

# **AN1438: Networked Lighting Control**

Networked lighting control (NLC) systems feature an intelligent network of individually addressable and sensor-rich luminaires and control devices that allows each component of the system to send and receive data. Specifically designed to meet the scale, reliability, and security demands required in commercial settings, Bluetooth® NLC is the only full-stack standard for wireless lighting control. By offering standardization from the radio through the device layer, Bluetooth® NLC enables true multi-vendor interoperability and mass adoption of wireless lighting control.

The Basic Lightness Controller NLC Profile specifies the requirements for a networked lighting control (NLC) product acting as a luminaire controller in a Bluetooth mesh system. The Basic Lightness Controller NLC Profile standardizes the use cases and implementation patterns of luminaire controllers to help improve interoperability and performance of systems based on Bluetooth mesh, such as NLC systems.

A common use case for the Basic Lightness Controller NLC Profile is a luminaire reacting to information published by occupancy and/or ambient light sensors as well as reacting to override events (e.g., manually dimming/brightening the lights or turning them on/off) in NLC systems. AVAILABLE PROFILES

- Basic Lightness Controller NLC Profile
- Occupancy Sensor NLC Profile
- Ambient Light Sensor NLC Profile
- Dimming Control NLC Profile
- Basic Scene Selector NLC Profile



## 1 Introduction

Part of the reason that Bluetooth technology has become such a prevalent technology in people's lives is its reliance on standards. Through standards, the Bluetooth connections made between phones, PCs, headphones, game controllers, vehicle entertainment systems, and countless others all work because they all have built and established trust in the device profiles that initiate these connections, and they are universally used and available.

The Bluetooth SIG is taking that to the next step with the release of their new Network Lighting Control (NLC) bundle of device profiles. These standardized profiles will improve interoperability and scalability, simplify integration in the field, and grow the Bluetooth ecosystem. The same Silicon Labs Bluetooth SoCs and modules that support the new feature enhancements can also support the following NLC profiles: ambient light sensor, basic scene selector, dimming control, basic lightness controller, and occupancy sensor.

## 1.1 Lighting Control Device Profiles

Device profiles define which options and features of the Bluetooth Mesh specifications are mandatory for a certain kind of end product. The first suite of mesh device profiles, collectively referred to as NLC profiles, build on Bluetooth Mesh to enable the world's first fullstack, multi-vendor interoperable wireless standard for wireless lighting control, Bluetooth® NLC.

## 1.2 NLC Profile Relationships

A device implementing the Basic Lightness Controller NLC Profile interacts with devices implementing the following NLC profiles: Occupancy Sensor NLC Profile, Ambient Light Sensor NLC Profile, Dimming Control NLC Profile, Basic Scene Selector NLC Profile (see Figure 1).



Figure 1. Interaction of a Basic Lightness Controller with other NLC Profiles

## 2 Implementation

Silicon Labs provides five examples to demonstrate this feature, one for every NLC Profile (except the Energy Monitor NLC Profile). All the examples and their main functionality logics are detailed below.

### 2.1 Bluetooth Mesh - NLC Basic Lightness Controller

*btmesh\_soc\_nlc\_basic\_lightness\_controller* - This is an example of a Bluetooth Mesh application providing the functionality of the Basic Lightness Controller NLC Profile. It demonstrates how to control a light source, an LED mounted on a mainboard, and a radio board or similar hardware, connected to a Bluetooth Mesh network. The light source lightness can be controlled with a Generic Level Client, e.g. another radio board running the Bluetooth Mesh - NLC Dimming Control application, or with Bluetooth Mesh smartphone application.

The device listens to messages from other NLC devices, namely Occupancy Sensor, Ambient Light Sensor, Dimming Control, and Basic Scene Selector nodes. Only a very simple implementation is needed on the application level.

```
* Callback for setting Light Lightness by PWM level (0x0001 - FFFE)
                           **********************************
*****
void sl btmesh lighting level pwm cb(uint16 t level)
{
 app led set level(level);
}
* Callback for setting Light Color by PWM level (0x0001 - FFFE)
*****
                  void sl btmesh lighting color pwm cb(uint16 t color)
{
 app led set color(color);
}
```

#### 2.2 Bluetooth Mesh - NLC Basic Scene Selector

*btmesh\_soc\_nlc\_basic\_scene\_selector* - This is an example of a Low Power Node-enabled Bluetooth Mesh application providing the functionality of the Basic Scene Selector NLC Profile. Once the node is provisioned and a Basic Lightness Controller subscribes to the client, the two buttons of the mainboard are used to publish the messages that will recall stored scenes on the server. The example provides this recall functionality for scenes #1 and #2. The scene store functionality is provided by the mobile application.

Push Button presses control Basic Lightness Controllers in the network by scene recall requests. This requires the scene selection logic to be implemented in the button handler function.

```
* Button press Callbacks
*****
void app_button_press_cb(uint8_t button, uint8 t duration)
{
 (void) duration;
 char scene[SCENE BUF LEN];
#if SL SIMPLE BUTTON COUNT == 1
 if (button == BUTTON PRESS BUTTON 0) {
   // Switch between scene 1 and 2
   current scene = current scene == 1 ? 2 : 1;
   sl btmesh select scene(current scene);
 }
#endif
#if SL SIMPLE BUTTON COUNT >= 2
 // Select scene by pushed button
 current scene = button == BUTTON PRESS BUTTON 0 ? 1 : 2;
 sl btmesh select scene(current scene);
#endif
 // Create unique device name using the last two bytes of the device UUID
 snprintf(scene, SCENE BUF LEN, "Selected scene: %d", current scene);
 lcd print(scene, SL BTMESH WSTK LCD ROW CURRENT SCENE CFG VAL);
}
```

#### 2.3 Bluetooth Mesh - NLC Dimming Control

*btmesh\_soc\_nlc\_dimming\_control* - This is an example of a Low Power Node-enabled Bluetooth Mesh application providing the functionality of the Dimming Control NLC Profile. Once the node is provisioned and a Basic Lightness Controller subscribes to the client, the two buttons of the mainboard are used to publish the messages that will change the state of the Basic Lightness Controller node.

Buttons can control the following models:

- Generic On/Off Client model can turn the light on and off or toggle
- Generic Level Client model can control the light brightness

Push Button presses control Basic Lightness Controllers in the network by Generic Level Delta or Generic On/Off messages. A slightly more complex logic is needed for the dimming in the button handler function.

```
* Button press Callbacks
 void app_button_press_cb(uint8_t button, uint8_t duration)
#if SL SIMPLE BUTTON COUNT == 1
 if (button == BUTTON PRESS BUTTON 0) {
   switch (duration) {
     // Handling of button press less than 0.25s
     case APP BUTTON PRESS DURATION SHORT: {
       sl btmesh generic level client ext delta set unack(delta);
     } break;
     // Handling of button press greater than 0.25s and less than 1s
     case APP BUTTON PRESS DURATION MEDIUM: {
       sl btmesh generic level client ext delta set unack(-delta);
     } break;
     // Handling of button press greater than 1s
     case APP BUTTON PRESS DURATION LONG:
     case APP BUTTON PRESS DURATION VERYLONG: {
       sl btmesh change switch position (SL BTMESH LIGHTING CLIENT TOGGLE);
     } break;
     default:
       break;
   }
 }
#endif
#if SL SIMPLE BUTTON COUNT >= 2
 // Selecting action by duration
 switch (duration) {
   // Handling of button press less than 0.25s
   case APP BUTTON PRESS DURATION SHORT: {
     int32 t delta set;
     delta set = button == BUTTON PRESS BUTTON 0 ? -delta : delta;
     sl btmesh generic level client ext delta set unack(delta set);
     break;
   }
   // Anything more than 0.25s
   case APP BUTTON PRESS DURATION MEDIUM:
   case APP BUTTON PRESS DURATION LONG:
   case APP BUTTON PRESS DURATION VERYLONG: {
     uint8 t on off = button == BUTTON PRESS BUTTON 0 ? SL BTMESH LIGHTING CLIENT OFF
SL BTMESH LIGHTING CLIENT ON;
     sl btmesh change switch position (on off);
     break;
   }
   default:
     break;
 }
#endif
}
```

#### 2.4 Bluetooth Mesh - NLC Ambient Light Sensor

*btmesh\_soc\_nlc\_sensor\_ambient\_light* - This example makes ambient light sensor measurements and forwards them to another node implementing the Basic Lightness Controller NLC Profile.

The device measures ambient light and sends these measurements to the network. Properly configured NLC Basic Lightness Controllers then can act on the received data. This example contains a Mock sensor logic in the button handler, in case there is no actual light sensor present on the board, thus revealing a possible logic to implement.

```
* Callback for button press
                            void app_button_press_cb(uint8_t button, uint8_t duration)
{
 #if defined (SL CATALOG SENSOR LIGHT LUX MOCK PRESENT)
 float uvi;
 float lux;
 sl status t sc;
 sc = sl sensor light get(&lux, &uvi);
 app_assert_status_f(sc, "Failed to get lux and uvi");
 #if (SL_SIMPLE_BUTTON_COUNT >= 2)
 // button pressed
 if (duration == APP BUTTON PRESS DURATION SHORT) {
   if (button == BUTTON PRESS BUTTON 0) {
     app log("PB0 pressed" APP LOG NL);
     sl_sensor_light_set(lux - LUX_SMALL_CHANGE, uvi);
   } else if (button == BUTTON_PRESS_BUTTON_1) {
     app_log("PB1 pressed" APP_LOG_NL);
     sl sensor light set(lux + LUX SMALL CHANGE, uvi);
  } else if (duration == APP BUTTON PRESS DURATION MEDIUM) {
   if (button == BUTTON PRESS BUTTON 0) {
     app log("PB0 medium pressed" APP LOG NL);
     sl_sensor_light_set(lux - LUX_LARGE_CHANGE, uvi);
   } else if (button == BUTTON_PRESS BUTTON 1) {
     app log("PB1 medium pressed" APP LOG NL);
     sl_sensor_light_set(lux + LUX LARGE CHANGE, uvi);
   }
 #elif (SL SIMPLE BUTTON COUNT == 1)
  (void) duration;
 if (button == BUTTON PRESS BUTTON 0)
   app log("PB0 pressed" APP LOG NL);
   sl sensor light set(lux + LUX LARGE CHANGE, uvi);
 #endif // SL SIMPLE BUTTON COUNT
 #else
  (void) duration;
  (void) button;
 #endif // SL CATALOG SENSOR LIGHT MOCK PRESENT
}
```

## 2.5 Bluetooth Mesh - NLC Occupancy Sensor

*btmesh\_soc\_nlc\_sensor\_occupancy* – This example makes Occupancy Sensor measurements and forwards them to another node implementing the Basic Lightness Controller NLC Profile. This measurement can be faked by pressing the buttons on the mainboard:

- BTN1 increases people count
- BTN0 decreases people count

In case of only one button, a shorter press increases, while a longer one decreases the people count.

Push Button presses imitate people count changes which can control a properly configured NLC Basic Lightness Controller. Here as well, buttons simulate the increase/decrease of people, implemented in the button press handler function.

```
* Callback for button press
                       void app_button_press_cb(uint8_t button, uint8_t duration)
{
 (void) duration;
 // button pressed
 if (duration == APP BUTTON PRESS DURATION SHORT) {
   if (button == BUTTON PRESS BUTTON 0) {
    app log("PB0 pressed" APP LOG NL);
    sl_btmesh_people_count_decrease();
   } else if (button == BUTTON PRESS BUTTON 1) {
    app log("PB1 pressed" APP LOG NL);
    sl btmesh people count increase();
   }
 } else if (duration == APP BUTTON PRESS DURATION MEDIUM) {
   if ( (button == BUTTON PRESS BUTTON 0)) {
    app log("PB0 medium pressed" APP LOG NL);
    sl btmesh people count increase();
   }
 }
}
```

## 3 Network Setup

To create a demonstration network, at least five WSTK + radio boards or Thunderboards are needed, preferably ones equipped with display, and at least one of them with light sensor for the *btmesh\_soc\_nlc\_sensor\_ambient\_light* example and another one with RGB LED for the *btmesh\_soc\_nlc\_basic\_lightness\_controller* example.

## 3.1 Flashing

You may use the provided demo binaries or create your own project and flash it to the board with the bootloader of your choice. Doing these should lead to the terminal outputs below:

## 3.1.1 Bluetooth Mesh - NLC Basic Lightness Controller

.BT Mesh NLC Basic Lightness Controller initialized > Device name: 'NLC Light 9356' BT mesh node is unprovisioned, started unprovisioned beaconing...

## 3.1.2 Bluetooth Mesh - NLC Basic Scene Selector

BT Mesh NLC Basic scene selector initialized

> Device name: 'NLC Scene node 88f8'

BT mesh node is unprovisioned, started unprovisioned beaconing...

3.1.3 Bluetooth Mesh - NLC Dimming Control

.BT Mesh NLC Dimming Control initialized

> Device name: 'NLC Dimmer 3006'

BT mesh node is unprovisioned, started unprovisioned beaconing...

## 3.1.4 Bluetooth Mesh - NLC Ambient Light Sensor

.BT mesh NLC Ambient Light Sensor initialized > Device name: 'NLC Amb. Light 52bb' BT mesh node is unprovisioned, started unprovisioned beaconing...

3.1.5 Bluetooth Mesh - NLC Occupancy Sensor

BT mesh NLC Occupancy Sensor initialized

> Device name: 'NLC Occupancy bd74'

BT mesh node is unprovisioned, started unprovisioned beaconing...

## 3.2 Bluetooth Mesh App

With the help of Silicon Labs' own Bluetooth Mesh App, scan for the Nodes and start to provision them one by one into your own NLC Mesh network.

2:30 🎾 🗖	* 0 *	▼ ₩ 🖌 🕯	2:33 💅 🔳	* 🖂 🕈 🕍 🕯		2:35 🎾 🔳	* ∈	V Vo	48
	SILICON LABS	Scan :	÷	Sto	op ←	Subnet 0			
Subnets Provisio	on NLC	IOP Test	B4:E3:F9:C5:92	::E8			Devices		
			(( <b>•</b> )) -76 dBm UUID: 2eaa75fb-8c62-8	► 3c5a-9e43-0b26f8499356	B4:E3	8:F9:C5:92:E8			
			68:0A:E2:28:8E	:E7	LC Sta	ates	,	'n	
			-47 dBm UUID: ec4a296e-da1a-	\$ 4c5e-8324-80f368f13006	LC Oc	ccupancy Mode	(	3	•
			84:71:27:6E:EF	:E7	LC Or	n Off	(	3	
			-37 dBm UUID: 09463848-0633	-c556-be59-b6692cb0bd74	LC Pro	operty States nbientLuxLevelC	Dn		
<b>No subnets</b> Tap Scan to start searching for Bluetooth mesh devices.			68:0A:E2:28:7D	):20	Lu	x G			
			-53 dBm	≥753-a7b6-ae077e3b52bb		Pla	ace value here		
			14:B4:57:D5:	l:ED			Send		
			((中)) -73 dBm	>	Store	current node's st	ate to scene		
					Scene	:1		Sto	ore
					Scene	2		Sto	ore
					<b>84:71</b> 200a	.:27:6E:EF:E7			?
						50 00 70 00			

Figure 2. Before and After the Provisioning Process

\* ⊝ ♥ 🖫 ⊿ 🕯

C

C

C

С

\* ⊖ ♥ 🖫 ⊿ 🕯

С

C

C

2:40 🤣 🖸

3:09 🤣 🗉

14:B4:57:D5:04:ED

B4:E3:F9:C5:92:E8

Global Low Power Timeout

Low Power

Proxy

Relay

Retransmission

Firmware Update

14:B4:57:D5:04:ED

B4:E3:E9:C5:92:E8 Connected BT mesh node provisioning is started (result: 0x0000) BT mesh node is provisioned (address: 0x2008, iv\_index: 0x0) B4:E3:F9:C5:92:E8 [E] Status: sc = 0x041c (?) Lighting server lightbulb state load from PS failed or nvm is empty, use defaults. Proxv [I] On power up state is OFF BT mesh Lightness: 0% [D] Status: sc = 0x0514 (?) Lighting server state publish failed Friend (mdl=0x1006,elem=0,state=0x0001) Retransmission [D] Status: sc = 0x0514 (?) Lighting server state publish failed Firmware Update (mdl=0x1000,elem=0,state=0x0000) [D] Status: sc = 0x0514 (?) Lighting server state publish failed (mdl=0x1300,elem=0,state=0x0080) [E] Status: sc = 0x041c (?) LC server lc\_state load from PS failed or nvm is empty, use defaults. [E] Status: sc = 0x041c (?) LC server lc\_property\_state load from PS failed or nvm is empty, use defaults. [I] Friend mode initialization [I] Dummy LC ON/OFF change - same state as before [I] Lightness update -same value (0) BT mesh Friendship established with LPN (netkey idx: 0, lpn addr: 0x2007)





Connected BT mesh node provisioning is started (result: 0x0000) BT mesh node is provisioned (address: 0x2007, iv\_index: 0x0) [I] Trying to initialize lpn... Disconnected [I] LPN initialized BT mesh LPN on [I] Trying to find a friend...



silabs.com | Building a more connected world.

## 3.2.3 Bluetooth Mesh - NLC Dimming Control

Connected BT mesh node provisioning is started (result: 0x0000) BT mesh node is provisioned (address: 0x200a, iv\_index: 0x0) [I] Trying to initialize lpn... Disconnected [I] LPN initialized BT mesh LPN on [I] Trying to find a friend...

## Figure 5. Provisioning and Configuration of the NLC Dimming Control Node

3.2.4 Bluetooth Mesh - NLC Ambient Light Sensor

Connected BT mesh node provisioning is started (result: 0x0000) BT mesh node is provisioned (address: 0x2006, iv\_index: 0x0) Illuminance: 0.00 lx Disconnected

Figure 6. Provisioning and Configuration of the NLC Ambient Light Sensor Node

 2-40 % ■
 \* ○ ♥ ■ ▲ ▲

 Configuration
 Info

 Name
 68:0A:E2:28:7D:20

 Proxy
 C

 Relay
 C

 Firmware Update
 >



#### 3.2.5 Bluetooth Mesh - NLC Occupancy Sensor



### Figure 7. Provisioning and Configuration of the NLC Occupancy Sensor Node

#### 3.3 Altering the Network

After the full setup, the network parameters are adjustable by changing the LC Property States, and the network can be tested by triggering changes with the push buttons. Switching LC Mode OFF, the NLC Dimming Control Node can be used, and by switching it ON, the NLC Ambient Light Sensor going to be active. By switching Occupancy Mode ON, the NLC Occupancy Sensor Node will take over.

3:54 🕫 🌞 🔳	* ⊖ ♥ 🔣 🔺 🗎	People count: 1
		PB0 medium pressed
← Subnet 0		People count: 2
Devices		PB0 medium pressed
		People count: 3
B4:E3:F9:C5:92:E8		PB1 medium pressed
2008		Illuminance: 50.00 lx
LC States		PB1 medium pressed
LC Mode	с 🔴	Illuminance: 100.00 lx
LC Occupancy Mode	C 🛑	PB1 medium pressed
LC On Off	C O	Illuminance: 150.00 lx
		BT mesh Lightness: 95%
RegulatorKid		[T] lightness change kind 129 value 60323
RegulatorKiu		[T] Primary level undate: from 30105 to 30107
rieganator rita		[T] Dummy ON/OFE change - same state as before
RegulatorKpd		[1] Lightness undate: from 62873 to 62875
RegulatorKpu		[I] Lightness update. From 02075 to 02075
TimeFade		[1] LC ON/OFE state shared 0 to 1
		LI LC UN/UFF State changed 0 to 1
TimeFadeOn		BI mesh Lightness: 100%
TimeFadeStandbyAuto		[1] Lightness_change, Kina 129, Value 65535
TimeFadeStandbyManual		[1] Primary level update: from 30107 to 32767
The Original Dates		[1] Dummy ON/OFF change - same state as before
TimeOccupaticyDelay		[I] Lightness update: from 62875 to 65535
TimeProlong		[I] lightness_change, kind 130, value 65535
TimeRunOn		[I] LC ON/OFF state changed 1 to 0
	_	[I] LC ON/OFF state changed 0 to 1

Figure 8. Configuration of the Network and Node Logs

## 4 Custom Project

To create a custom NLC Profile project, you have to add the appropriate NLC component to your project, which in turn will install the corresponding NLC Profile component. This is only valid for SoC projects. In the case of NCP, only the NLC Profile component is required.

▼ Bluetooth Mesh	Ambient Light Sensor Profile				
▼ NLC					
Basic Lightness Controller Profile	Description Ambient Light Sensor role in an NLC network. This component includes other components needed for an SoC application to act as an Ambient Light Sensor node in an NLC network, as well as components providing the relevant metadata for the role.				
Basic Scene Selector Profile					
Dimming Control Profile					
Occupancy Sensor Profile	Quality Tags PRODUCTION No-Code				
▼ Profile Metadata					
🛇 Ambient Light Sensor Profile Metadata 🔅					
Basic Lightness Controller Profile Metadata 🌼	Dependencies ~				
Basic Scene Selector Profile Metadata 💠	btmesh_nlc_ambient_light_sensor requires 0 components No Dependencies				
Dimming Control Profile Metadata 💠 💠	Dependents				
Occupancy Sensor Profile Metadata 🌼	0 components require btmesh_nlc_ambient_light_sensor No Dependent Components				

Figure 9. Simplicity Studio 5 Component Manager

## **Simplicity Studio**

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!



www.silabs.com/IoT



www.silabs.com/simplicity



www.silabs.com/quality



Support & Community www.silabs.com/community

#### Disclaimer

Silicon Labs intends to provide customers with the latest, accurate, and in-depth documentation of all peripherals and modules available for system and software implementers using or intending to use the Silicon Labs products. Characterization data, available modules and peripherals, memory sizes and memory addresses refer to each specific device, and "Typical" parameters provided can and do vary in different applications. Application examples described herein are for illustrative purposes only. Silicon Labs reserves the right to make changes without further notice to the product information, specifications, and descriptions herein, and does not give warranties as to the accuracy or completeness of the included information. Without prior notification, Silicon Labs may update product firmware during the manufacturing process for security or reliability reasons. Such changes will not alter the specifications or the performance of the product. Silicon Labs shall have no liability for the consequences of use of the information supplied in this document. This document does not imply or expressly grant any license to design or fabricate any integrated circuits. The products are not designed or authorized to be used within any FDA Class III devices, applications for which FDA premarket approval is required or Life Support Systems without the specific written consent of Silicon Labs. A "Life Support System" is any product or system intended to support or sustain life and/or health, which, if it fails, can be reasonably expected to result in significant personal injury or death. Silicon Labs products are not designed or authorized for military applications. Silicon Labs product shall under no circumstances be used in weapons of mass destruction including (but not limited to) nuclear, biological or chemical weapons, or missiles capable of delivering such weapons. Silicon Labs disclaims all express and implied warranties and shall not be responsible or liable for any injuries or damages related to use of a Silicon Lab

#### **Trademark Information**

Silicon Laboratories Inc.<sup>®</sup>, Silicon Laboratories<sup>®</sup>, Silicon Labs<sup>®</sup>, SiLabs<sup>®</sup> and the Silicon Labs logo<sup>®</sup>, Bluegiga<sup>®</sup>, Bluegiga Logo<sup>®</sup>, EFM<sup>®</sup>, EFM32<sup>®</sup>, EFR, Ember<sup>®</sup>, Energy Micro, Energy Micro logo and combinations thereof, "the world's most energy friendly microcontrollers", Redpine Signals<sup>®</sup>, WiSeConnect, n-Link, ThreadArch<sup>®</sup>, EZLink<sup>®</sup>, EZRadio<sup>®</sup>, EZRadio<sup>®</sup>, Gecko<sup>®</sup>, Gecko OS, Gecko OS Studio, Precision32<sup>®</sup>, Simplicity Studio<sup>®</sup>, Telegesis, the Telegesis Logo<sup>®</sup>, USBXpress<sup>®</sup>, Zentri, the Zentri logo and Zentri DMS, Z-Wave<sup>®</sup>, and others are trademarks or registered trademarks of Silicon Labs. ARM, CORTEX, Cortex-M3 and THUMB are trademarks or registered trademarks of ARM Holdings. Keil is a registered trademark of ARM Limited. Wi-Fi is a registered trademark of the Wi-Fi Alliance. All other products or brand names mentioned herein are trademarks of their respective holders.



Silicon Laboratories Inc. 400 West Cesar Chavez Austin, TX 78701 USA

## www.silabs.com