

# QSG106: Getting Started with EmberZNet PRO

---

This quick start guide provides basic information on configuring, building, and installing applications for the Mighty Gecko (EFR32MG) family of SoCs using the EmberZNet PRO stack v. 6.6.0 and higher and Simplicity Studio version 4.

This guide is designed for developers who are new to EmberZNet PRO and the Silicon Labs development hardware. It provides instructions to get started using the example applications provided with the EmberZNet PRO stack.

## KEY FEATURES

---

- Product overview
- Setting up your development environment
- Installing Simplicity Studio and EmberZNet PRO
- Creating an example application network
- Using the Network Analyzer

## 1 Product Overview

Before following the procedures in this guide you must have

- Purchased your Mighty Gecko (EFR32MG) Mesh Networking Kit (see <http://www.silabs.com/products/wireless/mesh-networking/zigbee/Pages/zigbee.aspx>):
- Downloaded the required software components, as described below. A card included in your development hardware kit contains a link to a Getting Started page, which will direct you to links for the Silicon Labs software products.

### 1.1 Software Components

See the stack release notes for version restrictions and compatibility constraints for the stack and these components. To develop EmberZNet PRO applications, you will need the following. Installation instructions are provided in section [2.3 Install Simplicity Studio and the EmberZNet PRO Stack](#).

- The Simplicity Studio version 4 development environment, which incorporates AppBuilder. If you do not have version 4, please connect to <http://www.silabs.com/products/mcu/Pages/simplicity-studio-v4.aspx> to download it. AppBuilder is an interactive GUI tool that allows you to configure a body of Silicon Labs-supplied code to implement applications. Online help for AppBuilder and other Simplicity Studio modules is provided.
- The EmberZNet PRO stack, an advanced implementation of a Zigbee stack, installed through Simplicity Studio. The stack API is documented in an online API reference. The stack is delivered as a collection of libraries that you can link to your applications. A description of each library is provided in the development environment. The release notes contain details on the folders installed along with their contents.
- Simplicity Commander, installed along with Simplicity Studio. A GUI with limited functionality can be accessed through Simplicity Studio's Tools menu. Most functions are accessible through a CLI invoked by opening a command prompt in the Simplicity Commander directory (\SiliconLabs\SimplicityStudio\v4\developer\adapter\_packs\commander). See [UG162: Simplicity Commander Reference Guide](#) for more information.
- Compiler toolchain:
  - GCC (The GNU Compiler Collection) is provided with Simplicity Studio. GCC is used in this document. However, you must use IAR to compile the following:
    - Any application for the EM3x
    - Any of the dynamic protocol examples
    - The Z3 Switch with Micrium example

**Note:** Application images created with GCC are larger than those created with IAR. If you use GCC to compile the example applications in this SDK, you must use a part with at least 512 kB of flash.

- IAR Embedded Workbench for ARM (IAR-EWARM) (must be used with the Dynamic Multiprotocol and Z3Switch with Micrium examples, as well as any application for the EM3x).

**Note:** See the SDK Release Notes for the IAR version supported by this version of the EmberZNet SDK. Download the supported version from the Silicon Labs Support Portal, as described at the end of section [2.3 Install Simplicity Studio and the EmberZNet PRO Stack](#). Refer to the "QuickStart Installation Information" section of the IAR installer for additional information about the installation process and how to configure your license. Once IAR-EWARM is installed, the next time Simplicity Studio starts it will automatically detect and configure the IDE to use IAR-EWARM.

While Simplicity Studio and Simplicity Commander can be run on a Mac OS or Linux machine, these instructions assume you are working with a Microsoft Windows-based PC. If you are using a non-Windows system, IAR-EWARM must be run via WINE or some other form of emulator or virtual machine.

### 1.2 Support

You can access the Silicon Labs support portal at <https://www.silabs.com/support> through Simplicity Studio's Resources tab, as described in section [3.4 Accessing Documentation and Other Resources](#). Use the support portal to contact Customer Support for any questions you might have during the development process.

### 1.3 Documentation

Stack documentation is accessed through Simplicity Studio, as described in section [3.4 Accessing Documentation and Other Resources](#). Simplicity Studio also provides links to hardware documentation and other application notes. See the release notes for further details about your EmberZNet PRO software.

#### 1.4 Gecko Platform

The Gecko Platform is a set of drivers and other lower layer features that interact directly with Silicon Labs chips and modules. Gecko Platform components include EMLIB, EMDRV, RAIL Library, NVM3, and mbedTLS. Application developers using EmberZNet plugins, Hardware Configurator interface, or APIs may not need to interact directly with the Gecko Platform as the code does that for you. For more information about Gecko Platform, see release notes that can be found in Simplicity Studio's Launcher Perspective, under **SDK Documentation > EmberZNet SDK 6.6.n.n > Release Notes**.

## 2 Setting Up Your Development Environment

### 2.1 Connect the WSTK

Connect your WSTK, with radio board mounted, to your PC using a USB cable. By having it connected when Simplicity Studio installs, Simplicity Studio will automatically obtain the relevant additional resources it needs.

**Note:** For best performance in Simplicity Studio, be sure that the power switch on your WSTK is in the Advanced Energy Monitoring or "AEM" position as shown in the following figure.

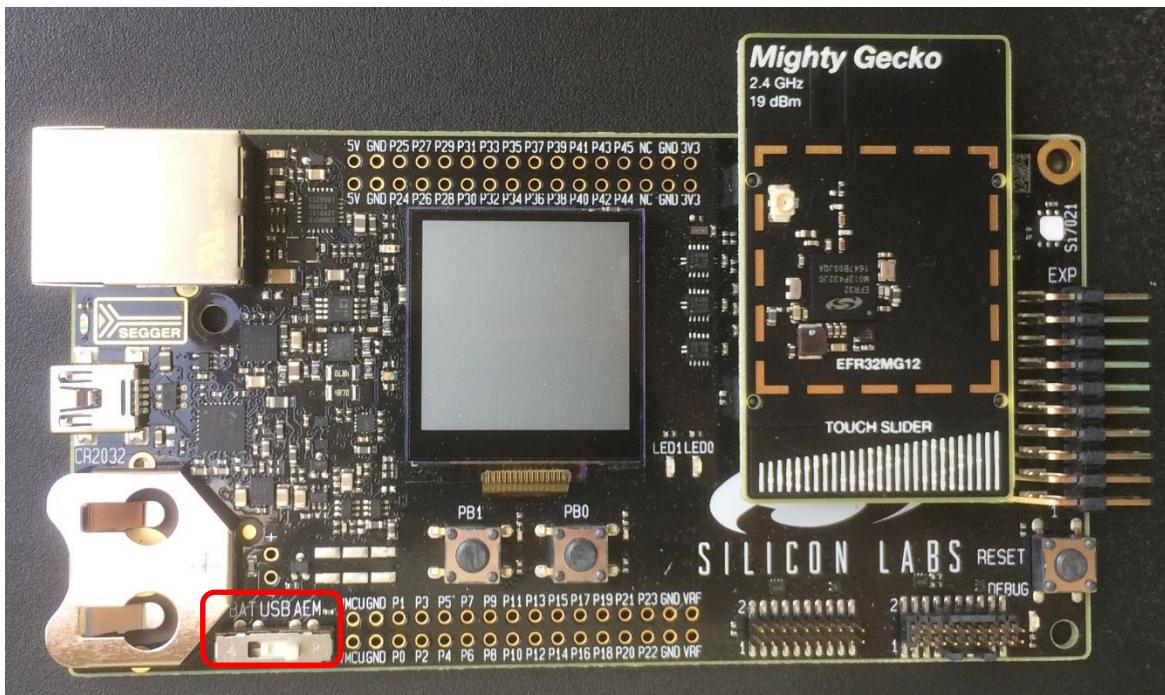


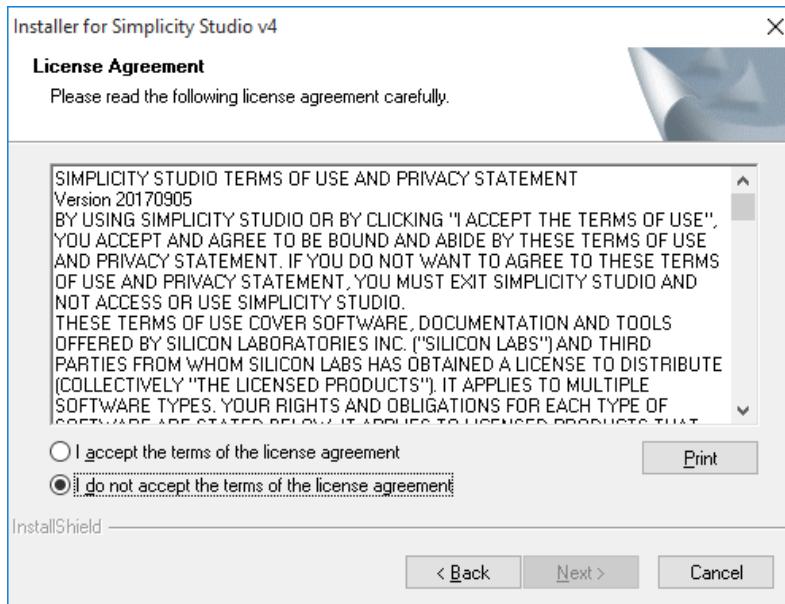
Figure 1. EFR32MG12 on a WSTK

### 2.2 Register your Development Kit

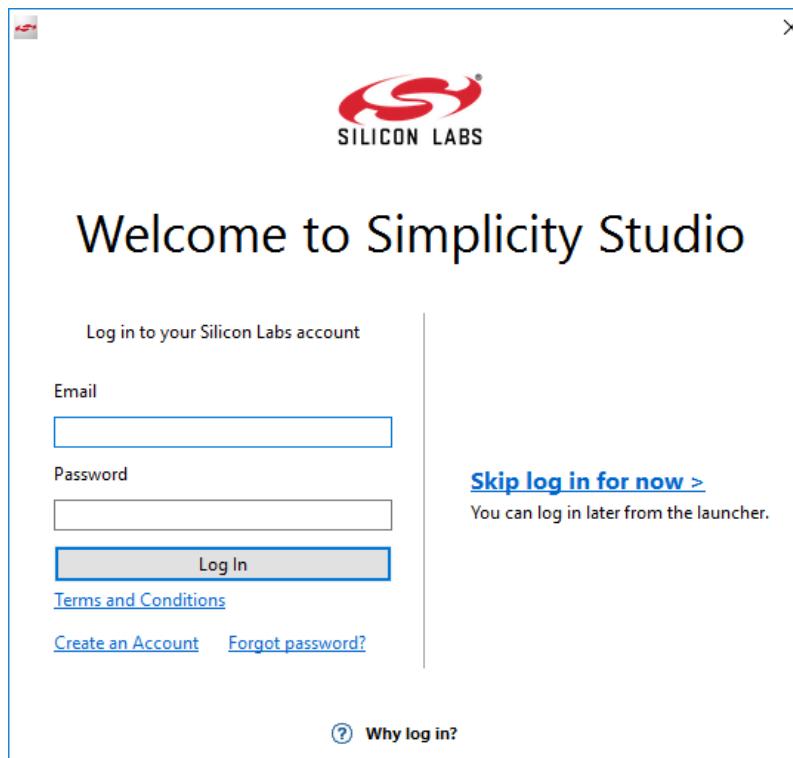
Before you install Simplicity Studio, you need to create an account on the support portal. Be sure to record your account username and password as you will use it to log in to Simplicity Studio. In order to install the EmberZNet PRO stack from Simplicity Studio, you must also register your kit on <https://siliconlabs.force.com/KitRegistration>, using your Mighty Gecko Kit serial number. You can register your kit through Simplicity Studio during installation if you prefer.

### 2.3 Install Simplicity Studio and the EmberZNet PRO Stack

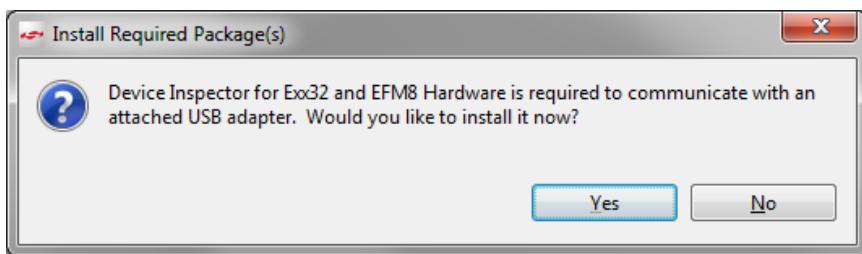
1. Run the Simplicity Studio installation application.
2. When the Simplicity Studio installer first launches, it presents a License Agreement dialog. Accept the terms of the agreement and click **Next >**.



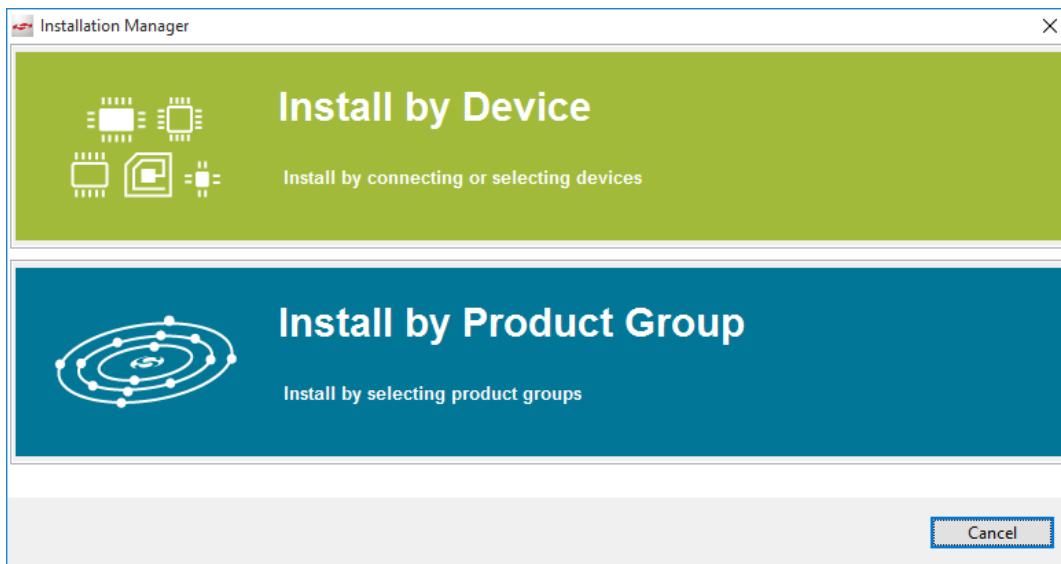
3. Choose a destination location, click **Next >** and then click **Install**.
4. When the application launches, you are invited to log in. Log in using your support account username and password. Although you can skip log in here, you must be logged in and have registered your development kit to download a protected stack such as EmberZNet PRO.



5. After login, Simplicity Studio adds software information. Once initial software installation is complete, Simplicity Studio checks for connected hardware. If you have the WSTK connected by USB cable, Simplicity Studio will detect the USB cable and prompt you to download a Device Inspector. Click **Yes**.

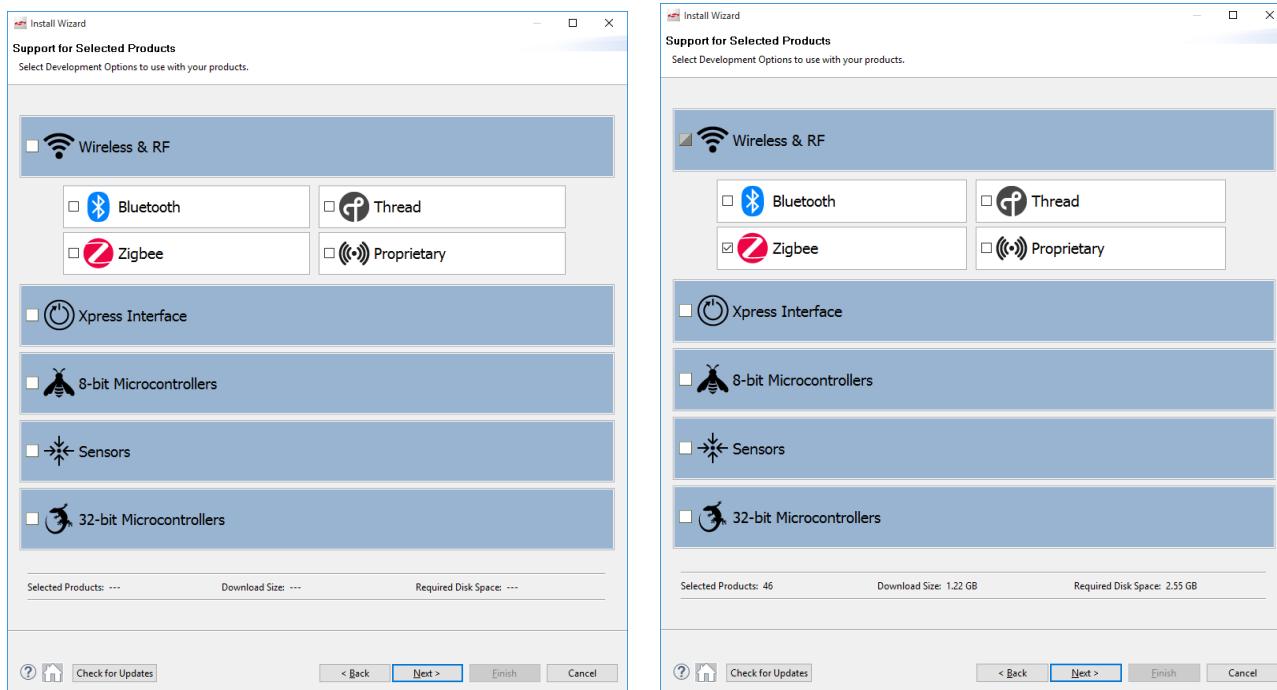


6. After some additional items are installed, you are offered the option of installing by device (step 8) or installing by product group (step 7). **Install by Product Group** will give you a more targeted set of installation options.

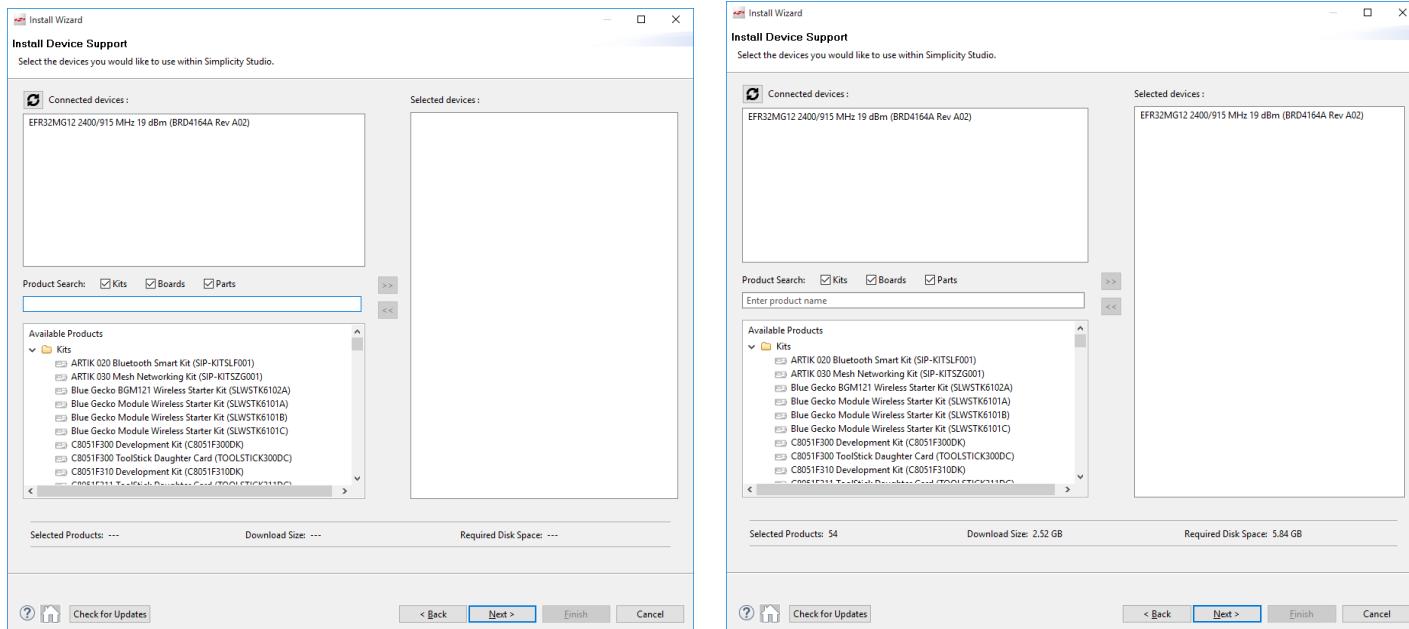


Throughout these procedures at any time you can click Home ( ) to return to this dialog.

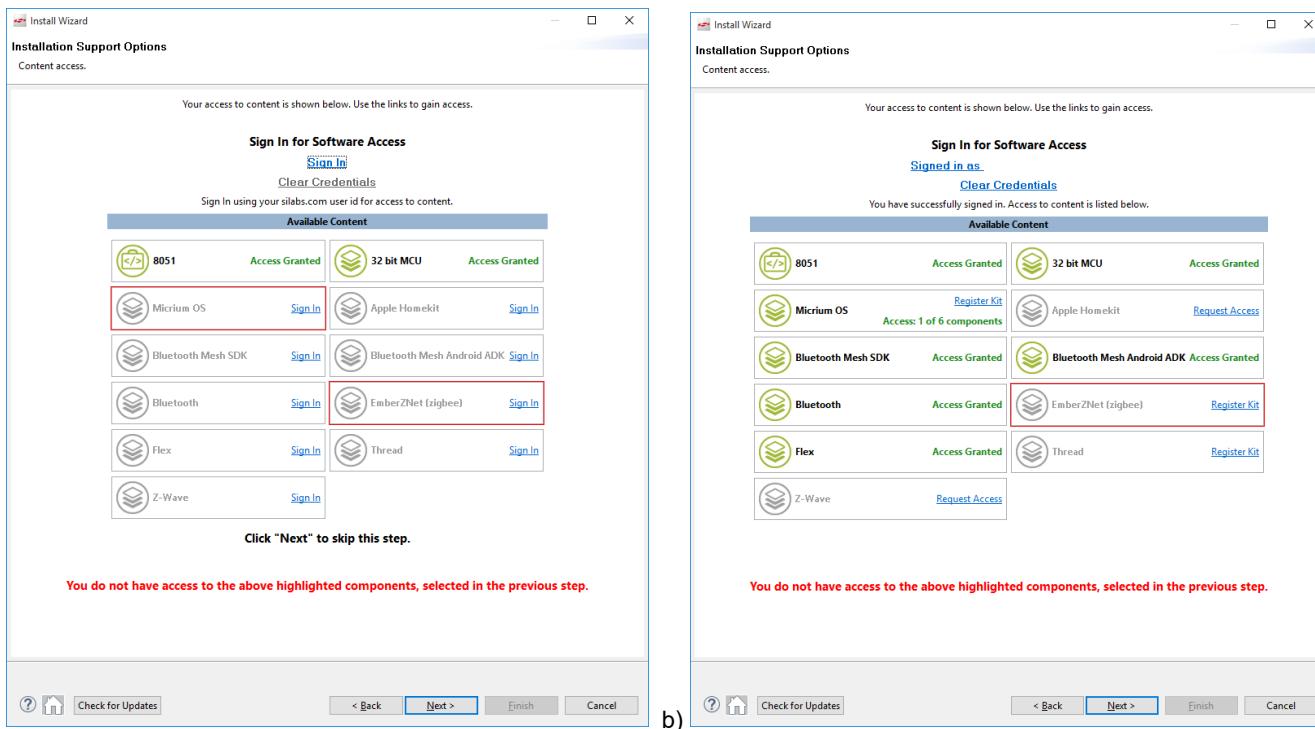
7. If you click **Install by Product Group** you are offered a list of product groups. Click the SDKs you want to install, or click **Wireless & RF** to check all. Note that if you plan to work on Dynamic Multiprotocol examples you must install both the Zigbee and Bluetooth product groups. Click **Next** and go to step 9.



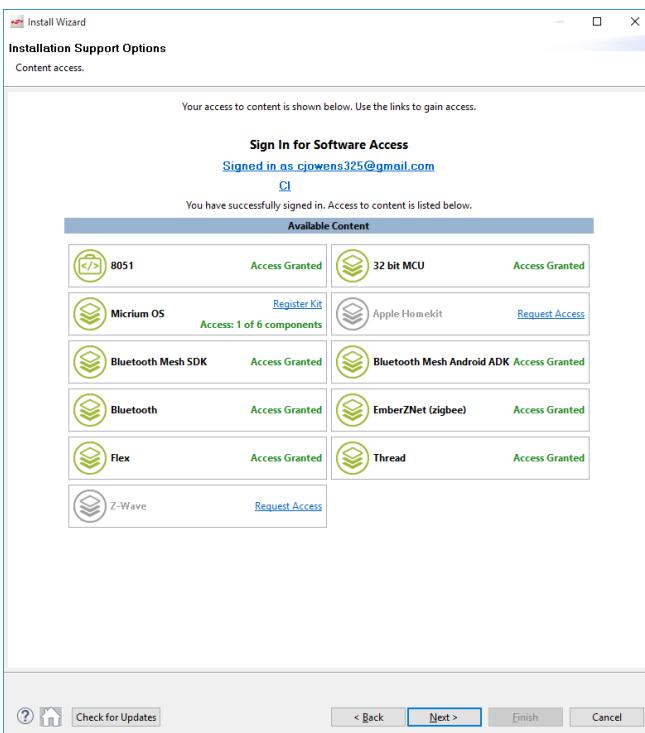
8. If you click **Install by Device**, an Install Device Support dialog appears. After a short delay, it shows your connected device. If the connected device does not show, click **Refresh**. Select either a connected device, or search for a product and select it. When a product is selected click **>>** to add it to the Selected Device pane. Simplicity Studio calculates available space required for installation. You can also click a selected device and click **<<** to remove it. Click **Next** to continue.



9. The next dialog varies depending on whether you have signed in and registered your kit. If you have not signed in, you have no access to restricted content and must sign in first (see the following figure a). If you have signed in but not registered the kit, you can see some restricted content but not EmberZNet (see the following figure b). Click **Register Kit**.



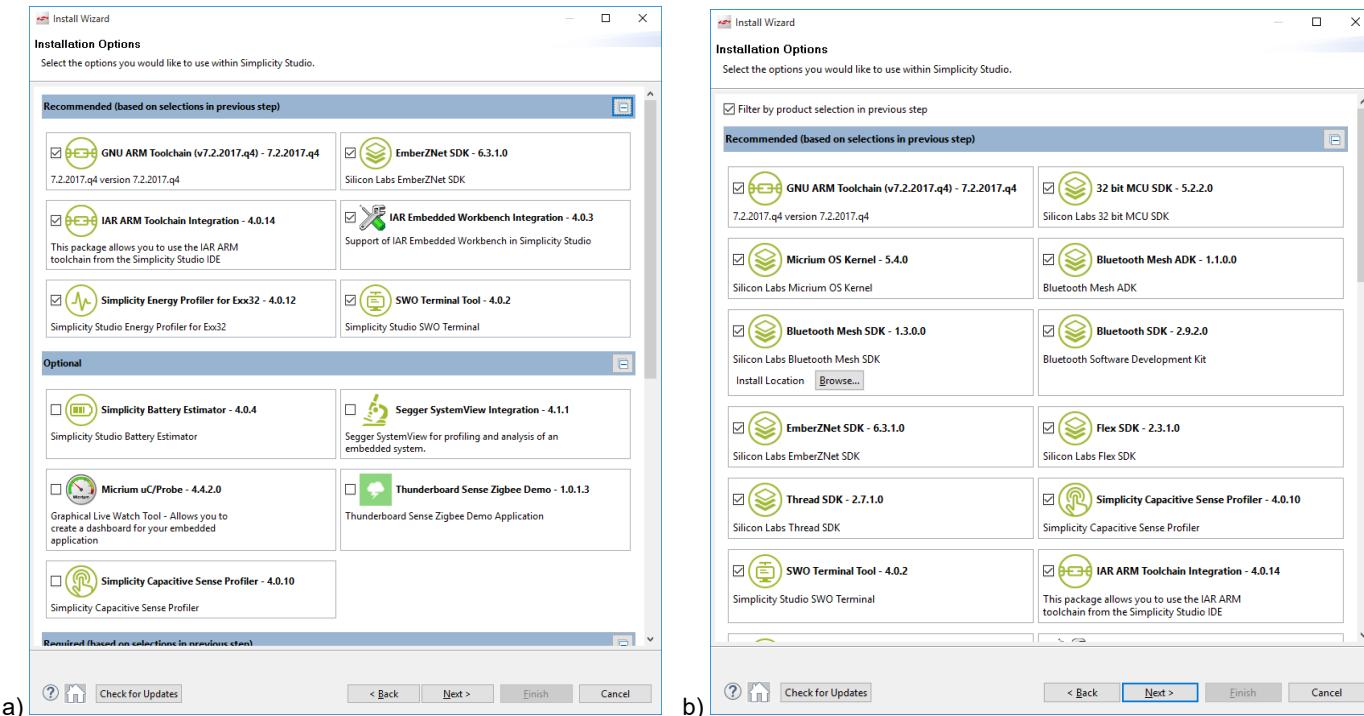
10. If you have already signed in and registered your kit, and see EmberZNet on the list of accessible components, click **Next**.



11. The **Installation Options** dialog shows the tools and software packages that can be installed (your versions may be different). The following shows Installation options after selecting the Zigbee product group (a), and after selecting an EFR32MG device (b). In both views you can uncheck anything you don't want to install. If you have installed by Product Group, the selection is filtered more specifically to your needs than if you have installed by device, and installing all checked options is recommended. If you have installed by device, and are unchecking: items:

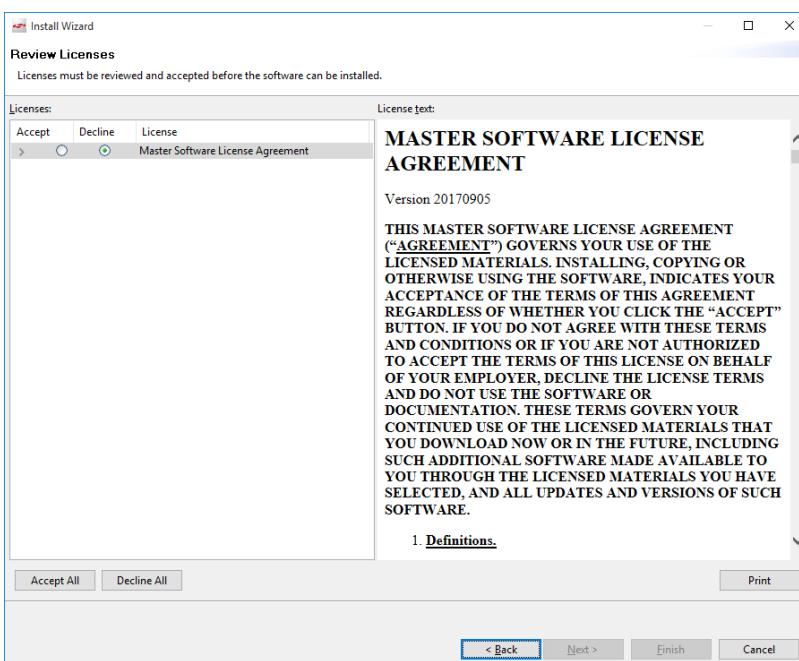
- If you plan to use GCC, leave GNU ARM Toolchain checked.
- If you plan to work on Dynamic Multiprotocol examples, leave Bluetooth SDK checked.

Click **Next >**

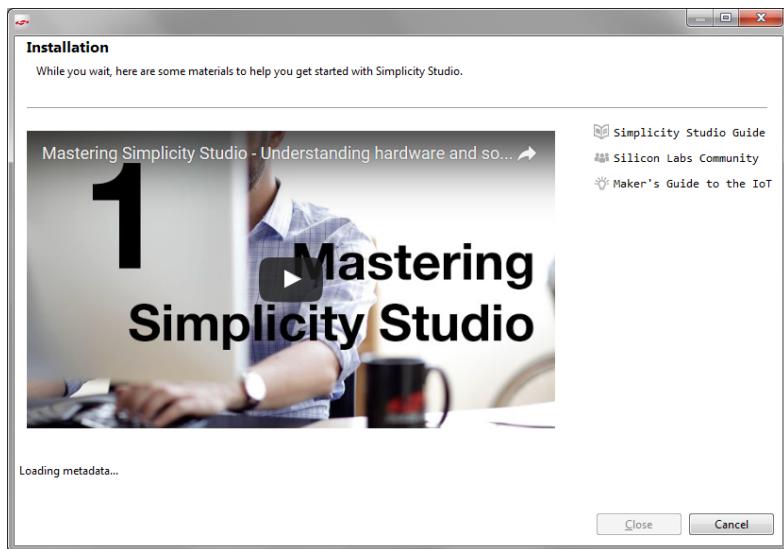


**Note:** Previous stack versions are shown under **Other Options**.

12. Studio displays a Review Licenses dialog. Accept the licenses shown and click **Finish**. Note that this dialog will present again if in the future you install a component with a separate license.



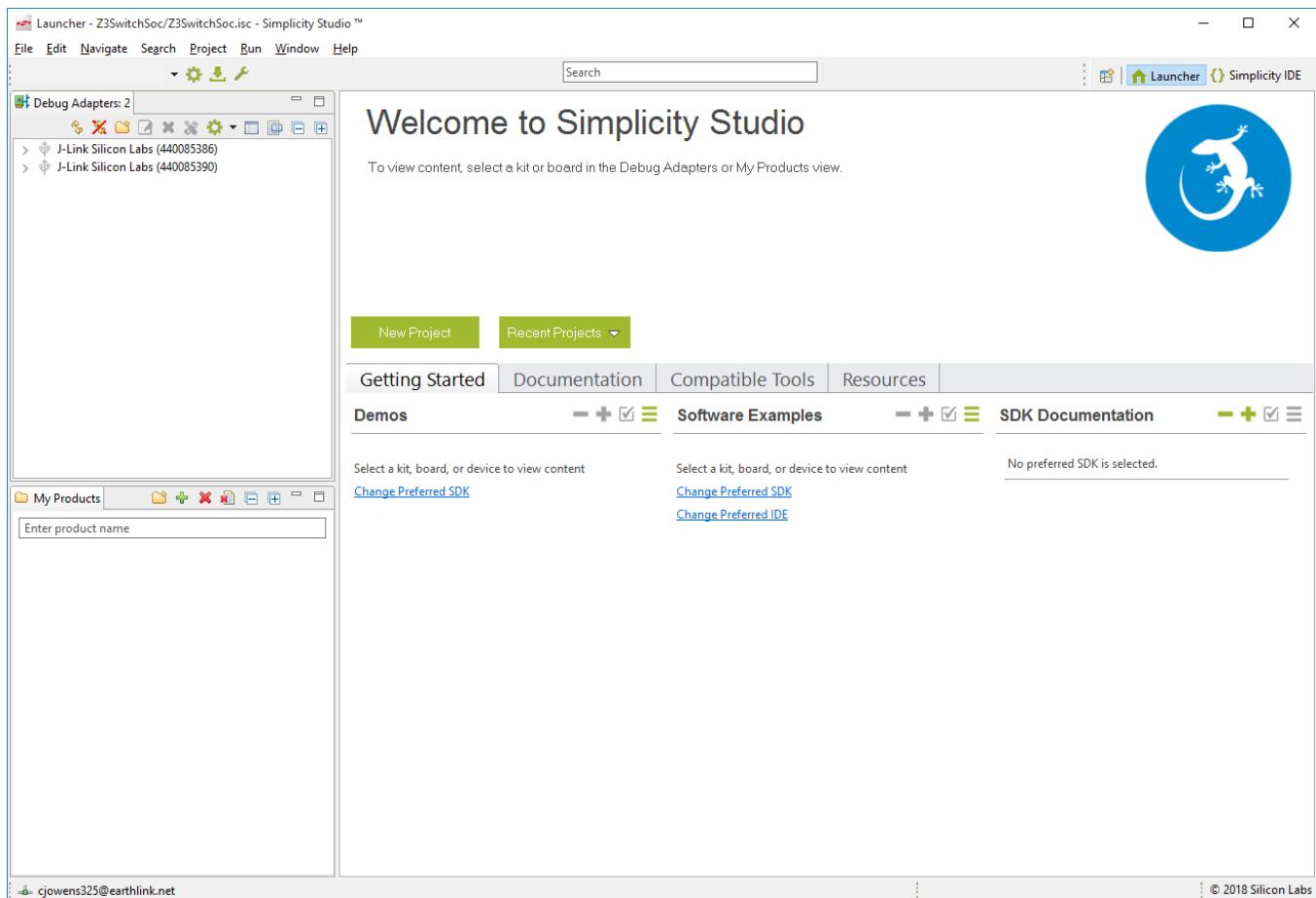
Installation takes several minutes. During installation, Simplicity Studio offers you viewing and reading options to learn more about the environment.



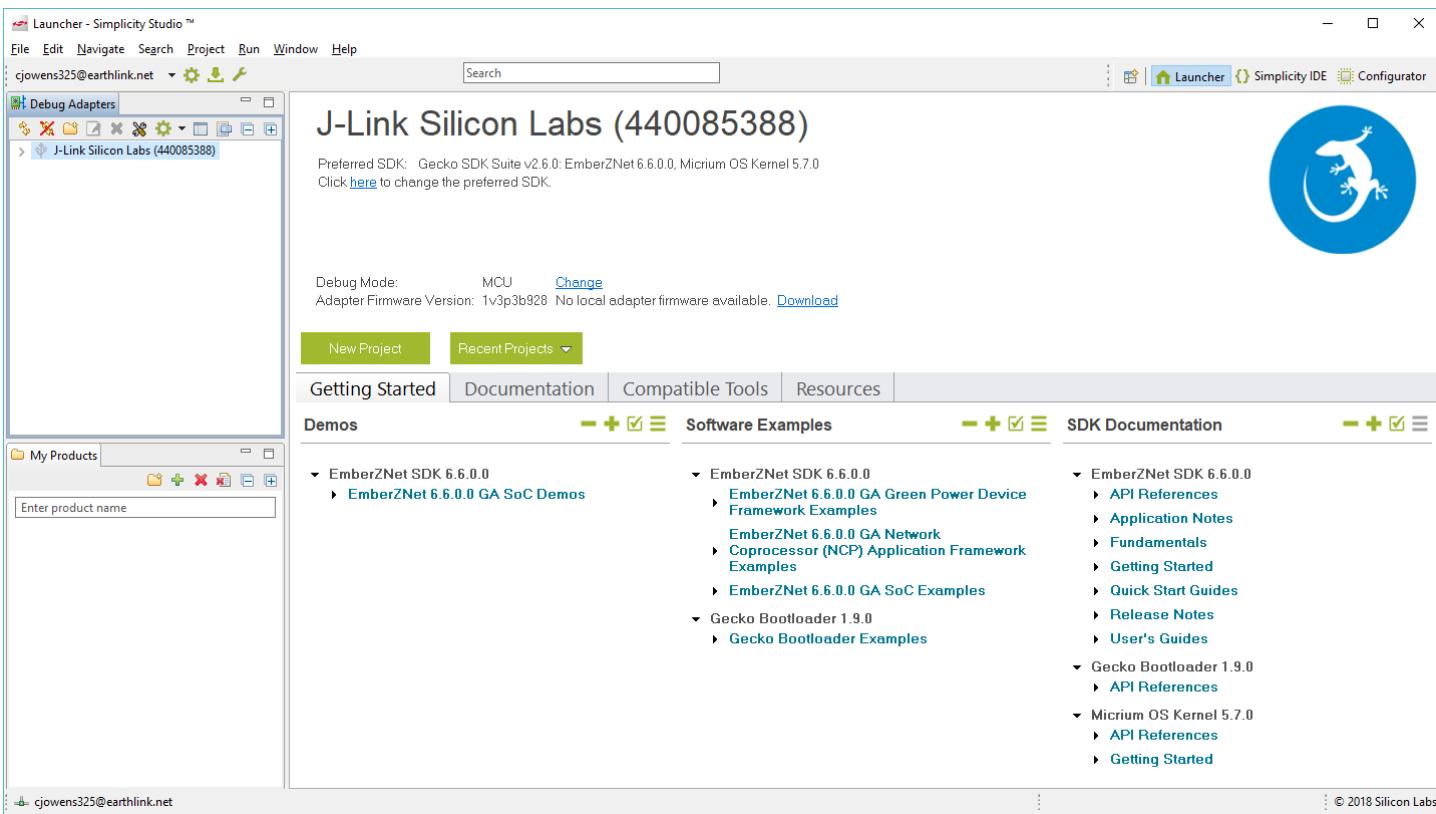
13. After installation is complete, restart Simplicity Studio.
14. When Simplicity Studio restarts, you are invited to take a tour. To clear this option now or at any time during or after the tour, click **Exit tour**.



15. The Launcher perspective opens, but it is not yet fully populated. Click a device in the Debug Adapters tab or find and select a part in the My Products tab. Note that USB-connected WSTK devices are identified as J-Link devices as shown.



16. The Launcher perspective then is populated with the software components and functionality associated with your hardware and stack. Note if you see Stackless applications, click the link to change the preferred SDK, as described in section [3.2 Changing the Preferred SDK](#). Update your device firmware as described in section [3.3 Updating Adapter Firmware](#).



Finally, if you plan to use IAR as your compiler (required for dynamic multiprotocol and Micrium OS examples), find the Release Notes on the SDK Documentation list and check for software version requirements, in particular for IAR-EWARM. To install IAR-EWARM:

1. On the Launcher page's Resources tab, click **Technical Support**.
2. Scroll down to the bottom of the page, and click **Contact Support**
3. If you are not already signed in, sign in.
4. Click the Software Releases tab. In the View list select **Latest EmberZNet Software**. Click **Go**. In the results is a link to the appropriate IAR-EWARM version.
5. Download the IAR package (takes approximately 1 hour).
6. Install IAR.
7. In the IAR License Wizard, click **Register with IAR Systems to get an evaluation license**.
8. Complete the registration and IAR will provide a 30-day evaluation license.

### 3 Functionality in the Launcher Perspective

Perspectives are made up of a number of tiles or panes, called views, as well as the content in those views. You can perform a number of functions in the Launcher Perspective, shown in the following figure. Additional information on some of these is provided later in the section. Note: Your installed version may be different than the version shown in the graphics in this section.

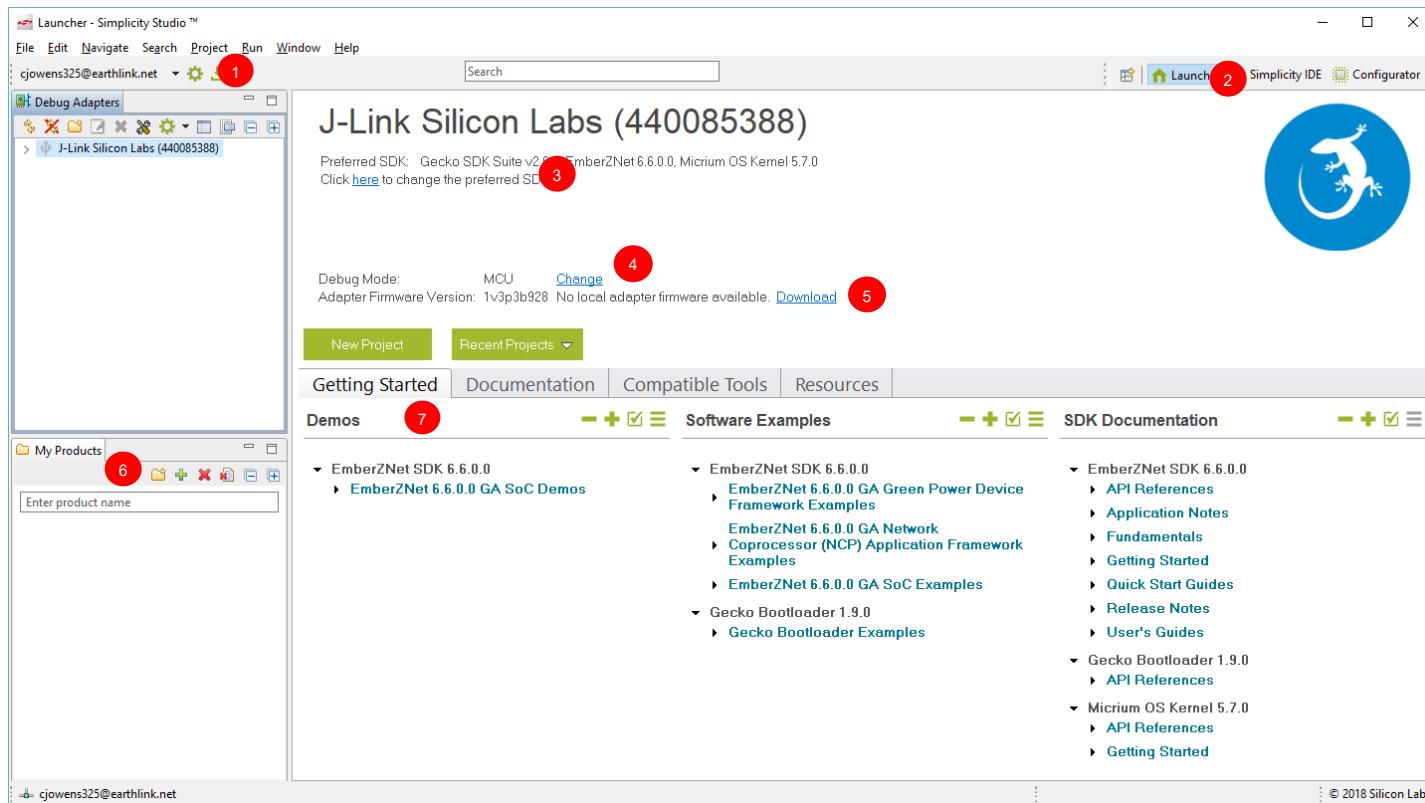
On the toolbar (1) you can:

- Sign in or out
- Open application settings (  )
- Update your software and firmware (  , see section [3.1 Updating Software/New Elements](#) for more information)
- Open the Tools menu (  ) to access tools such as Simplicity Commander or Energy Profiler.
- Search for information online, including entries in the Community forums.
- Change perspectives (2). As you open the Simplicity IDE or other tools, buttons for their perspectives are displayed in the upper right. Use those buttons to easily navigate back to the Launcher perspective or to other perspectives. You can change the layouts of various perspectives by expanding or relocating views, or adding or removing views. To return to the default layout, right-click the perspective button in the upper right and select **Reset**.

In the main view you can:

- Change your preferred SDK (3, see section [3.2 Changing the Preferred SDK](#) for more information - legacy functionality, rarely used).
- Change debug mode (4).
- Update adapter firmware (5, see section [3.3 Updating Adapter Firmware](#) for more information).
- Create solutions of multiple parts (6). If you are developing for complex networks with a number of different parts involved, you can add them all to the solution and then select the one you are working on from the list. You do not need to have the hardware connected to your computer.
- Access demos, examples, documentation, and other resources from the Getting Started and other tabs (7).

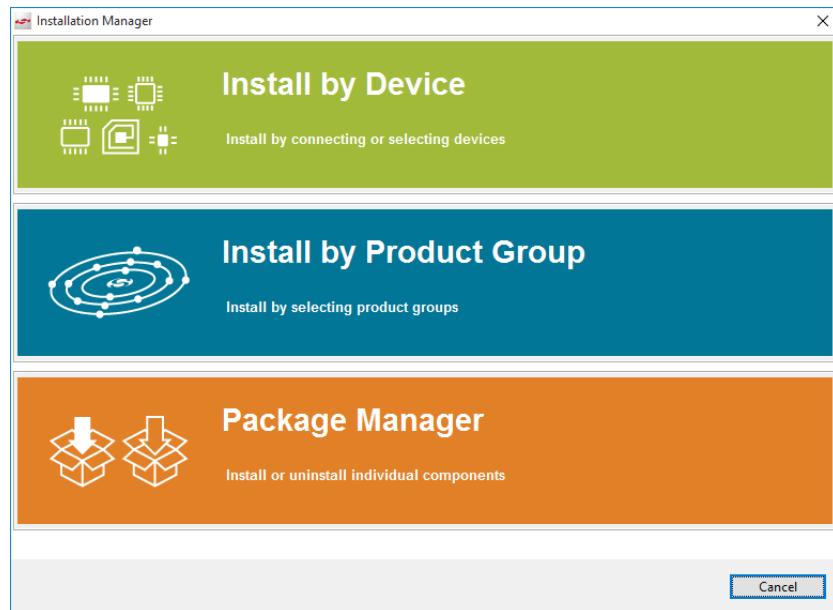
 Use these controls to manage groups of items (Collapse All, Expand All, Customize, and Show All, respectively). See section [3.4 Accessing Documentation and Other Resources](#) for more information.



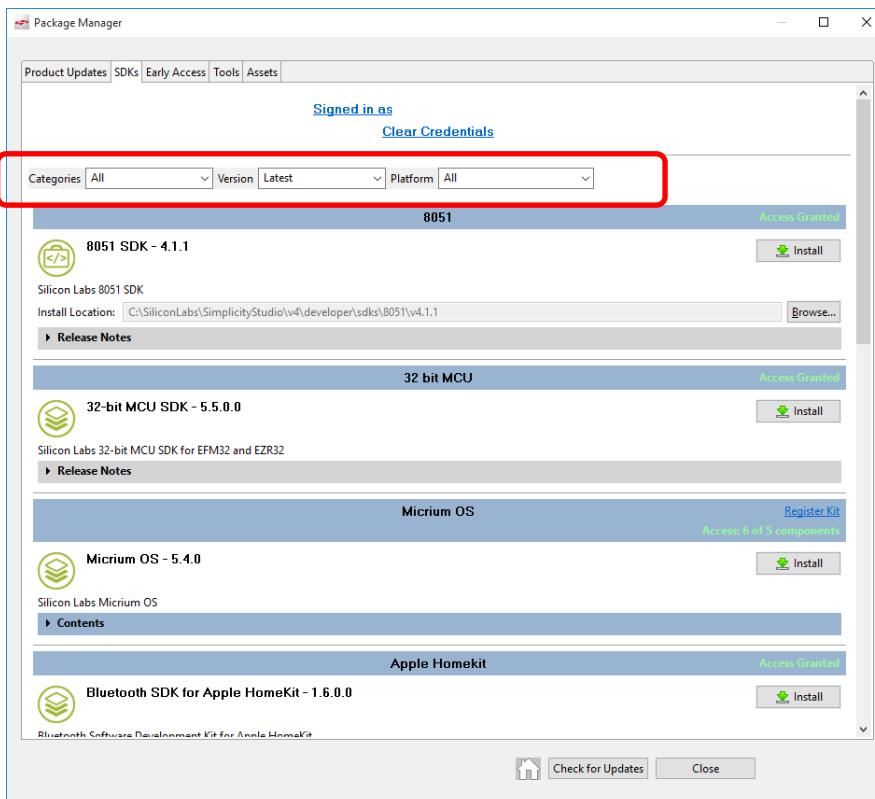
### 3.1 Updating Software/New Elements

The Update Software icon will be red if updates to installed components are available. If Simplicity Studio detects an available update, and you are in another perspective, you will be notified that an update is available.

To download a new or updated component, click the Update Software icon. Click Package Manager. Note: If you are installing based on a new device, or want to install a new product group, you can do so through this dialog as well. In subsequent dialogs, click Home ( ) to return to this dialog. Note that Simplicity Studio does not show you options, such as the GNU ARM Toolchain, that have already been installed.

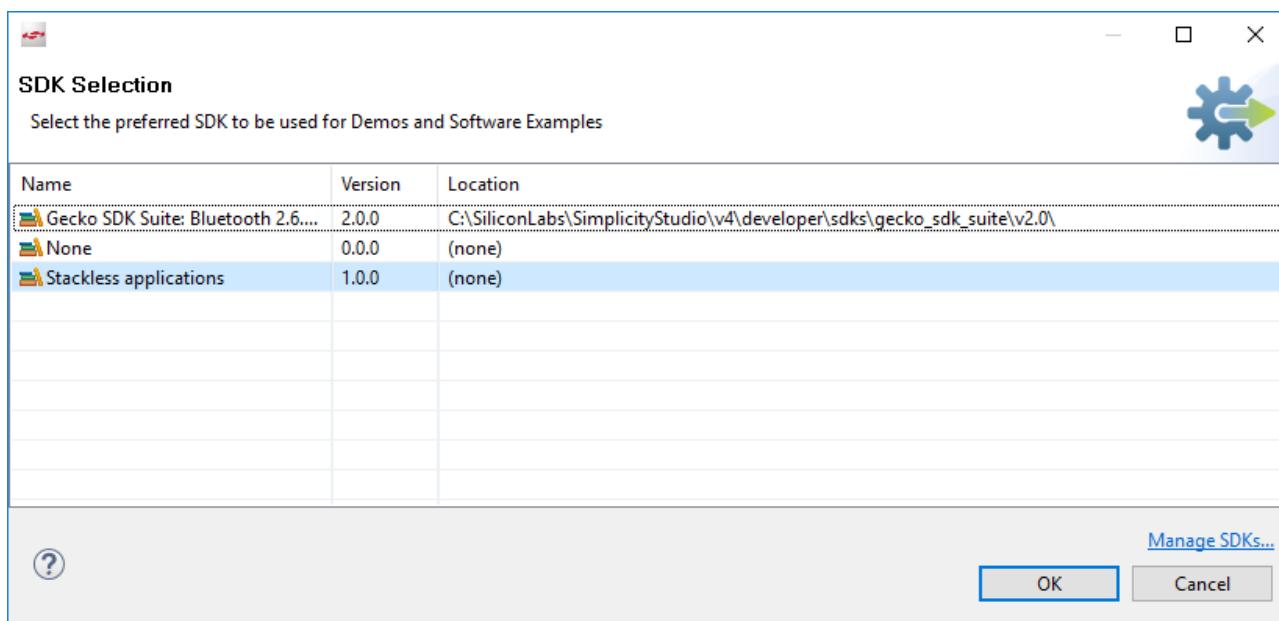


Simplicity Studio shows you available updates or SDKs in the Package Manager dialog. You can update all or select individual updates for installation. Click the tabs in the Package Manager dialog to see other components available for installation. Use the filters to reduce long lists.



### 3.2 Changing the Preferred SDK

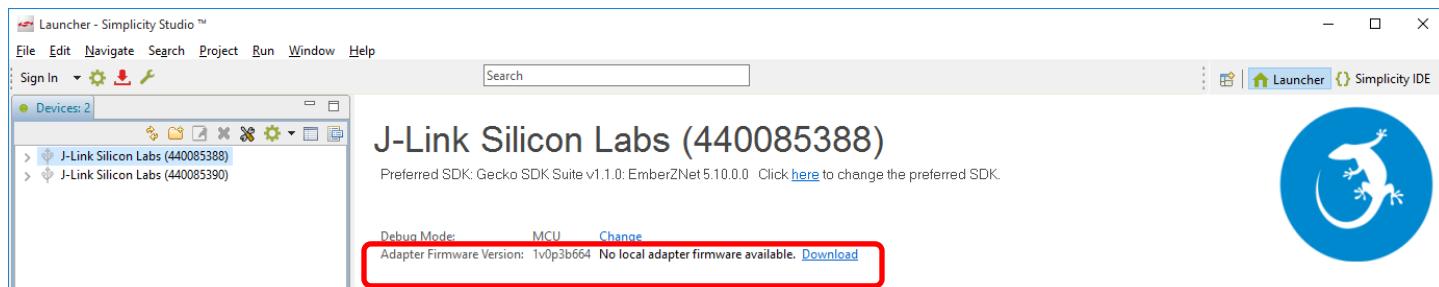
Use this function if on startup Simplicity Studio defaults to Stackless applications. Otherwise, most Silicon Labs protocol stack users will have one SDK available to them, the Gecko SDK Suite.



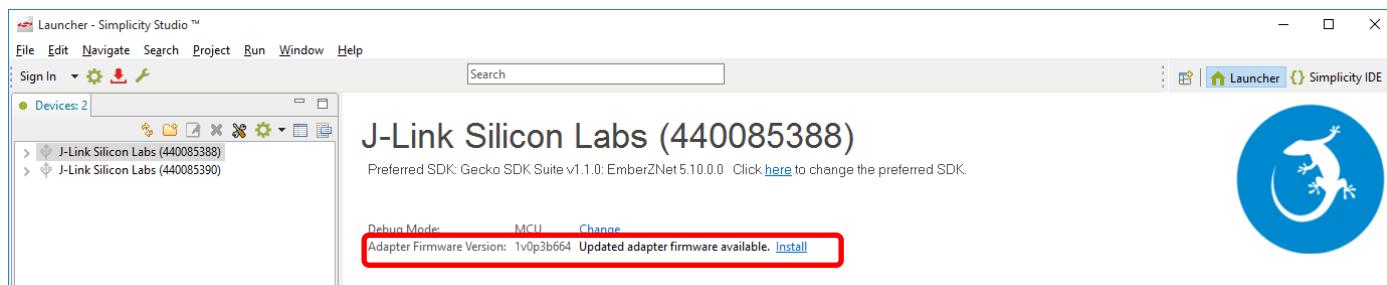
Within that suite you can have multiple protocols installed. The protocol used in any given instance is controlled either by the example you select, or the stack you select if you go through the 'New Project' interface. In general, you should add or remove protocol stacks through the Simplicity Studio update manager. If you need to install a stack or the Gecko SDK Suite outside of the normal installation process, you will receive separate instructions.

### 3.3 Updating Adapter Firmware

Initially the Launcher perspective may display "No local adapter firmware available." Click **Download** to download any updates.



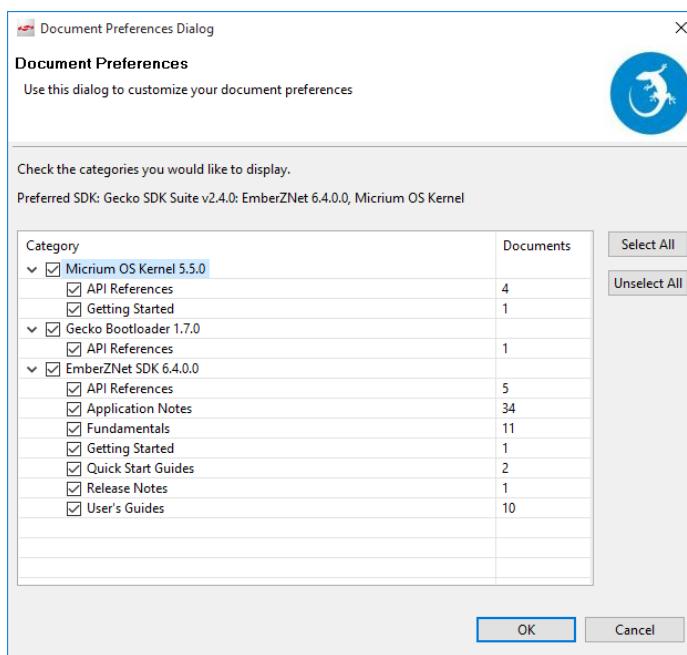
If an update is available, click **Install** to install the firmware.



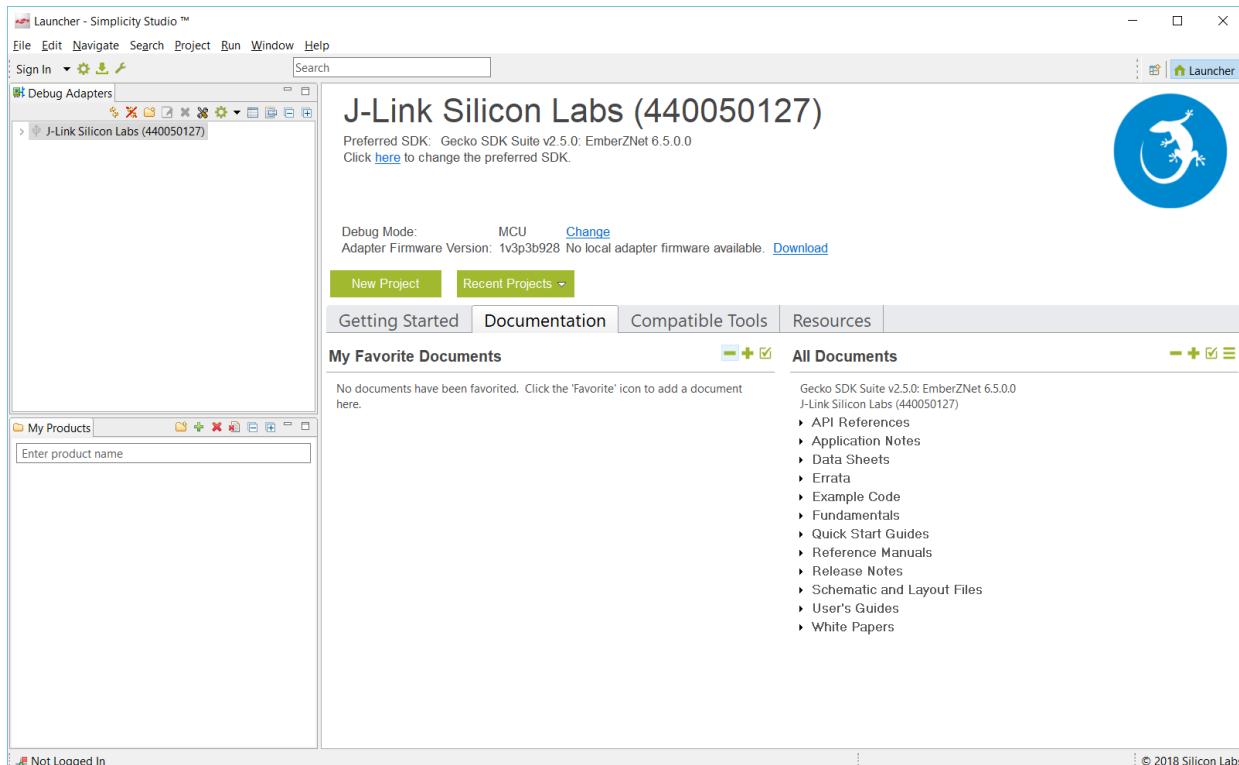
Once you have installed a current update, the version is displayed. Simplicity Studio will notify you if another firmware update is available.

### 3.4 Accessing Documentation and Other Resources

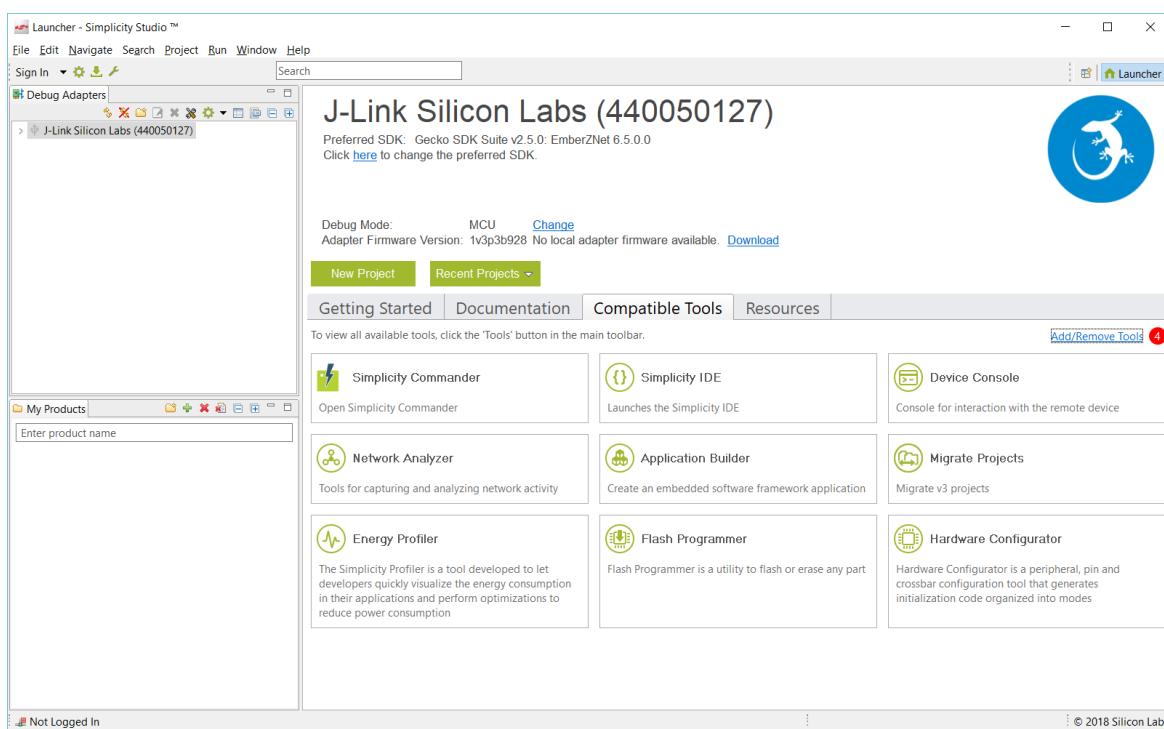
The Getting Started tab provides access to demos, example applications, and stack-related documentation. To show/hide specific categories, click . Select or deselect categories, then click **OK**.



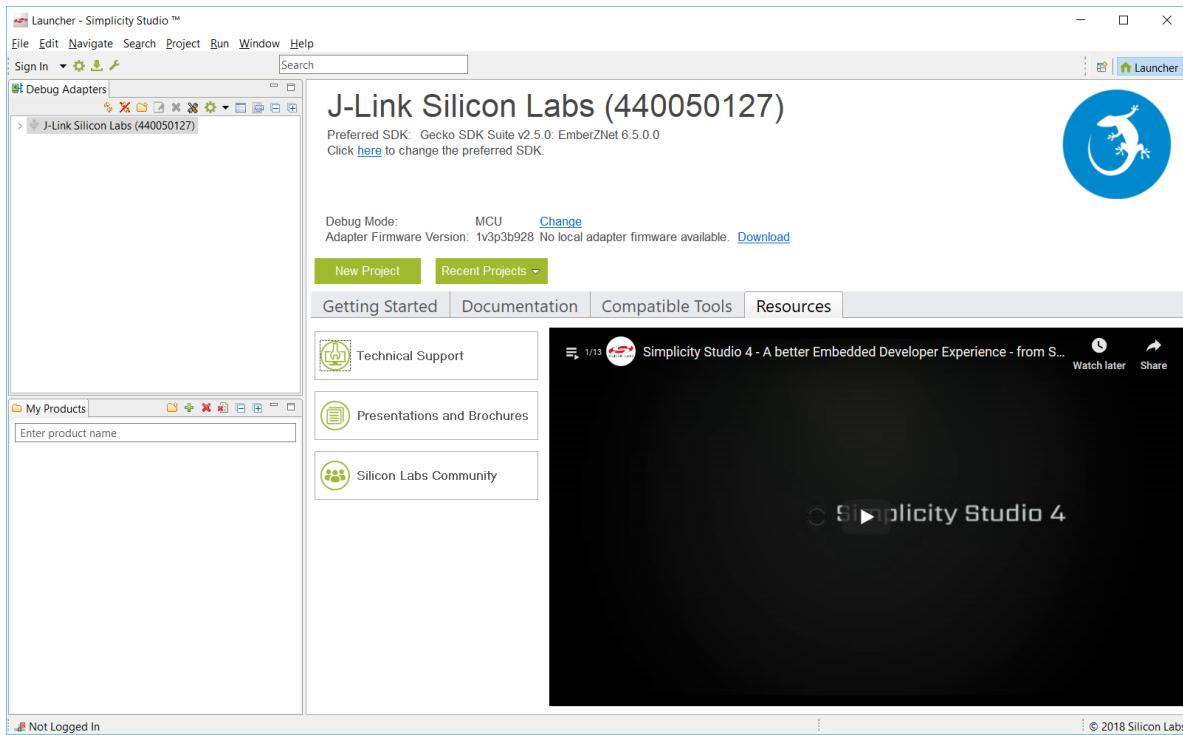
The **Documentation** tab lists all documentation on the right, including documents on the hardware platform you are using as well as the SDK documents. Documents you select as favorites are shown on the left. Click the star icon on any document to move it to the My Favorite Documents list.



The **Compatible Tools** tab is an alternative way to access the tools available through the Tools dropdown.



The **Resources** tab provides access to support, marketing collateral, and the Silicon Labs community.



## 4 About Demos and Examples

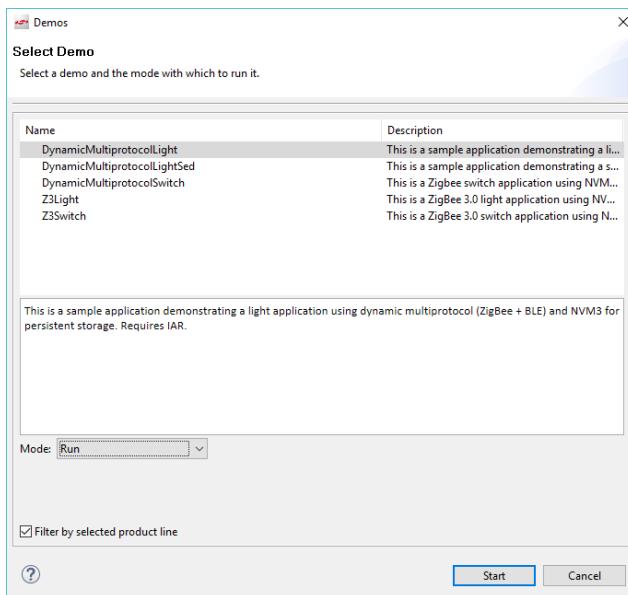
Because starting application development from scratch is difficult, the EmberZNet SDK comes with a number of built-in demos and examples covering the most frequent use cases. Demos are pre-built application images that you can run immediately. Software examples can be modified before building the application image. The software examples with the same names as the demos provide the demo functionality.

**Note:** The demos and examples you see are determined by the part selected. If you are using a custom solution with more than one part, be sure to click on the part you are working with to see only those items applicable to that part.

Silicon Labs recommends that you start with an example and modify it according to your needs instead of starting with a blank project. The examples provide default configurations needed by the stack and a basic application structure that you can build on. However, if you want to start with a blank project you can.

### 4.1 Demos

Demos are prebuilt application examples that can be directly downloaded to your device. To download and run a demo on your device, if you have more than one device connected make sure the target device is selected in the Device pane. Drop down the demo list and click the demo. In the Mode drop-down in the next dialog, select **Run**. Click **Start**.



The EmberZNet demos are:

**DynamicMultiprotocolLight**, **DynamicMultiprotocolLightSed**, and **DynamicMultiprotocolSwitch**: Companion applications demonstrating Zigbee/Bluetooth LE dynamic multiprotocol functionality. See [QSG155: Using the Silicon Labs Dynamic Multiprotocol Demonstration](#) for instructions on using the demos.

**Z3Light**: Zigbee 3.0 Light application.

**Z3Switch**: Zigbee 3.0 Switch application.

For the Light and Switch demos, press Button0 on the Switch device to initiate the network. The light device should bind to the network automatically. Once the switch has finished finding and binding, you can use Button0 as an On/Off toggle, and Button1 to send level commands. The level changes may be difficult to detect on the board.

## 4.2 Software Examples

**Note:** Examples provided for the EFR32xG12 and newer parts include Silicon Labs Gecko Bootloader examples. Examples are provided for all compatible Simplicity Studio SDKs. When configuring security for a Gecko Bootloader example, you must use Simplicity Commander, not the Simplicity Studio IDE interface. For more information on using the Gecko Bootloader see *UG266: Silicon Labs Gecko Bootloader User Guide*.

The EmberZNet software examples are grouped by category. If you are not familiar with the differences between System-on-Chip (SoC) and Network Coprocessor (NCP) application models, see *UG103.03: Application Development Fundamentals: Design Choices*. For more information on Green Power Devices see *UG392: Using Silicon Labs Green Power with EmberZNet PRO*.

You can start an example from the list on the Launcher perspective, as described in section [5 Working with Example Applications](#). This list is filtered by the selected device. To see all examples, including the Host and NCP examples, and to have additional configuration options including project name, click **New Project**. If you access examples through **New Project**, the NCP examples are based on the **Customizable network coprocessor (NCP) applications** type, and the SoC and Host examples are based on the **Silicon Labs Zigbee** type. Be sure to check the complete example description for any platform restrictions.

**Note:** The checkbox “Start with a blank application” on the screen listing the applications produces an incorrectly configured starting project, and should not be used.

The following descriptions group related examples together. Examples followed by (SoC) or (Host) are available through the **Silicon Labs Zigbee|EmberZNet <version> SoC** or **EmberZNet <version> Host** stacks, respectively. Examples followed by (NCP) are available through **Customizable network coprocessor (NCP) applications**. Examples followed by (GP) are available through **Green Power Device Framework**.

## 4.3 Dynamic Multiprotocol Demonstration

These applications are the source for the EmberZNet/Bluetooth dynamic multiprotocol demos described in *QSG155: Using the Silicon Labs Dynamic Multiprotocol Demonstration*. All applications are configured with NVM3 for persistent storage.

**DynamicMultiprotocolLight (SoC):** Demonstrates the Zigbee and Bluetooth stacks running concurrently as MicriumOS tasks.

**DynamicMultiprotocolLightSed (SoC):** Demonstrates the Zigbee and Bluetooth stacks running concurrently as MicriumOS tasks. The Light is configured as a Zigbee Sleepy End Device (SED) and searches for a Network to join.

**DynamicMultiprotocolSwitch (SoC):** This application only uses Zigbee and is not a dynamic multiprotocol application itself. It is to be used with the **DynamicMultiprotocolLight** sample to demonstrate the control of a dynamic multiprotocol device from a Zigbee network.

## 4.4 Zigbee 3.0 Network

**Z3Light (SoC):** Zigbee light application. Acting as a router it can form a distributed network. Acting as a touchlink target it can touchlink with the **Z3Switch**, which is acting as a touchlink initiator.

**Z3LightGPCCombo (SoC):** Zigbee light application with Green Power combo basic functionality.

**Z3Switch (SoC):** Zigbee switch application, acting as an end device, can join the distributed network.

**Z3Gateway (Host):** Simple gateway application that can form a centralized network, and the light and the switch can join the centralized network by performing network steering.

## 4.5 Host/NCP Applications Controlling an LED

**XncpLed (Host):** Presents a HOST application for communicating with an NCP application using custom EZSP commands. It is meant to be used with the NCP sample application xNCP LED.

**xNCP LED (NCP):** This sample scenario presents an NCP application for communicating with a UNIX HOST using custom EZSP commands. This application is meant to be used with the HOST sample application **XncpLed**.

## 4.6 NCP Application Examples

All applications can be built as configured, or optionally can be augmented with customizations for target hardware, initialization, main loop processing, event definition/handling, and messaging with the host.

**NCP SPI (NCP):** This network coprocessor (NCP) application supports communication with a host application over an SPI interface.

**NCP UART HW (NCP):** This network coprocessor (NCP) application supports communication with a host application over a UART interface with hardware flow control.

**NCP UART SW (NCP):** This network coprocessor (NCP) application supports communication with a host application over a UART interface with software flow control.

## 4.7 Green Power Device

**Green Power Device:** Green Power Device example.

## 5 Working with Example Applications

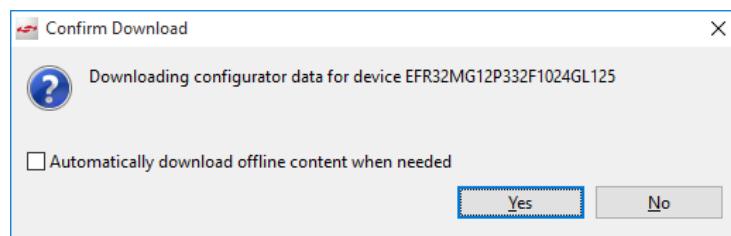
In these instructions you will compile and load two example applications, Z3Light and Z3Switch. Section 6 [Creating a Network](#) describes how to use the examples to create a network. Section 7 [Using the Network Analyzer](#) describes how to use Network Analyzer to observe traffic across the network.

When working with example applications in Simplicity Studio, you will first execute the following steps:

1. Select an example application.
2. Generate application files.
3. Compile and flash the application (and, the first time, a bootloader) to the radio board.

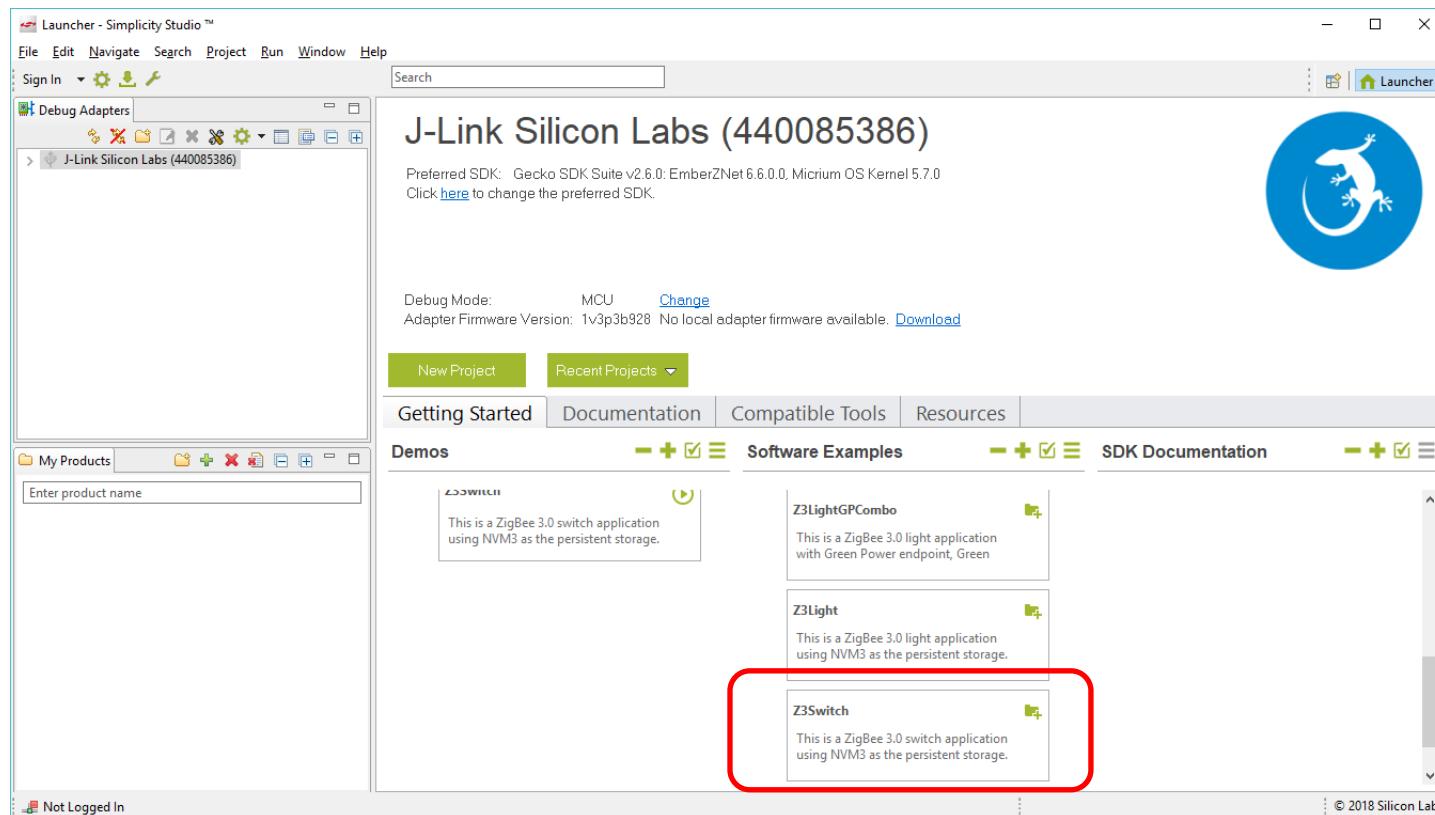
These steps are described in detail in the following sections. These procedures are illustrated for a WSTK with an EFR32MG. **Note:** Your SDK version may be later than the version shown in the procedure illustrations.

Note: SDK version 6.0 and later contain a number of changes to the way hardware peripherals are configured and managed. You may see the following dialog when you open an example or generate code. Always click **Yes**.

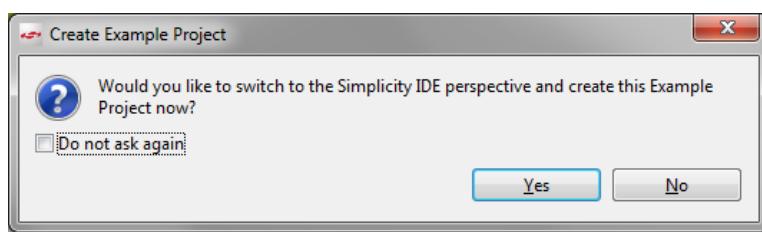


### 5.1 Selecting an Example Application

1. In the Launcher perspective, click an example application, in this case Z3Switch. Your project will be based on this example, and on the device you have selected in the Devices or Solutions tabs on the left.



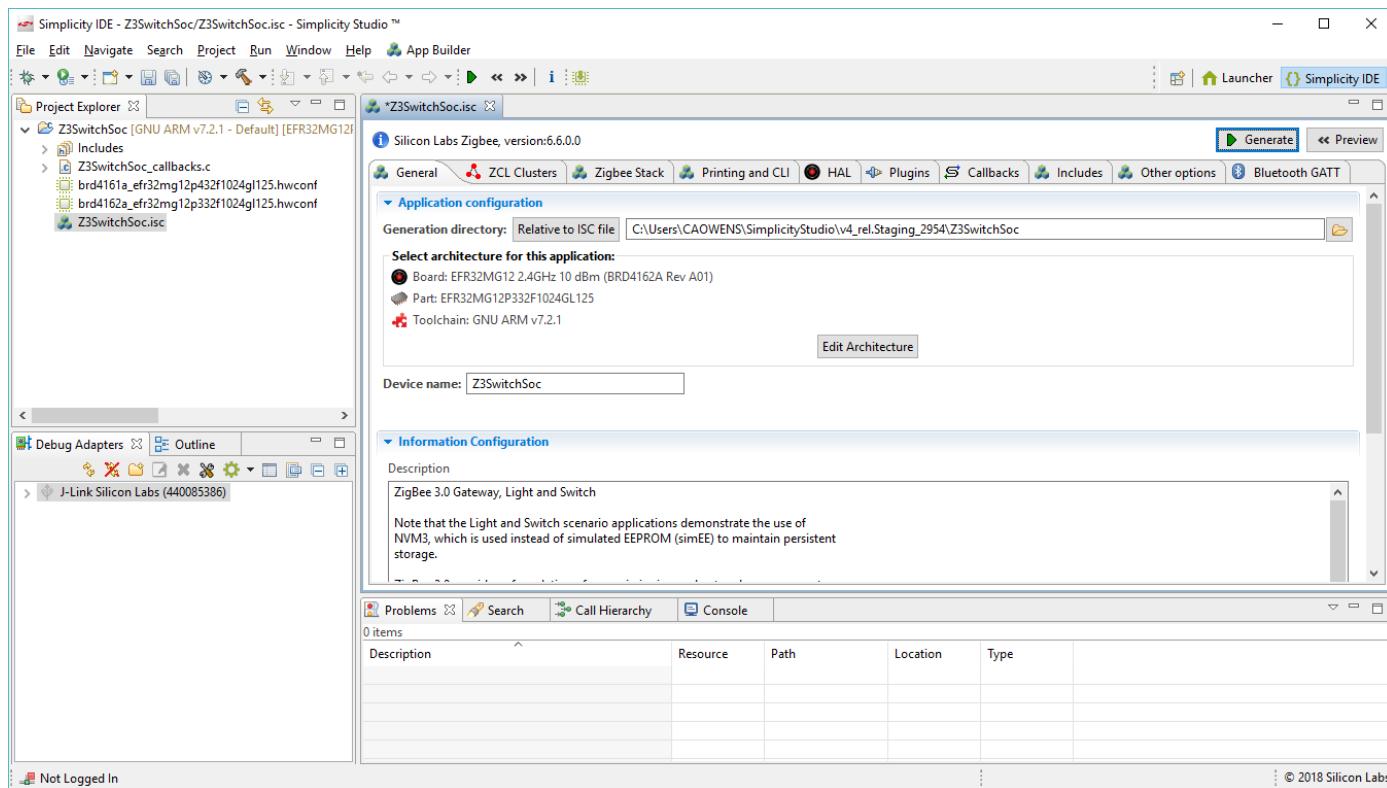
2. You are asked if you want to switch to the Simplicity IDE. Click **Yes**.



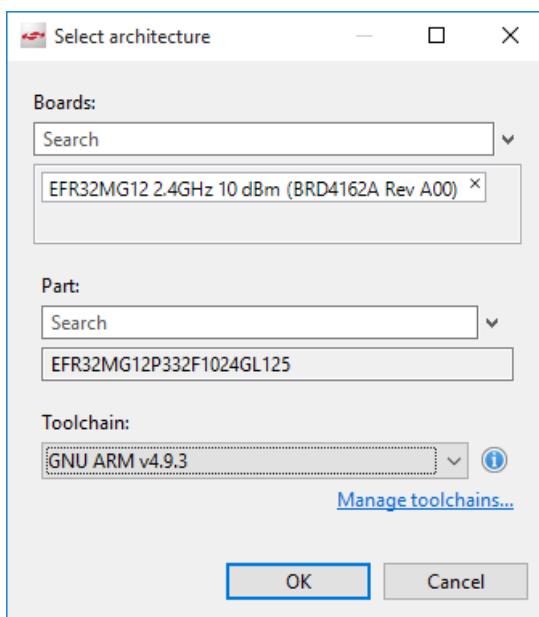
3. Simplicity IDE opens with the new project in AppBuilder view, and the focus on the General tab.

**Note:** You now have a Simplicity IDE button next to the Launcher button in the upper right.

Make sure the toolchain shown is the one you want to use. If you have both IAR and GCC installed, GCC is the default. Note that if you are compiling the example for a part with less than 512 kB, such as the EFR32xG1, or you are compiling the Dynamic Multiprotocol Light(s) or Switch examples, you must use IAR.

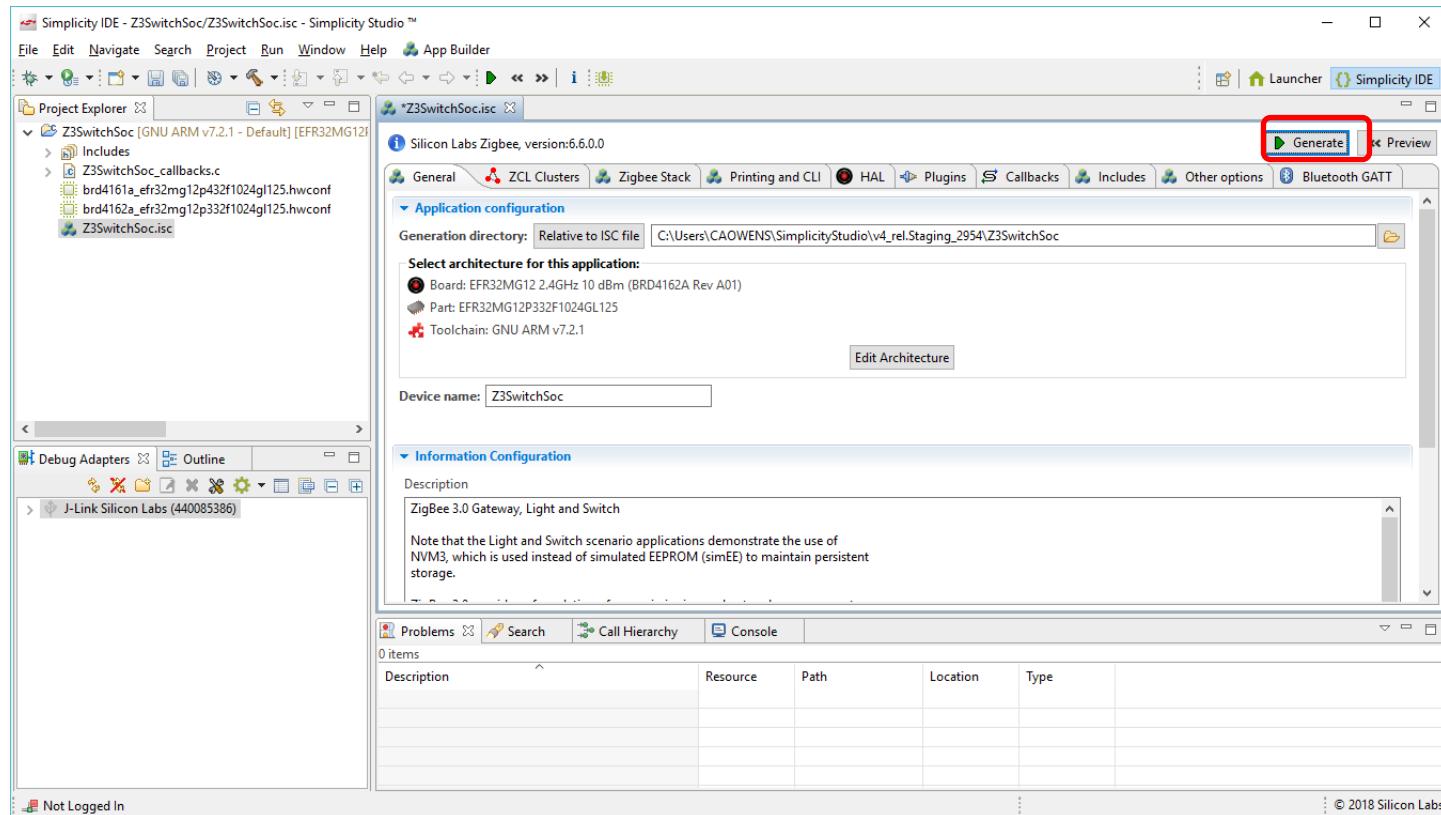


4. To change the toolchain click **Edit Architecture**. In the resulting dialog, select the desired toolchain and click **OK**.

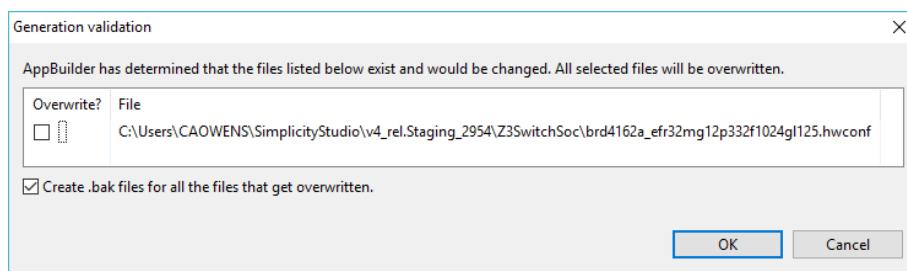


## 5.2 Generating the Application Source Files

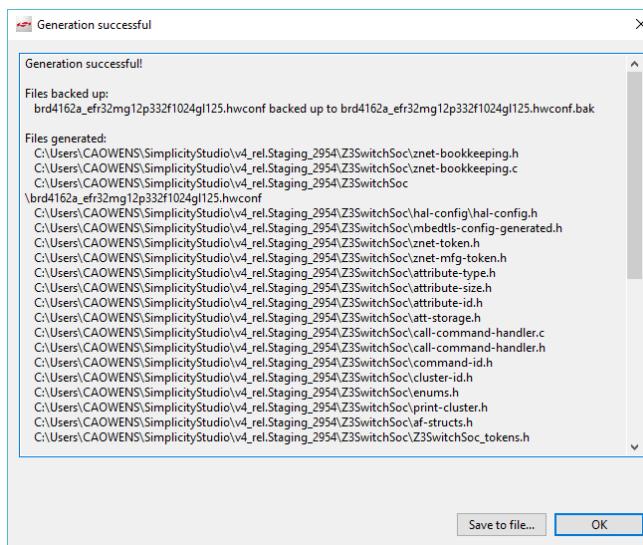
1. In the Simplicity IDE, click **Generate**.



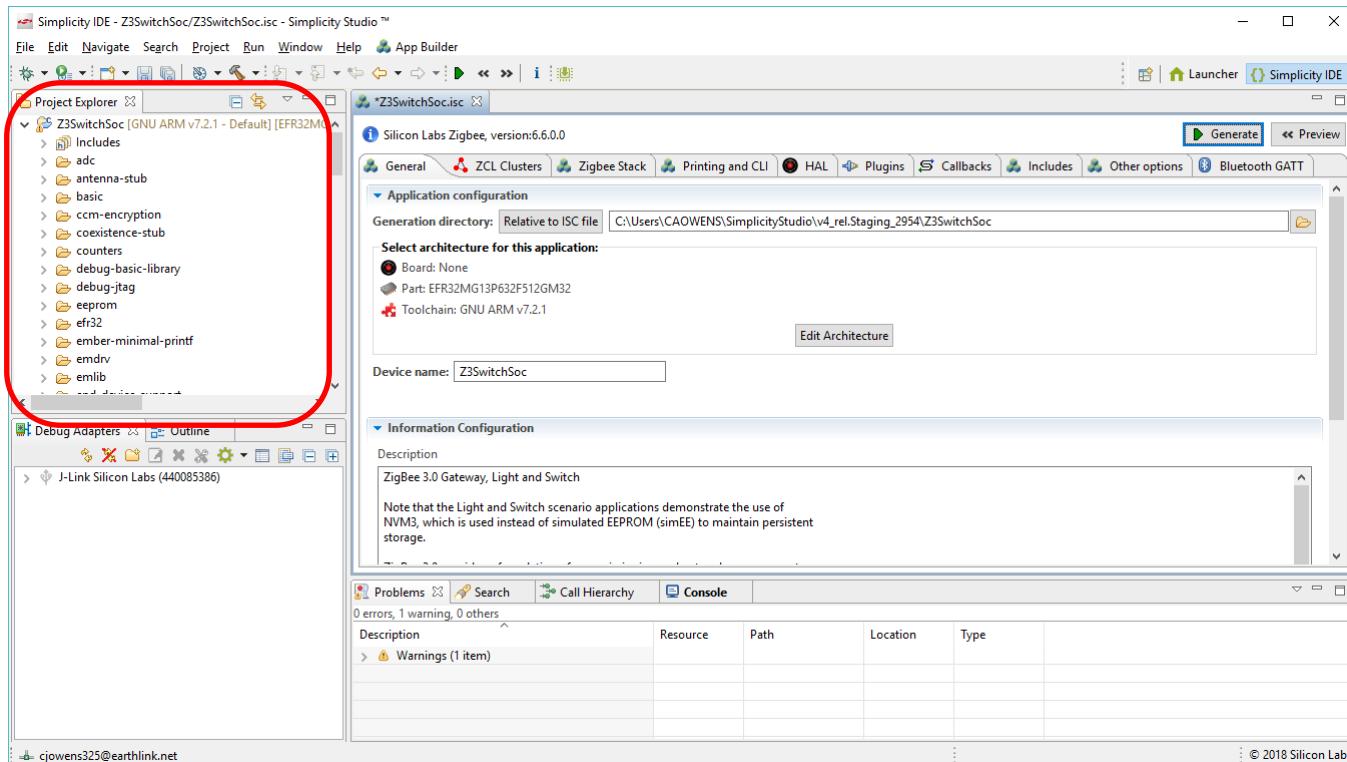
If you get the following overwrite warning, click **OK**.



2. Once generation is complete, a dialog reporting results is displayed. Click **OK**.



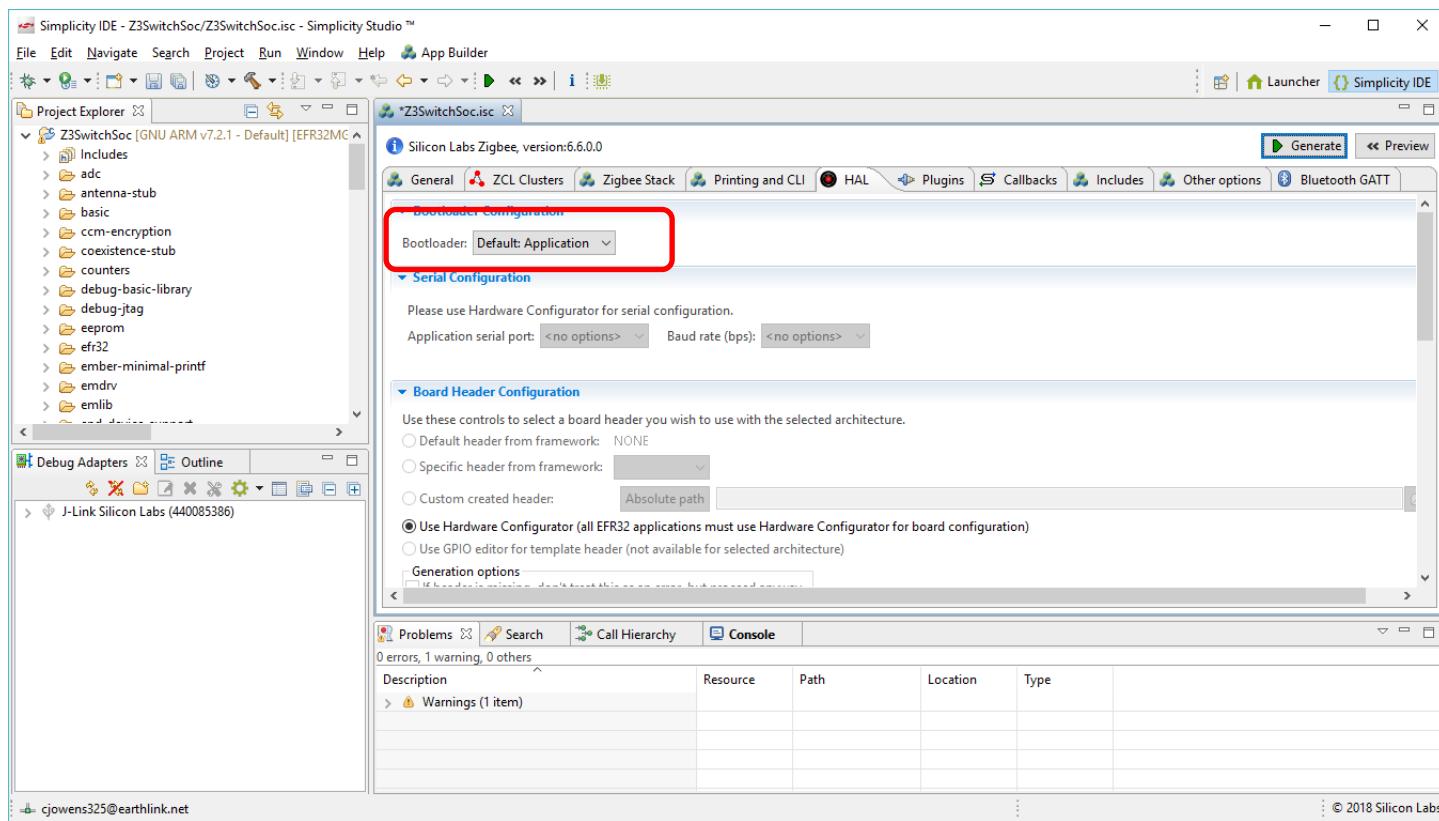
The generated files are shown in the Project Explorer view.



## 5.3 Compiling and Flashing the Application

### 5.3.1 About Bootloading

Because this sample application is built with a bootloader (configured under the HAL tab), you will need to load a bootloader before you run the application for the first time.



A bootloader is a program stored in reserved flash memory that can initialize a device, update firmware images, and possibly perform some integrity checks. Silicon Labs networking devices use bootloaders that perform firmware updates in two different modes: standalone (also called standalone bootloaders) and application (also called application bootloaders). An application bootloader performs a firmware image update by reprogramming the flash with an update image stored in internal or external memory. Silicon Labs recommends that you always flash a bootloader image along with your application, so that flash memory usage is appropriately allocated from the beginning. For more information about bootloaders see [UG103.6: Application Development Fundamentals: Bootloading](#).

In March of 2017, Silicon Labs introduced the Gecko Bootloader, a code library configurable through Simplicity Studio's IDE to generate bootloaders that can be used with a variety of Silicon Labs protocol stacks. The Gecko Bootloader is used with all EFR32xG parts.

The Gecko Bootloader works with a specialized firmware update image format, ending in the extension .gbl (the GBL file). When you build an application both .s37 and GBL files are generated. The exact format of the GBL file depends on the hardware you selected.

**Note:** When working with the Gecko Bootloader, you must use Simplicity Commander to enable some configuration option such as security features. See [UG266: Silicon Labs Gecko Bootloader User's Guide](#).

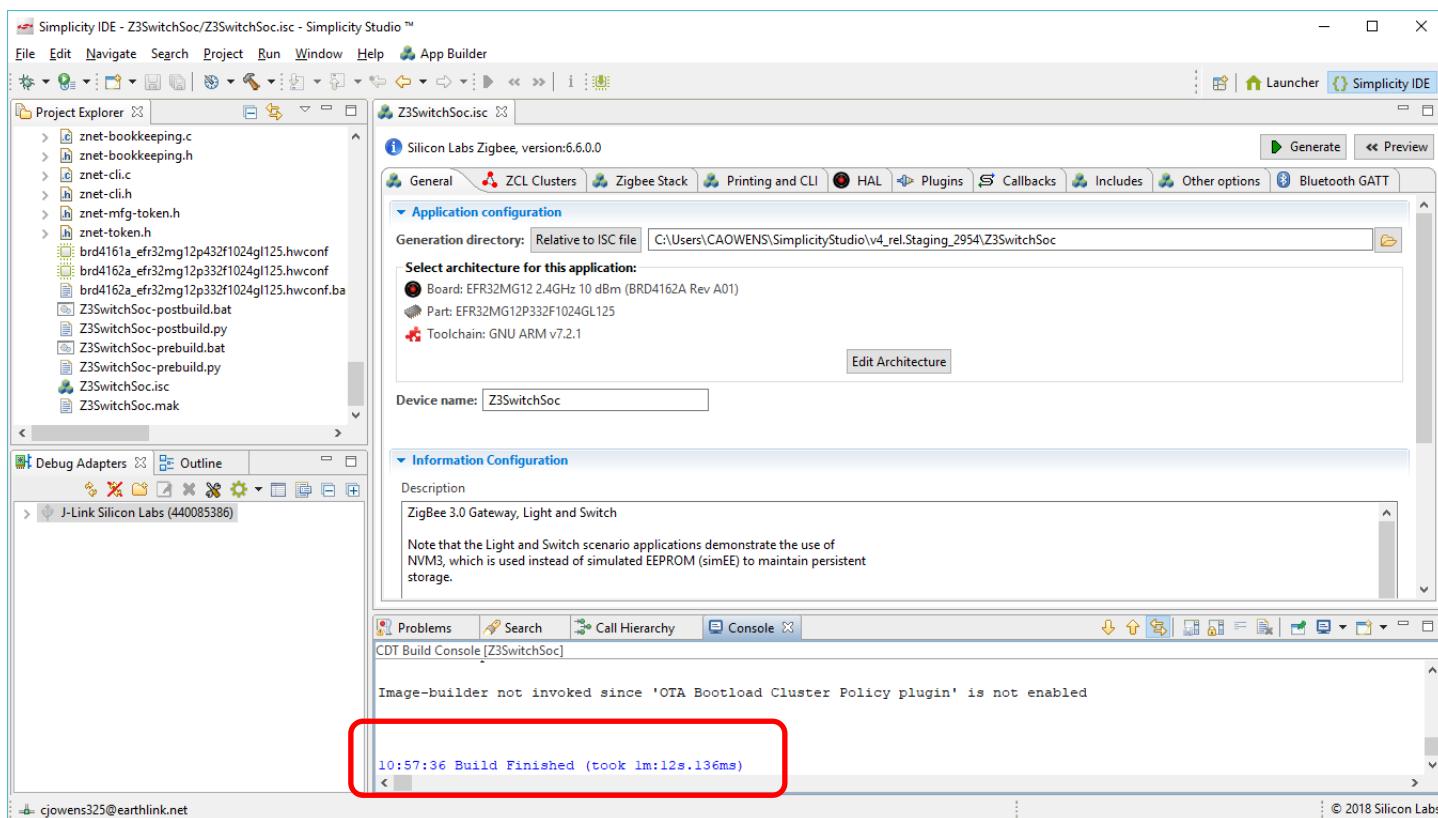
### 5.3.2 Building and Flashing Files

- After you generate your project files, click the **Build** control in the top tool bar (  ).

If the Build control is not enabled, click the device. Your example application will compile based on its build configuration. You can change the build configuration at any time in the Project Explorer View by right clicking on the project and going to **Build Configurations > Set Active**.

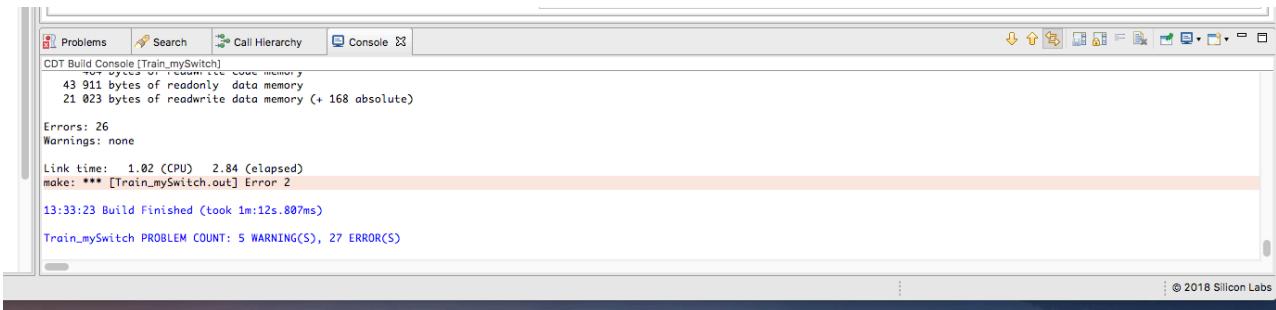
During the build, progress is reported both in a window, which can be run in the background, and also in the lower right. The process may take over a minute.

Build completion is reported in the Build Console.



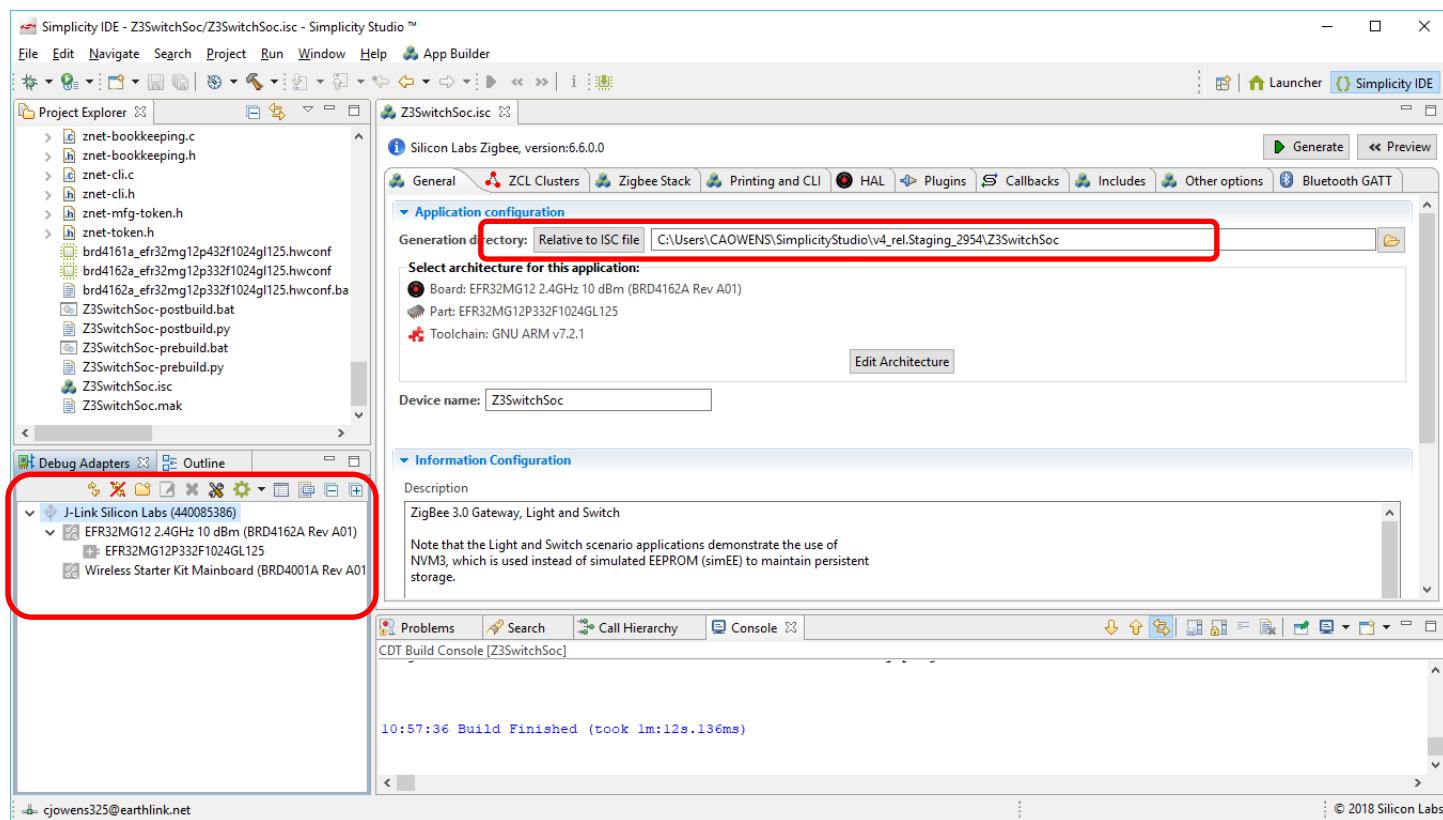
The build should complete with no errors. The line above the completion line is not an error. It refers to an option, not enabled in the default example configuration, to build a special application image used by the Zigbee Cluster Library Over-the-Air (OTA) bootloader functionality.

If any errors occur they are highlighted in red in the console. Contact technical support for assistance.

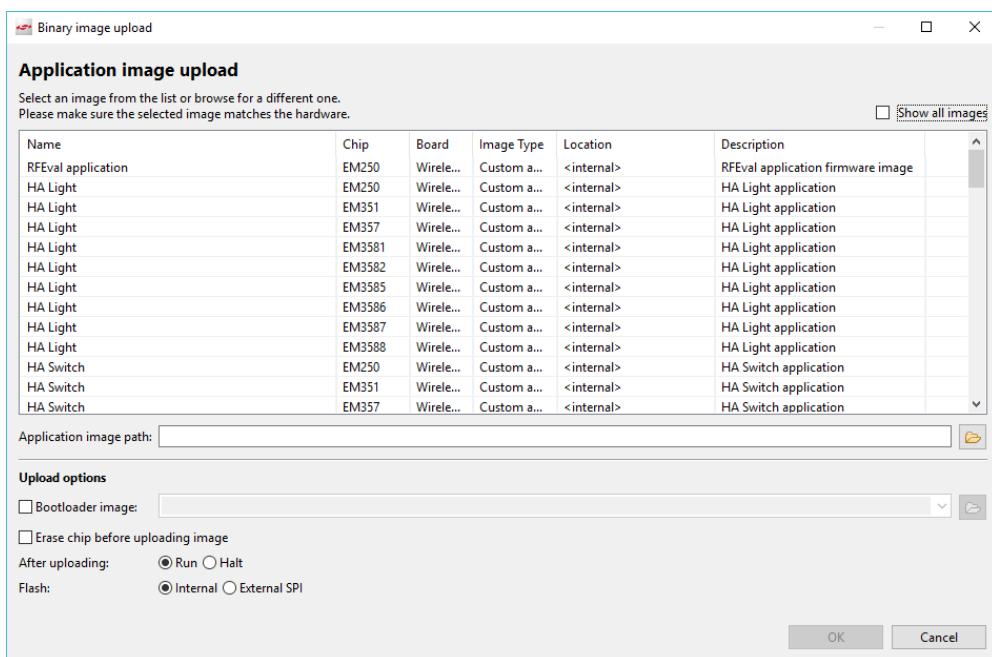


Note: If you are using GCC on a Windows platform, and experience a build error concerning a missing include file, you may have encountered a known issue concerning GCC and long pathnames. If this happens, you may be able to work around the problem by opening the Windows Registry and setting the 'HKLM\SYSTEM\CurrentControlSet\Control\FileSystem' registry key to a value of 1. If after making this adjustment you still experience build issues with GCC and the example applications, please contact Silicon Labs technical support.

2. To load the application and the bootloader images, first make sure your hardware is displayed in the Device perspective. Expand the radio board to show the part number, as you will need that along with the board number to find the correct bootloader folder. Note that the folder in which the example was generated is displayed on the General tab.

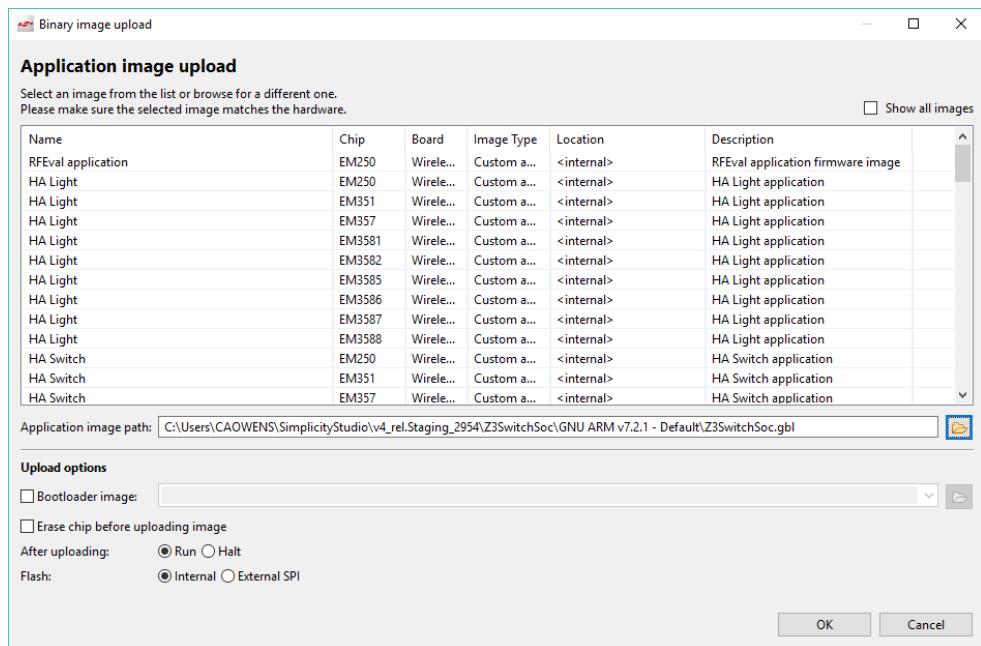


3. Right-click the device and select **Upload Application**. The Application Image Upload dialog is displayed.



4. Browse to the folder with your compiled application and select the .gbl file (see section 5.3.1 About Bootloading for more information).

If you compiled the image with GCC, the files are in <folder on General tab>\GNU ARM v.n.n - Default.  
If you compiled the image with IAR EWARM, the files are in <folder on General tab>\IAR ARM - <qualifier>.



5. If you have not already loaded a bootloader, check **Bootloader image**, then browse to the folder containing a prebuilt bootloader image. Images are located in the Simplicity Studio platform bootloader folder under sample apps. In this case open the 'SPI Flash Single' folder, for example:

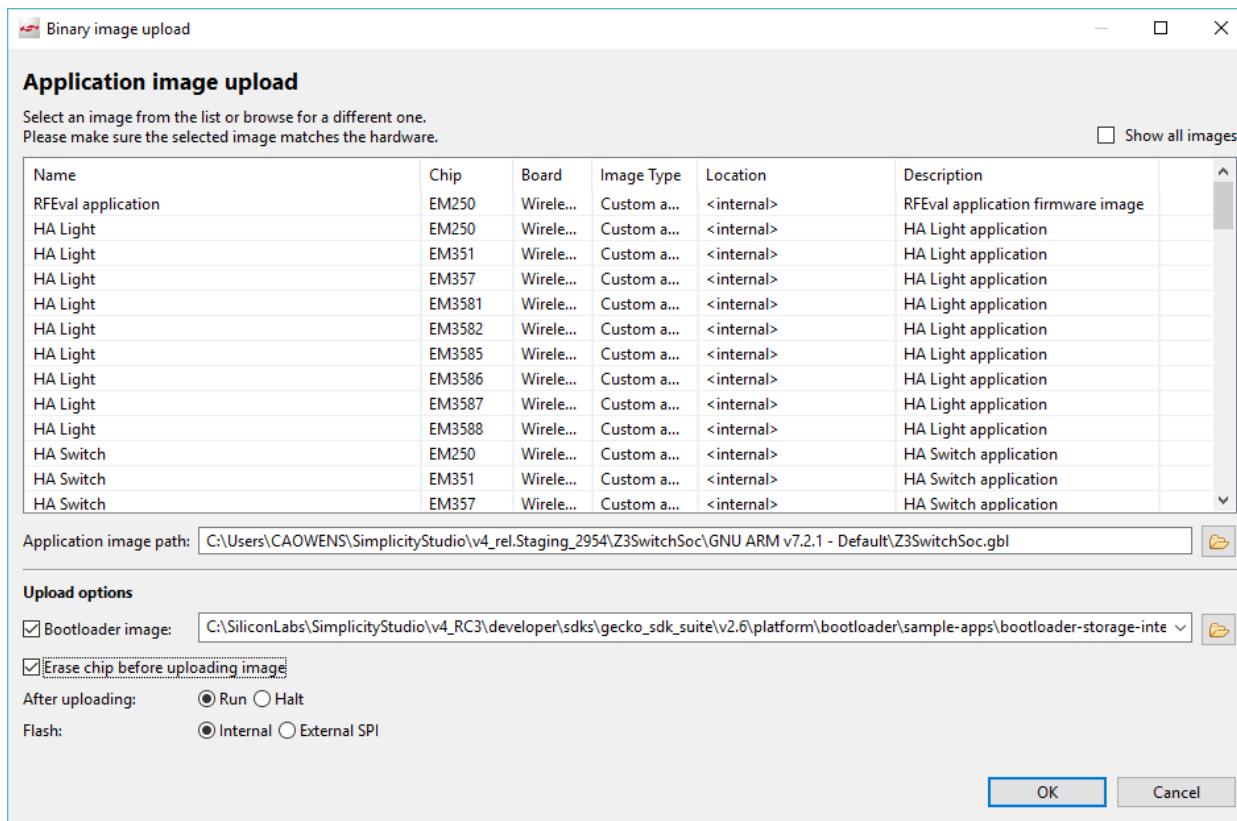
C:\SiliconLabs\SimplicityStudio\v4\developer\sdks\gecko\_sdk\_suite\<version>\platform\bootloader\sample-apps\bootloader-storage-spiflash-single

Open the folder that corresponds to your board and part number and select the .s37 file, for example:

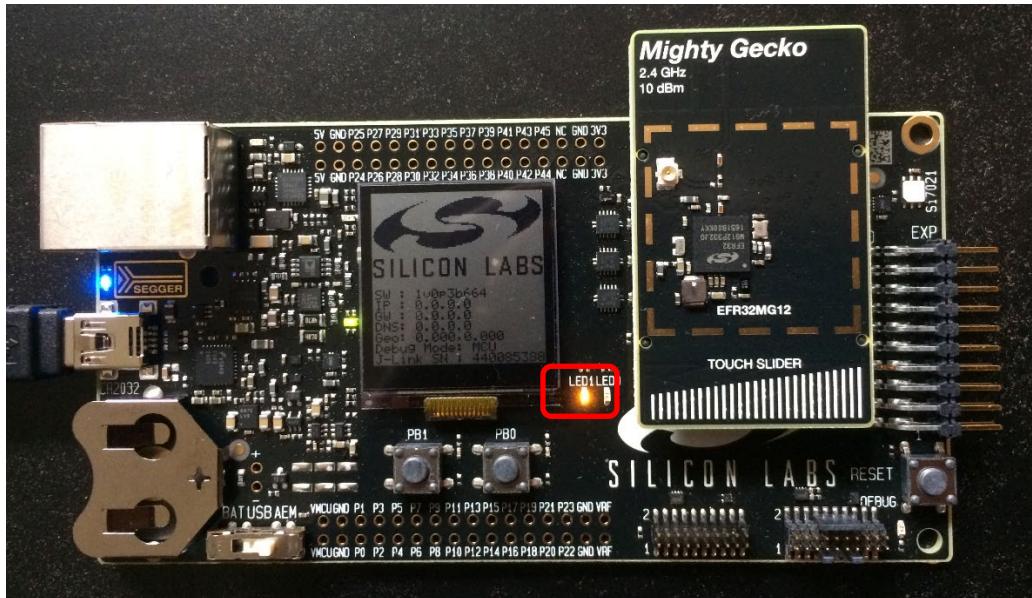
\efr32mg12p332f1024gl125-brd4162a\bootloader-storage-spiflash-single-combined.s37

6. Check **Erase Chip**, to make sure that the main flash block is erased before your new image is uploaded. New users will typically always check this.
- The **After uploading** options are **Run** (begin executing the code immediately) and **Halt** (wait for an event, such as a debugger to connect or manual initiation of a boot sequence). During initial development you will typically leave this set to **Run**.
  - The Flash options determine the storage location, and are **Internal** and **External SPI**. Leave the option set to **Internal**.

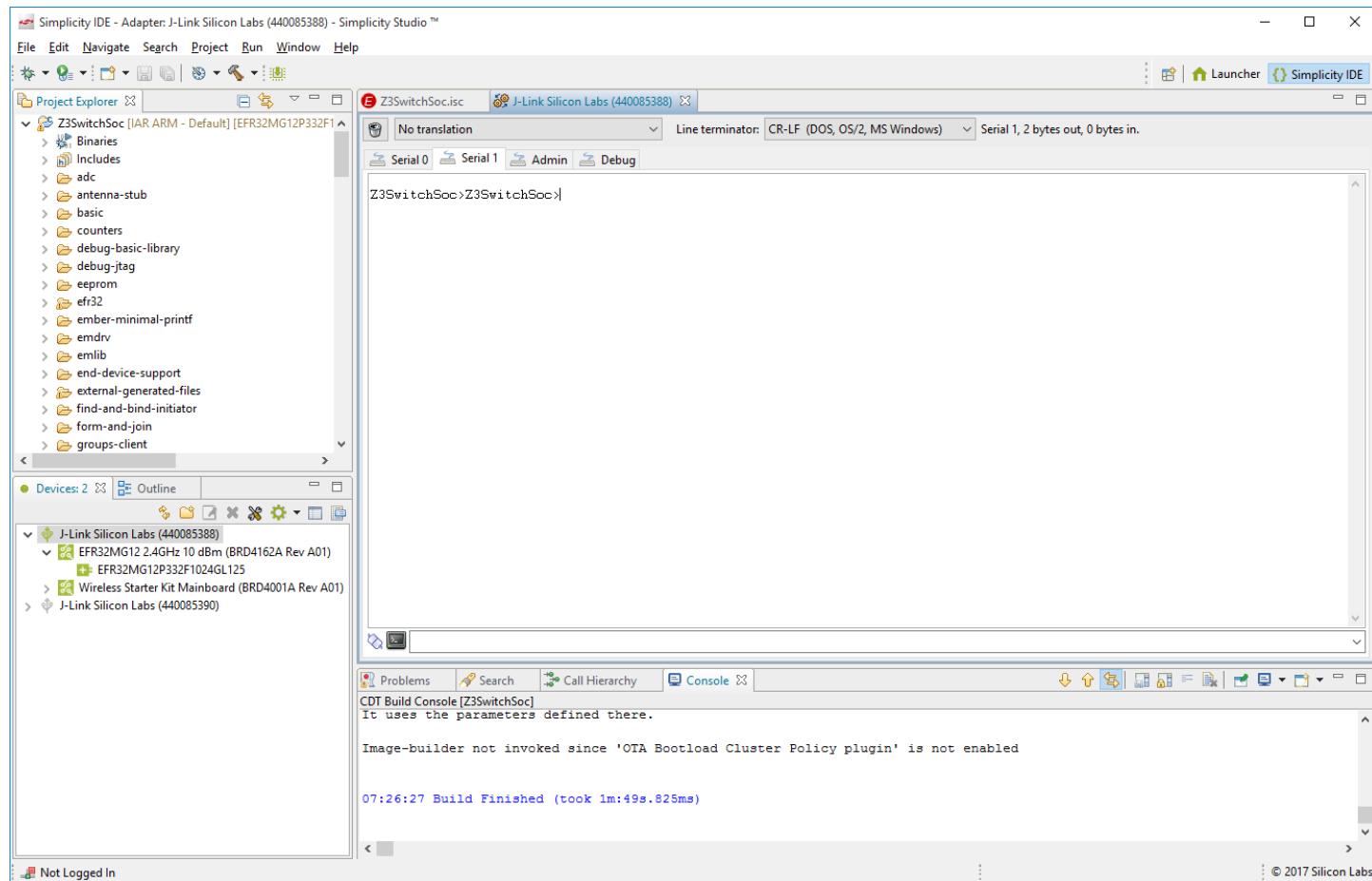
Your completed dialog should resemble the following:



7. Click **OK**. Load progress is shown in the lower right. When the load progress clears, on the WSTK you know that the application has loaded if LED1 is blinking slowly on and off.



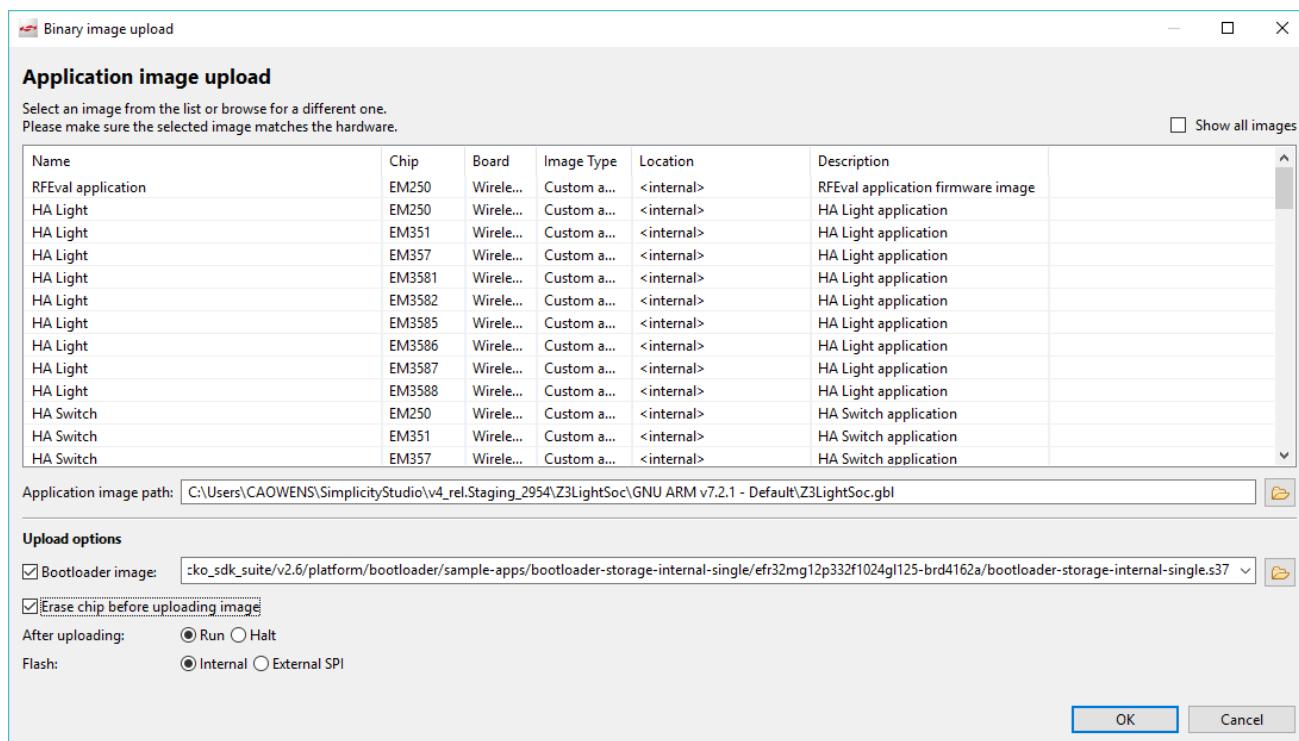
8. You can also right-click on the device and select **Launch Console**. In the console window, you will see four tabs: Serial 0 (the Virtual UART interface), Serial 1 (the physical UART interface), Admin (where you can configure the debug adapter such as ISA3 for EM35x or WSTK for EFR32), and Debug (where you see raw binary data over the debug interface). Click the Serial 1 tab, and press enter. You should see a prompt that corresponds to the project name. Note that the icons next to the device are now green, indicating a serial connection to the console.



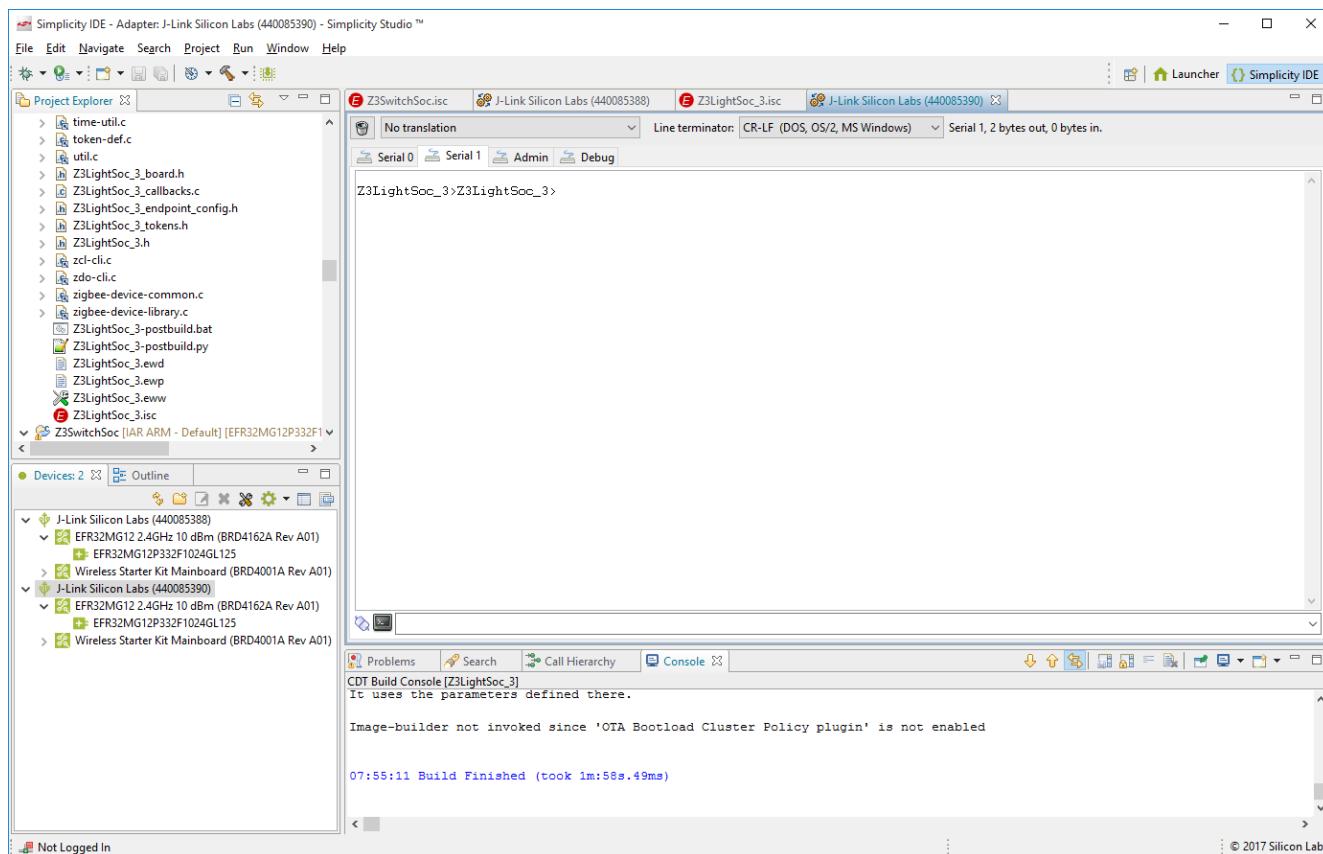
**Note:** Before you can load a different application, you must disconnect from the console. Right-click the device and select **Disconnect**.

Now repeat the procedure for the Z3Light example, by clicking Launcher in the upper right, and following the steps beginning in section [5.1 Selecting an Example Application](#).

Since Simplicity Studio remembers your last download location and you are using identical parts for the light and switch devices, it is more efficient to check and select the bootloader image first, and then browse to your application.



Z3Light also blinks LED1, and after you launch the console you can see the name of the project in a prompt on the Serial1 tab of a connected console.



## 6 Creating a Network

Once you have downloaded both the light and switch applications to different devices, you can create a network. Make sure that the switch device is close to the light device.

When the light application boots it immediately forms its own distributed network and advertises itself as a find-and-bind target. Go back to the switch console so you can observe the events. On the switch device, press button 1 to initiate commissioning. You'll see the process occur and then an indicator that commissioning has been successful. The results should indicate 0x00 (success), not 0x01 (failure).

The screenshot shows the Simplicity IDE interface with the following details:

- Project Explorer:** Shows files like time-util.c, token-def.c, util.c, Z3LightSoc\_3\_board.h, Z3LightSoc\_3\_callbacks.c, Z3LightSoc\_3\_endpoint\_config.h, Z3LightSoc\_3\_tokens.h, Z3LightSoc\_3.h, zcl-clc.c, zdo-clc.c, zigbee-device-common.c, zigbee-device-library.c, Z3LightSoc\_3\_postbuild.bat, Z3LightSoc\_3\_postbuild.py, Z3LightSoc\_3\_evd, Z3LightSoc\_3\_evp, Z3LightSoc\_3\_ewv, Z3LightSoc\_3\_isc.
- Devices:** A list of connected devices:
  - J-Link Silicon Labs (440085388) with EFR32MG12 2.4GHz 10 dBm (BRD4162A Rev A01) and Wireless Starter Kit Mainboard (BRD4001A Rev A01) listed.
  - J-Link Silicon Labs (440085390) with EFR32MG12 2.4GHz 10 dBm (BRD4162A Rev A01) and Wireless Starter Kit Mainboard (BRD4001A Rev A01) listed.
- Serial Port Output:** Displays the following log entries, with two lines highlighted with red boxes:
 

```
Z3SwitchSoc>Z3SwitchSoc>Touchlink network start: 0x000
T00000000:Inter-PAN TX (8) [11 00 02 07 C7 8E FD 00 ], 0x00
T00000000:Inter-PAN TX (9) [11 01 06 07 C7 8E FD FF FF ], 0x00
RX inter-PAN message (unicast):
src pan id: 0xDODF
src long id: (>)000B57FFFE315B93
profile id: 0xC05E
cluster id: 0x1000 (ZLL Commissioning)
T00000000:Inter-PAN TX (9) [11 02 06 07 C7 8E FD 00 00 ], 0x00
EMBER_NETWORK_DOWN
EMBER_NETWORK_UP 0x0001
Touchlink network complete: 0x00
Find and bind initiator start: 0x00
Processing message: len=3 profile=0104 cluster=0003
T00000000:RX len 3, ep FF, clus 0x0003 (Identify) FC 01 seq 03 cmd 01 payload[]
Next poll nwk 0 in 30000ms
Processing message: len=12 profile=0000 cluster=0013
RX ZDO command 0x0013, status: 0x00
Device Announce 0x0001
poll nwk 0 no data
Processing message: len=5 profile=0104 cluster=0003
T00000000:RX len 5, ep 01, clus 0x0003 (Identify) FC 09 seq 03 cmd 00 payload[B3 00 ]
T00000000:TX (resp) Ucast 0x00
TX buffer: [00 03 0B 00 00 ]
Processing message: len=12 profile=0000 cluster=8001
Processing message: len=28 profile=0000 cluster=8004
Find and bind initiator complete: 0x00
```
- Console:** Shows the message "Image-builder not invoked since 'OTA Bootload Cluster Policy plugin' is not enabled".
- Build Status:** Shows "07:55:11 Build Finished (took 1m:58s.49ms)".

Note: The light only advertises for a three minute window, so if you got interrupted and then commissioning fails, try pressing any button on the light device to re-start advertising.

Now, if you press button 0 on the switch, you'll notice that LED0 on the light toggles on and off, and you can see the activity on both the switch console and the light console.

```

Simplicity Studio ™
Ip

Z3SwitchSoc.isc J-Link Silicon Labs (440085388) Z3LightSoc_3.isc J-Link Silicon Labs (440085390)
Serial No translation Line terminator: CR-LF (DOS, OS/2, MS Windows) Serial 1, 2 bytes out, 0 bytes in.

Serial 0 Serial 1 Admin Debug

src pan id: 0xD0DF
src long id: (0)000B57FFFE315B93
profile id: 0xC0SE
cluster id: 0x1000 (ZLL Commissioning)
T00000000:Inter-PAN TX (9) [11 02 06 07 C7 8E FD 00 00 ] 1, 0x00
EMBER_NETWORK_DOWN
EMBER_NETWORK_UP 0x0001
Touchlink network complete: 0x00
Find and bind initiator start: 0x00
Processing message: len=3 profile=0104 cluster=0003
T00000000:RX len 3, ep FF, clus 0x0003 (Identify) FC 01 seq 03 cmd 01 payload[]
Next poll nwk 0 in 30000 ms
Processing message: len=12 profile=0000 cluster=0013
RX: ZDO, command 0x0013, status: 0x00
Device Announce: 0x0001
poll nwk 0: no data
Processing message: len=5 profile=0104 cluster=0003
T00000000:RX len 5, ep 01, clus 0x0003 (Identify) FC 09 seq 03 cmd 00 payload[B3 00 ]
T00000000:TX (resp) Ucast 0x00
TX buffer: [00 03 0B 00 00 ]
Processing message: len=12 profile=0000 cluster=8001
Processing message: len=25 profile=0000 cluster=8004
Find and bind initiator complete: 0x00
Send to bindings: 0x00
Processing message: len=5 profile=0104 cluster=0006
T00000000:RX len 5, ep 01, clus 0x0006 (On/off) FC 08 seq 04 cmd 0B payload[02 00 ]
Send to bindings: 0x00
Processing message: len=5 profile=0104 cluster=0006
T00000000:RX len 5, ep 01, clus 0x0006 (On/off) FC 08 seq 05 cmd 0B payload[02 00 ]

```

```

Simplicity Studio ™
Ip

Z3SwitchSoc.isc J-Link Silicon Labs (440085388) Z3LightSoc_3.isc J-Link Silicon Labs (440085390)
Serial No translation Line terminator: CR-LF (DOS, OS/2, MS Windows) Serial 1, 2 bytes out, 0 bytes in.

Serial 0 Serial 1 Admin Debug

cluster id: 0x1000 (ZLL Commissioning)
T00000000:Inter-PAN TX (58) [19 00 03 07 C7 8E FD 03 00 03 93 5B 31 FE FF 57 0B 00 01 5E C0 01 01 00 00 00 93 5B 31 FE FF
57 0B 00 02 5E C0 00 02 00 01 93 5B 31 FE FF 57 0B 00 F2 5E C0 66 00 00 00 02 ]. 0x00
RX inter-PAN message (unicast)
src pan id: 0x8899
src long id: (0)000B57FFFE315B89
profile id: 0xC0SE
cluster id: 0x1000 (ZLL Commissioning)
RX inter-PAN message (unicast)
src pan id: 0x8899
src long id: (0)000B57FFFE315B89
profile id: 0xC0SE
cluster id: 0x1000 (ZLL Commissioning)
EMBER_NETWORK_DOWN
EMBER_NETWORK_UP 0x0002
Open network for joining: 0x00
join for 254 sec: 0x00
NWK Creator Security: Open network: 0x00
Find and Bind Target: Start target: 0x00
Find and bind target start: 0x00
RX: ZDO, command 0x0013, status: 0x00
Device Announce: 0x0002
T00000000:RX len 3, ep FF, clus 0x0003 (Identify) FC 01 seq 03 cmd 01 payload[]
RX: ZDO, command 0x0013, status: 0x00
Device Announce: 0x0001
T00000000:RX len 5, ep 01, clus 0x0003 (Identify) FC 00 seq 03 cmd 0B payload[00 00 ]
T00000000:RX len 3, ep 01, clus 0x0006 (On/off) FC 01 seq 04 cmd 02 payload[]
T00000000:RX len 3, ep 01, clus 0x0006 (On/off) FC 01 seq 05 cmd 02 payload[]
join for 40 sec: 0x00
NWK Creator Security: Open network: 0x00

```

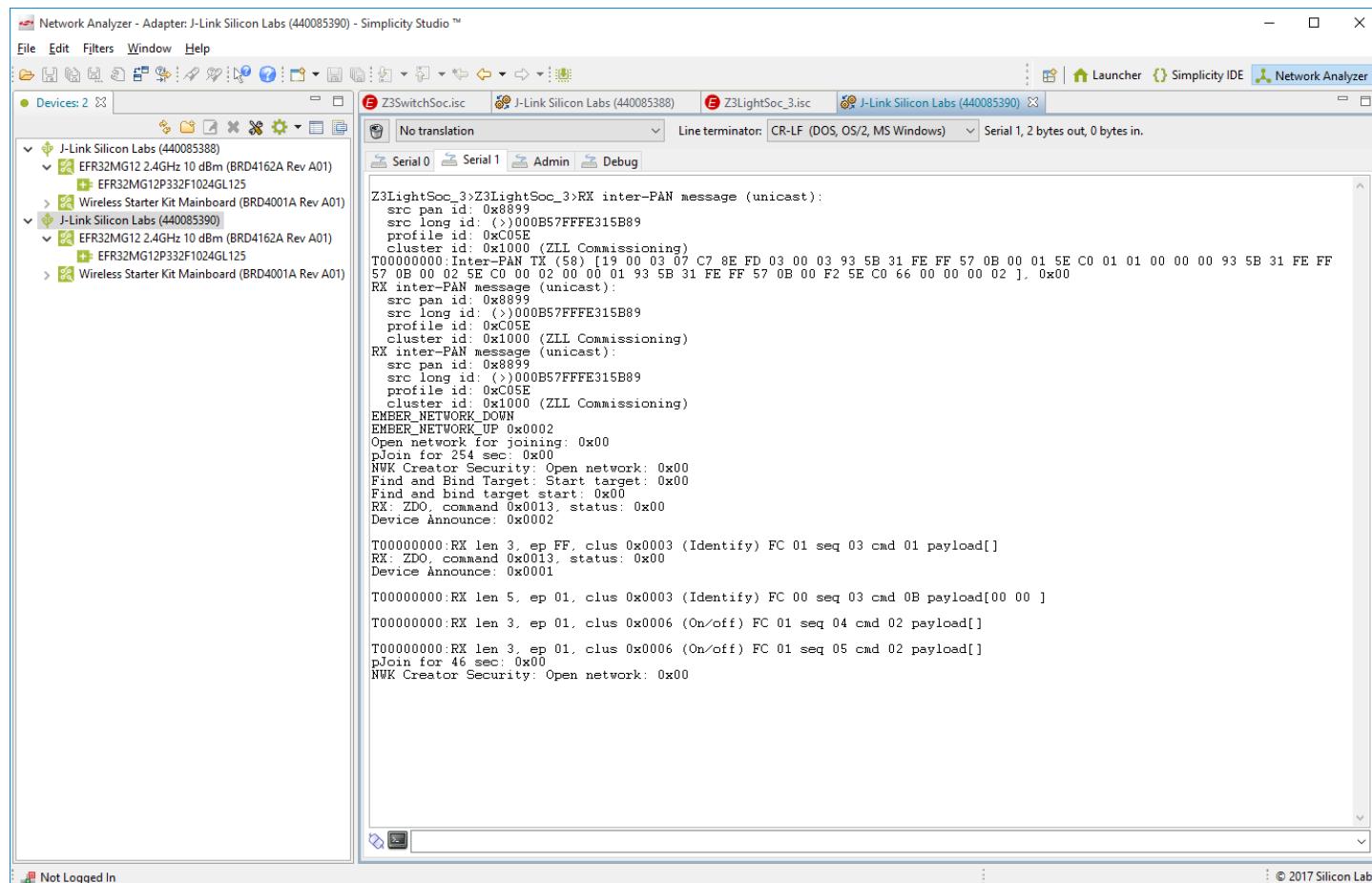
you can see a message such as the following, indicating you are sending an On/Off toggle command to the light:

T00000000:RX len 5, ep 01, clus 0x0006 (On/off) FC 08 seq 0C cmd 0B payload[02 00 ]

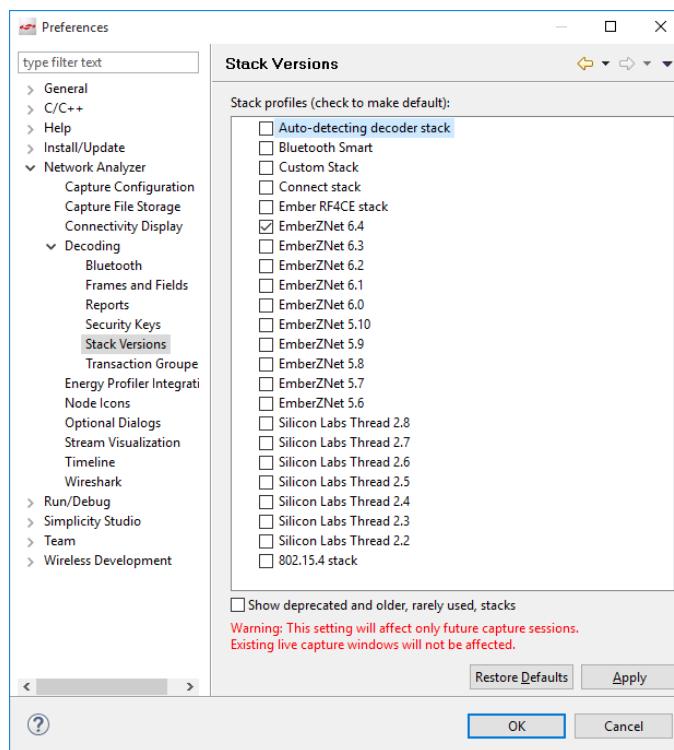
## 7 Using the Network Analyzer

Now that your network is set up, you can evaluate the data being transmitted using the Network Analyzer tool. Network Analyzer helps you debug network connectivity by displaying radio packets and certain debug interface events in a format that is easy to visualize and analyze.

1. Click the Launcher button in the upper right and select Network Analyzer from the Tools menu. The Network Analyzer opens with your console window(s) still displaying data.



2. Make sure that Network Analyzer is set to decode the correct protocol. Select **Window > Preferences > Network Analyzer > Decoding > Stack Versions**, and verify it is set correctly. If you need to change it, click the correct stack, click **Apply**, and then **OK**.



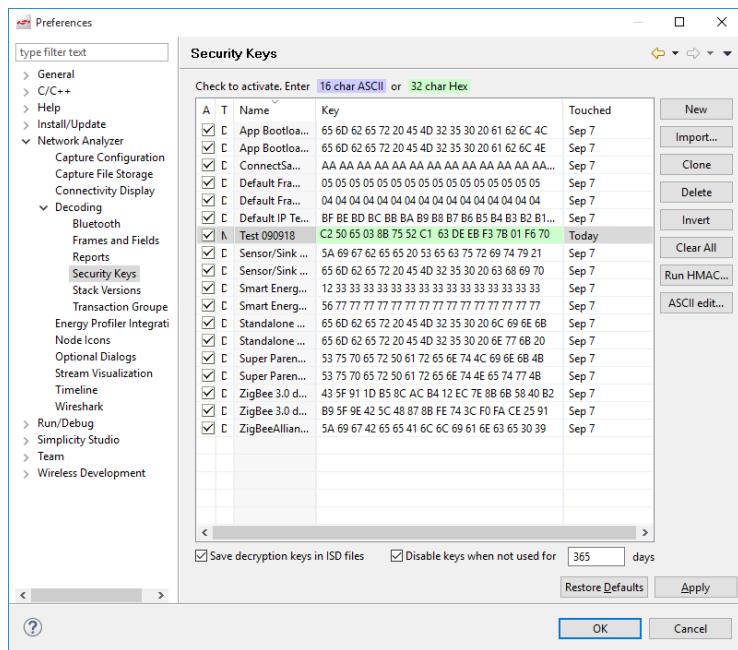
3. To make sure that packets decode correctly, manually enter the NWK key. In either the Switch or Light console window, type the following, being sure to include the 's' in 'keys':

```
Keys print
```

4. In the information returned, find the network key and copy it:

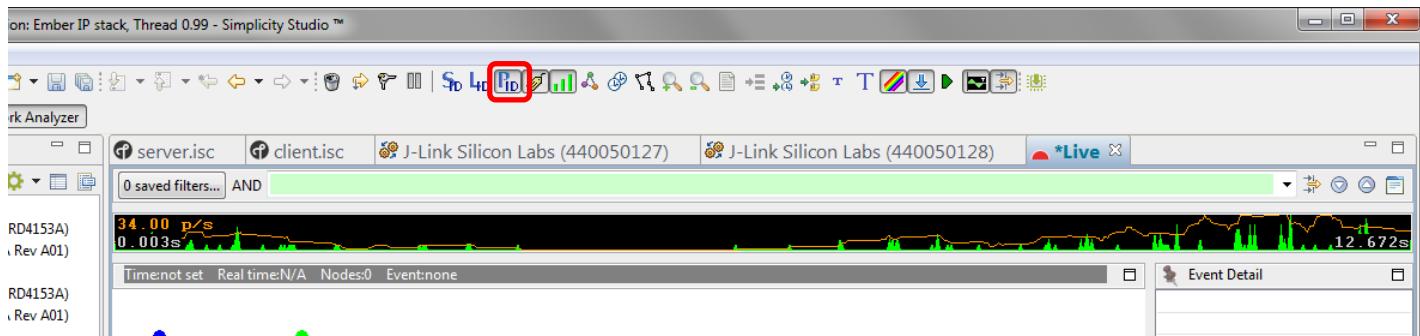
```
NWK Key: EF DE 0C 69 5B 72 6E C4 41 27 C6 E6 F1 36 26 26
```

5. In Window > Preferences, open Network Analyzer > Decoding > Security Keys, click **New**, name the new entry, and paste the copied key into it. Click **Apply**. Click **OK** to leave.



6. Right-click on the light or the switch device, and select **Start Capture**. Do the same for the other device.  
 7. If you are in an environment with a number of wireless devices, you may have a very noisy Network Analyzer environment, as reflected both in the event traffic and in the map. To show additional information in the map, click on the map.

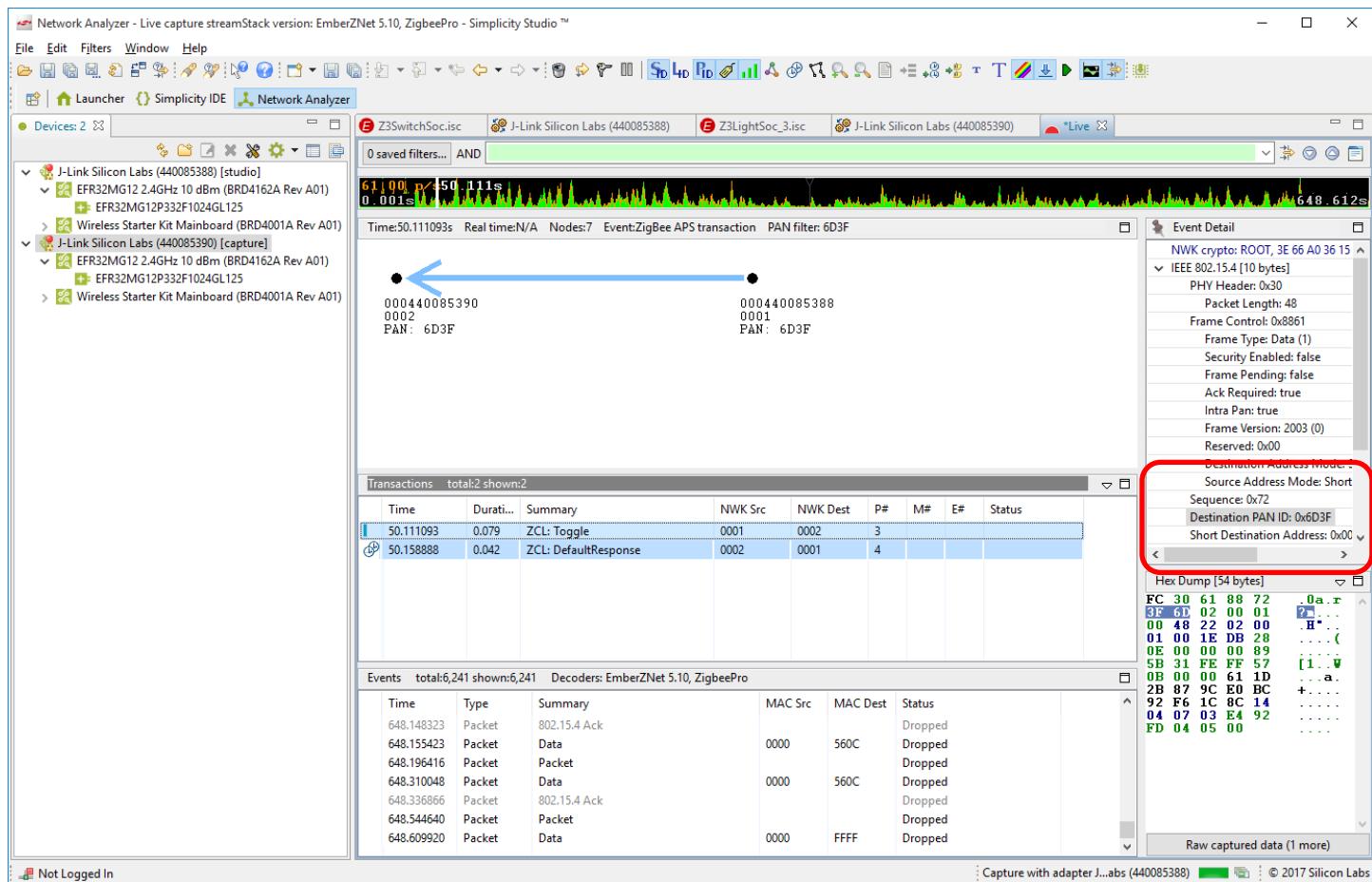
On the toolbar, click the PAN ID button:



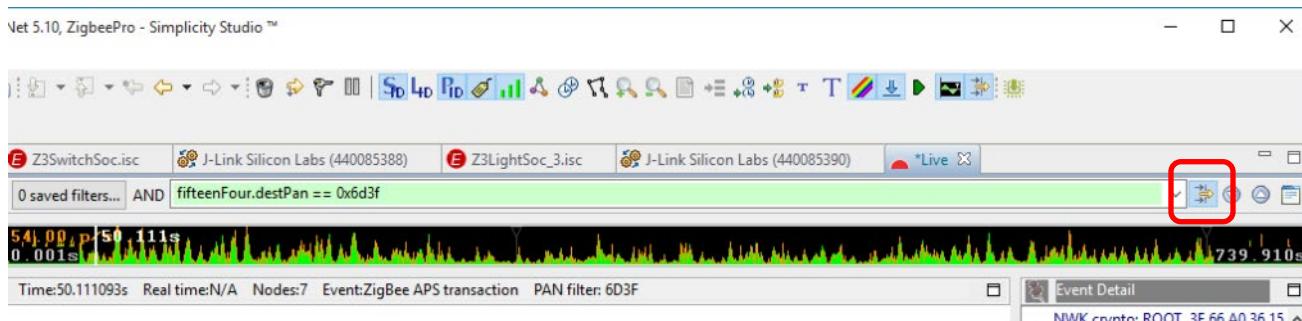
8. Right-click on the representation of your Switch device (the dot that has the same ISA3 adapter name or WSTK name or J-Link serial number as the device) and select **Show only this PAN**.

To filter transactions:

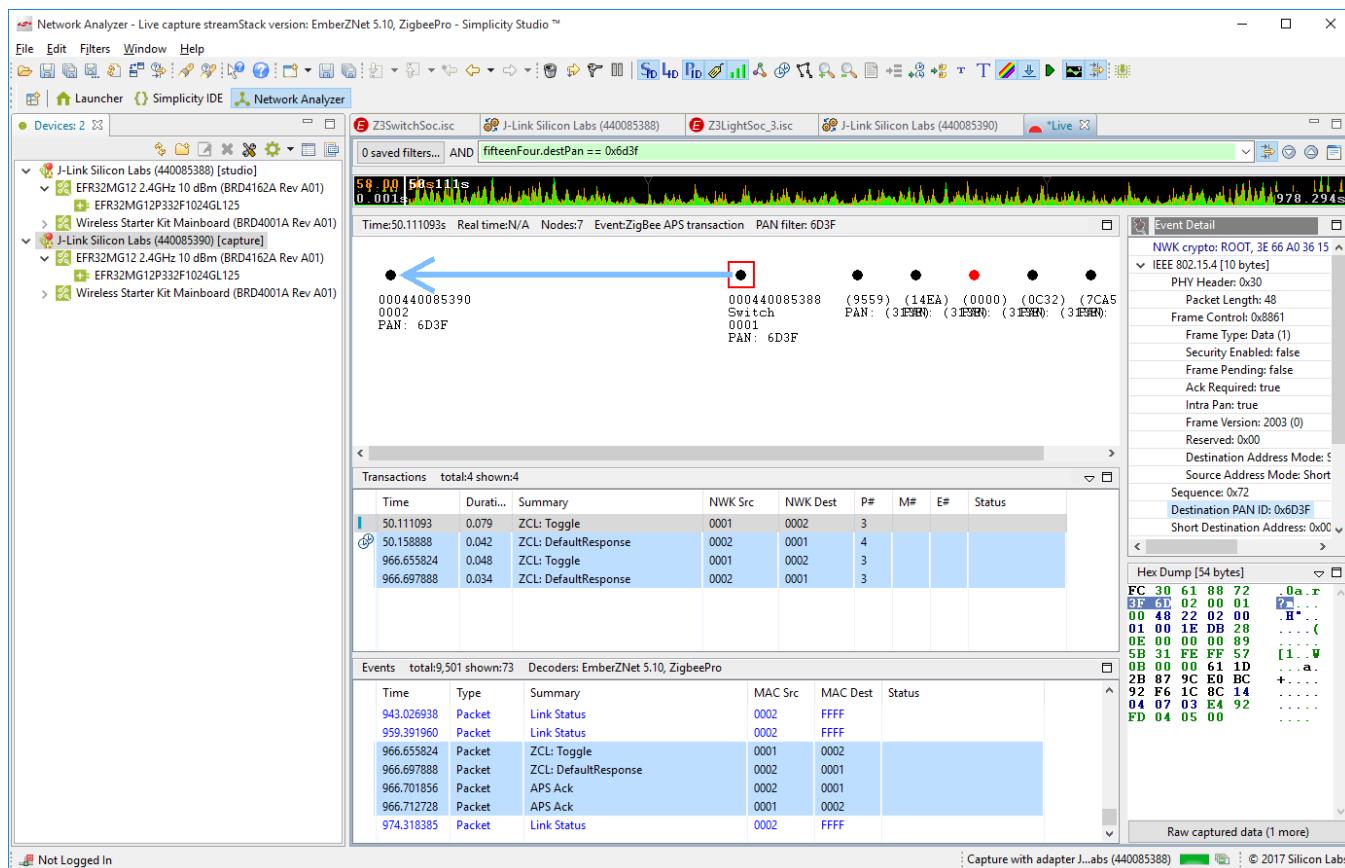
1. Press button 0 on the switch to get a complete transaction (in blue).



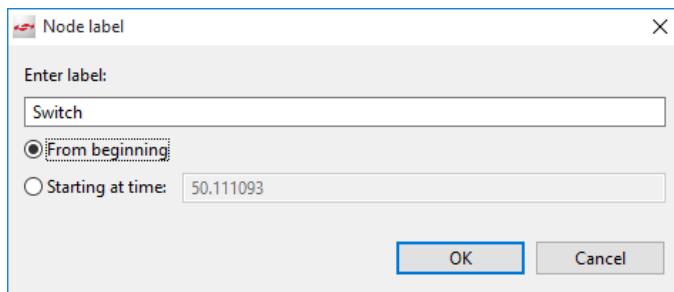
2. Click on one of the blue transactions.
3. In the Event Detail, expand IEEE802.15.4 and scroll down until you see the Destination PAN ID
4. Right-click on it, and select **Add to filter**.
5. Apply the filter by clicking next to the green filter expression field.



Now when you press button 0 you can clearly see each event associated with the transaction.



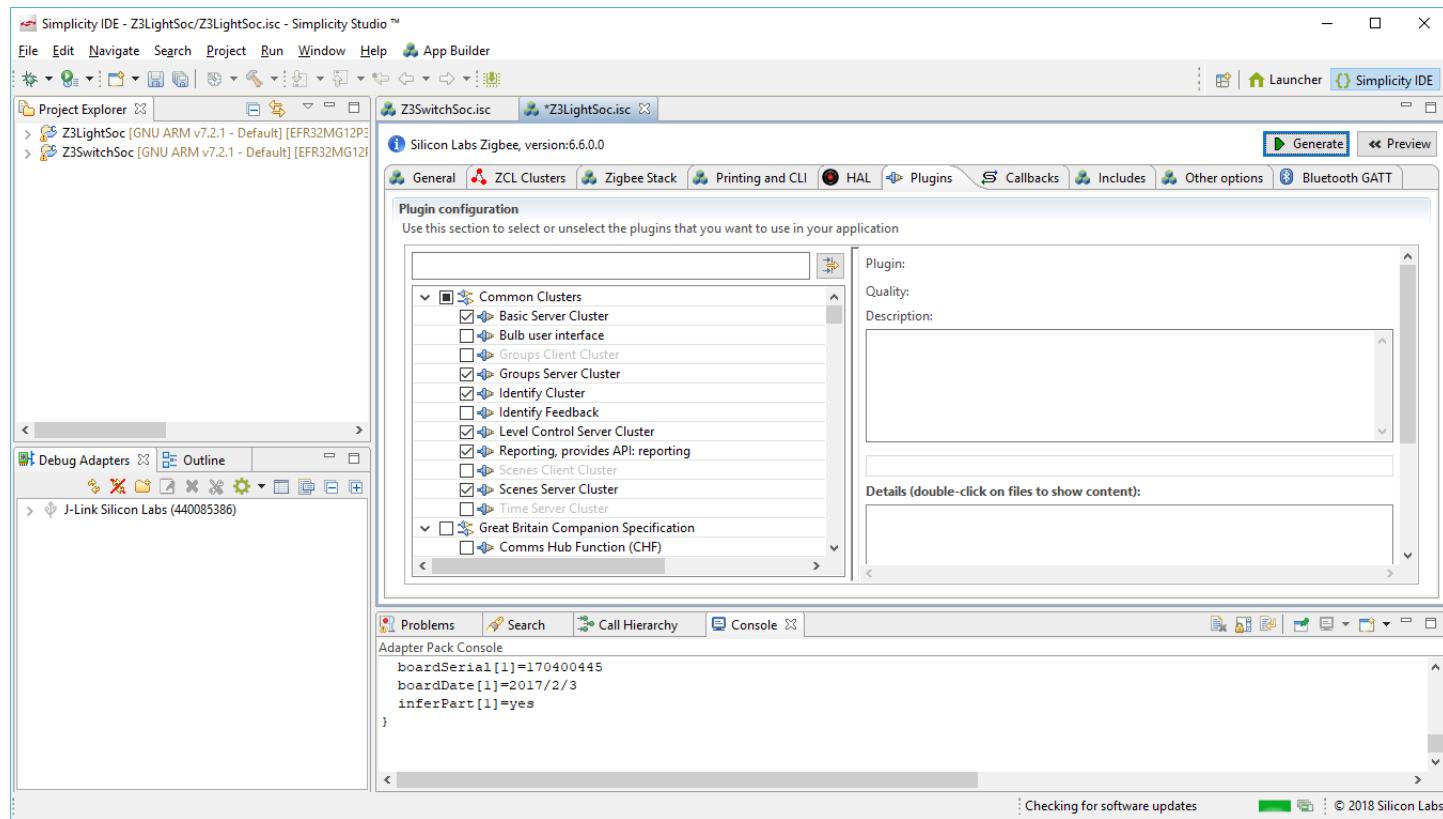
When analyzing more complex networks, you can drag and reposition the items shown in the map. By right-clicking on a device, you can also show connectivity and add labels. Labelling is useful not only in map, but also in the log. To label the full log, click **From beginning**.



## 8 Next Steps

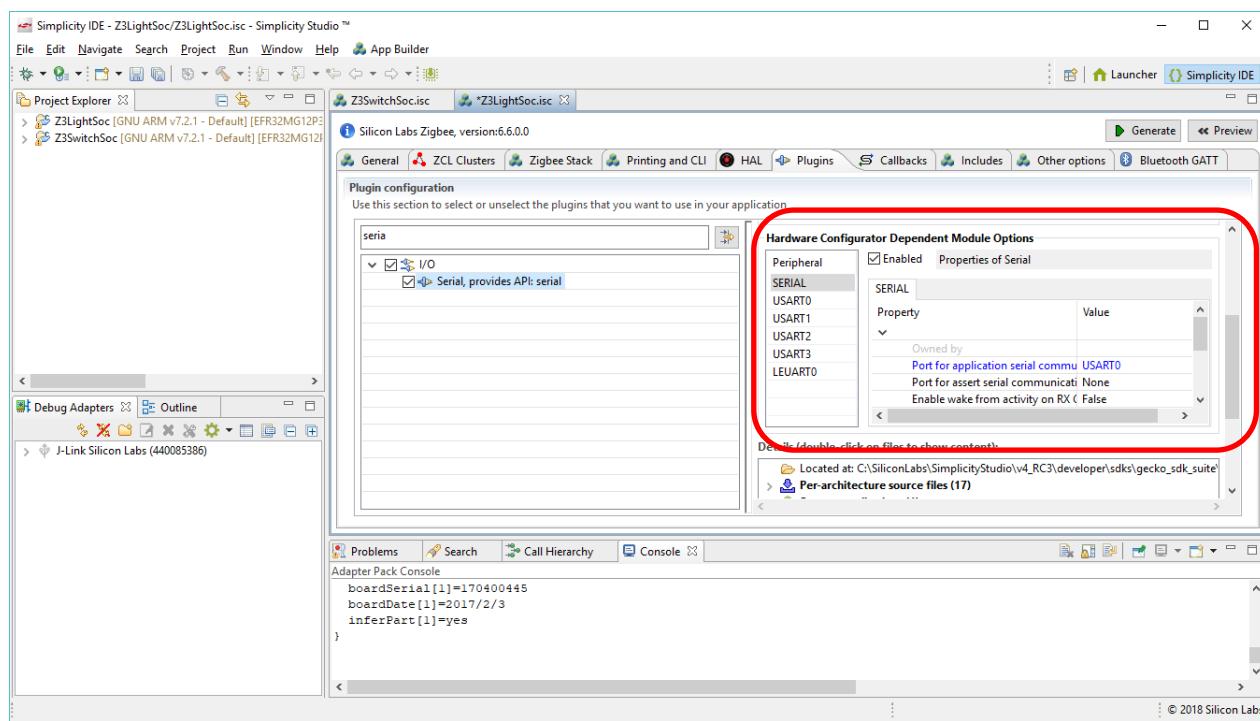
### 8.1 Example Applications

Explore configuring the example application to meet your needs. Much of the software configuration can be done through plugins. Select a plugin to see more information about it and its configuration options, if any.

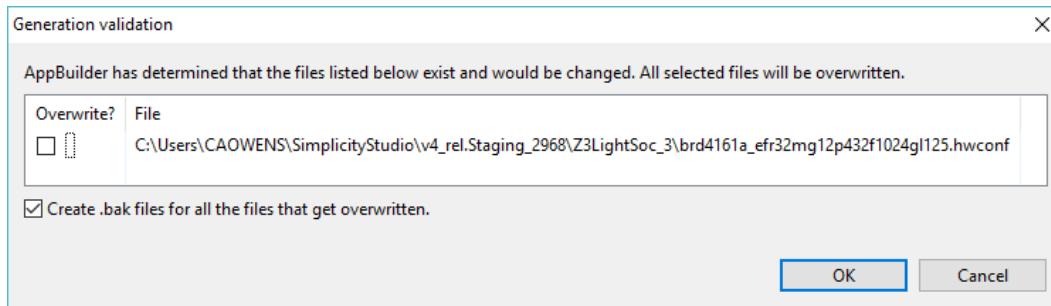


## 8.2 Hardware Configurator

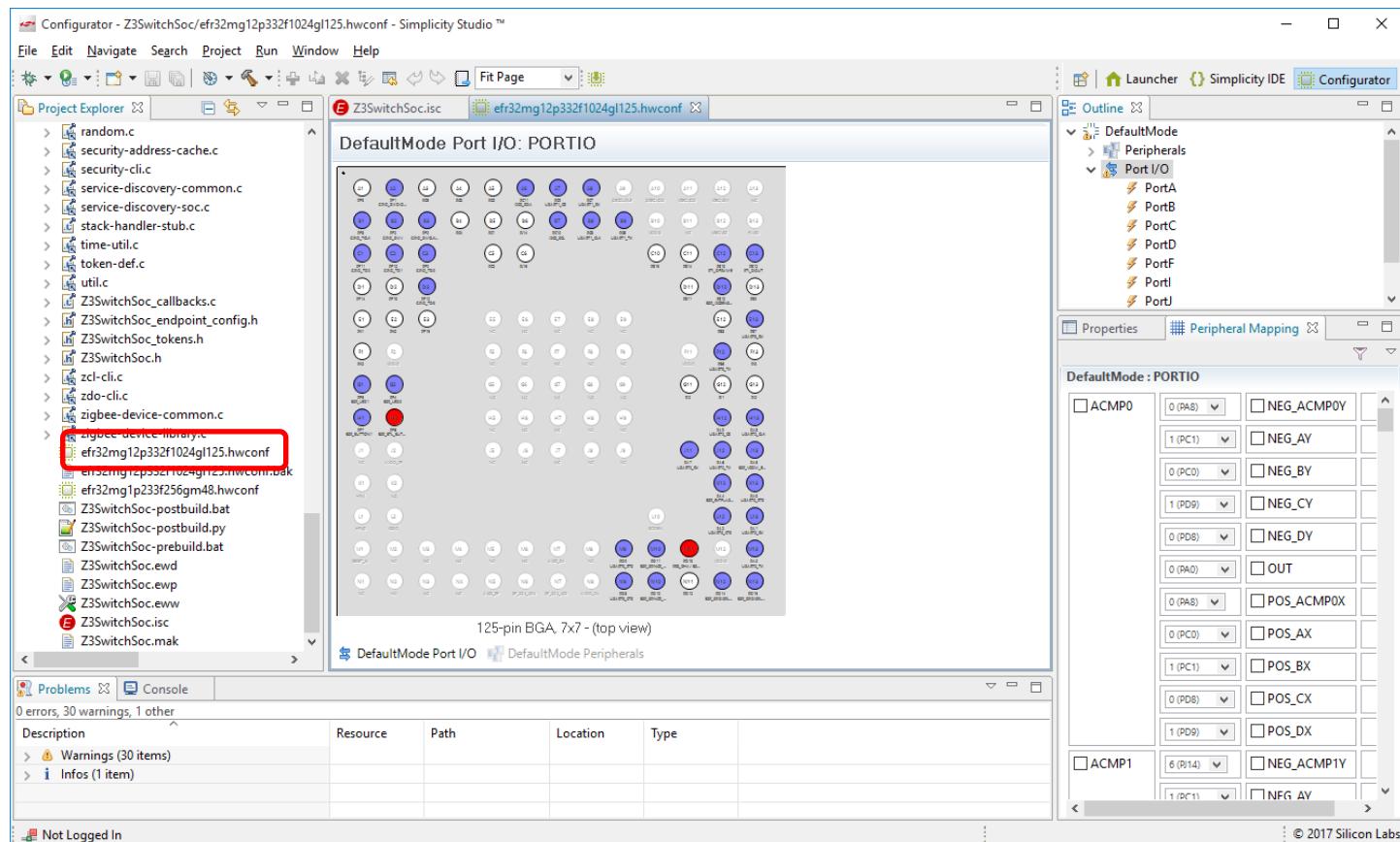
Simplicity Studio offers a Hardware Configurator tool that allows you to easily configure new peripherals or change the properties of existing ones. Hardware Configurator options are available on many of the HAL and I/O plugins.



**Note:** If you change hardware configuration options your changes are saved to a temporary file. In order to have the changes included in the project, you must check the Overwrite checkbox in the following dialog, displayed at the end of file generation.



You can also open the Hardware Configurator tool by double-clicking on the .hwconf file that was generated along with your other project files or by clicking **Open Hardware Configurator** on the HAL tab (scroll down to the bottom of the tab). Some configuration options are only available through the Hardware Configurator tool. Click **DefaultMode Peripherals** under the pin graphic to change to a peripherals view.



See [AN1115: Configuring Peripherals for 32-Bit Devices in Simplicity Studio](#) for more information about the Hardware Configurator both within the Simplicity IDE and as a separate tool.

### 8.3 Energy Profiler

You may also want to explore using the Energy Profiler tool. Energy Profiler provides energy debugging capability by displaying graphical real-time energy consumption information. This can be particularly useful for developing a low-power application. See [UG343: Multi-Node Energy Profiler User's Guide](#) for more information.

**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](http://www.silabs.com/IoT)



**SW/HW**  
[www.silabs.com/simplicity](http://www.silabs.com/simplicity)



**Quality**  
[www.silabs.com/quality](http://www.silabs.com/quality)



**Support and Community**  
[community.silabs.com](http://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 Laboratories Inc.**  
400 West Cesar Chavez  
Austin, TX 78701  
USA

<http://www.silabs.com>