应用文档:

泰凌 BLE Mesh OTA 指南

AN-16112901-C1

Ver 1.0.0

2018/8/17

简介:

本文档主要介绍泰凌私有 BLE Mesh OTA 的流程、OTA 所需要的理论时间计算，以及开发接口。

TELINK SEMICONDUCTOR

Published by
Telink Semiconductor

Bldg 3, 1500 Zuchongzhi Rd,
Zhangjiang Hi-Tech Park, Shanghai, China

Legal Disclaimer

Telink Semiconductor reserves the right to make changes without further notice to any products herein to improve reliability, function or design. Telink Semiconductor disclaims any and all liability for any errors, inaccuracies or incompleteness contained herein or in any other disclosure relating to any product.

Telink Semiconductor does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights, nor the rights of others

The products shown herein are not designed for use in medical, life-saving, or life-sustaining applications. Customers using or selling Telink Semiconductor products not expressly indicated for use in such applications do so entirely at their own risk and agree to fully indemnify Telink Semiconductor for any damages arising or resulting from such use or sale.

Information:

For further information on the technology, product and business term, please contact Telink Semiconductor Company (www.telink-semi.com).

For sales or technical support, please send email to the address of:

telinkcnsales@telink-semi.com

telinkcnsupport@telink-semi.com

版本历史

版本	主要改动	日期	作者
1.0.0	初始版本	2018/8	SQF, Cynthia

目录

图目录

1 概述

当直连的灯节点通过 APP 更新固件时，它可以启动 Mesh OTA，从而将固件中继传递给同一 Mesh 网络中的其它节点。可以支持多达 255 个节点的同步 OTA 固件升级。

本文档旨在给用户提供泰凌 BLE Mesh OTA 的指导说明，可适用于基于 8266/8267/8269/8258（进行中）芯片的 BLE Mesh 方案。

2 Mesh OTA 流程

1. 连接 APP 与 Mesh 内的任意一个灯，后续用 L1 来表示该灯。读取所有节点的版本号（可以选择在 OTP 之前读取，或者仅在 OTA 完成之后再读取）：

   通过 OTA_ReadVer =11 23 34 00 00 ff ff c7 11 02 10 00 来获取。

   - 11 23 34：序列号（SNO）。每次发送命令后，SNO 会变化。
   - 00 00：源地址（src address）
   - ff ff：目的地址（dest address）
   - c7：Mesh OTA 命令的操作码
   - 11 02：厂商代码（vendor id ）
   - 10：转发次数
   - 00：子命令，表示读取版本信息

   版本号 notify 返回的数据如下：

   - c8 11 02 00 56 31 2e 32 00 00 00 00 00
   - c8：与 Mesh OTA 命令（c7）对应的 Notify 响应命令的操作码
   - 11 02：厂商代码（vendor id ）
   - 00：子命令，表示后续的内容是版本信息
   - 56 31 2e 32：版本号
   - 00 00 00 00 00：预留

2. 获取 L1 OTA 状态，如果获取的 OTA 状态非 0，表示有另外的 Master 正在发起 Mesh OTA。此时不能开启新的 Mesh OTA。

   通过以下指令获取： 12 23 34 00 00 00 00 c7 11 02 10 05

   指令数据格式如下：

   - 12 23 34：序列号（SNO）
   - 00 00：源地址（src address）
   - 00 00：目的地址（dest address）
   - c7：Mesh OTA 命令的操作码
   - 11 02：厂商代码（vendor id ）
   - 10：转发次数
   - 05：子命令，表示获取 OTA 状态

   OTA 状态信息会通过 notify 上报：C8 11 02 05 00

   Notify 数据格式如下：

   - c8：与 Mesh OTA 命令（c7）对应的 Notify 响应命令的操作码
   - 11 02：厂商代码（vendor id ）
   - 05: 子命令，表示后续的内容是 OTA 状态
   - 00: 表示 OTA state

     Idle: 00

     Slave: 01

     Master: 02

     Only relay: 03

     Complete: 04

3. 设置 L1 OTA mode 为 MeshOTA 或 GATT OTA，并设置设备类型。

   可通过以下指令设置：

   13 23 34 00 00 00 00 c7 11 02 10 06 01 04 00

   命令格式如下：

   - 13 23 34：序列号（SNO）
   - 00 00：源地址（src address）
   - 00 00：目的地址（dest address）
   - c7：Mesh OTA 命令的操作码
   - 11 02：厂商代码（vendor id ）
   - 10：转发次数
   - 06: 子命令，表示设置 OTA 模式（OTA mode）与设备类型（Device type）。
   - 01：设置 OTA 模式为 00: GATT OTA 或 01: MeshOTA
   - 04 00：设置设备类型为 0x0004（GATT OTA 模式请忽略此字段）

   OTA mode response 会通过 notify 上报，数据如下：

   c8 11 02 06 00

   命令格式如下：

   - c8：与 Mesh OTA 命令（c7）对应的 Notify 响应命令的操作码
   - 11 02：厂商代码（vendor id ）
   - 06：子命令，表示 OTA 模式（OTA mode）与设备类型（Device type）设置的结果。
   - 00：设置 OK

     非 00，或者没有收到 response 回复（旧版本），则表示设置不成功，即 BLE 直连节点不支持 Mesh OTA。

4. Master 发起第一阶段的 GATT OTA FLOW。

5. 完成第一阶段的 GATT OTA flow 后，BLE 直连节点 L1 会自动发起 CMD-OTA_Start 命令后，调用 mesh_ota_master_start_firmware_from_own(); 开始把本身的 firmware 通过 mesh 进行 OTA。

   Mesh OTA 的第一个包包含版本信息和设备类型信息，其它 mesh 节点收到这个包后，会先通过 **mesh_ota_slave_need_ota()**函数进行比较，如果符合条件（目前是要求设备类型相同，且版本号要大于当前版本），则接收后续的 OTA 数据，否则不接收 OTA 数据。

   L1 在发送 OTA 数据的过程中，会上报当前 OTA 进度信息给 APP，进度每增加 1%就上报一次。上报格式如下：

   c8 11 02 04 05

   - c8：OTA read (c7)的 notify response 命令码
   - 11 02：厂商代码（vendor id ）
   - 04：表示后续的内容是进度信息
   - 05：表示 5%的进度，发送 firmware 数据最大上报的进度是 95%。96%~99%是后续阶段使用。

6. 当 L1 上报给 APP 99%的进度信息后，APP 开始接收到每一个 Mesh 节点的 OTA 是否成功的 notify 信息。格式如下：

   - c8 11 02 03 00
   - c8：OTA read (C7)的 notify response 命令码
   - 11 02：厂商代码（vendor id ）
   - 03：表示后续的内容是校验值结果
   - 00：表示校验成功；另外 01 表示校验失败（也即 OTA 失败）

   另外，Mesh 节点接收完所有的 OTA 数据且比较校验结果完成后，如果成功，则慢闪 6 秒，如果失败则快闪 6 秒。不管是否 ok，均进行 OTA 重启。

7. 直连节点 L1 自动 reboot（直连节点根据 firmware 类型-节点的产品类型-是否一致来决定生效/擦除新固件），待重启完成后，APP 重连 L1 节点，并读取 OTA 状态，参考步骤 2，如果返回值是 04，表示 OTA 成功。

8. 对所有节点的版本号进行读取，并再次比对，确认是否升级 OTA，读取方式参考步骤 1。

   如果有某个节点没有成功，则需要重复整个 OTA 过程。在重新进行的 OTA 过程中，之前已经 OTA 成功的节点会忽略本次 OTA 过程，因为此时版本号比较已经不符合版本号判断条件。

9. 在 Mesh OTA 过程中，APP 可以发送命令来终止当前 OTA 进程。

   CMD-MeshOTA_Stop = ff ff c6 11 02 FE FF

   注意，目的地址是 0xFFFF，当 OTA 发送端收到该命令时，终止当前 OTA，不再发送 OTA 数据，且把 OTA 参数初始化。当 OTA 接收端收到该命令时，则进行重启，清空 OTA 接收区。

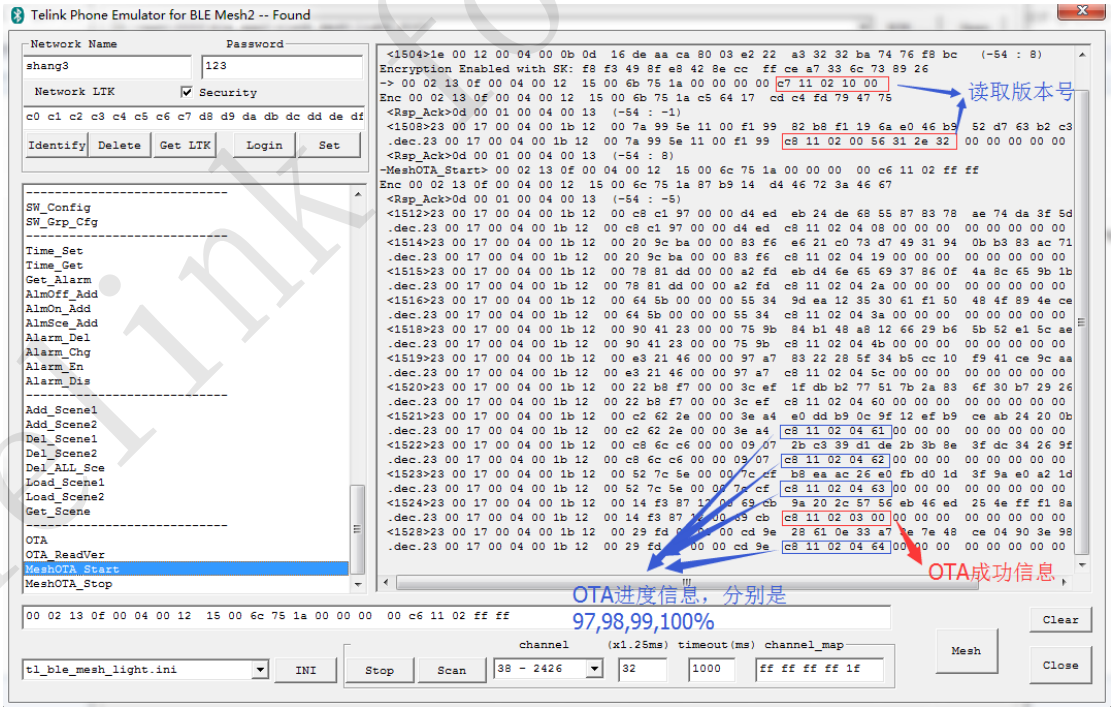

图 1　APP 的 Mesh OTA 命令

3 OTA 时间

理论上一次 Mesh OTA 过程所需时间为(firmware size / 13) * (0.04*4) + 10（单位：秒）。

例如：若 firmware size 为 48kB，则 OTA 所需时间为 614 秒，即 10.2 分钟。

4 开发接口

1） int mesh_ota_slave_need_ota(u8 *params) //版本号的比较函数，用户需要根据自己的版本号定义进行相应更改。

功能：该函数用于检查版本号和设备类型。目前的判断是，当设备类型一致，且版本号要高于当前版本时，才开始接收 OTA。详看函数的实现过程。

2） mesh_ota_master_start(u8 *adr, u32 len, u8 dev_mode)　　// 库函数

功能：该函数用于设置 Mesh OTA 启动信息。

- ✧ adr：待 OTA 的 firmware 的首地址
- ✧ len：待 OTA 的 firmware 的长度
- ✧ mode: 待 OTA 的 firmware 的设备类型，即对应的 LIGHT_MODE 的值

3） mesh_ota_master_start_firmware_from_own();

功能：该函数用于启动 Mesh OTA，将直连节点的固件数据传递给同一 Mesh 网络中的其他节点。

4） mesh_ota_master_start_firmware_from_21000();

功能：该函数用于启动 Mesh OTA，将待 OTA 的固件数据存放在 flash 0x21000。可用于对 Gateway 节点进行 OTA。

5） mesh_ota_slave_save_data(); //为库函数，用户可不关心此函数

当 Gateway 或者某个直连节点启动 Mesh OTA 后，会以 Mesh 的形式发送 OTA 数据。别的 Slave 节点收到这个数据后，会进入 mesh_ota_slave_save_data()函数进行处理。

当 Slave 节点没有收到启动命令或者版本号比较不符合时，也即 Slave 没有处于 OTA 状态，但后续又收到了 OTA 数据的命令，此时会返回非零值，并且此时可以认为 mesh_ota_slave_save_data()直接丢弃了该数据。比如当 Slave 在 Gateway Mesh OTA 开始后一段时间才上电时，也会出现这种情况。

另外，如果在往 flash 写 OTA 数据之后回读数据发现出错的时候，也会返回非零值。

其他情况返回 0。

6） mesh_ota_slave_set_response() //为库函数，用户可不关心此函数

响应 LGT_CMD_MESH_OTA_READ 命令，用户需要关注校验结果的反馈，以及 OTA 进度的反馈(自动反馈)，详情参考“Mesh OTA 流程”章节。