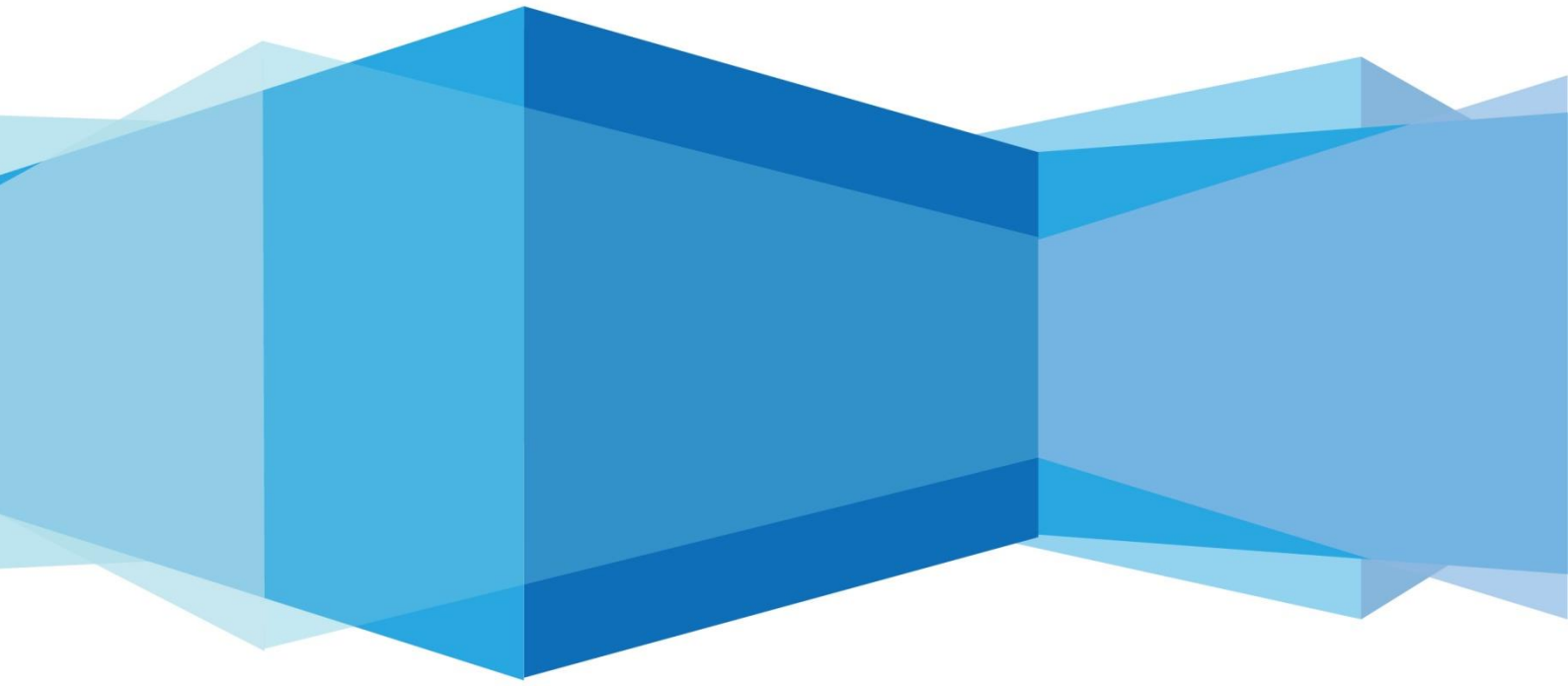
lierda® 利尔达科技集团 | 证券代码 832149

Lierda NB86-G

OpenCPU_FOTA 应用笔记

版本：Rev1.0

日期：2019-04-03



利尔达科技集团股份有限公司
LIERDA SCIENCE & TECHNOLOGY GROUP CO.,LTD

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	2019-03-29	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用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，收音机电脑或者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

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），文章概述了使用 OpenCPU 方案时进行 FOTA 升级时的操作介绍和注意事项。

2. 概述

使用 OpenCPU 方案时，客户的应用的代码在模组内开发，在应用程序更新时就可以使用远程升级的方式进行程序更新（FOTA 升级），远程方式能大大降低通过人工近端进行升级所投入的成本。FOTA(Fireware Over The Air)： 通过 NB-IoT 模组内置 LwM2M 协议的 5 号对象，实现对模组本身的升级，此升级方式采用差分升级的方式，升级流程如图 2-1.

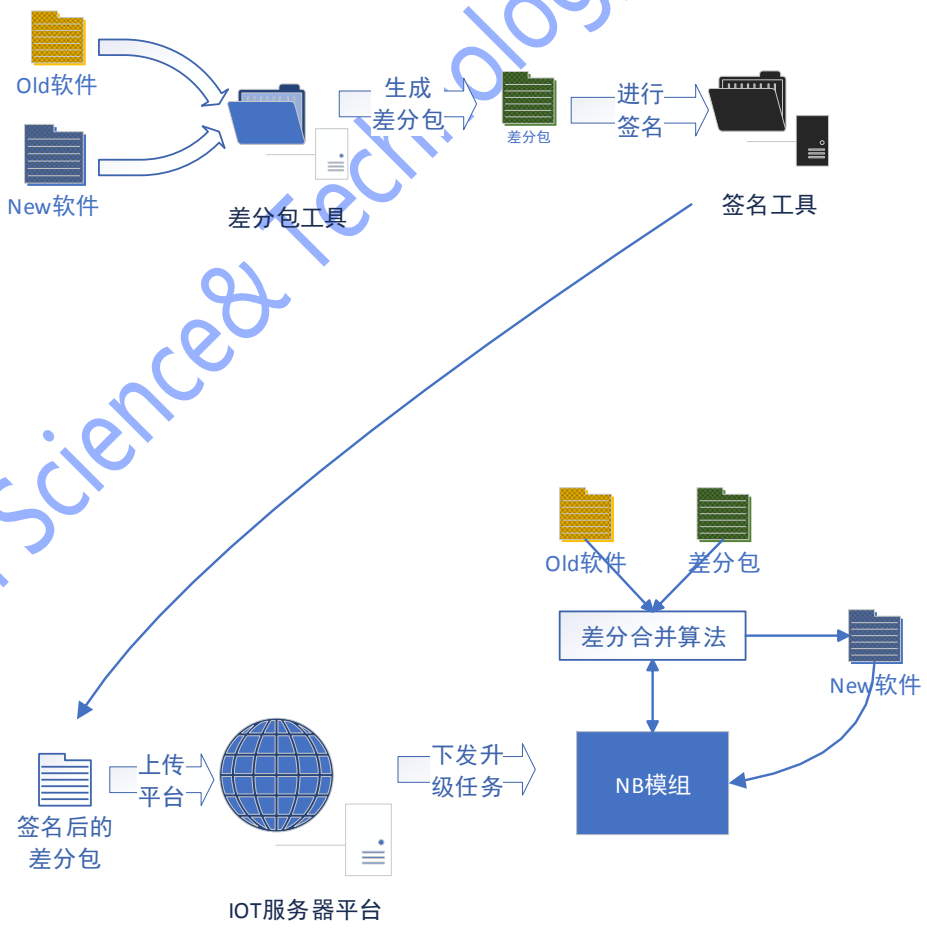


图 2-1 FOTA 升级流程

3. FOTA 升级操作流程

在 FOTA 升级之前，需要确保设备在 IoT 平台上已经成功注册，且能正常发送数据。

3.1. 生成差分包

1、 安装 FotaCreator 软件

2、 生成差分包（xx.bin 文件）

打开 FotaCreator 软件界面如图 3-1.

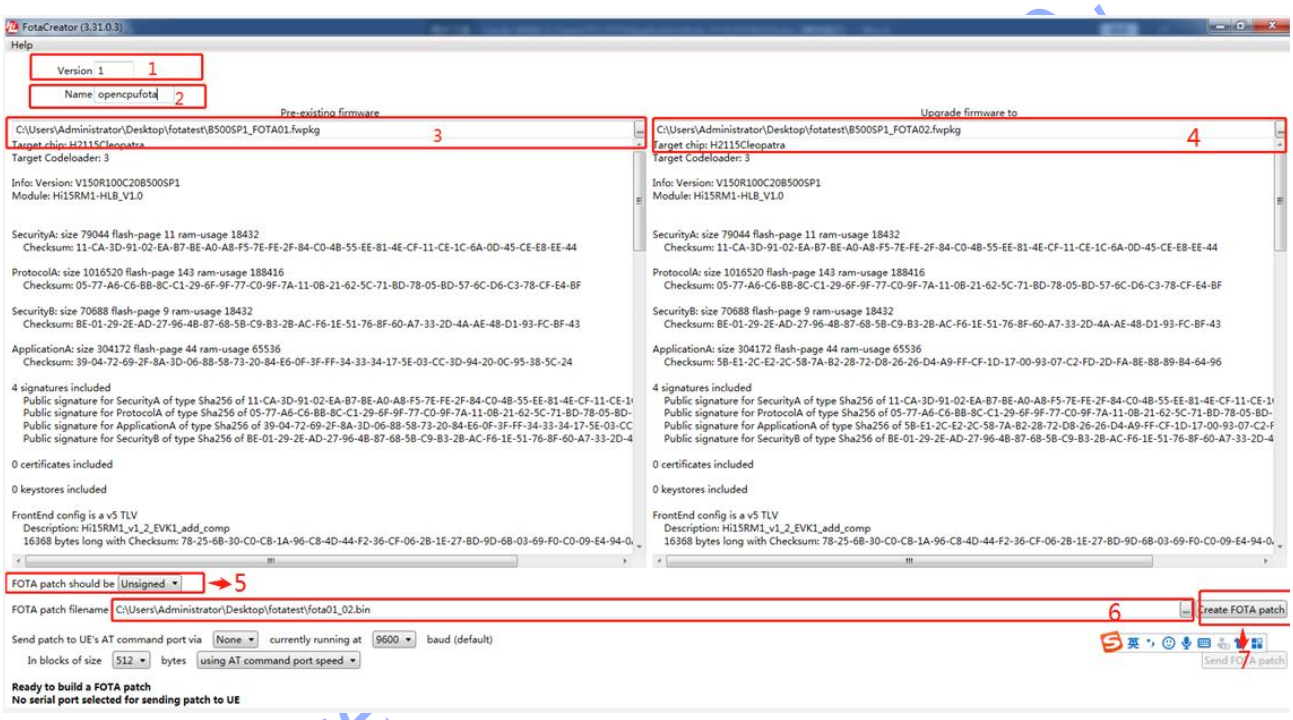


图 3-1 FOTA 升级工具界面参数配置

对图 3-1 中序号做如下说明：

1：为差分包版本号，这个版本选项填写 1 即可，且这个参数只能不变或者递增变化，但建议不要更改，一直填 1 即可，如果生成的差分包升级失败了，可以更改这个参数重新尝试（增大这个参数，如上一个差分包填 1，这个就可以填 2）

2：为差分包名字，将以包头的形式写入差分包中名字，可以随便填一个

3：原始固件路径

4：要更新的固件路径

5：选择 unsigned

6:生成差分固件的位置

7：执行生成

按照 3-1 中的序号操作，最后点击序号 6 即可生成要用的 xxx.bin 差分固件如图 3-2.

图 3-2 FOTA 差分包

3、将生成的 XX.bin 文件压缩为 XX.zip 文件，XX.zip 文件就为未签名前的差分包

3.2. 差分包签名

1、获取华为离线签名工具（signtool）。（到华为开发者中心下载）

signtool.zip

2、 打开签名工具进行差分包签名，可参考《Lierda NB Module V150_FOTA_用户使用指导》手册的 3.1 小节。

Lierda NB Module V150_FOT

3.3. 上传签名后的差分包到 IOT 平台

可参考《Lierda NB Module V150_FOTA_用户使用指导》手册的 3.2 小节。

3.4. 创建升级任务

可参考《Lierda NB Module V150_FOTA_用户使用指导》手册的 3.3 小节。

4. OpenCPU FOTA 规避注意事项

4.1. 概述

NB86-G OpenCPU 方案 FOTA 升级以差分的形式进行升级：

- 在同版本（B300SP5 至 B300SP5）的升级时，客户的应用代码在应用核内，FOTA 升级只改变应用核中改变的地方；
- 在跨版本（B300SP3 至 B300SP5）进行 FOTA 升级时，这个时候就可能改变应用核、安全核、协议核里改变的部分。

- FOTA 差分包存储在 180KB 的独立内存中，和应用核代码区无关，所以在升级的差分包大小不能大于 180KB。
- 在 FOTA 升级时，平台虽然创建了升级任务，但并没有下发到模组，只有到模组上报一包数据到平台后，平台觉得终端此时是在线状态，平台才会下发 FOTA 任务给终端。
- 终端在进行 FOTA 升级时不能做发数据业务，此时客户需要做规避，可以使用事件状态 API 接口查询此时终端的状态，若处于 FOTA 状态则进行规避即可。

4.2. 事件状态接口介绍

4.2.1. 事件状态更新函数

函数原型：void lierda_module_status_read(void)

此函数用于更新事件状态结构体内事件的状态，在查询事件状态前需调用此函数进行事件状态更新。具体说明请参考《Lierda NB-IoT 模组 OpenCPU API 使用文档》。

4.2.2. 事件状态结构体

```
typedef struct
{
    char    charCGATT[AT_MAX_STRING_LEN];//模组附着网络状态
    char    charCSCON[AT_MAX_STRING_LEN];//模组连接基站状态，设置 AT+CSCON=1 才能生效
    char    charCEREG[AT_MAX_STRING_LEN];//模组连接核心网状态，设置 AT+CEREG=1 才能生效
    char    charCPSMS[AT_MAX_STRING_LEN];//PSM 模式指示
    char    charNPSMR[AT_MAX_STRING_LEN];//模组是否进入 PSM 状态，设置 AT+NPSMR=1 之后才能生效
    char    charNMSTATUS[AT_PLUS_MAX_STRING_LEN];// LWM2M 注册状态
    char    charFOTASTATUS[AT_PLUS_MAX_STRING_LEN];//FOTA 状态
    char    charSockNum[AT_PLUS_MAX_STRING_LEN];//TCP 创建的 Socket 状态，建立连接 TCP 服务器之后才能生效
}lierdaModStatus;
```

4.2.3.示例

查询各事件状态，然后进行打印

```
#include "lierda_module_status.h"


lierda_module_status_read();
lierdaLog("STATUS: %s\r\n %s\r\n %s\r\n%s ", module_status.charCGATT,
module_status.charCSCON,module_status.charCEREG, module_status. charFOTASTATUS);
```

4.3. FOTA 事件状态获取函数

4.3.1.函数原型

```
lierda_fotaSta lierda_FotaStatus(void);
* @参数    param： 空参数
* @返回值   FOTA 状态，枚举变量
typedef enum
{
    FOTA_NON=0,//无 FOTA 状态
    FOTA_DOWNLOADING,//FOTA 下载中
    FOTA_DOWNLOAD_FAILED,//FOTA 下载失败
    FOTA_DOWNLOADED,//FOTA 下载结束
    FOTA_UPDATING,//FOTA 加载中
    FOTA_UPDATE_SUCCESS,//FOTA 升级成功
    FOTA_UPDATE_FAILED,//FOTA 升级失败
    FOTA_UPDATE_OVER,//FOTA 升级结束
}lierda_fotaSta;
```

4.3.2.FOTA 事件状态获取函数说明

此函数利于事件状态接口更新来封装的函数，客户可直接采用 lierda_FotaStatus()此函数来查询 FOTA 的状态，也可自己利用事件状态接口来封装自己需要的 FOTA 状态接口。

lierda_FotaStatus()函数源码如下：

```
lierda_fotaSta lierda_FotaStatus(void)
```

```
{
    lierda_module_status_read();//更新事件状态
    if(strstr(module_status.charFOTASTATUS,"FIRMWARE")==NULL)
        return FOTA_NON;
    else if(strstr(module_status.charFOTASTATUS,"FIRMWARE DOWNLOADING")!=NULL)
        return FOTA_DOWNLOADING;
    else if(strstr(module_status.charFOTASTATUS,"FIRMWARE DOWNLOAD FAILED")!=NULL)
        return FOTA_DOWNLOAD_FAILED;
    else if(strstr(module_status.charFOTASTATUS,"FIRMWARE DOWNLOADED")!=NULL)
        return FOTA_DOWNLOADED;
    else if(strstr(module_status.charFOTASTATUS,"FIRMWARE UPDATING")!=NULL)
        return FOTA_UPDATING;
    else if(strstr(module_status.charFOTASTATUS,"FIRMWARE UPDATE SUCCESS")!=NULL)
        return FOTA_UPDATE_SUCCESS;
    else if(strstr(module_status.charFOTASTATUS,"FIRMWARE UPDATE FAILED")!=NULL)
        return FOTA_UPDATE_FAILED;
    else if(strstr(module_status.charFOTASTATUS,"FIRMWARE UPDATE OVER")!=NULL)
        return FOTA_UPDATE_OVER;
    return FOTA_NON;
}
```

4.3.3. 测试代码

```
    switch(lierda_FotaStatus())//获取FOTA状态
    {
    case FOTA_NON://此时不处于FOTA状态
        lierdaLog("DBG_INFO:无FOTA事件，可以发数据");
        lierdaSendMsgToPlatform((uint8 *) "30313233343536373839",
10,MSG_NON_NORAI, 0, 0);  //发送数据(ASCII)
       lierdaSendMsgToPlatform((uint8 *) "1234567890", 10, MSG_NON_NORAI, 0,1);
//发送数据(原数据)
        break;
    case FOTA_UPDATE_OVER://FOTA
        lierdaLog("DBG_INFO:FOTA事件结束，可以发数据");
        lierdaSendMsgToPlatform((uint8 *) "30313233343536373839",
```

```
10,MSG_NON_NORAI, 0, 0);  //发送数据(ASCII)
         lierdaSendMsgToPlatform((uint8 *) "1234567890", 10, MSG_NON_NORAI, 0,1);
//发送数据(原数据)
          break;
       default:lierdaLog("DBG_INFO:FOTA ING.......");break;
          }
```

4.4. FOTA 过程中能否数据发送判断函数

4.4.1. 函数原型

lierdaFota lierda_FotaEnableData(void);

参数 param：空参数

返回值 LierdaFota_DataEnable：可以做发数据做业务 LierdaFota_DataDisable：不可以发数据做业务

此 API 为了方便开发者在有 FOTA 需求时，规避 FOTA 中不能发数据的 API 接口，建议在发数据前调用此 API 判断是否可以做业务。

4.4.2. 测试代码

```
lierda_module_status_read();//更新此时事件状态
   if (strstr(module_status.charNMSTATUS,"MO_DATA_ENABLED")!=NULL)//判断LWM2M是
否注册成功
   {
      if(lierda_FotaEnableData()==LierdaFota_DataEnable)
      {
          lierdaLog("DBG_INFO:无FOTA事件或FOTA事件结束，可以发数据");
          lierdaSendMsgToPlatform((uint8 *) "30313233343536373839",
10,MSG_NON_NORAI, 0, 0);  //发送数据(ASCII)
         lierdaSendMsgToPlatform((uint8 *) "1234567890", 10, MSG_NON_NORAI, 0,1);
//发送数据(原数据)
      }
      else
      {
          lierdaLog("DBG_INFO:正在FOTA中，不可以发数据");
      }
   }
   else
          lierdaLog("DBG_INFO:LWM2M 服务器未注册");
```

4.5. OpenCPU FOTA 规避软件设计指导

4.5.1. FOTA 事件状态

FOTA 任务下发后，FOTA 事件状态有 5 个状态：

FIRMWARE DOWNLOADING：FOTA 差分包下载中

FIRMWARE DOWNLOADED：差分校验恢复

FIRMWARE UPDATE SUCCESS：FOTA 升级成功

FIRMWARE UPDATE FAILED：FOTA 升级失败

FIRMWARE UPDATE OVER：FOTA 升级结束

终端无论处在 PSM、DRX 还是 EDRX 模式，在发数据前进行事件状态更新后对 FOTA 状态进行查询，若终端此时处在 FOTA 状态则不能进行数据发送。

对于 FOTA 重启后什么时候可以发数据做业务，建议检测到 FIRMWARE UPDATE OVER 后再进行业务数据的发送。

客户可以利用 4.3 小节和 4.4 小节的 API 函数来进行 FOTA 规避，也可以利用 4.2 小节的事件状态接口根据自己业务需求来做 FOTA 规避。

4.5.2. 示例

```
static void Lwm2m_sendData(void)
{
    lierda_module_status_read();//更新此时事件状态
    if (strstr(module_status.charNMSTATUS,"MO_DATA_ENABLED")!=NULL)//判断LWM2M是否注册成功
    {
        if(lierda_FotaEnableData()==LierdaFota_DataEnable)
        {
            lierdaLog("DBG_INFO:无FOTA事件或FOTA事件结束，可以发数据");
            lierdaSendMsgToPlatform((uint8 *) "30313233343536373839", 10,MSG_NON_NORAI, 0, 0);  //发送数据(ASCII)
            lierdaSendMsgToPlatform((uint8 *) "1234567890", 10, MSG_NON_NORAI, 0,1); //发送数据(原数据)
        }
        else
        {
            lierdaLog("DBG_INFO:正在FOTA中，不可以发数据");
        }
    }
    else
```

```
    lierdaLog("DBG_INFO:LWM2M服务器未注册");
}
```

测试 log

5. FOTA 注意事项

- FOTA 过程中，终端不可断电，不可做复位操作。
- 生成的差分包大小不能超过 180KB。
- FOTA 过程中不能进行发送数据业务。
- FOTA 过程中，不能发 “AT+NRB”重启指令。
- 注意带 RA 快速释放发数据的方式，若使用带 RA 的指令发数据，而且 IDLE 态时间较短，若此时网络状态不是很好就很有可能收不到平台下发 observe /5/0/3 （订阅升级状态），就触发不了 FOTA 任务。

6. 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 6-1 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	