BEKEN-Mesh使用手册

Beken Corporation
Building 41, Capital of Tech Leaders, 1387 Zhangdong Road,
Zhangjiang High-Tech Park, Pudong New District, Shanghai, China
Tel： (86)21 51086811
Fax： (86)21 60871089

目录

版本	发布/更新日期	更新人员	重要变更内容
V1.0.0	2020/09/16	关文瑞	初始版本

1. 目的

本文档的目的是简要阐述基于博通BK系列芯片开发MESH工程的软件结构，便于初开发人员加快对工程的理解以及方便添加自定制的相关功能。

2. 工程结构划分

完整的BKXXXX-AliOS工程划分为三个基本项目，每个项目有单独的代码工程进行编辑与编译：

1. Project-Boot：芯片初始化流程。该部分包含芯片上电之后一系列流程，此外也会执行OTA升级重启后的一些Flash数据转移动作。

2. Project-Controller：博通蓝牙协议栈Controller工程。该部分融合了蓝牙Controller层协议以及芯片内部资源交互，为上层APP执行蓝牙功能提供标准HCI指令接口。

3. Ali-OS：蓝牙Host协议及BLE Mesh Stack和相关应用。该部分代码源自于AliGenie提供的开源软件代码，由博通蓝牙研发组进行修改和优化。关于源码的引用以及开源协议，可参考alibaba官方github地址：
   https://github.com/alibaba/genie-bt-mesh-stack

三个工程在各自的运行环境下单独调试单独编译，最终按照预划分Flash结构统一整合成一个镜像文件。

3. 运行环境

3.1. 平台代码

博通提供的基础平台代码为Ali-OS的运行给予软件及硬件支持。该部分代码以芯片版本名称命名，存放路径：./platform/mcu/bkxxxx

基础平台为操作系统提供了软件功能接口以及平台驱动抽象层，基础工程以镜像文件形式存放于平台目录下bin文件夹内，该文件夹包含的批处理文件在应用文件编译时自动执行最终的镜像文件整合工作。

与芯片相关的板级配置位于：./board/bkxxxxdevkit

3.2. Ali-OS 代码

工程的主体部分为AliGenie开源的操作系统及蓝牙MESH协议。此代码工程运行开发环境VS Code，程序扩展有阿里官方的插件alios-studio，基于gcc编译器使用的插件工具可以方便的完成代码调试工作。使用说明可参考：

https://github.com/alibaba/AliOS-Things/wiki/AliOS-Things-Studio

4. 用户自定义

芯片平台部分为接口代码和编译完好的镜像文件，由博通进行修订和发布，暂不支持用户自定义，有相关问题可以咨询开发人员。

上层代码源于Ali-OS开源工程，其融合了操作系统、BLE Host、MESH Stack以及应用APP和其他辅助模块，目前官方仍处于开发中。在开源版本基础之上，博通进行自有工程的增补和修正，使其对BK芯片平台更加友好，并完成了通用的客户需求。工程代码用户可见，版本的更新和发布信息可咨询开发人员。在此工程结构下，用户可以更多的关注上层APP，以减少开发复杂度和缩短开发周期。

目前已发布的版本中包含一个完整的Mesh应用代码示例，位于目录./app/example/bluetooth/light，是一个三色灯控的程序，用户可以依照示例代码编写自己的应用工程，或添加自定义协议。

5. 示例程序

Light工程是一个Mesh网络节点的示例，是一个具有基本组网和控制功能的三色灯。该工程有两个主要文件，一个是组合节点element和所包含的model的文件light.c，另一个是用以配置板级LED输出逻辑的文件light_board.c。

在数组elements[ ]内，放入该节点包含的全部元素：

```
struct bt_mesh_elem elements[] = {
    BT_MESH_ELEM(0, root_models, vnd_models, 0),
    BT_MESH_ELEM(0, s0_models, BT_MESH_MODEL_NONE, 0),
    BT_MESH_ELEM(0, s1_models, BT_MESH_MODEL_NONE, 0),
};
```

在每个元素内中填写所包含的model：

```
struct bt_mesh_model s1_models[] = {
    MESH_MODEL_GEN_LEVEL_SRV(&g_elem_state[0]),
    MESH_MODEL_HSL_SAT_SRV(&g_elem_state[0]),
};
```

其中由蓝牙SIG定义的标准模型，可以在目录genie_app/Bluetooth/mesh/mesh_model下，查找已添加 的部分。用户亦可参阅

Mesh Model Bluetooth Specification补充相应功能和所需要的model，此外可以依照格式自定义vendor model。

在状态结构体S_ELEM_STATE中存放了全部model所涉及的state，变量g_elem_state来完成状态的临时存储和切换；g_powerup作为flash存储内容的一个暂存，用以查询当前状态和存储状态的对比。

```
S_ELEM_STATE g_elem_state[MESH_ELEM_STATE_COUNT];
S_MODEL_POWERUP g_powerup[MESH_ELEM_STATE_COUNT];
```

Application_start()是系统成功初始化后向应用层的一个调用，在这里完成genie的初始化，以及LED的初上电设置。

```
int application_start(int argc, char **argv)
{
    led_startup();

    /* genie initilize */
    genie_init();

    BT_INFO("BUILD_TIME:%s", __DATE__","__TIME__);

    return 0;
}
```

led_startup()中首先完成了LED灯驱动的加载，在函数_init_light_para()中完成各项state的初始化，会对flash存储的历史状态数据进行读取和恢复。

_user_init()，是用户层的初始化，此处定义为mesh model之上的更高私有协议机制，由用户自行添加。当前函数中定义了model状态延时存储的定时器。

user_event() 是用户层对于各项GENIE EVENT处理的接口，每一项event在genie层面执行通用处理之后进入该级。用户可在该层制定相应的动作，如灯的控制逻辑或者私有协议交互。协议主控制流程状态机已经完成，例如用户态初始化GENIE_EVT_SDK_MESH_INIT的启动，mesh消息传递至LED的控制处理入口GENIE_EVT_SDK_ACTION_DONE。用户可以根据状态时序自定义每个event下的执行动作。

```
void user_event(E_GENIE_EVENT event, void *p_arg)
{
    E_GENIE_EVENT next_event = event;

    //BT_DBG_R("%s, %s %p\n", __func__, genie_event_str[event], p_arg
    switch(event) {
        case GENIE_EVT_SW_RESET:
        case GENIE_EVT_HW_RESET_START:
            BT_DBG_R("FLASH x5");      Yulong, 2 months ago • Add the
            _led_flash(5, 1);
            break;
        case GENIE_EVT_HW_RESET_DONE:
```

_led_ctrl() 是一个示例LED灯的控制逻辑，触发于GENIE_EVT_SDK_ACTION_DONE的event。该事件由每项model处理完毕接收到的指令后发起。在_led_ctrl()函数内，通过分析消息状态的设置，来执行相应的控制操作。根据model message的内容，执行相应的参数分析及数据转化，最后分解成RGB三通道的参数，传入PWM控制端口。

```
static void _led_ctrl(S_ELEM_STATE *p_elem)
```

关于灯控参数的解析，代码中给出了色温和HSL两种示例。

```
void led_ctl_set_handler(uint16_t ctl_lightness, uint16_t temperature, uint16_t ctl_UV)
```

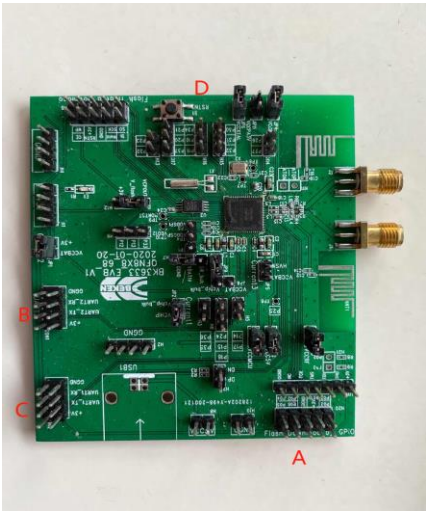
```
void led_hsl_set_handler(uint16_t hue, uint16_t saturation, uint16_t lightness)
```

经过一系列计算，最终执行PWM控制。

```
static void _light_lighten(light_channel_e channel, uint16_t state)
```

6. 开发板及代码工程使用

6.1. 主板布局及镜像文件烧录

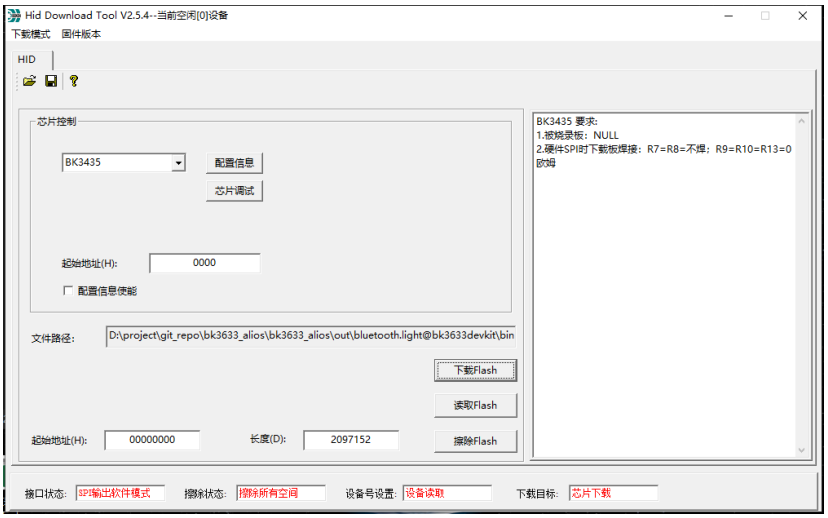


图示为博通提供的镶嵌BKXXXX系列BLE芯片的开发板，其详细制版结构和芯片手册请另外咨询获取。如图中标记提示，A作为程序下载端口，使用博通提供的专用下载工具；B、C为UART输出，示例代码中默认使用B所代表的UART2打印串口Log；D为接引管脚，用户可用于调试或外接LED灯来查看程序效果，如示例

工程Light中控制的三色灯R、G、B三个管脚分别为P10、P11、P12。

使用VS Code环境下alios-studio提供的编译工具，图标或命令行均可，选定目标工程之后执行编译。

连接好下载端口之后，使用烧写工具下载。



6.2. 修改调试参数

在mesh协议代码中，每一层都单独添加了调试日志开关，可以在对应.c文件的开头找到需要使能的变量，使能操作一般执行在相同目录或应用工程目录下的.mk的文件中，用户和根据使用情况自行开关。

```
#define BT_DBG_ENABLED IS_ENABLED(CONFIG_BT_MESH_DEBUG_MODEL)
```

```
GLOBAL_DEFINES += CONFIG_BT_MESH_DEBUG_FLASH
GLOBAL_DEFINES += CONFIG_BT_MESH_DEBUG_EVENT
GLOBAL_DEFINES += CONFIG_BT_MESH_DEBUG_MODEL
```

在使能了相应的开关后，连接调试串口并上电，可以看到相应输出。预设的串口输出波特率为1000000.

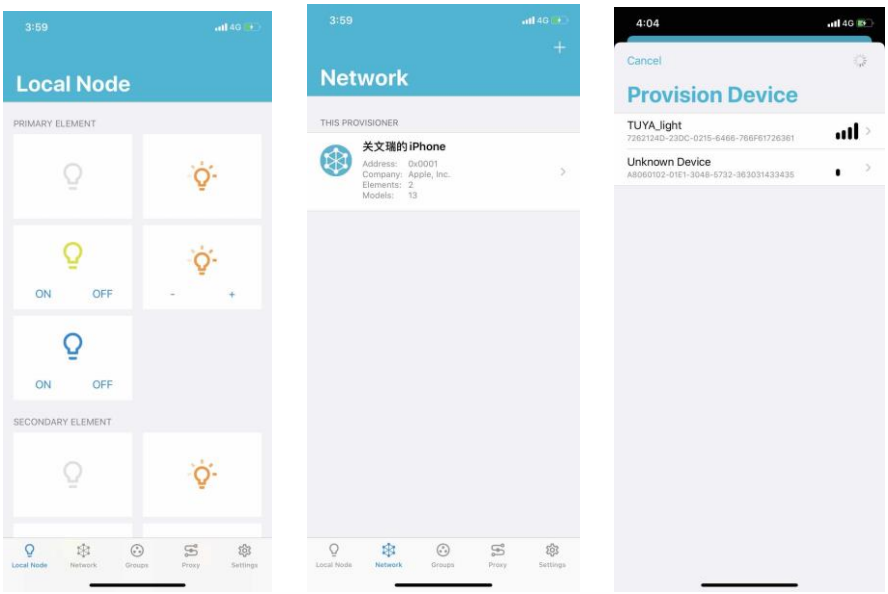
```
[15:31:53.758] flash #2, /2, size:512KB
[15:31:53.981] code area end addr:0x48858
[15:31:53.993] flash_init end, id 0x514013
[15:31:53.993] soc_driver init ok
[15:31:53.993]
[15:31:53.993] start sys_init
[15:31:53.993]
[15:31:53.993] sys_init
[15:31:53.993] soc_system_init
[15:31:53.993] func_init
[15:31:53.993] rom_env_init L 40
[15:31:53.993] func_init
[15:31:53.993] [FUNC]intc_init
[15:31:53.993] [FUNC]func_init OVER!!!
[15:31:53.993]
[15:31:53.993] The APP code build at 10:42:22, Sep 16 2020
[15:31:53.993]
[15:31:53.993] trace should have cli to control!!!
[15:31:53.993] app start
```

在tri_tuple_default.h文件下定义宏DEFAULT_MAC为设备的蓝牙地址。设备名称可以通过light.mk下的CONFIG_BT_DEVICE_NAME变量修改。

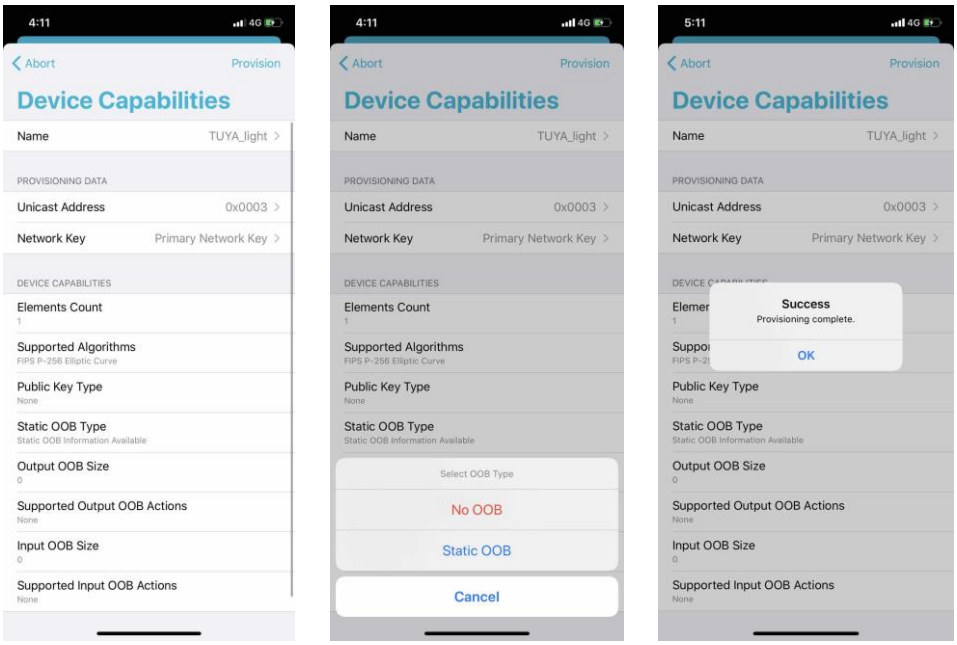
```
GLOBAL_DEFINES += CONFIG_BT_DEVICE_NAME=\"TUYA_light\"
```

6.3. 示例工程演示

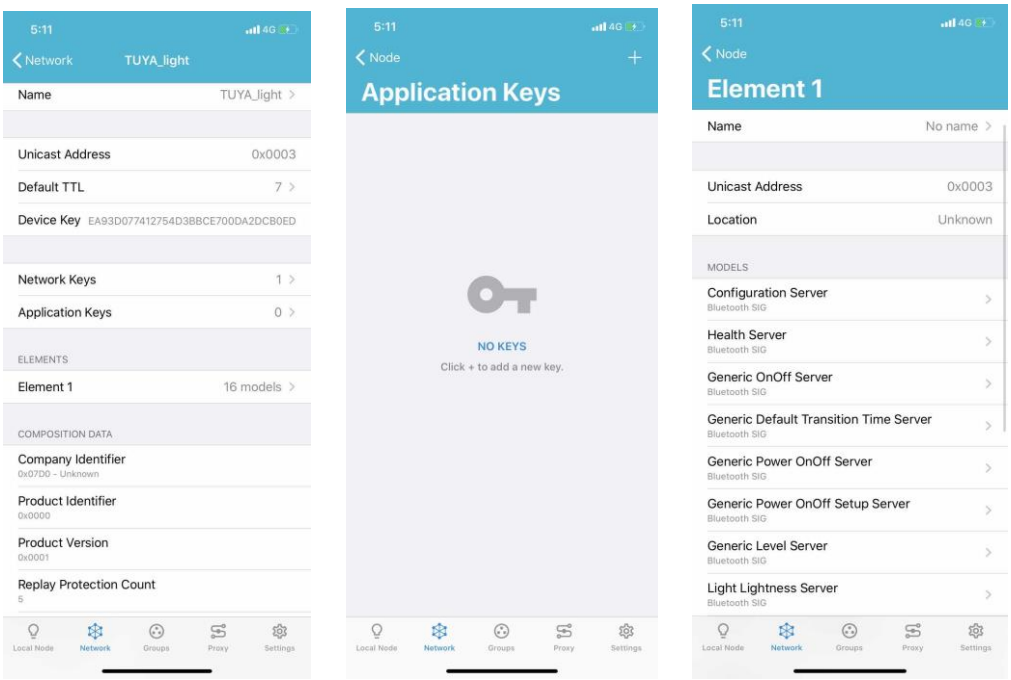
Light工程所提供的灯控效果，可以通过在开发板外接LED来展示。完成一个完整的演示流程，需要手机端应用的配合。手机端可以使用开源的应用nRF Mesh，iOS平台可以获取。工程内节点模型执行的标准SIG Mesh协议，用户也可使用其他适配的应用程序。



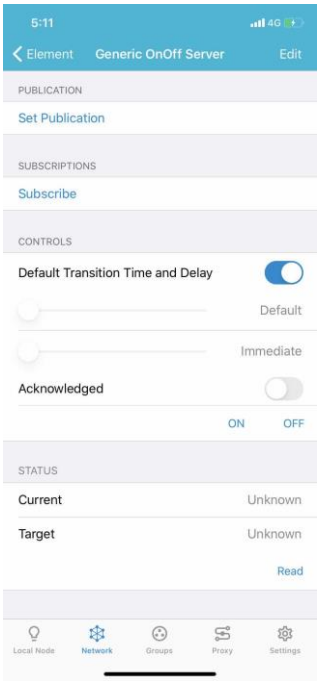
主界面下选择network界面，点击右上角的‘+’来开启广播扫描，在扫描到的设备列表中选择名称对应的目标设备，来进一步获取设备信息。



在设备信息界面点击右上角Provision，选择No OOB，进入Provision流程直到提示成功。



为节点添加Application Key，手机端预设Key的相关操作可以在setting中找到。添加之后，进入需要测试的model绑定该APP Key，然后可以执行该Model下的特殊操作。



图示为Generic OnOff Model下的专有功能，实现Mesh灯控开关以及相关的辅助操作。由于SIG Mesh中Model定义众多，控制协议复杂，nRF Mesh也没有完备的为所有Model提供功能按钮，目前BK代码只提供示例功能的Model协议，用户可根据需求参阅Specification自行添加，或自定制Vendor Model。