QSG170: Silicon Labs OpenThread Quick-Start Guide

This document describes how to get started with OpenThread development using the Silicon Labs OpenThread software development kit (SDK) and Simplicity Studio® 5 with a compatible wireless starter kit.

KEY POINTS

- About the Silicon Labs OpenThread SDK
- About Example Applications and Demos
- Getting started with development
- Next steps
- Development tools

1 Introduction

Google's OpenThread is an open-source implementation of Thread. Google has released OpenThread to make the networking technology used in Google Nest products more broadly available to developers, in order to accelerate the development of products for the connected home and commercial buildings.

With a narrow platform abstraction layer and a small memory footprint, OpenThread is highly portable. It supports both system-on-chip (SoC), network co-processor (NCP), and radio co-processor (RCP) designs. OpenThread implements all features defined in the Thread 1.3.0 specification (backward compatible with both the 1.1 and 1.2 specifications). This specification defines an IPv6-based reliable, secure, and low-power wireless device-to-device communication protocol for home and commercial building applications.

This guide describes how to get started developing OpenThread applications using the Silicon Labs OpenThread SDK, delivered with the Gecko SDK, and Simplicity Studio 5. Simplicity Studio 5 (SSv5) includes everything needed for IoT product development with Silicon Labs devices, including a resource and project launcher, software configuration tools, full IDE with GNU toolchain, and analysis tools. This document focuses on use and development in the Simplicity Studio 5 environment. Alternatively, the Gecko SDK may be installed manually by downloading or cloning the latest from GitHub. See https://github.com/SiliconLabs/gecko_sdk for more information.

1.1 About the Silicon Labs OpenThread SDK

The Silicon Labs OpenThread SDK is based on the Gecko Platform component-based design, where each component provides a specific function. Components are made up of a collection of source files and properties. The component-based design enables customization by adding, configuring, and removing components. The application developer can use SSv5's Project Configurator and Component Editor to easily assemble the desired features by including those components that match the required functionality and by configuring the various properties associated with those components.

The Silicon Labs OpenThread SDK is based on the OpenThread stack available in GitHub but with a number of enhancements, including support of additional platforms and functionality, additional example applications, and additional metadata to allow for the seamless integration into SSv5.

Platforms and functionality: Silicon Labs has enhanced the OpenThread source code and the platform abstraction layer supporting the EFR32 platform in order to offer additional features provided by the Gecko Platform, such as:

- Wi-Fi coexistence
- Antenna diversity
- FreeRTOS support
- Non-Volatile Memory storage support (NVM3)
- Hardware acceleration with MbedTLS
- Multi-PAN Radio Co-Processor (RCP)
- Co-Processor Communication support (CPC)
- PSA Vault support
- Proprietary Sub-GHz
- Power manager support

Example applications: The example applications from GitHub are included in the SDK, along with new example applications showcasing additional functionality, such as OpenThread/Bluetooth Low Energy Dynamic Multiprotocol support. These examples are described in section 2 About Example Applications and Demos.

The Silicon Labs OpenThread SDK contains the complete OpenThread GitHub directory structure but does not use the makefile build system provided by the GitHub solution. Instead, the Silicon Labs OpenThread SDK uses the build system provided by SSv5. The GitHub makefile build options have been made available as component configuration options and are discussed in section 3.2 Configuring the Project. When using the Silicon Labs OpenThread SDK to build a Host-RCP application (such as an OpenThread Border Router), make sure to use compatible RCP and host code from the same SDK.

See the release notes available in Simplicity Studio and through docs.silabs.com for a description of the special features and the OpenThread stack version used for the release.

1.1.1 Simplicity Studio 5 (SSv5)

SSv5 is the core development environment designed to support the Silicon Labs IoT portfolio of system-on-chips (SoCs) and modules. It provides access to target device-specific web and SDK resources; software and hardware configuration tools; an integrated development

environment (IDE) featuring industry-standard code editors, compilers and debuggers; and advanced, value-add tools for network analysis and code-correlated energy profiling.

SSv5 is designed to simplify developer workflow. It intelligently recognizes all Silicon Labs evaluation and development kit parts and, based on the selected development target, presents appropriate software development kits (SDKs) and other development resources.

The Silicon Labs OpenThread SDK is downloaded through SSv5. The GNU Compiler Collection (GCC) is provided with SSv5. Other important development tools provided with SSv5 are introduced in section 5 Development Tools.

1.1.2 Gecko Bootloader

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: Bootloader Fundamentals*.

1.1.3 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. For more information about Gecko Platform, see release notes that can be found in SSv5's Documentation tab, as well as online API documentation in https://docs.silabs.com/.

1.2 Prerequisites

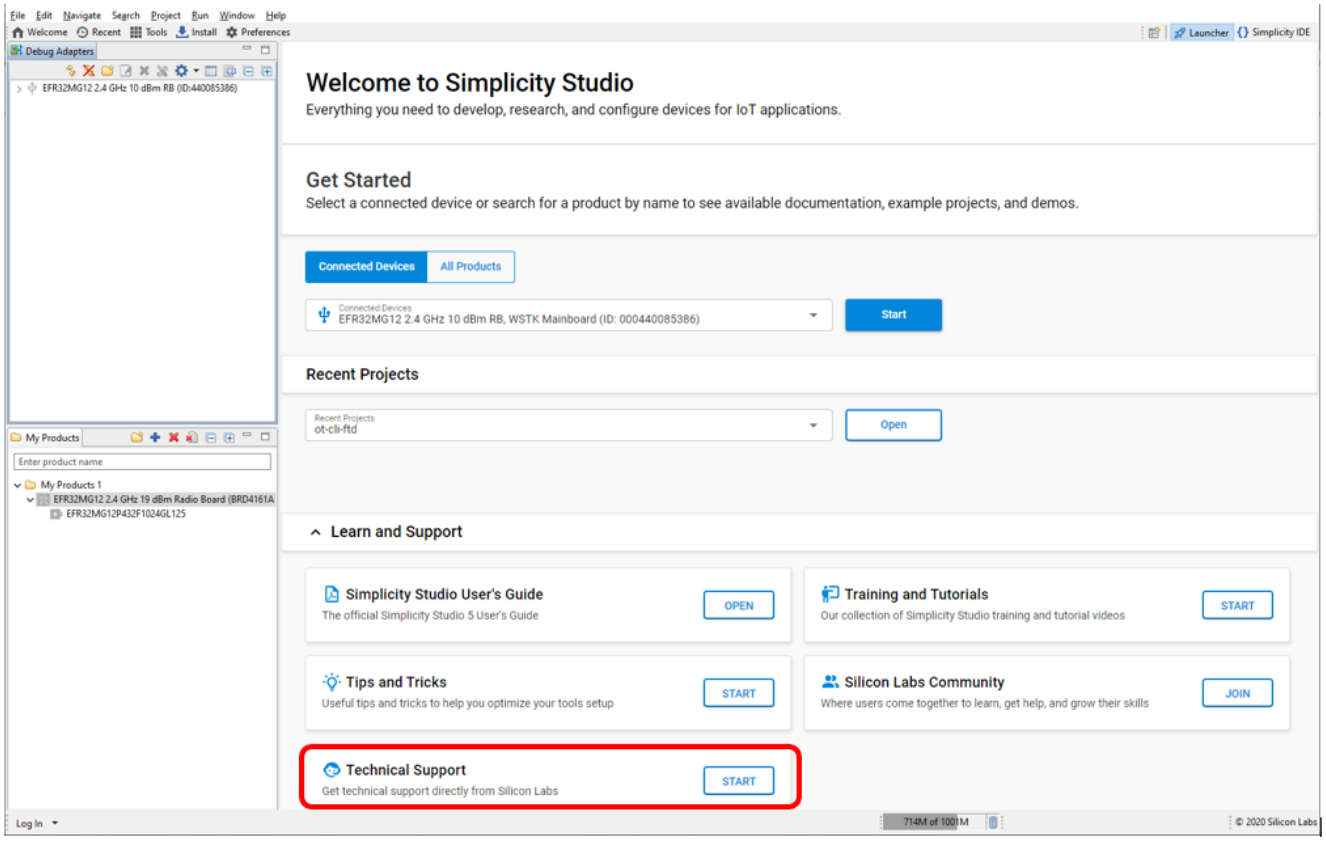
Before following the procedures in this guide you must have:

- Purchased one of the Wireless Gecko (EFR32) Portfolio Wireless Kits.
- Downloaded SSv5 and the Gecko SDK, which includes the Silicon Labs OpenThread SDK, and be generally familiar with the SSv5 Launcher perspective. SSv5 installation and getting started instructions along with a set of detailed references can be found in the online *Simplicity Studio 5 User's Guide*, available on https://docs.silabs.com/ and through the SSv5 help menu.

For a compiler, use the version of the GNU ARM toolchain installed with the Silicon Labs OpenThread SDK. The IAR Embedded Workbench for ARM (IAR EWARM) Compiler is currently not supported with OpenThread.

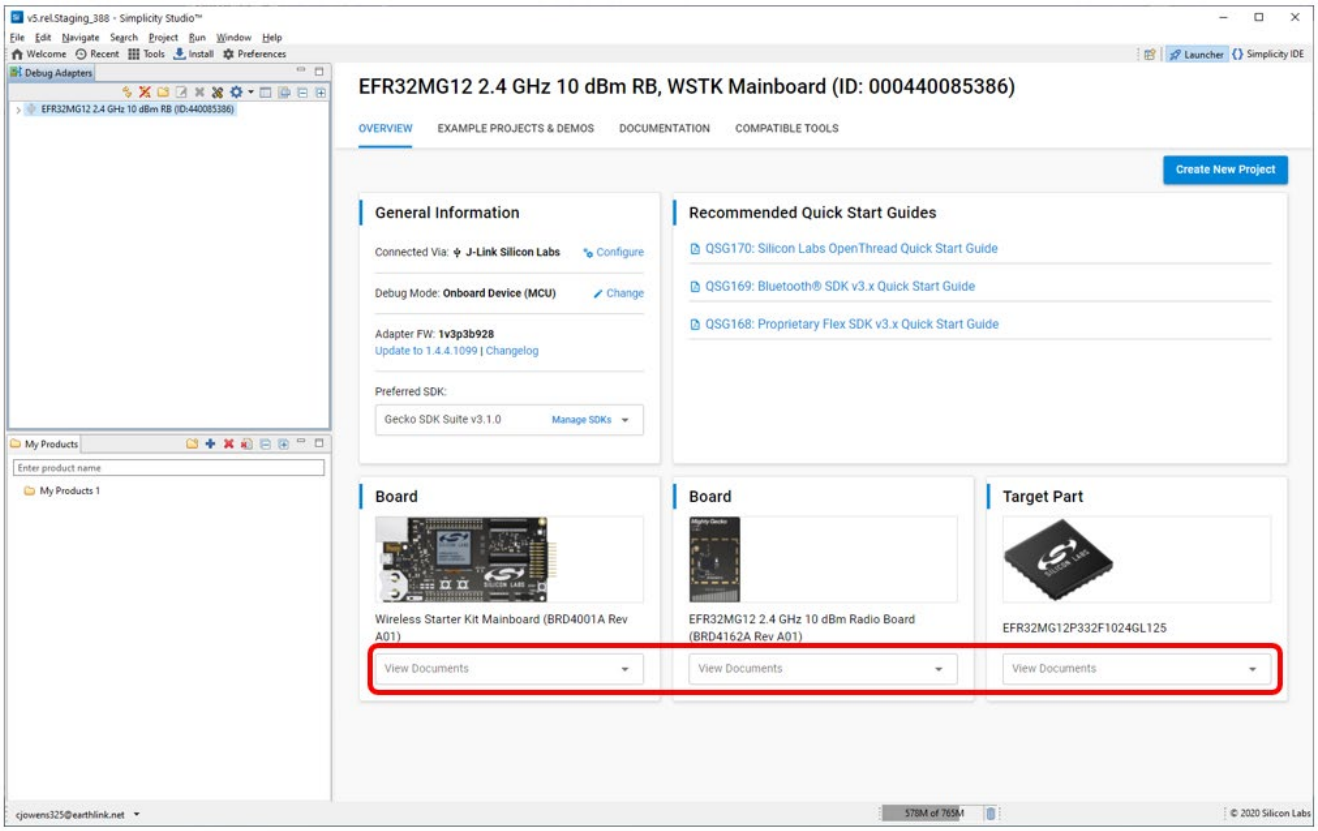
1.3 Support

You can access the Silicon Labs support portal at https://www.silabs.com/support through SSv5's Welcome view under Learn and Support. Use the support portal to contact Customer Support for any questions you might have during the development process.

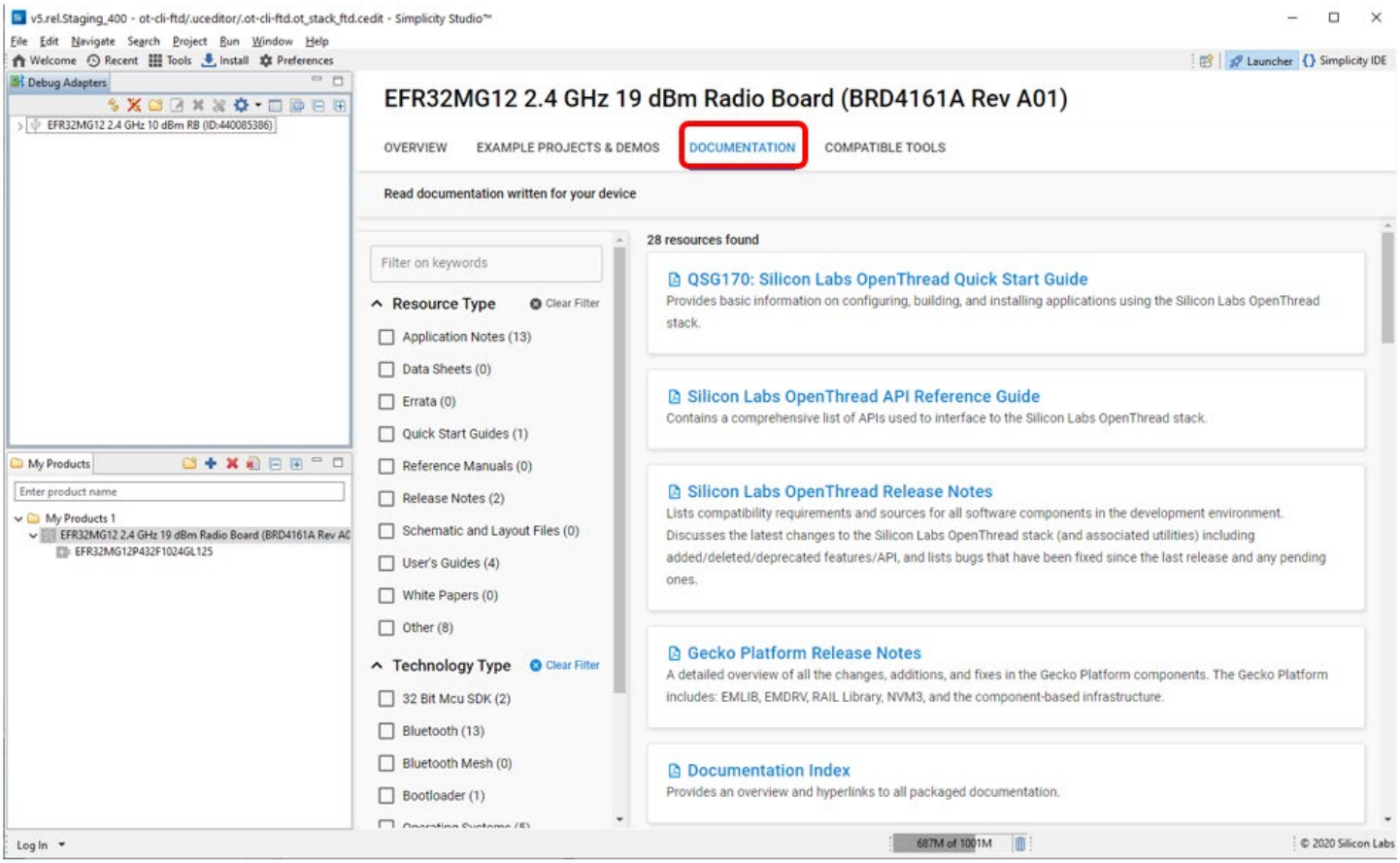


1.4 Documentation

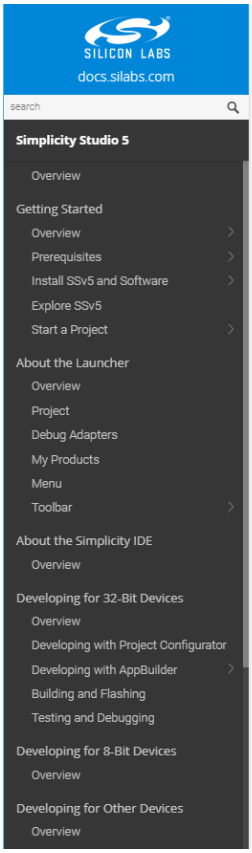
Relevant documentation is available through SSv5. It is filtered based on your selected part. Hardware-specific documentation can be accessed through links on the part OVERVIEW tab.



SDK documentation and other references are available through the DOCUMENTATION tab. Filter with the Thread Technology Type checkbox to see documentation most closely related to the OpenThread SDK.

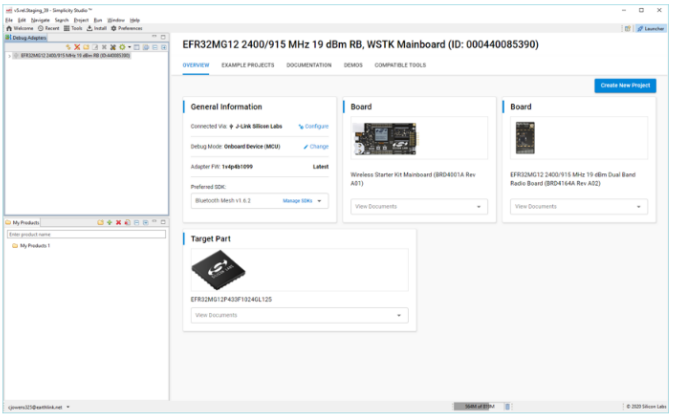


SSv5 and its tools are documented in the online [Simplicity Studio 5 User's Guide](#).



Simplicity Studio® 5 User's Guide

Simplicity Studio is the core development environment designed to support the Silicon Labs IoT portfolio of system-on-chips (SoCs) and modules. It provides access to target device-specific web and SDK resources; software and hardware configuration tools; an integrated development environment (IDE) featuring industry-standard code editors, compilers and debuggers; and advanced, value-add tools for network analysis and code-correlated energy profiling.



Simplicity Studio is designed to simplify developer workflow. It intelligently recognizes all evaluation and development kit parts released by Silicon Labs and, based on the selected development target, presents appropriate software development kits (SDKs) and other development resources.

Simplicity Studio 5 (SSv5) focuses on developer experience, leveraging feedback from customers, employees and competitive reviews. Developers of all experience levels will benefit from an optimized workflow that supports them through the development journey and produces quicker project progression and device configuration.

The SSv5 User Guide pages are organized into the following groups. The first three are generally applicable. They are followed by groups specific to different target devices. Finally, a reference to the tools included with SSv5 is provided.

- Getting Started describes how to install SSv5 and the software components relevant to the target device, and provides general overviews to the SSv5

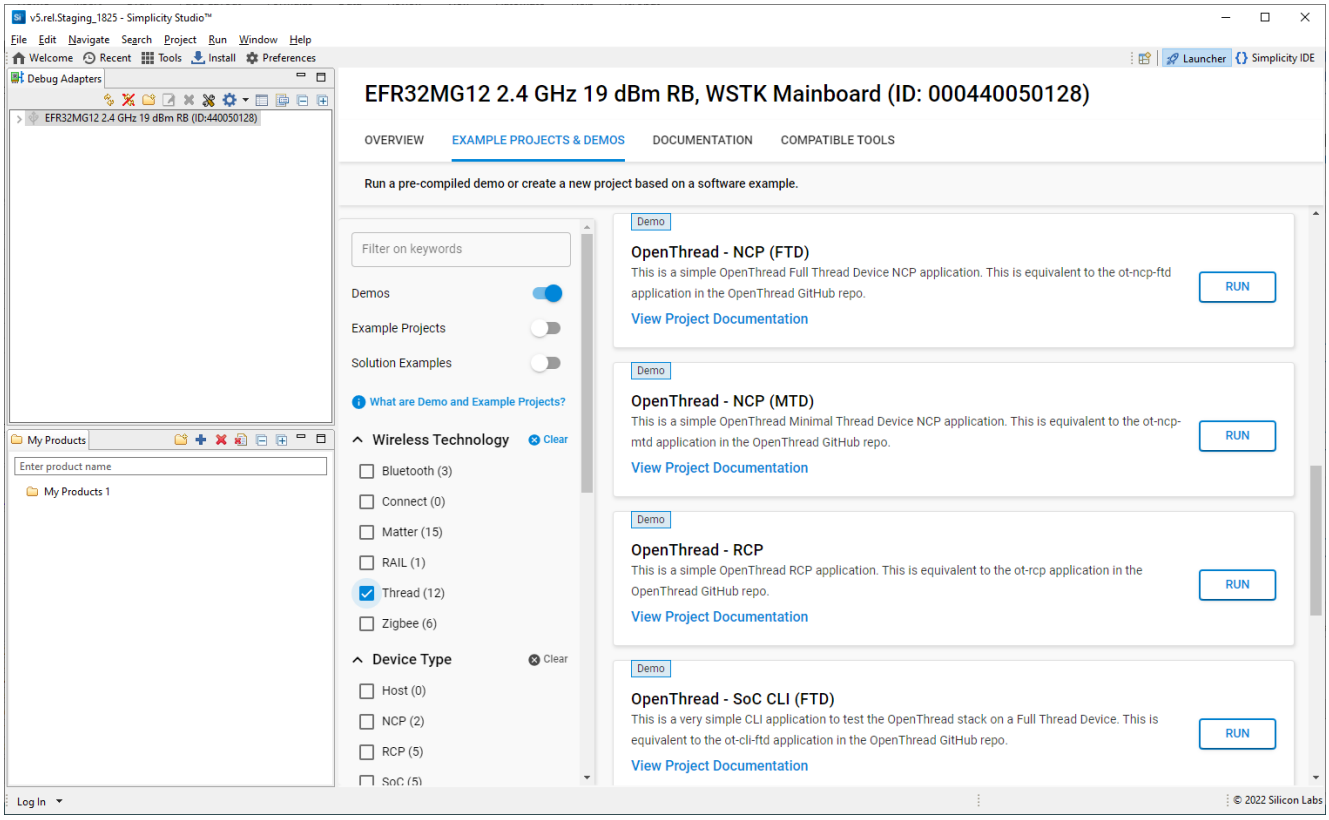
2 About Example Applications and Demos

Because starting application development from scratch is difficult, the Silicon Labs OpenThread SDK comes with a number of built-in example applications and demos covering the most frequent use cases and are preconfigured code designed to illustrate common application functions. Silicon Labs strongly recommends starting development from one of the example applications.

Like everything in SSv5, the examples and the demos shown on the EXAMPLE PROJECTS & DEMOS tab are filtered based on the part you have connected or selected.

2.1 Demos

Demos are prebuilt firmware images that are ready to download to a compatible device. The quickest way to find if a demo is available for your part is by adding the part or board information in the My Products view and then navigating to the EXAMPLE PROJECTS & DEMOS tab in the Launcher Perspective. Turn off the Example Projects filter. The Solution Examples filter is provided for future use.



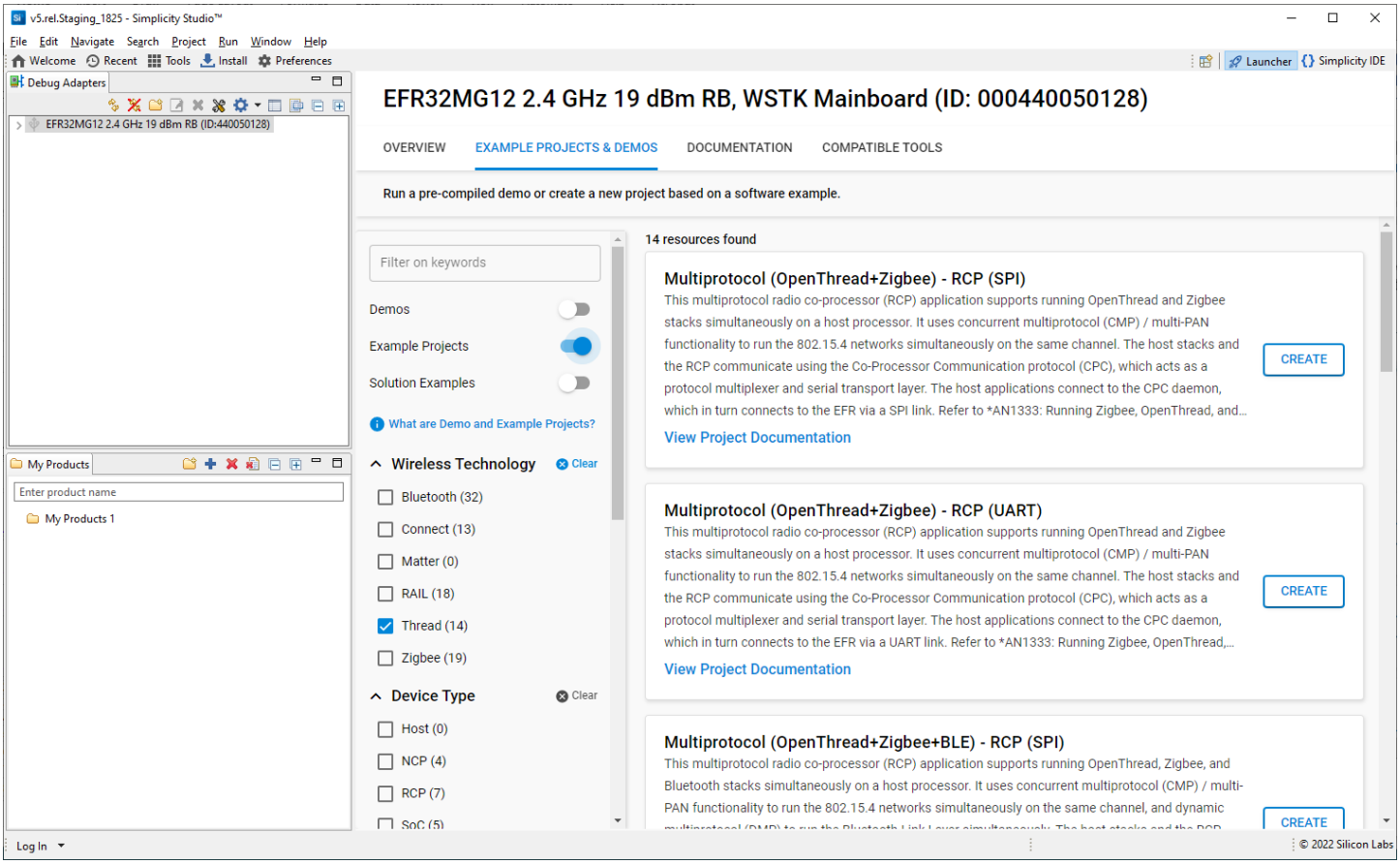
Precompiled demo application images provided with the OpenThread SDK are compatible with the following boards:

- brd4161a
- brd4166a
- brd4168a
- brd4180a
- brd4304a

Please refer to the Silicon Labs OpenThread SDK release notes for a full list of supported parts.

2.2 Software Examples

Since typically you will finish by flashing a compiled application image to a device, connect a device to your computer and select it in the Debug Adapters view. In the EXAMPLE PROJECTS & DEMOS tab on the Launcher perspective, filter the example projects based on the 'Thread' technology type and turn off **Demos.**



The example applications provided with the Silicon Labs OpenThread SDK are as follows. Most of the examples come in two variants, Full Thread Device (FTD) and Minimal Thread Device (MTD), and are geared towards System-on-Chip (SoC), Network Co-Processor (NCP), and Radio Co-Processor (RCP) application models.

In general, the example applications are simple examples illustrating the following functionality:

- Testing with a CLI
- Network Co-Processor (NCP)
- Radio Co-Processor (RCP)
- Sleepy device
- Dynamic multiprotocol
- Multi-PAN RCP
- Multi-PAN RCP and CPC over SPI/UART

OpenThread – SoC CLI (FTD): A CLI application equivalent to the ot-cli-ftd application in the OpenThread GitHub repo to test the OpenThread stack on a Full Thread Device (FTD) for SoC designs. This application can be used to test the OpenThread stack against the Thread Test Harness for interoperability testing.

OpenThread – SoC CLI (MTD): A CLI application equivalent to the ot-cli-mtd application in the OpenThread GitHub repo to test the OpenThread stack on a Minimal Thread Device (MTD) for SoC designs.

OpenThread – NCP (FTD): An OpenThread Full Thread Device (FTD) application equivalent to the ot-ncp-ftd application in the Open-Thread GitHub repo for NCP designs. While the application layer is on the host processor, this application runs the OpenThread core on the 802.15.4 SoC.

OpenThread – NCP (MTD): AnOpenThread Minimal Thread Device (MTD) application equivalent to the ot-ncp-mtd application in the OpenThread GitHub repo for NCP designs. Similar to ot-ncp-ftd, this application runs the OpenThread core on the 802.15.4 SoC

OpenThread – RCP: An OpenThread Radio Co-Processor application equivalent to the ot-rcp application in the OpenThread GitHub repo. While the application layer and the OpenThread core is on the host processor, this application only runs the minimal OpenThread controller on the 802.15.4 SoC. This RCP software example must be used with a compatible OpenThread Border Router. This can be built from the same Silicon Labs OpenThread SDK or deployed from a docker image. For more details, refer to *AN1256: Using the Silicon Labs RCP with the OpenThread Border Router*.

OpenThread – SoC Sleepy Demo (FTD): An application to start and form a Thread network on a Full Thread Device (FTD) for the sleepy-demo. This application is used in conjunction with the **OpenThread – SoC Sleepy Demo (MTD)** app.

OpenThread – SoC Sleepy Demo (MTD): An application to demonstrate Sleepy End Device (SED) behavior on a Minimal Thread Device (MTD) that attaches to a Thread network started by a node running the **OpenThread – SoC Sleepy Demo (FTD)** app. This application demonstrates power manager feature support and deep sleep (EM2) mode for EFR32.

OpenThread – SoC Synchronized Sleepy Demo (SSED): An application to demonstrate Synchronized Sleepy End Device (SSED) behavior using Co-ordinated Sampled Listening (CSL) that attaches to a Thread network started by a node running the **OpenThread – SoC Sleepy Demo (FTD)** app. This application demonstrates power manager feature support and deep sleep (EM2) mode for EFR32.

OpenThread BLE DMP – SoC Free RTOS: An application to test Dynamic Multiprotocol (DMP) with OpenThread and Bluetooth running on FreeRTOS. For more details about this application, refer to *AN1265: Dynamic Multiprotocol Development with Bluetooth and OpenThread in GSDK v3.x*.

OpenThread (MultiPan) – RCP (UART): A multipan 802.15.4 RCP (Radio Co-Processor) application. In this application, access to 802.15.4 is provided using SPINEL carried over the CPC (Co-Processor Communication) protocol using a UART connection.

OpenThread (MultiPan) – RCP (SPI): A multipan 802.15.4 RCP (Radio Co-Processor) application. In this application, access to 802.15.4 is provided using SPINEL carried over the CPC (Co-Processor Communication) protocol using a SPI connection.

OpenThread (MultiPan/BLE) – RCP (UART): A multiprotocol and multipan RCP (Radio Co-Processor) application. This application runs multipan 802.15.4 and the Bluetooth Link Layer using DMP (Dynamic MultiProtocol). Access to 802.15.4 is provided using the SPINEL protocol, and access to Bluetooth is provided using the standard Bluetooth HCI (Host Controller Interface) protocol. Both are carried over the CPC (Co-Processor Communication) protocol using a UART connection.

OpenThread (MultiPan/BLE) – RCP (SPI): A multiprotocol and multipan RCP (Radio Co-Processor) application. This application runs multipan 802.15.4 and the Bluetooth Link Layer using DMP (Dynamic MultiProtocol). Access to 802.15.4 is provided using the SPINEL protocol, and access to Bluetooth is provided using the standard Bluetooth HCI (Host Controller Interface) protocol. Both are carried over the CPC (Co-Processor Communication) protocol using a SPI connection.

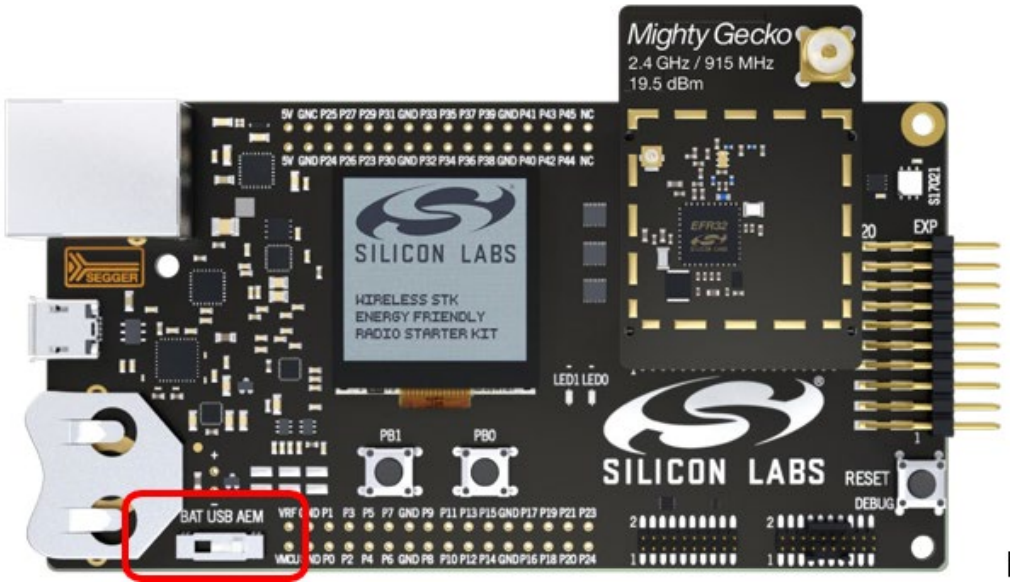
For more details about multi-PAN RCP support, refer to *AN1333: Running Zigbee, OpenThread, and Bluetooth Concurrently on a Linux Host with a Multiprotocol RCP*.

3 Getting Started with Development

This section assumes that you have downloaded SSv5 and the Silicon Labs OpenThread SDK, and are familiar with the features of the SSv5 Launcher perspective.

You should have your mainboard connected.

Note: For best performance in Simplicity Studio 5, be sure that the power switch on your mainboard is in the Advanced Energy Monitoring or “AEM” position, as shown in the following figure.



In these instructions you will compile and load a simple **ot-cli-ftd** application on two nodes. Section 3.4 Creating a Network describes how to use the examples to create a network. Section 5.4 Network Analyzer describes how to use Network Analyzer to observe traffic across the network.

When working with an example application in Simplicity Studio, you will be executing the steps in the following order:

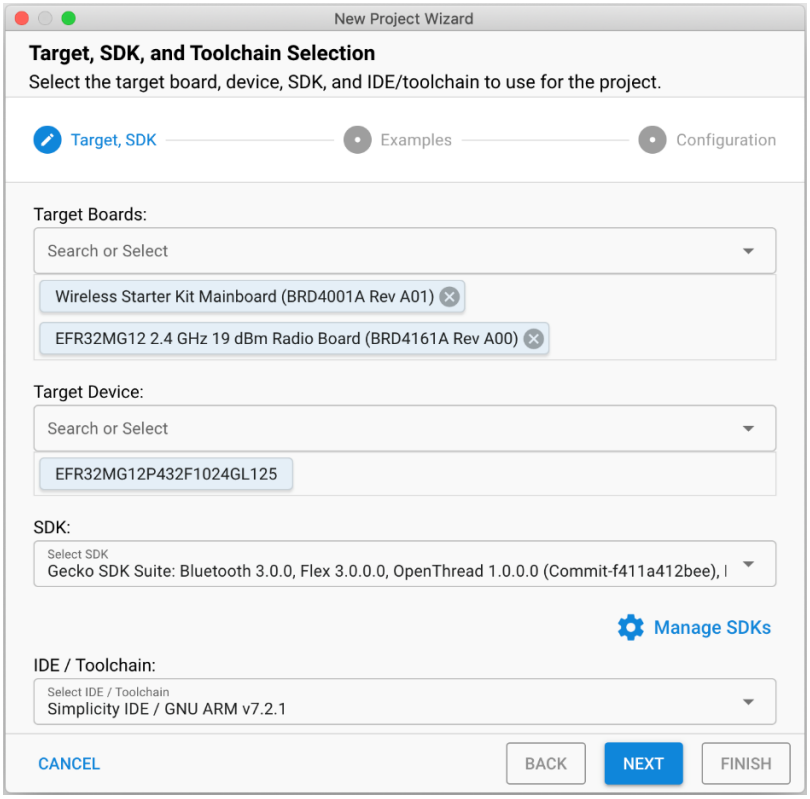
1. Create a project based on an example.
2. Configure the project.
3. Build the application image and flash it to your device

These steps are described in detail in the following sections. These procedures are illustrated for a mainboard with an EFR32MG12. Note: Your SDK version may be later than the version shown in the figures.

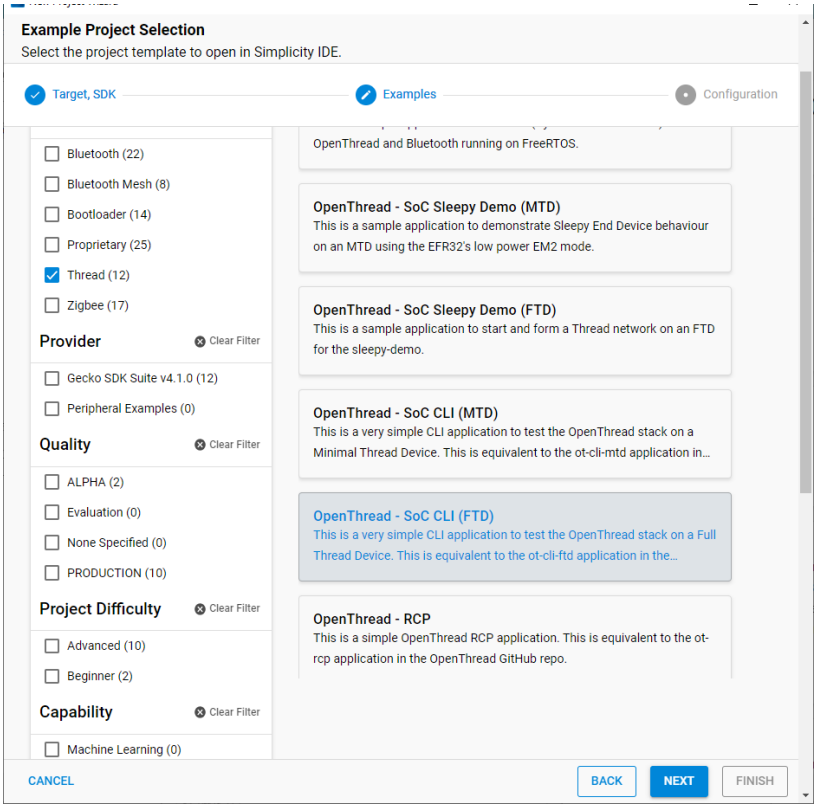
3.1 Creating a Project Based on an Example

Simplicity Studio 5 (SSv5) offers a variety of ways to begin a project using an example application. The online *Simplicity Studio 5 User's Guide*, available both through https://docs.silabs.com/ and the SSv5 help menu, describes them all. This guide uses the **File > New > Silicon Labs Project Wizard** method, because it takes you through all three of the Project Creation Dialogs.

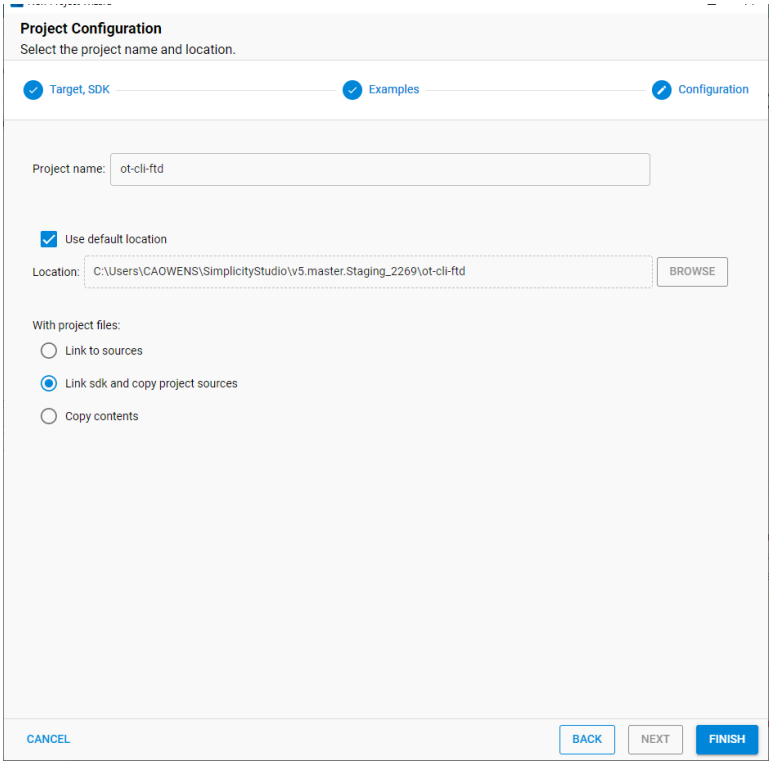
1. Open SSv5's File menu and select **New > Silicon Labs Project Wizard**. The Target, SDK, and Toolchain Selection dialog opens. Do not change the default **Simplicity IDE / GNU** toolchain supported by OpenThread.



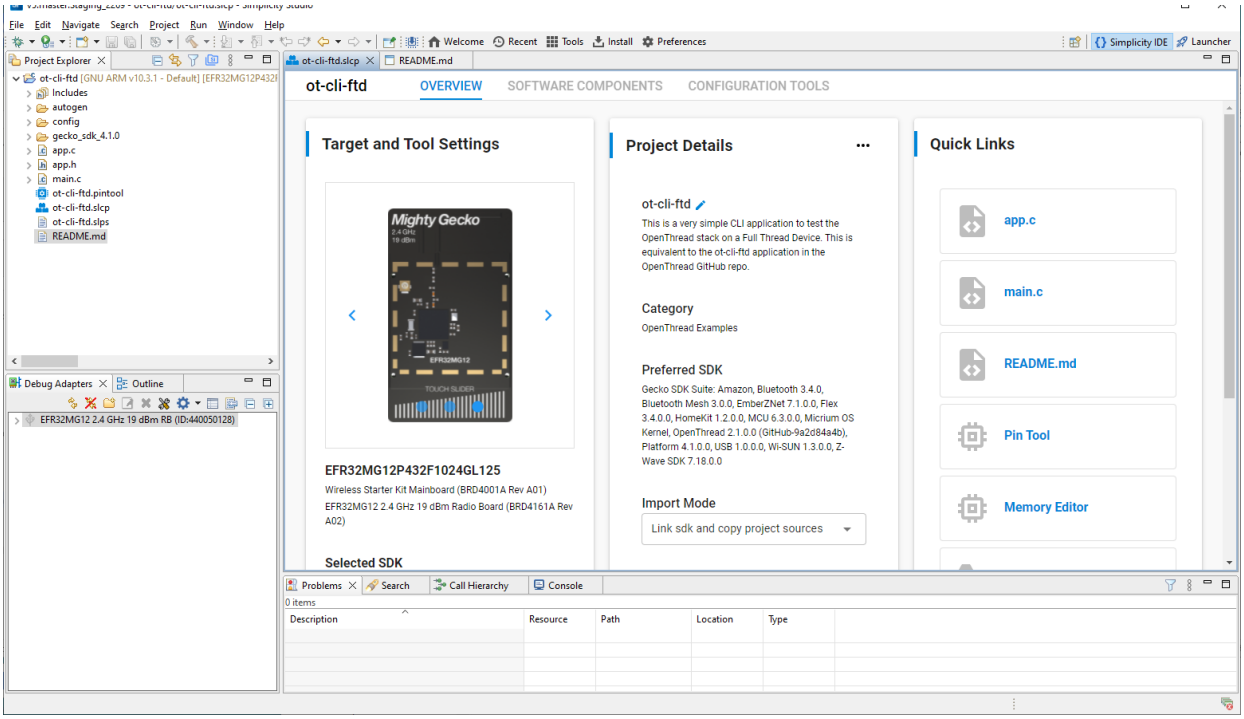
2. The Example Project Selection dialog opens. Use the 'Thread' Technology Type and Keyword filters to search for a specific example, in this case **OpenThread – SoC CLI (FTD)**. Select it and click **NEXT**.



3. The Project Configuration dialog opens. Here you can rename your project, change the default project file location, and determine if you will link to or copy project files. Note that if you change any linked resource, it is changed for any other project that references it. Click **FINISH.**



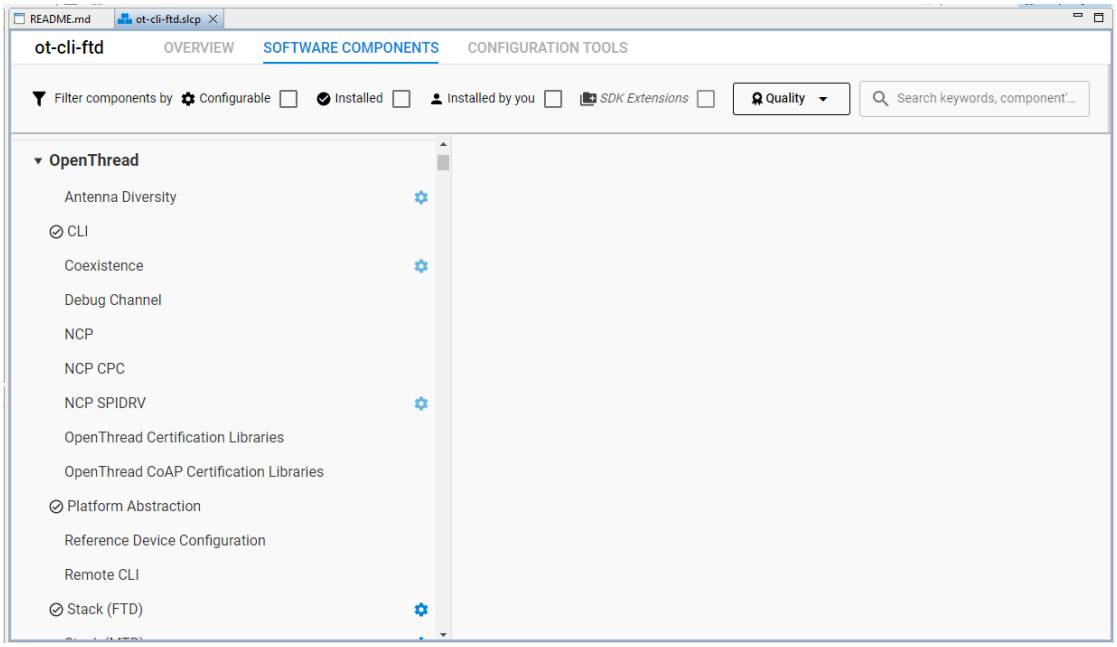
4. The Simplicity IDE Perspective opens with a description of the project on a readme tab. Click the <project>.slcp tab to open the Project Configurator's OVERVIEW tab. See the online *Simplicity Studio 5 User's Guide* for details about the functionality available through the Simplicity IDE perspective and the Project Configurator.



3.2 Configuring the Project

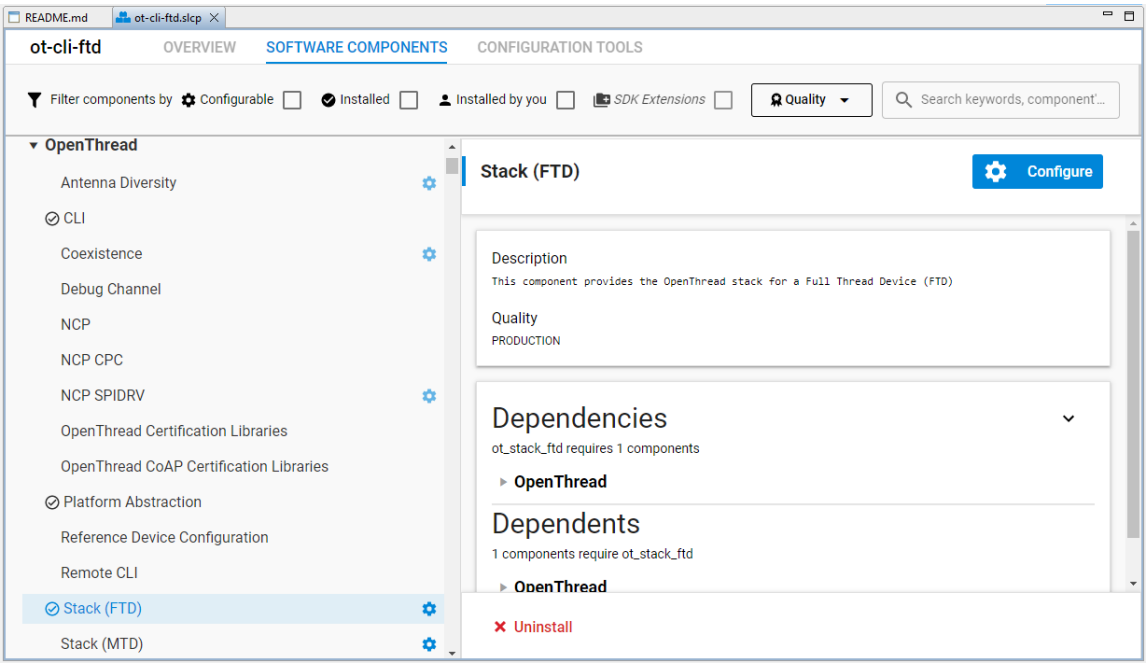
Silicon Labs OpenThread applications are built on a Gecko Platform component structure. Click the SOFTWARE COMPONENTS tab to see a complete list of Component categories.

Note: All EFR32 parts have a unique RSSI offset. In addition, board, antenna and enclosure design can also impact RSSI. When creating a new project, install the **RAIL Utility, RSSI** component. This feature includes the default RSSI Offset Silicon Labs has measured for each part. This offset can be modified if necessary after RF testing of your complete product.



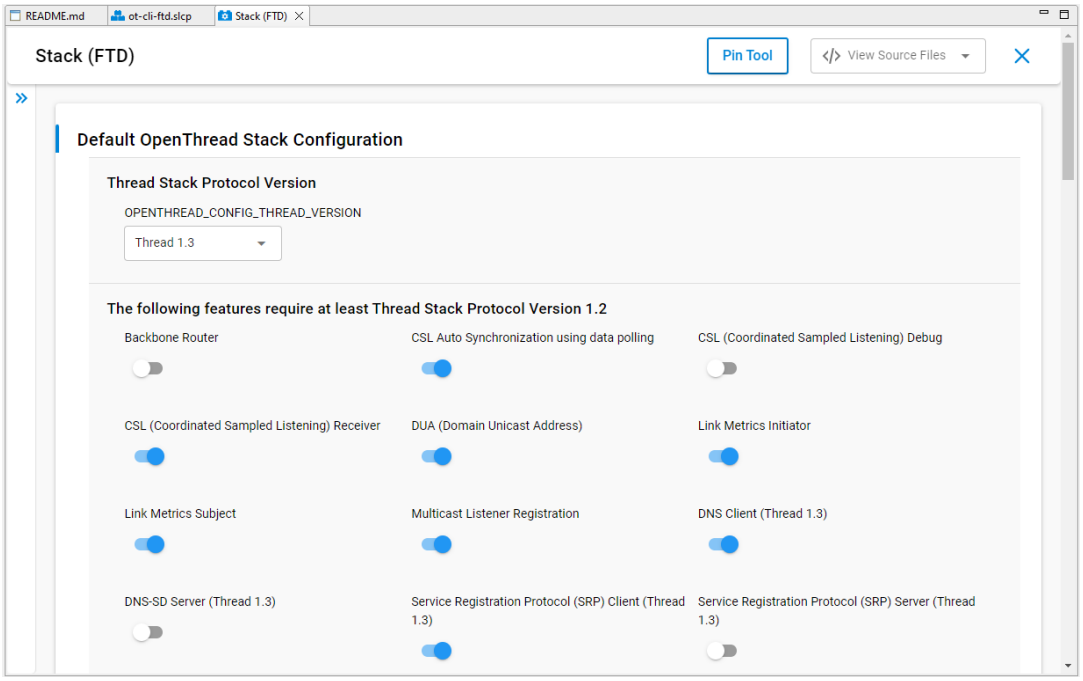
The project is configured by installing and uninstalling components, and configuring installed components. Installed components are checked. Click **Installed Components** to see a filtered list of components installed by the example application.

Configurable components have a gear symbol. Select a component to see information about it.

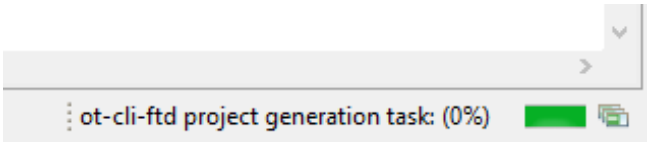


If the component is configurable, click **CONFIGURE** to open the Component Editor.

For example, in the Stack (FTD) component you can enable or disable various stack functions. These are equivalent to build options that can be specified to the makefile when building the sample applications in GitHub.



Any changes you make are autosaved, and project files are autogenerated. Progress is shown in the lower right corner of the Simplicity IDE perspective.

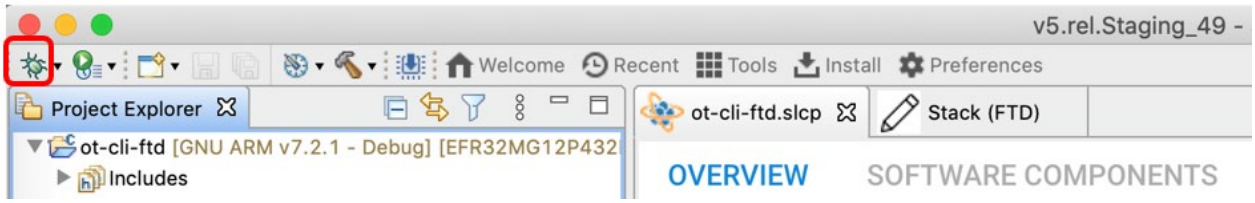


3.3 Building the Project

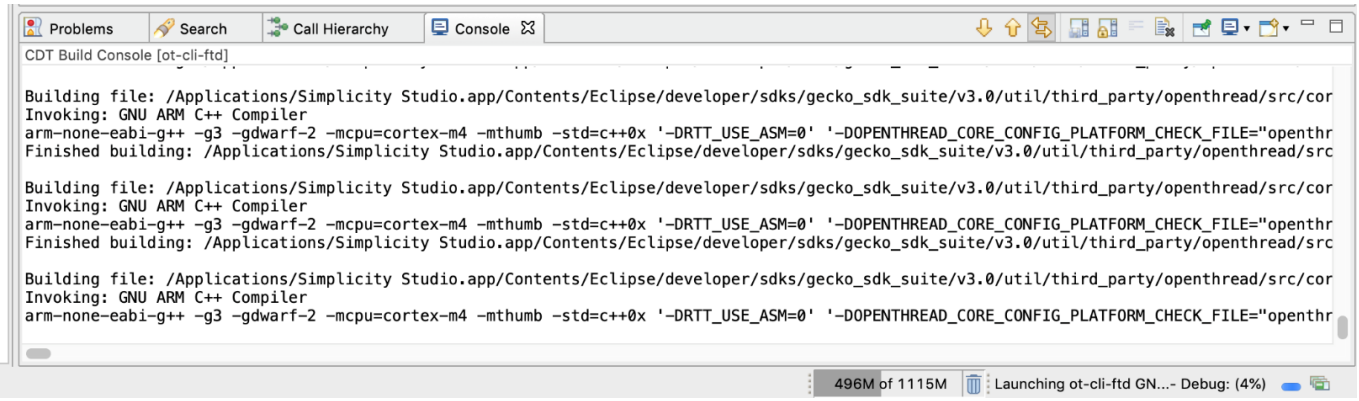
You can either compile and flash the application automatically, or manually compile it and then flash it.

3.3.1 Automatically Compile and Flash

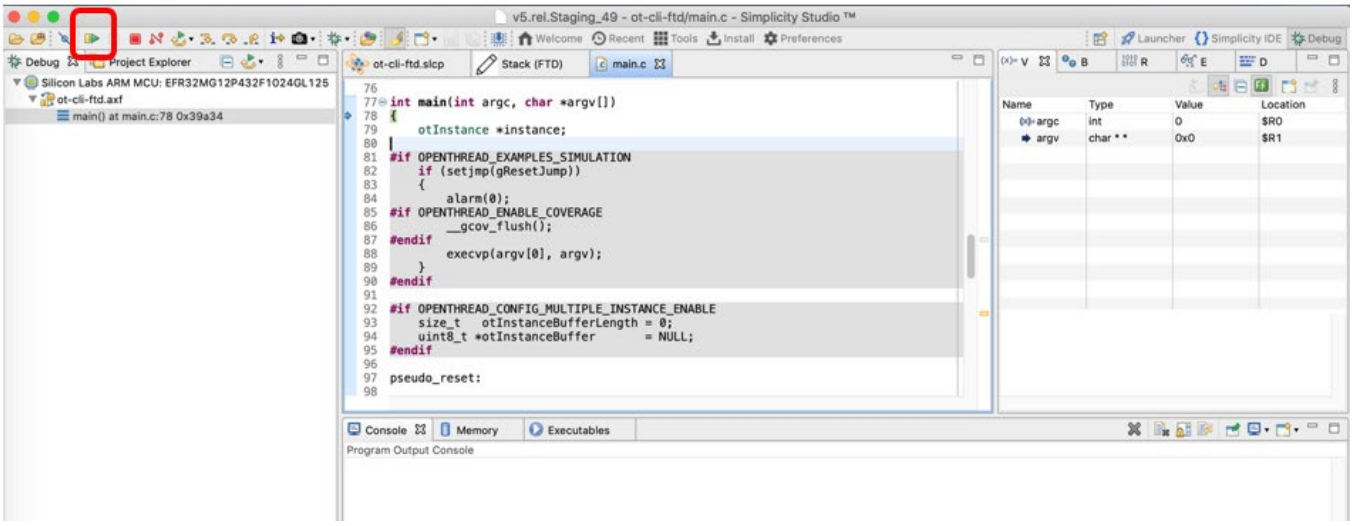
1. You can automatically compile and flash the application to your connected development hardware in the Simplicity IDE. After you click OK on the Generation Confirmation dialog, the Simplicity IDE returns. Click the **Debug** control.



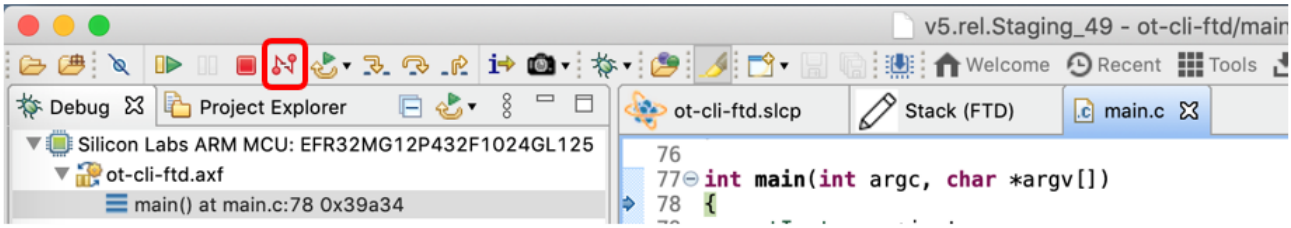
2. Progress is displayed in the console and a progress bar in the lower right.



3. When building and flashing are complete a Debug perspective is displayed. Click the **Resume** control to start the application running on the device.

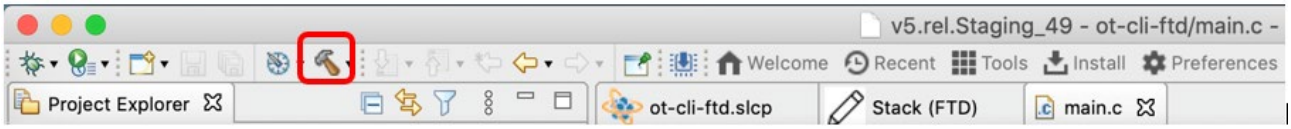


Next to the Resume control are **Suspend**, **Disconnect**, **Reconnect**, and **stepping** controls. Click **Disconnect** when you are ready to exit Debug mode.



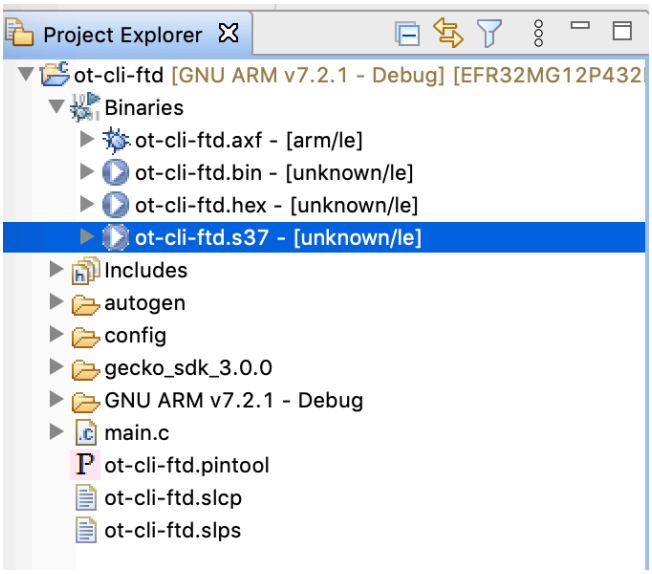
3.3.2 Manually Compile and Flash

1. After you generate your project files, instead of clicking Debug in the Simplicity IDE, click the **Build** control (hammer icon) in the top tool bar.

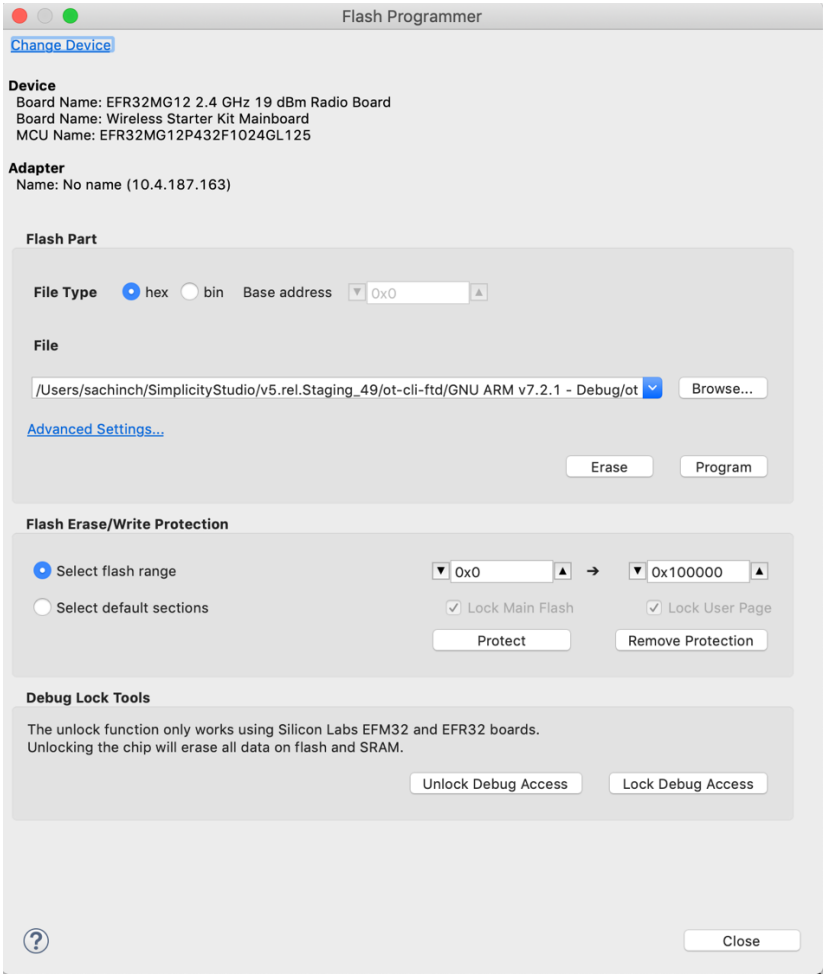


2. You can load the binary image through Project Explorer view.

   Locate the <project>.bin, .hex, or .s37 file in the compiler subdirectory.



Right-click the file and select **Flash Programmer**. Alternately, you may see a **Flash to Device...** option, in which case, you will need to select a device to program, which should take you to the Flash Programmer dialog. The Flash Programmer opens with the file path populated. Click **PROGRAM**.



3.4 Flashing a Bootloader

All Silicon Labs examples require that a bootloader be installed. By default, a new device is factory-programmed with a dummy bootloader. If you have a new device, haven't cleared the bootloader region for your part or have a supported bootloader image already flashed on your device, skip this step and continue with the next section.

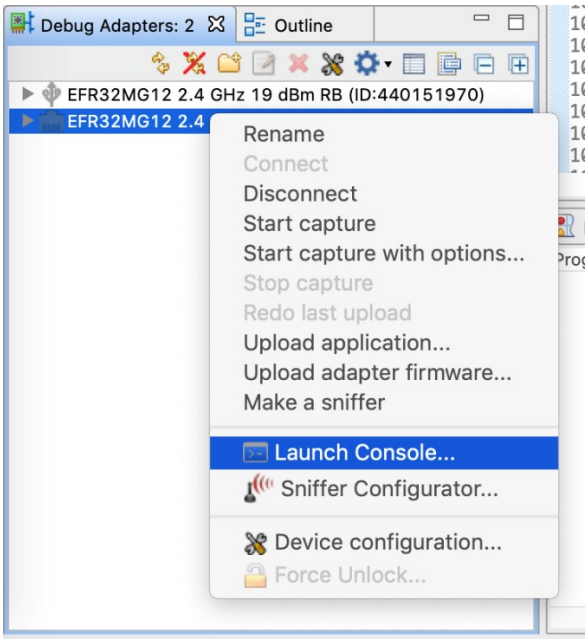
With Silicon Labs OpenThread, the bootloader serves to start the application code within the image you created and flashed in the previous procedure. Once you have installed a bootloader image, it remains installed until you erase the bootloader region for your device.

If you need to flash a bootloader, select a bootloader example and build and flash it as you would any application. See *UG266: Silicon Labs Gecko Bootloader User's Guide for GSDK 3.2 and Lower* or *UG489: Silicon Labs Gecko Bootloader User's Guide for GSDK 4.0 and Higher* for more information.

3.5 Creating a Network

Depending on the example application, you may be able to interact with it through your development environment's Console interface using a CLI (command line interpreter). The console interface allows you to form a network and send data.

To launch the Console interface, in the Simplicity IDE perspective right-click on your J-Link device in the Devices View / Debug Adapters Window. Choose **Launch Console**. Alternatively, from the Tools icon in the Simplicity IDE toolbar, menu in the Launcher perspective select Device Console. To get a prompt on the Console, choose Serial 1 tab and press Enter.



To create a two-node network using the ot-cli-ftd example, flash the **ot-cli-ftd.s37** application image on two separate nodes. Launch the console for the two nodes and execute the commands given below:

Form a network

On Device A, enter:

```
> dataset init new
Done
> dataset commit active
Done
> ifconfig up
Done
> thread start
Done
```

The previous commands should initialize the node with new network parameters and form a network. Check the status of the node after a few seconds by executing the following command. The node should now be a leader.

```
> state
leader
Done
```

Check the network parameters by running the following command:

```
> dataset
dataset
Active Timestamp: 1
Channel: 18
Channel Mask: 07fff800
Ext PAN ID: 4e08d5129185b233
Mesh Local Prefix: fda6:83fa:7cc1:e809/64
Master Key: d5bf78c4bbd96605c6e55a32fa2c03ab
Network Name: OpenThread-21ec
PAN ID: 0x21ec
PSKc: 3be0de54e5d0f2cd5622de0716680588
Security Policy: 0, onrcb
Done
```

Attach a second node (device B) to the network

Note: Depending on the network parameters generated by node A (as seen above), update the channel number and the master key in the following commands.

On Device B enter:

```
 > dataset channel 18
Done
> dataset networkkey d5bf78c4bbd96605c6e55a32fa2c03ab
Done
> dataset commit active
Done
> ifconfig up
Done
> thread start
Done
```

The above commands should cause the node to attach to the network started by node A. Check the status of the node after a few seconds by executing the following command. The node should now be a child (which will eventually update to a router).

```
> state
child
Done
```

Test network connectivity

For this step, compute Node B's IP addresses and ping it from Node A as follows.

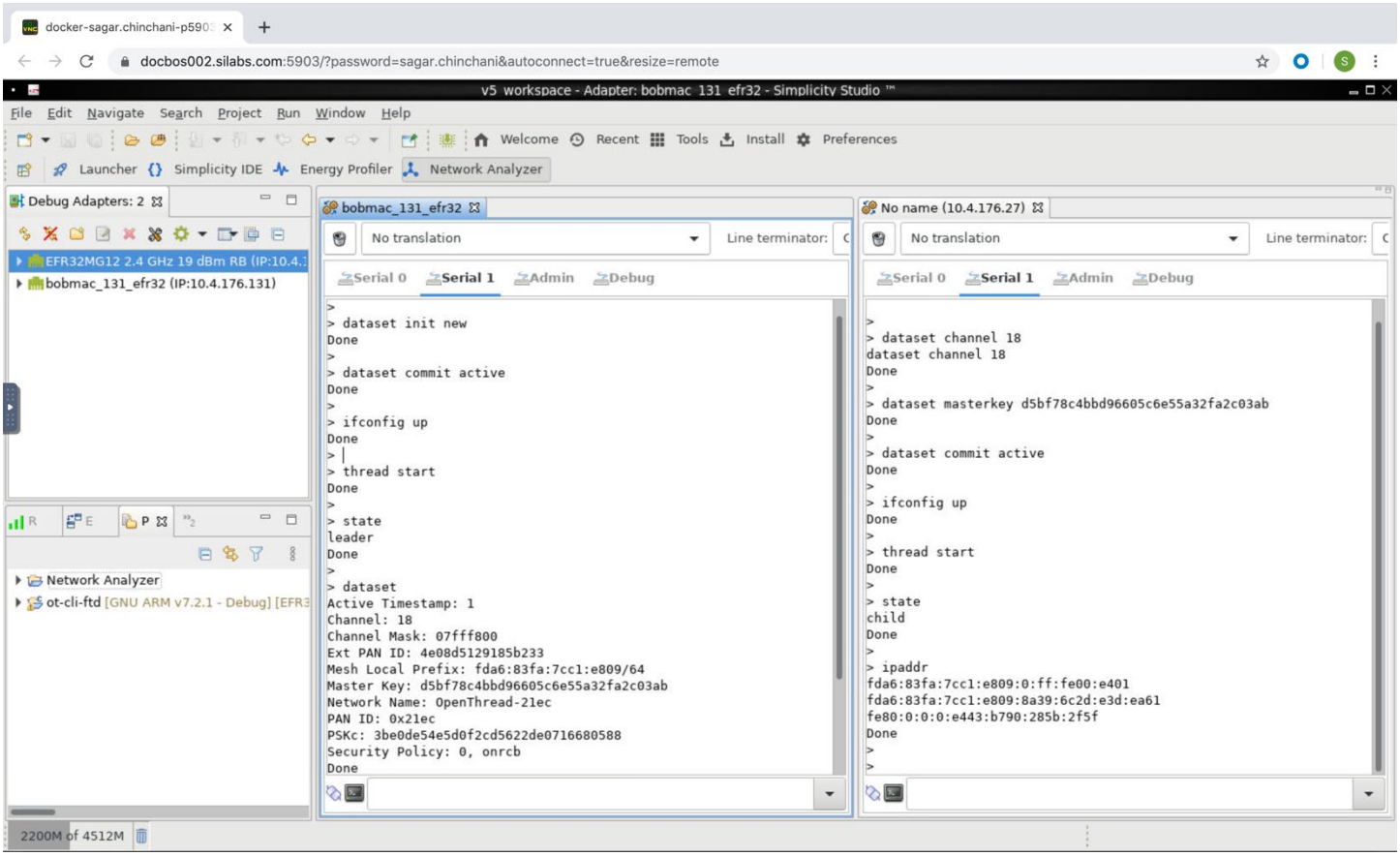
On device B enter:

```
> ipaddr
ipaddr
fda6:83fa:7cc1:e809:0:ff:fe00:e000
fda6:83fa:7cc1:e809:8a39:6c2d:e3d:ea61
fe80:0:0:0:e443:b790:285b:2f5f
Done
```

On node A enter:

```
> ping fe80:0:0:0:e443:b790:285b:2f5f
ping fe80:0:0:0:e443:b790:285b:2f5f
Done
>
> 16 bytes from fe80:0:0:0:e443:b790:285b:2f5f: icmp_seq=2 hlim=64 time=21ms
```

The following figure shows the console at the end of the procedure.



4 Next Steps

Some next steps:

- Run additional cli commands supported by the ot-cli-ftd application (as covered in Chapter 3) to explore OpenThread functionality. A complete list of OpenThread CLI commands can be found here: https://github.com/openthread/openthread/blob/master/src/cli/README.md
- Compile and flash a different example application and explore the functionality it provides. For example, sleepy end device (SED) behavior is supported by the sleepy-demo-mtd application. Some interesting details about this application have been covered in section 5.3 Multi-Node Energy Profiler.
- Explore configuring one of the example applications to meet your needs. Much of the software configuration can be done through components. Select a component to see more information about it and, check if it is configurable.
- Explore the documentation.

Document number	Title
Getting Started	
QSG170	Silicon Labs OpenThread Quick Start Guide
	OpenThread Release Notes
	Gecko Platform Release Notes
Fundamentals	
UG103.11	Thread Fundamentals
UG103.01	Wireless Networking Application Development Fundamentals
OpenThread References	
AN1256	Using the Silicon Labs RCP with the OpenThread Border Router
	API reference
UG162	Simplicity Commander Reference Guide
AN1372	Configuring OpenThread Applications for Thread 1.3
AN1264	Using OpenThread with FreeRTOS
AN1329	Using Silicon Labs Secure Vault Features with OpenThread
AN1350	Single-Band Proprietary Sub-GHz Support with OpenThread
AN1351	Using the Co-Processor Communication Daemon (CPCd)
Bootloading	
UG103.06	Bootloader Fundamentals
UG489/UG266	Silicon Labs Gecko Bootloader User's Guide for GSDK 4.0 and Higher/3.2 and Lower
AN1218	Series 2 Secure Boot with RTSL
Memory Management	
UG103.07	Non-Volatile Data Storage Fundamentals
AN1135	Using Third Generation Non-Volatile Memory (NVM3) Data Storage
Dynamic Multiprotocol	
UG103.16	Multiprotocol Fundamentals
AN1265	Dynamic Multiprotocol Development with Bluetooth® and OpenThread in GSDK v3.x.
UG305	Dynamic Multiprotocol User's Guide
AN1333	Running Zigbee, OpenThread, and Bluetooth Concurrently on a Linux Host with a Multiprotocol RCP
Testing	
AN718	Manufacturing Test Overview
AN700.1	Manufacturing Test Guidelines for the EFR32

Document number	Title
Performance	
AN1141	Thread Mesh Network Performance
AN1142	Mesh Network Performance Comparison
Coexistence	
UG103.17	Wi-Fi Coexistence Fundamentals
AN1017	Zigbee and Thread Coexistence with Wi-Fi
AN1294	Configuring Antenna Diversity for OpenThread
Security	
AN1190	Series 2 Secure Debug
AN1222	Production Programming of Series 2 Devices
AN1247	Anti-Tamper Protection Configuration and Use
AN1268	Authenticating Silicon Labs Devices using Device Certificates
AN1271	Secure Key Storage
AN1303	Programming Series 2 Devices Using the Debug Challenge Interface (DCI) and Serial Wire Debug (SWD)
AN1311	Integrating Crypto Functionality Using PSA Crypto Compared to Mbed TLS

5 Development Tools

5.1 Simplicity Commander

Simplicity Commander is a simple flashing tool, which can be used to flash firmware images, erase flash, lock and unlock debug access, and write-protect flash pages via the J-Link interface. Both GUI and CLI (Command Line Interface) are available. See *UG162: Simplicity Commander Reference Guide* for more information.

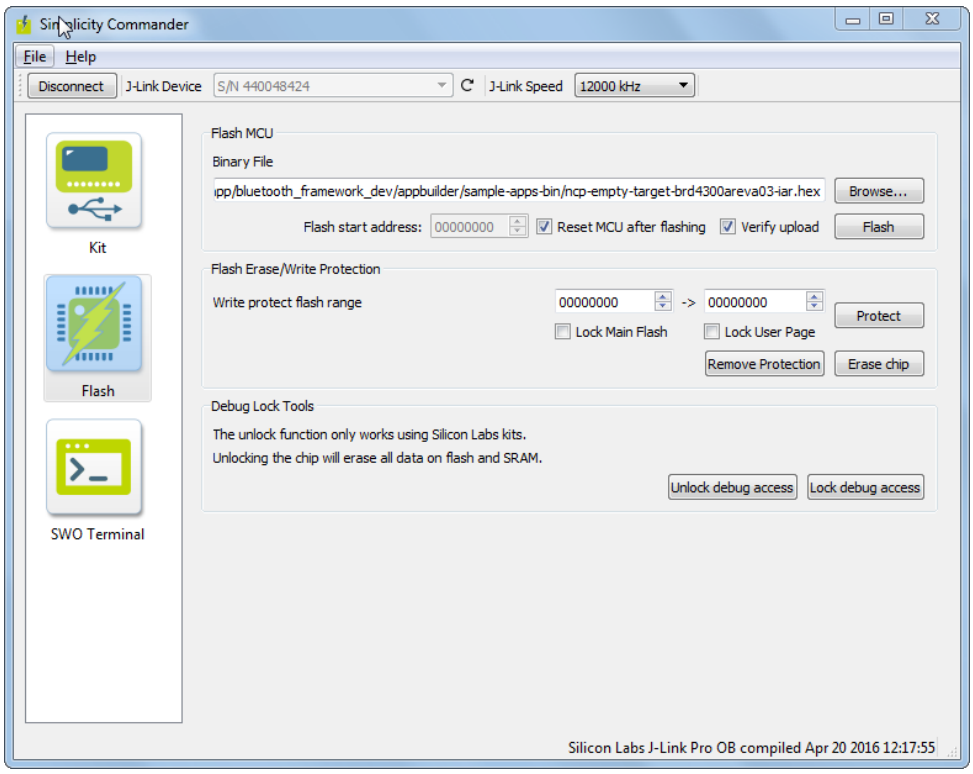
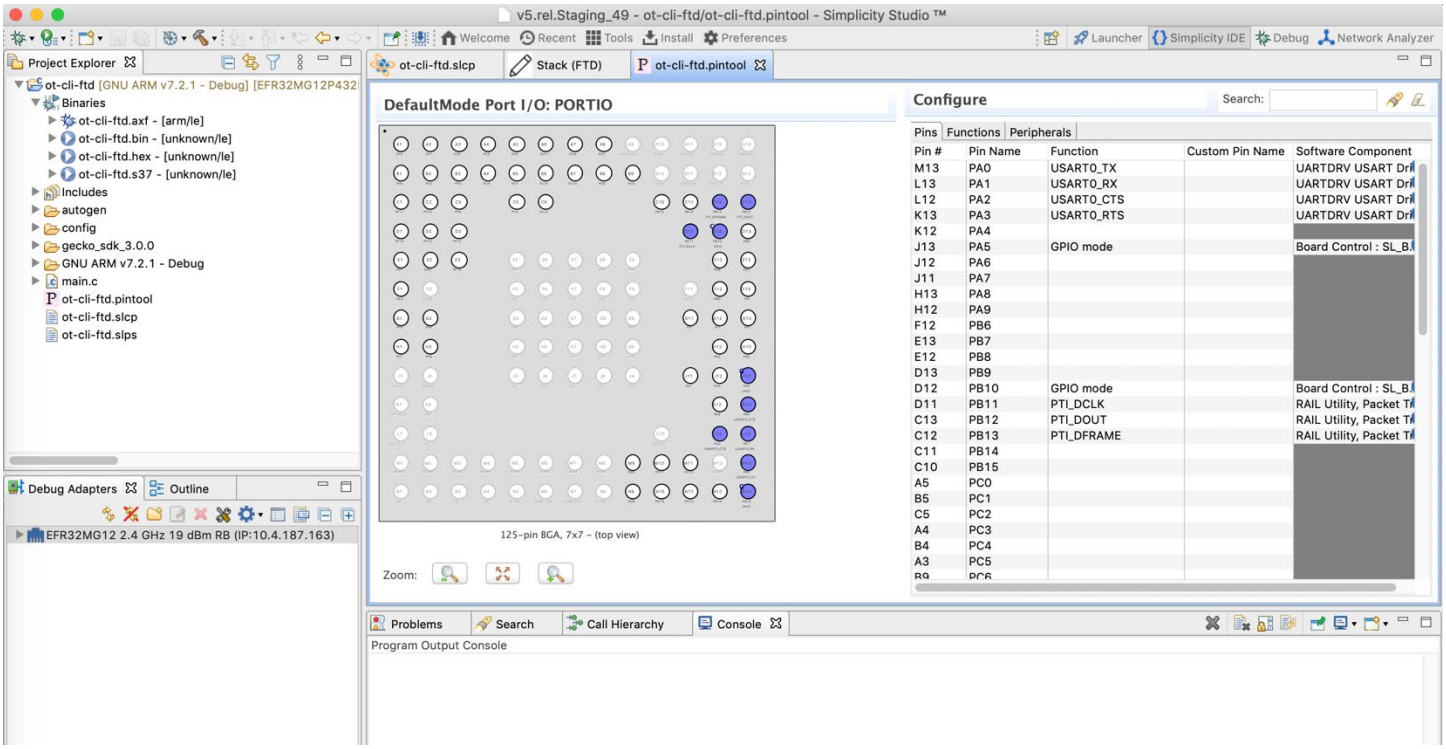


Figure 5.1. Simplicity Commander

5.2 Pin Tool

Simplicity Studio offers a Pin tool that allows you to easily configure new peripherals or change the properties of existing ones. In the CONFIGURATION TOOLS tab, click Open on the Pin Tool card or double-click the file <project>.pintool in the Project Explorer view.

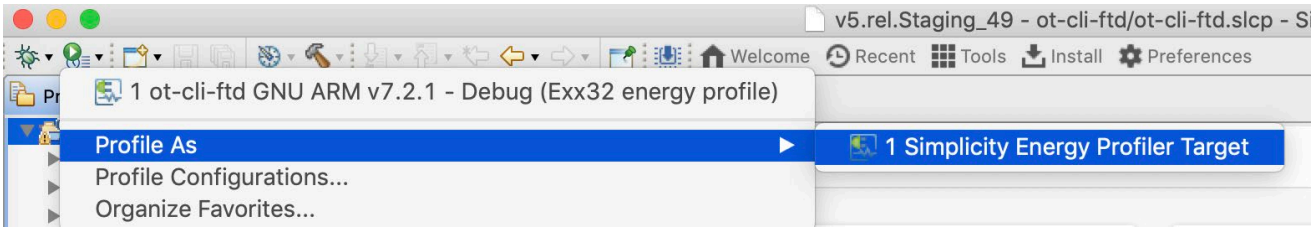


Double-click a Software Component to open the Component Editor and configure that function. Pin Tool does not autosave.

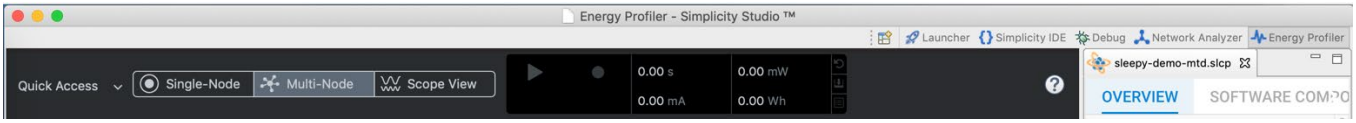
5.3 Multi-Node Energy Profiler

Multi-Node Energy profiler is an add-on tool, with which you can easily measure the energy consumption of your device in runtime. You can easily find peak and average consumption, and check for sleep mode current.

To profile the current project, drop down the Profile as menu in the Simplicity IDE perspective and select Profile as / Simplicity Energy Profiler target. This automatically builds your project, uploads it to the device, and starts Energy Profiler.



For details on how to launch the energy profiler or switch between Simplicity IDE and Energy Profiler perspectives, refer to *UG343: Multi-Node Energy Profiler User's Guide*.



For a node running the ot-cli-ftd application, the Energy Profiler capture should be as shown in the following figure:

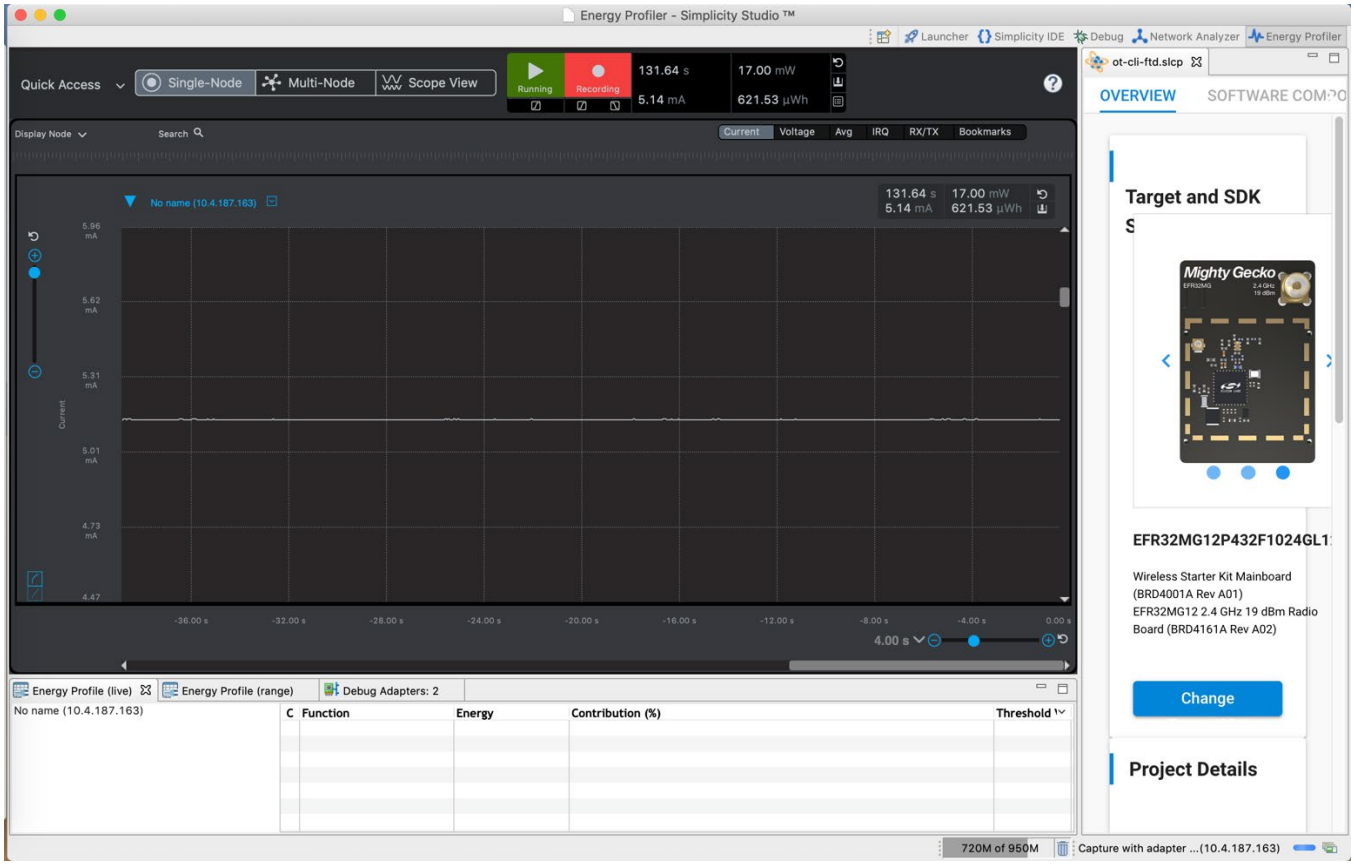


Figure 5.2. Current consumption capture on a node operating in EM0 (active) mode

Please note that the energy consumption results you observe may differ as the peripherals and their energy mode are highly dependent on the specific device.

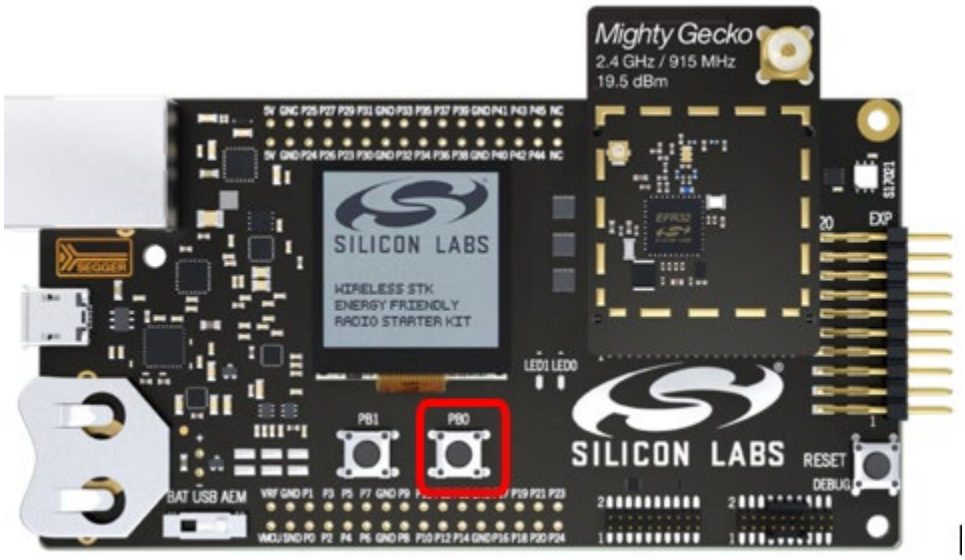
5.3.1 Energy Profiler with Sleepy-demo applications

While the energy profiler can be used on any running project, a good example is the **OpenThread – SoC Sleepy Demo (MTD)** application available with the Silicon Labs OpenThread SDK. It demonstrates Sleepy End Device (SED) behavior using the EFR32's low power EM2 mode.

For this exercise, you will create a two-node network using the **OpenThread – SoC Sleepy Demo (FTD)** and **OpenThread – SoC Sleepy Demo (MTD)** application and monitor the current consumption on the sleepy node.

1. To build the **OpenThread – SoC Sleepy Demo (FTD)** and **OpenThread – SoC Sleepy Demo (MTD)** example applications, either flash the precompiled demo applications as described in section 2.1 Demos or build your own application as described in section 3 Getting Started with Development.
2. Since the **OpenThread – SoC Sleepy Demo (FTD)** and **OpenThread – SoC Sleepy Demo (MTD)** applications are already configured with default network parameters, flashing the applications should automatically form a 2-node network (without specifying any additional commands). Accordingly,
   - On Node A, flash the **OpenThread – SoC Sleepy Demo (FTD)** application. This should cause Node A to form a network.
   - On Node B, flash the **OpenThread – SoC Sleepy Demo (MTD)** application. This should cause node B to attach as a Minimal Thread Device (MTD) to the network formed by Node A.]
   - Try testing network connectivity by sending a ping from Node A(FTD) to Node B(MTD) as described in section 3.5 Creating a Network.

3. Launch the Energy Profiler, start a capture on Node B(MTD), and press button 0 Node B's mainboard.



Pressing button 0 on the Node B(MTD) mainboard will cause the node to toggle between operating as a Minimal End Device (MED) and a Sleepy End Device (SED) with the RX off when idle. As the node uses EFR32's low power EM2 (deep sleep) mode, current consumption will significantly drop (from mA to µA range) as seen in the capture below.

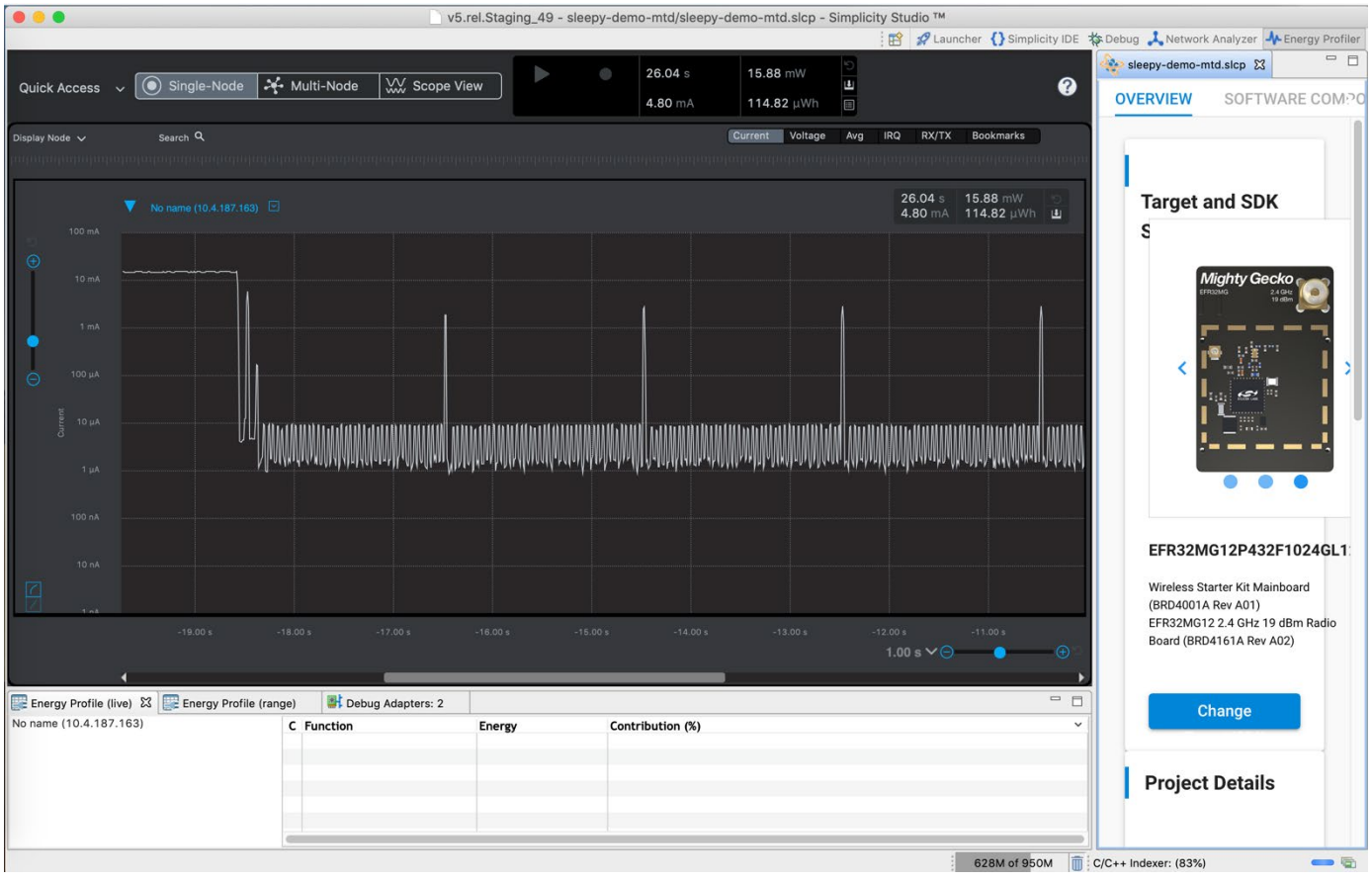


Figure 5.3. Current consumption capture on a node operating in EM2 (deep sleep) mode

The regular spikes seen every 2 seconds (i.e. the sleep period) represents the tiny interval during which the node briefly wakes up to send a data poll message to its parent, after which it goes back to sleep.

Profiling can be paused by clicking Play. For further analysis, click one of the peaks, and zoom in with time axis (x-axis) and the current axis (y-axis). Note that the maximum consumption may now be greater than it appeared on the diagram before you zoomed in. This is because in zoomed-out mode, the displayed values are averaged. If you need exact values, always zoom in. To measure average consumption, simply click and drag your mouse over a time interval. A new window appears in the upper right corner showing consumption information for the given interval.

Figure 5.4 Energy profiler capture on a node operating in EM2 (deep sleep) mode for a given interval

Multi-node Energy Profiler is also able to simultaneously measure the consumption of multiple devices. To start measuring a new device click the Quick Access menu (upper left corner), select **Start Energy Capture** and choose the **Multi-Node View**. To stop measuring, click the Quick Access menu, and select **End/Save session**.

To learn more about how to use this tool, see *UG343: Multi-Node Energy Profiler User's Guide*.

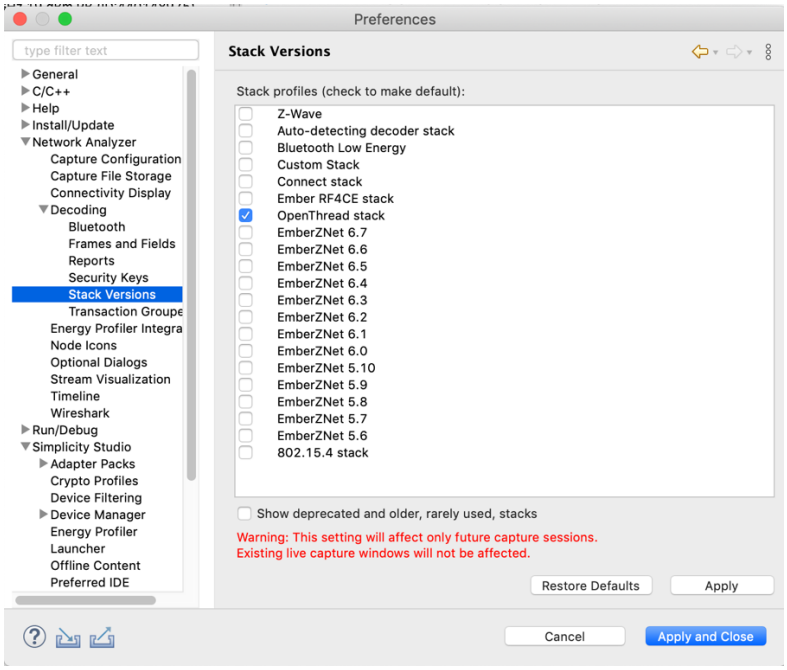
5.4 Network Analyzer

Silicon Labs Network Analyzer is a packet capture and debugging tool that can be used to debug connectivity between wireless nodes running OpenThread stack on EFR32 platform. It significantly accelerates the network and application development process with graphical views of network traffic, activity, and duration. See the online *Simplicity Studio 5 User's Guide*, available both through https://docs.silabs.com/ and the SSv5 help menu, for a guide to Network Analyzer.

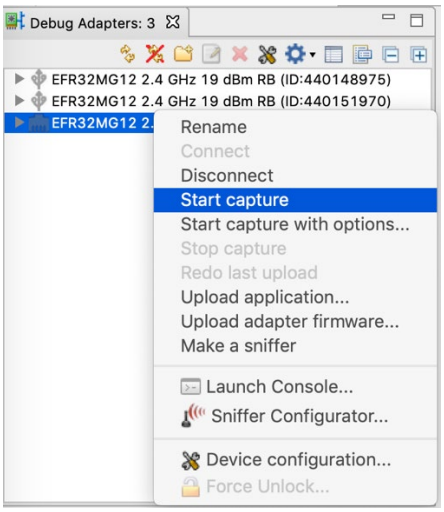
The Packet Trace application captures the packets directly from the Packet Trace Interface (PTI) available on the Wireless Gecko SoCs and modules. It therefore provides a more accurate capture of the packets compared to air-based capture.

To capture OpenThread packets using the Silicon Labs Network Analyzer:

1. In the Preferences window (**Simplicity Studio > Preferences)** , under **Network Analyzer > Decoding > Stack Versions**, select the OpenThread stack option.



2. In the same Preferences window (**Simplicity Studio > Preferences)**, under **Network Analyzer > Decoding > Security Keys**, verify the correct Network Master Key used by the Thread network has been added to the list of known keys.
3. In the Debug Adapters view, right-click the device that is running on a Thread network and start capture.



You should now be able to see the Thread traffic as shown below. Clicking the packets to see more details about its contents in the Event Detail view (on the right)

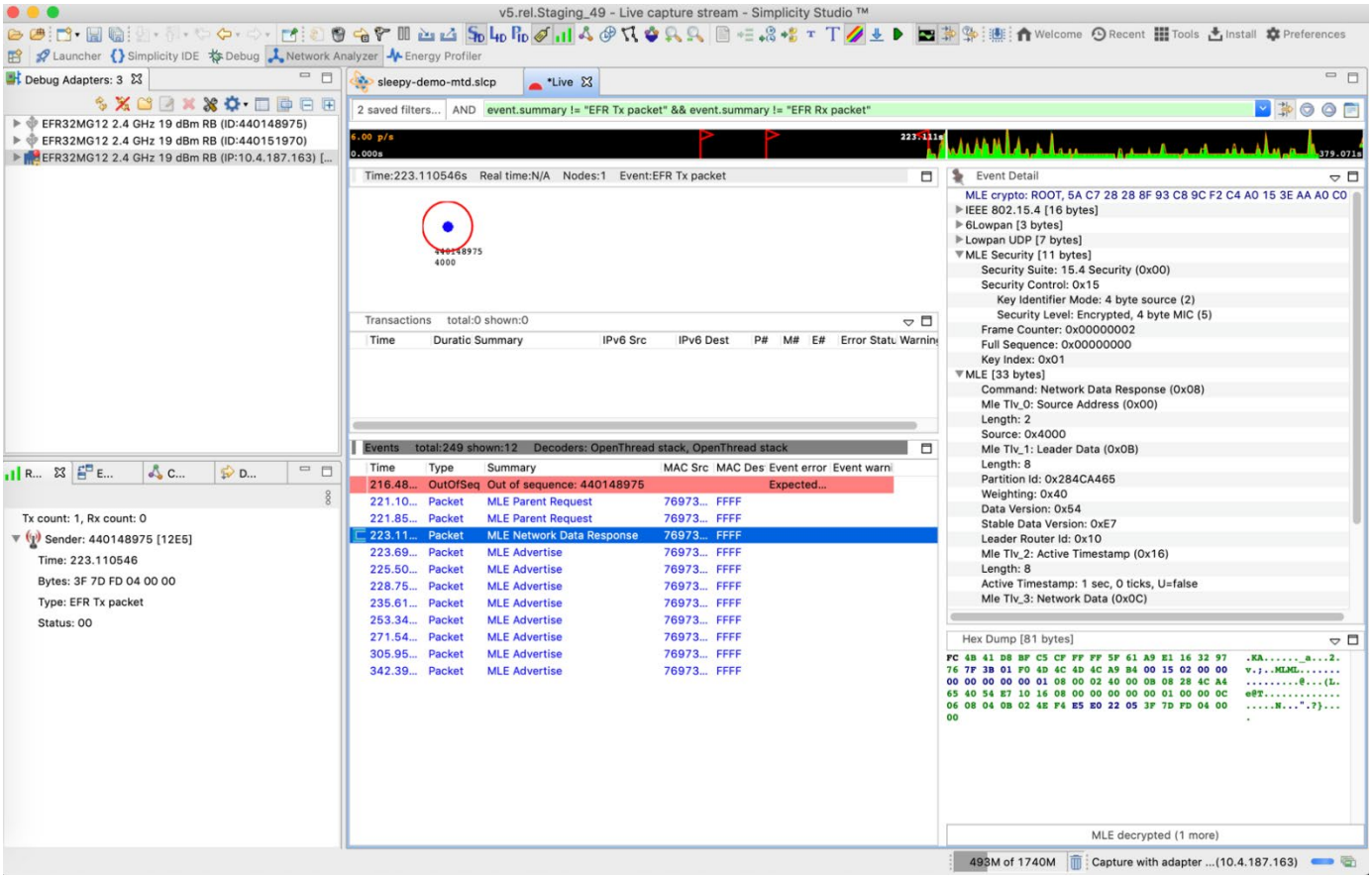


Figure 5.5. Thread Traffic Capture with Packet Trace

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 & Community
www.silabs.com/community

SILICON LABS

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

www.silabs.com