Application Note : Communication Protocol for Telink BLE Mesh Light APP

AN-BLE-15120202-E3

Ver 1.1.1

2016/6/3

Brief:

This document introduces communication protocol for Telink BLE (Bluetooth Low Energy) Mesh Light APP.

TELINK SEMICONDUCTOR

Published by
Telink Semiconductor

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

Legal Disclaimer

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

Revision History

Version	Major Changes	Date	Author
1.0.0	Initial release	2016/2	J.G.H. & Cynthia
1.1.0	Added specific ID reading function in Get_Alarm/Get_Scene command	2016/5	S.Q.F., Cynthia
1.1.1	Further illustrate Del_G1[12:13]	2016/6	S.Q.F., Cynthia

Table of contents

Table of tables

1 UUID part

```
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";
```

All data in following data format are in non-encryption mode.

2 MeshLightCommand (MeshLightCommand UUID 1912)

Table 1 Command format analysis

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” is lower byte. Sequence number should not be 0, and it will be added by 1 when a command is sent.

[4:5](Src): Source address. It’s fixed as “0x0000” for APP.

[6:7](dst): Destination address among the range of 0x0000~0xFFFF.

- ✧ 0x0000: Local addr. Only light directly connected with APP will respond to this command.
- ✧ 0x0001 ~ 0x00FF: Device addr, used for single light control. Only light with corresponding device address will respond to this command.
- ✧ 0x8000 ~ 0xfffe: Group address, used for group control. Only a group of lights with corresponding group address will respond to this command.
- ✧ 0xffff: Advertising address. All lights will respond to this command.

Rules: When dst[15] is 0/1, it indicates the address is device addr/group addr, respectively (except for 0x0000 and 0xffff).

For light, if device addr is not configured manually, higher byte and lower byte of device addr are 0x00 and the lowest byte of MAC by default. For example, if MAC is “FF:FF:33:18:d3:31”, default device addr is “0x0031”. If the lowest byte of MAC is 0, default device addr is “0x0001”.

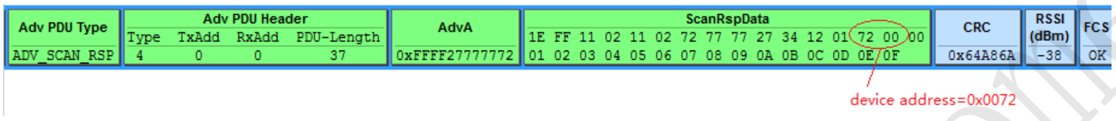
It’s recommended to follow the suggestions below to allocate device addr.

1) Use default value.
2) Sometimes there may be overlap to use default device addr. Solution: When a light is added, APP can automatically allocate a device addr (0x0001~0x00FF) via “Device_Addr” command, and will use auto increment by 1 for the next light’s device addr.

APP can also obtain device addr information from adv scan_response packet. Please refer to data structure “ll_adv_rsp_private_t” in SDK.

APP can obtain device addr according to any of the following methods.

1) Obtain device addr from scan response packet of adv packet. Please refer to data structure “adv_rsp_pri_data” in SDK for the format.



Adv PDU Type	Adv PDU Header				AdvA	ScanRspData	CRC	RSSI (dBm)	FCS
	Type	TxAdd	RxAdd	PDU-Length		1E FF 11 02 11 02 72 77 77 27 34 12 01 72 00 00			
ADV_SCAN_RSP	4	0	0	37	0xFFFF27777772	01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	0x64A86A	-38	OK

2) Obtain device addr from online status packet.

3) Obtain device addr via sending “Get_Dev_Addr” command.

[8](cmd): Command code. Both bit6 and bit7 are “1” by default. E.g. “cmd” of “ALL_ON” is “0xd0”.

[9:10](VendorID): Vendor ID. It’s 0x11 0x02 by default.

[8：10]: op code.

[11:13](Par): Parameter field, up to 10 bytes. Only 3 bytes are listed in Table 1.

In following sub-sections, command examples in **VC code** are given for developer’s reference.

2.1 ALL_ON

Turn on all lights simultaneously.

S: Send

Table 2 ALL_ON command example

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**

[1:3]: Sequence number is “0x111111”.

[4:5]: Source addr is fixed as “0x0000”.

[6:7]: Destination addr is “0xffff”, i.e. all lights will respond to this command.

[8]: “cmd” is “0xd0” (command code of “Turn on/off light”).

[9:10]: VendorID is “0x11 0x02” by default and it’s fixed.

[11]: “0x01” (parameter 0) indicates turning on light.

[12:13]: “01 00” (parameter 1 and 2) indicates turning on light after 1ms delay. [12] (lower byte) and [13] (higher byte) determine delay time in ms.

R: Receive

Write_rsp

2.2 ALL_OFF

Turn off all lights simultaneously.

S: Send

Table 3 ALL_OFF command example

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**

[1:3]: Sequence number is “0x121111”.

[4:5]: Source addr is fixed as “0x0000”.

[6:7]: Destination addr is “0xffff”, i.e. all lights will respond to this command.

[8]: “cmd” is “0xd0” (command code of “Turn on/off light”).

[9:10]: VendorID is “0x11 0x02” by default and it’s fixed.

[11]: “0x00” (parameter 0) indicates turning off light.

[12:13]: “01 00” (parameter 1 and 2) indicates turning off light after 1ms delay. [12] (lower byte) and [13] (higher byte) determine delay time in ms.

R: Receive

Write_rsp

2.3 ALL_ON_500MS

Turn on all lights simultaneously after 500ms delay. This command can solve the non-synchronization due to delay caused by command relay.

S: Send

Table 4 ALL_ON_500MS command example

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**

[1:3]: Sequence number is "0x111111".

[4:5]: Source addr is fixed as "0x0000".

[6:7]: Destination addr is "0xffff", i.e. all lights will respond to this command.

[8]: "cmd" is "0xd0" (command code of "Turn on/off light").

[9:10]: VendorID is "0x11 0x02" by default and it's fixed.

[11]: "0x01" (parameter 0) indicates turning on light.

[12:13]: "01 02 (0x0201)" (parameter 1 and 2) indicates turning on light after 513ms delay. [12] (lower byte) and [13] (higher byte) determine delay time in ms.

R: Receive

Write_rsp

2.4 ALL_OFF_500MS

Turn off all lights simultaneously after 500ms delay. This command can solve the asynchronization due to delay caused by command relay.

S: Send

Table 5 ALL_OFF_500MS command example

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**

[1:3]: Sequence number is “0x121111”.

[4:5]: Source addr is fixed as “0x0000”.

[6:7]: Destination addr is “0xffff”, i.e. all lights will respond to this command.

[8]: “cmd” is “0xd0” (command code of “Turn on/off light”).

[9:10]: VendorID is “0x11 0x02” by default and it’s fixed.

[11]: “0x00” (parameter 0) indicates turning off light.

[12:13]: “01 02 (0x0201)” (parameter 1 and 2) indicates turning off light after 513ms delay. [12] (lower byte) and [13] (higher byte) determine delay time in ms.

R: Receive

Write_rsp

2.5 Set_Lum

Set luminance for all lights.

S: Send

Table 6 Set_Lum command example

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

[1:3]: Sequence number is “0x131111”.

[4:5]: Source addr is fixed as “0x0000”.

[6:7]: Destination addr is “0x0000”, only locally connected light (i.e. light directly connected with BLE) will respond to this command.

[8]: “cmd” is “0xd2” (command code of “set luminance value”).

[9:10]: VendorID is “0x11 0x02” by default and it’s fixed.

[11]: “0x0A” indicates setting luminance value as “0x0A”.

R: Receive

Write_rsp

2.6 Music_Start

Music function procedure: Before it’s ready to transmit music data, “Music Start” command is first sent to inform light to store its current state, then music volume values (0~100) are continuously sent via “Set_Lum” command. After music ends, “Music Stop” command is sent to restore light state.

***Note**: To simplify command, opcode of “Music Start” and “Music Stop” is the same as “Set_Lum”.

S: Send

Table 7 Music_Start command example

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

[1:3]: Sequence number is “0x161111”.

[4:5]: Source addr is fixed as “0x0000”.

[6:7]: Destination addr is “0x0000”, only locally connected light (i.e. light directly connected with BLE) will respond to this command.

[8]: “cmd” is “0xd2” (command code of “Set luminance value”).

[9:10]: VendorID is “0x11 0x02” by default and it’s fixed.

[11]: “0xFE” indicates music start function.

R: Receive

Write_rsp

2.7 Music_Stop

Restore light state after “Music Stop” command is received.

S: Send

Table 8 Music_Stop command example

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

[1:3]: Sequence number is “0x171111”.

[4:5]: Source addr is fixed as “0x0000”.

[6:7]: Destination addr is “0x0000”, only locally connected light (i.e. light directly connected with BLE) will respond to this command.

[8]: “cmd” is “0xd2” (command code of “Set luminance value”).

[9:10]: VendorID is “0x11 0x02” by default and it’s fixed.

[11]: “0xFF” indicates music stop function.

R: Receive

Write_rsp

2.8 Set_R

Set luminance value of red light.

S: Send

Table 9 Set_R command example

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

[1:3]: Sequence number is “0x811111”.

[4:5]: Source addr is fixed as “0x0000”.

[6:7]: Destination addr is “0xffff”, i.e. all lights will respond to this command.

[8]: “cmd” is “0xe2” (command code of “set luminance value of single light”).

[9:10]: VendorID is “0x11 0x02” by default and it’s fixed.

[11]: “0x01” (parameter 0) indicates setting luminance of red light.

[12]: “0x00” indicates color index value is “0”. (Note: Index value range is 0x00~0xff, i.e. 0~255.)

R: Receive

Write_rsp

2.9 Set_G

Set luminance value of green light.

S: Send

Table 10 Set_G command example

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

[1:3]: Sequence number is “0x831111”.

[4:5]: Source addr is fixed as “0x0000”.

[6:7]: Destination addr is “0xffff”, i.e. all lights will respond to this command.

[8]: “cmd” is “0xe2” (command code of “set luminance value of single light”).

[9:10]: VendorID is “0x11 0x02” by default and it’s fixed.

[11]: “0x02” (parameter 0) indicates setting luminance of green light.

[12]: “0x00” indicates color index value is “0”. (Note: Index value range is 0x00~0xff, i.e. 0~255.)

R: Receive

Write_rsp

2.10 Set_B

Set luminance value of blue light.

S: Send

Table 11 Set_B command example

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

[1:3]: Sequence number is “0x851111”.

[4:5]: Source addr is fixed as “0x0000”.

[6:7]: Destination addr is “0xffff”, i.e. all lights will respond to this command.

[8]: "cmd" is "0xe2" (command code of "set luminance value of single light").

[9:10]: VendorID is "0x11 0x02" by default and it's fixed.

[11]: "0x03" (parameter 0) indicates setting luminance of blue light.

[12]: "0x00" indicates color index value is "0". (Note: Index value range is 0x00~0xff, i.e. 0~255.)

R: Receive

Write_rsp

2.11 Set_RGB

Set luminance value of RGB lights simultaneously.

S: Send

Table 12 Set_RGB command example

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**

[1:3]: Sequence number is "0x871111".

[4:5]: Source addr is fixed as "0x0000".

[6:7]: Destination addr is "0xffff", i.e. all lights will respond to this command.

[8]: "cmd" is "0xe2" (command code of "set luminance value of single light").

[9:10]: "VendorID" is "0x11 0x02" by default and it's fixed.

[11]: "0x04" (parameter 0) indicates setting luminance of RGB lights.

[12:14]: Color index value for "R", "G" and "B" are "0x70", "0x90" and "0xB0", respectively. (Note: Index value range is 0x00~0xff, i.e. 0~255.)

R: Receive

Write_rsp

2.12 Set_CT

Set CT value (percentage) of CT light.

S: Send

Table 13 Set_CT command example

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

[1:3]: Sequence number is “0x881111”.

[4:5]: Source addr is fixed as “0x0000”.

[6:7]: Destination addr is “0xffff”, i.e. all lights will respond to this command.

[8]: “cmd” is “0xe2” (command code of “set luminance value of single light”).

[9:10]: VendorID is “0x11 0x02” by default and it’s fixed.

[11]: “0x05” (parameter 0) indicates setting CT value.

[12]: “00” indicates CT value percentage is 0%. (Note: CT value range is 0x00~0x64, i.e. 0%~100%.)

R: Receive

Write_rsp

2.13 Device_Addr

1） Modify device addr (Note that it’s address used for device identification in mesh rather than mac address).

2） Obtain all lights of a group number, or manually obtain Device_Addr (Refer to “Get_Dev_Addr” command).

S: Send

Table 14 Device_Addr command example

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

[1:3]: Sequence number is “0x701111”.

[4:5]: Source addr is fixed as “0x0000”.

[6:7]: Destination addr is “0x0000”, only locally connected light (i.e. light directly connected with BLE) will respond to this command.

Destination address range: 0x0001~0x00FF (address of single light).

[8]: “cmd” is “0xe0” (command code of “set device address of single light”).

[9:10]: VendorID is “0x11 0x02” by default and it’s fixed.

[11:12]: “0x0011” indicates modifying device addr of corresponding light into “0x0011”.

Range: 0x0001~ 0x00FF. After setting, it’s not needed to manually read device addr, and light will automatically return its current device addr via “notify”. APP can check if device addr setting is successful accordingly.

R: Receive

1. Write_rsp
2. Data format of notify response (notify UUID is **0x1911**) to return device addr is as shown below.

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]: Sequence number is “0x701111”. In non-encryption scenario, it’s the same as corresponding command; in encryption scenario, it’s random number.

[4:5]: Source addr is “0x0011”, which is newly set device addr of corresponding light.

[6:7]: 0x1111, used for encryption algorithm check. In non-encryption scenario, its value is the same as [4:5].

[8]: “cmd” is “0xe1” (identifier for device addr **notify data**).

[9:10]: VendorID is “0x11 0x02” by default and it’s fixed.

[11:12]: “0x0011” indicates returned device addr of corresponding light.

[13:20]: reserved bytes, all set as “0x00”.

2.14 Get_Dev_Addr

Obtain all lights of a group number, or manually obtain Device_Addr.

S: Send

Table 15 Get_Dev_Addr command example

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

[1:3]: Sequence number is “0x721111”.

[4:5]: Source addr is fixed as “0x0000”.

[6:7]: “0x8001” indicates destination addr is Group1 (group addr). The operation target is a group.

- ✧ To obtain all lights of a group number, destination addr should be 0x8000~0xffff (group addr).
- ✧ To manually obtain Device_Addr, destination addr should be 0x0001~0x00FF (device addr for single light).

[8]: “cmd” is “0xe0” (multiplexed command).

[9:10]: VendorID is “0x11 0x02” by default and it’s fixed.

[11:12]: must be “0xffff”.

After this command is sent, a notify response will be received to return device addr of all matched lights.

R: Receive

1. Write_rsp
2. Please refer to “Device_Addr” command for data format of notify response (notify UUID is 0x1911).

2.15 Add_G1

Add a group.

S: Send

Table 16 Add_G1 command example

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**

[1:3]: Sequence number is “0x211111”.

[4:5]: Source addr is fixed as “0x0000”.

[6:7]: Destination addr is “0x0000”, only locally connected light (i.e. light directly connected with BLE) will respond to this command.

[8]: “cmd” is “0xd7” (command code of group operation).

[9:10]: VendorID is “0x11 0x02” by default and it’s fixed.

[11]: “0x01” (parameter 0) indicates adding a group.

[12:13]: “0x8001” indicates group number to be added is 0x8001 (Group1).

If light receives and correctly responds to this command, the indicating light will blink slowly (0.5Hz) for three times, notify response is sent to APP to return corresponding group addr. APP can check if corresponding group is successfully added according to this notify response.

R: Receive

1. Write_rsp
2. Please refer to “Get_G_8” command for data format of notify response.

2.16 Del_G1

Delete a group.

S: Send

Table 17 Del_G1 command example

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**

[1:3]: Sequence number is “0x411111”.

[4:5]: Source addr is fixed as “0x0000”.

[6:7]: Destination addr is “0x0000”, only locally connected light (i.e. light directly connected with BLE) will respond to this command.

[8]: “cmd” is “0xd7” (command code of group operation).

[9:10]: VendorID is “0x11 0x02” by default and it’s fixed.

[11]: “0x00” (parameter 0) indicates deleting a group.

[12:13]: “0x8001” indicates group number to be deleted is 0x8001 (Group1). If it’s set as “0xFFFF”, it indicates deleting all group numbers.

If light receives and correctly responds to this command, the indicating lights will blink slowly (0.5Hz) for three times, and notify response will be sent to APP to return corresponding group addr. APP can check if corresponding group is successfully deleted according to this notify response.

R: Receive

1. Write_rsp
2. Please refer to “Get_G_8” command for data format of notify response.

2.17 Kick_out

Kick light out of mesh network.

Function: Delete all parameters except for mac address of light, restore password and ltk to factory default values, and set mesh name as “out_of_mesh” or default value (“telink_mesh1”).

S: Send

Table 18 Kick_out command example

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

[1:3]: Sequence number is “0x501111”.

[4:5]: Source addr is fixed as “0x0000”.

[6:7]: Destination addr is “0x0000”, only locally connected light (i.e. light directly connected with BLE) will respond to this command.

[8]: “cmd” is “0xe3” (command code of “Kick_out”).

[9:10]: VendorID is “0x11 0x02” by default and it’s fixed.

[11]: Kick out option. If this parameter is “0” or taken out, mesh name is set as “out_of_mesh”; if this parameter is “1”, mesh name is reset to default value “telink_mesh1”.

R: Receive

Write_rsp

2.18 Get_G_8

Function: Obtain eight group numbers (only lower byte) of a light via a packet.

Since each light can be simultaneously assigned to eight groups at most, and the parameter field of response packet contains 10 bytes at most, if all group number information are reported, only five group numbers can be reported. To use “Get_G_8” command, APP must make sure all group numbers don’t exceed 0x80FF, so that only lower byte of group numbers are reported (APP will set higher byte as 0x80 automatically). Otherwise it’s needed to obtain group numbers via “Get_G_F4” and “Get_G_B4” commands instead.

It’s recommended to assign a group to group numbers (up to 8) among 0~0xff, thus a “Get_G_8” command can be used to obtain all group numbers of a light.

S: Send

Table 19 Get_G_8 command example

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

[1:3]: Sequence number is “0x601111”.

[4:5]: Source addr is fixed as “0x0000”.

[6:7]: Destination addr is “0x0000”, only locally connected light (i.e. light directly connected with BLE) will respond to this command.

[8]: “cmd” is “0xdd” (command code of “obtain group number”).

[9:10]: VendorID is “0x11 0x02” by default and it’s fixed.

[11]: “0x10” (parameter 0) indicates relay times in the mesh network after locally connected light receives this command.

[12]: “0x01” indicates the method to obtain group number is “obtain 8 group numbers of a light via a packet”. This byte is used to differ Get_G_8 from Get_G_F4 and Get_G_B4.

R: Receive

1. Write_rsp
2. Data format of notify response (notify UUID is 0x1911) to return group numbers is as shown below.

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]: Sequence number is “0x601111”. In non-encryption scenario, it’s the same as corresponding command; in encryption scenario, it’s random number.

[4:5]: Source addr is “0x0002” (plaintext), which is device addr of corresponding light. APP can check device source of the response packet accordingly.

[6:7]: 0x0002, used for encryption algorithm check. In non-encryption scenario, its value is the same as [4:5]. This parameter is negligible to APP except for decryption.

[8]: “cmd” is 0xd4 (identifier for group number notify data of Get_G_8).

[9：10]: VendorID is “0x11 0x02” by default and it’s fixed.

[11]: “0x02” indicates this light belongs to Group 0x0002 (group number identifier is 0x8002).

[12]: “0x03” indicates this light belongs to Group 0x0003 (group number identifier is 0x8003).

[13]: “0x04” indicates this light belongs to Group 0x0004 (group number identifier is 0x8004).

[14]: “0x05” indicates this light belongs to Group 0x0005 (group number identifier is 0x8005).

[15]: “0x06” indicates this light belongs to Group 0x0006 (group number identifier is 0x8006).

[16]: “0x07” indicates this light belongs to Group 0x0007 (group number identifier is 0x8007).

[17]: “0x08” indicates this light belongs to Group 0x0008 (group number identifier is 0x8008).

[18]: “0x09” indicates this light belongs to Group 0x0009 (group number identifier is 0x8009).

[19]: reserved byte, set as 0xff.

[20]: reserved byte, set as 0xff.

*Note: When group numbers of a light are less than 8, some bytes of [11:18] will be set as “0xFF” correspondingly.

2.19 Get_G_F4

Obtain former 4 group numbers (completed) of a light via a packet.

Note that get_Group_F4 should be used in combination with get_Group_B4 to obtain all 8 group numbers of a light.

S: Send

Table 20 Get_G_F4 command example

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

[1:3]: Sequence number is “0x611111”.

[4:5]: Source addr is fixed as “0x0000”.

[6:7]: Destination addr is “0x0000”, only locally connected light (i.e. light directly connected with BLE) will respond to this command.

[8]: “cmd” is “0xdd” (command code of “obtain group number”).

[9:10]: VendorID is “0x11 0x02” by default and it’s fixed.

[11]: “0x10” (parameter 0) indicates relay times in the mesh network after locally connected light receives this command.

[12]: “0x02” indicates the method to obtain group number is “obtain former 4 completed group numbers of a light via a packet”. This byte is used to differ Get_G_F4 from Get_G_8 and Get_G_B4.

R: Receive

1. Write_rsp
2. Data format of notify response (notify UUID is 0x1911) to return group numbers is as shown below.

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]：Sequence number is “0x611111”. In non-encryption scenario, it’s the same as corresponding command; in encryption scenario, it’s random number.

[4:5]：Source addr is “0x0002” (plaintext), which is device addr of corresponding light. APP can check device source of the response packet accordingly.

[6:7]： 0x0002, used for encryption algorithm check. In non-encryption scenario, its value is the same as [4:5]. This parameter is negligible to APP except for decryption.

[8]：“cmd” is 0xd5 (identifier for group number notify data of Get_G_F4).

[9:10]：VendorID is “0x11 0x02” by default and it’s fixed.

[11:12]：“0x8002” indicates this light belongs to Group 0x0002 (group number identifier is 0x8002).

[13:14]：“0x8003” indicates this light belongs to Group 0x0003 (group number identifier is 0x8003).

[15:16]：“0x8004” indicates this light belongs to Group 0x0004 (group number identifier is 0x8004).

[17:18]：“0x8005” indicates this light belongs to Group 0x0005 (group number identifier is 0x8005).

[19]：reserved byte, set as 0xff.

[20]：reserved byte, set as 0xff.

*Note: When group numbers are less than 4, some bytes of [11:18] will be set as “0xFF” correspondingly.

2.20 Get_G_B4

Obtain latter 4 group numbers (completed) of a light via a packet.

Note that get_Group_B4 should be used in combination with get_Group_F4 to obtain all 8 group numbers of a light.

S: Send

Table 21 Get_G_B4 command example

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

[1:3]：Sequence number is “0x621111”.

[4:5]：Source addr is fixed as “0x0000”.

[6:7]：Destination addr is “0x0000”, only locally connected light (i.e. light directly connected with BLE) will respond to this command.

[8]: “cmd” is “0xdd” (command code of “obtain group number”).

[9:10]: VendorID is “0x11 0x02” by default and it’s fixed.

[11]： “0x10” (parameter 0) indicates relay times in the mesh network after locally connected light receives this command.

[12]： ”0x03” indicates the method to obtain group number is “obtain latter 4 completed group numbers of a light via a packet”. This byte is used to differ Get_G_B4 from Get_G_8 and Get_G_F4.

R: Receive

1. Write_rsp
2. Data format of notify response (notify UUID is 0x1911) to return group numbers is as shown below.

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]：Sequence number is “0x621111”. In non-encryption scenario, it’s the same as corresponding command; in encryption scenario, it’s random number.

[4:5]：Source addr is “0x0002” (plaintext), which is device addr of corresponding light. APP can check device source of the response packet accordingly.

[6:7]： 0x0002, used for encryption algorithm check. In non-encryption scenario, its value is the same as [4:5]. This parameter is negligible to APP except for decryption.

[8]: “cmd” is 0xd6 (identifier for group number notify data of Get_G_B4).，

[9： 10]: VendorID is “0x11 0x02” by default and it’s fixed.

[11:12]： “0x8006” indicates this light belongs to Group 0x0006 (group number identifier is 0x8006).

[13:14]： “0x8007” indicates this light belongs to Group 0x0007 (group number identifier is 0x8007).

[15:16]：“0x8008” indicates this light belongs to Group 0x0008 (group number identifier is 0x8008).

[17:18]：“0x8009” indicates this light belongs to Group 0x0009 (group number identifier is 0x8009).

[19]：reserved byte, set as 0xff.

[20]：reserved byte, set as 0xff.

*Note: When group numbers are less than 4, some bytes of [11:18] will be set as “0xFF” correspondingly.

2.21 G1_On

Turn on lights of Group1.

Refer to “All_On” command. Note: “dst” field for “All_On” and “G1_On” are 0xffff and 0x8001, respectively.

2.22 G1_Off

Turn off lights of Group1.

Refer to “All_Off” command. Note: “dst” field for “All_Off” and “G1_Off” are 0xffff and 0x8001, respectively.

2.23 Status_All

Obtain information including luminance, ttc (time to cost) and hops of all lights.

S: Send

Table 22 Status_All command example

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

[1:3]：Sequence number is “0x511111”.

[4:5]：Source addr is fixed as “0x0000”.

[6:7]：Destination addr is “0xffff”, i.e. all lights will respond to this command.

Note: If there’re multiple lights in a mesh network, to avoid packet loss due to collision in air, it’s recommended to obtain status in the form of single light or use “online status” function.

[8]：“cmd” is “0xda” (command code of “obtain light status information”).

[9:10]: VendorID is “0x11 0x02” by default and it’s fixed.

[11]：“0x10” (parameter 0) indicates relay times in the mesh network after locally connected light receives this command.

R: Receive

1. Write_rsp
2. Data format of status notify response (notify UUID is 0x1911) is as shown below.

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]: Sequence number is “0x511111”. In non-encryption scenario, it’s the same as corresponding command; in encryption scenario, it’s random number.

[4:5]：Source addr is “0x0002”, which is device addr of corresponding light.

[6:7]：0x0002, used for encryption algorithm check. In non-encryption scenario, its value is the same as [4:5].

[8]: “cmd” is 0xdb (identifier for **status notify** data).

[9:10]：VendorID is “0x11 0x02” by default and it’s fixed.

[11]：“0xff” indicates pwm value of led1 is 0xff.

[12]：“0xff” indicates pwm value of led2 is 0xff.

[13]：“0xff” indicates pwm value of led3 is 0xff.

[14]：“0xff” indicates pwm value of led4 is 0xff.

[15]："0xff" indicates pwm value of led5 is 0xff.

[16]："0xff" indicates pwm value of led6 is 0xff.

[17]：Reserved byte, set as 0x00.

[18]：Reserved byte, set as 0x00.

[19]：0x04, ttc (time to cost) value in unit of ms. It indicates the duration when light receives this command after APP sends it. It's used to detect network delay. If it's to obtain status of the light directly connected with APP, the "ttc" is fixed as 0.

[20]：0x01, hops value. It indicates the hop number to relay this command from the light directly connected with APP to current light. If it's to obtain status of the light directly connected with APP, the "hops" is fixed as 0.

2.24 Status_G1

Obtain luminance, hop, ttc information of Group1.

Refer to "Status_All" command. Note: The "dst" field of "Status_All" and "Status_G1" are 0xffff and 0x8001, respectively.

Note: If there're multiple lights in a mesh network, to avoid packet loss due to collision in air, it's recommended to obtain status in the form of single light or use "online status" function.

2.25 User_All

Obtain customized luminance, ttc (time to cost) and hops information of all lights.

S: Send

Table 23 User_All command example

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

[1:3]：Sequence number is “0x561111”.

[4:5]：Source addr is fixed as “0x0000”.

[6:7]：Destination addr is “0xffff”, i.e. all lights will respond to this command.

Note: If there’re multiple lights in a mesh network, to avoid packet loss due to collision in air, it’s recommended to obtain status in the form of single light.

[8]：“cmd” is “0xea” (command code of “User_All” to obtain user customized information of lights).

[9:10]: VendorID is “0x11 0x02” by default and it’s fixed.

[11]：“0x10” (parameter 0) indicates relay times in the mesh network after locally connected light receives this command.

[12:20]：Parameter area to customize light status information which can be obtained from “rf_link_response_callback()” in SDK. This function can be used to define sub command of User_All.

R: Receive

1. Write_rsp
2. Data format of status notify response (notify UUID is 0x1911) is as shown below.

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]：Sequence number is “0x561111”. In non-encryption scenario, it’s the same as corresponding command; in encryption scenario, it’s random number.

[4:5]：Source addr is “0x0002”, which is device addr of corresponding light.

[6:7]：0x0002, used for encryption algorithm check. In non-encryption scenario, its value is the same as [4:5].

[8]：“cmd” is 0xeb (return information of User_All, identifier for **notify data**).，

[9:10]: VendorID is “0x11 0x02” by default and it’s fixed.

[11:20]：Data customized by user. Currently [11] is pre-defined as lower byte of device addr.

2.26 SW_Config

Demo command to configure switch and demonstrate that switch can receive command from APP. After switch receives this command, it will blink for n (configurable) times. This command is extensible as needed.

S: Send

Table 24 SW_Config command example

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

[1:3]： Sequence number is “0x581111”.

[4:5]： Source addr is fixed as “0x0000”.

[6:7]：Destination addr is “0xffff”, i.e. all switches in config mode will respond to this command.

[8]： “cmd” is “0xd3” (command code of “SW_Config”).

[9:10]: VendorID is “0x11 0x02” by default and it’s fixed.

[11]： “0x4” indicates light blinking times after switch receives this command.

R: Receive

1. Write_rsp

2.27 Time_Set

Set time (Year + Month + Day + Hour + Minute + Second) of lights.

S: Send

Table 25 Time_Set command example

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**							

[1:3]： Sequence number is “0x5a1111”.

[4:5]： Source addr is fixed as “0x0000”.

[6:7]：Destination addr is “0xffff”, i.e. all lights will respond to this command. For Time_set command, this field is set as “0xffff” generally.

[8]： “cmd” is “0xe4” (command code of “Time_Set”).

[9:10]: VendorID is “0x11 0x02” by default and it’s fixed.

[11:12]： ”0x07df” (parameter 0) indicates setting “year” as 2015.

[13]： ”0x08” indicates setting “month” as August.

[14]： ”0x06” indicates setting “day” as 6th.

[15]： ”0x09” indicates setting “hour” as 9.

[16]： ”0x00” indicates setting “minute” as 0.

[17]： ”0x00” indicates setting “second” as 0.

If setting succeeds, lights will blink slowly for three times. If setting fails, lights will blink fast for three times.

R: Receive

Write_rsp

2.28 Time_Get

Obtain time of lights.

S: Send

Table 26 Time_Get command example

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

[1:3]： Sequence number is “0x571111”.

[4:5]： Source addr is fixed as “0x0000”.

[6:7]：Destination addr is “0x0000”, only locally connected light (i.e. light directly connected with BLE) will respond to this command.

Note: If there're multiple lights in a mesh network, to avoid packet loss due to collision in air, it's recommended to obtain time in the form of single light.

[8]： "cmd" is "0xe8" (command code of "Time_Get").

[9:10]: VendorID is "0x11 0x02" by default and it's fixed.

[11]： "0x10" (parameter 0) indicates relay times in the mesh network after locally connected light receives this command.

R: Receive

1. Write_rsp
2. Data format of time notify response (notify UUID is 0x1911) to is as shown below.

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]：Sequence number is "0x571111". In non-encryption scenario, it's the same as corresponding command; in encryption scenario, it's random number.

[4:5]： Source addr is "0x0002", which is device addr of corresponding light.

[6:7]： 0x0002, used for encryption algorithm check. In non-encryption scenario, its value is the same as [4:5].

[8]： "cmd" is 0xe9 (identifier for **Time_response**).

[9:10]: VendorID is "0x11 0x02" by default and it's fixed.

[11:12]： "0x07df" indicates (parameter 0) indicates "year" is 2015.

[13]： "0x08" indicates "month" is August.

[14]： "0x06" indicates "day" is 6th.

[15]： "0x09" indicates "hour" is 9.

[16]： "0x00" indicates "minute" is 0.

[17]： "0x05" indicates "second" is 5.

[18:20]： Reserved bytes, set as "0xffffff".

2.29 Alarm clock operation commands

Alarm clock operations include five types: 0x00 (add alarm clock), 0x01 (delete alarm clock), 0x02 (modify parameters of alarm clock), 0x03 (Turn on alarm clock), 0x04 (Turn off alarm clock).

Alarm clock command format is shown as below (Please refer to code for details).

```
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

Obtain alarm clock list information.

S: Send

Table 27　Get_Alarm command example

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

[1:3]：Sequence number is “0x5b1111”.

[4:5]：Source addr is fixed as “0x0000”.

[6:7]：Destination addr is “0x0000”, only locally connected light (i.e. light directly connected with BLE) will respond to this command.

Note: If there’re multiple lights in a mesh network, to avoid packet loss due to collision in air, it’s recommended to obtain alarm clock information in the form of single light.

[8]: “cmd” is “0xe6” (command code of “Get_Alarm”).

[9:10]: VendorID is “0x11 0x02” by default and it’s fixed.

[11]: “0x10” (parameter 0) indicates relay times in the mesh network after locally connected light receives this command.

[12]: This byte indicates ID of alarm clock to be read.

If it’s 0x00, it indicates reading detailed information of all alarm clocks.

If it’s 0xFF, it indicates reading ID number of all alarm clocks. Currently up to 10 ID numbers can be returned.

If it’s 0x01~0x7F, it indicates reading information of corresponding ID. If it’s available for the light, complete data of this alarm clock information will be returned, and its total alarm clock number column is marked as 1. If it’s not available for the light, data of all zero will be returned (the 10 bytes, i.e. Byte 11~20 contained by the notify response to the obtained alarm clock information, are all 0). It can be used to check if alarm clock is successfully deleted for Alarm_Del.

R: Receive

1. Write_rsp
2. Data format of notify response (notify UUID is 0x1911) to return alarm clock information is as shown below.

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]：Sequence number is “0x621111. In non-encryption scenario, it’s the same as corresponding command; in encryption scenario, it’s random number.

[4:5]： Source addr is “0x0002”, which is device addr of corresponding light.

[6:7]： 0x0002, used for encryption algorithm check. In non-encryption scenario, its value is the same as [4:5].

[8]： “cmd” is 0xe7 (identifier for **alarm_response**).

[9:10]: VendorID is “0x11 0x02” by default and it’s fixed.

[11]： ”0xa5” (parameter 0) indicates the alarm clock data are valid.

[12]： Index number of alarm clock.

[13]： Refer to Alarm_Add [13].

[14:18]： Refer to Alarm_Add [14:18].

[19]： Scene index of alarm clock (valid for scene type alarm clock).

[20]： Alarm clock number of the light. If notify number received by APP doesn’t match this value, APP should re-send a Get_Alarm command.

Note: If [11: 19] are all “0x00”, it indicates the light has no alarm clock data currently.

2.29.2 AlarmOff_Add

Add an alarm clock (The light will be turned off when the alarm clock expires.)

Table 28 AlarmOff_Add command example

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**					

Please refer to “AlarmSce_Add”.

2.29.3 AlarmOn_Add

Add an alarm clock (The light will be turned on when the alarm clock expires.)

Table 29 AlarmOn_Add command example

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**					

Please refer to "AlarmSce_Add".

2.29.4 AlarmSce_Add

Add an alarm clock (The light will switch to specified scene mode when the alarm clock expires.)

S: Send

Table 30 AlarmSce_Add command example

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**					

[1:3]： Sequence number is "0x5c1111".

[4:5]： Source addr is fixed as "0x0000".

[6:7]： Destination addr is "0xffff", i.e. all lights will respond to this command.

[8]： "cmd" is "0xe5" (command code of "Alarm operation").

[9:10]: VendorID is "0x11 0x02" by default and it's fixed.

[11]： This byte indicates alarm operation type is "0x00 (Add alarm clock)".

[12]： This byte indicates alarm clock index number (1~16), which can be determined by get_next_shedule_idx() in VC code.

When APP wants to assign alarm clock index number for light automatically, first set initial index number as 0, then light will automatically assign following index numbers according to get_next_shedule_idx() as 1~16. Then APP can send an Alarm_Get command to read the index number.

[13]： Bit combination parameter. Refer to structure alarm_ev_t.par1.

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

- ✧ bit0~bit3：The four bits indicate light action when alarm clock expires. Three actions are defined currently: 0-off, 1-on, 2-scene.
- ✧ bit4~bit6：The three bits indicate alarm clock type. Two types are defined currently: 0-DAY type, 1-WEEK type.
- ✧ bit7：Alarm clock enabling bit. 1-enable. APP should set this bit as “1” in “Alarm_Add”, so that the alarm clock is enabled by default.

[14]：If alarm clock is DAY type, this byte indicates month (1~12). If alarm clock is WEEK type, this byte is reserved.

[15]：If alarm clock is DAY type, this byte indicates day of a month. If alarm clock is WEEK type, this byte indicates day of a week. (Bit0~bit6 indicate Sunday, Monday ~ Saturday, successively. Bit7 is reserved and should be set as 0 to legitimize this parameter.)

[16]：Hour.

[17]：Minute.

[18]：Second.

[19]：Scene id.

2.29.5 Alarm_Del

Delete alarm clock.

S: Send

Table 31 Alarm_Del command example

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**						

[1:3]：Sequence number is “0x5d1111”.

[4:5]：Source addr is fixed as “0x0000”.

[6:7]：Destination addr is “0xffff”, i.e. all lights will respond to this command.

[8]：“cmd” is “0xe5” (command code of “Alarm operation”).

[9：10]: VendorID is “0x11 0x02” by default and it’s fixed.

[11]：This byte indicates alarm operation type is “0x01 (Delete alarm clock)”.

[12]：This byte indicates alarm clock index number (1~16).

“0xff” indicates deleting all alarm clock information of corresponding light.

[13:18]：Reserved bytes, set as 0.

R: Receive

1. Write_rsp

2.29.6 Alarm_En

Turn on alarm clock.

S: Send

Table 32 Alarm_En command example

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**						

[1:3]：Sequence number is “0x601111”.

[4:5]：Source addr is fixed as “0x0000”.

[6:7]： Destination addr is “0xffff”, i.e. all lights will respond to this command.

[8]： “cmd” is “0xe5” (command code of “Alarm operation”).

[9： 10]: VendorID is “0x11 0x02” by default and it’s fixed.

[11]： This byte indicates alarm operation type is “0x03 (Turn on alarm clock)”.

[12]： This byte indicates alarm clock index number (1~16).

[13:18]： Reserved bytes, set as 0.

R: Receive

1. Write_rsp

2.29.7 Alarm_Dis

Turn off alarm clock.

S: Send

Table 33 Alarm_Dis command example

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**						

[1:3]： Sequence number is “0x611111”.

[4:5]： Source addr is fixed as “0x0000”.

[6:7]： Destination addr is “0xffff”, i.e. all lights will respond to this command.

[8]： “cmd” is “0xe5” (command code of “Alarm operation”).

[9:10]: VendorID is “0x11 0x02” by default and it’s fixed.

[11]： This byte indicates alarm operation type is “0x04 (Turn off alarm clock)”.

[12]： This byte indicates alarm clock index number (1~16).

[13:18]： Reserved bytes, set as 0.

R: Receive

1. Write_rsp

2.29.8 Alarm_Chg

Modify alarm clock information.

S: Send

Table 34 Alarm_Chg command example

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**						

[1:3]： Sequence number is “0x641111”.

[4:5]： Source addr is fixed as “0x0000”.

[6:7]： Destination addr is “0xffff”, i.e. all lights will respond to this command.

[8]： “cmd” is “0xe5” (command code of “Alarm operation”).

[9:10]： VendorID is “0x11 0x02” by default and it’s fixed.

[11]： This byte indicates alarm operation type is “0x02 (Modify parameters of alarm clock)”.

[12]： This byte indicates alarm clock index number (1~16).

[13:18]： Alarm parameter setting, please refer to “Alarm_Add” command.

R: Receive

1. Write_rsp

Note: After alarm clock operation (Add, Delete, Turn on/off, Modify) is executed, APP can read alarm clock information via “Alarm_Get” command to check if setting is successful.

2.30 Scene mode operation commands

2.30.1 Add_Scene

Add a scene.

S: Send

Table 35 Add_Scene command example

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**								

[1:3]: Sequence number is “0x661111”.

[4:5]: Source addr is fixed as “0x0000”.

[6:7]: Destination addr is “0x0000”, only locally connected light (i.e. light directly connected with BLE) will respond to this command.

[8]: “cmd” is “0xEE” (command code of scene operation).

[9:10]: VendorID is “0x11 0x02” by default and it’s fixed.

[11]: This byte indicates scene operation type is “0x01 (Add scene)”.

[12:16]: Packet of a scene (reserved 3-byte data not included). Please refer to scene data structure as below.

Scene data structure

```
typedef struct{
  u8 id;                //scene data index number
  u8 lum;               //luminance value
  u8 rgb[3];            //RGB values of light
  u8 rsv[3];            //reserved data
}scene_t;
```

2.30.2 Del_Scene

Delete scene.

S: Send

Table 36 Del_Scene command example

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

[1:3]: Sequence number is “0x691111”.

[4:5]: Source addr is fixed as “0x0000”.

[6:7]: Destination addr is “0x0000”, only locally connected light (i.e. light directly connected with BLE) will respond to this command.

[8]: “cmd” is “0xEE” (command code of scene operation).

[9:10]: VendorID is “0x11 0x02” by default and it’s fixed.

[11]: This byte indicates scene operation type is “0x00 (Delete scene)”.

[12]: Scene index number. If the byte is “0xff”, all scene modes will be deleted.

2.30.3 Load_Scene

Load scene mode.

S: Send

Table 37 Load_Scene command example

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

[1:3]: Sequence number is “0x6C1111”.

[4:5]: Source addr is fixed as “0x0000”.

[6:7]: Destination addr is “0xffff”, i.e. all lights will respond to this command.

[8]: “cmd” is “0xEF” (command code of “Load_scene” operation).

[9:10]: VendorID is “0x11 0x02” by default and it’s fixed.

[11]: This byte indicates scene index number.

2.30.4 Get_Scene

Obtain scene data list.

S: Send

Table 38 Get_Scene command example

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

[1:3]: Sequence number is “0x6e1111”.

[4:5]: Source addr is fixed as “0x0000”.

[6:7]: Destination addr is “0x0000”, only locally connected light (i.e. light directly connected with BLE) will respond to this command.

Note: If there’re multiple lights in a mesh network, to avoid packet loss due to collision in air, it’s recommended to obtain scene data in the form of single light.

[8]: “cmd” is “0xc0” (command code of “Get_Scene”).

[9:10]: VendorID is “0x11 0x02” by default and it’s fixed.

[11]: “0x10” (parameter 0) indicates relay times in the mesh network after locally connected light receives this command.

[12]: This byte indicates ID of scene to be read.

If it’s 0x00, it indicates reading detailed information of all scenes.

If it’s 0xFF, it indicates reading ID number of all scenes. Currently up to 10 ID numbers can be returned.

If it’s 0x01~0x7F, it indicates reading information of corresponding ID. If it’s available for the light, complete data of this scene information will be returned, and its total scene number column is marked as 1. If it’s not available for the light, data of all zero will be returned (the 10 bytes, i.e. Byte 11~20 contained by the notify response to the obtained scene, are all 0). It can be used to check if scene is successfully deleted for Del_Scene.

R: Receive

1. Write_rsp
2. Data format of scene notify response (notify UUID is 0x1911) is as shown below.

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]：Sequence number is “0x6e1111”. In non-encryption scenario, it’s the same as corresponding command; in encryption scenario, it’s random number.

[4:5]： Source addr is “0x0055”, which is device addr of corresponding light.

[6:7]： 0x0055, used for encryption algorithm check. In non-encryption scenario, its value is the same as [4:5].

[8]： “cmd” is 0xc1 (identifier for **scene_response**).

[9:10]: VendorID is “0x11 0x02” by default and it’s fixed.

[11]： Scene index number.

[11:18]： A completed scene packet. Please refer to scene data structure of “Add_Scene” command.

[19]： Scene number of the light. If scene notify number received by APP doesn’t match this value, APP should re-send a Get_Scene command.

[20]： Reserved byte, set as 0x00.

Note: If [11:19] are all “0x00”, it indicates the light has no scene data currently.

3 NotifyStatus（NotifyStatus UUID 1911）

3.1 MeshLightStatus

After online status function is enabled, when light is powered on for the first time, empty device address will be set as lower 8bits of mac address (lower byte)+ 0x00 (higher byte). If multiple lights are added to the same network, and there's device address overlapped, APP needs to configure a unique device address among 0x0001~0x00FF so that more user data can be reported.

Online_All

Enable online status function. Send the command below to AttHandle 0x12 (corresponding UUID of 0x1911) via write_req command.

S: Send

1
01

VC example:

[1]：”0x01” indicates enabling MeshLightStatus function.

R: Receive

Write_rsp

After MeshLightStatus function is enabled, each light in current network will report a notify data first to APP to display its status. Whenever status (luminance value, on/off state, online/offline state, etc.) of some light changes, another notify data will be reported to APP to update its status display.

The reported notify data format is as shown below.

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]：Sequence number is “0x000000”. In non-encryption scenario, it's set as 0 currently. In encryption scenario, it’s random number.

[4:5]： Source addr is “0x0000” currently.

[6:7]： dst (determination number). In non-encryption scenario, it’s set as 0 currently. In encryption scenario, it’s used for encryption check.

[8]： 0xdc (identifier for LightStatus reported data).

[9:10]： 0x0211. Vendor ID.

[11:14]&[15:18]: Reported data of two lights.

[11]："0x11” indicates device addr of the first light. Since it’s needed to manually configure device address of light node among 0x0001~0x00FF to use online status function, only lower byte of device address is reported.

[12]： "0x3c” indicates serial number (sn) of data reported by the first light. If sn is 0, it indicates the light node is at offline state (e.g. power off, out of range); if sn is value rather than 0, it indicates the light is at online state.

[13]： This byte (0~100) indicates luminance value of the first light.

[14]： Byte reserved for customer use. Its default value is “0xFF”.

[15]： "0x22” indicates device addr (lower byte) of the second light.

[16]： "0x4B” indicates serial number (sn) of data reported by the second light. If sn is 0, it indicates the light node is at offline state (e.g. power off, out of range); if sn is value rather than 0, it indicates the light is at online state.

[17]：This byte (0~100) indicates luminance value of the second light.

[18]： Byte reserved for customer use. Its default value is “0xFF”.

Note: If [15:18] is “0x00000000”, it indicates only a light reported data. Generally, only after Online_All command is just sent, the reported “online status” data contains data of two lights; otherwise only a light reports data.

[19]： Reserved byte, set as 00.

[20]： Reserved byte, set as 00.

3.2 User notify data

Light in BLE connection state can actively send notify data via calling “light_notify(u8 *p, u8 len)” interface function in SDK.

User notify data reported data format is as shown below.

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). In non-encryption scenario, it’s set as 0 currently. In encryption scenario, it’s random number.

[4:5]： Source addr is “0x0000” currently.

[6:7]： dst (determination number). In non-encryption scenario, it’s set as 0 currently. In encryption scenario, it’s used for encryption check.

[8]： 0xEA (Identifier for User notify data reported data).

[9:10]： 0x0211. Vendor ID.

[11]： It’s used as accumulating counter currently, and it’s customizable.

[12:20]： User customized data. Default value is 0.

4 MeshLightOTA（MeshLightOTA UUID 1913）

S: Send

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 example:

[1:2]：”0x0001” indicates index of firmware packet sequence which starts from 0.

[3:18]：16-byte firmware data.

[19:20]：CRC16 check value of data sequence [1:18]. Please refer to source code for CRC algorithm.

Generally, a packet loads 16-byte firmware. Exception: Penultimate packet (firmware ending part) contains data less than or equal to 16bytes. Final packet only contains idex and CRC check value (0 byte firmware) to indicate the end of OTA.

Please refer to “unsigned short crc16 (unsigned char *pD, int len)” in VC code (annotation, only for APP copy) or SDK for details on how to obtain CRC16 check value.

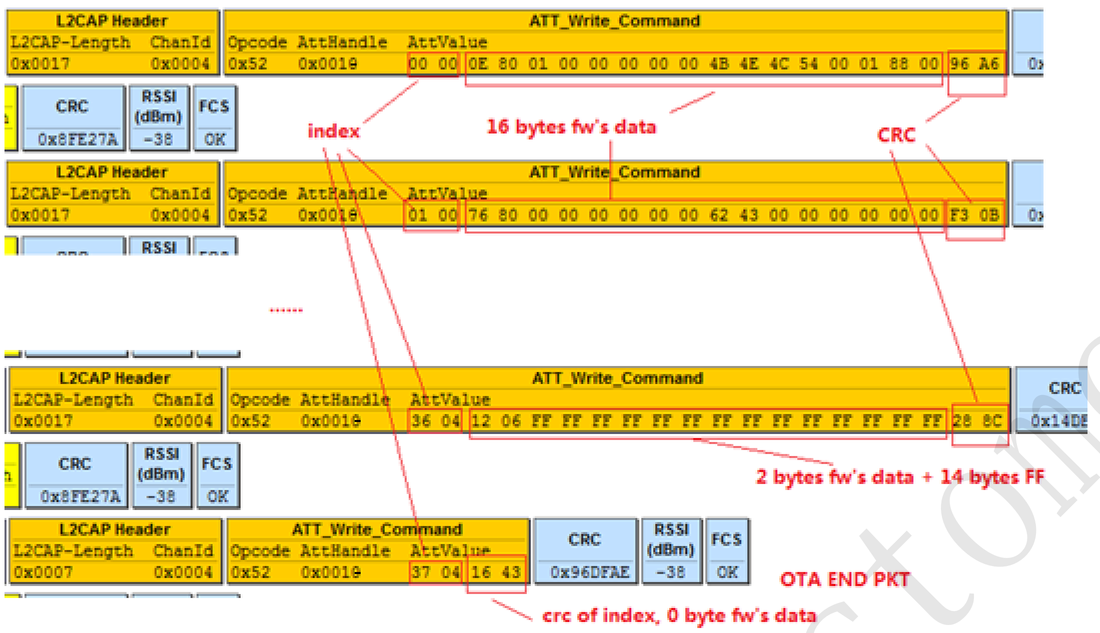
unsigned short crc16 (unsigned char *pD, int len) parameters:

pD = Address of first byte in AttValue (index starting address)

len = index length (=2) + (data length between index and crc）

E.g. len of the following four packets are 18, 18, 18 and 2, respectively.

R: Receive NULL



```
L2CAP Header                ATT_Write_Command
L2CAP-Length  ChanId        Opcode  AttHandle  AttValue
0x0017        0x0004        0x52    0x0019     00 00 0E 80 01 00 00 00 00 00 4B 4E 4C 54 00 01 88 00 96 A6

CRC         RSSI (dBm)   FCS
0x8FE27A    -38          OK

L2CAP Header                ATT_Write_Command
L2CAP-Length  ChanId        Opcode  AttHandle  AttValue
0x0017        0x0004        0x52    0x0019     01 00 76 80 00 00 00 00 00 00 62 43 00 00 00 00 00 00 F3 0B

......

L2CAP Header                ATT_Write_Command                                                                   CRC
L2CAP-Length  ChanId        Opcode  AttHandle  AttValue
0x0017        0x0004        0x52    0x0019     36 04 12 06 FF FF FF FF FF FF FF FF FF FF FF FF FF FF 28 8C   0x14DE

CRC         RSSI (dBm)   FCS
0x8FE27A    -38          OK

L2CAP Header                ATT_Write_Command             CRC        RSSI (dBm)   FCS
L2CAP-Length  ChanId        Opcode  AttHandle  AttValue
0x0007        0x0004        0x52    0x0019     37 04 16 43   0x96DFAE   -38          OK
```

index
16 bytes fw's data
CRC
2 bytes fw's data + 14 bytes FF
OTA END PKT
crc of index, 0 byte fw's data

APP OTA design procedure:

1. Light side: Demo code on light side contains OTA function which can be used directly (no need to modify).
2. APP side:

   Brief: Send firmware to be upgraded according to the requirement in this section.

   Design procedure reference:

   1) Copy firmware to be upgraded to local APP.
   2) Obtain firmware total size (25th ~28th byte value of firmware).
   3) Read 16-byte data starting from firmware addr=0, assemble data into command packet, set index as 0 and calculate crc, assemble index and crc into command packet (note that storage structure adopts little-Endian format), then send the packet.
   4) Check in callback function if BLE data is sent successfully.
      - ✧ If success: Continue to send the next packet, i.e. read 16-byte data starting from firmware addr=addr+16, assemble data, set index = index +1 and calculate crc, assemble index and crc, then send this packet.

- ✧ If failure: Re-send this packet.

5) Repeat step 4 until all data of firmware are sent.

Note: If final firmware packet contains less than 16bytes firmware data, a) "len" of the packet is still 16, and empty locations are set as "0xff"; b) modify "len".

6) Send the final packet which contains idex and CRC check value (0 byte firmware). OTA transmission procedure is finished.

User can add timeout check to avoid unexpected error, i.e. if OTA process is not finished after the preset duration, connection is disconnected, and OTA fails due to timeout.

5 MeshLightPair（MeshLightPair UUID 1914）

To simplify debugging, “rands” (random number by default) can be set as fixed via setting rands_fix_flag as 1 in ending part of user_init().

After this function is enabled, user can compare encrypted data of a command sent by APP with data sent by VC: if matched, it indicates encryption is OK; otherwise it’s needed to check if there’s error for encryption parameters.

Data encryption parameters:

```
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]: fixed as 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];    //Obtain from transmitted command, the 1st byte after sno
sec_ivs [7] = src[1];    // Obtain from transmitted command, the 2nd byte after sno
```

6 Factory Reset

User can follow power on sequence as follows to trigger factory reset function (i.e. restore light to factory default setting).

The factory reset trigger method is introduced according to demo code, and time parameters are configurable.

Step1: Power on light, then power down it within 3s.

Step2: Power on light, then power down it within 3s.

Step3: Power on light, then power down it within 3s.

Step4: Power on light, then power down it between 3s and 30s (3s and 30s not included).

Step5: Power on light, then power down it between 3s and 30s (3s and 30s not included).

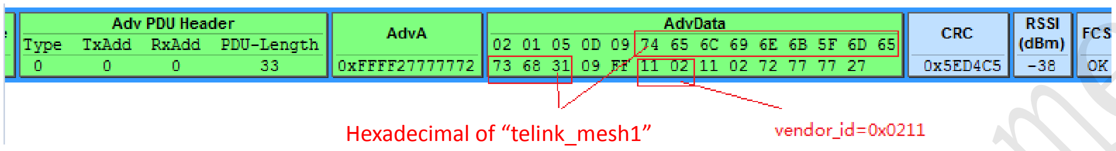
Step6: Power on light, it will blink for 3 times with frequency of 0.5Hz. Factory Reset function is triggered, and all parameters configured by user will be reset to factory default values.

User can modify the array below to configure time parameters for power on sequence.

```
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 Filtering

During APP development, it’s needed to filter adv packet of non-predefined mesh_name and verdor id in BLE device scan list. The mesh_name and vendor_id are predefined as “telink_mesh1” and “0x0211” respectively, and their values can be obtained from adv packet.



Adv PDU Header				AdvA	AdvData	CRC	RSSI (dBm)	FCS
Type	TxAdd	RxAdd	PDU-Length		02 01 05 0D 09 74 65 6C 69 6E 6B 5F 6D 65			
0	0	0	33	0xFFFF27777772	73 68 31 09 FF 11 02 11 02 72 77 77 27	0x5ED4C5	-38	OK