QSG155: Using the Silicon Labs Dynamic Multiprotocol Demonstration Applications in GSDK v2.x

This document shows how to use the dynamic protocol lighting demonstrations. Two demonstrations may be used, one incorporating Zigbee 3.0 and Bluetooth functionality and the other RAIL and Bluetooth functionality. RAIL (Radio Abstraction Interface Layer) is Silicon Labs' intuitive, easily-customizable radio interface layer that supports proprietary or standards-based wireless protocols. In both demonstrations, a dynamic multiprotocol light application can be controlled either from a protocol-specific switch application or from a Bluetooth-enabled smartphone app.

KEY POINTS

- Prerequisite for the demos
- Demo firmware installation
- Instructions for using the demos

1 Prerequisites

The Silicon Labs dynamic multiprotocol demonstrations are designed to illustrate dynamic multiprotocol operation without any need to configure or compile software. This document assumes that you have already obtained your hardware and installed the required software. You will need an EFR32MG Wireless Starter Kit (SLWSTK6000B). For the demo you will use at least two WSTK main boards with demonstration-specific radio boards installed. If you already have the WSTK main boards, you can purchase the required radio boards here.

Simplicity Studio must be installed to configure the boards. Installing Simplicity Studio and the required SDKs are described in their respective quick start guides, listed below.

- Zigbee/Bluetooth Demonstration
  - Radio boards: EFR32MG12 SoC radio boards (SLWRB4161A or SLWRB4162A). **Note:** You will use three boards if you would like to use the sleepy end device demonstration.
  - SDK: Version 6.6.0.0 or higher of the EmberZNet SDK, as described in *QSG106: Getting Started with EmberZNet PRO*.
- RAIL/Bluetooth Demonstration
  - Radio boards: EFR32MG12 2.4 GHz/915 MHz dual band radio boards (SLWRB4164A)
  - SDKs: Version 2.8.0.0 or higher of the Bluetooth SDK (*QSG139: Bluetooth Development with Simplicity Studio*), which contains the light application, and version 2.2.0 or higher of the Flex SDK (*QSG138: Getting Started with the Silicon Labs Flex SDK for the Wireless Gecko (EFR32™) Portfolio*), which contains the RAIL switch application.

  Note: This document describes using the precompiled images supplied with the SDKs. See section 6 Next Steps for information on continuing with customizing the source code for these demonstration images.

You should also download the **Wireless Gecko by Silicon Labs** smartphone application for Android or iOS. Note that there are several Silicon Labs apps - be sure to download **Wireless Gecko**, not Blue Gecko. The mobile application is available here:

Google Play: https://play.google.com/store/apps/details?id=com.siliconlabs.wirelessgecko

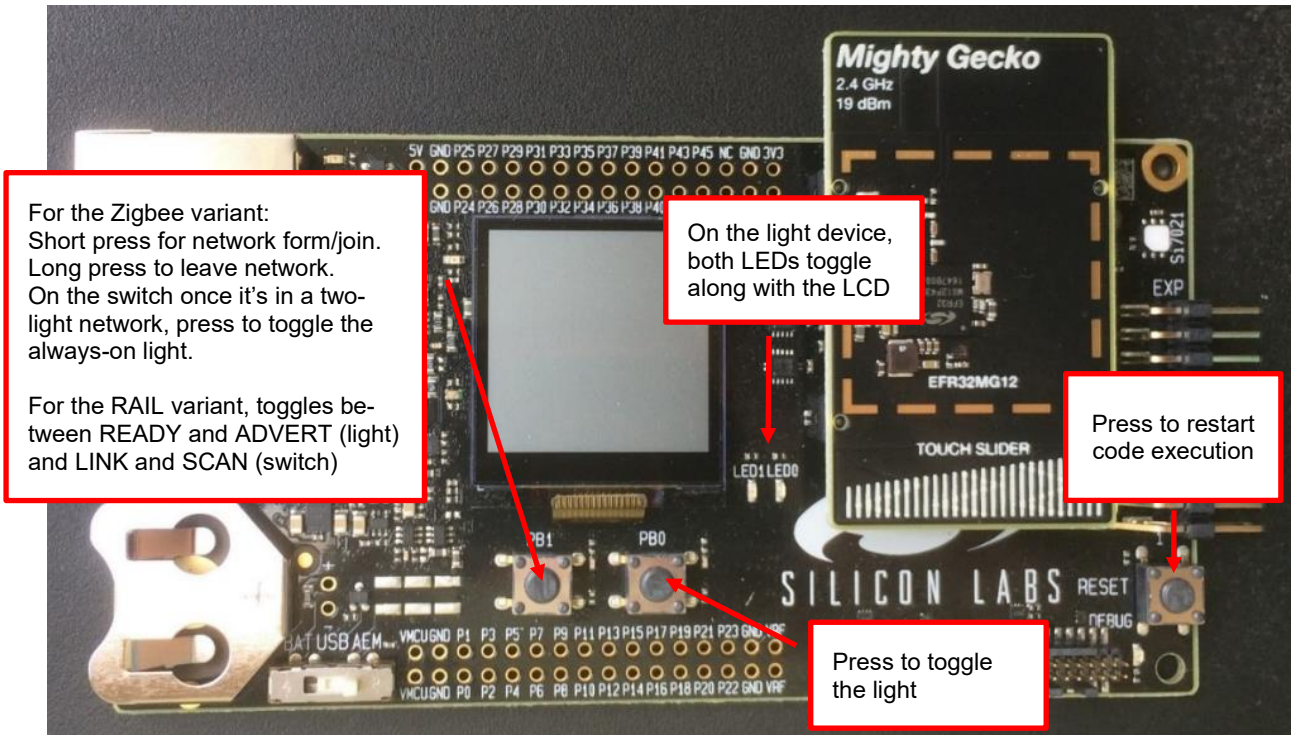
App Store: https://itunes.apple.com/us/app/wireless-gecko/id1315784335?mt=8

Note: The minimum requirement for the smartphone is Android 6 (API23). iPhones should run the most recent version of IOS.

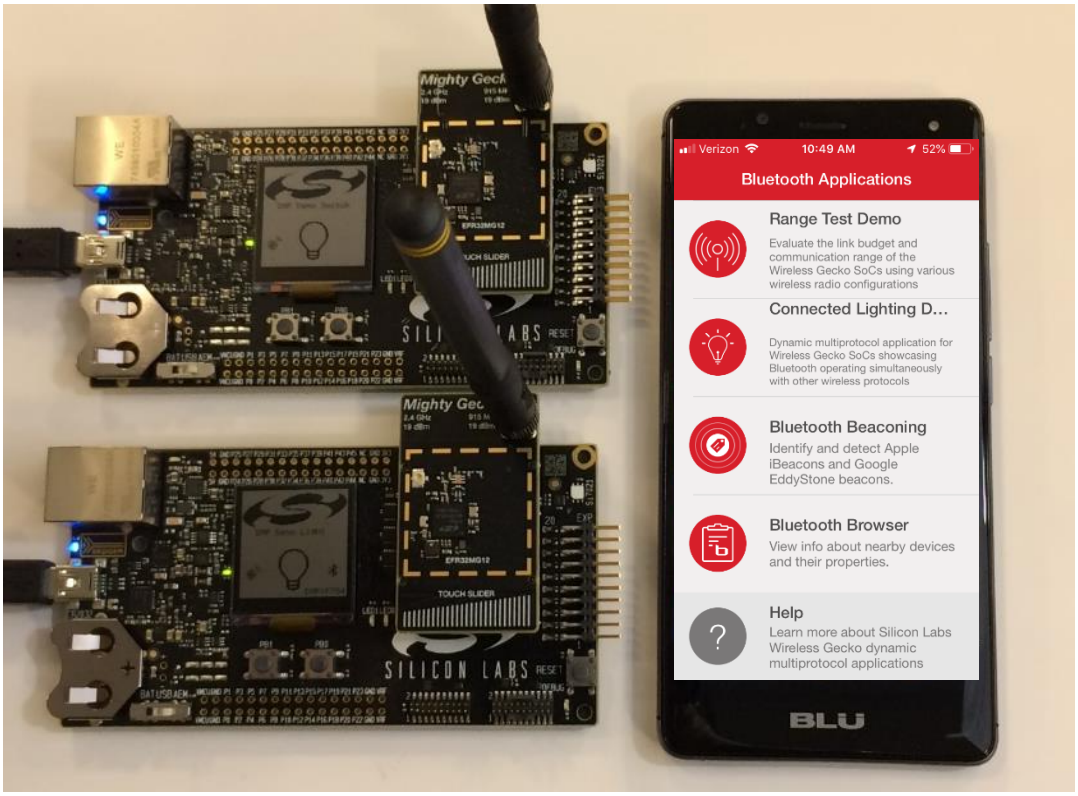
Use the support portal to contact Customer Support for any questions you might have about the demonstrations. You can access the Silicon Labs support portal at https://www.silabs.com/support through Simplicity Studio Resources. Click the “Email-Support” link and log in with your self-registered credentials.

2 Demonstration Devices

The following features on the WSTK are used in the demos.

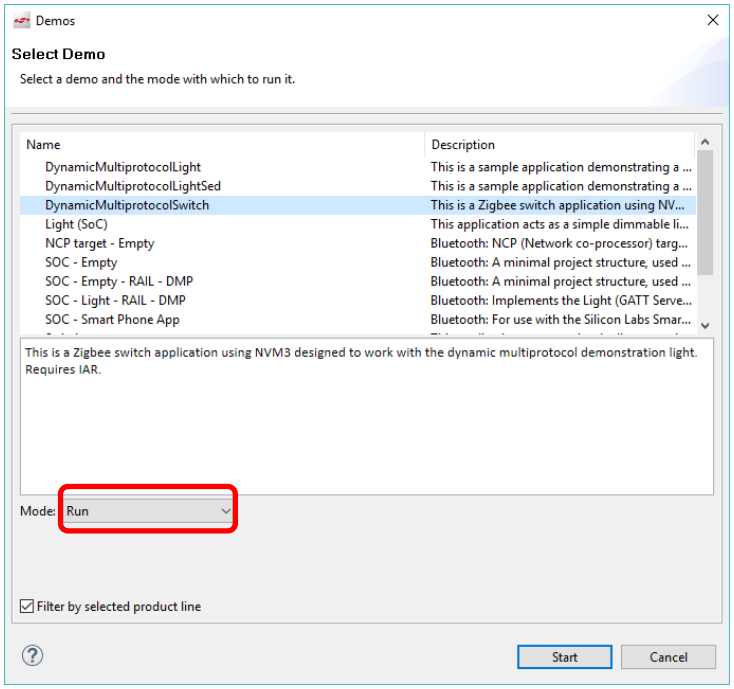


The following is a typical demo setup with two WSTKs and a smart phone (the sub-Gigahertz setup for RAIL/Bluetooth is shown).



3 Install the Demonstration Applications

Open Simplicity Studio. In general, to load a demo, click a device in the Devices pane, then click the demo. The Select Demo dialog opens. In the **Mode** drop-down, select **Run**. Click **Start**. The demo software downloads automatically to the selected device.

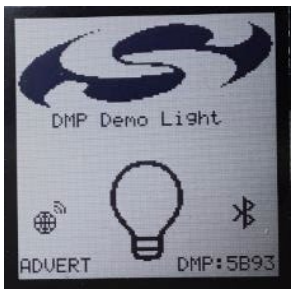


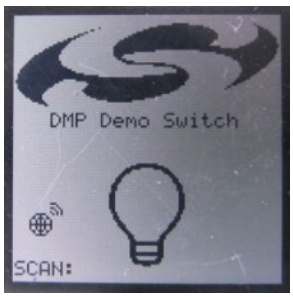
3.1 RAIL/Bluetooth

To run the RAIL/Bluetooth demo, in the Flex SDK, load one device with **RAIL: Switch**. Then go to the Bluetooth SDK and load the other device with **SOC - Light -RAIL - DMP**.

Note: The **SoC Demo Connect Light** application in the Flex SDK is used with the Connect/Bluetooth DMP demo, not the RAIL/Bluetooth demo.

After installation is complete, the LCDs show the application “Light Bulb” display. The RAIL Switch application briefly scans for a nearby switch and then shows the short ID of the switch with the strongest signal.

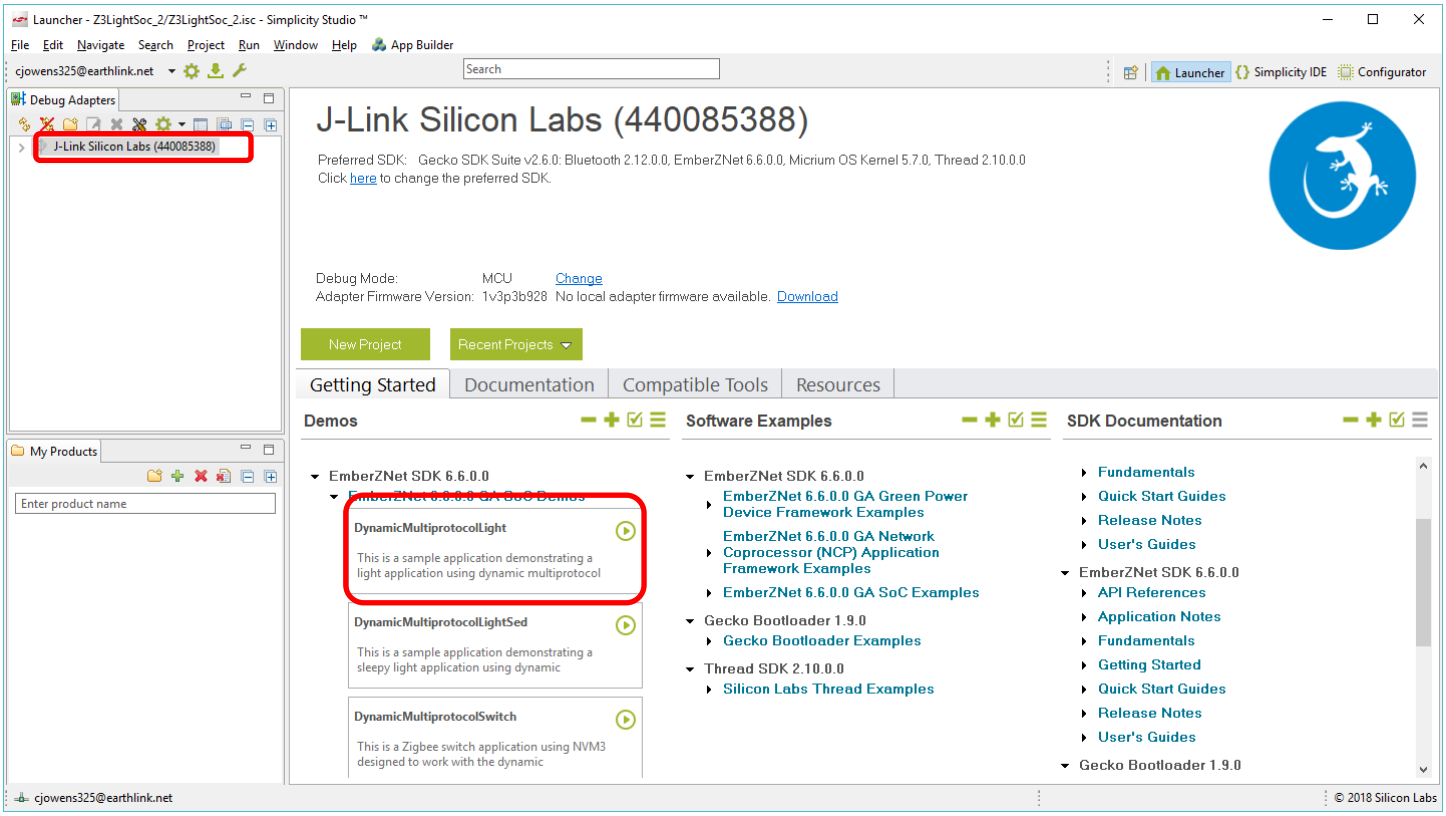




3.2 Zigbee/Bluetooth

To run the Zigbee/Bluetooth demo with both a full function device and a sleepy end device, load one device with **DynamicMultiprotocolLight** (the full function device demo), one device with **DynamicMultiprotocolSwitch,** and one device with **DynamicMultiprotocolLightSed** (the sleepy end device demo)

The **DynamicMultiprotocolLight** demo acts as a Zigbee coordinator and is required to form the network.



A sleepy end device (SED) powers down its radio when idle, and thus extends battery life. However, it must poll its parent to receive incoming messages and acknowledgments; no data is sent to the SED until the end device requests it. The full-function device (FFD) remains powered during operation and acts as a coordinator that forms and manages the network. The SED polls the FFD every 5 seconds in the demo application to receive any messages that were sent to it by the Switch or Bluetooth App while it was asleep.

When the Zigbee/Bluetooth demo applications download, they first show a help menu on the LCD, and then show the main display. The help menus are also displayed for approximately 10 seconds if you restart code execution by pressing the Reset button or power cycling the device. Note that resetting the application breaks the Bluetooth connection between devices, if one has been formed, and interrupts communication on the device network, if one has been formed. The Help menus are:

After installation is complete, the LCDs show the application “Light Bulb” display.

4 Use the Zigbee/Bluetooth Demonstration

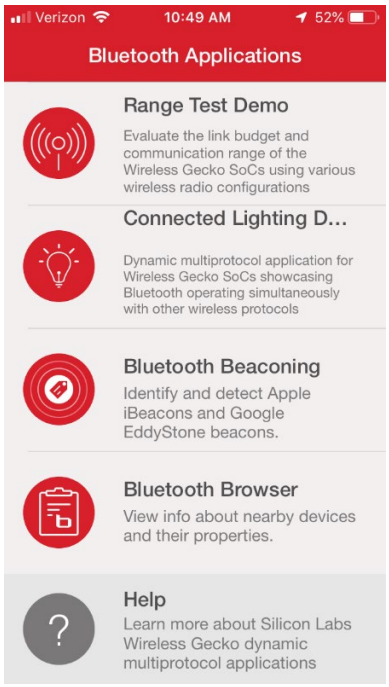
In this section, a WSTK loaded with the Full-Function **DynamicMultiprotocolLight** is referred to as the always-on device. The WSTK loaded with the Sleepy End Device **DynamicMultiprotocolLightSed** is referred to as the sleepy device. The following sections illustrate the demo options using:

- Either the always-on or sleepy devices with the Bluetooth Smartphone app alone
- The always-on device with the switch
- The sleepy device demo with the always-on device and the switch

4.1 With the Bluetooth Smartphone App Alone

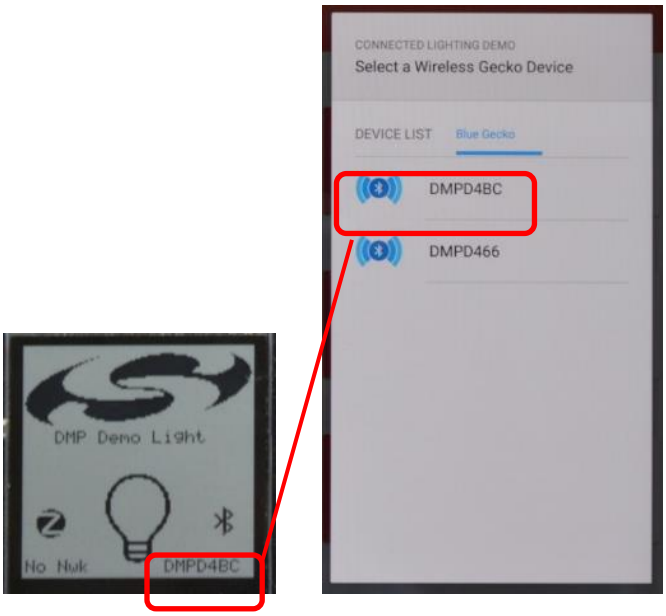
The **DynamicMultiprotocolLight** and the **DynamicMultiprotocolLightSed** applications behave identically when used with the Bluetooth Smartphone app alone. For simplicity only the **DynamicMultiprotocolLight** images have been shown.

When both devices are ready, open the app and tap **Connected Lighting Demo**.

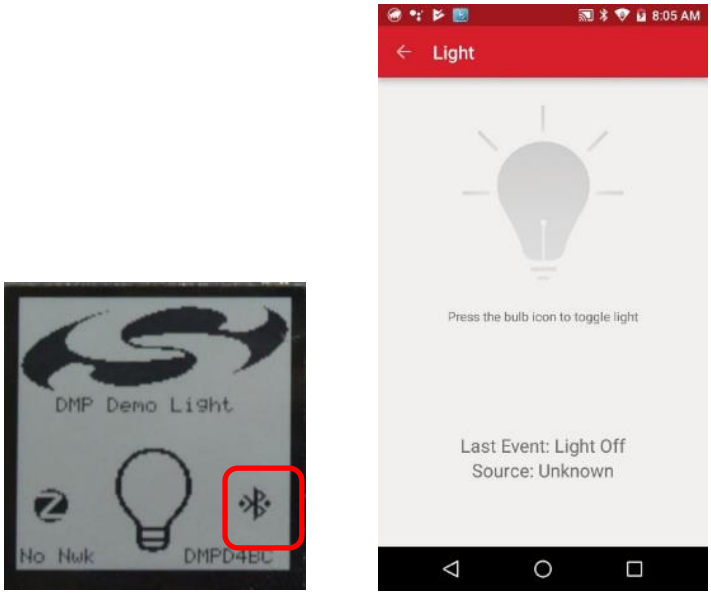


If you have a sleepy device and an always-on device in the demo you will see more than one device, with different IDs. Tap the one with the ID matching the ID on the device you wish to control.

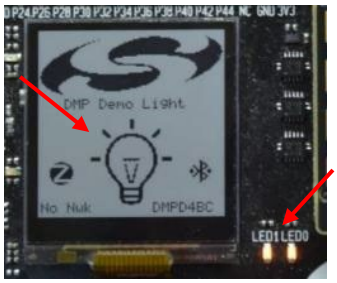
Note: The app can only connect to one device at a time. To control both the always-on and the sleepy devices you would need to have two smartphones, with each app connected to a different device.

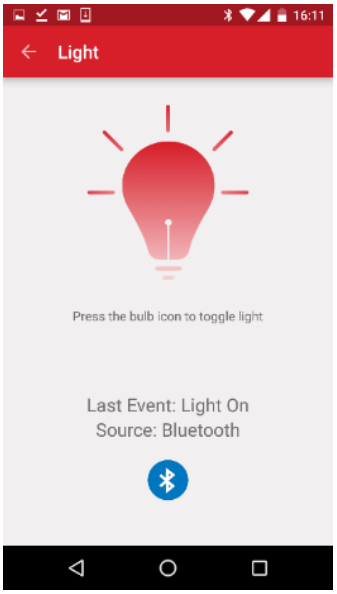


The app display changes to the Light Bulb, and the Bluetooth icon on the light's LCD changes as well, indicating a connection.



Tap the bulb icon on the smartphone app to toggle the light. The app display, the LCD display, and the LEDs all turn on. The app shows **Last Event: Light On** and **Source: Bluetooth.**





Tap again to turn the light off. The app shows **Last Event: Light Off** and **Source: Bluetooth**.

As you toggle the light with the app, an arrow is briefly displayed on the light LCD to indicate that the source of the command is Bluetooth.

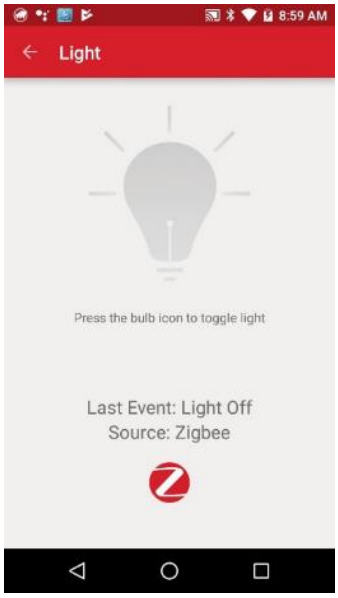
On the light device, press PB0 to toggle the light. The app display now shows **Source: Button.**

4.2 With the Switch and the Always-On Device

In order to operate the always-on device from the Zigbee switch application, you need to form a Zigbee network. Press PB1 on the always-on device. The display changes from **No Nwk** to **Forming** and then to a flashing PAN ID. While the PAN ID is flashing the application is in permit join mode. Press PB1 on the switch. The display changes to **Joining** and, after a brief delay, to the network's PAN ID.

Note: If you are in a busy environment, the switch might join to another network. Make sure the PAN IDs on both the always-on device and the switch are the same. If not, press and hold PB1 on the switch for more than 5 seconds to take it off the network. Press PB1 on the always-on device to put it back into permit join mode. Once the PAN ID is flashing, press PB1 on the switch.

Once the switch is connected, press PB0 on the switch to toggle the always-on device. The app shows **Source: Zigbee**.



Again, a briefly-displayed arrow shows the source of the command on the always-on device LCD.

Now if you press the bulb icon on the app, both the always-on device and the switch displays change.

4.3 With the Switch, the Always-On Device, and the Sleepy Device

Because the sleepy device serves only as a light, the always-on device running the **DynamicMultiprotocolLight** application is also required to form the network and to allow the sleepy device to join it.

If you have not already formed a Zigbee network as described in the previous section, you need to do so before adding the sleepy device to it. Press PB1 on the always-on device. The display changes from **No Nwk** to **Forming** and then to a flashing PAN ID. While the PAN ID is flashing the application is in permit join mode. Press PB1 on the switch. The display changes to **Joining** and, after a brief delay, to the network's PAN ID.

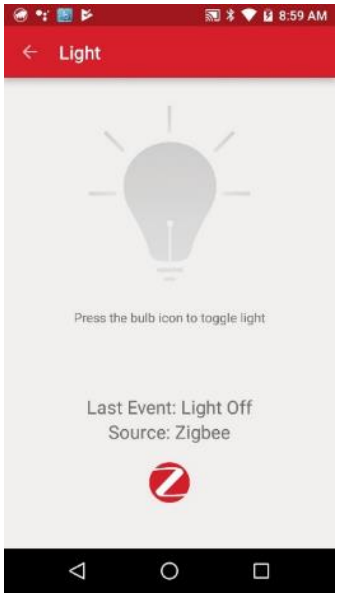
To add the sleepy device to the Zigbee network that has just been created, ensure that the sleepy device is displaying **No Nwk.** If not, hold PB1 on the sleepy device for more than 3 seconds to leave the network. Press PB1 on the always-on device again. The PAN ID on the display will begin to flash, indicating that it is open for other devices to join. Now press PB1 on the sleepy device. The display will change to **Joining** and, after a brief delay, to the network's PAN ID.

Note: If you are in a busy environment, the joining device (switch or sleepy device) might join to another network. Make sure the PAN IDs on both the always-on device and the joining device are the same. If not, press and hold PB1 on the joining device for more than 5 seconds to take it off the incorrect network. Press PB1 on the always-on device to put it back into permit join mode. Once the PAN ID is flashing, press PB1 on the joining device, then verify that the PAN IDs match.

Once all three devices are on the same network, PB0 on the switch controls the sleepy device, and PB1 controls the always-on device.

When the app is powered up with a sleepy device and an always-on device present it will show two Bluetooth devices. The app can be connected to either Bluetooth device but not both. Connect to the sleepy device as described in section 4.1 With the Bluetooth Smartphone App Alone.

Press PB0 on the switch to toggle the sleepy device. The app shows **Source: Zigbee**.



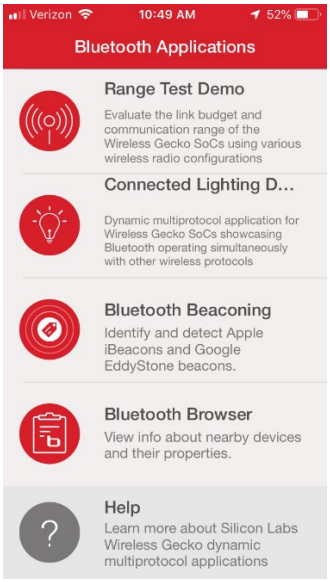
Again, a briefly-displayed arrow shows the source of the command on the Light LCD.

Now if you press the bulb icon on the app, both the light and the switch displays change.

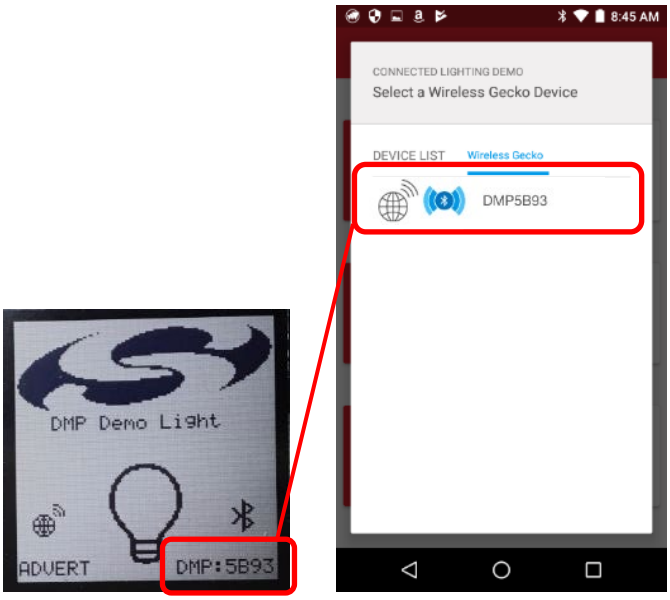
5 Use the RAIL/Bluetooth Demonstration

5.1 With the Bluetooth Smartphone App Alone

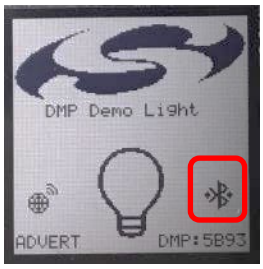
When both devices are ready, open the app and tap **Connected Lighting Demo**.

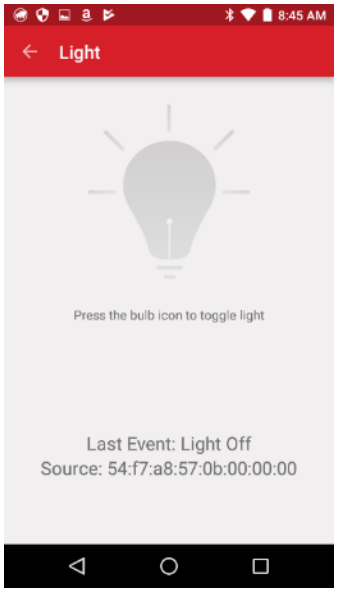


Tap the Bluetooth light device. If you see more than one, tap the one with the matching ID.

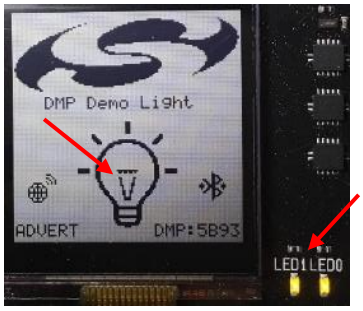


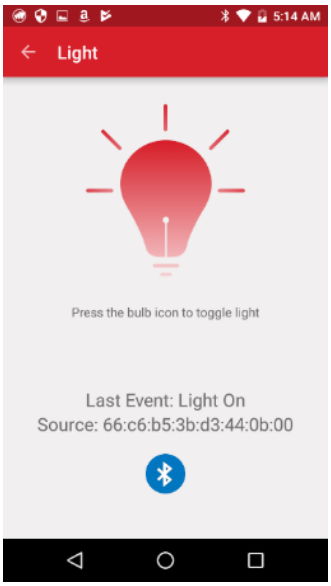
The app display changes to the Light Bulb, and the Bluetooth icon on the light's LCD changes as well, indicating a connection.





Tap the bulb icon on the smartphone app to toggle the light. The app display, the LCD display, and the LEDs all turn on. The app shows **Last Event: Light On. Source** is the MAC address of the device sending the command, in this case the smartphone.





Tap again to turn the light off. The app shows **Last Event: Light Off** and **Source:** Smartphone MAC address. As you toggle the light with the app, an arrow is briefly displayed on the light LCD to indicate that the source of the command is Bluetooth.

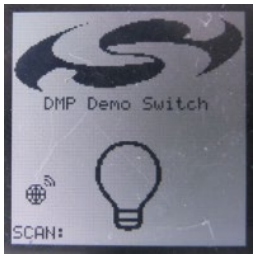
On the light device, press PB0 to toggle the light. The app display now shows the source as the MAC address of the light device.

Note that the light demo has two modes, Advertise (ADVERT) and READY. These are for operation by the switch device, as described next. The smartphone app toggles the light regardless of the mode.

5.2 With the Switch

In order to operate the light from the RAIL Switch application, you first need to link the two devices.

The RAIL switch starts in SCAN mode to look for lights in the vicinity. The light starts in ADVERTISE mode and broadcasts its address to nearby RAIL Switches.



Once a RAIL switch receives a light's advertisement packet it displays its short ID in the bottom left corner. If more lights are advertising at the same time, the switch displays the ID of the Light with the strongest signal:

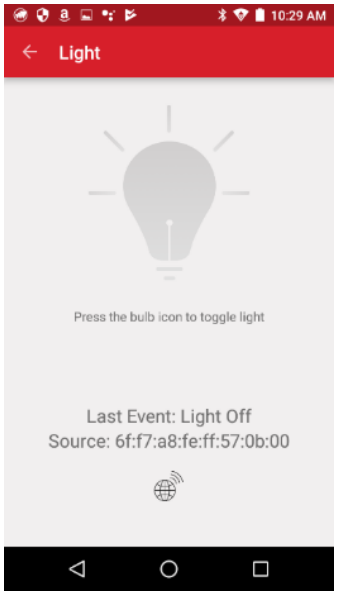
Press PB1 on the RAIL switch to store the light's ID and put the device into LINK mode:

Finally, press PB1 on the Light to place it to READY mode, in which it sends out the status of the light periodically and accepts toggle commands from the linked Switch:

You should now be able to toggle the light using the linked switch.

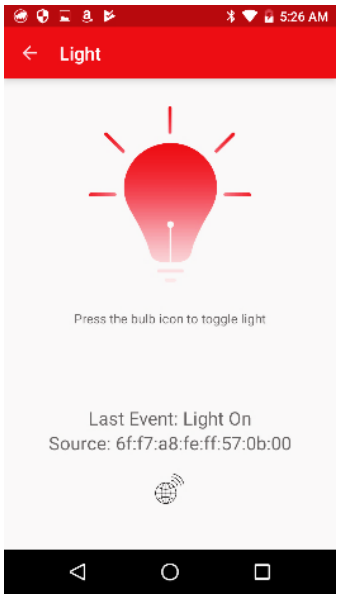
Note that more than one switch can be linked to a single light, but one switch cannot control more than one light.

The app shows the source as the MAC address of the switch.



Again, a briefly-displayed arrow shows the source of the command on the Light LCD.

Now if you press the bulb icon on the app, both the light and the switch displays change.



6 Next Steps

*UG305: Dynamic Multiprotocol User's Guid*e contains details on the functionality underlying dynamic multiprotocol applications. It also describes how the Radio Scheduler, a key component of the dynamic multiprotocol solution, works. The following figure shows the current documentation set for dynamic multiprotocol.

The source code for the smartphone application is also available. Contact Technical Support if you are interested.

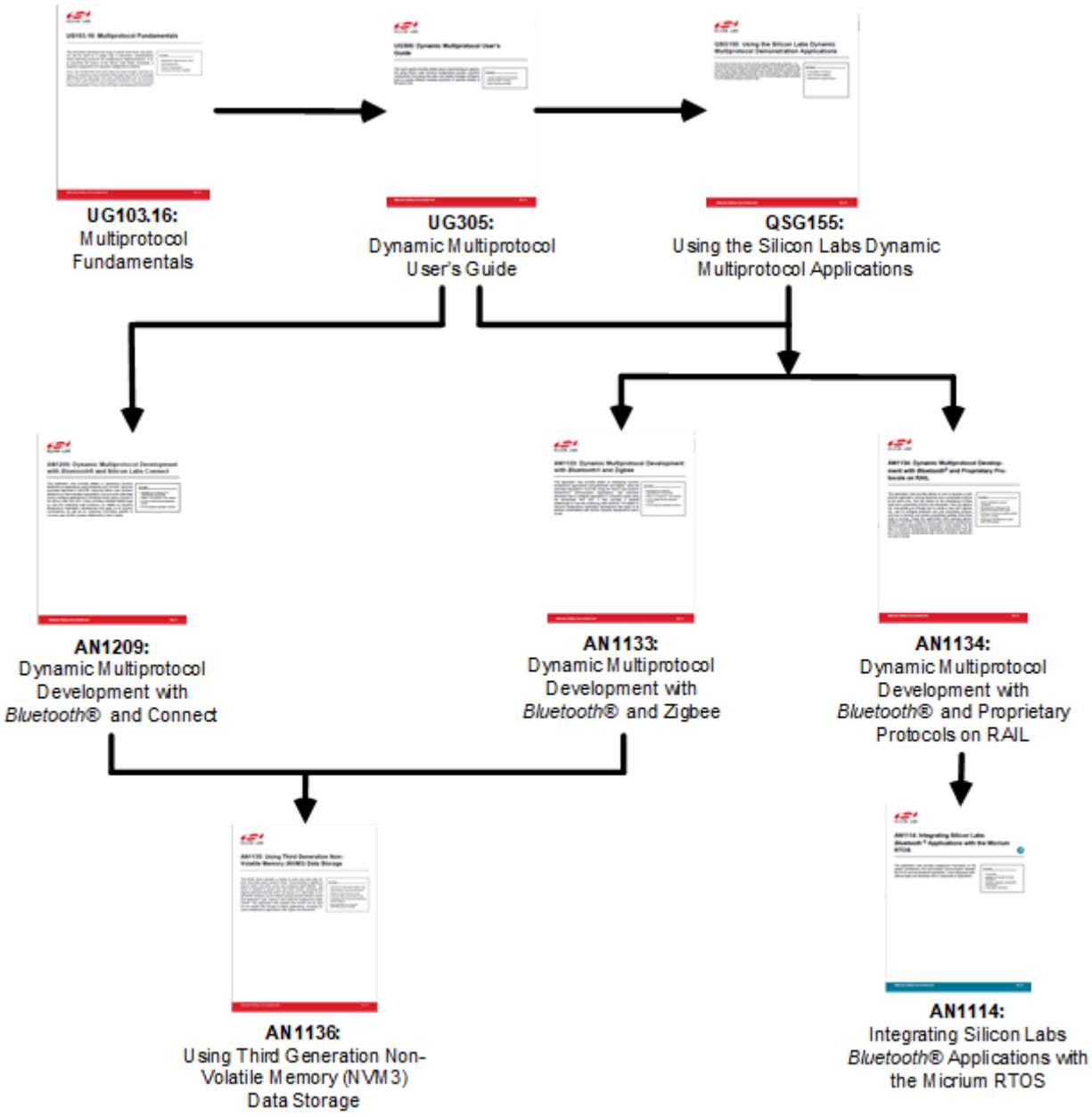


Figure 6-1. Dynamic Multiprotocol Documentation Set

6.1 Zigbee/Bluetooth

The EmberZNet SDK contains the code used to produce the demo images. *AN1133: Dynamic Multiprotocol Development with Bluetooth and Zigbee* contains a summary procedure for these examples, and *QSG106: Getting Started with EmberZNet* describes in detail how to generate, build, and flash example code. AN1133 also describes how to change configuration settings to make an EmberZNet application into a dynamic multiprotocol application, and contains details on the functionality underlying this demo.

Note that you must download the following in order to fully explore the examples:

- The EmberZNet SDK
  - Version 6.0.0.0 or higher for the switch and always-on light.
  - Version 6.4.0.0 or higher on all three devices for the sleepy light.
- The Bluetooth SDK
  - Version 2.6.0.0 or higher for the switch and sleepy light,
  - Version 2.10.0.0 or higher for the switch, always-on light and sleepy light
- Micrium OS kernel (as of Bluetooth SDK version 2.9.0 installed with the SDK)

You must also install and use IAR-EWARM as your compiler. The version of IAR-EWARM that you use must be compatible with your SDK version. See the release notes for that version of the SDK for compatible compiler version information.

The Zigbee dynamic multiprotocol solution is currently only supported for SoC architectures (not NCP). Support for NCP is not yet available. Please contact Silicon Labs Sales for more information on our multiprotocol software roadmap.

6.2 RAIL/Bluetooth

The Flex SDK (version 2.2.0.0 or higher) contains the code used to produce the Switch demo image. The Bluetooth SDK (version 2.8.0.0 or higher) contains the code used to produce the RAIL/Bluetooth dynamic multiprotocol light example. *AN1134: Dynamic Multiprotocol Development with Bluetooth and Proprietary Protocols on RAIL* contains summary procedures for these examples as well as instructions on starting with your own proprietary protocol-based application. *QSG138: Getting Started with the Silicon Labs Flex SDK for the Wireless Gecko (EFR32™) Portfolio* and *QSG139: Bluetooth Development with Simplicity Studio*), respectively, describe in detail how to make and flash example applications.

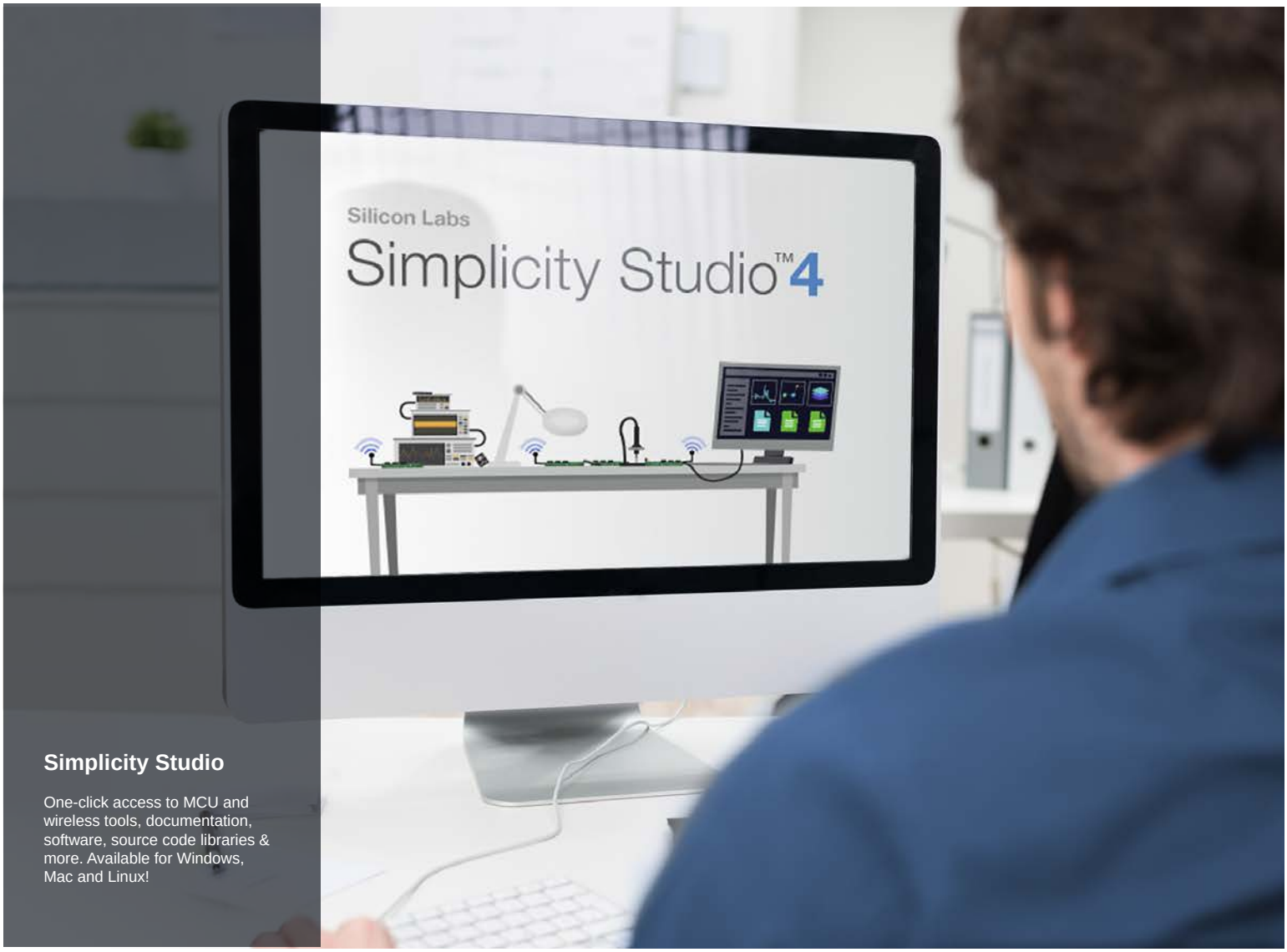
Note that you must download not only the Flex and Bluetooth SDKs, but also the Micrium OS kernel in order to fully explore the examples. As of Bluetooth SDK version 2.9.0, the Micrium OS kernel is installed with the SDK.

6.3 Connect/Bluetooth

Connect/Bluetooth dynamic multiprotocol implementations were supported beginning with Silicon Labs Flex SDK version 2.6.0.0. See *AN1209: Dynamic Multiprotocol Development with Bluetooth and Connect* for details. The SDK includes two applications that show a use case of a dynamic multiprotocol application using the Silicon Labs Connect stack.

- **Connect (SoC): Demo Connect Light**: This is a non-DMP application running the Connect stack only. When in "factory reset" state upon power up (or optionally on a user action), scans all allowed channels for the lowest energy and forms a Connect point-to-point network. Upon user action, it opens its network for another device to join.
- **Connect (SoC): Demo DMP Connect Switch**: On a user action (for example a button press) starts scanning on all allowed channels to find the above open network and connect.

A **Connect (SoC): Empty Example - DMP** sample application is also included in the Flex SDK. This is a minimal project structure that can be used as a starting point for custom applications that will run both Connect and Bluetooth LE protocols.



Simplicity Studio

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

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

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 products 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 Labs product in such unauthorized applications.

Trademark Information
Silicon Laboratories Inc.® , Silicon Laboratories®, Silicon Labs®, SiLabs® and the Silicon Labs logo®, Bluegiga®, Bluegiga Logo®, ClockBuilder®, CMEMS®, DSPLL®, EFM®, EFM32®, EFR, Ember®, Energy Micro, Energy Micro logo and combinations thereof, "the world's most energy friendly microcontrollers", Ember®, EZLink®, EZRadio®, EZRadioPRO®, Gecko®, Gecko OS, Gecko OS Studio, ISOmodem®, Precision32®, ProSLIC®, Simplicity Studio®, SiPHY®, Telegesis, the Telegesis Logo®, USBXpress® , Zentri, the Zentri logo and Zentri DMS, Z-Wave®, 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 LABS

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

http://www.silabs.com