应用文档:

泰凌 BLE Mesh Light APP 通信协议

AN-BLE-15120203-C5

Ver 1.3.1

2016/6/3

简介:

本文档介绍了泰凌 BLE Mesh Light APP 的通信协议。

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 & 标准化
1.0.0	初始版本	2015/12	J.G.H. & Cynthia
1.1.0	更新 Kick_out 和 User_All 命令； 新增“User notify data”说明	2015/12	S.Q.F., J.G.H., & Cynthia
1.2.0	修订 Device_Addr 和 Get_Dev_Addr 命令； 文字修订	2016/2	J.G.H. & Cynthia
1.3.0	Get_Alarm/Get_Scene 命令增加指定 ID 读取的功能	2016/5	S.Q.F., Cynthia
1.3.1	进一步说明 Del_G1[12:13]	2016/6	S.Q.F., Cynthia

1 目录

2 表目录

1 UUID 部分

```
publicstatic String ServiceTelink =
"00010203-0405-0607-0809-0a0b0c0d1910";
publicstatic String MeshLightStatus =
"00010203-0405-0607-0809-0a0b0c0d1911";
publicstatic String MeshLightCommand  =
"00010203-0405-0607-0809-0a0b0c0d1912";
publicstatic String MeshLightOTA =
"00010203-0405-0607-0809-0a0b0c0d1913";
public static String MeshLightPair =
"00010203-0405-0607-0809-0a0b0c0d1914";
```

以下部分的数据格式样例均是在非加密模式下的数据。

2 MeshLightCommand（MeshLightCommand UUID 1912）

表 1 命令格式解析

NO.	1	2	3	4	5	6	7	8	9	10	11	12	13
data	**Sn0**	**Sn1**	**Sn2**	**src**	**src**	**dst**	**dst**	**cmd**	**VendorID**	**VendorID**	**Par**	**Par**	**Par**

[1:3](SN0-SN2)：sequence number。Sn0为低字节。每发一个命令sequence number加1，并保证sequence number不为0。

[4:5](Src)：含义为source addr（源地址）。对于APP来说固定为0x0000。

[6:7](dst)：含义为destination addr（目的地址）。取值范围是0x0000-0xFFFF。具体解析如下：

- ✧ 0x0000：表示local addr，只有和APP直接连接的灯才会响应该命令。
- ✧ 0x0001 ~ 0x00FF：表示device addr，用于单灯控制，只有对应的device address的灯才会响应该命令。
- ✧ 0x8000-0xfffe：表示group address，用于按组进行控制，只有对应的group address的灯才会响应该命令。
- ✧ 0xffff：表示广播地址，所有的灯收到都会响应该命令。

规则：dst[15]为0时，表示该地址为device addr；dst[15]为1时，表示该地址为group addr；（0x0000和0xffff除外）

对于light，如果device addr没有进行手动配置过，device addr默认等于MAC（例如FF:FF:33:18:d3:31）的最低byte，即0x0031（高byte为0x00）。如果MAC的最低byte为0，则device addr默认为0x0001。

Device addr分配的建议做法：

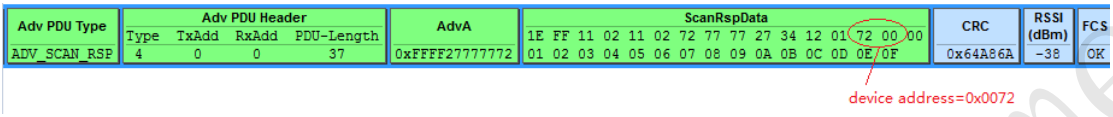
1) 使用默认值。
2) 但是这种情况会有重复的可能。解决方式：（一般在做APP的时候，都会有进行添加灯的动作）在添加灯的时候， APP就使用Device_Addr命令自动分配一个（0x0001-0x00FF），后续自动加1。这个应该是由APP自动完成，而不需要用户设置。

   另外在 adv的scan_response包中也包含有device addr的信息的，APP也可

以从那里获取，详见sdk的数据结构 ll_adv_rsp_private_t。

App获取device addr的方式：

1) 在广播包的scan response包中也包含有这个device addr，具体格式请参考SDK的adv_rsp_pri_data数据结构。



2) online status的包中获取。

3) 发送Get_Dev_Addr命令获取。

[8](cmd)：表示命令码。命令码格式默认bit6和bit7都为1。例如ALL_ON的cmd为0xd0。

[9:10](VendorID)：此处为vendor ID，默认为 0x11 0x02。

[8：10]统称为op code。

[11:13](Par)：表示参数区，par最多有10个字节，此处列出了3个。

命令示例说明如下。

2.1 ALL_ON

同时开灯。

S:发送

表 2 ALL_ON 命令示例

1	2	3	4	5	6	7	8	9	10	11	12	13
11	**11**	**11**	**00**	**00**	**ff**	**ff**	**d0**	**11**	**02**	**01**	**01**	**00**

VC例子：

[1:3]：0x111111，表示sequence no为0x111111。

[4:5]：0x0000，表示Src为0x0000，该值**不需要变动。**

[6:7]：0xffff，表示destination addr为0xffff，即所有的灯都会响应该命令。

[8]：0xd0，表示cmd为0xd0，为开关灯的命令码。

[9:10]：VendorID，默认为 0x11 0x02，该值**不需要变动**。

[11]：0x01表示开灯（第0个parameter的值）。

[12:13]：01 00表示延时1ms后开灯。解析如下：[12:13]表示第1到第2个parameter的值，这两个值组成一个16bit的数字，表示延时时间，单位为ms（注：[12]为低8位，[13]为高8位）。

R:接收

Write_rsp

2.2 ALL_OFF

同时关灯。

S:发送

表 3 ALL_OFF 命令示例

1	2	3	4	5	6	7	8	9	10	11	12	13
11	**11**	**12**	**00**	**00**	**ff**	**ff**	**d0**	**11**	**02**	**00**	**01**	**00**

VC例子：

[1:3]：0x121111，表示sequence no为0x121111。

[4:5]：0x0000，表示Src为0x0000，该值**不需要变动**。

[6:7]：0xffff，表示destination addr为0xffff，即所有的灯都会响应该命令。

[8]：0xd0，表示cmd为0xd0，为开关灯的命令码。

[9:10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11]：0x00表示关灯（第0个parameter的值）。

[12:13]：01 00表示延时1ms后关灯。解析如下：[12:13]表示第1到第2个parameter的值，这两个值组成一个16bit的数字，表示延时时间，单位为ms（注：[12]为低8位，[13]为高8位）。

R:接收

Write_rsp

2.3 ALL_ON_500MS

所有的灯，延时500ms后，统一开灯。目的是实现同时动作，消除因命令转发产生的延时，导致的不同步的现象。

S:发送

表 4 ALL_ON_500MS 命令示例

1	2	3	4	5	6	7	8	9	10	11	12	13
11	**11**	**11**	**00**	**00**	**ff**	**ff**	**d0**	**11**	**02**	**01**	**01**	**02**

VC例子：

[1:3]：0x111111，表示sequence no为0x111111。

[4:5]：0x0000，表示Src为0x0000，该值**不需要变动**。

[6:7]：0xffff，表示destination addr为0xffff，即所有的灯都会响应该命令。

[8]：0xd0，表示cmd为0xd0，为开关灯的命令码。

[9:10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11]：0x01表示开灯（第0个parameter的值）

[12:13]：0x0201表示延时513ms后开灯。解析如下：[12:13]表示第1到第2个parameter的值，这两个值组成一个16bit的数字，表示延时时间，单位为ms（注：[12]为低8位，[13]为高8位）。

R:接收

Write_rsp

2.4 ALL_OFF_500MS

所有的灯，延时500ms后，统一关灯。目的是实现同时动作，消除因命令转发产生的延时，导致的不同步的现象。

S:发送

表 5 ALL_OFF_500MS 命令示例

1	2	3	4	5	6	7	8	9	10	11	12	13
11	**11**	**12**	**00**	**00**	**ff**	**ff**	**d0**	**11**	**02**	**00**	**01**	**02**

VC例子：

[1:3]：0x121111，表示sequence no为0x121111。

[4:5]：0x0000，表示Src为0x0000，该值**不需要变动**。

[6:7]：0xffff，表示destination addr为0xffff，即所有的灯都会响应该命令。

[8]：0xd0，表示cmd为0xd0，为开关灯的命令码。

[9:10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11]：0x00表示关灯（第0个parameter的值）。

[12:13]：0x0201表示延时513ms后关灯。解析如下：[12:13]表示第1到第2个parameter的值，这两个值组成一个16bit的数字，表示延时时间，单位为ms（注：[12]为低8位，[13]为高8位）。

R:接收

Write_rsp

2.5 Set_Lum

设置所有的灯的亮度值。

S:发送

表 6 Set_Lum 命令示例

1	2	3	4	5	6	7	8	9	10	11
11	**11**	**13**	**00**	**00**	**00**	**00**	**D2**	**11**	**02**	**0A**

VC例子：

[1:3]：0x131111，表示sequence no为0x131111。

[4:5]：0x0000，表示Src为0x0000，该值**不需要变动**。

[6:7]：0x0000，表示destination addr为0x0000，即只对本地连接的灯响应该命令（和BLE直接连接的灯）。

[8]：0xd2，表示cmd为0xd2，为设置亮度值的命令码。

[9:10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11]：0x0A表示设置的亮度值为0x0A。

R:接收

Write_rsp

2.6 Music_Start

音乐功能流程：发送音乐数据的时候，先发送Music Start（用于提示light保存当前的灯的状态）；然后使用Set_Lum命令不断发送音乐的音量值(0~100)，当音乐播放结束的时候，发送Music Stop命令恢复灯的状态。

注：为了简化命令，Music Start、Music Stop的opcode和Set_Lum是一样的，只是用一个特殊的亮度表示。（Music Start使用0xfe，Music Stop使用0xff。）

Music Start命令，用于提示light保存当前的灯的状态，等待收到Music Stop命令后恢复灯的状态。

S:发送

表 7 Music_Start 命令示例

1	2	3	4	5	6	7	8	9	10	11
11	**11**	**16**	**00**	**00**	**00**	**00**	**D2**	**11**	**02**	**FE**

VC例子：

[1:3]：0x161111，表示sequence no为0x161111。

[4:5]：0x0000，表示Src为0x0000，该值**不需要变动**。

[6:7]：0x0000，表示destination addr为0x0000，即只对本地连接的灯响应该命令（和BLE直接连接的灯）。

[8]：0xd2，表示cmd为0xd2，为设置亮度值的命令码。

[9:10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11]：0xFE表示music start功能。

R:接收

Write_rsp

2.7 Music_Stop

收到Music Stop命令后恢复灯的状态。

S:发送

表 8 Music_Stop 命令示例

1	2	3	4	5	6	7	8	9	10	11
11	**11**	**17**	**00**	**00**	**00**	**00**	**D2**	**11**	**02**	**FF**

VC例子：

[1:3]：0x171111，表示sequence no为0x171111。

[4:5]：0x0000，表示Src为0x0000，该值**不需要变动**。

[6:7]：0x0000，表示destination addr为0x0000，即只对本地连接的灯响应该命令（和BLE直接连接的灯）。

[8]：0xd2，表示cmd为0xd2，为设置亮度值的命令码。

[9:10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11]：0xFF表示music stop功能。

R:接收

Write_rsp

2.8 Set_R

设置红灯的亮度值。

S:发送

表 9 Set_R 命令示例

1	2	3	4	5	6	7	8	9	10	11	12
11	**11**	**81**	**00**	**00**	**ff**	**ff**	**E2**	**11**	**02**	**01**	**00**

VC例子：

[1:3]：0x811111，表示sequence no为0x811111。

[4:5]：0x0000，表示Src为0x0000，该值**不需要变动**。

[6:7]：0xffff，表示destination addr为0xffff，即所有的灯都会响应该命令。

[8]:0xe2，表示cmd为0xe2，为设置单个灯的亮度值的命令码。

[9:10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11]：0x01表示设置的是红灯（第0个parameter的值）。

[12]：0x00表示颜色索引值为0。注：取值范围是0x00~0xff（即0~255）。

R:接收

Write_rsp

2.9 Set_G

设置绿灯的亮度值。

S:发送

表 10 Set_G 命令示例

1	2	3	4	5	6	7	8	9	10	11	12
11	**11**	**83**	**00**	**00**	**ff**	**ff**	**E2**	**11**	**02**	**02**	**00**

VC例子：

[1:3]：0x831111，表示sequence no为0x831111。

[4:5]：0x0000，表示Src为0x0000，该值**不需要变动**。

[6:7]：0xffff，表示destination addr为0xffff，即所有的灯都会响应该命令。

[8]：0xe2，表示cmd为0xe2，为设置单个灯的亮度值的命令码。

[9:10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11]：0x02表示设置的是绿灯（第0个parameter的值）。

[12]：0x00表示颜色索引值为0。注：取值范围是0x00~0xff（即0~255）

R:接收

Write_rsp

2.10 Set_B

设置蓝灯的亮度值。

S:发送

表 11 Set_B 命令示例

1	2	3	4	5	6	7	8	9	10	11	12
11	**11**	**85**	**00**	**00**	**ff**	**ff**	**E2**	**11**	**02**	**03**	**00**

VC例子：

[1:3]：0x851111，表示sequence no为0x851111。

[4:5]：0x0000，表示Src为0x0000，该值**不需要变动**。

[6:7]：0xffff，表示destination addr为0xffff，即所有的灯都会响应该命令。

[8]：0xe2，表示cmd为0xe2，为设置单个灯的亮度值的命令码。

[9:10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11]：0x03表示设置的是蓝灯（第0个parameter的值）。

[12]：0x00表示颜色索引值为0。注：取值范围是0x00~0xff（即0~255）。

R:接收

Write_rsp

2.11 Set_RGB

同时设置RGB三个灯的亮度值。

S:发送

表 12 Set_RGB 命令示例

1	2	3	4	5	6	7	8	9	10	11	12	13	14
11	**11**	**87**	**00**	**00**	**ff**	**ff**	**E2**	**11**	**02**	**04**	**70**	**90**	**b0**

VC例子：

[1:3]：0x871111，表示sequence no为0x871111。

[4:5]：0x0000，表示Src为0x0000，该值**不需要变动**。

[6:7]：0xffff，表示destination addr为0xffff，即所有的灯都会响应该命令。

[8]：0xe2，表示cmd为0xe2，为设置单个灯的亮度值的命令码。

[9:10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11]：0x04表示设置的是RGB三个灯（第0个parameter的值）。

[12:14]：分别表示R颜色索引值为0x70；G颜色索引值为0x90；B颜色索引值为0xB0。注：取值范围是0x00~0xff（即0~255）。

R:接收

Write_rsp

2.12 Set_CT

设置CT灯的CT值（百分比）。

S:发送

表 13 Set_CT 命令示例

1	2	3	4	5	6	7	8	9	10	11	12
11	**11**	**88**	**00**	**00**	**ff**	**ff**	**E2**	**11**	**02**	**05**	**00**

VC例子：

[1:3]：0x881111，表示sequence no为0x881111。

[4:5]：0x0000，表示Src为0x0000，该值**不需要变动**。

[6:7]：0xffff，表示destination addr为0xffff，即所有的灯都会响应该命令。

[8]：0xe2，表示cmd为0xe2，为设置单个灯的亮度值的命令码.

[9:10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11]：0x05表示设置的是CT值（第0个parameter的值）。

[12]：表示CT值的百分比为0%。注：取值范围是0x00~0x64（即0%~100%）。

R:接收

Write_rsp

2.13 Device_Addr

1） 更改设备地址（注意不是mac地址，是在mesh之间识别使用的地址，详细解析见“命令格式解析”）。

2） 获取一个组号里面所有的灯，或者手动去获取Device_Addr。(详见下一条命令Get_Dev_Addr)。

S:发送

表 14 Device_Addr 命令示例

1	2	3	4	5	6	7	8	9	10	11	12
11	**11**	**70**	**00**	**00**	**00**	**00**	**E0**	**11**	**02**	**11**	**00**

VC例子：

[1:3]：0x701111，表示sequence no为0x701111。

[4:5]：0x0000，表示Src为0x0000，该值**不需要变动**。

[6:7]：0x0000，表示destination addr为0x0000，即只对本地连接的灯响应该命令（和BLE直接连接的灯），表示将要操作对象是本地连接的灯。

设置地址时，此处的取值范围是0x0001~0x00FF（即单灯地址）。

[8]：0xe0，表示cmd为0xe0，为设置单个灯的device address的命令码。

[9:10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11:12]：0x0011表示将把对应的灯的device addr改为0x0011。

此处的取值范围是0x0001~ 0x00FF。设置完成后不需要手动去读，light会自动把当前地址通过notify的方式自动返回当前的device addr。APP可根据这个返回值判断是否设置成功。

R:接收

1. Write_rsp
2. 获取到的device addr的notify回复，数据格式如下（**notify所在的UUID为0x1911**）

1	2	3	4	5	6	7	8	9	10	11	12
11	**11**	**70**	**11**	**00**	**11**	**11**	**E1**	**11**	**02**	**11**	**00**
13	14	15	16	17	18	19	20				
00	**00**	**00**	**00**	**00**	**00**	**00**	**00**				

[1:3]：0x701111，表示sequence no。非加密情况下该值和对应的write命令的sno一致；加密情况下则是随机数。

[4:5]：0x0011，表示Src为0x0011，该值为对应的灯的device addr，因为已经设置device addr成功，所以返回的addr会启用新的。

[6:7]：0x1111，加密算法校验用。非加密情况下该值和[4：5]相同。

[8]：0xe1，表示cmd为0xe1，为device addr **notify数据**的标识符。

[9:10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11:12]：0x0011表示返回的该灯的device addr。

[13:20]：保留字节，此处全部设为0x00。

2.14 Get_Dev_Addr

获取一个组号里面所有的灯。或者手动去获取Device_Addr。

S:发送

表 15 Get_Dev_Addr 命令示例

1	2	3	4	5	6	7	8	9	10	11	12
11	**11**	**72**	**00**	**00**	**01**	**80**	**E0**	**11**	**02**	**ff**	**ff**

VC例子：

[1:3]：0x721111，表示sequence no为0x721111。

[4:5]：0x0000，表示Src为0x0000，该值**不需要变动**。

[6:7]：0x8001，表示destination addr为第一组，表示将要操作对象是一个组。

- ✧ 如果作为“获取一个组号里面所有的灯”，此处的取值范围是0x8000~0xffff（即组地址）。
- ✧ 如果作为“手动去获取Device_Addr”此处的取值范围是0x0001 ~ 0x00FF（即单灯地址）。

[8]：0xe0，表示cmd为0xe0，复用命令。

[9:10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11:12]：0xffff，规则约定此处必须为0xffff。

当命令发出后，会收到符合条件的所有灯的device addr的notify回复。

R:接收

1. Write_rsp
2. 获取到的device addr的notify回复，数据格式（notify所在的UUID为0x1911）请参考上一条Device_Addr命令。

2.15 Add_G1

添加组。

S:发送

表 16 Add_G1 命令示例

1	2	3	4	5	6	7	8	9	10	11	12	13
11	**11**	**21**	**00**	**00**	**00**	**00**	**D7**	**11**	**02**	**01**	**01**	**80**

VC例子：

[1:3]：0x211111，表示sequence no为0x211111。

[4:5]：0x0000，表示Src为0x0000，该值**不需要变动**。

[6:7]：0x0000，表示destination addr为0x0000，即只对本地连接的灯响应该命令（和BLE直接连接的灯），表示将要操作对象是本地连接的灯。

[8]：0xd7，表示cmd为0xd7，为**组操作**的命令码。

[9：10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11]：0x01表示添加组（第0个parameter的值）。

[12:13]：0x8001表示添加的组号为0x8001（即第一组，组号的识别规则及取值范围详细解析见“命令格式解析”）

当命令发出后，如果light能正确收到并响应，指示灯会0.5Hz慢闪3次，并且app会收到该light的所有group addr的notify回复。app可以根据这个回复来再次确认是否添加组成功。

R:接收

1. Write_rsp
2. 获取到的group addr的notify回复，数据格式请参考Get_G_8的notify回复。

2.16 Del_G1

删除组。

S:发送

表 17 Del_G1 命令示例

1	2	3	4	5	6	7	8	9	10	11	12	13
11	**11**	**41**	**00**	**00**	**00**	**00**	**D7**	**11**	**02**	**00**	**01**	**80**

VC例子：

[1:3]：0x411111，表示sequence no为0x411111。

[4:5]：0x0000，表示Src为0x0000，该值**不需要变动**。

[6:7]：0x0000，表示destination addr为0x0000，即只对本地连接的灯响应该命令（和BLE直接连接的灯），表示将要操作对象是本地连接的灯。

[8]：0xd7，表示cmd为0xd7，为**组操作**的命令码。

[9:10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11]：0x00表示删除组（第0个parameter的值）

[12:13]：0x8001表示删除的组号为0x8001（即第一组，组号的识别规则及取值范围详细解析见“命令格式解析”）。当该值为0xFFFF时表示删除所有的组号。

当命令发出后，如果light能正确收到并响应，指示灯会0.5Hz慢闪3次，并且app会收到该light的所有group addr的notify回复。app可以根据这个回复来再次确认是否删除组成功。

R:接收

1. Write_rsp
2. 获取到的group addr的notify回复，数据格式请参考Get_G_8的notify回复。

2.17 Kick_out

剔除出网络。

目前的功能是，删除灯的所有参数（mac地址除外），并把password、ltk恢复为出厂值，设置mesh name为“out_of_mesh”或者默认值(“telink_mesh1”)。

S:发送

表 18 Kick_out 命令示例

1	2	3	4	5	6	7	8	9	10	11
11	**11**	**50**	**00**	**00**	**00**	**00**	**E3**	**11**	**02**	**00**

VC例子：

[1:3]：0x501111，表示sequence no为0x501111。

[4:5]：0x0000，表示Src为0x0000，该值**不需要变动**。

[6:7]：0x0000，表示destination addr为0x0000，即只对本地连接的灯响应该命令（和BLE直接连接的灯），表示将要操作对象是本地连接的灯。

[8]：0xe3，表示cmd为0xe3，为**Kick_out**的命令码。

[9:10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11]: Kick out选项，当没有带这个参数，或者这个参数为0时，会设置light的mesh name为“out_of_mesh”；当这个参数为1时，会恢复light的mesh name为初始值“telink_mesh1”。

R:接收

Write_rsp

2.18 Get_G_8

通过一个数据包获取8个组号（仅低字节）。由于每个灯最多同时属于8个组号，然而我们的回复数据包的par最多可以有10个字节，如果把组号的全部信息都上报的话，最多只能上报5个组号。所以此命令要求app确认知道灯的所有的组号都小于等于0x80FF，上报数据的时候只上报组号的低字节，高字节在APP端自动补0x80。如果不能确认，则需要后续的Get_G_F4和Get_G_B4这两个命令来获取。

因为这个命令获取方式比较简洁，所以建议在APP设置分组信息时，都使用0xFF以下的组号，一般来说是够用的。

S:发送

表 19 Get_G_8 命令示例

1	2	3	4	5	6	7	8	9	10	11	12
11	**11**	**60**	**00**	**00**	**00**	**00**	**DD**	**11**	**02**	**10**	**01**

VC例子：

[1:3]：0x601111，表示sequence no为0x601111。

[4:5]：0x0000，表示Src为0x0000，该值**不需要变动**。

[6:7]：0x0000，表示destination addr为0x0000，即只对本地连接的灯响应该命令（和BLE直接连接的灯），表示将要操作对象是本地连接的灯。

[8]：0xdd，表示cmd为0xdd，为**获取组号**的命令码。

[9:10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11]：0x10表示本地连接的灯收到该命令后转发到mesh网络中的次数（第0个parameter的值）。

[12]：0x01表示获取组号的方式为“一个数据包就获取8个组号”。该字节是作为区分Get_G_8、Get_G_F4、Get_G_B4的作用。

R:接收

1. Write_rsp
2. 获取到的组号的notify回复，数据格式如下（notify所在的UUID为0x1911）

1	2	3	4	5	6	7	8	9	10	11	12
11	**11**	**60**	**02**	**00**	**02**	**00**	**D4**	**11**	**02**	**02**	**03**
13	14	15	16	17	18	19	20				
04	**05**	**06**	**07**	**08**	**09**	**ff**	**ff**				

[1:3]：0x601111，表示sequence no。非加密情况下该值和对应的write命令的sno一致；加密情况下则是随机数。

[4:5]：0x0002，表示Src为0x0002，该值为该组号信息所对应的灯的device addr，属于明文，app可根据此判断是哪一个device addr回的数据包。

[6:7]：0x0002，加密算法校验用。非加密情况下该值和[4：5]相同。除解密需要外，APP不需要用到此参数。

[8]：0xd4，表示cmd为0xd4，为Get_G_8的**组号notify数据**的标识符。

[9：10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11]：0x02表示该灯属于0x0002组，组号标识符为0x8002。

[12]：0x03表示该灯属于0x0003组，组号标识符为0x8003。

[13]：0x04表示该灯属于0x0004组，组号标识符为0x8004。

[14]：0x05表示该灯属于0x0005组，组号标识符为0x8005。

[15]：0x06表示该灯属于0x0006组，组号标识符为0x8006。

[16]：0x07表示该灯属于0x0007组，组号标识符为0x8007。

[17]：0x08表示该灯属于0x0008组，组号标识符为0x8008。

[18]：0x09表示该灯属于0x0009组，组号标识符为0x8009。

[19]：保留字节，此处设为0xff。

[20]：保留字节，此处设为0xff。

注：组号不足8个时，会在[11:18]多余的地方填充0xFF。

2.19 Get_G_F4

通过一个数据包获取某个灯的前4个完整的组号。

需要**注意**的是get_Group_F4和get_Group_B4，需要组合用。一个灯最多可以同时属于8个组号。get_Group_F4只能获取前4个完整的组号，get_Group_B4获取后4个组号，只发送Get_G_F4并不能知道全部的组号，即使前4个都为0xffff（该位置组号为空），并不代表后面4个也都是0xffff，因为不一定是连续放置的。

我们一般建议，app在设置组号的时候，把组号都控制在0~0xff之间（共计256个组号是足够的）。获取组号的时候使用get_Group_8就可以一次性获取完所有8个组号了。

S:发送

表 20 Get_G_F4 命令示例

1	2	3	4	5	6	7	8	9	10	11	12
11	**11**	**61**	**00**	**00**	**00**	**00**	**DD**	**11**	**02**	**10**	**02**

VC例子：

[1:3]：0x611111，表示sequence no为0x611111。

[4:5]：0x0000，表示Src为0x0000，该值**不需要变动**。

[6:7]：0x0000，表示destination addr为0x0000，即只对本地连接的灯响应该命令（和BLE直接连接的灯），表示将要操作对象是本地连接的灯。

[8]：0xdd，表示cmd为0xdd，为**获取组号**的命令码。

[9:10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11]：0x10表示本地连接的灯收到该命令后转发到mesh网络中的次数（第0个parameter的值）。

[12]：0x02表示获取组号的方式为“获取灯的前4个完整的组号”。该字节是作为区分Get_G_8、Get_G_F4、Get_G_B4的作用。

R:接收

1. Write_rsp
2. 获取到的组号的notify回复，数据格式如下（notify所在的UUID为0x1911）

1	2	3	4	5	6	7	8	9	10	11	12
11	**11**	**61**	**02**	**00**	**02**	**00**	**D5**	**11**	**02**	**02**	**80**
13	14	15	16	17	18	19	20				
03	**80**	**04**	**80**	**05**	**80**	**ff**	**ff**				

[1:3]：0x611111，表示sequence no。非加密情况下该值和对应的write命令的sno一致；加密情况下则是随机数。

[4:5]：0x0002，表示Src为0x0002，该值为该组号信息所对应的灯的device addr，属于明文，app可根据此判断是哪一个device addr回的数据包。

[6:7]：0x0002，加密算法校验用。非加密情况下该值和[4：5]相同。除解密需要外，APP不需要用到此参数。

[8]：0xd5，表示cmd为0xd5，为Get_G_F4的**组号notify数据**的标识符。

[9:10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11:12]：0x8002表示该灯属于0x0002组，组号标识符为0x8002。

[13:14]：0x8003表示该灯属于0x0003组，组号标识符为0x8003。

[15:16]：0x8004表示该灯属于0x0004组，组号标识符为0x8004。

[17:18]：0x8005表示该灯属于0x0005组，组号标识符为0x8005。

[19]：保留字节，此处设为0xff。

[20]：保留字节，此处设为0xff。

注：组号不足4个时，会在[11:18]多余的地方填充0xFF。

2.20 Get_G_B4

通过一个数据包获取某个灯的后4个完整的组号。

需要注意的是get_Group_F4和get_Group_B4，需要组合用，详见get_Group_F4解析。

S:发送

表 21 Get_G_B4 命令示例

1	2	3	4	5	6	7	8	9	10	11	12
11	**11**	**62**	**00**	**00**	**00**	**00**	**DD**	**11**	**02**	**10**	**03**

VC例子：

[1:3]：0x621111，表示sequence no为0x621111。

[4:5]：0x0000，表示Src为0x0000，该值**不需要变动**。

[6:7]：0x0000，表示destination addr为0x0000，即只对本地连接的灯响应该命令（和BLE直接连接的灯），表示将要操作对象是本地连接的灯。

[8]:0xdd，表示cmd为0xdd，为**获取组号**的命令码。

[9:10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11]：0x10表示本地连接的灯收到该命令后转发到mesh网络中的次数（第0个parameter的值）

[12]：0x03表示获取组号的方式为“获取灯的后4个完整的组号”。该字节是作为区分Get_G_8、Get_G_F4、Get_G_B4的作用。

R:接收

1. Write_rsp
2. 获取到的组号的notify回复，数据格式如下（notify所在的UUID为0x1911）

1	2	3	4	5	6	7	8	9	10	11	12
11	**11**	**62**	**02**	**00**	**02**	**00**	**D6**	**11**	**02**	**06**	**80**
13	14	15	16	17	18	19	20				
07	**80**	**08**	**80**	**09**	**80**	**ff**	**ff**				

[1:3]：0x621111，表示sequence no。非加密情况下该值和对应的write命令的sno一致；加密情况下则是随机数。

[4:5]：0x0002，表示Src为0x0002，该值为该组号信息所对应的灯的device addr，属于明文，app可根据此判断是哪一个device addr回的数据包。

[6:7]：0x0002，加密算法校验用。非加密情况下该值和[4：5]相同。除解密需要外，APP不需要用到此参数。

[8]:0xd6，表示cmd为0xd6，为Get_G_B4的**组号notify数据**的标识符。

[9：10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11:12]：0x8006表示该灯属于0x0006组，组号标识符为0x8006。

[13:14]：0x8007表示该灯属于0x0007组，组号标识符为0x8007。

[15:16]：0x8008表示该灯属于0x0008组，组号标识符为0x8008。

[17:18]：0x8009表示该灯属于0x0009组，组号标识符为0x8009。

[19]：保留字节，此处设为0xff。

[20]：保留字节，此处设为0xff。

注：组号不足4个时，会在[11:18]多余的地方填充0xFF。

2.21 G1_On

打开组1的灯。

参考All_On，区别是dst字段，All_On的dst字段是0xffff，G1_On的dst字段是0x8001。

2.22 G1_Off

关闭组1的灯。

参考All_Off，区别是dst字段，All_Off的dst字段是0xffff，G1_Off的dst字段是0x8001。

2.23 Status_All

获取所有的灯的亮度、ttc (time to cost)、hops（跳数）信息。

S:发送

表 22 Status_All 命令示例

1	2	3	4	5	6	7	8	9	10	11
11	**11**	**51**	**00**	**00**	**ff**	**ff**	**DA**	**11**	**02**	**10**

VC例子：

[1:3]：0x511111，表示sequence no为0x511111。

[4:5]：0x0000，表示Src为0x0000，该值**不需要变动**。

[6:7]：0xffff，表示destination addr为0xffff，所有的灯都会响应该命令。

注意：网络中灯较多的时候，为了避免因数据在空中冲突而出现丢失的情况，必须按单灯的方式获取。或者使用online status功能。

[8]：0xda，表示cmd为0xda，为**获取灯的状态信息**的命令码。

[9:10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11]：0x10表示本地连接的灯收到该命令后转发到mesh网络中的次数（第0个parameter的值）。

R:接收

1. Write_rsp
2. 获取到的status notify回复，数据格式如下（notify所在的UUID为0x1911）

1	2	3	4	5	6	7	8	9	10	11	12
11	**11**	**51**	**02**	**00**	**02**	**00**	**DB**	**11**	**02**	**ff**	**ff**
13	14	15	16	17	18	19	20				
ff	**ff**	**ff**	**ff**	**00**	**00**	**04**	**01**				

[1:3]：0x511111，表示sequence no。非加密情况下该值和对应的write命令的sno一致；加密情况下则是随机数。

[4:5]：0x0002，表示Src为0x0002，该值为该状态信息所对应的灯的device addr。

[6:7]：0x0002，加密算法校验用。非加密情况下该值和[4：5]相同。

[8]:0xdb，表示cmd为0xdb，为**status notify数据**的标识符。

[9：10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11]：0xff表示该led 1的pwm值为0xff。

[12]：0xff表示该led 2的pwm值为0xff。

[13]：0xff表示该led 3的pwm值为0xff。

[14]：0xff表示该led 4的pwm值为0xff。

[15]：0xff表示该led 5的pwm值为0xff。

[16]：0xff表示该led 6的pwm值为0xff。

[17]：保留字节，此处设为0x00。

[18]：保留字节，此处设为0x00。

[19]：0x04，ttc(time to cost)的值，单位为ms。表示灯收到该命令时，距离APP发出的时刻相差的时间。用来检测网络的延时。如果获取的是“和APP直接连接的灯”，则此处的ttc恒为0。

[20]：0x01，hops的值，表示该命令从和APP直接连接的灯到达该灯需要经过的跳数。如果获取的是“和APP直接连接的灯”，则此处的hops恒为0。

2.24 Status_G1

获取组1的灯的亮度、hop、ttc信息。

参考Status_All，区别是dst字段，Status_All的dst字段是0xffff，Status_G1的dst字段是0x8001。

注意：网络中灯较多的时候，为了避免因数据在空中冲突而出现丢失的情况，必须按单灯的方式获取。或者使用online status功能。

2.25 User_All

获取所有的灯的亮度、ttc (time to cost)、hops（跳数）信息（用户可以自定义）。

S:发送

表 23 User_All 命令示例

1	2	3	4	5	6	7	8	9	10	11	…
11	**11**	**56**	**00**	**00**	**ff**	**ff**	**EA**	**11**	**02**	**10**	**…**

VC例子：

[1:3]：0x561111，表示sequence no为0x561111。

[4:5]：0x0000，表示Src为0x0000，该值**不需要变动**。

[6:7]：0xffff，表示destination addr为0xffff，所有的灯都会响应该命令。

注意：网络中灯较多的时候，为了避免因数据在空中冲突而出现丢失的情况，必须按单灯的方式获取。

[8]：0xea，表示cmd为0xea，为**获取灯的用户自定义信息**的User_All命令码。

[9：10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11]：0x10表示本地连接的灯收到该命令后转发到mesh网络中的次数（第0个parameter的值）。

[12:20]：为可选择的参数，并可在SDK的rf_link_response_callback()中获取，此功能可用于定义User_All命令的子命令等。

R:接收

1. Write_rsp
2. 获取到的status notify回复，数据格式如下（notify所在的UUID为0x1911）

1	2	3	4	5	6	7	8	9	10	11	12
11	**11**	**56**	**02**	**00**	**02**	**00**	**EB**	**11**	**02**	**02**	**01**
13	14	15	16	17	18	19	20				
02	**03**	**04**	**05**	**06**	**07**	**08**	**09**				

[1:3]：0x561111，表示sequence no。非加密情况下该值和对应的write命令的sno一致；加密情况下则是随机数。

[4:5]：0x0002，表示Src为0x0002，该值为该状态信息所对应的灯的device addr。

[6:7]：0x0002，加密算法校验用。非加密情况下该值和[4：5]相同。

[8]：0xeb，表示cmd为0xeb，为User_All的返回信息：**notify数据**的标识符。

[9:10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11:20]：用户自定义数据，目前[11]为预定义的值，是device addr的低8bit，以示区别。用户可更改。

2.26 SW_Config

配置switch的demo命令，该命令只在switch端收到后会有响应动作。目前只是一个demo 命令，当switch收到该命令后，会闪灯n次（可配置）。目的是向客户演示 switch可以收到APP发过来的命令。具体的命令，如有需要，可进行扩展。

S:发送

表 24 SW_Config 命令示例

1	2	3	4	5	6	7	8	9	10	11
11	**11**	**58**	**00**	**00**	**ff**	**ff**	**D3**	**11**	**02**	**04**

VC例子：

[1:3]：0x581111，表示sequence no为0x581111。

[4:5]：0x0000，表示Src为0x0000，该值**不需要变动**。

[6:7]：0xffff，表示destination addr为0xffff，所有当前处于config模式的switch都会响应该命令。

[8]：0xd3，表示cmd为0xd3，为SW_Config的命令码。

[9:10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11]：0x4表示收到该命令后闪烁的次数。

R:接收

1. Write_rsp

2.27 Time_Set

设置时间。

S:发送

表 25 Time_Set 命令示例

1	2	3	4	5	6	7	8	9	10	11	12
11	**11**	**5a**	**00**	**00**	**ff**	**ff**	**E4**	**11**	**02**	**df**	**07**
13	14	15	16	17							
08	**06**	**09**	**00**	**00**							

VC例子：

[1:3]：0x5a1111，表示sequence no为0x5a1111。

[4:5]：0x0000，表示Src为0x0000，该值**不需要变动**。

[6:7]：0xffff，表示destination addr为0xffff，即所有的灯都会响应该命令。Time_Set命令一般都设置成0xffff。

[8]：0xe4，表示cmd为0xe4，为Time_Set的命令码。

[9:10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11:12]：0x07df表示年的字段：2015年（第0个parameter的值）

[13]：0x08表示月的字段：8月。

[14]：0x06表示日的字段：6日。

[15]：0x09表示时的字段：9时。

[16]：0x00表示分的字段：0分。

[17]：0x00表示秒的字段：0秒。

设置成功，灯会慢闪3次，设置失败，则会快闪3次。

R:接收

Write_rsp

2.28 Time_Get

获取灯的时间。

S:发送

表 26 Time_Get 命令示例

1	2	3	4	5	6	7	8	9	10	11
11	**11**	**57**	**00**	**00**	**00**	**00**	**E8**	**11**	**02**	**10**

VC例子：

[1:3]：0x571111，表示sequence no为0x571111。

[4:5]：0x0000，表示Src为0x0000，该值**不需要变动**。

[6:7]：0x0000，表示destination addr为0x0000，即只对本地连接的灯响应该命令（和BLE直接连接的灯），表示将要操作对象是本地连接的灯。

注意：因数据较多，为了避免因数据在空中冲突而出现丢失的情况，必须按单灯的方式获取。

[8]：0xe8，表示cmd为0xe8，为Time_Get的命令码。

[9:10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11]：0x10表示本地连接的灯收到该命令后转发到mesh网络中的次数（第0个parameter的值）

R:接收

1. Write_rsp
2. 获取到的time的notify回复，数据格式如下（notify所在的UUID为0x1911）

1	2	3	4	5	6	7	8	9	10	11	12
11	**11**	**57**	**02**	**00**	**02**	**00**	**E9**	**11**	**02**	**df**	**07**
13	14	15	16	17	18	19	20				
08	**06**	**09**	**00**	**05**	**ff**	**ff**	**ff**				

[1:3]：0x571111，表示sequence no。非加密情况下该值和对应的write命令的sno一致；加密情况下则是随机数。

[4:5]：0x0002，表示Src为0x0002，该值为该time信息所对应的灯的device addr。

[6:7]：0x0002，加密算法校验用。非加密情况下该值和[4：5]相同。

[8]：0xe9，表示cmd为0xe9，为**Time_response**的标识符。

[9：10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11:12]：0x07df表示年的字段：2015年（第0个parameter的值）

[13]：0x08表示月的字段：8月。

[14]：0x06表示日的字段：6日。

[15]：0x09表示时的字段：9时。

[16]：0x00表示分的字段：0分。

[17]：0x05表示秒的字段：5秒。

[18:20]：保留字节，此处设为0xffffff。

2.29 闹钟操作命令

闹钟操作类型目前有5个，分别为：0x00（添加闹钟）、0x01（删除闹钟）、0x02（修改闹钟参数）、0x03（打开闹钟）、0x04（关闭闹钟）。

闹钟命令格式（详见代码）：

```
typedef struct{ // max 10BYTES
    u8 event;
    u8 index;
    struct {
        u8 cmd : 4;
        u8 type : 3;
        u8 enable : 1;
    }par1;
    u8 month;
    union {
        u8 day;
        u8 week;    // BIT(n)
    }par2;
    u8 hour;
    u8 minute;
    u8 second;
    u8 scene_id;
}alarm_ev_t;
```

2.29.1 Get_Alarm

获取闹钟列表信息。

S:发送

表 27 Get_Alarm 命令示例

1	2	3	4	5	6	7	8	9	10	11	12
11	**11**	**5b**	**00**	**00**	**00**	**00**	**E6**	**11**	**02**	**10**	**00**

VC例子：

[1:3]：0x5b1111，表示sequence no为0x5b1111。

[4:5]：0x0000，表示Src为0x0000，该值**不需要变动**。

[6:7]：0x0000，表示destination addr为0x0000，即只对本地连接的灯响应该命令（和BLE直接连接的灯），表示将要操作对象是本地连接的灯。

注意：因数据较多，为了避免因数据在空中冲突而出现丢失的情况，必须按单灯的方式获取。

[8]：0xe6，表示cmd为0xe6，为Get_Alarm的命令码

[9:10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11]：0x10表示本地连接的灯收到该命令后转发到mesh网络中的次数（第0个parameter的值）

[12]：此BYTE表示要读取的闹钟的ID。

当为0x00时，表示读取所有的闹钟的详细信息。

当为0xFF时，表示读取所有的闹钟的id号，但目前最多返回10个。

当为0x01~0x7F时，表示只读取对应的id的信息。如果light中有这个id的信息，则返回这条闹钟信息的完整数据，并把该条信息的闹钟的总个数一栏标记为1。如果light中没有这个id的数据，则会返回全零的数据（获取到的闹钟信息的notify信息的11~20字节，这10个BYTE全为0），这个可用于删除闹钟时，进行回读判断是否删除成功。

R:接收

1. Write_rsp
2. 获取到的闹钟信息的notify回复，数据格式如下（notify所在的UUID为0x1911）

1	2	3	4	5	6	7	8	9	10	11	12
11	**11**	**62**	**02**	**00**	**02**	**00**	**E7**	**11**	**02**	**A5**	**01**
13	14	15	16	17	18	19	20				
81	**08**	**06**	**09**	**00**	**05**	**01**	**01**				

[1:3]：0x621111，表示sequence no。非加密情况下该值和对应的write命令的sno一致；加密情况下则是随机数。

[4:5]：0x0002，表示Src为0x0002，该值为该闹钟信息所对应的灯的device addr。

[6:7]：0x0002，加密算法校验用。非加密情况下该值和[4：5]相同。

[8]：0xe7，表示cmd为0xe7，为**alarm_response**的标识符。

[9:10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11]：0xa5表示该闹钟数据有效（第0个parameter的值）。

[12]：闹钟索引号。

[13]：参考Alarm_Add的[13]的bit组合参数。

[14:18]：参考Alarm_Add的[14:18]。

[19]：闹钟的scene索引（场景类型闹钟时有效）。

[20]：该灯存在的闹钟的总个数，当app收到的闹钟的notify个数和这个值不等的时候，应重新发送Get_Alarm命令。

注意：当[11：19]全为0x00时，表示该灯目前没有闹钟数据。

2.29.2 AlarmOff_Add

添加闹钟(关灯类型)。

表 28　AlarmOff_Add 命令示例

1	2	3	4	5	6	7	8	9	10	11	12
11	**11**	**5c**	**00**	**00**	**ff**	**ff**	**E5**	**11**	**02**	**00**	**01**
13	14	15	16	17	18	19					
80	**01**	**01**	**09**	**01**	**00**	**00**					

请参考AlarmSce_Add。

2.29.3 AlarmOn_Add

添加闹钟(开灯类型)。

表 29 AlarmOn_Add 命令示例

1	2	3	4	5	6	7	8	9	10	11	12
11	**11**	**5c**	**00**	**00**	**ff**	**ff**	**E5**	**11**	**02**	**00**	**02**
13	14	15	16	17	18	19					
81	**01**	**01**	**09**	**01**	**00**	**00**					

请参考AlarmSce_Add。

2.29.4 AlarmSce_Add

添加闹钟（场景类型）。

S:发送

表 30 AlarmSce_Add 命令示例

1	2	3	4	5	6	7	8	9	10	11	12
11	**11**	**5c**	**00**	**00**	**ff**	**ff**	**E5**	**11**	**02**	**00**	**03**
13	14	15	16	17	18	19					
82	**01**	**01**	**09**	**01**	**00**	**01**					

VC例子：

[1:3]：0x5c1111，表示sequence no为0x5c1111。

[4:5]：0x0000，表示Src为0x0000，该值**不需要变动**。

[6:7]：0xffff，表示destination addr为0xffff，即所有的灯都会响应该命令。

[8]：0xe5，表示cmd为0xe5，为Alarm操作的命令码。

[9:10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11]：表示闹钟操作类型，0x00（添加闹钟）。

[12]：表示闹钟索引号，目前一个灯可以设置最多16个闹钟，所以索引号取

值范围是 1~16，在索引值的确定，可以用VC代码中的get_next_shedule_idx()来确定。

另外当APP需要灯端来自动分配索引号的时候，可以设置索引号为0，然后灯会通过同样的算法get_next_shedule_idx()来自动分配一个索引号，自动分配的索引号取值范围是1~16。设置完后，APP发出Alarm_Get回读一下即可。

[13]：bit组合参数，详见结构体alarm_ev_t.par1

```
struct {
   u8 cmd : 4;
   u8 type : 3;
   u8 enable : 1;
}par1;
```

- ✧ bit0~bit3：闹钟的执行动作，目前定义了3个，0：off； 1：on； 2：scene。
- ✧ bit4~bit6：闹钟的类型，目前定义了2个，0：DAY； 1：WEEK。
- ✧ bit7：闹钟的使能标识。1为使能。当进行Alarm Add命令的时候，app应把该bit置1，默认为打开闹钟。

[14]：当闹钟类型为DAY时，为月，取值范围1~12。当闹钟类型为WEEK时，该字节为保留字节。

[15]：当闹钟类型为DAY时，为日。当闹钟类型为WEEK时，该字节为星期，bit0-bit6分别代表星期天(bit0)、星期一到星期六(bit1-bit6)。bit7为保留位，需要设为0，否则会认为该参数非法。

[16]：时。

[17]：分。

[18]：秒。

[19]：scene id。

2.29.5 Alarm_Del

删除闹钟。

S:发送

表 31 Alarm_Del 命令示例

1	2	3	4	5	6	7	8	9	10	11	12
11	**11**	**5d**	**00**	**00**	**ff**	**ff**	**E5**	**11**	**02**	**01**	**01**
13	14	15	16	17	18						
00	**00**	**00**	**00**	**00**	**00**						

VC例子：

[1:3]：0x5d1111，表示sequence no为0x5d1111。

[4:5]：0x0000，表示Src为0x0000，该值**不需要变动**。

[6:7]：0xffff，表示destination addr为0xffff，即所有的灯都会响应该命令。

[8]：0xe5，表示cmd为0xe5，为Alarm操作的命令码。

[9：10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11]：表示闹钟操作类型，0x01（删除闹钟）。

[12]：表示闹钟索引号，目前一个灯可以设置的最多16个闹钟，所以索引号取值范围是 1~16。

另外0xff表示删除该灯所有的闹钟信息。

[13:18]：保留位，设为0。

R:接收

1. Write_rsp

2.29.6 Alarm_En

打开闹钟。

S:发送

表 32 Alarm_En 命令示例

1	2	3	4	5	6	7	8	9	10	11	12
11	**11**	**60**	**00**	**00**	**ff**	**ff**	**E5**	**11**	**02**	**03**	**01**
13	14	15	16	17	18						
00	**00**	**00**	**00**	**00**	**00**						

VC例子：

[1:3]：0x601111，表示sequence no为0x601111。

[4:5]：0x0000，表示Src为0x0000，该值**不需要变动**。

[6:7]：0xffff，表示destination addr为0xffff，即所有的灯都会响应该命令。

[8]：0xe5，表示cmd为0xe5，为Alarm操作的命令码。

[9：10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11]：表示闹钟操作类型，0x03（打开闹钟）。

[12]：表示闹钟索引号，目前一个灯可以设置的最多16个闹钟，所以索引号取值范围是 1~16。

[13:18]：保留位，设为0。

R:接收

1. Write_rsp

2.29.7 Alarm_Dis

关闭闹钟。

S:发送

表 33 Alarm_Dis 命令示例

1	2	3	4	5	6	7	8	9	10	11	12
11	**11**	**61**	**00**	**00**	**ff**	**ff**	**E5**	**11**	**02**	**04**	**01**
13	14	15	16	17	18						
00	**00**	**00**	**00**	**00**	**00**						

VC例子：

[1:3]：0x611111，表示sequence no为0x611111。

[4:5]：0x0000，表示Src为0x0000，该值**不需要变动**。

[6:7]：0xffff，表示destination addr为0xffff，即所有的灯都会响应该命令。

[8]：0xe5，表示cmd为0xe5，为Alarm操作的命令码。

[9:10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11]：表示闹钟操作类型，0x04（关闭闹钟）

[12]：表示闹钟索引号，目前一个灯可以设置的最多16个闹钟，所以索引号取值范围是 1~16。

[13:18]：保留位，设为0。

R:接收

1. Write_rsp

2.29.8 Alarm_Chg

更改闹钟。

S:发送

表 34 Alarm_Chg 命令示例

1	2	3	4	5	6	7	8	9	10	11	12
11	**11**	**64**	**00**	**00**	**ff**	**ff**	**E5**	**11**	**02**	**02**	**01**
13	14	15	16	17	18						
80	**01**	**01**	**08**	**00**	**1e**						

VC例子：

[1:3]：0x641111，表示sequence no 为0x641111。

[4:5]：0x0000，表示Src为0x0000，该值**不需要变动**。

[6:7]：0xffff，表示destination addr为0xffff，即所有的灯都会响应该命令。

[8]：0xe5，表示cmd为0xe5，为Alarm操作的命令码。

[9:10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11]：表示闹钟操作类型，0x02（更改闹钟）。

[12]：表示闹钟索引号，目前一个灯可以设置的最多16个闹钟，所以索引号取值范围是 1~16。

[13:18]：闹钟参数设置，请参考Alarm_Add。

R:接收

1. Write_rsp

注：以上闹钟操作：添加闹钟、删除闹钟、修改闹钟参数、打开闹钟、关闭闹钟，执行完后，APP都应用Alarm_Get回读一下闹钟信息，以确认是否设置成功。

2.30 场景模式操作命令 (Scene)

2.30.1 Add_Scene

增加场景。

S:发送

表 35 Add_Scene 命令示例

1	2	3	4	5	6	7	8	9	10	11	12
11	**11**	**66**	**00**	**00**	**00**	**00**	**EE**	**11**	**02**	**01**	**01**
13	14	15	16								
64	**00**	**FF**	**FF**								

VC例子：

[1:3]：0x661111，表示sequence no为0x661111。

[4:5]：0x0000，表示Src为0x0000，该值**不需要变动**。

[6:7]：0x0000，表示destination addr为0x0000，即只有当前直连的灯会响应该命令。

[8]：0xEE，表示cmd为0xEE，为scene操作的命令码。

[9:10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11]：表示场景的操作类型，0x01（添加场景）。

[12:16]：1个scene的数据包（不包括保留的3byte数据），请参考**场景数据结构**。

场景数据结构

```
typedef struct{
   u8 id;                  //场景数据索引号
   u8 lum;                 //亮度值
   u8 rgb[3];              //灯的 RGB 值
   u8 rsv[3];              //保留数据
}scene_t;
```

2.30.2 Del_Scene

删除场景。

S:发送

表 36 Del_Scene 命令示例

1	2	3	4	5	6	7	8	9	10	11	12
11	**11**	**69**	**00**	**00**	**00**	**00**	**EE**	**11**	**02**	**00**	**01**

VC例子：

[1:3]：0x691111，表示sequence no为0x691111。

[4:5]：0x0000，表示Src为0x0000，该值**不需要变动**。

[6:7]：0x0000，表示destination addr为0x0000，即只有当前直连的灯会响应该命令。

[8]：0xEE，表示cmd为0xEE，为scene操作的命令码。

[9：10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11]：表示场景的操作类型，0x00（删除场景）。

[12]：场景索引，若该值为0xff，则删除所有场景。

2.30.3 Load_Scene

加载场景。

S:发送

表 37 Load_Scene 命令示例

1	2	3	4	5	6	7	8	9	10	11
11	**11**	**6C**	**00**	**00**	**FF**	**FF**	**EF**	**11**	**02**	**01**

VC例子：

[1:3]：0x6C1111，表示sequence no为0x6C1111。

[4:5]：0x0000，表示Src为0x0000，该值**不需要变动**。

[6:7]：0xffff，表示destination addr为0xffff，即所有的灯都会响应该命令。

[8]：0xEF，表示cmd为0xEF，为Load_scene操作的命令码。

[9:10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11]：表示场景索引。

2.30.4 Get_Scene

获取场景数据列表。

S:发送

表 38 Get_Scene 命令示例

1	2	3	4	5	6	7	8	9	10	11	12
11	**11**	**6e**	**00**	**00**	**00**	**00**	**C0**	**11**	**02**	**10**	**00**

VC例子：

[1:3]：0x6e1111，表示sequence no为0x6e1111。

[4:5]：0x0000，表示Src为0x0000，该值**不需要变动**。

[6:7]：0x0000，表示destination addr为0x0000，即只对本地连接的灯响应该命令（和BLE直接连接的灯），表示将要操作对象是本地连接的灯。

注意：因数据较多，为了避免因数据在空中冲突而出现丢失的情况，必须按单灯的方式获取。

[8]：0xc0，表示cmd为0xc0，为Get_Scene的命令码。

[9:10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11]：0x10表示本地连接的灯收到该命令后转发到mesh网络中的次数（第0个parameter的值）。

[12]：此BYTE表示要读取的场景的ID。

当为0x00时，表示读取所有的场景的详细信息。

当为0xFF时，表示读取所有的场景的id号，但目前最多返回10个。

当为0x01~0x7F时，表示只读取对应的id的信息。如果light中有这个id的信息，则返回这条场景信息的完整数据，并把该条信息的场景的总个数一栏标记为1。如果没有这个id的数据，则会返回全零的数据（获取到的scene的notify回复信息的11~20字节，这10个BYTE全为0），这个可用于删除场景时，进行回读判断是否删除成功。

R:接收

1. Write_rsp
2. 获取到的scene的notify回复，数据格式如下（notify所在的UUID为0x1911）

1	2	3	4	5	6	7	8	9	10	11	12
11	**11**	**6e**	**55**	**00**	**55**	**00**	**C1**	**11**	**02**	**01**	**64**
13	14	15	16	17	18	19	20				
00	**FF**	**FF**	**09**	**00**	**05**	**02**	**00**				

[1:3]：0x6e1111，表示sequence no。非加密情况下该值和对应的write命令的sno一致；加密情况下则是随机数。

[4:5]：0x0055，表示Src为0x0055，该值为该场景信息所对应的灯的device addr。

[6:7]：0x0055，加密算法校验用。非加密情况下该值和[4：5]相同。

[8]：0xc1，表示cmd为0xc1，为**scene_response**的标识符。

[9：10]VendorID：默认为 0x11 0x02，该值**不需要变动**。

[11]：场景索引号。

[11:18]：一个完整的scene数据包，请参考Add_Scene**场景数据结构**。

[19]：该灯存在的场景的总个数，当app收到的场景的notify个数和这个值不等的时候，应重新发送Get_Scene命令。

[20]：保留字节，为0x00。

注意：当[11:19]全为0x00时，表示该灯目前没有场景数据。

3 NotifyStatus（NotifyStatus UUID 1911）

3.1 MeshLightStatus

当使用 online status 功能时，灯端：当第一次上电时，如果 device address 为空，则会取 mac address 的低 8bit，高 8bit 为 0。当添加多个 light 到同一个网络时，如果存在相同的 device address，则需要 APP 进行配置，并保证该 device address 的范围是 0x0001—0x00FF。目的是为了，能更多的上报用户数据（本章节的后续将详细介绍）。

Online_All

打开 online status 功能。使用 write_req 命令往 0x12 的 AttHandle (对应的 UUID 为 0x1911)发送以下命令。

S:发送

1
01

VC例子：

[1]：0x01，表示打开MeshLightStatus功能。

R:接收

Write_rsp

打开MeshLightStatus功能后，当前网络的每个灯都会上报一个notify的数据到APP进行状态显示，然后就不再发送notify数据。后续只要某个灯的状态有变化（包括亮度值、开关状态、离线在线状态等），都会再上报一个notify数据到APP进行状态更新。

上报的notify数据格式如下：

1	2	3	4	5	6	7	8	9	10	11	12
00	**00**	**00**	**00**	**00**	**00**	**00**	**DC**	**11**	**02**	**11**	**3C**
13	14	15	16	17	18	19	20				
64	**FF**	**22**	**4B**	**64**	**FF**	**00**	**00**				

[1:3]：sno (sequence number)。非加密情况下暂时为0；加密情况下则是随机数。

[4:5]：src (source addr)。暂时都为0。

[6:7]：dst (determination number)。非加密情况下暂时为0；加密情况下则是作为加密校验用。

[8]：0xdc：LightStatus上报数据标识符。

[9:10]：0x0211。Vendor ID。

[11:14]和[15:18]分别是两个灯的上报数据。

[11]：0x11，表示第一个灯的device addr。此处没有上报完整的device address，目的是为了能更多的上报用户数据，所以要求当使用online status 功能时，需要手动配置light node的device address，并保证该device address的范围是0x0001~0x00FF。

[12]：0x3c，表示第一个灯上报的数据的序号sn。当sn为0时表示灯已经离开网络(比如断电，或者超距离等)，非0则表示在线。

[13]：第一个灯的亮度值，范围为0~100。

[14]：保留字段，供客户使用。目前默认值是0xFF。

[15]：0x22，表示第二个灯的device addr。

[16]：0x4B，表示第二个灯上报的数据的序号。当sn为0时表示灯已经离开网络(比如断电，或者超距离等)，非0则表示在线。

[17]：第二个灯的亮度值，范围为0~100。

[18]：保留字段，供客户使用。目前默认值是0xFF。

注：当[15:18]为0x00000000时，表示只有一个灯的上报数据。(0x00000000为空值)。一般情况下，只有刚发送Online_All命令后，得到的online status数据才会包含两个灯的数据，其他情况下都是只有一个灯的数据的。

[19]：00 保留字节。

[20]：00 保留字节。

3.2 User notify data

处于BLE连接状态的Light端可主动发送notify数据，通过调用SDK的light_notify(u8 *p, u8 len)接口。

User notify data上报数据格式如下：

1	2	3	4	5	6	7	8	9	10	11	12
00	**00**	**00**	**00**	**00**	**00**	**00**	**EA**	**11**	**02**	**06**	**00**
13	14	15	16	17	18	19	20				
00	**00**	**00**	**00**	**00**	**00**	**00**	**00**				

[1:3]：sno (sequence number)。非加密情况下暂时为0；加密情况下则是随机数。

[4:5]：src (source addr)。暂时都为0。

[6:7]：dst (determination number)。非加密情况下暂时为0；加密情况下则是作为加密校验用。

[8]：0xEA，User notify data上报数据标识符。

[9:10]：0x0211。Vendor ID。

[11]： 目前demo是做累加计数用，可自定义。

[12:20]：用户自定义的数据，默认为0。

4 MeshLightOTA（MeshLightOTA UUID 1913）

S:发送

1	2	3	4	5	6	7	8	9	10	11	12
01	**00**	**76**	**80**	**00**	**00**	**00**	**00**	**00**	**00**	**62**	**43**
13	14	15	16	17	18	19	20				
00	**00**	**00**	**00**	**00**	**00**	**F3**	**0B**				

VC例子：

[1:2]：0x0001，表示firmware的数据包序列index，从0开始。

[3:18]：16 bytes firmware data。

[19:20]：数据序列[1:18]的crc16校验值，具体算法参考源码。

一个数据包所加载的firmware为16BYTE。例外情况：倒数第二个包数据个数小于等于16（firmware的末尾部分）。最后一个数据包包含的firmware长度为0，只有idex和CRC校验值，以用作OTA结束标记。

crc16的校验值获取，请参考SDK或者VC代码中的unsigned short crc16 (unsigned char *pD, int len)；其中VC代码是注释的，因为在VC端用不到，仅供APP copy。

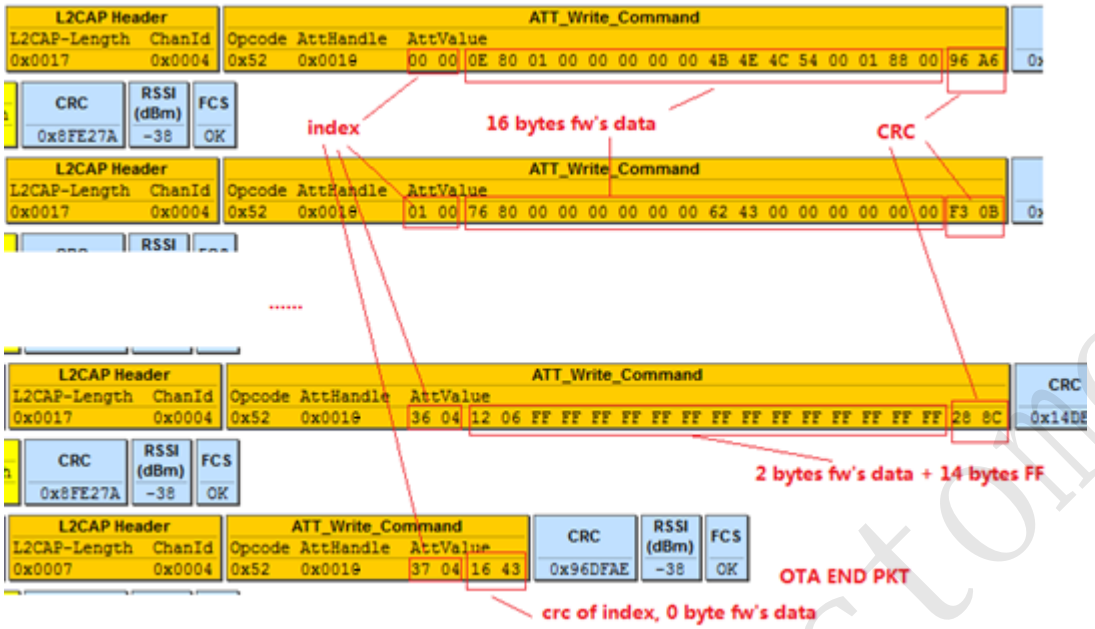
unsigned short crc16 (unsigned char *pD, int len)传参详解：

pD = 下图中AttValue的首字节的地址（即index首地址）

len = index的长度（等于2） + （inde 和 crc 之间的数据的长度）

比如下面4个数据包的len分别等于 18、18、18、2。

R:接收NULL



APP OTA 设计流程：

1. Light 端：实现 OTA 功能的时候，light 端的 demo code 已经包含了此功能，不需要进行任何的改动。
2. APP 端：

   流程简介：把待升级的 firmware 的内容都按本章节开始的要求发送出去即可。

   设计流程参考：

   1） 把待更新的 firmware 拷贝到 app 本地。

   2） 获取 firmware 总长度（firmware 的第 25 到 28 个 BYTE 的值为 firmware 的长度）。

   3） 从 firmware 的 addr = 0 地址开始读取 16 个字节的数据，填充在命令包对应的位置，然后使 index = 0 并计算 crc，然后填充 index 和 crc 在命令包对应的位置（注意存储结构：低字节在前）；然后发送出去。

   4） 在发送完 BLE 数据的回调函数中判断：

   ✧ 如果发送成功：则进行下一个数据包的发送：从 firmware 的 addr=addr+16 地址开始读取 16 个字节的数据，填充在命令包对应的位置，然后使 index = index +1 并计算 crc，然后填充 index 和 crc 在命令包对应的位置；然后发送出去。

✧ 如果发送失败：则需要重新发送该数据包。

5） 重复第 4 个步骤，直到把所有的数据都发送完成。

需要注意的是，最后一个需要发送 firmware 的数据包里面包含 firmware 的内容可能不满 16 个，此时可以在空的地方填 0xff，len 还是保持 16；或者不填充 0xff，改下 len 就可以了。（如果不改的话，会多出几个数据出来，在进行 Flash 数据比对确认的时候在最后的地方就会有差异。）

6） 发送 firmware 数据长度为 0 的数据包（用作结束标记）。至此就完成了整个 OTA 的发送流程。

该流程可考虑增加超时判断，即一定时间内如果 OTA 过程还未正常结束，则断开连接。防止意外情况发送。

5 MeshLightPair（MeshLightPair UUID 1914）

为了调试方便，现开放了写死 rands 的功能，打开此功能后，就可以比对同一个命令，app 发出的加密数据和 VC 发出的做对比，相同则表示加密 ok，否则请再次确认加密传参是否有错误。

打开功能的方法，在 user_init()的最后的地方，设置 rands_fix_flag = 1 即可，代码中已有此段代码，默认是屏蔽的。

数据加密参数介绍：

sec_ivm[0]: slave_mac_address[0];
sec_ivm[1]: slave_mac_address[1];
sec_ivm[2]: slave_mac_address[2];
sec_ivm[3]: slave_mac_address[3]
sec_ivm[4]:固定为 1
sec_ivm[5]: sno[0];
sec_ivm[6]: sno[1];
sec_ivm[6]: sno[2];

sec_ivs[0]: slave_mac_address[0];
sec_ivs[1]: slave_mac_address[1];
sec_ivs[2]: slave_mac_address[2];
sec_ivs [3] = sno[0];
sec_ivs [4] = sno[1];
sec_ivs [5] = sno[2];
sec_ivs [6] = src[0]; //从发送的命令中获取，在 sno 后的第一个 byte
sec_ivs [7] = src[1]; //从发送的命令中获取，在 sno 后的第二个 byte

6 Factory Reset

当需要进行恢复出厂设置时，可以进行如下 6 次开机序列来触发 Factory Reset 功能：

（现以 demo code 默认的数据进行说明，开发人员可以配置该时间参数）

第一步：上电，然后 3 秒钟内断电。

第二步：上电，然后 3 秒钟内断电。

第三步：上电，然后 3 秒钟内断电。

第四步：上电，然后在上电 3 秒钟之后，并且在 30 秒钟之前断电。

第五步：上电，然后在上电 3 秒钟之后，并且在 30 秒钟之前断电。

第六步：上电，然后灯会以 0.5Hz 闪烁 3 次，即表示触发了 Factory Reset 功能，所有用户配置的参数都会恢复为出厂默认值。

当需要重新配置该时间参数时，修改如下数组即可：

```
u8 factory_reset_serials[SERIALS_CNT * 2]   = {
0, 3,    // [0]:must 0
0, 3,    // [2]:must 0
0, 3,    // [4]:must 0
3, 30,
3, 30
};
```

7 Device 过滤

在进行 app 开发的时候，需要在 BLE 设备扫描列表中过滤不属于预定义的 mesh_name 和 vendor_id 的广播包。 mesh_name 和 vendor_id 的值可以广播包中获取，mesh_name 预定义为”telink_mesh1”，vendor_id 预定义为 0x0211。