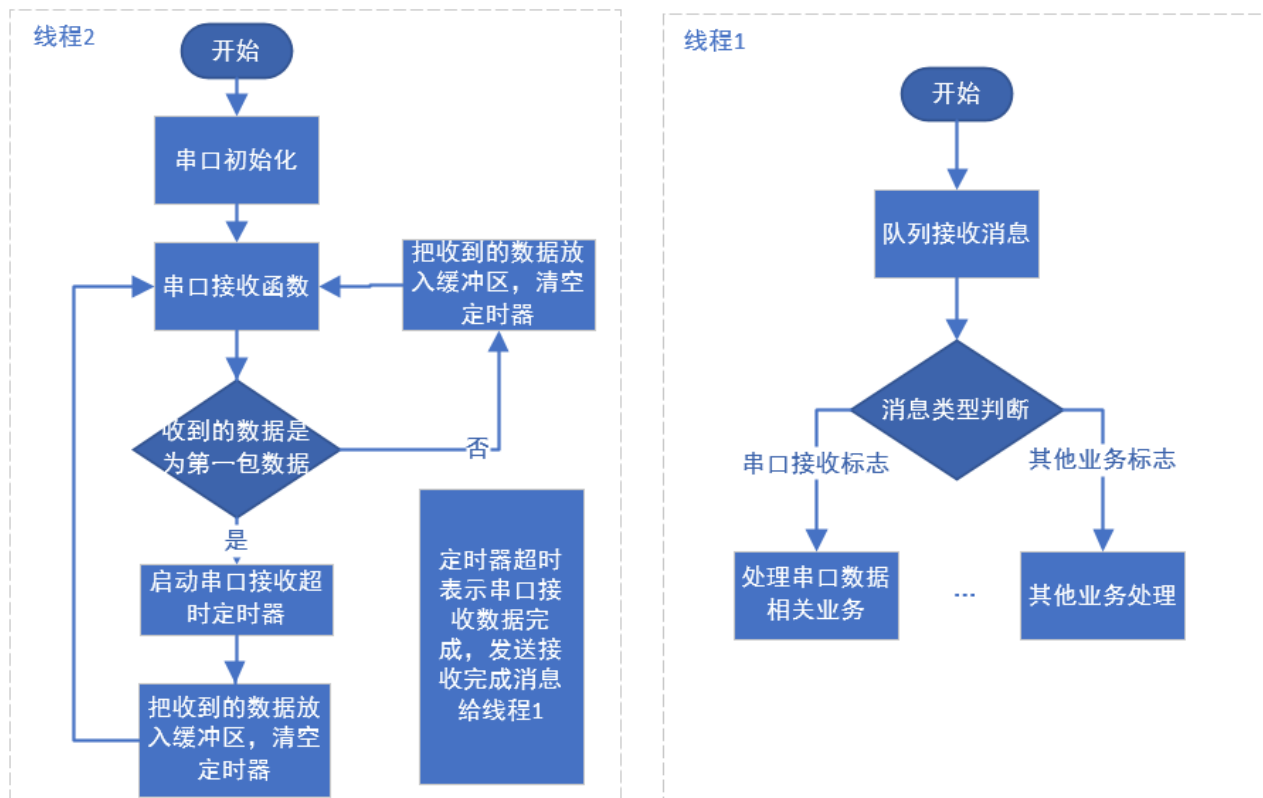


图 4-2 串口 3 接收流程参考

### 4.3.3. 串口发送

串口发送函数：lierdaUARTSend(UART\_HandleTypeDef \*huart,const uint8 \*buffer, uint32 length);

串口发送调用发送函数即可，数据发送的最大长度：1024 字节。

### 4.3.4. 关闭串口

关闭串口函数：void lierdaUARTClose(void);调用此函数后定义的第三路标准串口将被关闭，但 PSM 功耗不会恢复至 3μA 左右，依然保持在 2mA 附近。



## 4.4. 测试样例

### 4.4.1. 样例概述

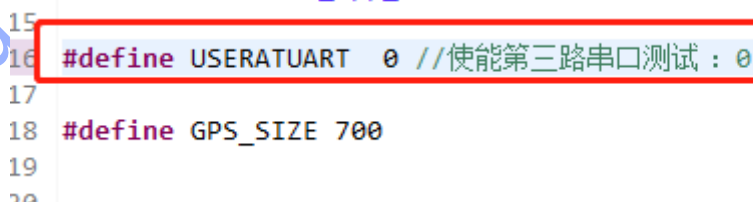
以华大北斗 GPS 数据为例（数据长度：600 字节左右），线程 2（串口数据接收）接收到数据后做包头包尾处理，当接收到完整一包的数据后发送接收完成消息给线程 1；线程 1（业务线程）在收到队列发来的消息后做判断，若消息是串口数据接收完成则将接收到的 GPS 数据发送给第三路串口。

GPS 测试数据：

```
$GNGGA,053930.000,3016.71672,N,11959.30664,E,1,12,0.74,88.3,M,6.8,M,,*70
$GPGSA,A,3,193,15,13,05,24,194,21,29,02,195,20,30,1.04,0.74,0.74,1*2E
$BDGSA,A,3,,,,,,,,,,,,,1.04,0.74,0.74,4*0E
$GPGSV,3,1,12,193,71,52,32,15,70,304,24,13,52,32,38,5,38,77,25*47
$GPGSV,3,2,12,24,37,174,24,194,36,138,27,21,35,314,22,29,26,234,17*44
$GPGSV,3,3,12,2,18,149,24,195,16,174,21,20,11,299,19,30,5,38,29*7E
$BDGSV,2,1,05,3,53,193,,1,38,129,,4,36,124,,2,35,240,*64
$BDGSV,2,2,05,5,10,261,*6C
$GNRMC,053930.000,A,3016.71672,N,11959.30664,E,0.000,50.63,021118,,,*74
$GNZDA,053930.000,02,11,2018,,*4D
$GPTIZ,E,8.0*33
```

串口配置：波特率：9600 数据位：8 校验位：无 停止位：1

NOTE：打开样例后，使能第三路 UART 测试如图 4-2：第三路 UART 测试使能。



```
15
16 #define USERATUART 0 //使能第三路串口测试 : 0
17
18 #define GPS_SIZE 700
19
20
```

图 4-3 第三路 UART 测试使能

### 4.4.2. 串口初始化

测试代码如下：

```

/*****
* @函数名      串口初始化函数
* @参数        无
* @返回值      无
*****/

static void Uart_init(void)
{
    UARTHandle.baudrate = 9600;
    UARTHandle.data_bits = UART_DATA_BITS_8;
    UARTHandle.parity = UART_PARITY_NONE;
    UARTHandle.stopbits = UART_STOP_BITS_1;
    UARTHandle.rx_pin = PIN_26;
    UARTHandle.tx_pin = PIN_24;
    lierdaUARTInit(&UARTHandle); //串口初始化
}

```

**NOTE:**初始化的 TX 和 RX 应定义在在同一个电源域，具体电源域的分配参考小节 2。

#### 4.4.3. 串口数据接收

第三路串口在接收数据时，当数据量比较大时串口接收 API 函数会分包接收，在使用时注意定时器的超时时间，尽量把超时时间定义长一些以保证能接收一包完整数据。接收任务示例代码如下：

```

/*****
* @函数名 串口3数据接收任务
* @参数 param : 空参数
* @返回值 无
*****/

static void lierda_UART_task(void *param)
{
    UNUSED(param);
    osDelay(500); //等待模组初始化完成
    lierdaLog("DBG_INFO: 串口3接收数据线程"); //通过AT指令串口打印Log
    Uart_init(); //串口初始化
    Uart_dataclear();
    for (;;)
    {

```

```

memset(usrt_buff, 0, uart3_SIZE);
uart3_buff_len = 0;
lierdaUARTReceive(&UARTHandle, usrt_buff, &uart3_buff_len, 0xFFFFFFFF); //
串口数据接收
if (uart3_buff_len > 0)
{
    if (UartData_Start == 0)
    {
        UartData_Start = 1;
        if (osTimerStart(osTimerId_liierda,30) != osOK)//启动一个30ms定时器
            lierdaLog("定时器启动失败");
        uartbuff_lenmonitor=0;//串口BUFF溢出检测清零
        Uart_dataclear(); //GPS BUFF清空
    }
    timer_count = 0; //定期器计数清空
    uartbuff_lenmonitor+=uart3_buff_len;
    if (uartbuff_lenmonitor > GPS_SIZE-1)//判断GPS buff是否溢出
    {
        uartbuff_lenmonitor=0;
        Uart_dataclear();
    }
    memcpy(Gps_usrt_buff + GPS_DATA_len, usrt_buff, uart3_buff_len);
    GPS_DATA_len+=uart3_buff_len;
}
osDelay(1); //用于任务切换
}
}

```

定时器回调函数实例代码如下，这里定义的接收超时时间为 300ms.

```

/*****
* @函数名      定时器回调函数 用于第三路串口接收超时定时
* @参数 param : 空参数
* @返回值 无
*****/
static void timer_Callback(void *argument)
{
    uint8 uart_flag=1;
    if(argument==(void *)1)
    {
        timer_count++;
        if(timer_count==10)
        {
            if(UartData_Start==1)
            {

```

```

        if (mess_QueueId != NULL)
            osMessageQueuePut(mess_QueueId, &uart_flag, 0, osNoWait); //发送串口
接收完成标志
        UartData_Start=0;
        timer_count=0;
        osTimerStop (osTimerId_lierda);
    }
}
}
}
}

```

#### 4.4.4. 串口数据发送

示例代码如下：

```

/*****
* @函数名   串口3主线程
* @参数 param : 空参数
* @返回值  无
*****/
static void lierda_App_task(void *param)
{
    uint8 temp=0;
    UNUSED(param);
    osDelay(500); //等待模组初始化完成
    lierdaLog("DBG_INFO:主线程"); //通过AT指令串口打印Log
    for(;;)
    {
        if (mess_QueueId != NULL)
        {
            if (osMessageQueueGet(mess_QueueId, &temp, NULL, 0xffffffff)== osOK) //
队列接收串口接收完消息标志
            {
                if (temp == 1)
                {
                    if (strstr((const char *) Gps_usrt_buff, "$GNGGA")) //判断是不是
数据的包头
                    {
                        if (strstr((const char *) Gps_usrt_buff, "$GPTIZ")) //判断是
否接收完成
                        {
                            lierdaLog("收到串口数据");
                            lierdaUARTSend(&UARTHandle,
Gps_usrt_buff, GPS_DATA_len); //发送接收到的数据
                            Uart_dataclear();

```

```

    }
    else
    {
        Uart_dataclear();
    }
}
else
{
    Uart_dataclear();
}
}
}
osDelay(1); //用于任务切换
}
}

```

#### 4.4.5. 样例测试结果

1、串口调试助手测试结果如图 4-3。

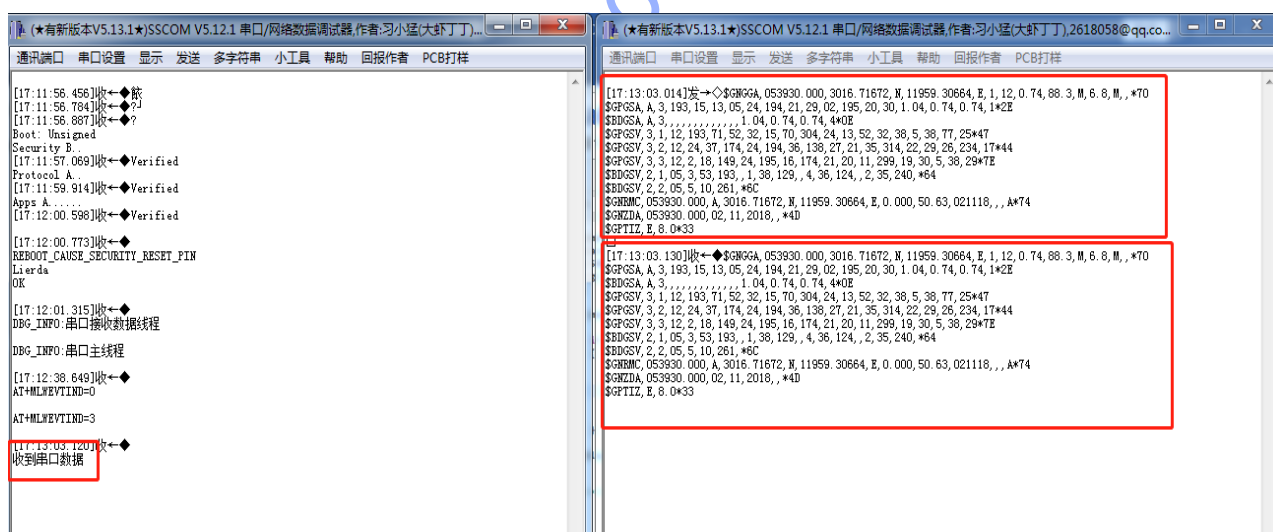


图 4-4 测试样例结果

2、功耗实测：

发送/接收数据长度：600 字节左右

Rx 功耗：Avg:2.038mA

功耗图如图 4-4

Tx 功耗：Avg:3.409mA

功耗图如图 4-5

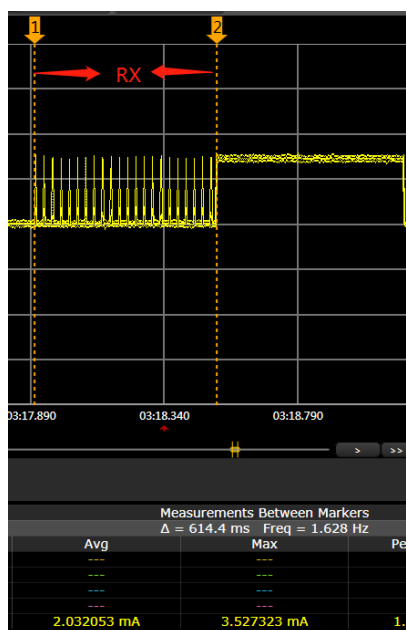


图 4-5 Rx 功耗测试

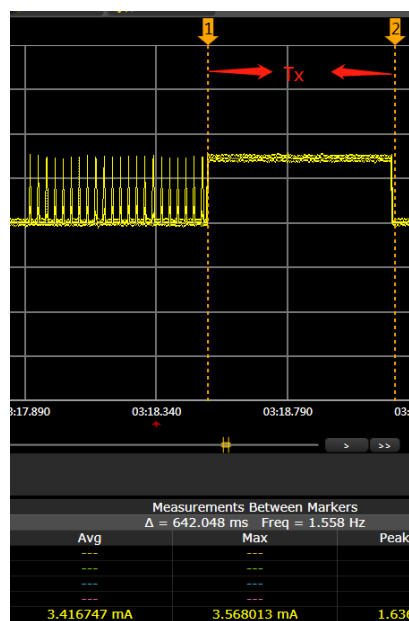


图 4-5 Tx 功耗测试

**NOTE:** 第三路串口不是低功耗串口，使用会引起 PSM 功耗高 2.0mA 左右如图 4-6。使用 `lierdaUARTClose` 函数关闭第三路串口功耗依旧高 2.0mA。

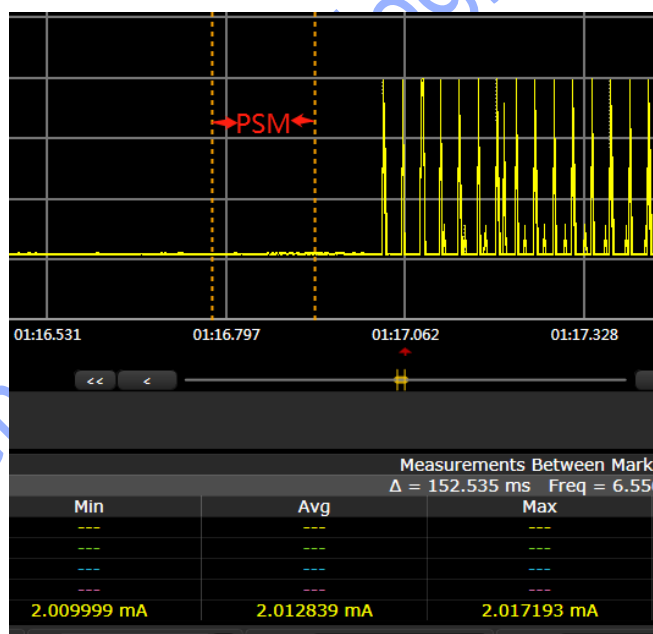


图 4-6 PSM 功耗图

4、第三路串口 Tx 波形如图 4-7。

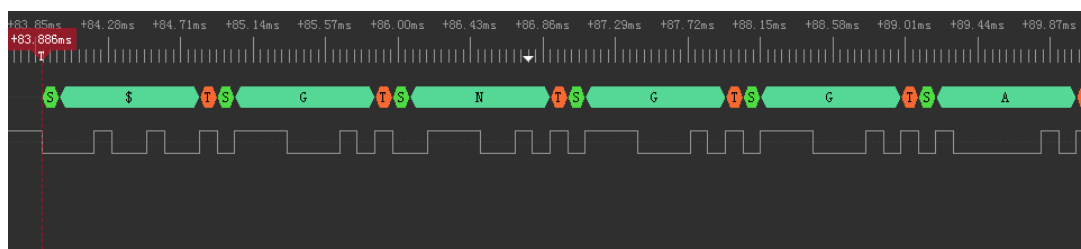


图 4-7 第三路串口 Tx 波形图

#### 4.4.6. 设计要点

- 定义的 Tx 引脚和 Rx 引脚要处于同一电源域，否则会导致第三路串口通信异常。
- 若选择 1.8V 电平的引脚请注意与外部做电平匹配。
- 在定义串口接收超时定时器时尽量定义较大的超时时间，以便于接收完整的数据包。

### 5. 相关文档及术语缩写

以下相关文档提供了文档的名称，版本请以最新发布的为准。

表格 5-1 相关文档

序号	文档名称	注释
[1]	NB86-G硬件应用手册	
[2]	Lierda NB Module V150_AT CommandSet_B300SP2	
[3]	Lierda NB86-EVK测试终端固件烧写教程	
[4]	Lierda NB-IoT模组API使用文档	
[5]	Lierda NB-IoT模组DEMO说明文档	
[6]	Lierda NB-IoT模组V150 OpenCPU开发环境搭建指南	
[7]	Lierda NB86-EVK操作使用手册	