



QSG113: Getting Started with Silicon Labs Thread

This quick start guide provides basic information on configuring, building, and installing applications for the Mighty Gecko (EFR32MG) family of SoCs using the Silicon Labs Thread stack version 2.4 and higher and Simplicity Studio version 4.

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

KEY POINTS

- Product overview
- Setting up your development environment
- Installing Simplicity Studio and Silicon Labs Thread
- 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/thread/Pages/thread.aspx> for more information):
- 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 Silicon Labs Thread applications, you will need the following. Installation instructions are provided in the section **Install Simplicity Studio and the Silicon Labs Thread 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 Silicon Labs Thread stack, an advanced implementation of a wireless protocol stack, installed through Simplicity Studio. The stack API is documented in online API reference as well as other documents available through Simplicity Studio. 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.
- IAR Embedded Workbench for ARM (IAR-EWARM) 7.80 (optional). May be used as a compiler in the Simplicity Studio development environment as an alternative to GCC (The GNU Compiler Collection), which is provided with Simplicity Studio. GCC is used in this document.

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.

Download the supported version from the Silicon Labs Support Portal, as described at the end of section **Install Simplicity Studio and the Silicon Labs Thread 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 **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 **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 Silicon Labs Thread software.

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.

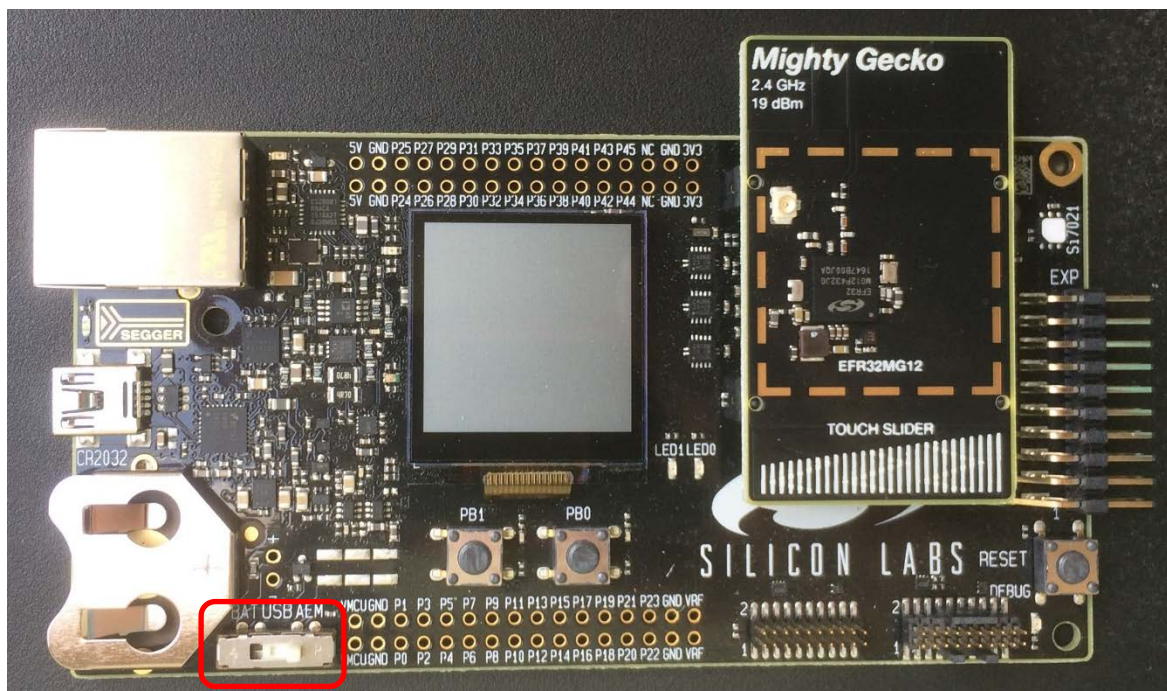


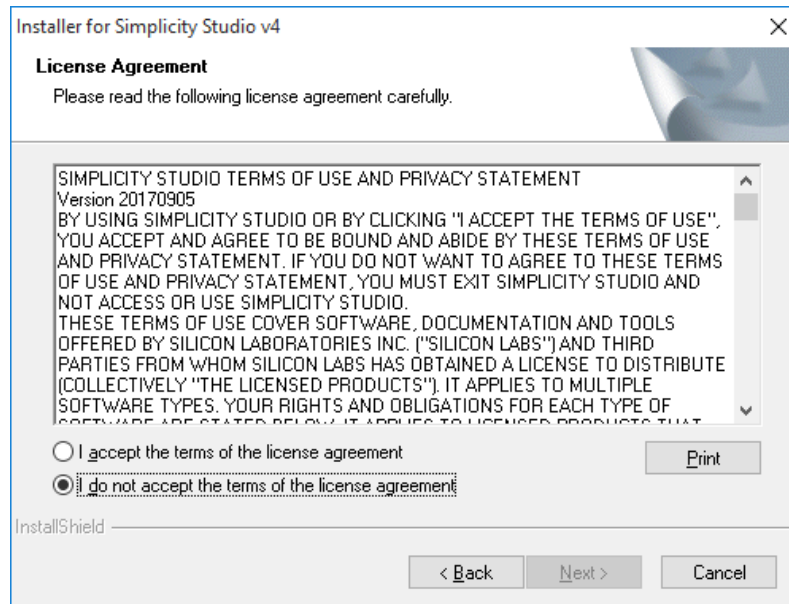
Figure 1. EFR32MG12 on a WSTK

2.2 Register your Development Kit

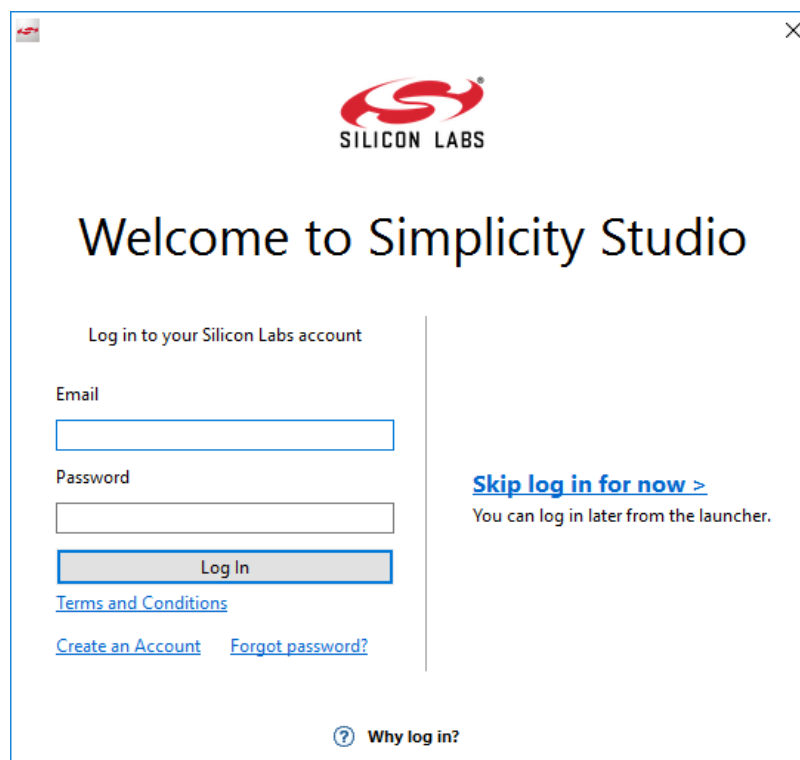
Before you install Simplicity Studio, you need to create an account on the support portal. Be sure to record your account username and password as you will use it to log in to Simplicity Studio. In order to install the Silicon Labs Thread stack from Simplicity Studio, you must also register your kit 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 Silicon Labs Thread Stack

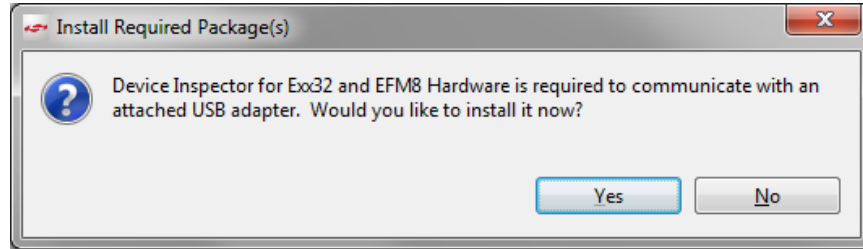
1. Run the Simplicity Studio installation application.
2. When the Simplicity Studio installer first launches, it presents a License Agreement dialog. Accept the terms of the agreement and click **Next >**.



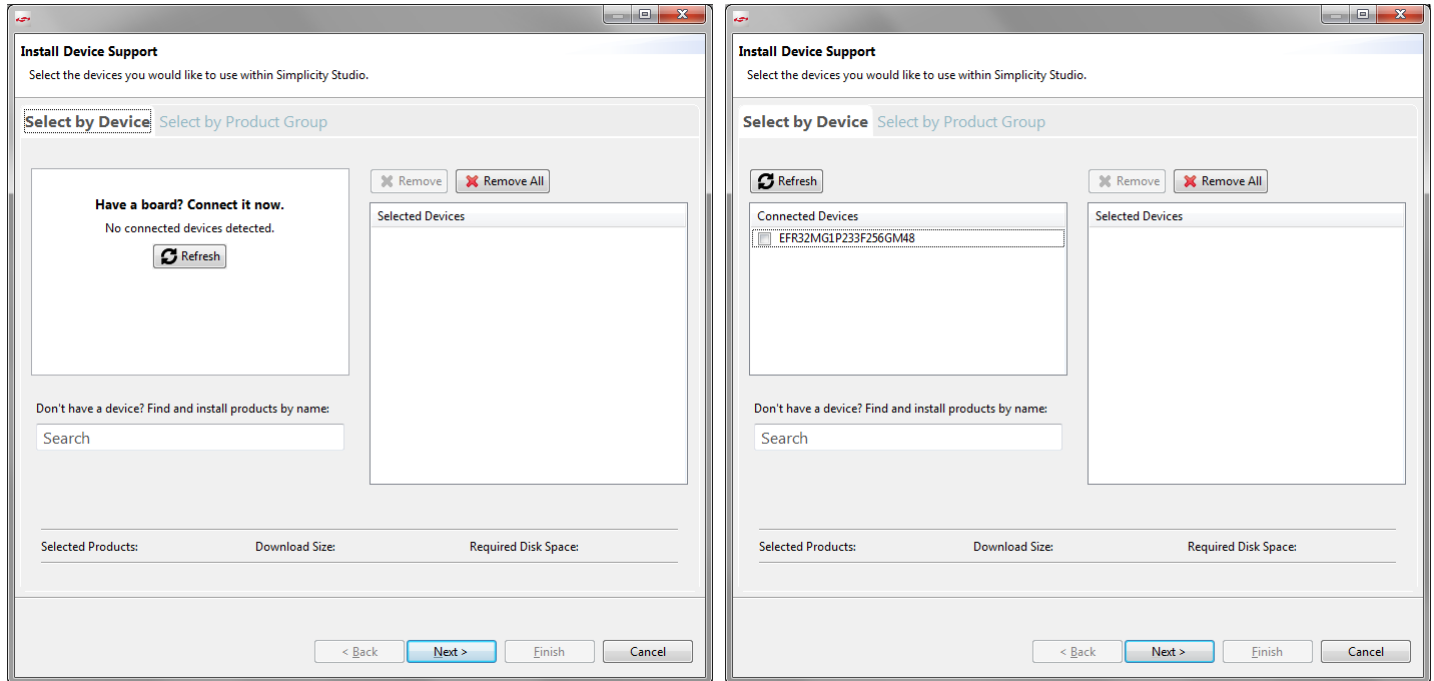
3. Choose a destination location, click **Next >** and then click Install.
4. When the application launches, you are invited to log in. Log in using your support account username and password. Although you can skip log in here, you must be logged in and have registered your development kit to download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



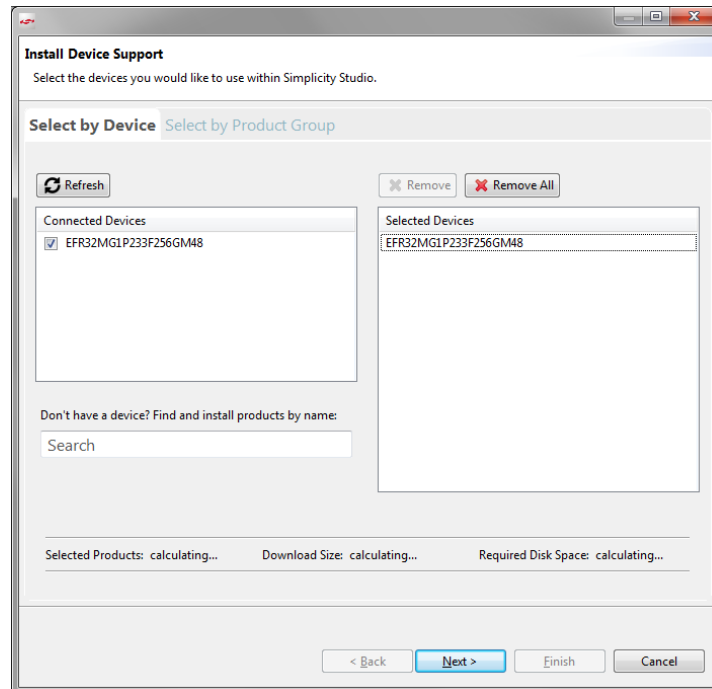
- After login, Simplicity Studio adds software information. Once initial software installation is complete, Simplicity Studio checks for connected hardware. If you have the WSTK connected by USB cable, Simplicity Studio will detect the USB cable and prompt you to download a Device Inspector. Click **Yes**.



- An Install Device Support dialog appears. After a short delay, it shows your connected device. If the connected device does not show, click **Refresh**. The following figure shows the Install Device Support dialog before and after the connected device is displayed.

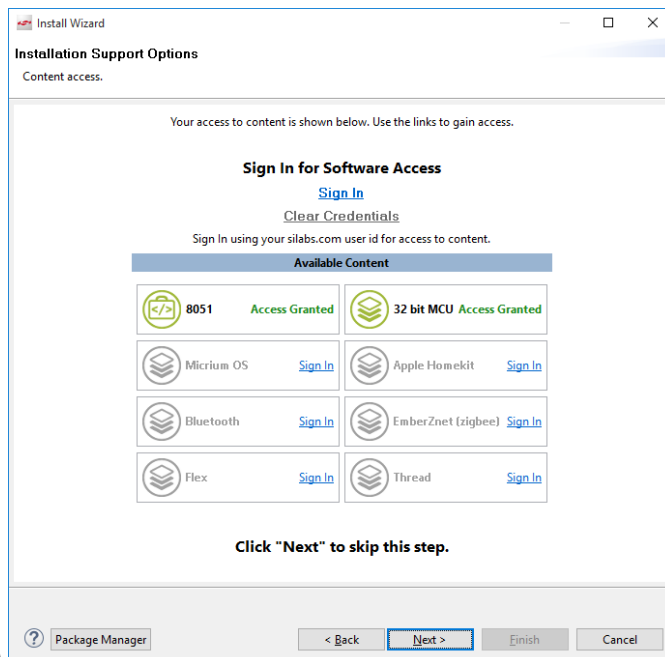


- Click the checkbox next to the device to select it. Selecting the device allows Simplicity Studio to present the relevant software packages for you to install. Click **Next**.

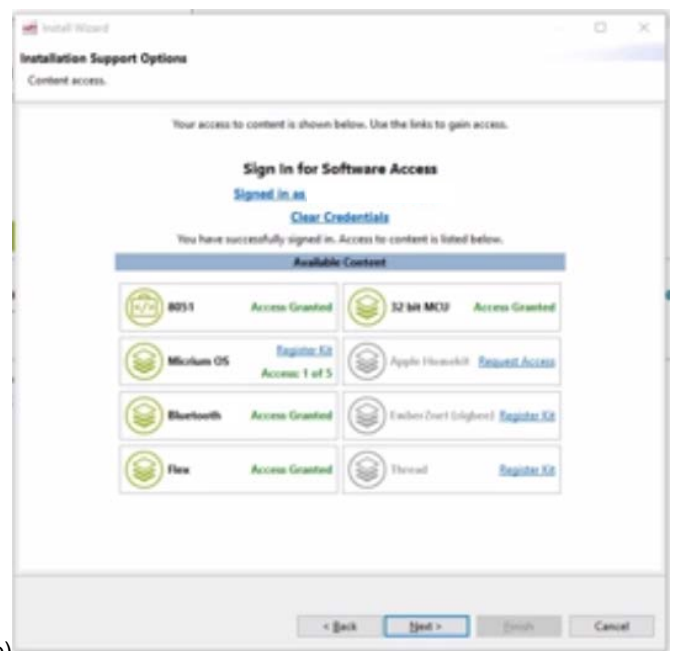


Note: You can also click the **Select by Product Group** tab to install device support for all devices in one or more product groups.

- 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 Silicon Labs Thread (see the following figure b). Click **Register Kit**.

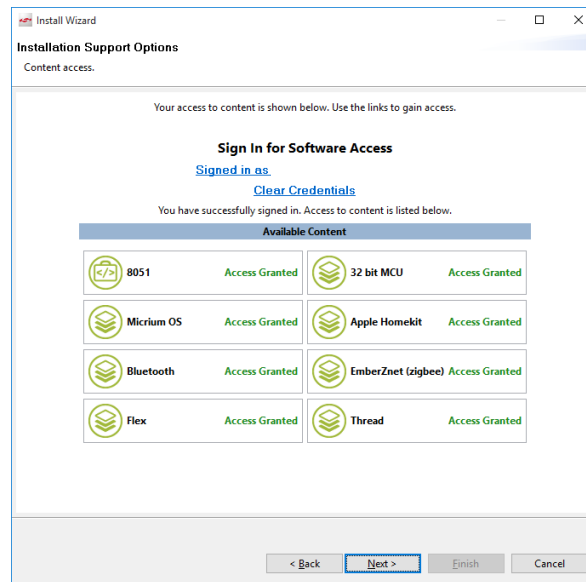


a)



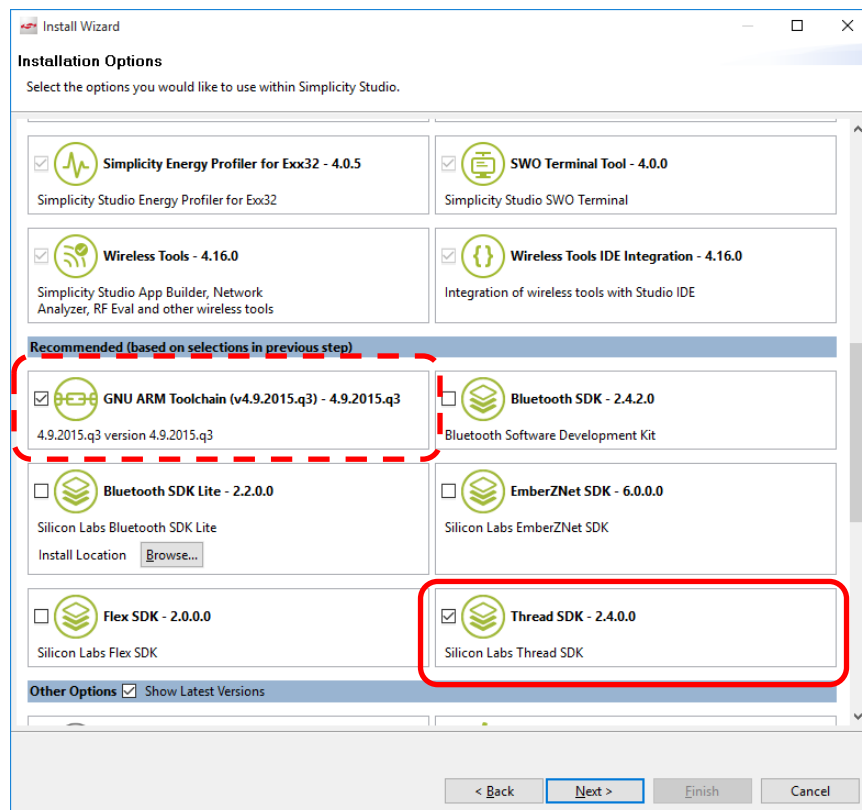
b)

9. If you have already signed in and registered your kit, and see that access is granted to Thread, click **Next**.

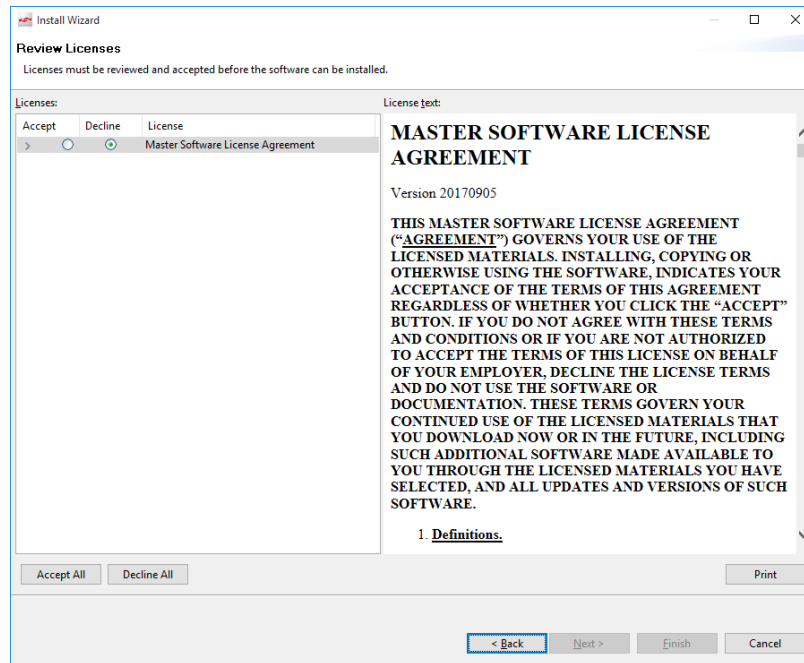


10. The **Installation Options** dialog shows the tools and software packages that can be installed. By default the list is filtered by the product connected to your computer, or that you selected in the Solutions area of the Launcher view. Required items are listed first. Scroll down to see Recommended options, including the current versions of all SDKs. All SDKs are checked by default. Uncheck any you do not wish to install. Leave GNU ARM Toolchain checked if you plan to use GCC. Limiting your installation makes it easier to find documentation and examples, and reduces the time Simplicity Studio spends checking for updates on startup. Simplicity Studio recalculates installation size requirements, which may take several seconds. Click **Next >** once the control is enabled.

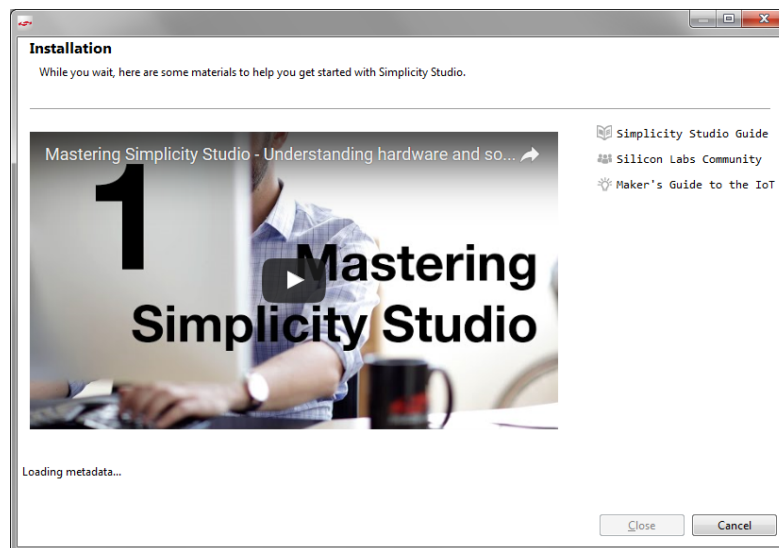
Note: Previous stack versions are shown under **Other Options**.



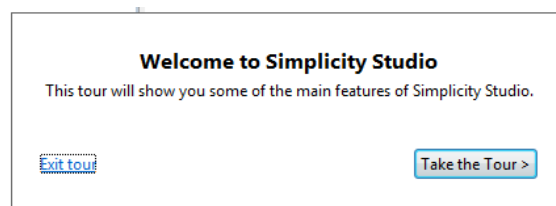
11. Studio displays a Review Licenses dialog. Accept the license(s) shown and click **Finish**. Note that this dialog will present again if in the future you install a component with a separate license.



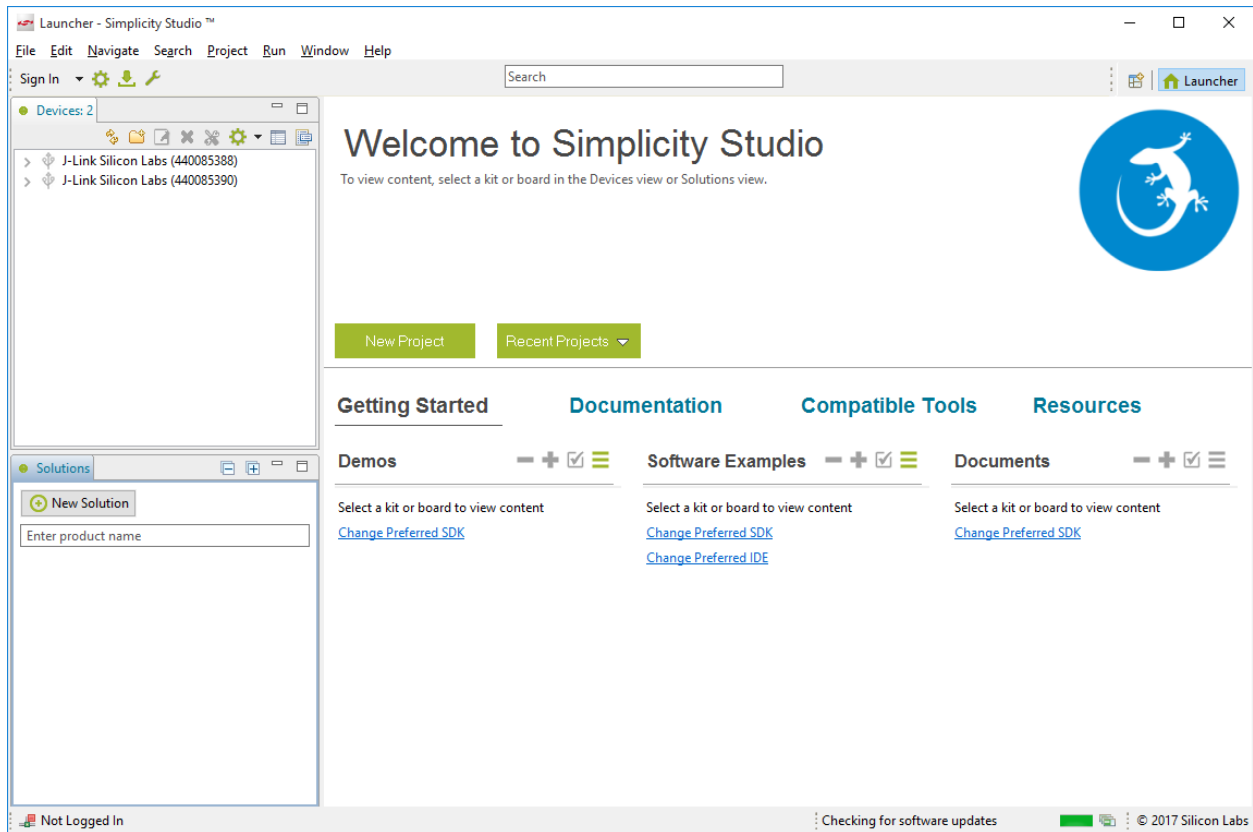
Installation takes several minutes. During installation, Simplicity Studio offers you viewing and reading options to learn more about the environment.



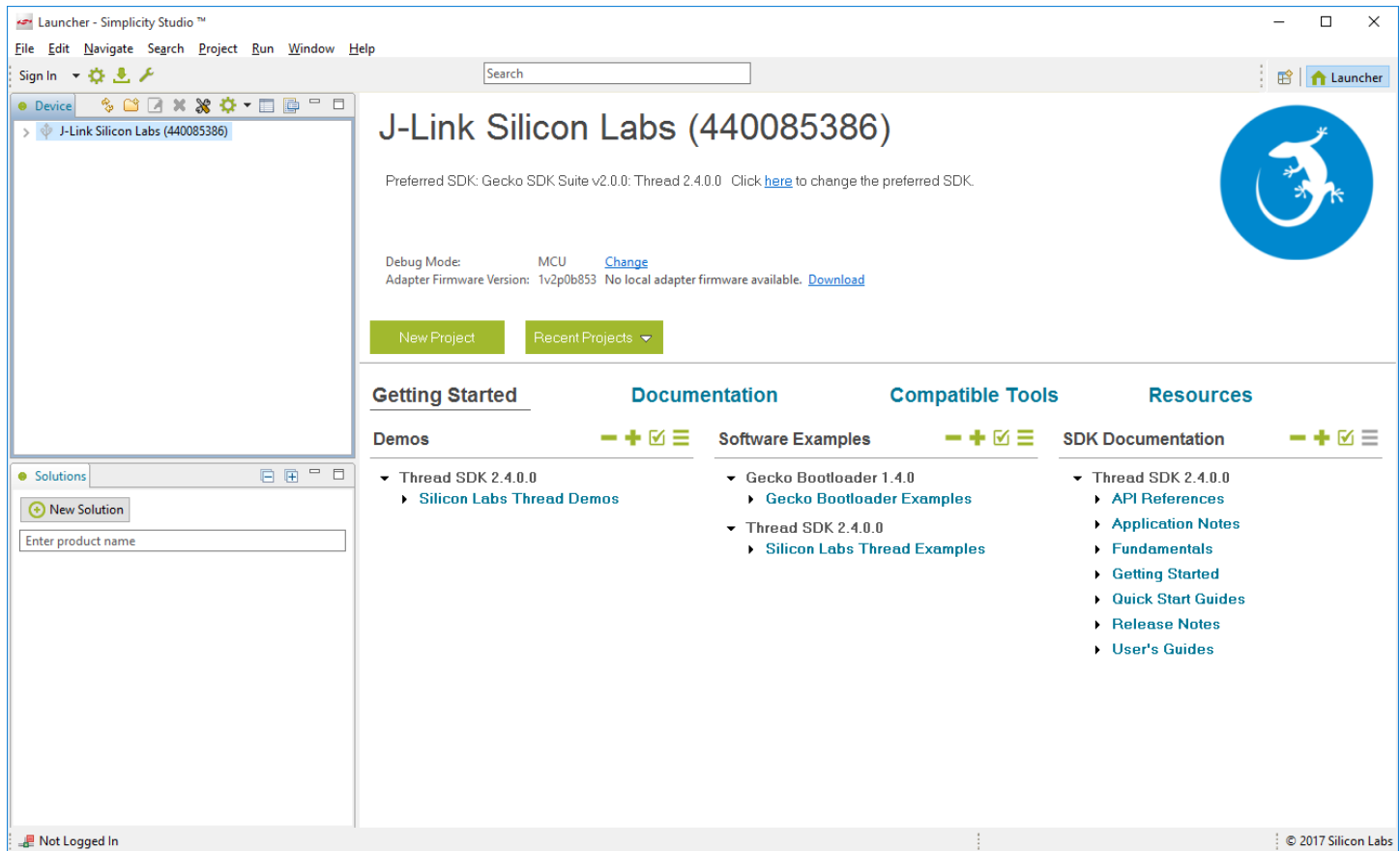
12. After installation is complete, restart Simplicity Studio.
13. When Simplicity Studio restarts, you are invited to take a tour. To clear this option now or at any time during or after the tour, click **Exit tour**.



14. The Launcher perspective opens, but it is not yet fully populated. Click the connection entry in the devices tab. Note that USB-connected WSTK devices are identified as J-Links as shown.



- The Launcher perspective then is populated with the software components and functionality associated with your hardware and stack. Update your device firmware as described in section **Updating Adapter Firmware**.






Finally, if you plan to use IAR as your compiler, find the Release Notes on the list of documents and check for software version requirements, in particular for IAR-EWARM. To install IAR-EWARM:

- On the Launcher page's Resources tab, click **Technical Support**.
- Scroll down to the bottom of the page, and click **Contact Support**
- If you are not already signed in, sign in.
- Click the Software Releases tab. In the View list select **_Latest Thread Software**. Click **Go**. In the results is a link to the appropriate IAR-EWARM version.
- Download the IAR package (takes approximately 1 hour).
- Install IAR.
- In the IAR License Wizard, click **Register with IAR Systems to get an evaluation license**.
- 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. Additional information on some of these is provided later in the section.

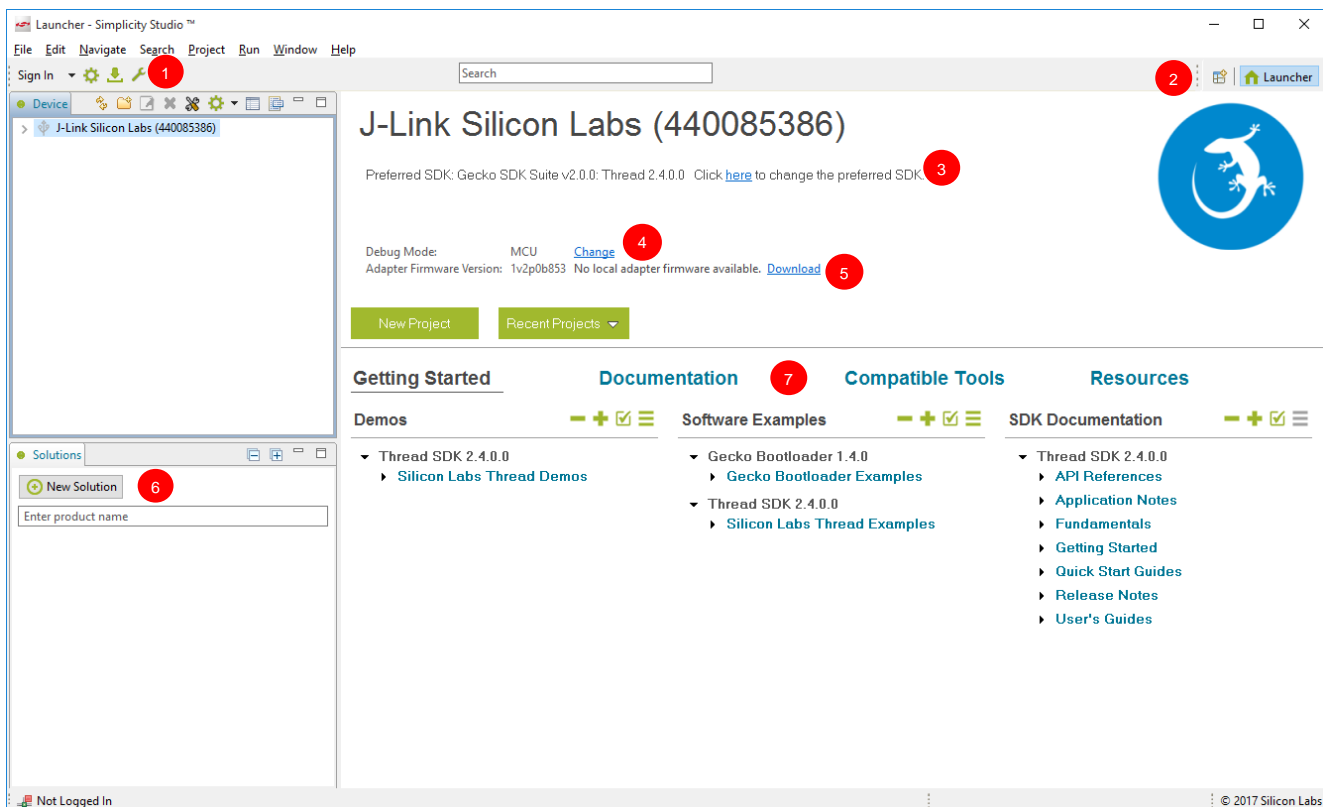
On the toolbar (1) you can:

- Sign in or out
- Open application settings ()
- Update your software and firmware ( , see section [Downloading Updates or Installing Additional Components](#) for more information)
- Open the Tools menu () to access tools such as Simplicity Commander or Energy Profiler.
- Search for information on line, 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 [Changing the Preferred SDK](#) for more information - legacy functionality, rarely used).
- Change debug mode (4).
- Update adapter firmware (5, see [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).

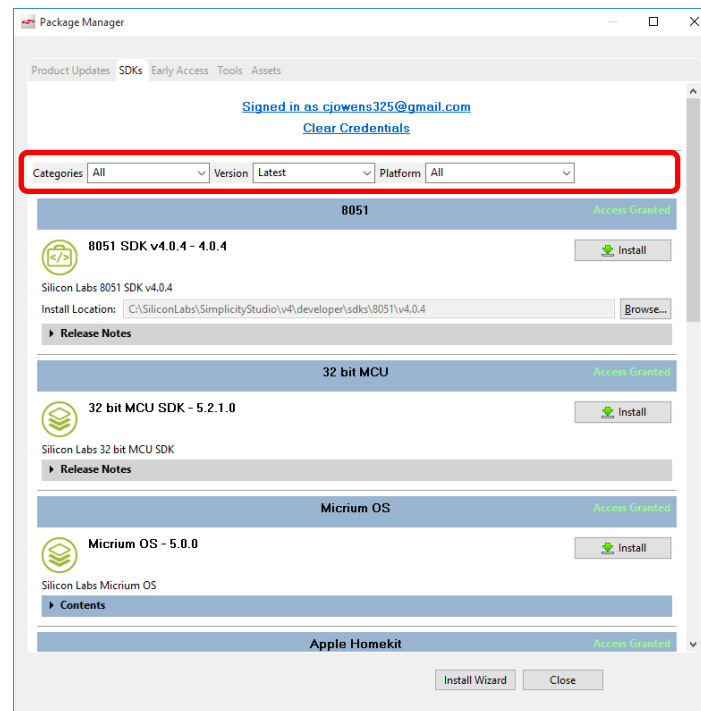
 Use these controls to manage groups of items (Collapse All, Expand All, Customize, and Show All, respectively). See [Accessing Documentation and Other Resources](#) for more information.



3.1 Downloading Updates

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

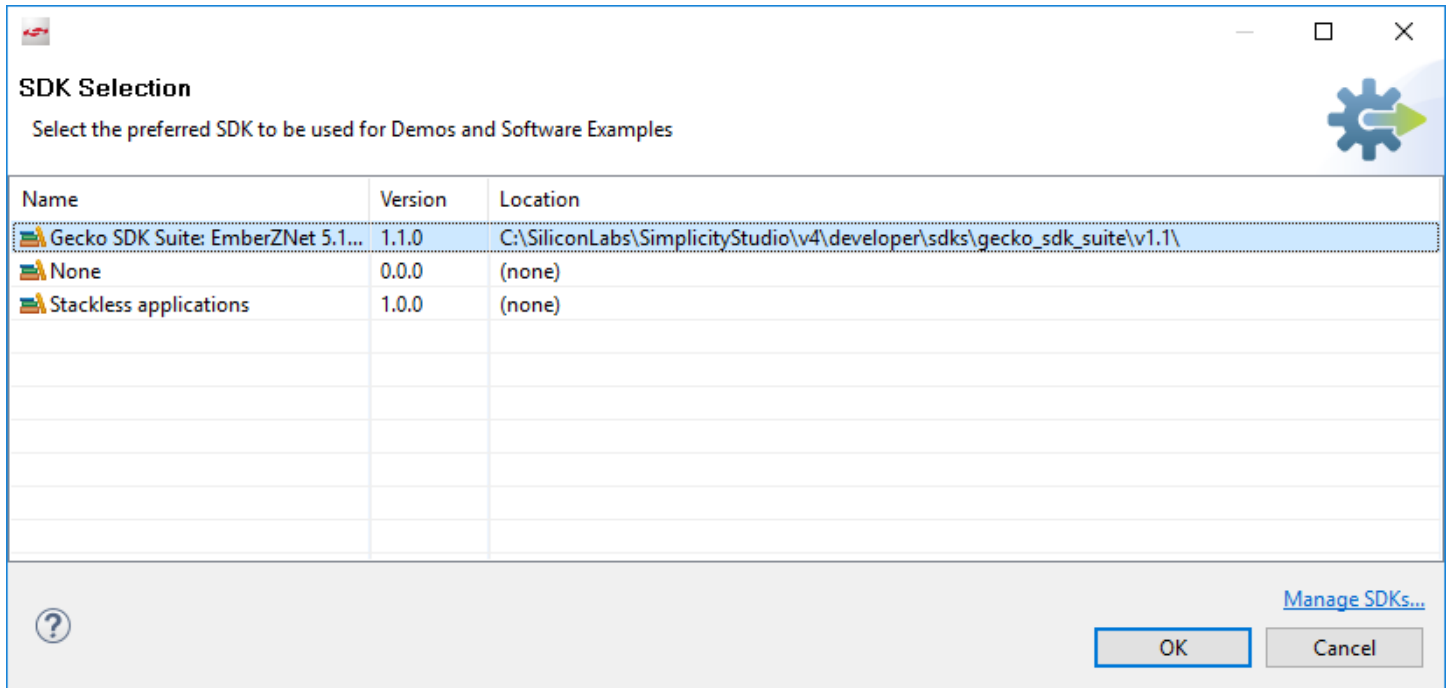
When you click the Download Update icon or accept updates in the notification dialog, Simplicity Studio shows you available updates or SDKs in the Package Manager dialog. You can update all or select individual updates for installation. Click the tabs in the Package Manager dialog to see other components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



Note: If you are using a new device or product family, you can use the **Install Wizard** button to access the installation interface provided during initial installation. This makes installation of all components related to a selected device(s) or an entire family easier.

3.2 Changing the Preferred SDK

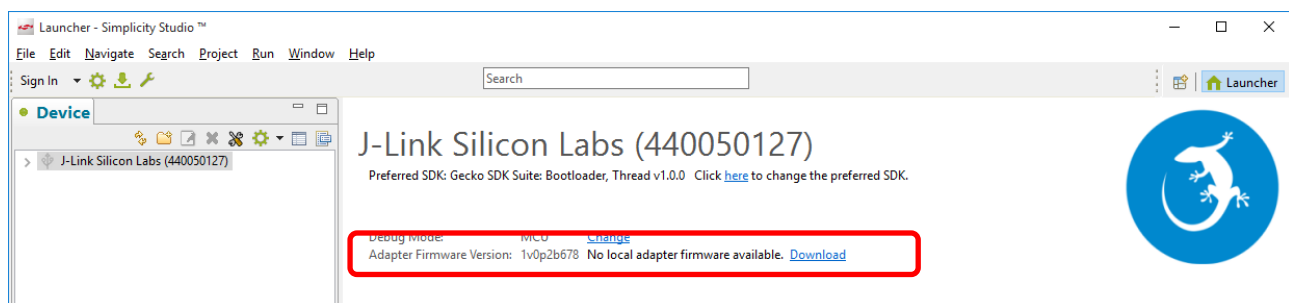
This is a legacy function. In general, most Silicon Labs protocol stack users will have one SDK available to them, the Gecko SDK Suite.



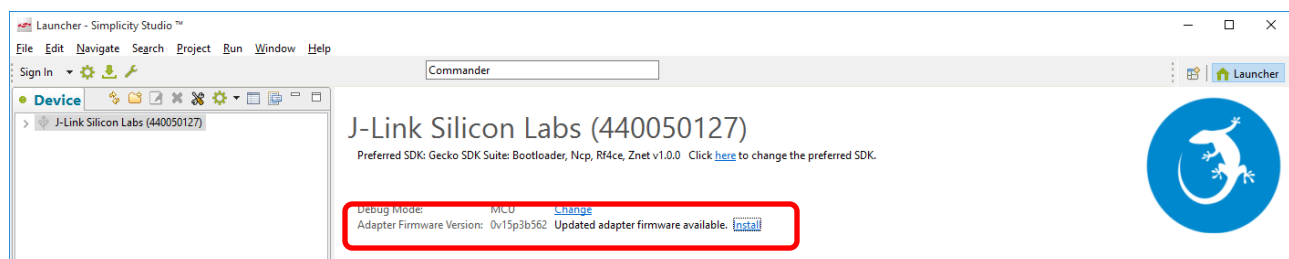
Within that suite you can have multiple protocols installed. The protocol used in any given instance is controlled either by the example you select, or the stack you select if you go through the 'New Project' interface. In general, you should add or remove protocol stacks through the Simplicity Studio 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.




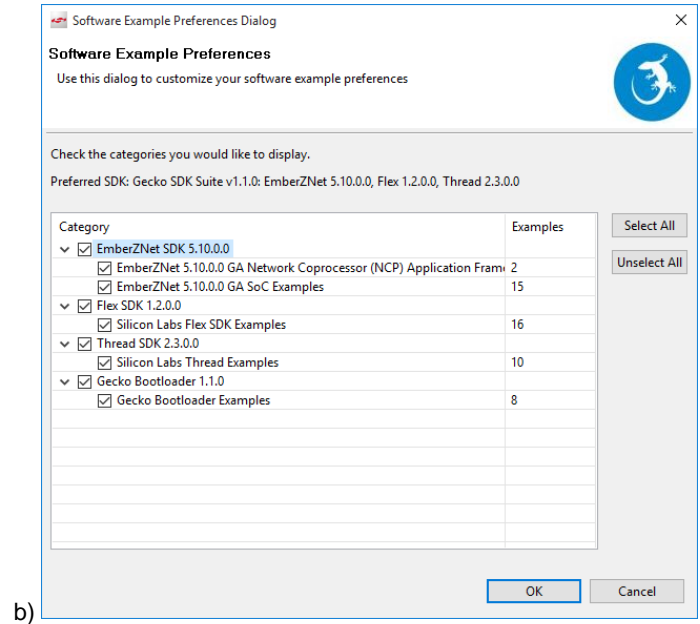
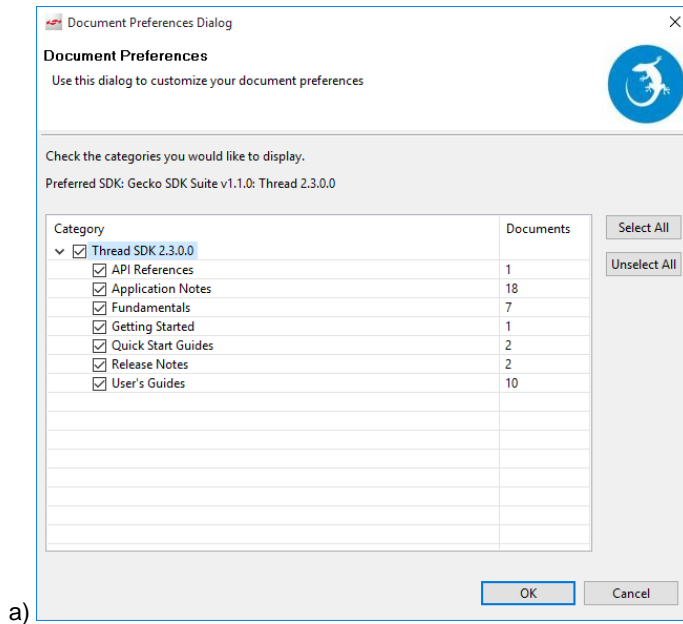
If an update is available, click **Install** to install the firmware.



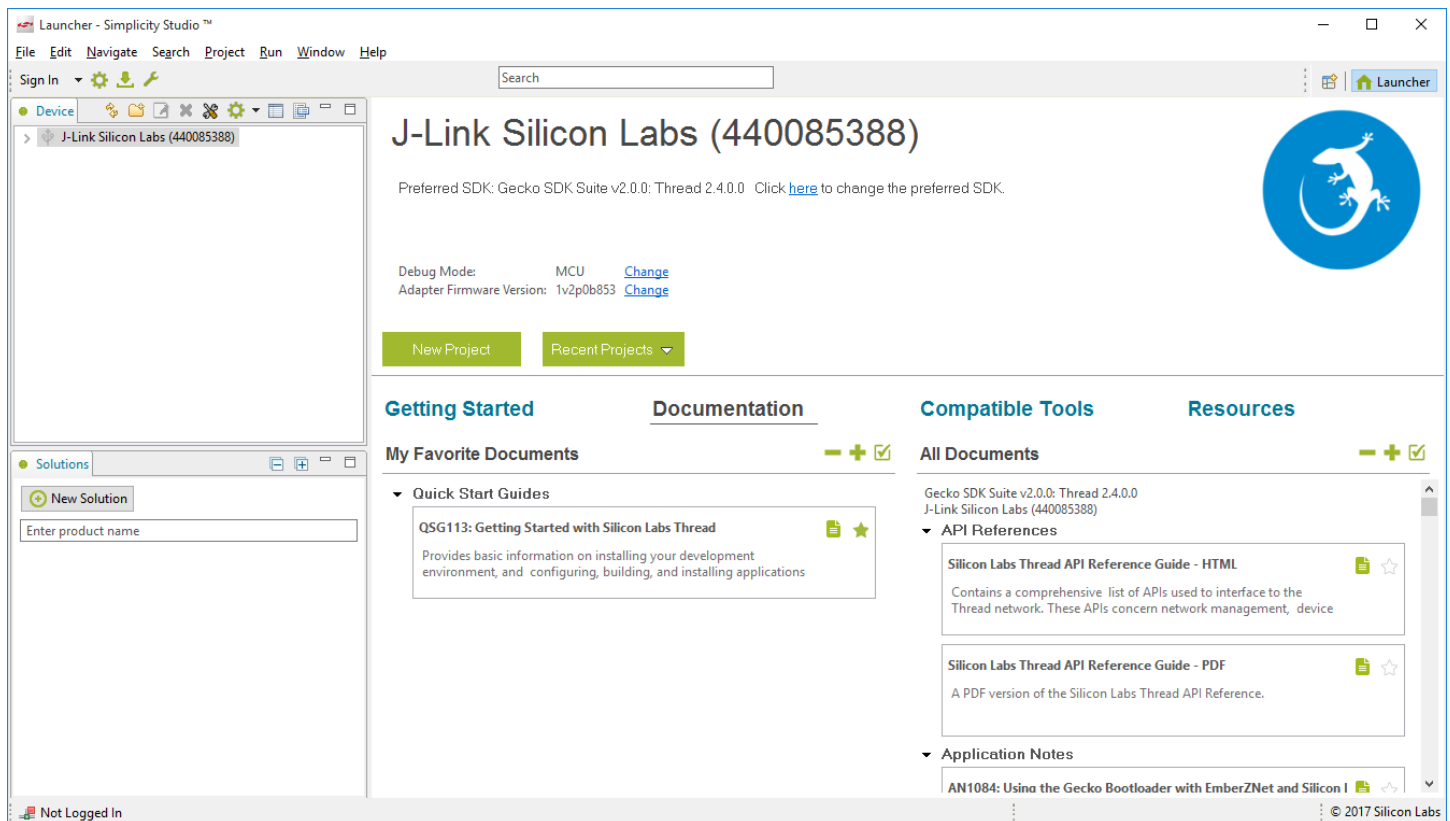
Once you have installed a current update, the version is displayed. Studio will notify you if another firmware update is available.

3.4 Accessing Documentation and Other Resources

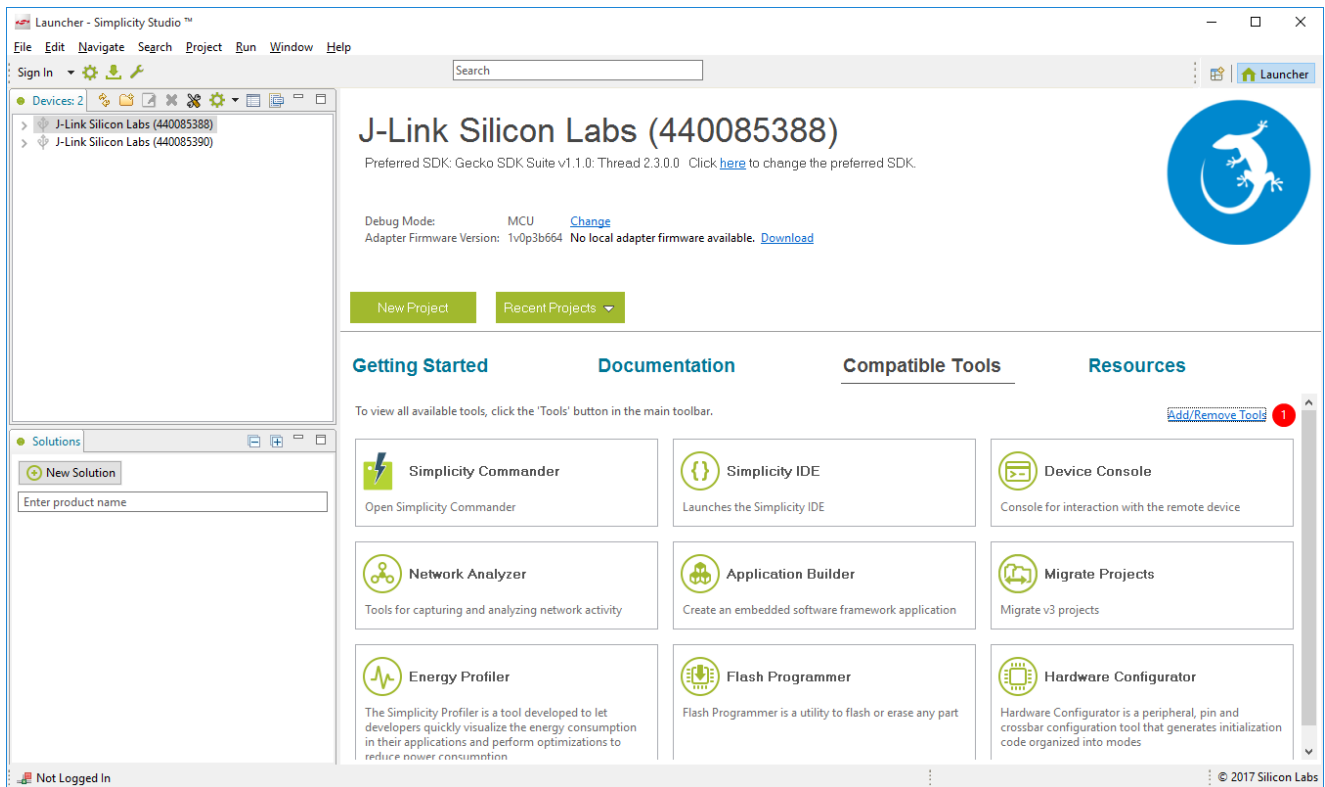
The **Getting Started** tab provides access to demos, example applications, and stack related documentation. To show/hide specific categories, click . Select or deselect categories, then click **OK**. If you have more than one protocol SDK installed, categories are grouped by protocol, as shown for software examples in (b) below.



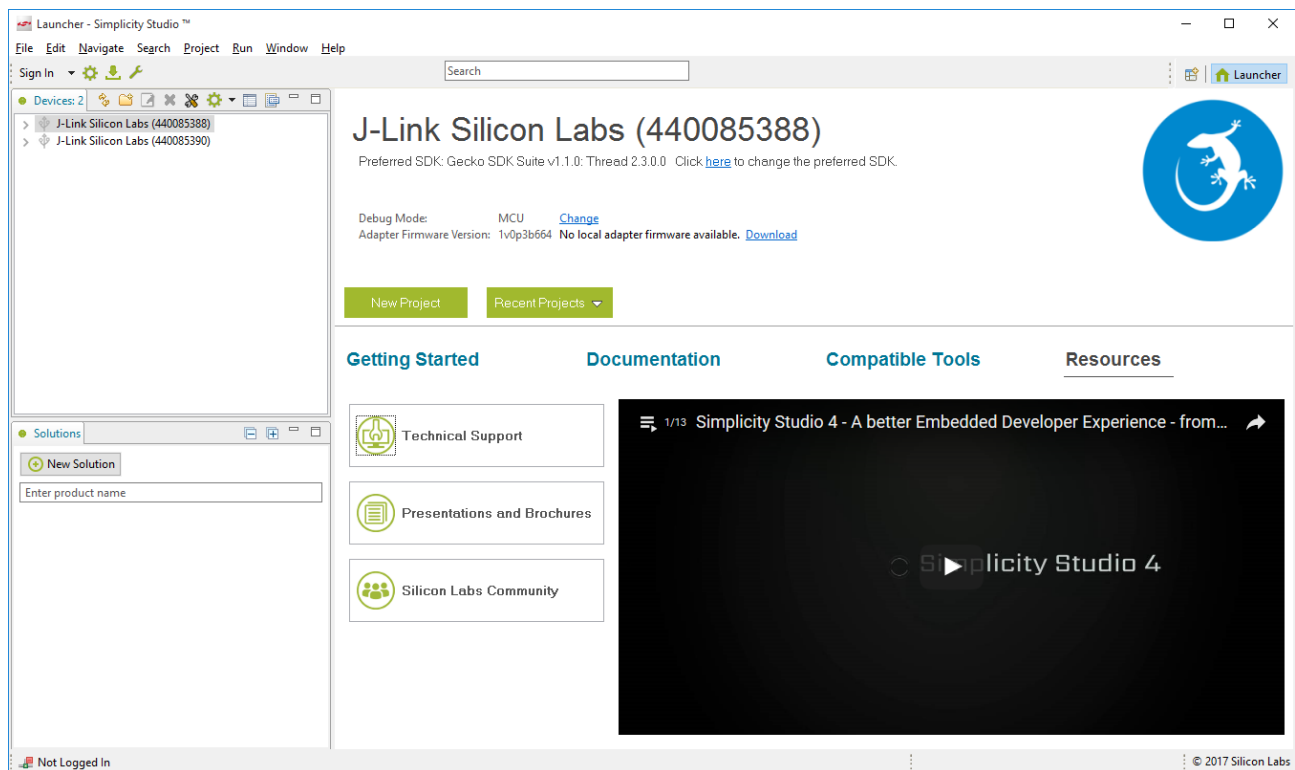
The **Documentation** tab combines documentation about the stack and about the hardware. Click the star icon on any document to move it to the My Favorite Documents list.



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 Silicon Labs Thread 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. 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.

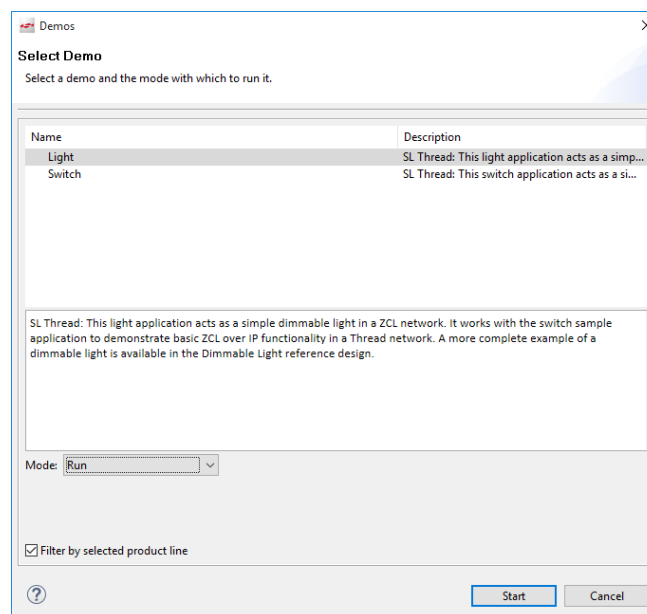
4.1 Demos

Demos are prebuilt application examples that can be directly downloaded to your device.

Light (SoC): Acts as a simple dimmable light that works with the switch application to demonstrate basic ZCL over IP functionality in a Thread network. A more complete example of a dimmable light is available in the Dimmable Light reference design.

Switch: Acts as a simple dimmer switch. It works with the light application to demonstrate basic ZCL over IP functionality in a Thread network.

To download and run a demo on your device, drop down the demo list and click the demo. In the Mode drop-down in the next dialog, select **Run**. Click **Start**.



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 Silicon Labs Thread software examples are as follows. Examples marked with * do not have tiles on the launcher page but can be accessed in the New Project dialogs. Be sure to check the complete example descriptions for any platform restrictions.

***Border Router Management App:** This is a component of the Silicon Labs Border Router Reference Design for Unix hosts, which demonstrates device commissioning with the Thread Group Commissioning App and routing between a Thread network and adjacent IP networks.

***Capacitive Touch Sensing Switch:** This ZCL over IP sensor application provides open/close state of a magnetic reed switch and device tamper detection, and is a component of the Silicon Labs Border Router Reference Design.

***Contact Sensor:** This ZCL over IP sensor application provides open/close state of a magnetic reed switch and device tamper detection, and is a component of the Silicon Labs Border Router Reference Design.

***Dimmable Light:** This ZCL over IP light application provides pwm control for a dimming function, and is a component of the Border Router Reference Design.

Client (Sleepy): This client application acts as a sleepy data sensor in a wireless sensor network. It reports information to a server node that acts as a sink.

Client: This client application acts as a data sensor in a wireless sensor network. It reports information to a server node that acts as a sink. The two client examples and the associated server example show a simple, proprietary CoAP-based messaging model rather than using ZCL/IP as the application layer.

Light (SoC): This light application acts as a simple dimmable light in a ZCL network. It works with the switch sample application to demonstrate basic ZCL over IP functionality in a Thread network. A more complete example of a dimmable light is available in the Dimmable Light reference design.

***Light (Host):** This light application acts as a simple dimmable light in a ZCL network. It works with the switch sample application to demonstrate basic ZCL over IP functionality in a Thread network. A more complete example of a dimmable light is available in the Dimmable Light reference design.

NCP-SPI: This network coprocessor (NCP) application supports communication with a host application over a SPI interface. It can be built as configured, or optionally can be augmented with customized extensions for initialization, main loop processing, event definition/handling, and messaging with the host.

NCP UART (Software Flow Control): This NCP application can be built as configured, or optionally can be augmented with customized extensions for initialization, main loop processing, event definition/handling, and messaging with the host. Note that in order to utilize software flow control with the WSTK boards, the expansion header ("EXP") pins must be used as the USB-to-serial interface on the WSTK does not support software flow control.

NCP UART HW (Hardware Flow Control): This network coprocessor (NCP) application supports communication with a host application over a UART interface with hardware flow control (RTS/CTS). This NCP application can be built as configured, or optionally can be augmented with customized extensions for initialization, main loop processing, event definition/handling, and messaging with the host.

***Occupancy Sensor:** This ZCL over IP sensor application provides occupancy sensing, temperature measurement, relative humidity measurement, and illuminance measurement. It is intended to be used with the Border router Reference design.

Sensor/Actuator: This ZCL over IP sensor/actuator node application provides LED0 and buzzer (EM35x-DEV only) control, and reports temperature and button 0 state. It is intended to be used with the Border Router reference design.

Server (SoC): This server application acts as a data sink in a wireless sensor network. It collects information from client nodes that act as sensors.

***Server (Host):** This server application acts as a data sink in a wireless sensor network. It collects information from client nodes that act as sensors.

***Smart Outlet:** This ZCL over IP sensor and actuator application provides outlet on/off control, over-current and over-temperature shut-down, and power, temperature, relative humidity and illuminance measurements. It is intended to be used with the Border Router reference Design.

Switch: This switch application acts as a simple dimmer switch in a ZCL network. It works with the light sample application to demonstrate basic ZCL over IP functionality in a Thread network.

4.3 Starting with a Blank Application

While Silicon Labs strongly recommends starting development from one of the example applications, if you want to start with a blank application you can.

1. On the launcher page click **New Project**.
2. Click **Silicon Labs Thread**, and click **Next**.
3. Check Start with a blank application, and click **Next**.
4. Name your application, and click **Next**.

5. (If you have installed IAR and GCC) Select your toolchain. GCC is set as active by default, but you can change the active toolchain to IAR. You can also change the toolchain once the project has opened.
6. Click **Finish**. The Simplicity IDE opens, but nothing is configured.

5 Getting Started with Application Development

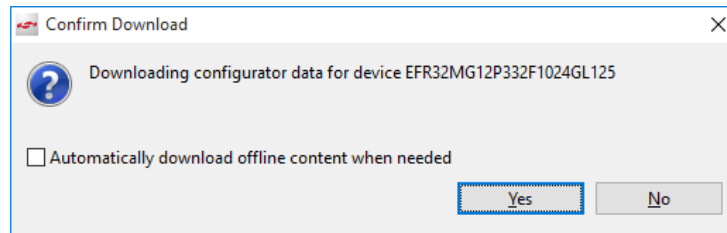
In these instructions you will compile and load two example applications, Light and Switch, to create a simple Thread network. When the applications are loaded and the Client has joined the network, you can observe traffic across the network in the console, and also in Network Analyzer.

When working with example applications in Simplicity Studio, you will 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.
4. Interact with the application.

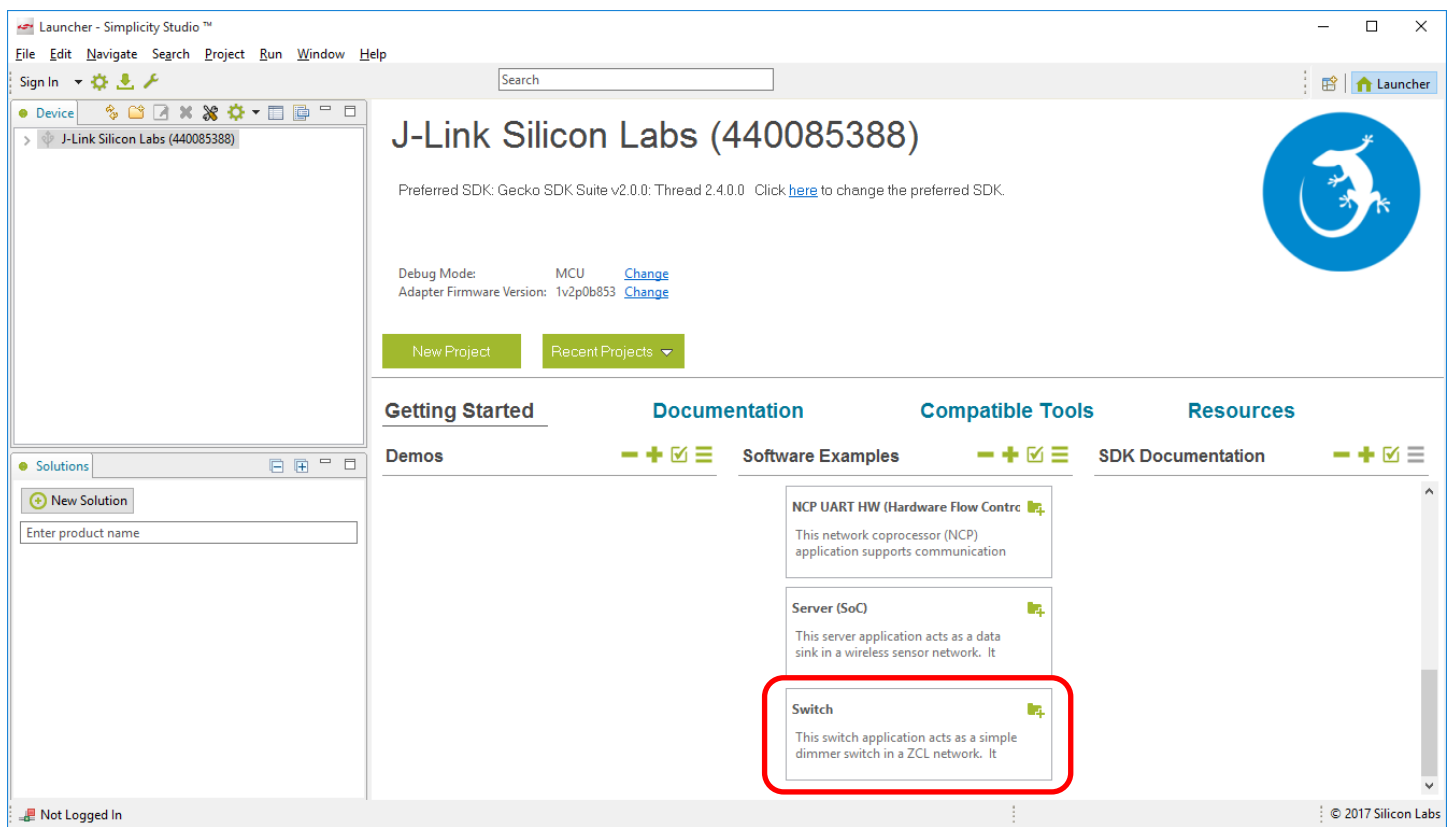
These steps are described in detail in the following sections. These procedures are illustrated for a WSTK with an EFR32MG12.

Note: SDK version 2.4 contains 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**.

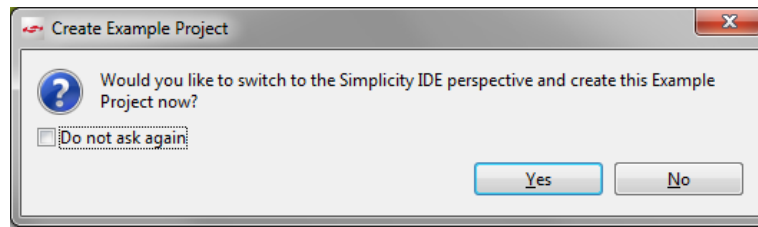


5.1 Selecting an Example Application

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



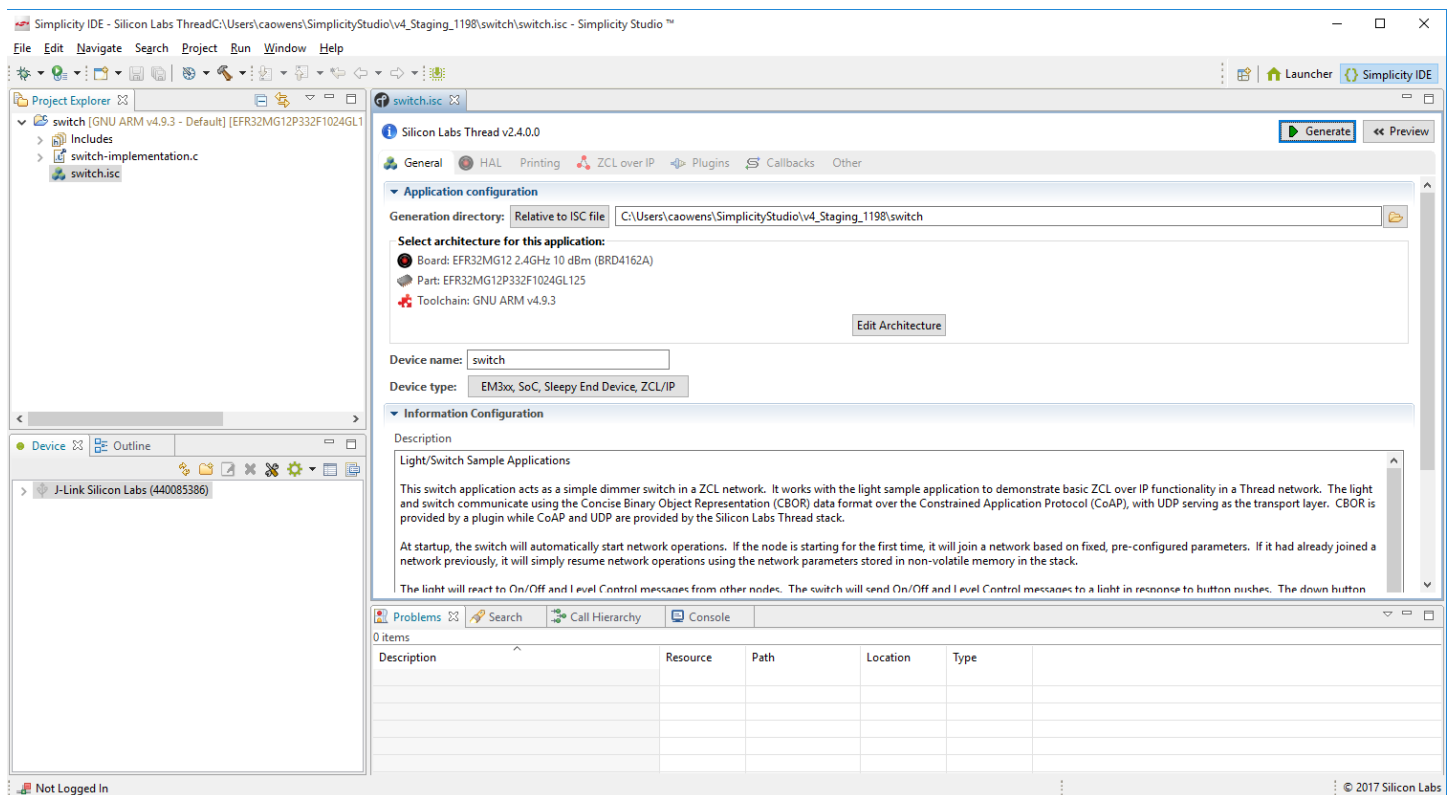
- You are asked if you want to switch to the Simplicity IDE. Click **Yes**.



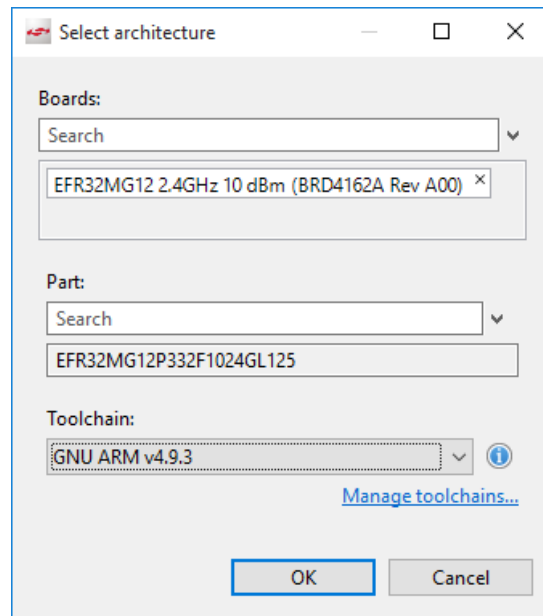
- Simplicity IDE opens with the new project in AppBuilder view, and the focus on the General tab.

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 an example for a part with less than 512 kB, such as the EFR32xG1, you must use IAR due to code size differences between GC and IAR.



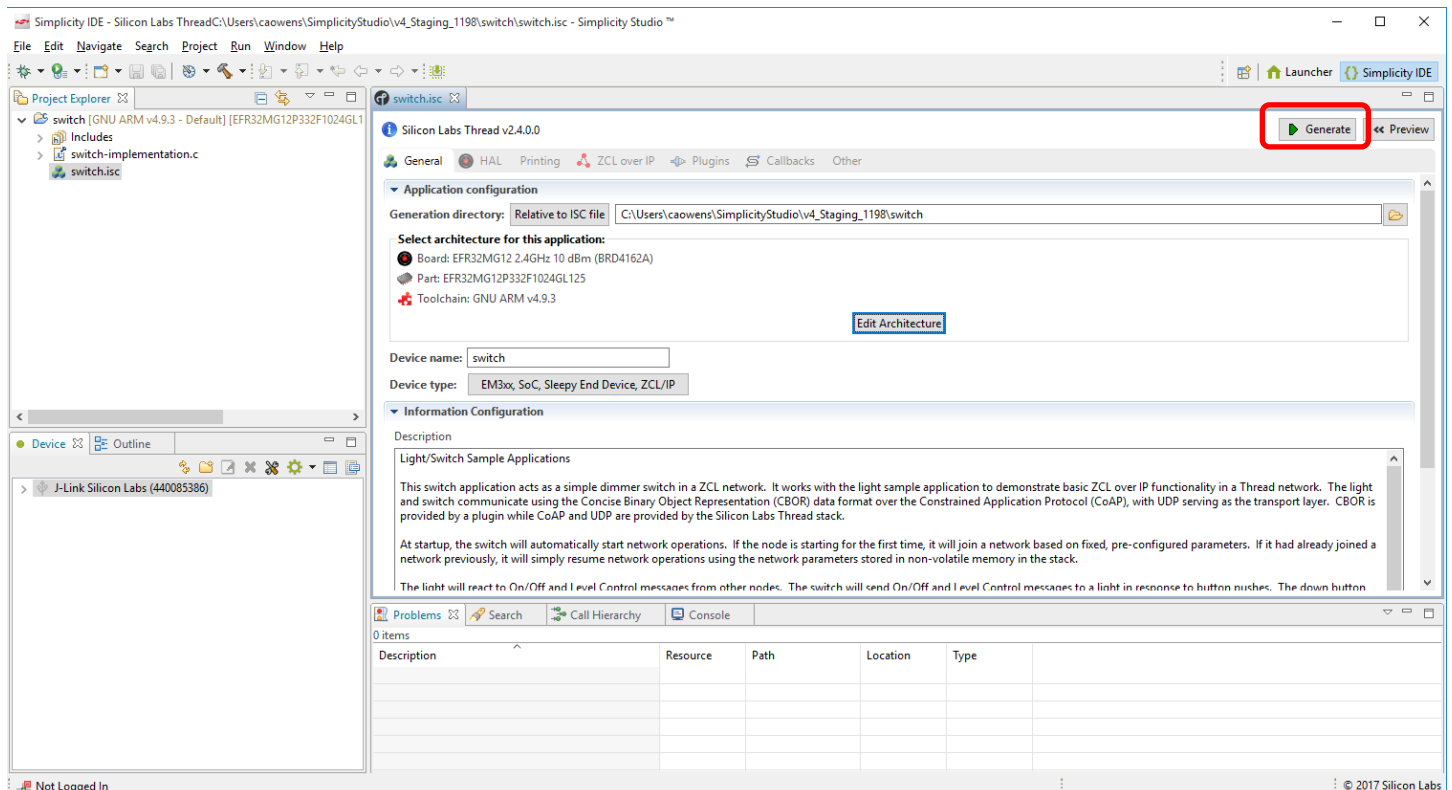
- To change the toolchain click **Edit Architecture**. In the resulting dialog, select the desired toolchain and click **OK**.



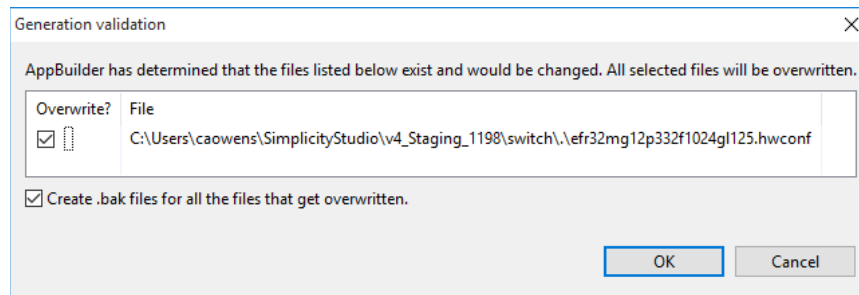
Note: You can also start an example project by clicking **New Project** in the Launcher perspective or selecting **New** from the Project menu in the Simplicity IDE perspective, and completing a series of dialogs. This path allows you more input into the project creation, including changing the project name and modifying the default board and part selections. It also provides access to additional examples.

5.2 Generating the Application Source Files

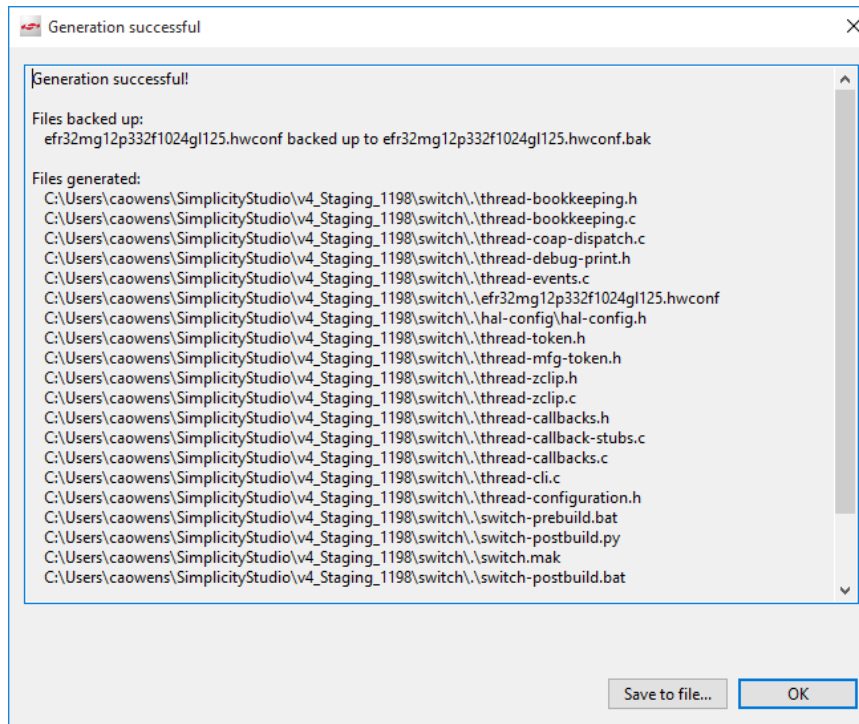
- In the Simplicity IDE, click **Generate**.



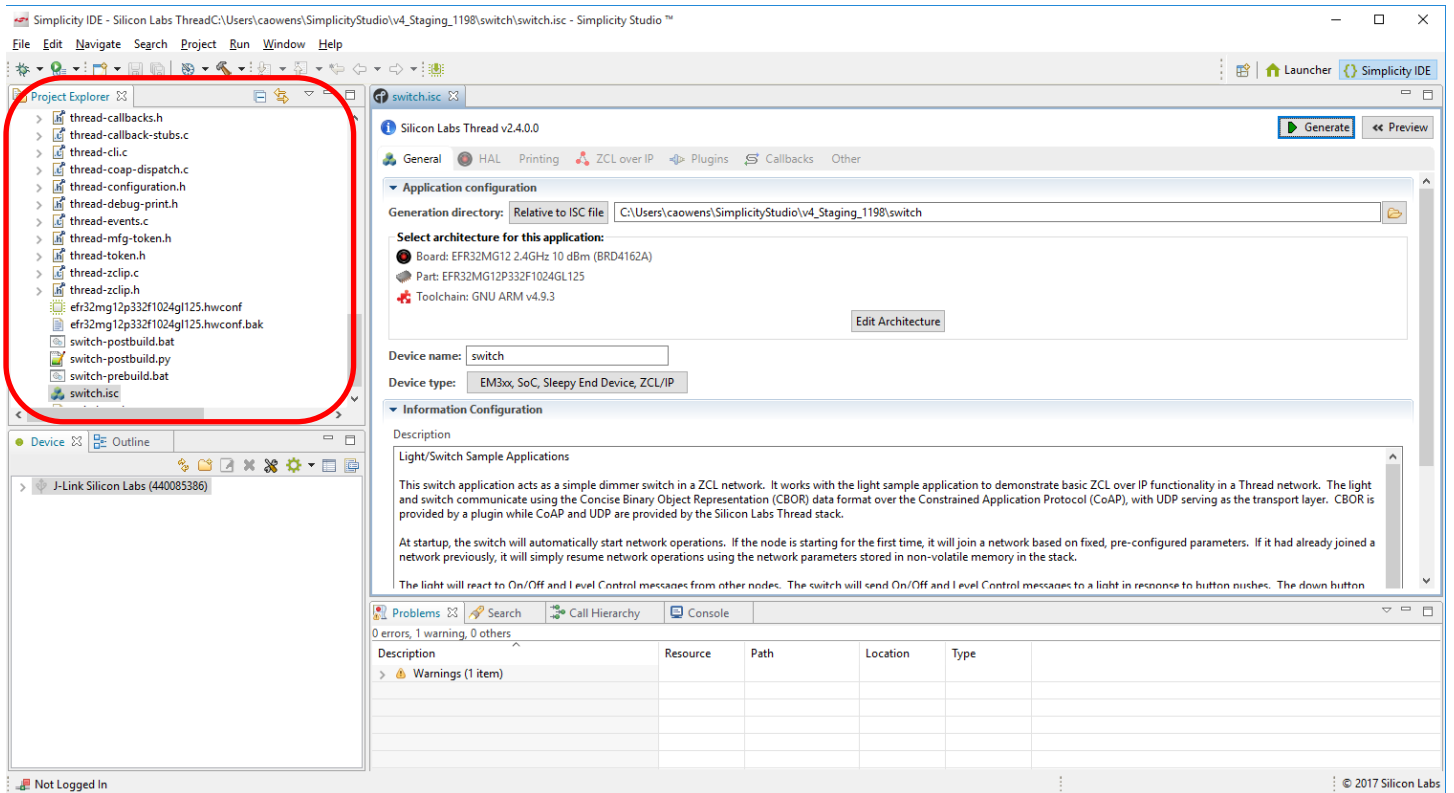
If you get the following overwrite warning, click **OK**.



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



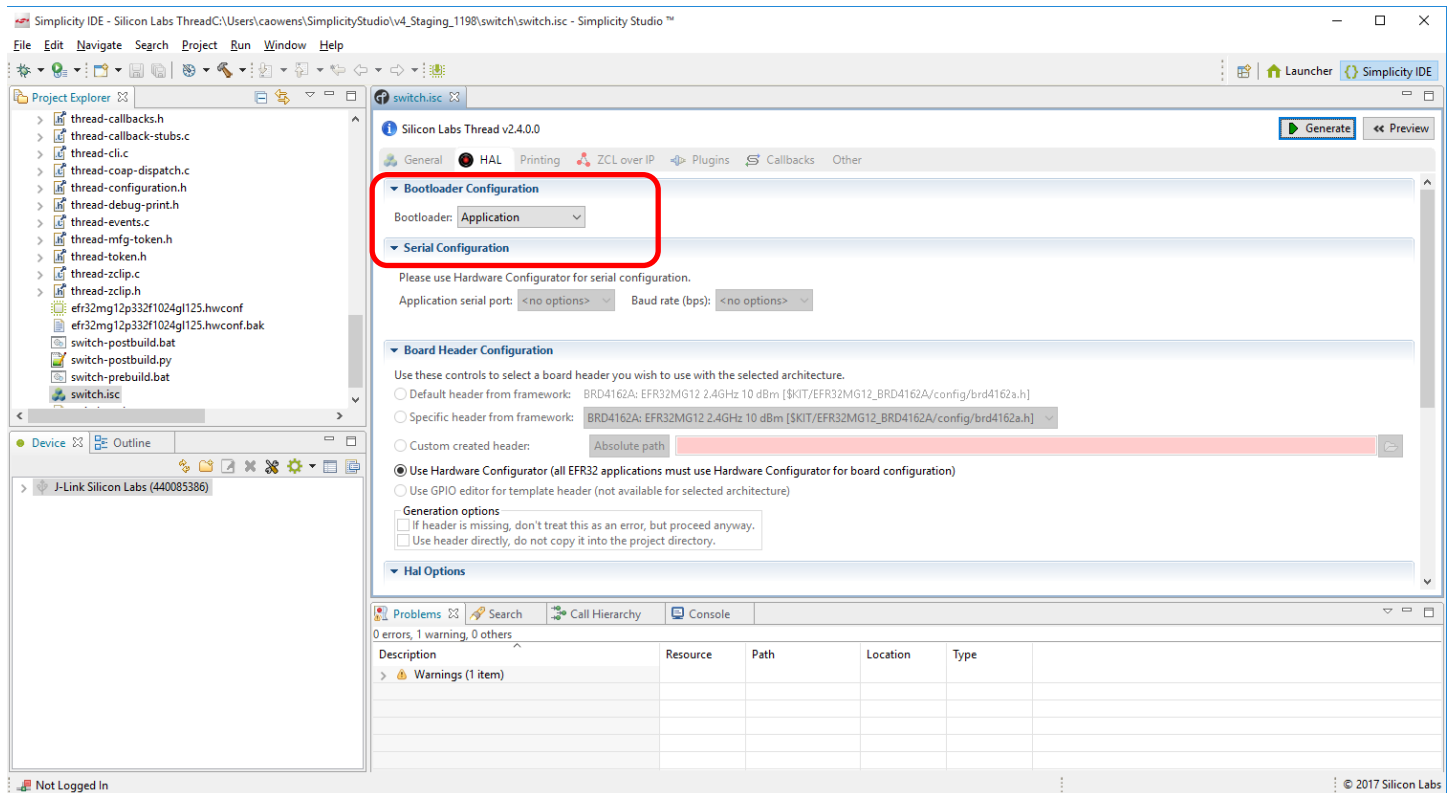
The generated files are shown in the Project Explorer view.



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*.

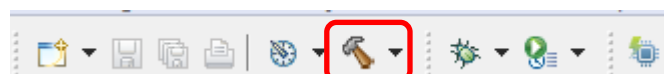
In March of 2017, Silicon Labs introduced the Gecko Bootloader, a code library configurable through Simplicity Studio's IDE to generate bootloaders that can be used with a variety of Silicon Labs protocol stacks. The Gecko Bootloader is used with all EFR32xG parts.

The Gecko Bootloader works with a specialized firmware update image format, with update images ending in the extension .gbl (GBL files). 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.

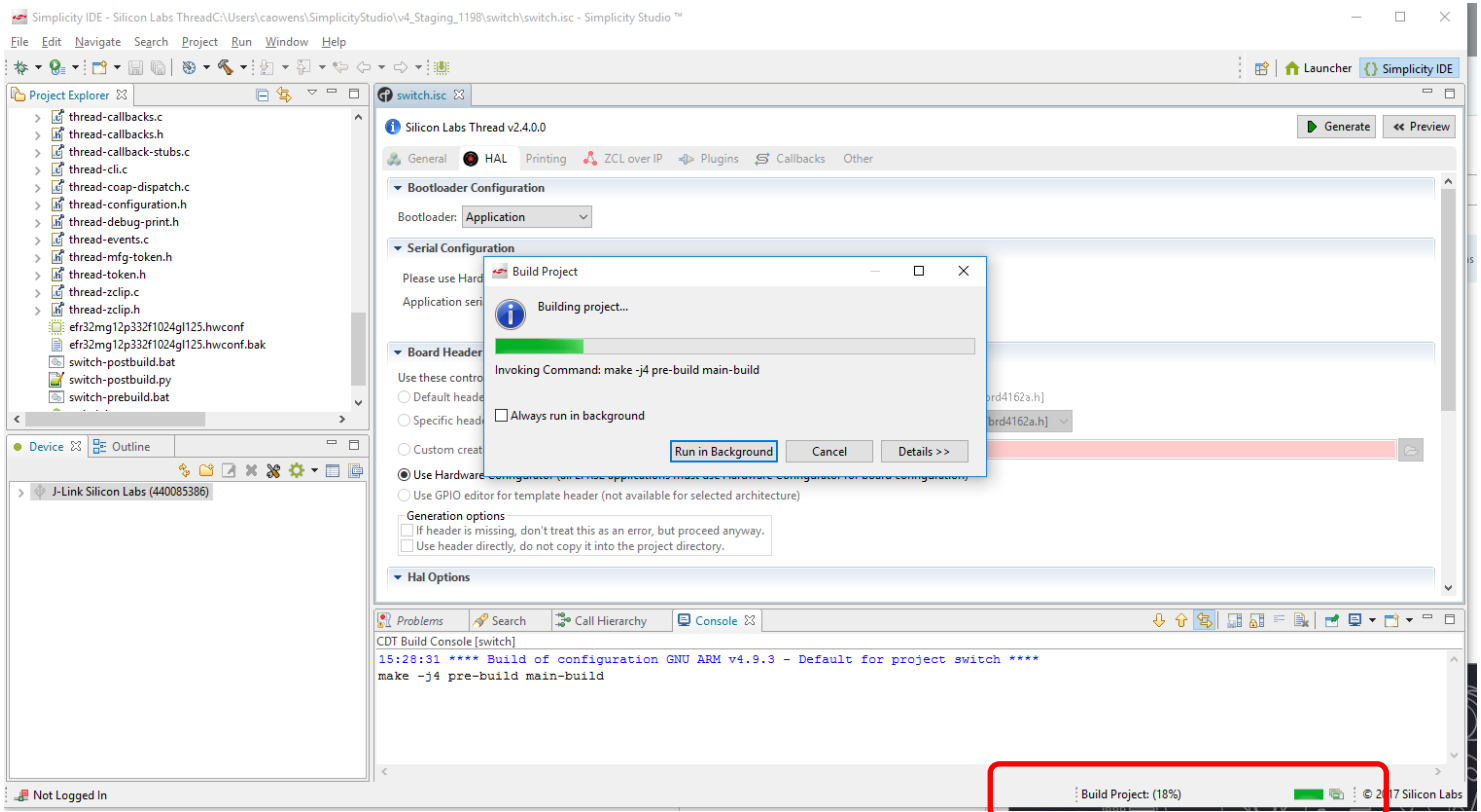
Note: When working with the Gecko Bootloader, you must use Simplicity Commander to enable some configuration options such as security features. See *UG266: Silicon Labs Gecko Bootloader User's Guide*.

5.3.2 Building and Flashing Files

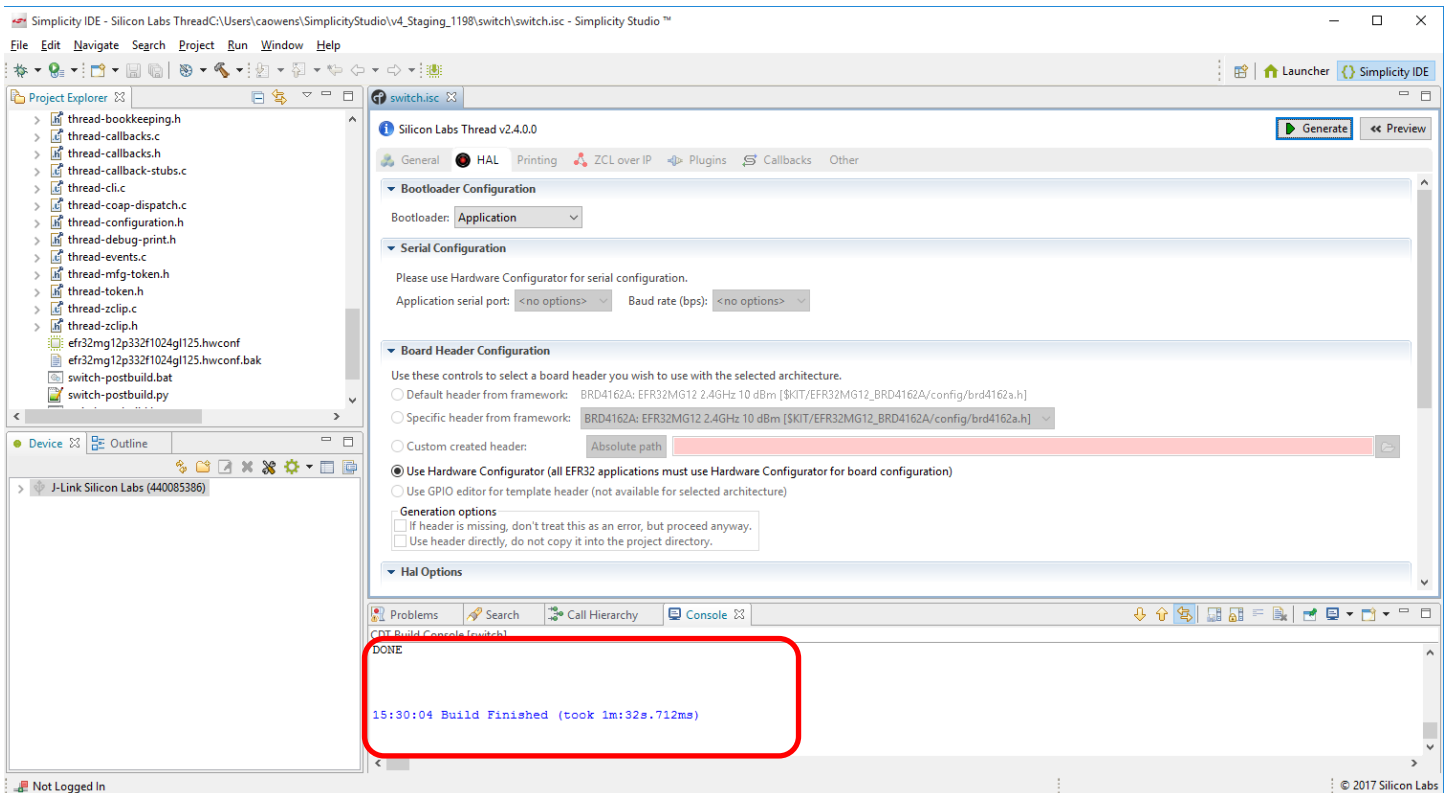
1. After you generate your project files, click the **Build** control in the top tool bar. If the Build control is not enabled, click the device. Your sample application will compile based on its build configuration. You can change the build configuration at any time in the Project Explorer View by right 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.



Build completion is reported in the Build Console.



The build should complete with no errors. If any errors occur they are highlighted in the console. Contact technical support for assistance.

```

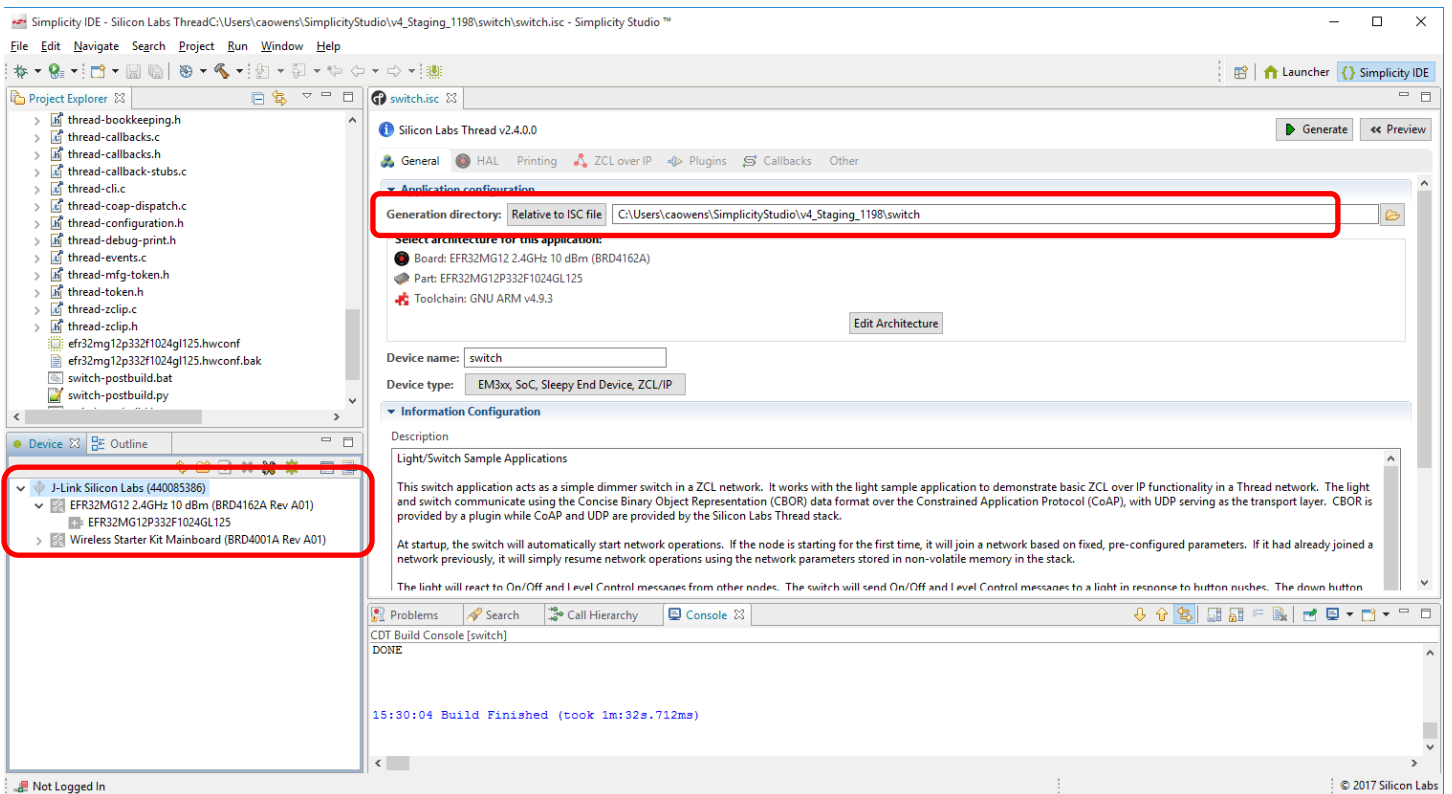
CDT Build Console [ncp_custom_tokens_em3587]

Warnings: 1
make: *** [ncp-callbacks.o] Error 2
make: *** Waiting for unfinished jobs....
Finished building: ../ncp-callback-stubs.c

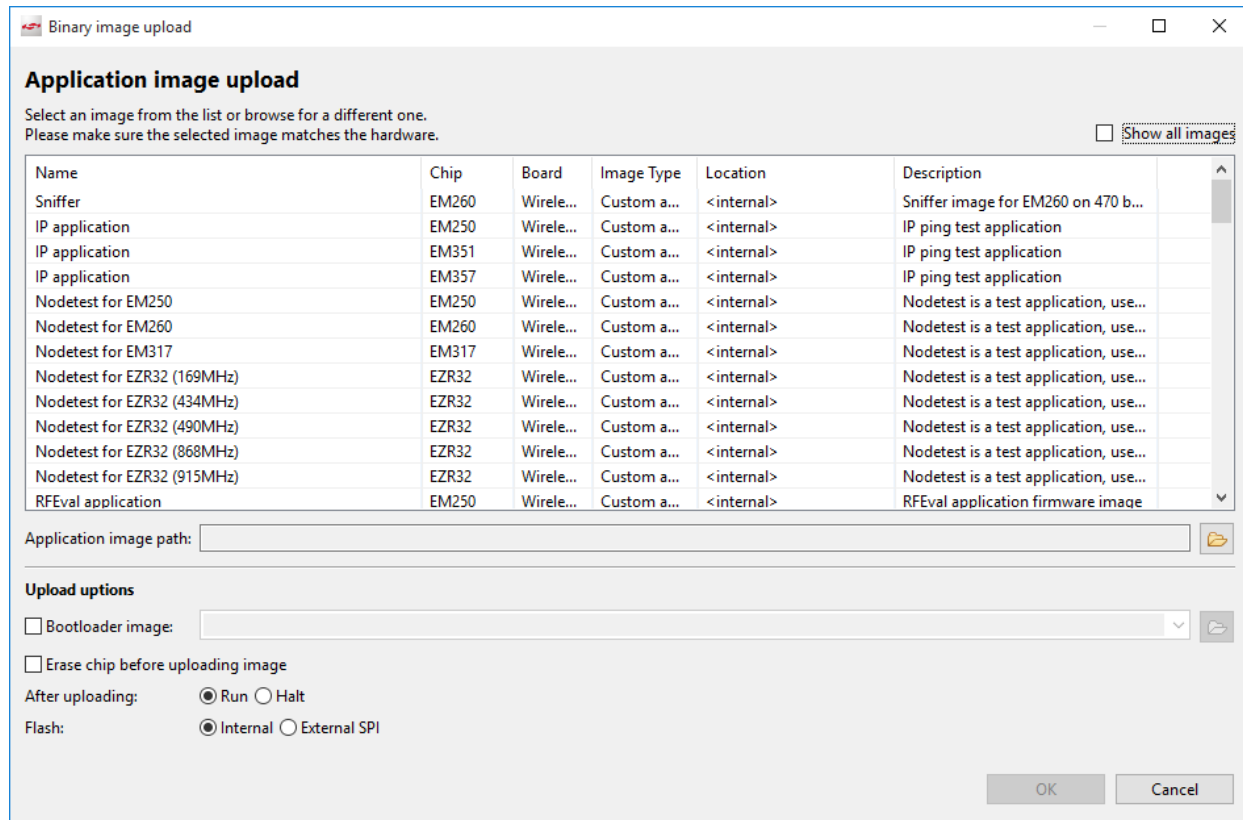
14:10:22 Build Finished (took 49s.748ms)
    
```

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.

- 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 to find the correct bootloader file. Note that the folder in which the example was generated is displayed on the General tab.



- Right-click on the device and select **Upload Application**. The Application Image Upload dialog is displayed. Your list may differ from the following example.



- Browse to the folder with your compiled application and select the GBL file (see section **About Bootloading** for more information).

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

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

- If you have not already loaded a bootloader, check **Bootloader image**, then browse to the folder containing a prebuilt bootloader image corresponding to the part number for your radio board. Images are located in the Simplicity Studio bootloader folder under platform (for example: C:\SiliconLabs\SimplicityStudio\v4\developer\sdk\gecko_sdk_suite\<version>\platform\bootloader\). Browse to sample-apps and the bootloader-storage-spiflash folder. Select the .s37 file corresponding to the radio board part number, for example 'bootloader-storage-spiflash-efr32mg12p432f1024gl125.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:

Binary image upload

Application image upload

Select an image from the list or browse for a different one.
Please make sure the selected image matches the hardware.

☐ Show all images

Name	Chip	Board	Image Type	Location	Description
Sniffer	EM260	Wirele...	Custom a...	<internal>	Sniffer image for EM260 on 470 b...
IP application	EM250	Wirele...	Custom a...	<internal>	IP ping test application
IP application	EM351	Wirele...	Custom a...	<internal>	IP ping test application
IP application	EM357	Wirele...	Custom a...	<internal>	IP ping test application
Notetest for EM250	EM250	Wirele...	Custom a...	<internal>	Notetest is a test application, use...
Notetest for EM260	EM260	Wirele...	Custom a...	<internal>	Notetest is a test application, use...
Notetest for EM317	EM317	Wirele...	Custom a...	<internal>	Notetest is a test application, use...
Notetest for EZR32 (169MHz)	EZR32	Wirele...	Custom a...	<internal>	Notetest is a test application, use...
Notetest for EZR32 (434MHz)	EZR32	Wirele...	Custom a...	<internal>	Notetest is a test application, use...
Notetest for EZR32 (490MHz)	EZR32	Wirele...	Custom a...	<internal>	Notetest is a test application, use...
Notetest for EZR32 (868MHz)	EZR32	Wirele...	Custom a...	<internal>	Notetest is a test application, use...
Notetest for EZR32 (915MHz)	EZR32	Wirele...	Custom a...	<internal>	Notetest is a test application, use...
RFEval application	EM250	Wirele...	Custom a...	<internal>	RFEval application firmware image

Application image path:

C:\Users\caowens\SimplicityStudio\v4_Staging_1198\switch\GNU ARM v4.9.3 - Default\switch.gbl

Upload options

☒ Bootloader image: C:\SiliconLabs\SimplicityStudio\v4\archive\Common-Deliverable\platform\bootloader\sample-apps\bootloader-storage-spiflash\efr32m...

☒ Erase chip before uploading image

After uploading:

☒ Run
☐ Halt

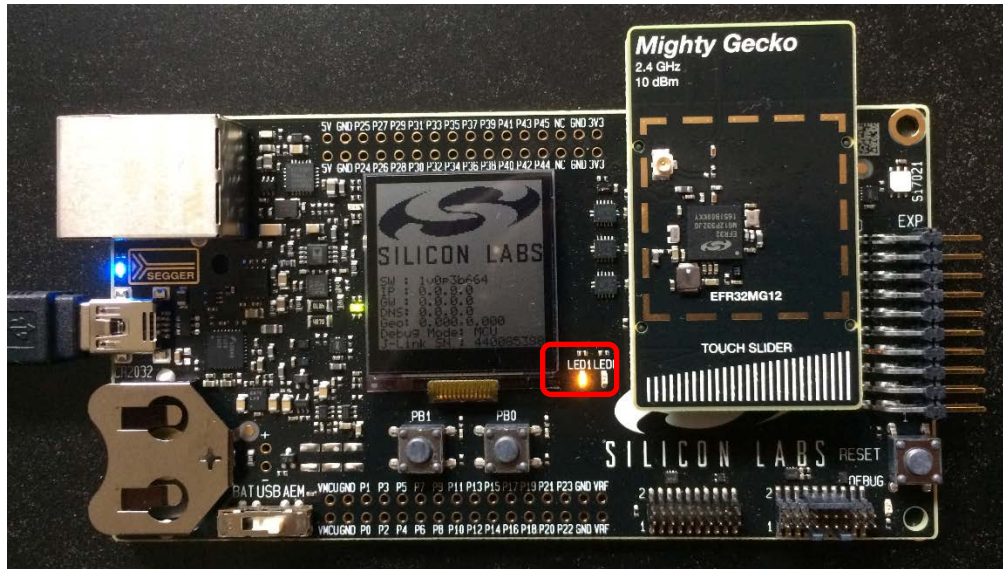
Flash:

☒ Internal
☐ External SPI

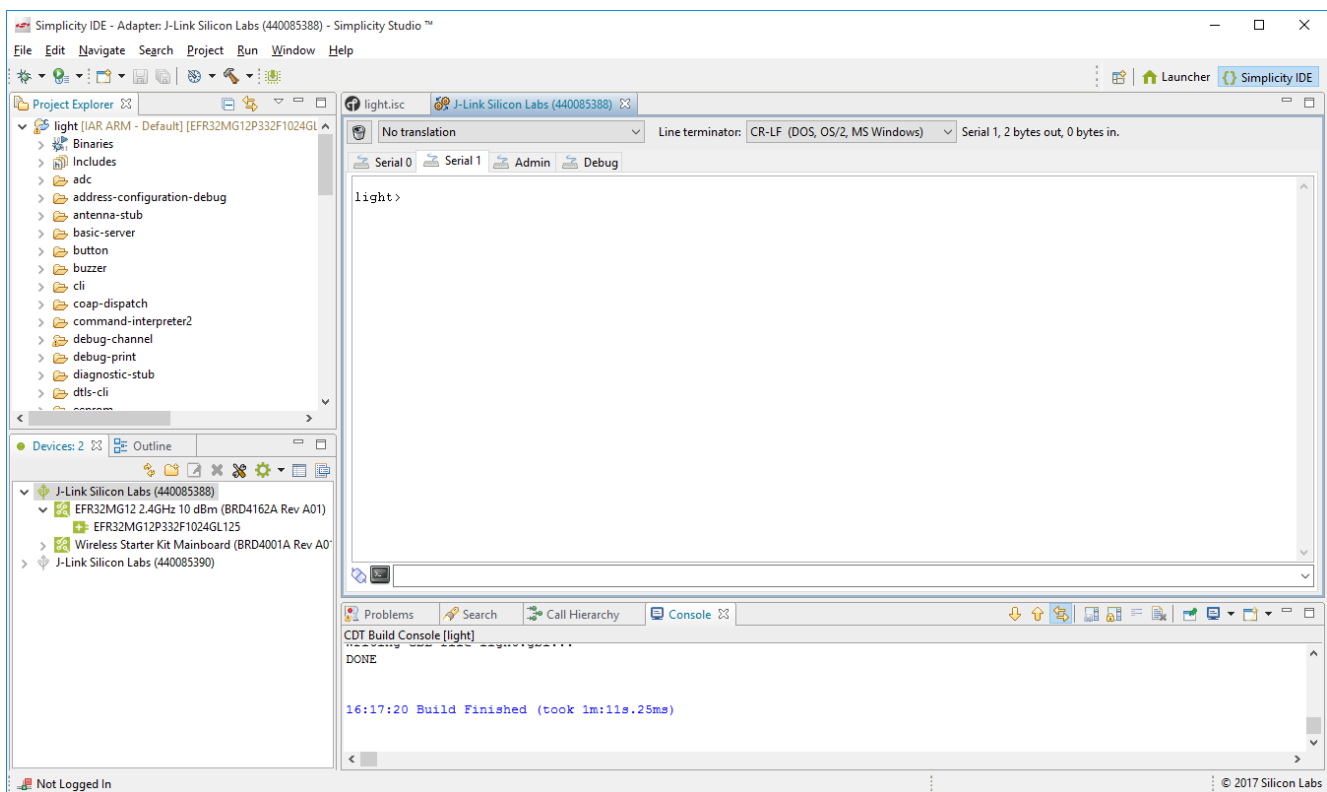
OK

Cancel

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 slowly turning on and off.



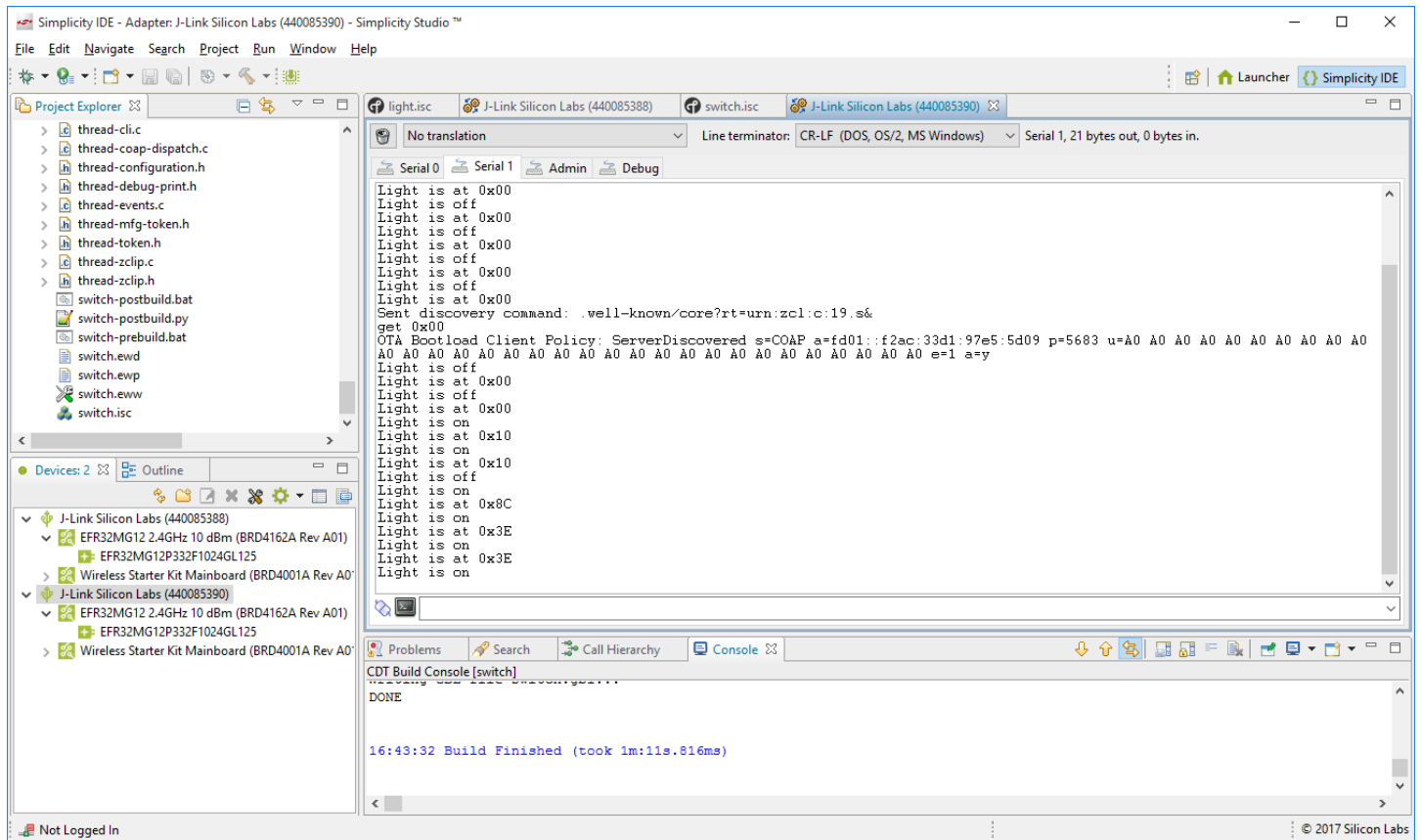
8. You can also right-click on the device and select **Launch Console**. In the console window, click the **Serial 1** tab, and press enter. You should see a prompt that corresponds to the project name. Note that the icons next to the device are now green, indicating a serial connection to the console.



Now repeat the procedure for the Switch example, by clicking **Launcher** in the upper right, and following the steps beginning in section **Selecting an Example Application**. Like the Light example, after successful start the Switch example turns LED1 on and off, and you can see the name of the project in a prompt on the Serial1 tab of a connected console.

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

The switch console displays feedback from the light example application.

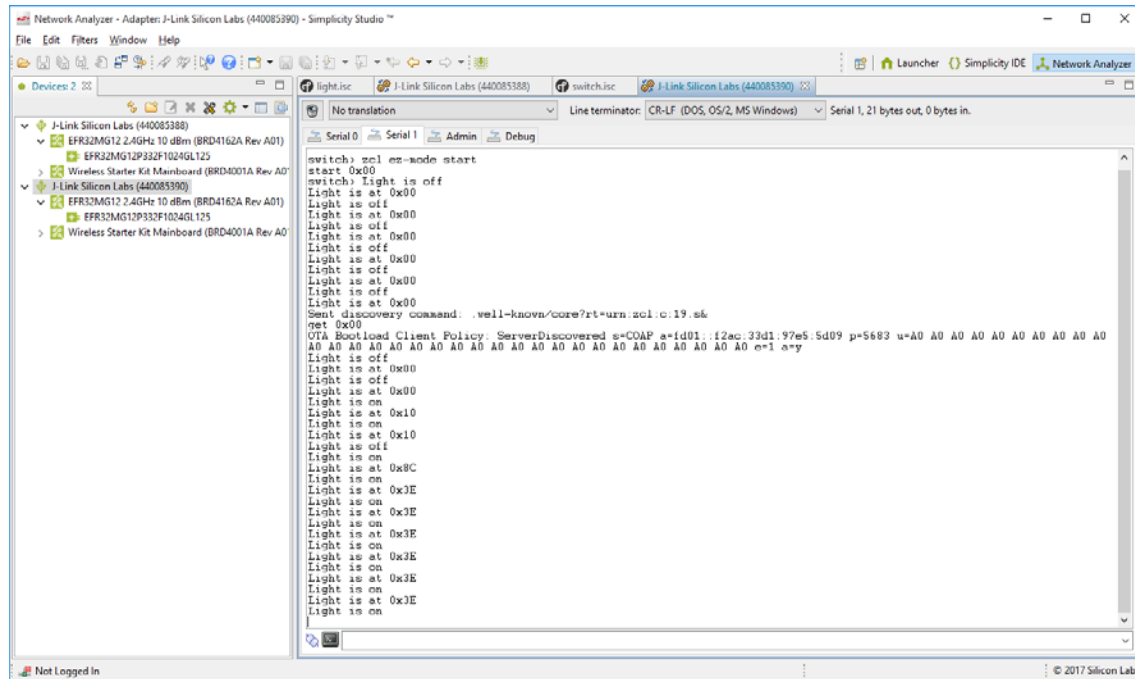


The light and switch sample applications both contain OTA Bootload Client support, meaning that they support over-the-air upgrades by way of the OTA Bootload Cluster. Both the light and switch will automatically start OTA Bootload operations once on a network. The switch searches for an OTA Bootload server, while the light acts as an OTA Bootload server and responds to discovery queries. For more information on the ZCL/IP OTA Upgrade, see *UG278: Zigbee Cluster Library over IP (ZCL/IP) User's Guide*.

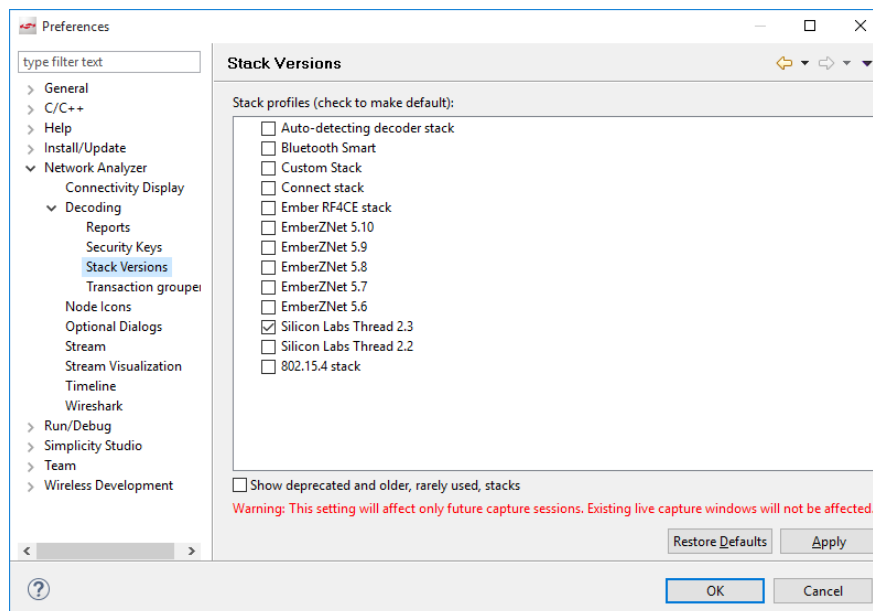
7 Using the Network Analyzer

Now that your network is functioning, you can evaluate the data being transmitted using the Network Analyzer tool.

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 **File > Preferences > Network Analyzer > Decoding > Stack Versions**, verify it is set to the correct stack version, not to Autodetecting. If you need to change it, click the correct stack, click **Apply**, then **OK**.

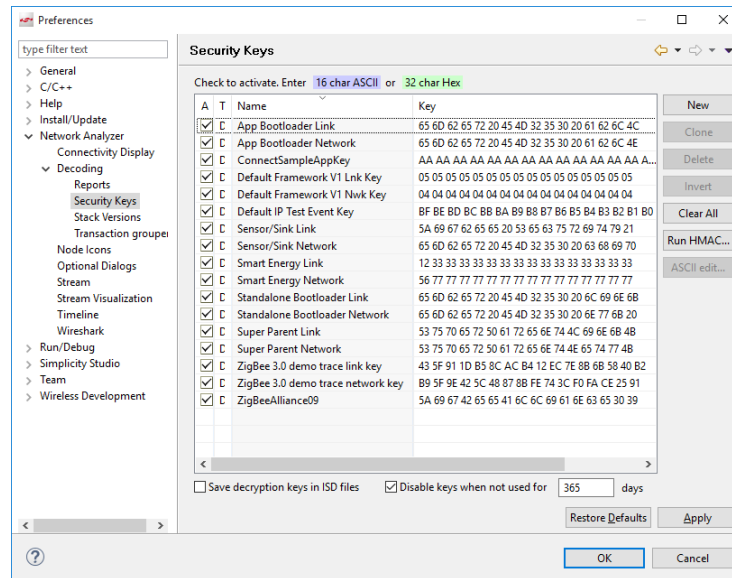


- To make sure that packets decode correctly, enter the Master Key for the Thread network. For security reasons, this key cannot be retrieved at runtime from the Thread stack, so it must be known to the developer through other means. While Thread applications deployed into the field are expected to use a randomly generated Master Key when starting the network, these Switch and Light examples applications use a hardcoded Master Key that can be found in the switch-implementation.c or light-implementation.c files as shown below:

```
static const EmberKeyData masterKey = {
    { 0x65, 0x6D, 0x62, 0x65, 0x72, 0x20, 0x45, 0x4D,
      0x32, 0x35, 0x30, 0x20, 0x63, 0x68, 0x69, 0x70, },
};
```

This key may already have been pre-loaded into Network Analyzer as the “Sensor/Sink Network” key, but you can use the instructions below to verify the existence of the key or enter it if it doesn’t already appear in the list of available keys:

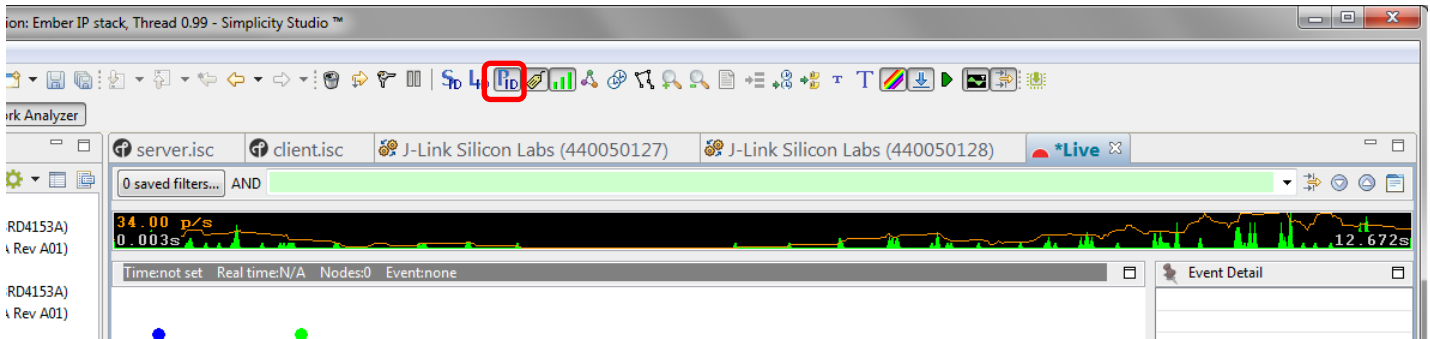
- Select **File > Preferences > Network Analyzer > Decoding > Security Keys**.



- Check if the desired key is already in the list. Note that the “Key” column header can be clicked to sort the list alphanumerically by the Key field.
 - If the key does already exist, proceed to Step 6 below to ensure the key is enabled for use.
 - Click **New**.
 - Name the new entry, and enter the hexadecimal bytes of the key into it, omitting any non-whitespace separator characters as well as any “0x” prefix.
 - Ensure that the key is active for decoding by enabling the A (Active) checkbox on the same row as the key.
 - Click **Apply**.
 - Click **OK** to leave.
- Right-click on the Light device, and select **Start Capture**. Do the same for the Switch device.


5. 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 filter the map view:

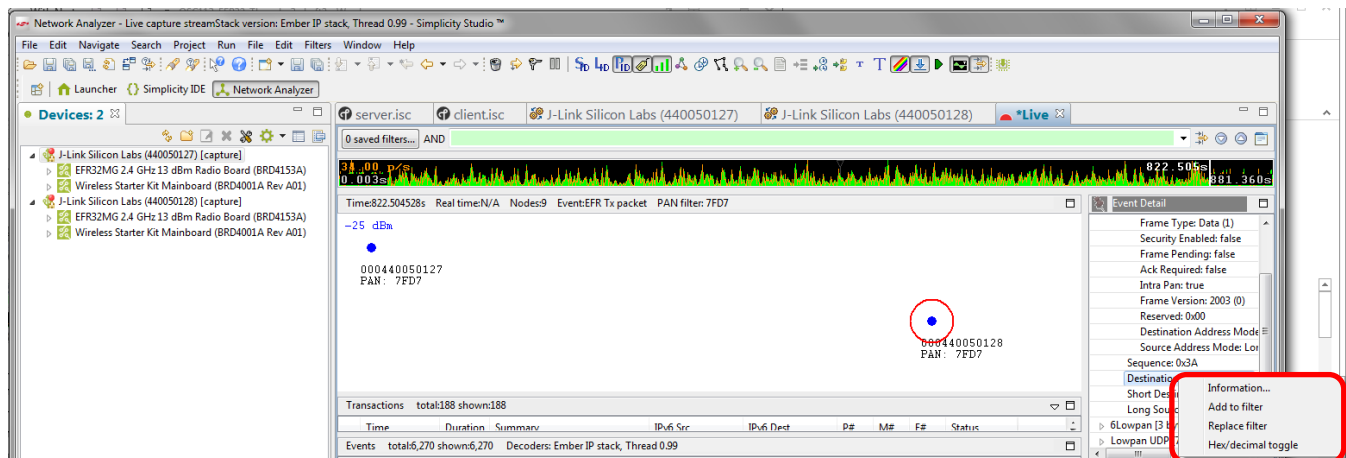
1. Click on the map. On the toolbar, click the PAN ID button:



2. Look for the dot in the map that contains the ISA3 adapter name or WSTK name or J-Link serial number just below it that matches that of the debug adapter to which your Switch device is connected. This is an indication that the dot in question represents the Switch device.
3. Right-click on the representation of your Switch device and select **Show only this PAN**.

To filter transactions:

1. In the Events frame, click one of the MLE Advertise events (in blue).
2. In the Event Detail, scroll down until you see the Destination PAN ID
3. Right-click on it, and select **Add to filter**.
4. Apply the filter by clicking  next to the green filter expression field.



6. You should see the over-the-air message being captured live in the Transactions window:

The screenshot shows the Network Analyzer interface with the following components:

- Menu Bar:** File, Edit, Filters, Window, Help.
- Toolbar:** Various icons for file operations, settings, and analysis.
- Device List (Left):** Shows two devices: J-Link Silicon Labs (440085388) [studio] and J-Link Silicon Labs (440085390) [capture].
- Map View (Center):** Displays a network topology with two nodes. A red circle highlights the node 000440085388.
- Event Detail (Right):** Shows details for the selected event, including MLE crypto, IEEE 802.15.4, 6LoWPAN, and Message Integrity Code.
- Transactions (Right):** A table showing network transactions with columns for Time, Duration, Summary, IPv6 Src, IPv6 Dest, P#, M#, and E#.
- Events (Right):** A table showing network events with columns for Time, Type, Summary, MAC Src, MAC D..., and Status.

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**.

Node label

Enter label:

☐ From beginning

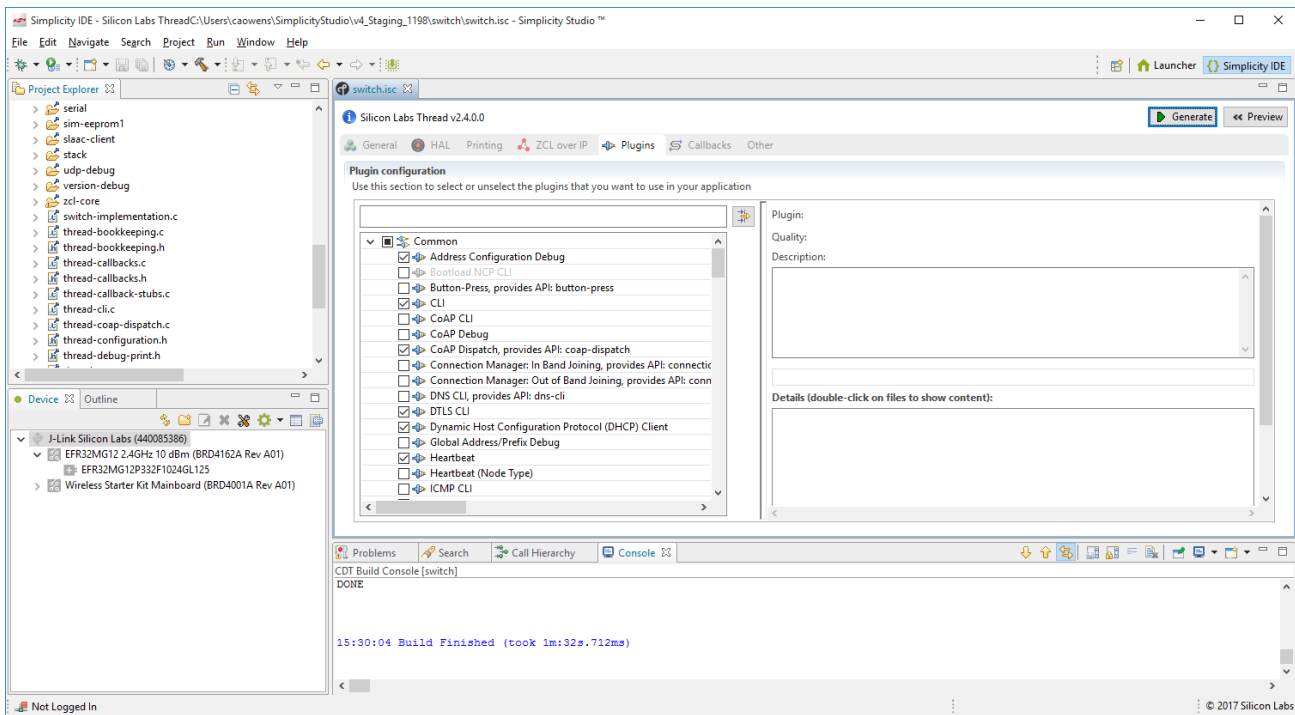
☒ Starting at time: 694.818692

OK

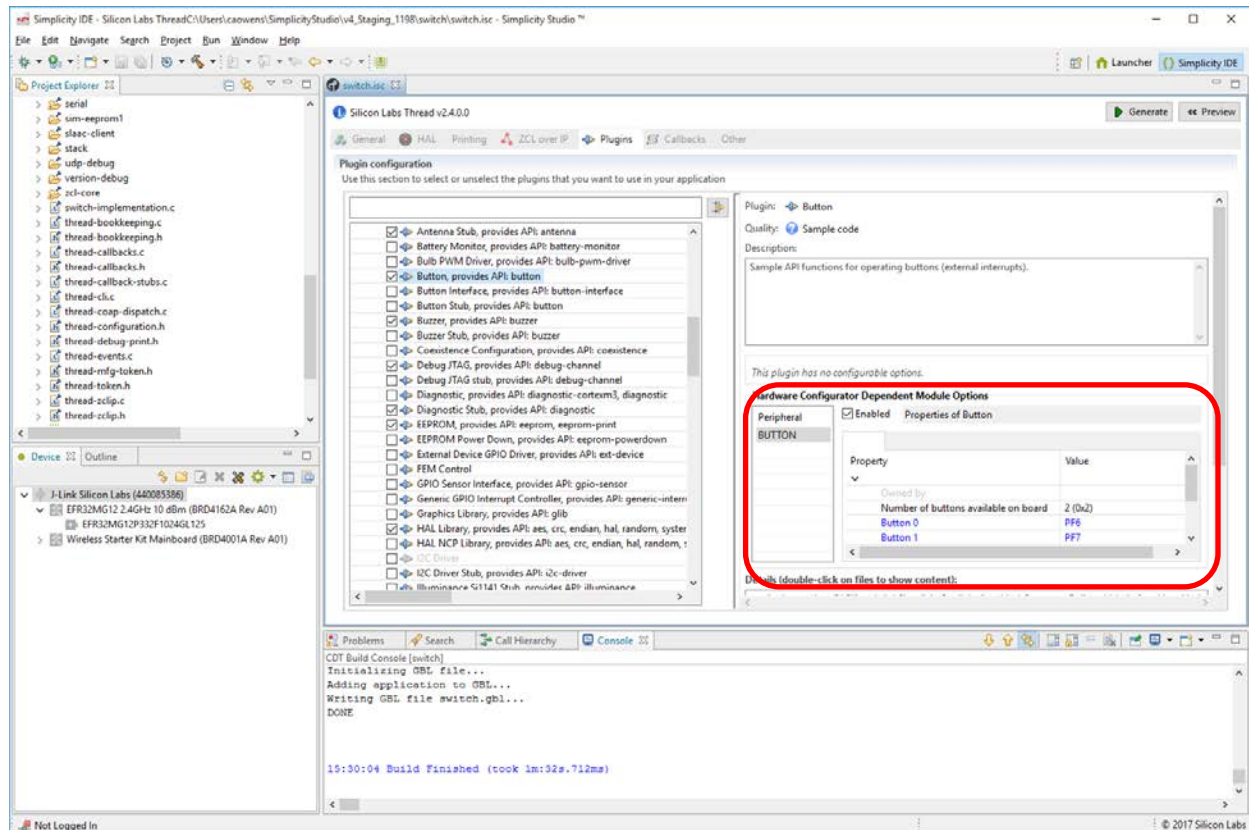
Cancel

8 Next Steps

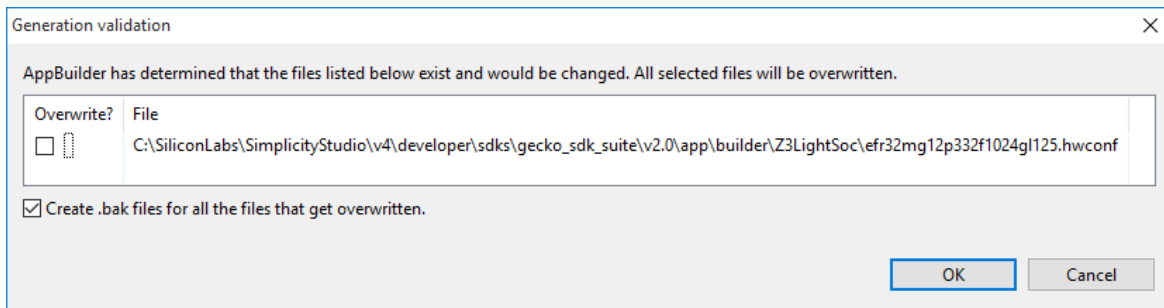
Explore configuring the example application to meet your needs. Much of the software configuration can be done through plugins. Select a plugin to see more information about it and its configuration options, if any.



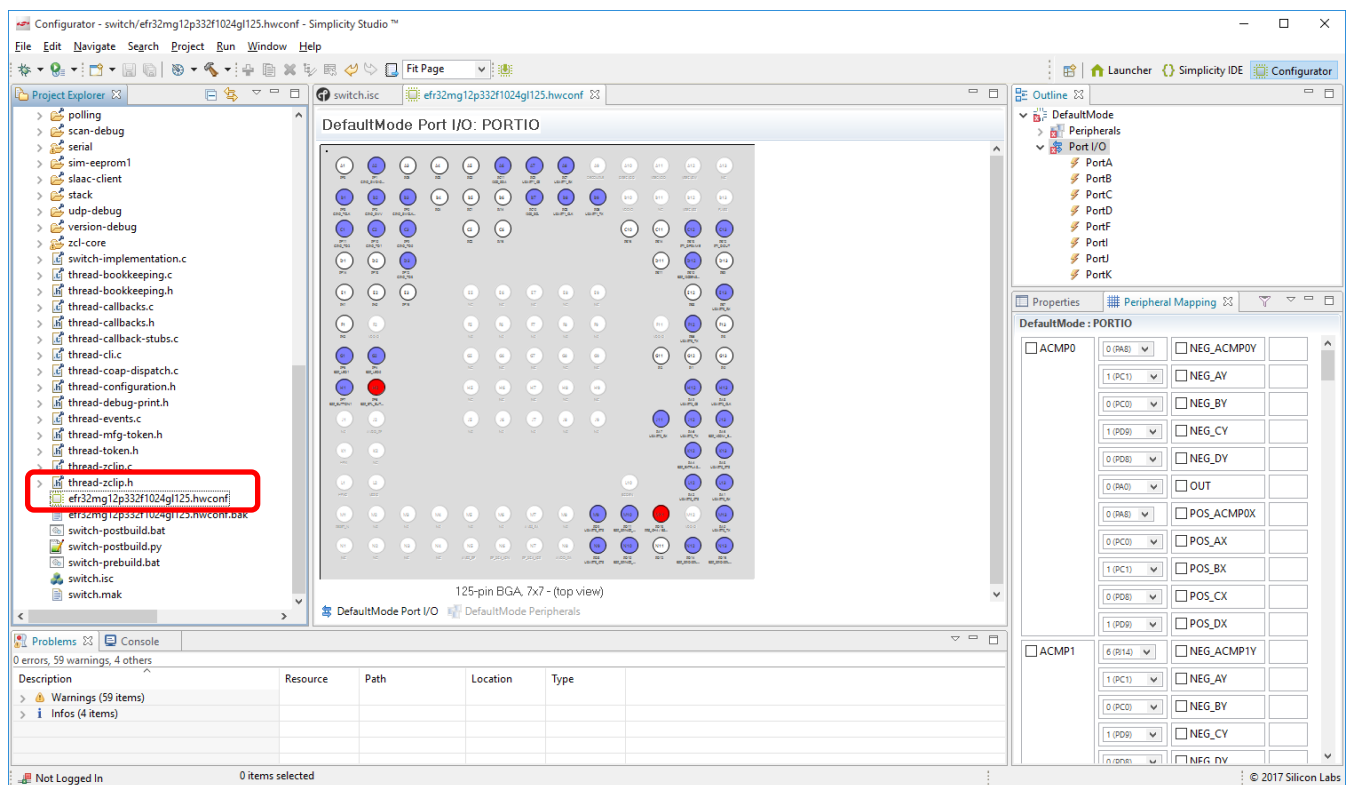
Simplicity Studio offers a Hardware Configurator tool that allows you to easily configure new peripherals or change the properties of existing ones. Hardware Configurator options are available on many of the HAL plugins.



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.



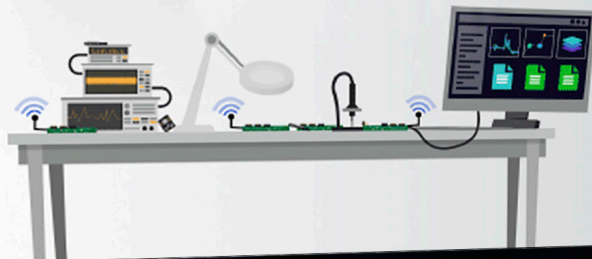
You can also open the Hardware Configurator tool by double-clicking on the .hwconf file that was generated along with your other project files, or by clicking Open Hardware Configurator on the HAL tab. and open the Hardware Configurator tool. Click **DefaultMode Peripherals** under the pin graphic to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



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.

Silicon Labs

Simplicity Studio™4



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 and limitation to product information, specifications, and descriptions herein, and does not give warranties as to the accuracy or completeness of the included information. Silicon Labs shall have no liability for the consequences of use of the information supplied herein. This document does not imply or express copyright licenses granted hereunder to design or fabricate any integrated circuits. The products are not designed or authorized to be used within any Life Support System 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.

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®, ISOModem®, Micrium, Precision32®, ProSLIC®, Simplicity Studio®, SiPHY®, Telegesis, the Telegesis Logo®, USBXpress®, Zentri 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. 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>