

ST Bluetooth Mesh Light LC Server model

Introduction

This document describes the implementation of Light Lightness Control (LC) Server model, its dependence on other models, features, configuration, APIs, etc. Prerequisite is a basic knowledge of Bluetooth mesh and Bluetooth mesh lighting models.

Light Lightness Control (LC) Server model enables sensor driven automated lighting control in Bluetooth mesh networks.

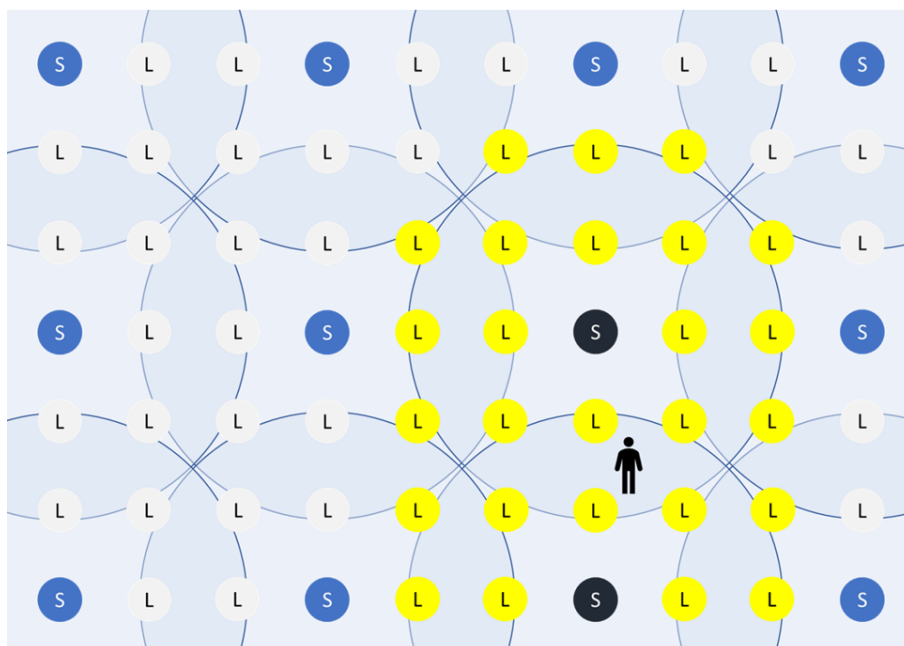
Each light node supported by Light LC Server model has its own light controller, which is implemented as a software state machine. Light controller processes incoming messages from sensor nodes (for example, occupancy sensor) and sets the light level accordingly.

The figure below shows a floor with a Bluetooth mesh network, enabled with lighting control system. When someone enters this area, the system switches the lights on in his/her vicinity. This is implemented by using distributed sensor nodes and light nodes, which are configured to process incoming messages from nearby sensor nodes.

In the figure below, the circle represents the occupancy detection range of a sensor node (S) whose center is represented by S itself. This circle also includes all the lights (L), which support Light LC Server model, controlled by their inbuilt light controllers. The controllers receive a sensor occupancy message from the sensor, which is at the center of its circle, and the corresponding light turns on.

Different sensors can control a light node for occupancy detection. Moreover, light functionality and sensor functionality can coexist on the same node.

Figure 1. Floor with light controllers for occupancy detection



1 Features

ST Bluetooth Mesh Light LC Sever model enables all the functions of Light LC and other lighting models. It also can help you to configure several parameters.

The main features are:

- decentralized lighting control architecture;
- automatic light control with the help of occupancy sensors;
- manual light control (overriding automatic operation) via switches or apps;
- mechanism to mitigate popcorn effect due to over-the-air packet loss;
- smooth turning on and off of the lights;
- configurable light output in various light controller states;
- configurable timing for light turning on duration;
- support for ambient light sensors for adaptive lighting and daylight harvesting.

2 Structure and dependencies

A lighting node enabled with Light LC Server model is a Bluetooth mesh node with two elements.

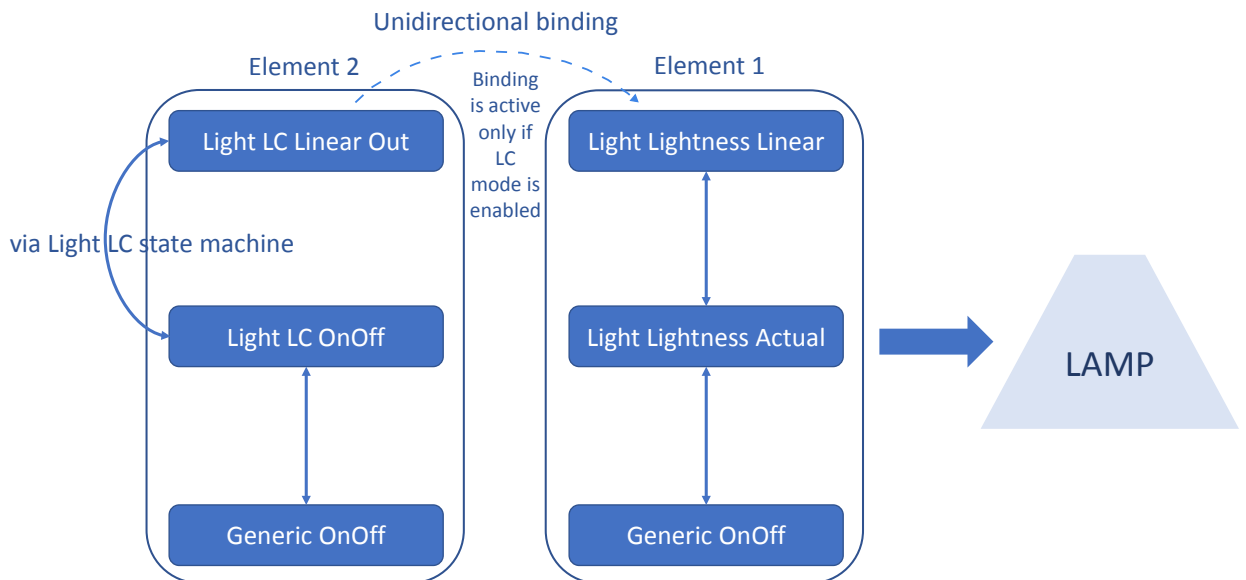
The node supports a hierarchy of several other models. This support for various server models enables the node control through simple switches as well as powerful consoles.

2.1 Model state relation

The figure below shows important mesh model states supported by a lighting controller node on two elements: Element 1 and Element 2. Element 1 is the main element that controls the lamp brightness. The light controller sets the Light LC Linear Out state of the Light LC Server model on Element 2.

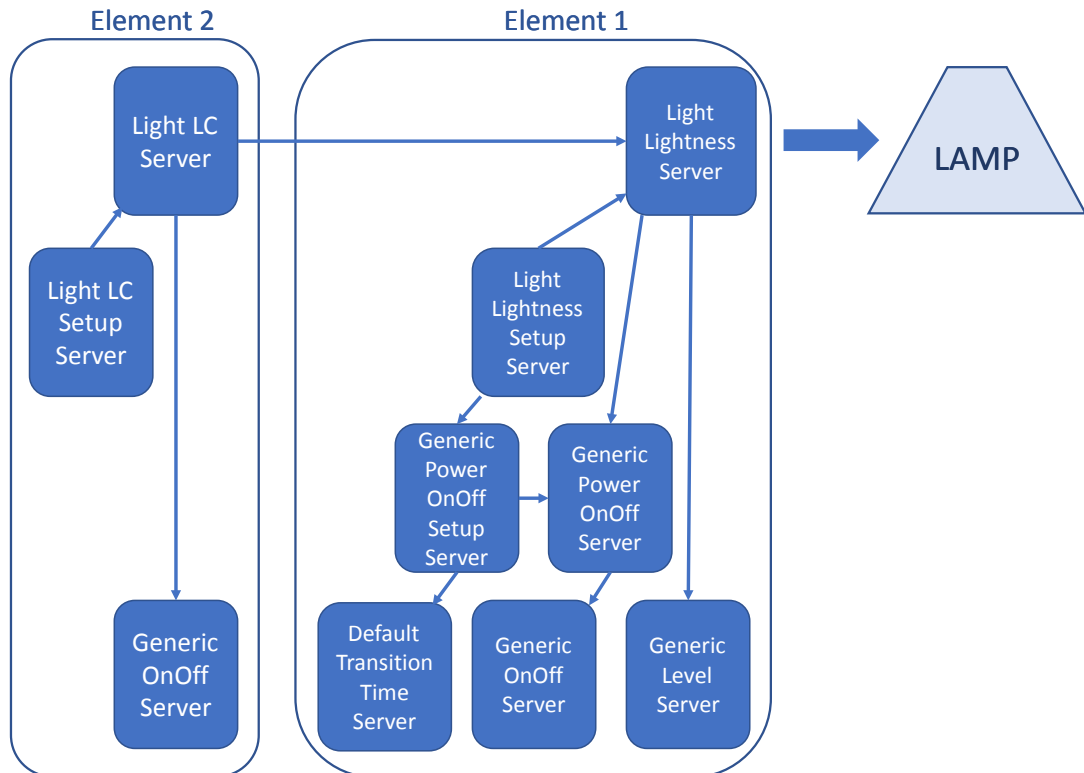
Light LC Linear Out state is conditionally bound to the Light Lightness Linear state of the Light Lightness model on Element 1. The binding of these two states breaks if Element 1 tries to set any of its state independently of Element 2. Thus, a change in Light LC Linear Out state (on Element 2) triggers a change in Light Lightness Linear state (on Element 1) but not vice versa. If Light Lightness Linear state changes without a change in Light LC Linear Out state, the two states are not bound anymore to each other. Consequently, the light controller enters OFF state (see Section 3.4 for controller states). If the lighting controller is required to control a lamp in the future, a dedicated light LC client message (Light LC Mode Set) is required to enable the controller.

Figure 2. Relation among the states of lighting controller in a Bluetooth mesh node



2.2 Server model relation

Lighting control node supports various mesh models on different elements as shown in the figure below.

Figure 3. Relation among the models of lighting controller in a Bluetooth mesh node


- On Element 1:
 - Light Lightness Server extends Generic Power OnOff Server and Generic Level Server
 - Light Lightness Setup Server extends Light Lightness Server
 - Generic Power OnOff Server extends Generic OnOff Server
 - Generic Power OnOff Setup Server extends Generic Power OnOff Server and Default Transition Timer Server
 - Default Transition Timer Server is root model (it does not extend any other model)
 - Generic Level Server is root model (it does not extend any other model)
 - Generic OnOff Server is root model (it does not extend any other model)
- On Element 2:
 - Light LC Server extends Generic OnOff Server on the same element and Light Lightness Server on a different element
 - Light LC Setup Server extends Light LC Server

2.3 Messages supported by a Light LC node

As explained in [Section 2.2](#) , different switches, controllers, or sensors can control a light output:

- a switch supporting Generic OnOff Client model with publish address as Element 1 address
 - Generic OnOff Set
 - Generic OnOff Set Unacknowledged
- a slider supporting Generic Level Client model with publish address as Element 1 address
 - Generic Level Set
 - Generic Level Set Unacknowledged
 - Generic Delta Set
 - Generic Delta Set Unacknowledged
 - Generic Move Set
 - Generic Move Set Unacknowledged

- A switch or a slider supporting Light Lightness Client model with publish address as Element 1 address
 - Light Lightness Set
 - Light Lightness Set Unacknowledged
 - Light Lightness Linear Set
 - Light Lightness Linear Set Unacknowledged
- a switch supporting Generic OnOff Client model with publish address as Element 2 address
 - Generic OnOff Set
 - Generic OnOff Set Unacknowledged
- a switch supporting Light LC Client model with publish address as Element 2 address
 - Light LC Light OnOff Set
 - Light LC Light OnOff Set Unacknowledged
- a sensor node supporting Sensor Server model with publish address as Element 2 address:
 - sensor status with the following sensor PIDs for occupancy detection
 - Motion Sensed Property
 - People Count Property
 - Presence Detected Property
 - Sensor Status with following PID for daylight harvesting
 - Present Ambient Light Level Property

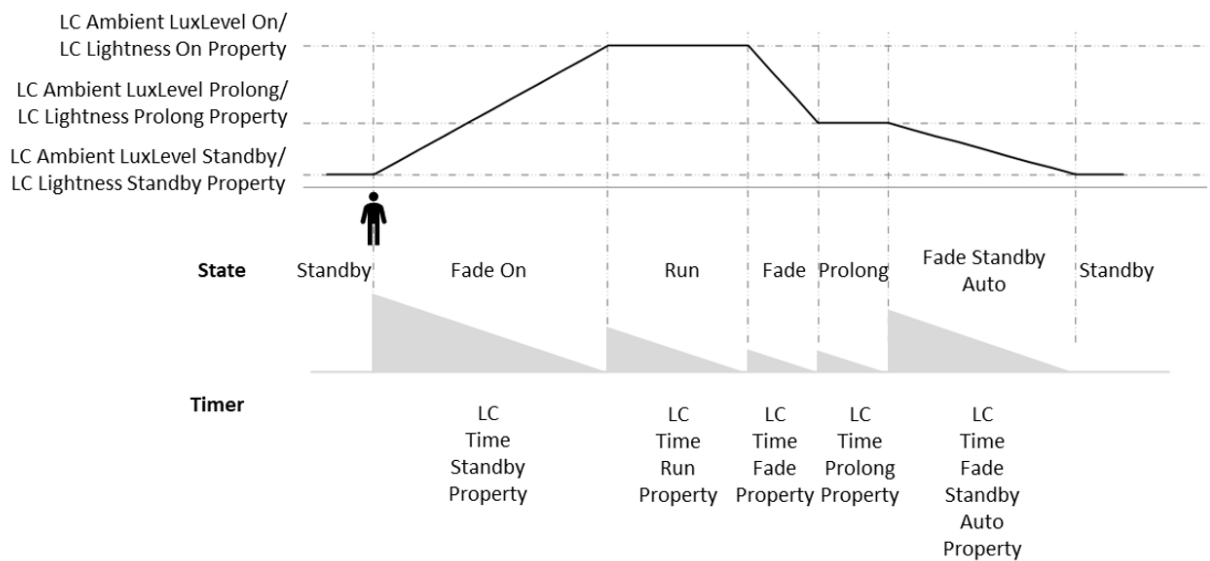
3 Light lightness controller

Light lightness controller is implemented as a state machine, which handles all the inputs from switches, consoles, sensors, timers, etc. It provides Light LC Linear Out state as output.

Light LC Linear Out state is bound to Light Lightness Linear state (Section 2.1).

The figure below shows the typical controller response upon occupancy detection.

Figure 4. Light lightness controller behaviour



The vertical axis represents the controller output in terms of lightness level and lux level (Section 3.2). For example, when the controller is in standby, the output lightness level is Lightness Standby and lux level is LuxLevel Standby. The reference values, as set by the client, are represented by Light Control Lightness Standby property and Light Control Ambient LuxLevel Standby property, respectively.

The horizontal axis represents time. The time spent by the controller in different states is represented by the corresponding time property values. For example, the duration of fade state is represented by Light Control Time Fade property. As soon as the controller enters a state, the controller timer is initialized with the corresponding time property value and counts down to zero. Refer to Section 3.1 for transition timer behavior, states and events.

An administrator or the build management system can set all Light Control property values using Light LC Client model message.

Light output varies from initial level to final level through different states: Fade On, Fade, Fade Standby Auto and Fade Standby Manual. While in other states (Standby, Run, and Prolong), the light output is constant. The controller cannot determine the light output if it is in Off state (Section 2.1).

The change of light level occurs in a finite number of steps and with a given step resolution (see the figures of Section 3.5 showing light level change in steps). The total time taken during the transition is equal to the number of steps multiplied for step resolution.

The transition time is defined by the corresponding property values. It could be also transmitted as optional parameter in Light OnOff message.

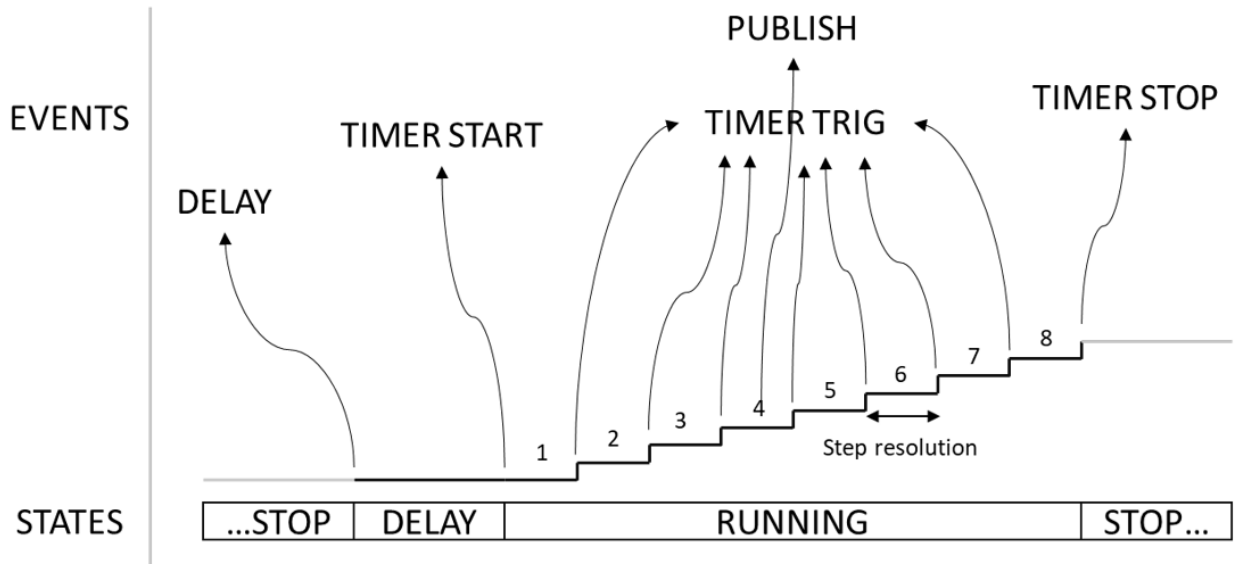
If the transition parameters are received as optional parameters in Light LC OnOff message, the total transition time can be computed using the number of steps and the step resolution.

If optional parameters are absent, the total transition time is set as the corresponding property value. For example, in Fade On state, the total transition time is Light LC Control Time Fade On. In this case, the number of steps is not shown. The number of steps can be very large (with a very small step resolution) or steps can be very few (with a very large step resolution). At each step, it is necessary to compute transition parameters and the intermediate state value, followed by PI regulator, to adjust the light level. In this case, the step resolution is set by LIGHT_LC_DEFAULT_TRANSITION_RES_MS in the SDK.

3.1 Transition timer

The timer is an important part of Light LC state machine. The light lightness controller uses transition state machine to handle state transitions. The figure below represents the states and events of transition timer during state transition.

Figure 5. Transition state machine behavior



The events received from the transition state machine are used as inputs for the light lightness controller. If a property value is modified while it is used, the change is not reflected until its next usage. If the delay is 0 and the transition time is 0, the state transition immediately happens. For example, in the figure above, as soon as the controller detects occupancy, the Light Control Time Fade On property value is 0 and the Light Control Time Occupancy delay is 0, the Light LC state goes from Standby to Run, and the light output is set according to property values of Run state.

Table 1. Transition states

State	Description
TRANSITION_STATUS_STOP	Transition timer stopped or not running
TRANSITION_STATUS_DELAY	Transition state machine is in delay state. After delay time is passed, it moves to running state
TRANSITION_STATUS_RUNNING	Transition timer is running

Table 2. Transition events

Event	Description
TRANSITION_EVENT_ABORT	Aborted
TRANSITION_EVENT_DELAY	Delay period started
TRANSITION_EVENT_TIMER_START	Transition started with timer duration = transition time
TRANSITION_EVENT_TIMER_TRIG	Timer triggered event is generated at each step (see Figure 5)
TRANSITION_EVENT_PUBLISH	Publish event indicates state publishing during transition. It happens after 1 second is elapsed in TRANSITION_STATUS_RUNNING if transition time > 2 s
TRANSITION_EVENT_TIMER_STOP	Stop event indicates that transition is completed and timer is stopped

RELATED LINKS

[Section 7.4.1.3 of Mesh Model Specification v.01.1](#)

3.2 Proportional integral feedback regulator

Proportional integral feedback regulator is part of the light lightness controller. It features daylight harvesting. Ambient light sensors provide the ambient lux level given to PI block to keep light level within a configured value.

RELATED LINKS

[Section 6.2.6 of Mesh Model Specification v.01.1](#)

[Section 9.4 of the white paper "Building a Sensor-Driven Lighting Control System Based on Bluetooth Mesh v1.0"](#)

[For Light LC mode, refer to Section 6.2.3.1 of Mesh Model Specification v1.0.1](#)

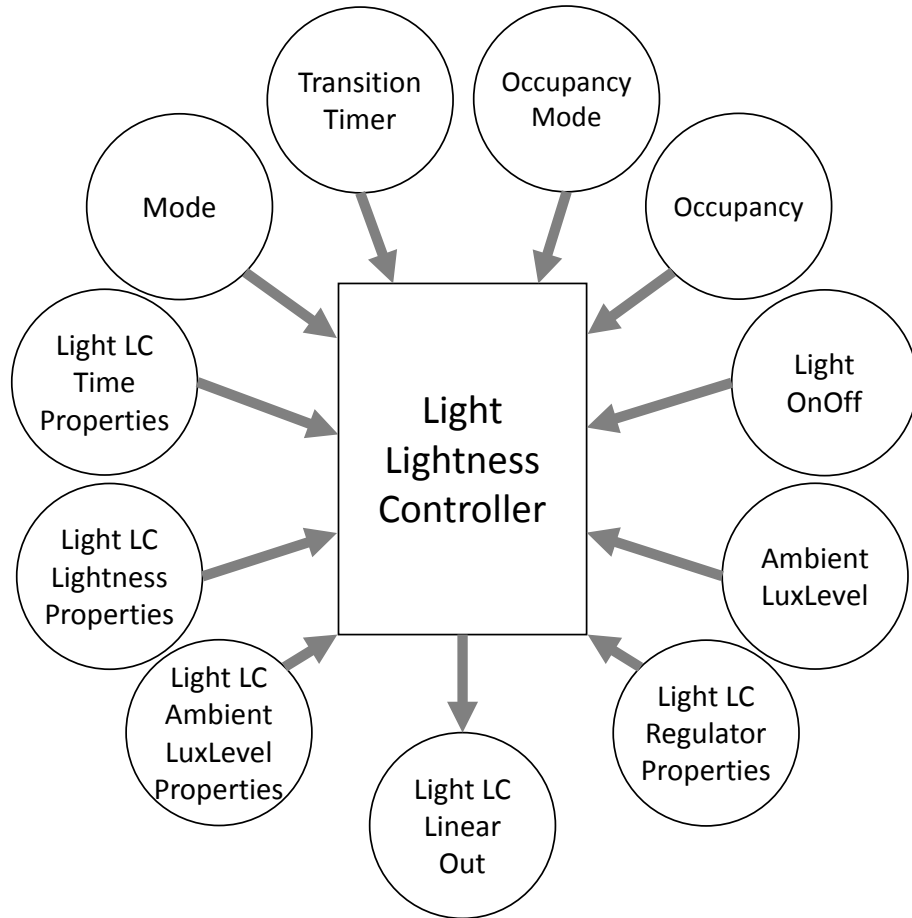
[For Occupancy mode, refer to Section 6.2.3.2 of Mesh Model Specification v1.0.1](#)

3.3 Light lightness controller blocks

Light LC state machine has multiple inputs:

- Light LC mode: indicates if the controller is enabled or disabled
- Transition timer: used to update light level and controller state
- Occupancy mode: indicates the controller transition from Standby state when occupancy is detected
- Occupancy: detected by the motion sensor, which exists on the same element or on a different element, or on both
- Light LC Light OnOff Set message: sent by a switch or app whose element (supporting Light LC Client model or Generic OnOff Client model) publishes OnOff message with publish address as Light LC Server model element address
- Ambient LuxLevel: is used by the PI regulator and represents the ambient lux level and is detected by the ambient light sensor, which exists on the same element or on a different element, or on both
- Light LC properties:
 - Light LC Time Occupancy Delay
 - Light LC Time Fade On
 - Light LC Time Run
 - Light LC Time Fade
 - Light LC Time Prolong
 - Light LC Time Fade Standby Auto
 - Light LC Time Fade Standby Manual
 - Light LC Lightness On
 - Light LC Lightness Prolong
 - Light LC Lightness Standby
 - Light LC Ambient LuxLevel On
 - Light LC Ambient LuxLevel Prolong
 - Light LC Ambient LuxLevel Standby
 - Light LC Regulator Kiu
 - Light LC Regulator Kid
 - Light LC Regulator Kpu
 - Light LC Regulator Kpd
 - Light LC Regulator Accuracy

The output of the Light LC state machine is Light LC Linear Out state. It is bound to Light Lightness Linear (Section 2.1) to control the light level.

Figure 6. Light lightness controller inputs and outputs


3.4 Controller states and events

Table 3. Controller states

Light LC state	Description
OFF	In this state, the controller does not control the light level
STANDBY	The controller is ready to process light on message or occupancy message from sensor
POST_STANDBY	Temporary state to handle delays
PRE_FADE_ON	Temporary state to handle delays
FADE_ON	Light level transition from Standby to Run
RUN	Light level is constant
FADE	Light level transition from Run to Prolong
PROLONG	Light level is constant
FADE_STANDBY_AUTO	Light level transition from Prolong to Standby
PRE_FADE_STANDBY_MANUAL	Temporary state to handle delays
FADE_STANDBY_MANUAL	Light level transition from Prolong to Standby on receiving Light Off message

Table 4. Controller events

Light LC event	Description
MODE_OFF	Mode off event is generated on receiving Light LC Mode Set message or on unsolicited change in Light Lightness Linear state
MODE_ON	Mode on event is generated on receiving Light LC Mode Set message
LIGHT_OFF	Light off event is generated on receiving Light LC OnOff Set message
LIGHT_ON	Light on event is generated on receiving Light LC OnOff Set message
OCCUPANCY_ON	Occupancy on event is generated on receiving occupancy detection from occupancy sensors
AMBIENT_LUX_MODIFIED	Ambient lux modified event is generated on receiving ambient lux level information from sensors
TIMER_ON	Timer on event is generated corresponding to transition timer start event from transition timer
TIMER_TRIGGER	Timer trigger event is generated corresponding to transition timer trigger event from transition timer
TIMER_OFF	Timer off event is generated corresponding to transition timer stop event from transition timer

3.5 Controller state transition on processing of events

Table 5 and Table 6 represent the light controller behavior.

The state transitions can be looked at in two ways:

1. to map from the current state to different states on processing different events;
2. to map from different states to the next state.

Table 5. States grouped according to current state

Current state	Event	Next state
OFF	MODE_ON	STANDBY
STANDBY	MODE_OFF	OFF
	AMBIENT_LUX_MODIFIED	STANDBY
	LIGHT_ON	POST_STANDBY
	OCCUPANCY_ON	POST_STANDBY
POST_STANDBY	MODE_OFF	OFF
	AMBIENT_LUX_MODIFIED	POST_STANDBY
	TIMER_ON	FADE_ON
	TIMER_OFF	RUN
	LIGHT_OFF	STANDBY
	LIGHT_ON	POST_STANDBY
PRE_FADE_ON	MODE_OFF	OFF
	AMBIENT_LUX_MODIFIED	PRE_FADE_ON
	TIMER_ON	FADE_ON
	TIMER_OFF	RUN
	LIGHT_OFF	PRE_FADE_STANDBY_MANUAL
	LIGHT_ON	PRE_FADE_ON
FADE_ON	MODE_OFF	OFF
	AMBIENT_LUX_MODIFIED	FADE_ON

Current state	Event	Next state
FADE_ON	TIMER_TRIGGER	FADE_ON
	TIMER_OFF	RUN
	LIGHT_OFF	PRE_FADE_STANDBY_MANUAL
	LIGHT_ON	PRE_FADE_ON
RUN	MODE_OFF	OFF
	AMBIENT_LUX_MODIFIED	RUN
	TIMER_OFF	FADE
	LIGHT_OFF	PRE_FADE_STANDBY_MANUAL
	LIGHT_ON	RUN
	OCCUPANCY_ON	RUN
FADE	MODE_OFF	OFF
	AMBIENT_LUX_MODIFIED	FADE
	TIMER_TRIGGER	FADE
	TIMER_OFF	PROLONG
	LIGHT_OFF	PRE_FADE_STANDBY_MANUAL
	LIGHT_ON	PRE_FADE_ON
	OCCUPANCY_ON	PRE_FADE_ON
PROLONG	MODE_OFF	OFF
	AMBIENT_LUX_MODIFIED	PROLONG
	TIMER_OFF	FADE_STANDBY_AUTO
	LIGHT_OFF	PRE_FADE_STANDBY_MANUAL
	LIGHT_ON	PRE_FADE_ON
	OCCUPANCY_ON	PRE_FADE_ON
FADE_STANDBY_AUTO	MODE_OFF	OFF
	AMBIENT_LUX_MODIFIED	FADE_STANDBY_AUTO
	TIMER_TRIGGER	FADE_STANDBY_AUTO
	TIMER_OFF	STANDBY
	LIGHT_OFF	PRE_FADE_STANDBY_MANUAL
	LIGHT_ON	PRE_FADE_ON
	OCCUPANCY_ON	PRE_FADE_ON
PRE_FADE_STANDBY_MANUAL	MODE_OFF	OFF
	AMBIENT_LUX_MODIFIED	PRE_FADE_STANDBY_MANUAL
	TIMER_ON	FADE_STANDBY_MANUAL
	TIMER_OFF	STANDBY
	LIGHT_ON	PRE_FADE_ON
FADE_STANDBY_MANUAL	MODE_OFF	OFF
	AMBIENT_LUX_MODIFIED	FADE_STANDBY_MANUAL
	TIMER_TRIGGER	FADE_STANDBY_MANUAL
	TIMER_OFF	STANDBY

Current state	Event	Next state
FADE_STANDBY_MANUAL	LIGHT_ON	PRE_FADE_ON

Table 6. States grouped according to the next state

Current state	Event	Next state	
STANDBY	MODE_OFF	OFF	
POST_STANDBY			
PRE_FADE_ON			
FADE_ON			
RUN			
FADE			
PROLONG			
FADE_STANDBY_AUTO			
PRE_FADE_STANDBY_MANUAL			
FADE_STANDBY_MANUAL			
OFF	MODE_ON	STANDBY	
STANDBY	AMBIENT_LUX_MODIFIED		
POST_STANDBY	LIGHT_OFF		
FADE_STANDBY_AUTO	TIMER_OFF		
PRE_FADE_STANDBY_MANUAL	TIMER_OFF		
FADE_STANDBY_MANUAL	TIMER_OFF		
STANDBY	LIGHT_ON	POST_STANDBY	
STANDBY	OCCUPANCY_ON		
POST_STANDBY	LIGHT_ON		
POST_STANDBY	AMBIENT_LUX_MODIFIED		
PRE_FADE_ON	LIGHT_ON	PRE_FADE_ON	
PRE_FADE_ON	AMBIENT_LUX_MODIFIED		
FADE_ON	LIGHT_ON		
FADE	LIGHT_ON		
FADE	OCCUPANCY_ON		
PROLONG	LIGHT_ON		
PROLONG	OCCUPANCY_ON		
FADE_STANDBY_AUTO	LIGHT_ON		
FADE_STANDBY_AUTO	OCCUPANCY_ON		
PRE_FADE_STANDBY_MANUAL	LIGHT_ON		
FADE_STANDBY_MANUAL	LIGHT_ON		
POST_STANDBY	TIMER_ON		FADE_ON
PRE_FADE_ON	TIMER_ON		
FADE_ON	TIMER_TRIGGER		
FADE_ON	AMBIENT_LUX_MODIFIED		

Current state	Event	Next state
POST_STANDBY	TIMER_OFF	RUN
PRE_FADE_ON	TIMER_OFF	
FADE_ON	TIMER_OFF	
RUN	LIGHT_ON	
RUN	OCCUPANCY_ON	
RUN	AMBIENT_LUX_MODIFIED	
RUN	TIMER_OFF	FADE
FADE	TIMER_TRIGGER	
FADE	AMBIENT_LUX_MODIFIED	
FADE	TIMER_OFF	PROLONG
PROLONG	AMBIENT_LUX_MODIFIED	
PROLONG	TIMER_OFF	FADE_STANDBY_AUTO
FADE_STANDBY_AUTO	TIMER_TRIGGER	
FADE_STANDBY_AUTO	AMBIENT_LUX_MODIFIED	
PRE_FADE_ON	LIGHT_OFF	PRE_FADE_STANDBY_MANUAL
FADE_ON	LIGHT_OFF	
RUN	LIGHT_OFF	
FADE	LIGHT_OFF	
PROLONG	LIGHT_OFF	
FADE_STANDBY_AUTO	LIGHT_OFF	
PRE_FADE_STANDBY_MANUAL	AMBIENT_LUX_MODIFIED	
PRE_FADE_STANDBY_MANUAL	TIMER_ON	FADE_STANDBY_MANUAL
FADE_STANDBY_MANUAL	TIMER_TRIGGER	
FADE_STANDBY_MANUAL	AMBIENT_LUX_MODIFIED	

In off state, the controller is disabled, and it does not process incoming messages on Element 2 (Figure 3). It indicates that Light LC Linear Out is not bound with Light Lightness Linear (Figure 2). In this case, the light can be controlled via client messages with any of the generic or lighting models supported on Element 1 (Figure 3). To enable the controller dedicated, Light LC Mode Set is required.

RELATED LINKS

For Light LC state machine – Part 1 and Light LC state machine – Part 2, refer to Mesh Model Specification v1.0.1




For Light LC mode Set, refer to Section 6.3.5.1.2 of Mesh Model Specification v1.0.1

3.6 Examples of light controller response in different scenarios

The light controller modifies the light output when receiving occupancy detection, light on, or light off messages. The following subsections show several examples to understand the response of the light controller in different states with respect to the different events it processes.

The values used in the examples are:

- Light Control Time Fade On = 1000 ms
- Light Control Time Run = 600 ms
- Light Control Time Fade = 300 ms
- Light Control Time Prolong = 300 ms

- Light Control Time Fade Standby Auto = 800 ms
 - Light Control Time Fade Standby Manual = 200 ms
 - Light Control Time Occupancy Delay ≥ 0
 - Light Control Lightness Standby = 1000
 - Light Control Lightness On = 11000
 - Light Control Lightness Prolong = 5000
 - Default step resolution = 100 ms
-
- Event: Occupancy detection represented by 
 - Event: Light On represented by , with optional parameters:
 - Number of steps: 15
 - Step resolution: 100 ms
 - Delay: 200 ms
 - Event: Light Off represented by , with optional parameters:
 - Number of steps: 4
 - Step resolution: 100 ms
 - Delay: 200 ms

RELATED LINKS

[Mesh Device Properties v2.0](#)

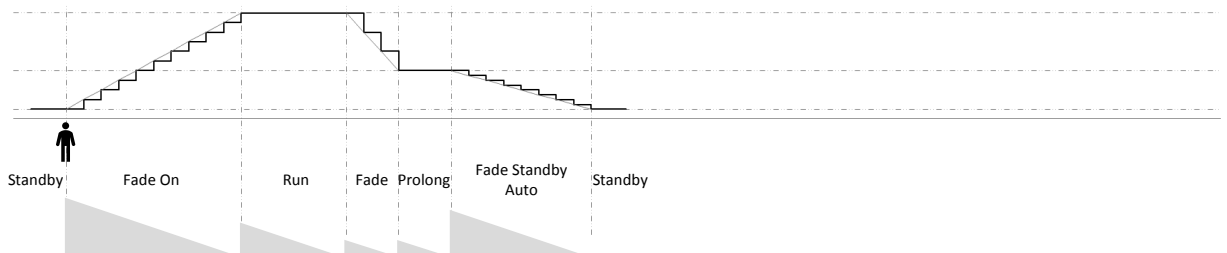
For the calculation of the total time during transition, refer to Section 1.4.1 of Mesh Model Specification v1.0.1

To compute transition parameters and intermediate state value, refer to Section 6.2.5.13.2 and Section 6.2.5.13.3 of Mesh Model Specification v.01.1

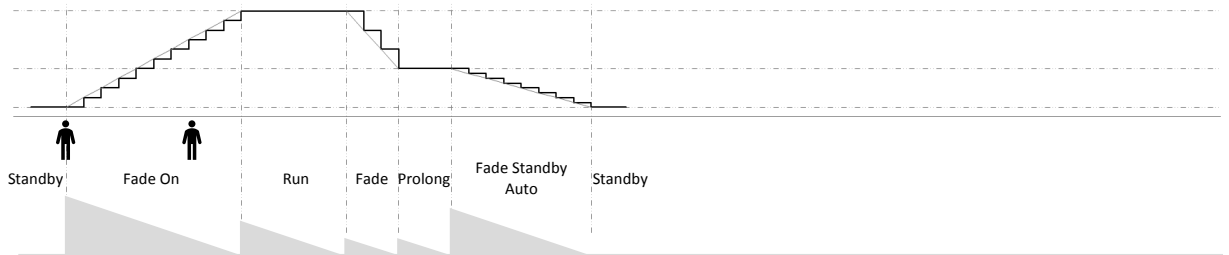
3.6.1 Example 1

Example 1 shows the controller response corresponding to occupancy detection by the controller during Standby state (Light Control Time Occupancy Delay = 0).

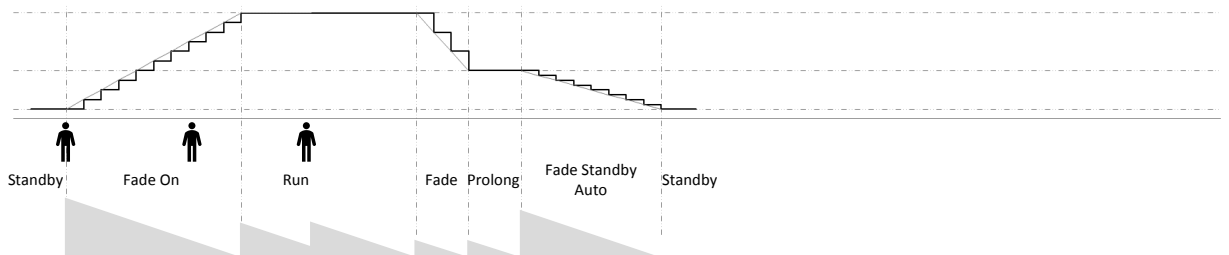
Figure 7. Controller behavior for Example 1


3.6.2 Example 2

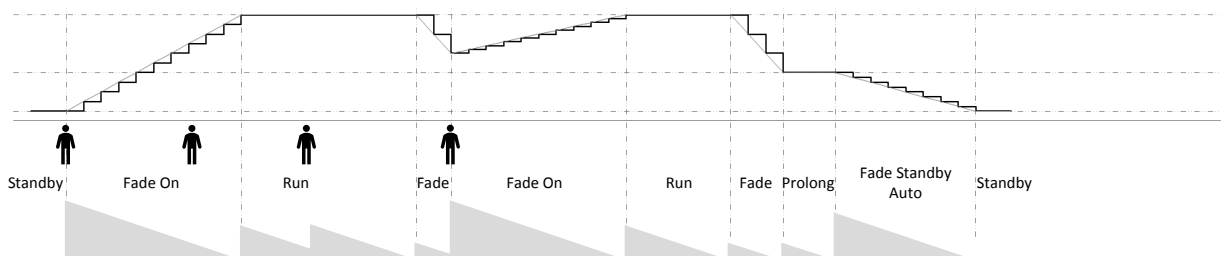
Example 2 shows the controller response corresponding to occupancy detection by the controller during Standby state and during Fade On state. The controller response is the same of Example 1 (Light Control Time Occupancy Delay = 0).

Figure 8. Controller behavior for Example 2

3.6.3 Example 3

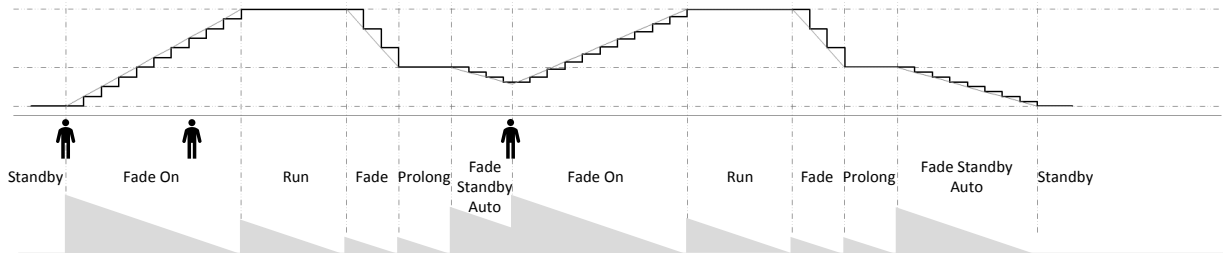
Occupancy is also detected during Run state. In this case, the timer is reset and started again for the duration given by Light Control Time Run (Light Control Time Occupancy Delay = 0).

Figure 9. Controller behavior for Example 3

3.6.4 Example 4

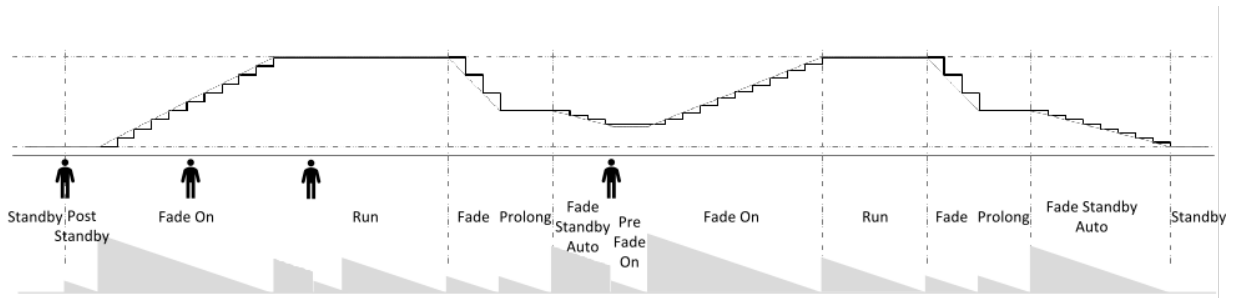
Occupancy is also detected in Fade state. The controller moves to Fade On state with initial light level equal to the level achieved in Fade state. This ensures a smooth transition of light output (Light Control Time Occupancy Delay = 0).

Figure 10. Controller behavior for Example 4

3.6.5 Example 5

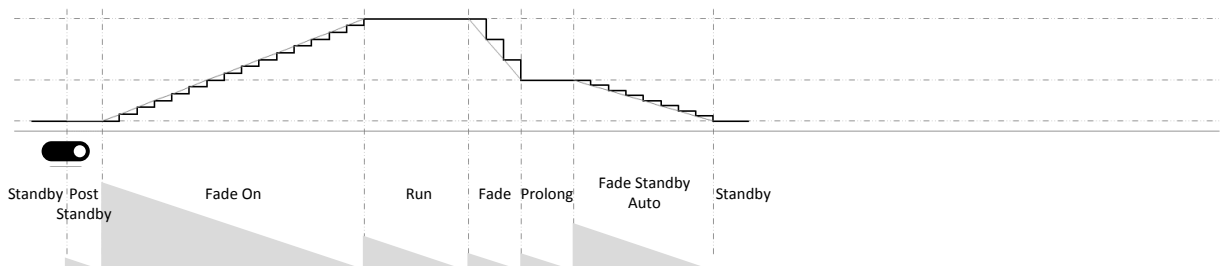
Example 5 shows the controller response in case occupancy is detected during Standby, Fade On and Fade Standby Auto State (Light Control Time Occupancy Delay = 0).

Figure 11. Controller behavior for Example 5

3.6.6 Example 6

Example 6 shows the controller response on detecting occupancy during different states where Light Control Time Occupancy Delay = 200ms. Due to this non-zero delay, additional time (that is, delay) is added during Run and Pre Fade On states. The light output is constant during this delay period.

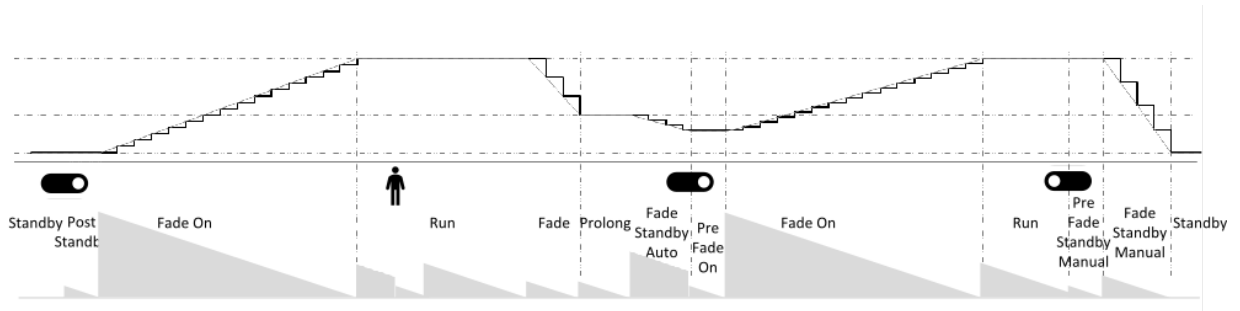
Figure 12. Controller behavior for Example 6

3.6.7 Example 7

Example 7 shows the controller response on receiving Light LC on message (Light Control Time Occupancy Delay = 200 ms).

Figure 13. Controller behavior for Example 7

3.6.8 Example 8

Example 8 shows the controller response on receiving Light LC On message during Standby, Fade Standby Auto, and Light LC Off message during Run state (Light Control Time Occupancy Delay = 200 ms).

Figure 14. Controller behaviour for Example 8



3.7 Occupancy detection

The light controller supports the processing of sensor status messages to enable automated lighting control, using occupancy detection (Figure 6).

Sensor nodes (supporting Sensor Server model) publish occupancy status, which can be processed by the controller.

Light LC Server model supports Sensor Status messages of sensor properties that report occupancy detection as follows:

- motion-sensed property value > 0
- people count property value > 0
- presence-detected property > 0

3.8 Adaptive control for daylight harvesting

For adaptive lighting and daylight harvesting, the Light LC Server model processes the sensor status message on the basis of the Present Ambient Light Level property of the ambient light sensor nodes.

This input is given to PI regulator to adjust the output light level accordingly (Section 3.2).

You can also set accuracy and coefficients of the PI regulator to modify the behavior of adaptive light control.

3.9 Distributed controllers and decentralized architecture

The light lightness controller of each light node does not depend on the other controllers.

After the initial configuration, the controller processes messages from the sensor nodes, occupancy sensors, and ambient light sensors.

There is no central node that controls the lighting nodes. Hence, the reliability of the lighting system depends on the sensor nodes and the light nodes.

In case of an ambient light sensor malfunction, the lights are still controlled by the occupancy sensors without considering the ambient light level and bypassing the PI regulator.

Note: Keep redundancy in the occupancy sensor nodes for long-term reliability of the system.

3.10 Light LC APIs and application callbacks

3.10.1 Light LC Server model APIs

In light_lc.h file, the Light LC module exposes a set of APIs used by application.

3.10.1.1 Light_LC_SleepDurationMs_Get

Table 7. Light_LC_SleepDurationMs_Get

Type	Description
Prototype	MOBLEUINT32 Light_LC_SleepDurationMs_Get(void)

Type	Description
Behavior description	It returns the time after which it is necessary to call <code>Light_LC_Process</code> . During this time, Light LC module can remain inactive.
Input parameter	None.
Output parameter	Sleep time in milliseconds.

3.10.1.2 *Light_LC_Process*

Table 8. Light_LC_Process

Type	Description
Prototype	<code>void Light_LC_Process(void)</code>
Behavior description	Scheduler to control light output periodically and to publish appropriately the Light LC status.
Input parameter	None.
Output parameter	None.

3.10.1.3 *Light_LCs_Init*

Table 9. Light_LCs_Init

Type	Description
Prototype	<code>MOBLE_RESULT Light_LCs_Init (void* lcsBuff,MOBLEUINT8 lcsElementIndex, const light_lc_cb_t* lcs_cb, MOBLEUINT16 sizeBuff)</code>
Behavior description	Light LC Server initialization
Input parameter	lcsBuff: buffer to be allocated to the Light LC Server structure lcsElementIndex: element index on which the Light LC Server is supported lcs_cb: reference to application callbacks used by the lighting controller sizeBuff: buffer size
Output parameter	Success if the initialization is successful, else fail.

3.10.2 Application callbacks

In `light_lc.h`, the Light LC module exposes a set of callbacks to be implemented by the application.

3.10.2.1 *LightLCs_ModeGet_cb*

Table 10. LightLCs_ModeGet_cb

Type	Description
Prototype	<code>void (*LightLCs_ModeGet_cb) (MODEL_MessageHeader_t *pmsgParam)</code>
Behavior description	Callback corresponding to the valid Get message of the Light LC mode received from the client. It is used to inform the controller application that the lighting controller module is processing of Light LC Get message.
Input parameter	Message parameters
Output parameter	None

3.10.2.2 LightLCs_ModeSet_cb
Table 11. LightLCs_ModeSet_cb

Type	Description
Prototype	void (*LightLCs_ModeSet_cb)(MOBLEUINT8 const* pRxData, MODEL_MessageHeader_t *pmsgParam)
Behavior Description	Callback corresponding to the valid Set message of the Light LC mode received from the client. It is used to inform the controller application that the lighting controller module is processing the Light LC Set message.
Input Parameter	pRxData: received data pmsgParam: message parameters
Output Parameter	None

3.10.2.3 LightLCs_ModeSetUnack_cb
Table 12. LightLCs_ModeSetUnack_cb

Type	Description
Prototype	void (*LightLCs_ModeSetUnack_cb)(MOBLEUINT8 const* pRxData, MODEL_MessageHeader_t *pmsgParam)
Behavior description	Callback corresponding to the valid Set Unacknowledged message of the Light LC mode received from the client. It is used to inform the controller application that the lighting controller module is processing the Light LC Set Unacknowledged message.
Input parameter	pRxData: received data pmsgParam: message parameters
Output parameter	None

3.10.2.4 LightLCs_OmGet_cb
Table 13. LightLCs_OmGet_cb

Type	Description
Prototype	void (*LightLCs_OmGet_cb)(MODEL_MessageHeader_t *pmsgParam)
Behavior description	Callback corresponding to the valid Get message of the Light LC Occupancy mode received from the client. It is used to inform the controller application that the lighting controller module is processing the Light LC Occupancy Get message.
Input parameter	pmsgParam: message parameters
Output parameter	None

3.10.2.5 LightLCs_OmSet_cb
Table 14. LightLCs_OmSet_cb

Type	Description
Prototype	void (*LightLCs_OmSet_cb)(MOBLEUINT8 const* pRxData, MODEL_MessageHeader_t *pmsgParam)
Behavior description	Callback corresponding to the valid Set message of the Light LC Occupancy mode received from the client. It is used to inform the controller application that the lighting controller module is processing the Light LC Occupancy Set message.
Input parameter	pRxData: received data

Type	Description
	pmsgParam: message parameters
Output parameter	None

3.10.2.6 *LightLCs_OmSetUnack_cb*

Table 15. LightLCs_OmSetUnack_cb

Type	Description
Prototype	void (*LightLCs_OmSetUnack_cb) (MOBLEUINT8 const* pRxData, MODEL_MessageHeader_t *pmsgParam)
Behavior description	Callback corresponding to the valid Set message of the Light LC Occupancy mode received from the client. It is used to inform the controller application that the lighting controller module is processing the Light LC Occupancy Set message.
Input parameter	pRxData: received data pmsgParam: message parameters
Output parameter	None

3.10.2.7 *LightLCs_OnOffGet_cb*

Table 16. LightLCs_OnOffGet_cb

Type	Description
Prototype	void (*LightLCs_OnOffGet_cb) (MODEL_MessageHeader_t *pmsgParam)
Behavior description	Callback corresponding to the valid Get message of the Light LC OnOff mode received from the client. It is used to inform the controller application that the lighting controller module is processing the Light LC OnOff Get message.
Input parameter	pmsgParam: message parameters
Output parameter	None

3.10.2.8 *LightLC_OnOffSet_cb*

Table 17. LightLC_OnOffSet_cb

Type	Description
Prototype	void (*LightLC_OnOffSet_cb) (MOBLEUINT8 const* pRxData, MODEL_MessageHeader_t *pmsgParam)
Behavior description	Callback corresponding to the valid Set message of the Light LC OnOff mode received from the client. It is used to inform the controller application that the lighting controller module is processing the Light LC OnOff Set message.
Input parameter	pRxData: received data pmsgParam: message parameters
Output parameter	None

3.10.2.9 *LightLCs_OnOffSetUnack_cb*

Table 18. LightLCs_OnOffSetUnack_cb

Type	Description
Prototype	void (*LightLCs_OnOffSetUnack_cb) (MOBLEUINT8 const* pRxData, MODEL_MessageHeader_t *pmsgParam)

Type	Description
Behavior description	Callback corresponding to the valid Set Unacknowledged message of the Light LC OnOff mode received from the client. It is used to inform the controller application that the lighting controller module is processing the Light LC OnOff Set Unacknowledged message.
Input parameter	pRxData: received data pmsgParam: message parameters
Output parameter	None

3.10.2.10 *LightLCs_PropertyGet_cb*

Table 19. LightLCs_PropertyGet_cb

Type	Description
Prototype	void (*LightLCs_PropertyGet_cb) (MOBLEUINT8 const* pRxData, MODEL_MessageHeader_t *pmsgParam)
Behavior description	Callback corresponding to the valid Get message of the Light LC Property mode received from the client. It is used to inform the controller application that the lighting controller module is processing the Light LC Property Get message.
Input parameter	pRxData: received data pmsgParam: message parameters
Output parameter	None

3.10.2.11 *LightLCs_PropertySet_cb*

Table 20. LightLCs_PropertySet_cb

Type	Description
Prototype	void (*LightLCs_PropertySet_cb) (MOBLEUINT8 const* pRxData, MODEL_MessageHeader_t *pmsgParam)
Behavior description	Callback corresponding to the valid Set message of the Light LC Property mode received from the client. It is used to inform the controller application that the lighting controller module is processing the Light LC Property Set message.
Input parameter	pRxData: received data pmsgParam: message parameters
Output parameter	None

3.10.2.12 *LightLCs_PropertySetUnack_cb*

Table 21. LightLCs_PropertySetUnack_cb

Type	Description
Prototype	void (*LightLCs_PropertySetUnack_cb) (MOBLEUINT8 const* pRxData, MODEL_MessageHeader_t *pmsgParam)
Behavior description	Callback corresponding to the valid Set Unacknowledged message of the Light LC Property mode received from the client. It is used to inform the controller application that the lighting controller module is processing the Light LC Property Set Unacknowledged message.
Input parameter	pRxData: received data pmsgParam: message parameters
Output parameter	None

4 References

- [Mesh Model Specification v1.0.1](#)
- [Mesh Device Properties v2.0](#)
- [Building a Sensor-Driven Lighting Control System Based on Bluetooth Mesh v1.0](#)
- [STBLEMesh - BLE Mesh application for Android and iOS](#)
- [STSW-BNRG-Mesh - Mesh over Bluetooth Low Energy](#)
- [STM32CubeWB - STM32Cube MCU Package for STM32WB series](#)
- [X-CUBE-BLEMESH1 - Mesh over Bluetooth low energy software expansion for STM32Cube](#)

Revision history

Table 22. Document revision history

Date	Revision	Changes
27-Oct-2021	1	Initial release.

Contents

1	Features	2
2	Structure and dependencies	3
2.1	Model state relation	3
2.2	Server model relation	3
2.3	Messages supported by a Light LC node	4
3	Light lightness controller	6
3.1	Transition timer	7
3.2	Proportional integral feedback regulator	8
3.3	Light lightness controller blocks	8
3.4	Controller states and events	9
3.5	Controller state transition on processing of events	10
3.6	Examples of light controller response in different scenarios	13
3.6.1	Example 1	14
3.6.2	Example 2	14
3.6.3	Example 3	15
3.6.4	Example 4	15
3.6.5	Example 5	15
3.6.6	Example 6	16
3.6.7	Example 7	16
3.6.8	Example 8	16
3.7	Occupancy detection	17
3.8	Adaptive control for daylight harvesting	17
3.9	Distributed controllers and decentralized architecture	17
3.10	Light LC APIs and application callbacks	17
3.10.1	Light LC Server model APIs	17
3.10.2	Application callbacks	18
4	References	22
	Revision history	23
	List of tables	25
	List of figures	26

List of tables

Table 1.	Transition states	7
Table 2.	Transition events	7
Table 3.	Controller states	9
Table 4.	Controller events	10
Table 5.	States grouped according to current state	10
Table 6.	States grouped according to the next state	12
Table 7.	Light_LC_SleepDurationMs_Get	17
Table 8.	Light_LC_Process	18
Table 9.	Light_LCs_Init	18
Table 10.	LightLCs_ModeGet_cb	18
Table 11.	LightLCs_ModeSet_cb	19
Table 12.	LightLCs_ModeSetUnack_cb	19
Table 13.	LightLCs_OmGet_cb	19
Table 14.	LightLCs_OmSet_cb	19
Table 15.	LightLCs_OmSetUnack_cb	20
Table 16.	LightLCs_OnOffGet_cb	20
Table 17.	LightLC_OnOffSet_cb	20
Table 18.	LightLCs_OnOffSetUnack_cb	20
Table 19.	LightLCs_PropertyGet_cb	21
Table 20.	LightLCs_PropertySet_cb	21
Table 21.	LightLCs_PropertySetUnack_cb	21
Table 22.	Document revision history	23
Table 1.	Transition states	0
Table 2.	Transition events	0
Table 5.	States grouped according to current state	0
Table 6.	States grouped according to the next state	0

List of figures

Figure 1.	Floor with light controllers for occupancy detection	1
Figure 2.	Relation among the states of lighting controller in a Bluetooth mesh node	3
Figure 3.	Relation among the models of lighting controller in a Bluetooth mesh node.	4
Figure 4.	Light lightness controller behaviour	6
Figure 5.	Transition state machine behavior.	7
Figure 6.	Light lightness controller inputs and outputs	9
Figure 7.	Controller behavior for Example 1	14
Figure 8.	Controller behavior for Example 2	15
Figure 9.	Controller behavior for Example 3	15
Figure 10.	Controller behavior for Example 4	15
Figure 11.	Controller behavior for Example 5	16
Figure 12.	Controller behavior for Example 6	16
Figure 13.	Controller behavior for Example 7	16
Figure 14.	Controller behaviour for Example 8	17

IMPORTANT NOTICE – PLEASE READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, please refer to www.st.com/trademarks. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2021 STMicroelectronics – All rights reserved