

**Please note that Cypress is an Infineon Technologies Company.**

The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

**Continuity of document content**

The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

**Continuity of ordering part numbers**

Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.



**ModusToolbox™**



# MeshClient and ClientControlMesh App User Guide

Document Number. 002-26575 Rev. \*C

Cypress Semiconductor  
198 Champion Court  
San Jose, CA 95134-1709  
[www.cypress.com](http://www.cypress.com)

# Contents

<b>About This Document.....</b>	<b>3</b>
Acronyms and Abbreviations .....	3
IoT Resources and Technical Support .....	3
<b>1 Overview .....</b>	<b>4</b>
1.1 Mesh Libraries .....	6
<b>2 MeshClient Applications Overview .....</b>	<b>7</b>
2.1 Provisioning .....	7
2.2 Configuration .....	7
2.3 Control.....	7
<b>3 Using the MeshClient Application.....</b>	<b>8</b>
3.1 Creating and Opening a Mesh Network.....	8
3.2 Adding a Node.....	10
3.3 Creating and Managing Groups.....	14
3.4 Configuring Devices .....	16
<b>4 References .....</b>	<b>19</b>
<b>Document Revision History .....</b>	<b>20</b>
<b>Worldwide Sales and Design Support.....</b>	<b>21</b>

## About This Document

This document provides quick start instructions for the MeshClient and the ClientControlMesh applications, which are part of WICED® Studio and ModusToolbox™.

## Acronyms and Abbreviations

In most cases, acronyms and abbreviations are defined on first use. For a comprehensive list of acronyms and other terms used in Cypress documents, go to [www.cypress.com/glossary](http://www.cypress.com/glossary).

## IoT Resources and Technical Support

Cypress provides a wealth of data at <http://www.cypress.com/internet-things-iot> to help you to select the right IoT device for your design, and quickly and effectively integrate the device into your design. Cypress provides customer access to a wide range of information, including technical documentation, schematic diagrams, product bill of materials, PCB layout information, and software updates. Customers can acquire technical documentation and software from the Cypress Support Community website (<http://community.cypress.com/>).

# 1 Overview

Bluetooth SDK in WICED® Studio and ModusToolbox offer wide variety of Bluetooth SIG Mesh 1.0-related products. One of them is a set of portable libraries that can be used on any platform to create an application to provision and control the mesh. WICED Studio and ModusToolbox support Bluetooth Mesh on CYW20706, CYW20735, CYW20719, CYW20819, and Cypress modules such as CYBT-213043-02 based on these silicon devices.

The MeshClient and the ClientControlMesh applications provide a sample Windows implementation that show how to use interfaces exposed by the mesh libraries. The MeshClient works only with Windows 10. The MeshClient application uses PC's built-in Bluetooth radio, or an external Bluetooth dongle to communicate with Bluetooth mesh. The MeshClient application implements all layers of the mesh stack. The ClientControlMesh application implements only the application layer. It uses the Mesh Models and Mesh Core libraries residing on the embedded device that requires a Cypress device to act as a client and hence requires an extra evaluation board to be connected to the PC for mesh operation. Any of the Cypress devices that support Bluetooth mesh can be used for this application irrespective of the device used by the mesh nodes. The ClientControlMesh can be used with any version of Windows operating system.

The MeshClient and the ClientControlMesh Windows applications are installed with WICED Studio and ModusToolbox installation as part of Bluetooth SDK.

## ■ App paths in WICED Studio (Change 6.4 to appropriate version based on the WICED Studio version being used)

If the default path for the installation is used, the **MeshClient** project is in:

*C:\Users\<user>\Documents\WICED-Studio-6.4\common\apps\snip\mesh\peerapps\Windows\MeshClient*

To open the application on Windows machine:

- ☐ Go to: *C:\Users\<user>\Documents\WICED-Studio-6.4\common\apps\snip\mesh\peerapps\Windows\MeshClient\Release\x86*
- ☐ Double-click the **MeshClient** application.

The Windows **ClientControlMesh** project is in:

*C:\Users\<user>\Documents\WICED-Studio-6.4\common\apps\snip\mesh\VS\_ClientControl*

To open the application on a Windows machine:

- ☐ Go to: *C:\Users\<user>\Documents\WICED-Studio-6.4\common\apps\snip\mesh\VS\_ClientControl\Release*
- ☐ Double-click the **ClientControlMesh** application.

A version of the **ClientControlMesh** project is provided for Linux and OSX and can be found in:

*C:\Users\<user>\Documents\WICED-Studio-6.4\common\apps\snip\mesh\Qt\_ClientControl*

## ■ App paths in ModusToolbox

**MeshClient** and **ClientControlMesh** are provided in the wiced\_btstack project in the ModusToolbox IDE, which is created and used by any WICED Bluetooth application created in the IDE.

The **MeshClient** project (Windows only) can be found in the IDE Project Explorer pane in:

*wiced\_btstack\tools\btstack-peer-apps-mesh\Windows\MeshClient\Release\x86*

Similarly, the Windows **ClientControlMesh** project is in:

*wiced\_btstack\tools\btstack-host-apps-mesh\VS\_ClientControl*

A version of the **ClientControlMesh** project is provided for Linux and OSX and can be found in:

*wiced\_btstack\tools\btstack-host-apps-mesh\Qt\_ClientControl*

To open the applications, select either the embedded Mesh application project or the wiced\_btstack project in the ModusToolbox IDE Project Explorer pane, then click the appropriate tool execution link from the Quick Panel.

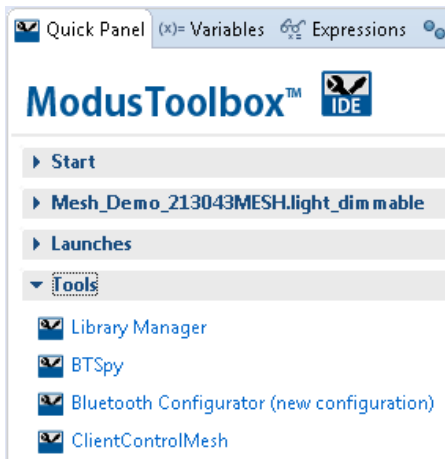


Figure 1. Tools in Quick Panel

Alternately, navigate to the workspace location in the file system, and locate the executable under the appropriate path based on the Project Explorer locations listed above, and double click to run the programs.

The Windows ClientControlMesh and the MeshClient applications can be built using Microsoft Visual Studio 2019 or later release.

### Operating System (OS) Requirements

- **MeshClient** - The MeshClient application relies upon the Windows Bluetooth stack version which is available only in Windows 10 Creators Update. While launching the application, if you see the error "This application requires Windows 10 Creator Updates", install these updates to run this application. Go to the following link click **Update now** to download and install these updates:  
<https://www.microsoft.com/en-us/software-download/windows10>
- **ClientControlMesh** - The ClientControlMesh does not use Windows stack and can be executed on any version of Windows OS.

Figure 2 shows the software block diagram of the MeshClient (left) and ClientControlMesh (right) applications.

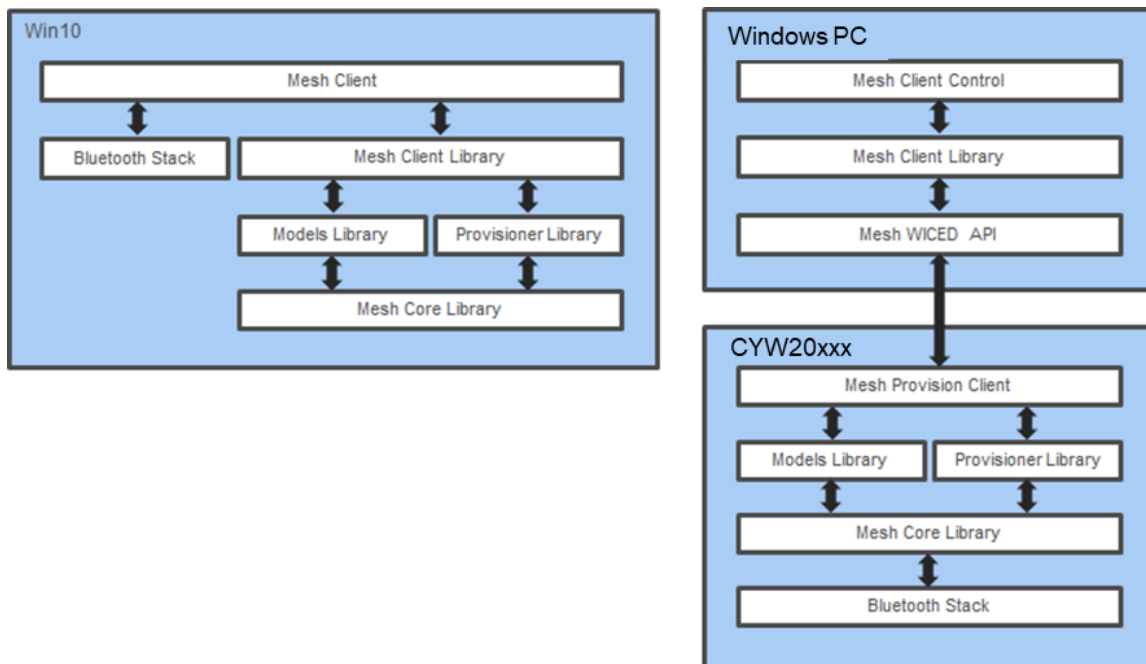


Figure 2. MeshClient (Left) and ClientControlMesh (Right) Software Block Diagram

The MeshClient application uses Bluetooth stack as it exists on Windows 10 OS. It uses GATT Proxy connection [1] to control the mesh. The ClientControlMesh application uses Bluetooth stack of the Cypress silicon. It can support both GATT Proxy and advertising channel to provision and control mesh devices.

The MeshClient and ClientControlMesh applications expose the functionality of various client models defined in the Mesh specifications including Configuration, Health, Default Transition Time, OnOff, Level, Power OnOff, Light Lightness, Light HSL, Light CTL, Sensor, and a sample Vendor-Specific client models. Other client and server models can be added in future releases.

## 1.1 Mesh Libraries

The MeshClient library executes the state machines required for provisioning and configuration. It provides an interface to the application to test the mesh functionality. The library maintains the database for the mesh network.

In the MeshClient application, to execute Bluetooth functionality such as starting BLE scan, establishing a connection to a specific device, or sending a data packet, the MeshClient library executes methods that are provided by the MeshClient application, which in turn uses Bluetooth Stack of the OS. On the other hand, in the ClientControlMesh application, the MeshClient library uses Mesh WICED API to control embedded application to perform all the mesh related work.

The Mesh Core, Models, Provisioner and Core libraries implement all the functionality as defined in the Bluetooth SIG Mesh Profile [1] and Mesh Models [2] specifications.

## 2 MeshClient Applications Overview

### 2.1 Provisioning

Provisioning is a process of adding new nodes into a Mesh network. Provisioning is performed by a special node called a “Provisioner”. The MeshClient/ClientControlMesh applications perform as a Provisioner in the mesh network. MeshClient and ClientControlMesh applications maintain the database for the network, initiate the scan for unprovisioned devices, and perform the provisioning procedure as defined in Mesh Profile specification [1]. As a result of execution of the provisioning procedure, the Provisioner provides to the new node a bare minimum of the information to be a part of the mesh network (network key, IV index) and establishes the Device Key for the new node that is used between the Provisioner and the node during the configuration stage.

While the Mesh specification allows provisioning over the Advertising (PB-ADV) and GATT (PB-GATT) bearers, the MeshClient uses the GATT bearer only because it relies on the Microsoft Bluetooth stack as transport. The MeshClient Control can be configured to use any bearer.

### 2.2 Configuration

It is not enough just to provision a device to make it a fully functional mesh node of the mesh network. The following is a partial list of things that the Provisioner needs to perform during the configuration:

- Read the new node’s composition data to find out the device capabilities. For example, based on the information in the composition data, the Provisioner can figure out if it is a switch, a light bulb, or some other device.
- Set up the features that the new node should support. For example, if the node supports GATT Proxy or Friend role, the Provisioner needs to specify if the node should use the feature.
- Add network keys if the node should also be a part of other subnets, and add application keys for use with various models.
- Bind appropriate application keys to appropriate models of the new node. For example, the Provisioner can specify one application key to be used to configure the bulb and different application key to control the bulb.
- Configure various network parameters. For example, the Provisioner can specify the number of times the node should retransmit the message if it performs as a relay, and the number of times and frequency at which the node should publish the status messages.
- Configure the new device to be a part of a group.
- Configure clients, for example on/off switch, to control a specific server such as a light bulb, or a group of servers such as all light bulbs in a room.

### 2.3 Control

After the new node has been provisioned and configured, it can send and receive messages to and from devices in the same mesh network. For example, when you provision and configure a switch, the switch can send ON/OFF commands to a bulb or to all bulbs in the room.

The MeshClient and ClientControlMesh can act as various actuators including an on/off switch, a dimmer and a color control. For that purpose, they support corresponding client models and can send various Get/Set commands to control mesh devices. For example, the application can send a command to dim the light bulb to a certain level, or to adjust the color temperature of the light bulb.

Similar to any other client, the application can send messages to a single device or to a group of the devices. The replies are typically received from each device. When the application addresses the group with acknowledged message, each device in the group would send a reply. The mesh stack monitors how many replies have been received, and if reply from some specific node is not received, the Device Unreachable message is sent to the application.

Depending on the type of the device, some devices may act purely as clients, others like servers, and some can act simultaneously as client and servers. A simple generic on/off switch is an example of a clean client. An HSL light bulb is an example of a pure server. There can be a node which is wired to two bulbs. There can be a power strip with one switch and several outlets, and the switch can be configured to control one of the outlets, or all outlets or the strip, or several strips.



## 3 Using the MeshClient Application

See Overview for the location and execution instructions for MeshClient and ClientControlMesh applications.

If a Windows 10 PC is used, it is recommended to use MeshClient application as it does not require an external device to run Bluetooth stack.

The user interface of the MeshClient and the ClientControlMesh applications are very similar. The only key difference between the two applications is COM port selection and Baud rate setting. These fields are not available in the MeshClient app as it uses PC's in-built Bluetooth. The ClientControlMesh application requires to talk to an external Cypress Evaluation board/device over the HCI UART. Hence these fields are provided in the ClientControlMesh application. See Using the MeshClient Application to learn how to select COM port and baud rate. Rest of this document uses screenshots from the MeshClient app as most of other fields and buttons are similar in both apps.

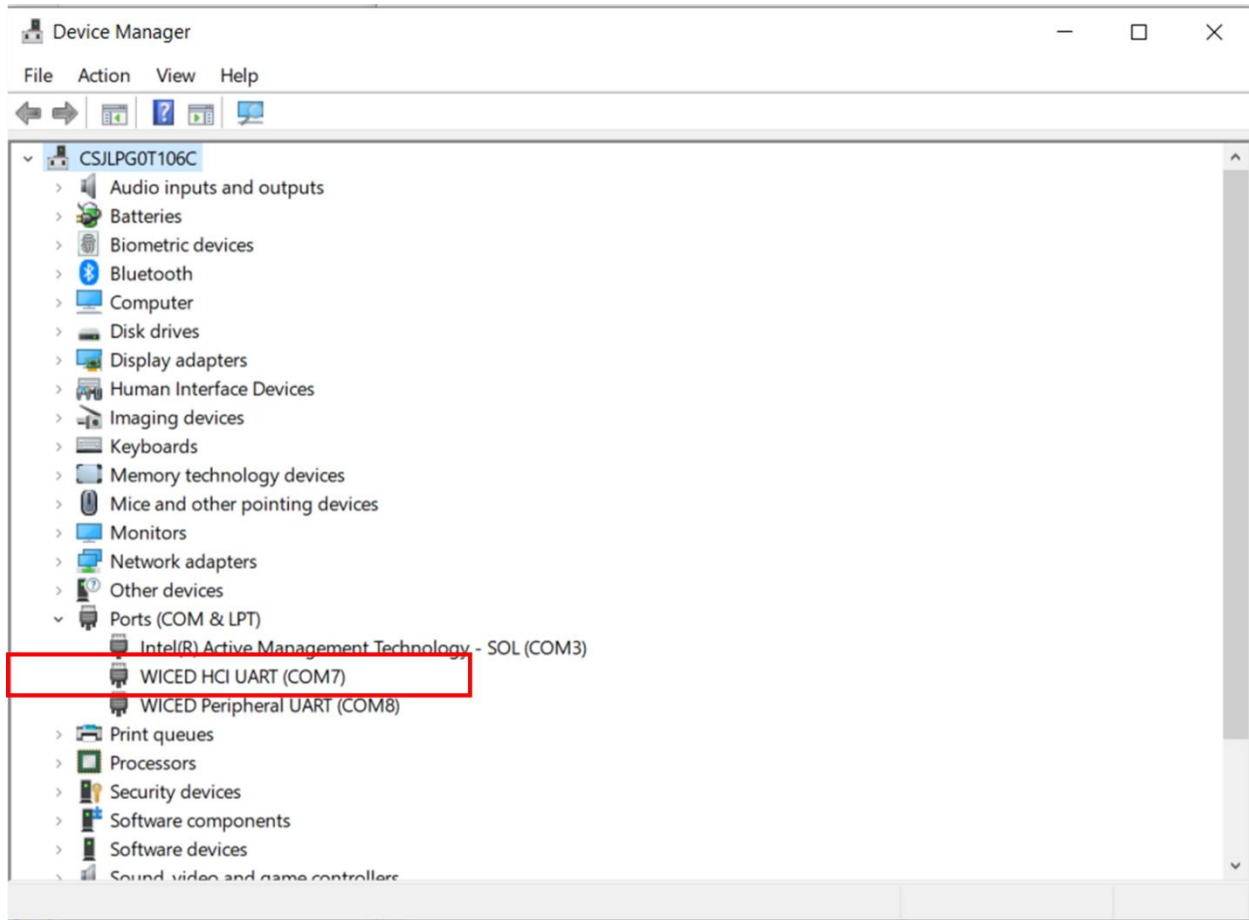
### 3.1 Creating and Opening a Mesh Network

**Step 1:** Jump to step 2 if you are using the MeshClient application. Continue here if using the ClientControlMesh application.

Program one evaluation board with mesh\_provision\_client snip application. This snip is available as part of WICED Studio installation under `apps\snip\mesh\mesh_provision_client`. If using WICED Studio, refer to the respective kit's user guide to learn how to build and download an application on to the board. If using ModusToolbox, see the Getting Started with Bluetooth Mesh Application Note [4] to learn how to download code examples from GitHub and program the board.

Once the board is programmed and connected to the PC, check the COM port number for the HCI UART. To check the COM port number, go to **Device Manager** on your PC and expand **Ports (COM & LPT)**. Here, look for **WICED HCI UART**. This is the COM port number to be used in the ClientControlMesh application. See the following screenshot.

**Note:** If the PC is detecting HCI and PUART ports as **USB Serial Port** without any distinction, then the lower COM port number is likely to be HCI UART's COM port number.



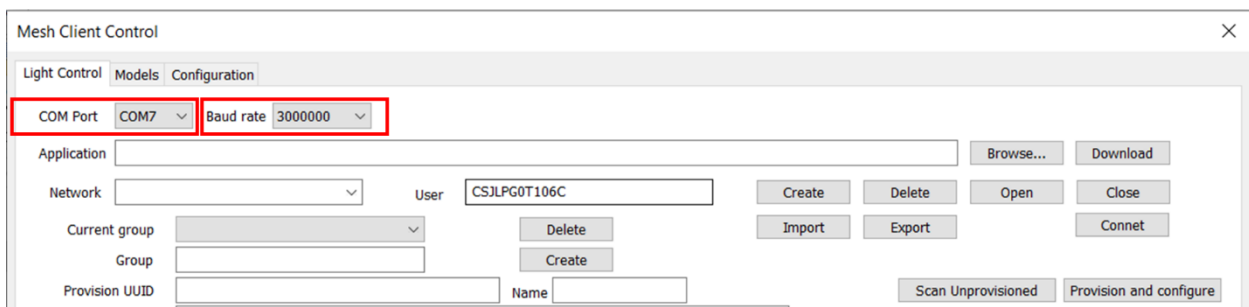
Once the COM port is identified, open the ClientControlMesh application.

**(Note:** If the board is connected to PC after opening the ClientControlMesh application, the ClientControlMesh application will not detect the COM port. So, make sure you open application after the board is connected and enumerated.)

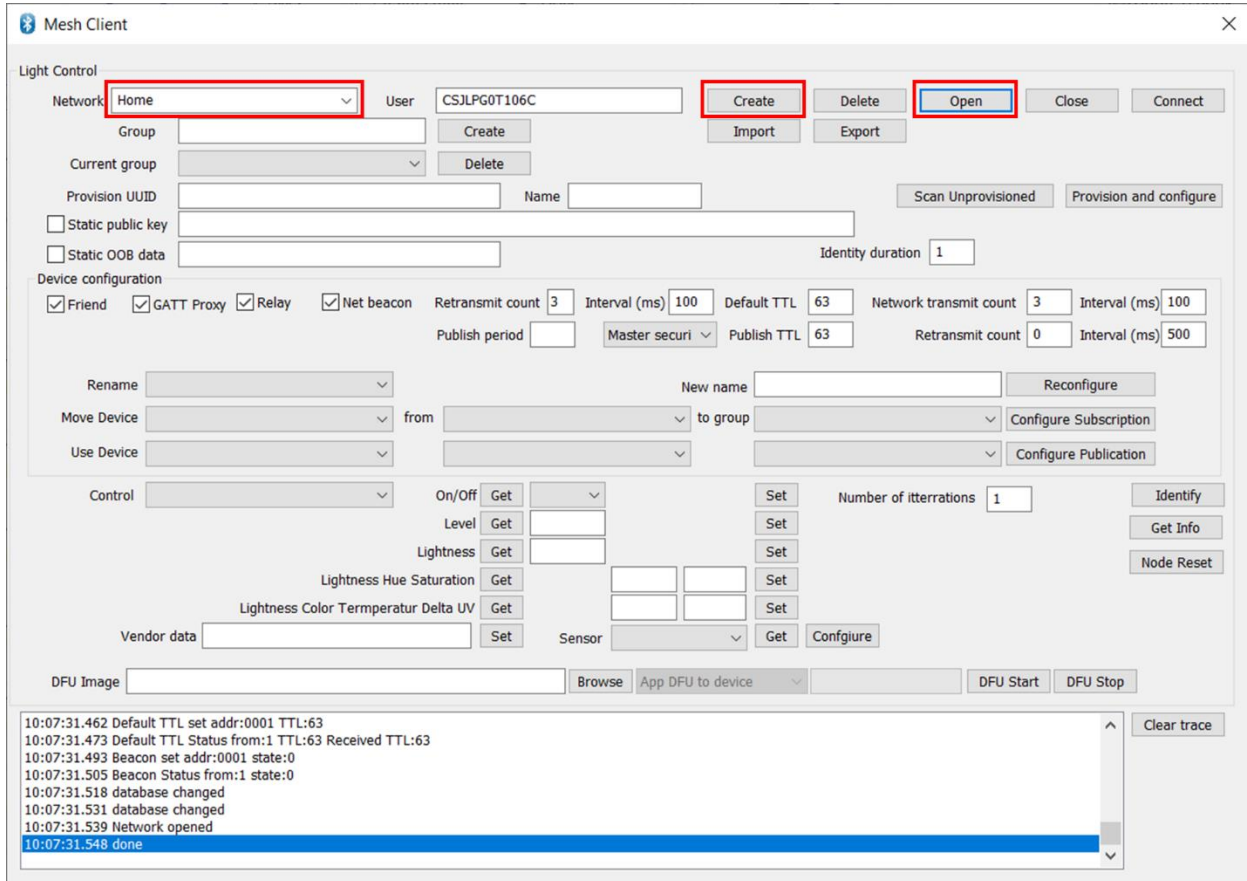
Select this HCI-UART COM port from the **COM port** dropdown menu.

Then, select **Baud rate** 300000 if using CYW920819EVB-02 or any other chip-on-board evaluation boards.

Select **Baud rate** 115200 if the CYBT-213043-EVAL or CYBT-213043-MESH EZ-BT Mesh evaluation kits are being used to run provision control client.



**Step 2:** In the **Network** field, type the string that you want to use as the network name. Click **Create** and then, click **Open**. See the following screenshot.



When a network is created, the MeshClient creates the required network attributes such as the mesh UUID, network and application keys, and saves the information in the mesh database which is stored in a JSON file in the directory where the application is started from. The schema of the Mesh Provisioner database is described in corresponding document from Bluetooth SIG [3].

There can be multiple networks controlled by the same PC, for example “Home”, “Office”, “Parent’s house”.

When you click **Open**, the MeshClient configures the stack with the parameters of the selected network. Similarly, the ClientControlMesh talks over the selected COM port to configure the stack running on the embedded platform. The “done” trace at the end of the configuration process indicates that the stack has been configured successfully.

## 3.2 Adding a Node

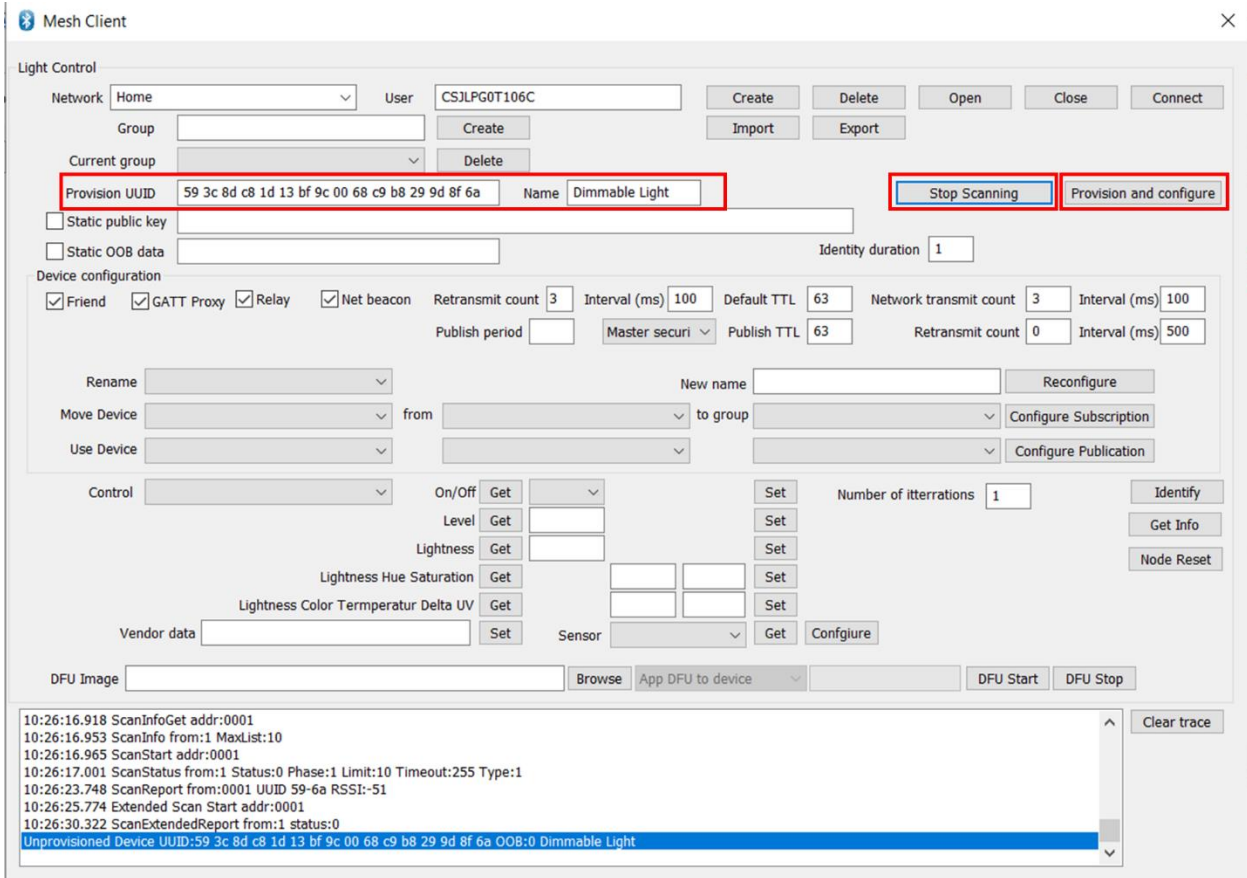
Use WICED Studio or ModusToolbox to build and download one of the mesh samples to a WICED evaluation board. In the following description, the BLE\_Mesh\_LightDimmable code example is used. See the respective kits’ user guide/getting started guide to learn how to program the board.

Do the following to provision a new node and also see the following screenshot:

1. In the MeshClient window, click **Scan Unprovisioned**.

The title of the button changes to **Stop Scanning** to indicate that the scan is active.

2. The trace window displays the UUIDs of the devices that are in the radio range. The **Provisioned UUID** field is automatically filled with the UUID of the last discovered device.
3. When you see the device that you want to work with, click **Stop Scanning**.
4. Click **Provision and configure**.



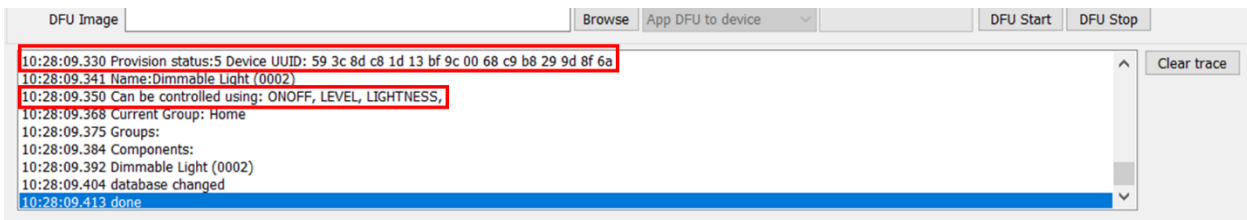
The Mesh Client application window displays the following sections:

- Light Control:** Includes fields for Network (Home), User (CSJLPG0T106C), Group, Current group, Provision UUID (59 3c 8d c8 1d 13 bf 9c 00 68 c9 b8 29 9d 8f 6a), Name (Dimmable Light), and buttons for Stop Scanning and Provision and configure.
- Static public key:** A text input field.
- Static OOB data:** A text input field.
- Device configuration:** Includes checkboxes for Friend, GATT Proxy, Relay, and Net beacon. It also features fields for Retransmit count (3), Interval (ms) (100), Default TTL (63), Network transmit count (3), Interval (ms) (100), Publish period, Master security, Publish TTL (63), and Retransmit count (0). There are also buttons for Rename, Move Device, Use Device, and Reconfigure.
- Control:** Includes a dropdown for Control, buttons for On/Off, Level, Lightness, and Lightness Hue Saturation, and a field for Number of iterations (1). There are also buttons for Identify, Get Info, and Node Reset.
- Vendor data:** A text input field.
- Sensor:** A dropdown menu.
- DFU Image:** A text input field and buttons for Browse, App DFU to device, DFU Start, and DFU Stop.
- Trace window:** A scrollable area showing the following log entries:
 

```

10:26:16.918 ScanInfoGet addr:0001
10:26:16.953 ScanInfo from:1 MaxList:10
10:26:16.965 ScanStart addr:0001
10:26:17.001 ScanStatus from:1 Status:0 Phase:1 Limit:10 Timeout:255 Type:1
10:26:23.748 ScanReport from:0001 UUID 59-6a RSSI: -51
10:26:25.774 Extended Scan Start addr:0001
10:26:30.322 ScanExtendedReport from:1 status:0
Unprovisioned Device UUID:59 3c 8d c8 1d 13 bf 9c 00 68 c9 b8 29 9d 8f 6a OOB:0 Dimmable Light
      
```

The provisioning and configuration process consists of several of steps. While the process is being executed, the status is displayed in the trace window. At the end of the process, **Provision status:5** appears in the trace window, indicating that the process has been completed successfully. The MeshClient also queries the library for the methods available for the application to control the device. For example, the provision device in the trace below can be controlled using On Off, Level as well as Lightness.



The trace window displays the following log entries:

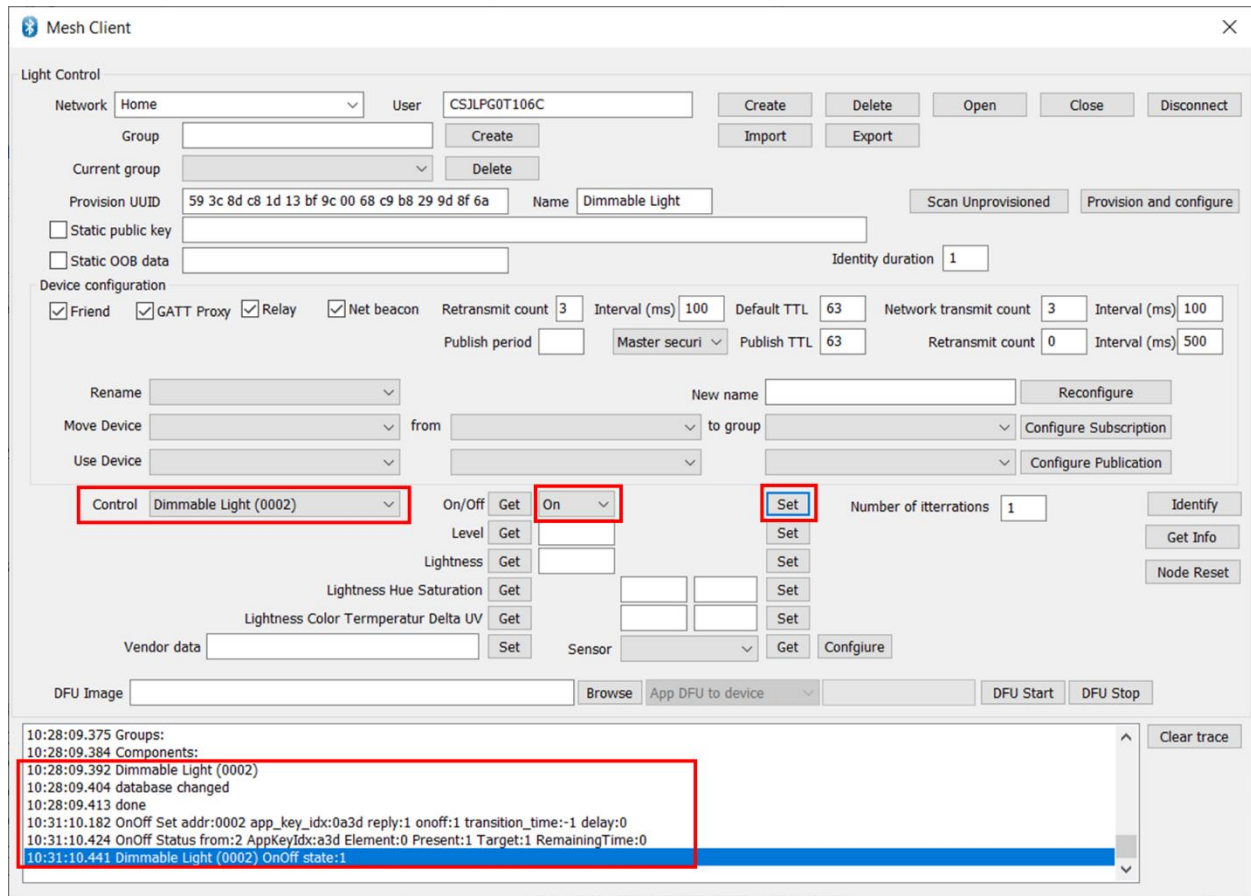
```

10:28:09.330 Provision status:5 Device UUID: 59 3c 8d c8 1d 13 bf 9c 00 68 c9 b8 29 9d 8f 6a
10:28:09.341 Name:Dimmable Light (0002)
10:28:09.350 Can be controlled using: ONOFF, LEVEL, LIGHTNESS,
10:28:09.368 Current Group: Home
10:28:09.375 Groups:
10:28:09.384 Components:
10:28:09.392 Dimmable Light (0002)
10:28:09.404 database changed
10:28:09.413 done
  
```

The trace window will print out the results of the operation.

If the device has been configured to be a GATT Proxy, the MeshClient will keep the connection to the new device open. If needed, click **Disconnect** to drop the GATT connection. When MeshClient is not connected to the mesh network, click **Connect** to establish the GATT connection.

The device is ready to use. For example, you can select the device and issue a **Get** the **On/Off status** command, or set a desired state to **On** and issue a **Set** command. Before sending any command to the node, ensure that app is connected to the proxy node i.e. the **Connect/Disconnect** button must shows **Disconnect**.



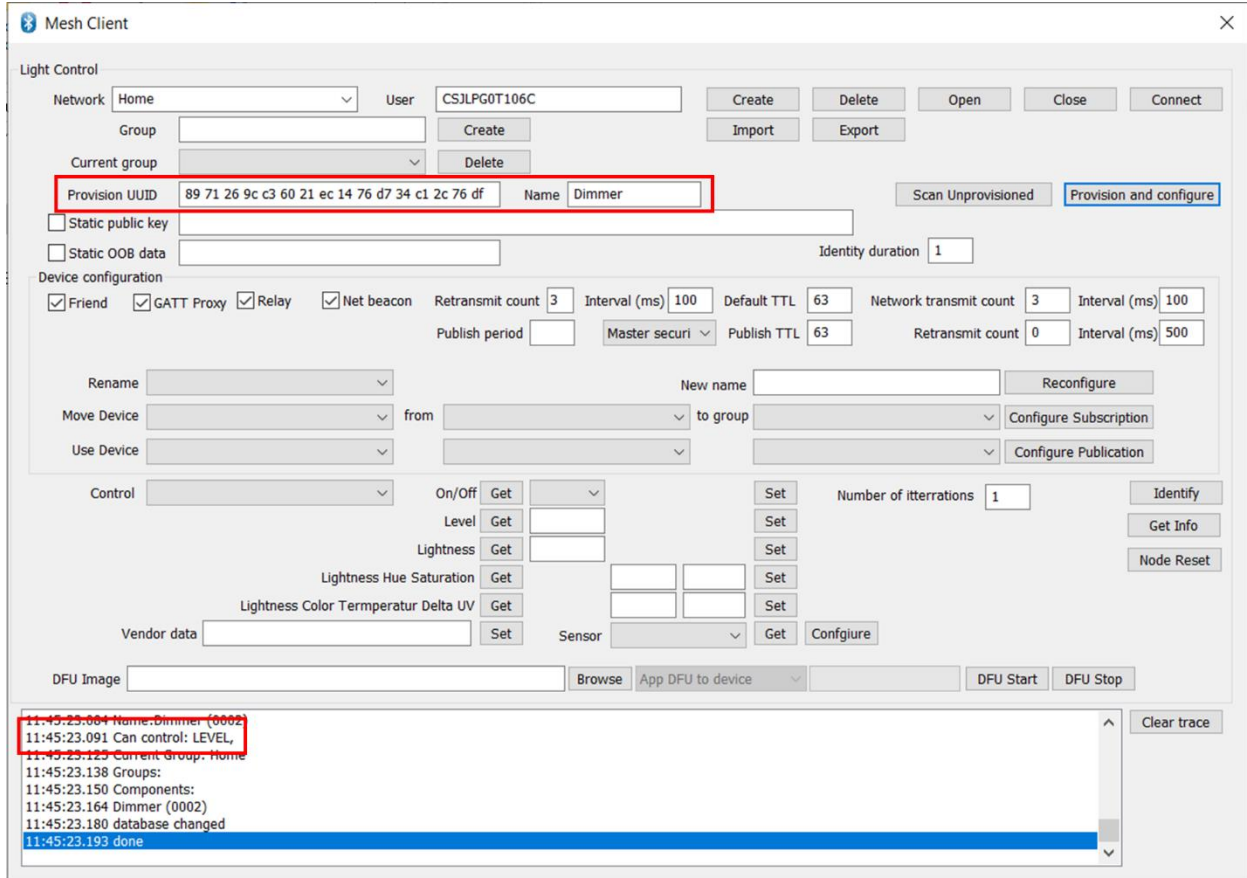
The screenshot shows the Mesh Client application interface. The top section is titled "Light Control" and includes fields for Network (Home), User (CSJLPG0T106C), and buttons for Create, Delete, Open, Close, and Disconnect. Below this are fields for Group, Current group, Provision UUID, and Name (Dimmable Light). There are also buttons for Scan Unprovisioned and Provision and configure. The middle section is titled "Device configuration" and includes checkboxes for Friend, GATT Proxy, Relay, and Net beacon. It also has fields for Retransmit count, Interval (ms), Default TTL, Network transmit count, Publish period, Master security, Publish TTL, and Retransmit count. The bottom section is titled "Control" and includes a dropdown menu for Dimmable Light (0002). It has buttons for On/Off, Get, On, Set, Level, Get, Lightness, Get, Lightness Hue Saturation, Get, Lightness Color Temperatur Delta UV, Get, Vendor data, Set, Sensor, Get, and Configure. The bottom-most section is a log window showing the following messages:

```

10:28:09.375 Groups:
10:28:09.384 Components:
10:28:09.392 Dimmable Light (0002)
10:28:09.404 database changed
10:28:09.413 done
10:31:10.182 OnOff Set addr:0002 app_key_idx:0a3d reply:1 onoff:1 transition_time:-1 delay:0
10:31:10.424 OnOff Status from:2 AppKeyIdx:a3d Element:0 Present:1 Target:1 RemainingTime:0
10:31:10.441 Dimmable Light (0002) OnOff state:1

```

When a switch/dimmer or any other client is provisioned, instead of the “Can be controlled using” tag, the trace window will show “Can control”. For example, following screenshot shows the “Can Control” trace when a dimmer (level client model) is provisioned. As it is a level client, it can control level as shown in the traces.



The screenshot shows the Mesh Client application window. The 'Light Control' section is active, displaying the 'Provision UUID' as 89 71 26 9c c3 60 21 ec 14 76 d7 34 c1 2c 76 df and the 'Name' as Dimmer. The 'Device configuration' section shows various settings like Friend, GATT Proxy, Relay, Net beacon, Retransmit count, Interval (ms), Default TTL, Network transmit count, Publish period, Master security, Publish TTL, and Retransmit count. The 'Control' section shows various controls like On/Off, Level, Lightness, Lightness Hue Saturation, Lightness Color Temperature Delta UV, Vendor data, and Sensor. The 'DFU Image' section shows a 'Browse' button and 'App DFU to device' dropdown. The 'Trace' window at the bottom shows the following log entries:

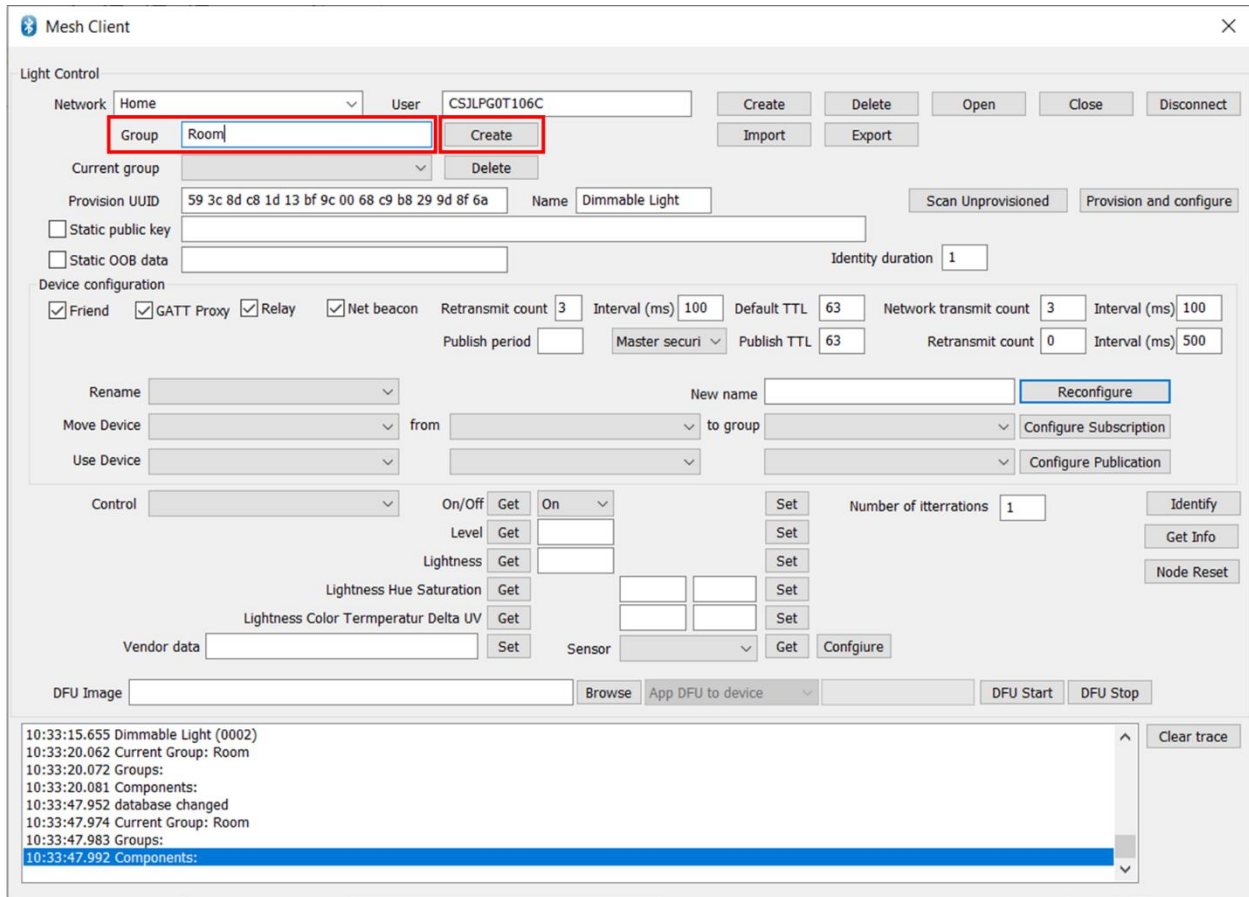
```

11:45:23.684 Name: Dimmer (0002)
11:45:23.091 Can control: LEVEL,
11:45:23.129 Current Group: Home
11:45:23.138 Groups:
11:45:23.150 Components:
11:45:23.164 Dimmer (0002)
11:45:23.180 database changed
11:45:23.193 done
  
```



### 3.3 Creating and Managing Groups

A group can be used to issue commands to several devices at the same time. To create a group in the current network, type in a string in the **Group** field, and click **Create** next to it.



**Mesh Client**

Light Control

Network: Home User: CSJLPG0T106C [Create] [Delete] [Open] [Close] [Disconnect]

Group: Room [Create] [Import] [Export]

Current group: [Delete]

Provision UUID: 59 3c 8d c8 1d 13 bf 9c 00 68 c9 b8 29 9d 8f 6a Name: Dimmable Light [Scan Unprovisioned] [Provision and configure]

☐ Static public key

☐ Static OOB data Identity duration: 1

Device configuration

☒ Friend ☒ GATT Proxy ☒ Relay ☒ Net beacon Retransmit count: 3 Interval (ms): 100 Default TTL: 63 Network transmit count: 3 Interval (ms): 100

Publish period: Master securi Publish TTL: 63 Retransmit count: 0 Interval (ms): 500

Rename: [Reconfigure]

Move Device: from to group [Configure Subscription]

Use Device: [Configure Publication]

Control: On/Off Get On [Set] Number of iterations: 1 [Identify]

Level Get [Set]

Lightness Get [Set]

Lightness Hue Saturation Get [Set]

Lightness Color Temperatur Delta UV Get [Set]

Vendor data Set Sensor [Get] [Configure]

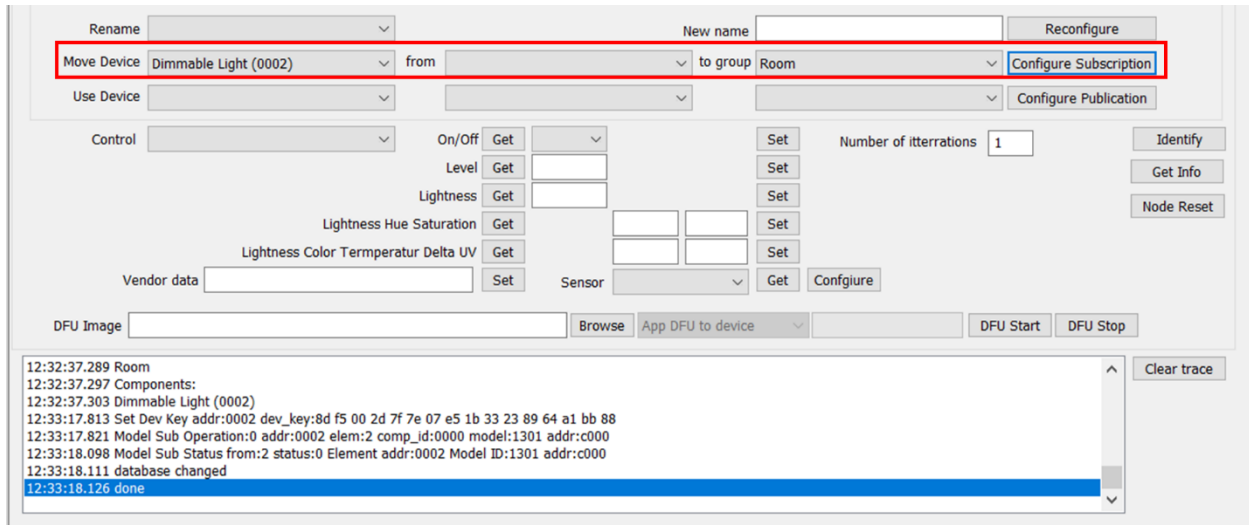
DFU Image: [Browse] App DFU to device [DFU Start] [DFU Stop]

10:33:15.655 Dimmable Light (0002)  
 10:33:20.062 Current Group: Room  
 10:33:20.072 Groups:  
 10:33:20.081 Components:  
 10:33:47.952 database changed  
 10:33:47.974 Current Group: Room  
 10:33:47.983 Groups:  
 10:33:47.992 Components:

[Clear trace]

You can then select the group in the Current group field. The nodes now will be provisioned in to the created group.

You can also move devices between previously created groups. **Configure Subscriptions** puts the device in to a requested group, the device is now subscribed to process unicast messages destined to that device as well as messages addressed to the group.



The screenshot shows the MeshClient application interface. The 'Move Device' section is highlighted with a red box, showing 'Dimmable Light (0002)' being moved from the current group to the 'Room' group. The 'Configure Subscription' button is also highlighted. The 'Control' section shows various settings for the device, including 'On/Off', 'Level', 'Lightness', 'Lightness Hue Saturation', 'Lightness Color Temperatur Delta UV', 'Vendor data', and 'Sensor'. The 'DFU Image' section shows a 'Browse' button and 'App DFU to device' dropdown. The trace window at the bottom shows the following log entries:

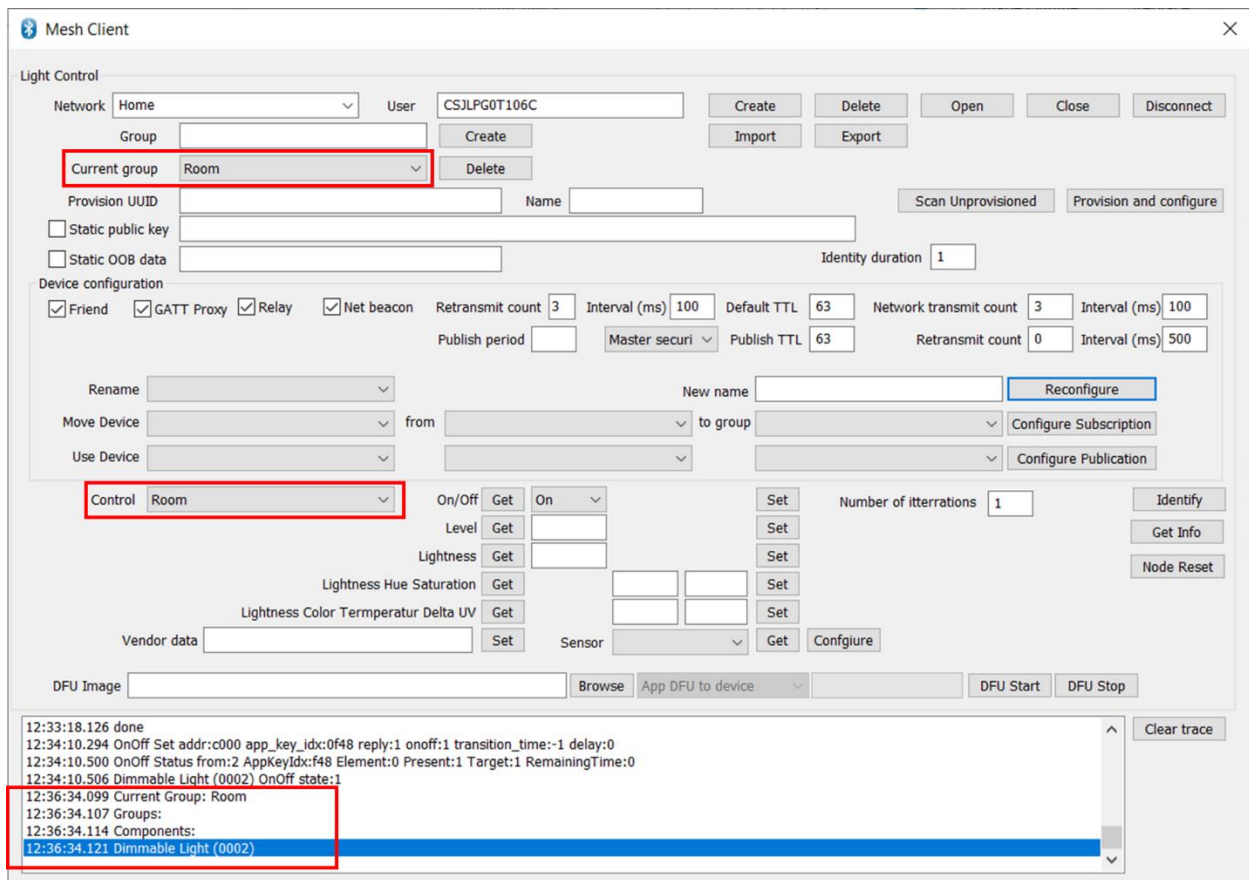
```

12:32:37.289 Room
12:32:37.297 Components:
12:32:37.303 Dimmable Light (0002)
12:33:17.813 Set Dev Key addr:0002 dev_key:8d f5 00 2d 7f 7e 07 e5 1b 33 23 89 64 a1 bb 88
12:33:17.821 Model Sub Operation:0 addr:0002 elem:2 comp_id:0000 model:1301 addr:c000
12:33:18.098 Model Sub Status from:2 status:0 Element addr:0002 Model ID:1301 addr:c000
12:33:18.111 database changed
12:33:18.126 done

```

If a new device is provisioned while the current selected group is 'Room', the device will be assigned to this group. You do not need to perform the Move operation. In the Control field, you can select an individual device, a group address, or the name of the network to unicast, multicast, or broadcast mesh control messages respectively.

When the current group is changed, the trace window will display the content of the new current group.



The screenshot shows the MeshClient application interface. The 'Current group' is set to 'Room'. The 'Control' section shows various settings for the device, including 'On/Off', 'Level', 'Lightness', 'Lightness Hue Saturation', 'Lightness Color Temperatur Delta UV', 'Vendor data', and 'Sensor'. The 'DFU Image' section shows a 'Browse' button and 'App DFU to device' dropdown. The trace window at the bottom shows the following log entries:

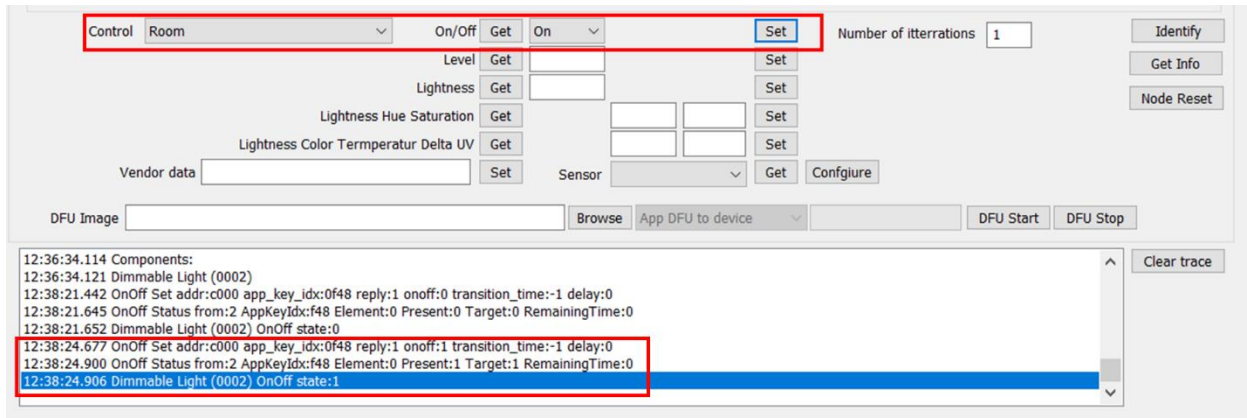
```

12:33:18.126 done
12:34:10.294 OnOff Set addr:c000 app_key_idx:0f48 reply:1 onoff:1 transition_time:-1 delay:0
12:34:10.500 OnOff Status from:2 AppKeyIdx:f48 Element:0 Present:1 Target:1 RemainingTime:0
12:34:10.506 Dimmable Light (0002) OnOff state:1
12:36:34.099 Current Group: Room
12:36:34.107 Groups:
12:36:34.114 Components:
12:36:34.121 Dimmable Light (0002)

```

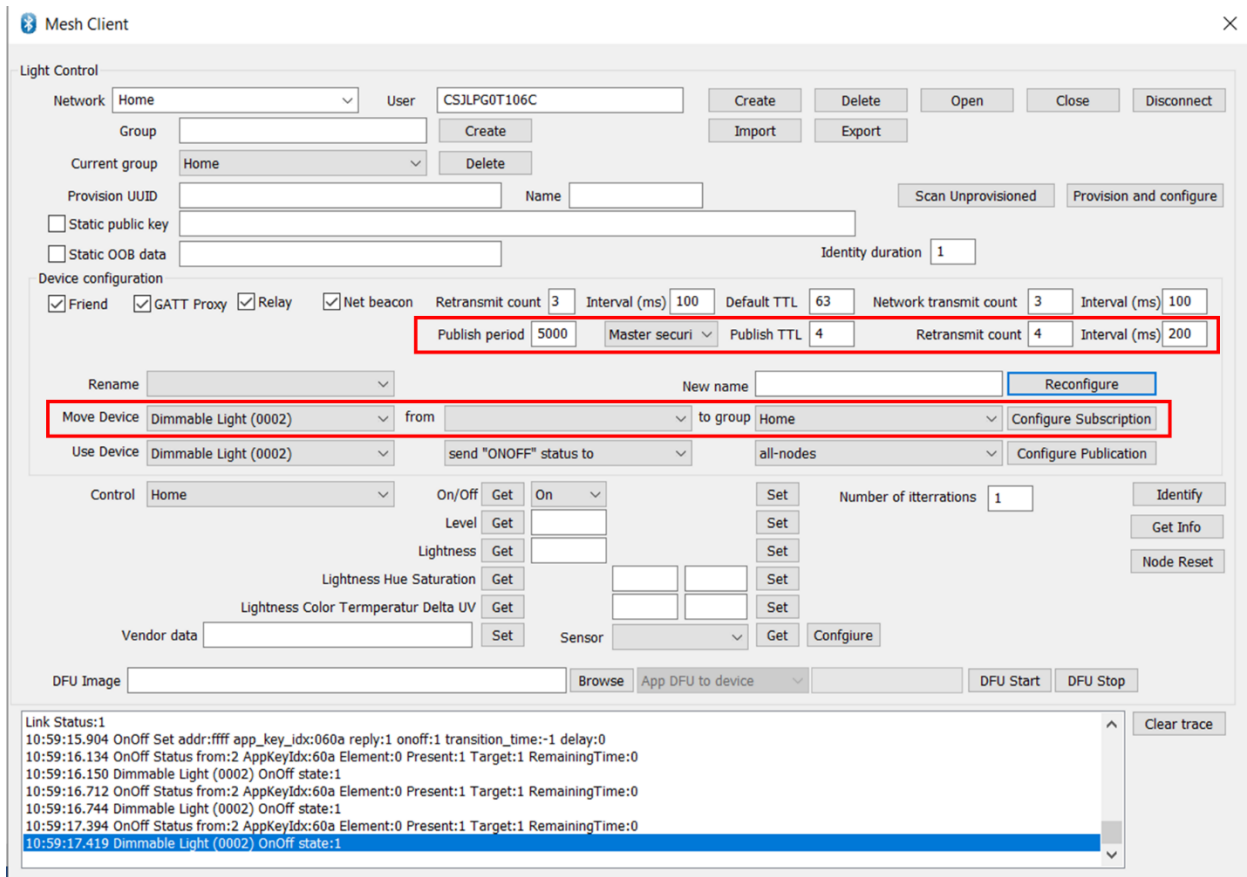


Entire group can be controlled using a single command. Select the group name to be controlled in **Control** field, select the action, and click **Set**.



## 3.4 Configuring Devices

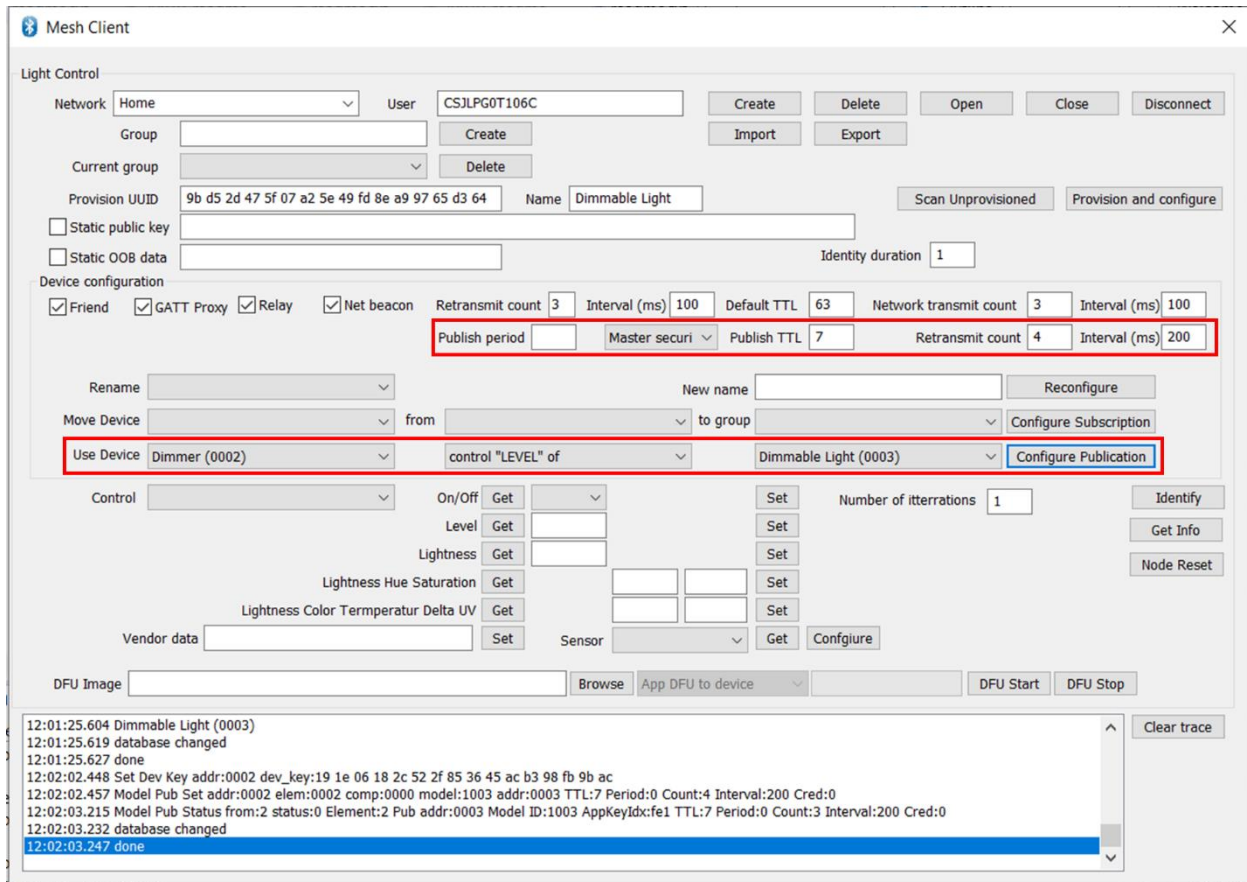
The device configuration section allows you to configure multiple parameters for sending or relaying messages. By default, parameters from the initial provisioning are used; you can specify a new name for an already configured device, select the device in the **Rename** field, and click **Reconfigure**.



A device can also be configured to publish status change to a specific node, and specific group, or to all devices in the network. For example, a light bulb can publish hue/saturation/lightness state periodically.

Similarly, a controlling device can be configured to send messages to individual devices or to a group. This allows you to configure a switch to control one or more lights.

To configure a destination for the messages originated by the device, select the device in the **Use Device** field, select the method (for example, if this is a Dimmer, a LEVEL methods will be available), desired destination, and click **Configure Publication**.



If the Dimmer was provisioned while the group room was selected as the current group, the Dimmer will already be configured to send level to all devices in the group room. However, it can be reconfigured to send messages to any single device, or to any other group in the network.

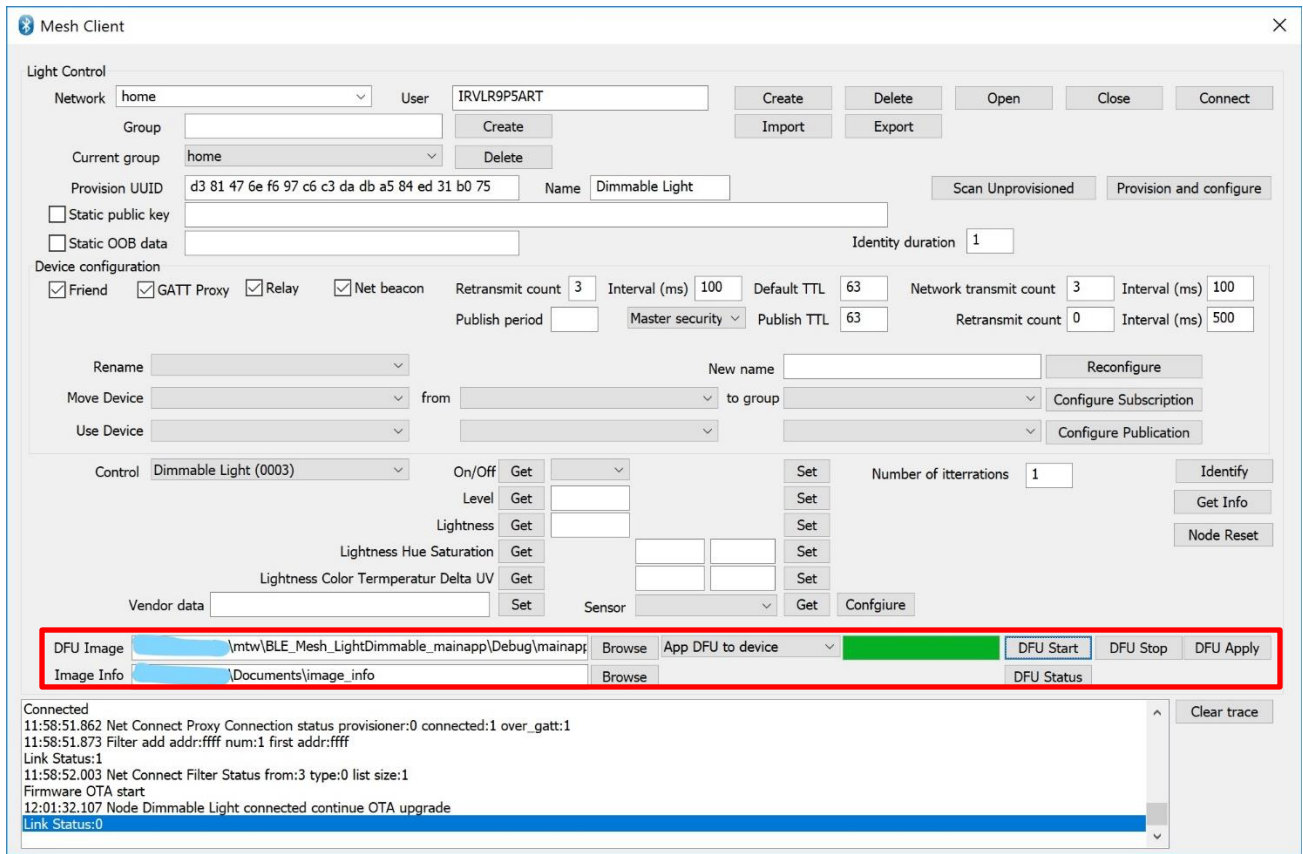
The publication parameters are used from the Publish period, Publish TTL, Retransmit count, and Interval fields.

### 3.5 Over-The-Air Device Firmware Upgrade

The device firmware is updated from the build PC to the development kit using the MeshClient DFU interface. Open the mesh **Network**, named **home** in Figure 3. Scan unprovisioned devices, then Provision and Configure. Then, click **Disconnect** (Connect toggles as shown in Figure 3). Next, select the device from the **Control** dropdown list and click **Connect**. Use **Browse** to select the update image and the image information file. Select **App DFU to device** from the transfer type dropdown list. Click **DFU Start** to begin the transfer.

The image information file is a plain text file formatted as follows:

```
# Company ID (2 bytes)
CID=0x0131
# Firmware ID (2 bytes Product ID + 2 bytes HW Version ID + 4 bytes FW Version)
FWID=0x3026000101010002
```



**Mesh Client**

**Light Control**

Network:  User:

Group:

Current group:

Provision UUID:  Name:

☐ Static public key

☐ Static OOB data  Identity duration:

**Device configuration**

☒ Friend ☒ GATT Proxy ☒ Relay ☒ Net beacon Retransmit count:  Interval (ms):  Default TTL:  Network transmit count:  Interval (ms):

Publish period:  Master security:  Publish TTL:  Retransmit count:  Interval (ms):

Rename:  New name:

Move Device:  from:  to group:

Use Device:

Control:  On/Off:   Number of iterations:

Level:

Lightness:

Lightness Hue Saturation:

Lightness Color Temperature Delta UV:

Vendor data:   Sensor:

**DFU Image**   App DFU to device:

**Image Info**

**Connected**

11:58:51.862 Net Connect Proxy Connection status provisioner:0 connected:1 over\_gatt:1

11:58:51.873 Filter add addr:ffff num:1 first addr:ffff

Link Status:1

11:58:52.003 Net Connect Filter Status from:3 type:0 list size:1

Firmware OTA start

12:01:32.107 Node Dimmable Light connected continue OTA upgrade

Link Status:0

Figure 3. Mesh Client DFU

## 4 References

- [1] Mesh Profile Specification v1.0
- [2] Mesh Models Specification v1.0
- [3] Mesh Provisioner Database Specification v1.0
- [4] [AN227069 - Getting Started with Bluetooth Mesh](#)

## Document Revision History

Document Title: MeshClient and ClientControlMesh App User Guide

Document Number: 002-26575

Revision	ECN	Issue Date	Description of Change
**	6489357	05/01/2018	Initial release
*A	6556129	04/24/2019	Removed Associated Part Family Updated for BT SDK release
*B	6701354	10/15/2019	Updated for ModusToolbox 2.0
*C	6804876	02/11/2020	Updated for latest WICED Studio and ModusToolbox changes

# Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at [Cypress Locations](#).

## Products

Arm® Cortex® Microcontrollers	<a href="http://cypress.com/arm">cypress.com/arm</a>
Automotive	<a href="http://cypress.com/automotive">cypress.com/automotive</a>
Clocks & Buffers	<a href="http://cypress.com/clocks">cypress.com/clocks</a>
Interface	<a href="http://cypress.com/interface">cypress.com/interface</a>
Internet of Things	<a href="http://cypress.com/iot">cypress.com/iot</a>
Memory	<a href="http://cypress.com/memory">cypress.com/memory</a>
Microcontrollers	<a href="http://cypress.com/mcu">cypress.com/mcu</a>
PSoC	<a href="http://cypress.com/psoc">cypress.com/psoc</a>
Power Management ICs	<a href="http://cypress.com/pmhc">cypress.com/pmhc</a>
Touch Sensing	<a href="http://cypress.com/touch">cypress.com/touch</a>
USB Controllers	<a href="http://cypress.com/usb">cypress.com/usb</a>
Wireless Connectivity	<a href="http://cypress.com/wireless">cypress.com/wireless</a>

## PSoC® Solutions

[PSoC 1](#) | [PSoC 3](#) | [PSoC 4](#) | [PSoC 5LP](#) | [PSoC 6 MCU](#)

## Cypress Developer Community

[Community](#) | [Code Examples](#) | [Projects](#) | [Videos](#) | [Blogs](#) | [Training](#) | [Components](#)

## Technical Support

[cypress.com/support](http://cypress.com/support)

All other trademarks or registered trademarks referenced herein are the property of their respective owners.



Cypress Semiconductor  
198 Champion Court  
San Jose, CA 95134-1709

© Cypress Semiconductor Corporation, 2018-2020. This document is the property of Cypress Semiconductor Corporation and its subsidiaries ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. No computing device can be absolutely secure. Therefore, despite security measures implemented in Cypress hardware or software products, Cypress shall have no liability arising out of any security breach, such as unauthorized access to or use of a Cypress product. CYPRESS DOES NOT REPRESENT, WARRANT, OR GUARANTEE THAT CYPRESS PRODUCTS, OR SYSTEMS CREATED USING CYPRESS PRODUCTS, WILL BE FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (collectively, "Security Breach"). Cypress disclaims any liability relating to any Security Breach, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from any Security Breach. In addition, the products described in these materials may contain design defects or errors known as errata which may cause the product to deviate from published specifications. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. "High-Risk Device" means any device or system whose failure could cause personal injury, death, or property damage. Examples of High-Risk Devices are weapons, nuclear installations, surgical implants, and other medical devices. "Critical Component" means any component of a High-Risk Device whose failure to perform can be reasonably expected to cause, directly or indirectly, the failure of the High-Risk Device, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from any use of a Cypress product as a Critical Component in a High-Risk Device. You shall indemnify and hold Cypress, its directors, officers, employees, agents, affiliates, distributors, and assigns harmless from and against all claims, costs, damages, and expenses, arising out of any claim, including claims for product liability, personal injury or death, or property damage arising from any use of a Cypress product as a Critical Component in a High-Risk Device. Cypress products are not intended or authorized for use as a Critical Component in any High-Risk Device except to the limited extent that (i) Cypress's published data sheet for the product explicitly states Cypress has qualified the product for use in a specific High-Risk Device, or (ii) Cypress has given you advance written authorization to use the product as a Critical Component in the specific High-Risk Device and you have signed a separate indemnification agreement.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit [cypress.com](http://cypress.com). Other names and brands may be claimed as property of their respective owners.