QSG168: Proprietary Flex SDK v3.x Quick-Start Guide

This version of QSG168 has been deprecated with the release of Simplicity SDK Suite 2025.6.1 For the latest version, see docs.silabs.com.

****

This quick start guide provides basic information on configuring, building, and installing applications for the EFR32 using the Silicon Labs Flex Software Development Kit (SDK) v3.x with Simplicity Studio® 5. The Flex SDK provides two paths to application development. The first begins with Silicon Labs RAIL (Radio Abstraction Interface Layer), which is an intuitive, versatile radio interface layer that is designed to support the implementation of arbitrary wireless protocols.

The second uses the Silicon Labs Connect protocol stack, which provides a fully-featured, easily-customizable wireless networking solution optimized for devices that require low power consumption and are used in a simple network topology.

This guide is designed for developers who are new to the Silicon Labs Flex SDK, the Simplicity Studio development environment, and Silicon Labs development hardware. It provides instructions to get started using both the Connect and RAIL examples provided with the Flex SDK used on the EFR32.

Proprietary is supported on all EFR32FG devices. For others, check the device's data sheet under Ordering Information > Protocol Stack to see if Proprietary is supported. In Proprietary SDK version 2.7.n, Connect is not supported on EFR32xG22.

KEY POINTS

- Product overview
- Setting up the development environment
- Discovering the SDK
- Working with example applications.
- Using the Pin Tool

1 Proprietary Flex SDK Product Overview

The Silicon Labs Proprietary Flex SDK supports developers who wish to take advantage of configurable protocol functionality provided in Silicon Labs Connect and the underlying RAIL library, as well as those who wish to start application development on top of RAIL to develop custom lower-level radio control, and network protocols.

This section covers:

- Background information on RAIL, Connect, and the example applications included with the Flex SDK
- Background information on the Gecko Bootloader
- Prerequisites for application development using the Flex SDK
- Support
- Documentation

1.1 About Connect and RAIL

The Flex SDK provides two paths to application development. The first begins with Silicon Labs RAIL (Radio Abstraction Interface Layer), which is an intuitive, versatile radio interface layer that is designed to support the implementation of arbitrary wireless protocols. The second uses Silicon Labs Connect protocol stack, which provides a fully-featured, easily-customizable wireless networking solution optimized for devices that require low power consumption and are used in a simple network topology. Connect example functionality is provided through easily-configurable components that can be turned on or off as desired.

Whether you begin development with Connect or RAIL is determined by the example application you select as a starting point in Simplicity Studio. Silicon Labs recommends that you start from a Connect example if you want to include the following functions without further development:

- MAC layer functionality including frequency hopping and security
- Network formation and, for star networks, routing support
- Application-level functionality, including diagnostics, I/O, mailbox, and sleepy end device management
- Bootloading, including serial and Broadcast or Unicast OTA (over-the-air)

The following sections provide additional detail on Connect and RAIL, including brief descriptions of the example applications. When you create a project based on an example, the Project Details on the Simplicity Studio IDE Overview tab provides additional detail about the example and interfacing with it.

1.1.1 Silicon Labs Connect

Silicon Labs Connect functionality for the EFR32 is implemented on top of the RAIL library. Silicon Labs Connect supports many combinations of radio modulation, frequency and data rates. The stack includes all MAC layer functions such as scanning and joining, setting up of a point-to-point or star network, device types such as sleepy end nodes, routers or coordinators, radio configuration, frequency hopping and LBT (Listen Before Talk) protocols required for regulatory compliance in each geographical region, and PHY configuration for each of these regions. With all this functionality already implemented in the stack, developers can focus on their application development and not worry about the lower-level radio and network details.

The Flex SDK includes a number of Connect example applications including the following.

Connect SoC - Empty: A minimal Connect project structure, used as a starting point for custom applications.

Connect SoC - Direct mode Device: Demonstrates direct communication between nodes in range. The network parameters are commissioned by the application.

Connect SoC - MAC Mode Device: Demonstrates direct MAC mode communication between nodes in range.

Connect SoC - Sensor and **Connect SoC - Sink**: The sensor and sink applications demonstrate how to set up a star network topology in which communication occurs in both directions between the sink and the sensor(s) nodes.

Connect – SoC ECDH Key Exchange. Illustrates how to share the network key between multiple devices in a secure way (using Elliptic-curve Diffie-Hellman (ECDH) key agreement protocol).

Connect – SoC Switch. In combination with the **Connect Bluetooth DMP – SoC Light** example, demonstrates a simple wireless communication between two or more boards using the Connect SDK.

Connect Bluetooth DMP – SoC Empty. An RTOS-based project that provides a skeleton for Connect, with the only function being a Bluetooth LE Task with a basic CLI interface.

Connect Bluetooth DMP – SoC Light. In combination with the **Connect – SoC Switch** example, demonstrates a simple wireless communication between two or more boards using the Connect SDK.

Connect – NCP. A Co-Processor Communication secondary implementation of the Connect API. It can be used with the CPC daemon and the Connect host library and example applications. All can be found on github. See the readme of the application or *UG435.08 Network Co-Processor Applications with Silicon Labs Connect v3.x* for further info.

1.1.2 Silicon Labs RAIL

Silicon Labs RAIL provides an intuitive, versatile radio interface layer designed to support proprietary or standards-based wireless protocols. RAIL is delivered as a library that you can link to your applications. A description of library functions is provided in the development environment. The RAIL API is documented in an online API reference available through Simplicity Studio and online at https://docs.silabs.com/.

RAIL:

- Implements commonly-used radio functionality, so that it does not have to be written in the application or stack.
- Eliminates the need for developers to become expert in RF register details of complex wireless SoCs.
- Simplifies code migration to new wireless ICs and the development of new stacks by providing a common radio interface layer.
- Allows lab evaluation in addition to application code development.

The RAIL library supports APIs for:

- General Radio Operation
- Channel definition and selection
- Output power configuration
- Transmit
- Clear Channel Assessment before Transmit
- Scheduled Transmit
- Energy Detection
- Receive
- Packet Filtering
- Calibration
- CW (Carrier Wave) Transmission
- Modulated Transmission
- RFSense configuration as wake source

The Flex SDK includes example RAIL application code to demonstrate the capabilities of the device and the RAIL library. These examples are provided as source code to offer a starting point for application development. The following examples, among others, are included in the current release.

RAIL – SoC RAILtest

RAILtest is a general test tool for the RAIL library. RAILtest is developed by the core engineering team working on the RAIL library. As each RAIL library feature is implemented, a RAILtest serial command is added to allow scripted testing and ad hoc experimentation. Many of the RAILtest serial commands can be used for lab evaluation.

RAILtest includes commands to:

- Transmit and receive packets.
- Schedule transmits at a specific time in the RAIL timebase.
- Configure RAIL address filtering to receive only specific packets.
- Enable CCA mechanisms (CSMA/LBT) to validate that a channel is clear before transmit.
- Set a timer callback in the RAIL timebase to see how the RAIL timer API works.
- Change the transmit channel within the current configuration's band.
- Change the transmit power level.

- Enable RF energy sensing of specified duration across the 2.4 GHz and/or Sub-GHz bands, and sleep to wake on this event.
- Output a continuous unmodulated tone for debugging.
- Output a continuous modulated PN9 stream for debugging.
- Enter into direct mode where data can be sent and received using asynchronous GPIOs as input and output.

RAIL – SoC Range Test

The Range Test examples enable over-the-air range testing between two devices customized with user-defined parameters. Range Test is designed to be run on Silicon Labs hardware without the need for commands from a host computer. This capability allows for mobility during range testing activities.

RAIL – SoC Range Test BLE and IEEEE80215.4

The Range Test examples enable over-the-air range testing between two devices customized with user-defined parameters. Five predefined PHYs can be used: BLE: 125kbps, BLE: 500kbps, BLE: 1Mbps, BLE: 2Mbps, IEEE80215.4: 250kbps.

RAIL – SoC Switch: Demonstrates the simplest exchange of transmit and receive operation between a light and a switch.

RAIL – SoC Light: Demonstrates the simplest exchange of transmit and receive operation between a RAIL light and a RAIL switch. The light is capable of periodically and on change reporting its status back to the switch. This is not the dynamic multiprotocol light example application.

RAIL – SoC Mode Switch: Demonstrates an exchange of transmit and receive operations between two (or more) nodes expanded by the Wi-SUN Mode Switching features implemented in RAIL.

RAIL – SoC Simple TRX Multi-PHY: Demonstrates the use of multiple PHYs selectable by channels. By default, channel 0 is configured to 2.4 GHz, 250 kbps, and channel 1 is configured to 915 MHz, 500 kbps (both packets are receivable by a single-PHY application using the correct pre-configured PHY). For details see AN971: EFR32 Radio Configurator Guide.

RAIL – SoC Simple TRX: Demonstrates the simplest transmit and receive functions based on RAIL.

RAIL – SoC Simple TRX Standards: Demonstrates the simplest transmit and receive functions based on RAIL based on IEEE Std. 802.15.4.

RAIL – SoC Simple TRX with Auto-ACK: Demonstrates the simplest exchange transmit and ack operation between two nodes, based on RAIL.

RAIL – NCP Simple TRX with CPC Support: Same as *RAIL – SoC Simple TRX*, but implements a Co-Processor Communication endpoint, so the device can be controlled through CPCd from a Linux computer, using UART interface.

RAIL – NCP Simple TRX with CPC Support (SPI): Same as *RAIL – SoC Simple TRX with CPC Support*, but uses SPI interface to communicate with CPCd.

RAIL – SoC Long Preamble Duty Cycle: Demonstrates how Rx duty cycling can be implemented using a long preamble.

RAIL – SoC Burst Duty Cycle: Demonstrates how Rx duty cycle can be implemented using repeated transmissions

RAIL – SoC Energy Mode: Demonstrates the low power modes (EM0-Active, EM1-Sleep, EM2-Deep Sleep). of the EFR32.

RAIL – SoC Empty: A minimal RAIL project structure, used as a starting point for custom applications.

RAIL Bluetooth DMP – SoC Range Test: Range Test with Bluetooth connectivity. It runs on top of Micrium OS RTOS and multiprotocol RAIL.

RAIL Bluetooth DMP – SoC Range Test BLE and IEEE802.15.4: Range Test BLE and IEEE802.15.4 with Bluetooth connectivity. It runs on top of Micrium OS RTOS and multiprotocol RAIL.

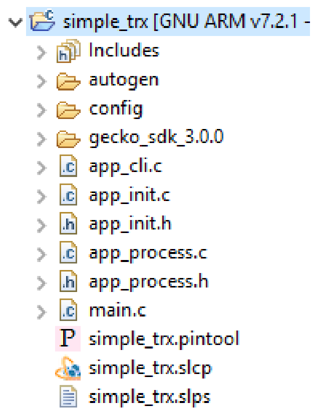
RAIL - SoC Wireless M-bus Meter: Implements a Wireless M-Bus meter application. For details, see *AN1119: Using RAIL for Wireless M-Bus Applications with EFR32*. Uses the multi-PHY configurator and is capable of limited multi-PHY features, like asymmetric bidirectional modes. For details see *AN1253: EFR32 Radio Configurator Guide for Simplicity Studio 5*.

RAIL - SoC Wireless M-bus Collector: Implements a Wireless M-Bus collector application. For details, see *AN1119: Using RAIL for Wireless M-Bus Applications with EFR32*. Uses the multi-PHY configurator. For details see *AN1253: EFR32 Radio Configurator Guide for Simplicity Studio 5*.

1.1.3 Project Structure

Projects always include the following parts:

- **autogen** folder: Only the autogen folder includes generated code. It includes the PHY configuration (*rail_config.c*), init code, the linker script, and other generated code used by components, like the command descriptors for the CLI interface.
- **config** folder: Component configuration headers are located in this folder. These can be edited with the Simplicity IDE Component Editor, but directly editing the header file is also possible. The Component Editor is available through the Project Configurator's **Configure** control, available only for configurable components.
- **gecko_sdk** folder (with version number): Contains source and binary files added by components.
- **files in the root folder**: Only the application specific files should be in the root folder, including source files, the project configurator (.*slcp*) file and the Pin Tool (.*pintool*) file.



All projects include *main.c*, which is not recommended to modify. Instead, add the initialization code to *app_init.c*, and implement the main loop in *app_process.c*. This way, the System components can initialize components, and call the "process" function of the components that require it. Additionally, enabling an RTOS will convert *app_process*'s main loop into an RTOS task.

These files are also available in Connect projects, and can be used for example for init and process tasks that are independent of the stack running or not. However, it is still recommended to use `emberAfInitCallback()` and `emberAfTickCallback()`, so the Connect stack could prioritize urgent stack operations above the application.

1.2 The 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. For more information about bootloaders see *UG103.6: Bootloader Fundamentals*.

The Gecko Bootloader is 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 bootloaders work with specialized firmware update image formats. The Gecko Bootloader update images has the extension .gbl.

Examples provided for the EFR32 parts include Silicon Labs Gecko Bootloader examples. Examples are provided for all compatible Simplicity Studio SDKs. For more information on using the Gecko Bootloader see *UG266: Silicon Labs Gecko Bootloader User's Guide in GSDK 3.2 and Lower* and *UG489: Silicon Labs Gecko Bootloader User's Guide in GSDK 4.0 and Higher.*

Note: When working with the Gecko Bootloader, you must use Simplicity Commander to enable some configuration options such as security features.

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 Simplicity Studio's Documentation tab.

1.4 Prerequisites

Before following the procedures in this guide you must have

- Purchased one of the Wireless Gecko (EFR32) Portfolio Wireless Kits.
- Installed the Flex SDK. The Flex SDK is provided as part of the Gecko SDK (GSDK), the suite of Silicon Labs SDKs. To quickly get started with the GSDK, install Simplicity Studio 5, which will set up your development environment and walk you through GSDK installation. Simplicity Studio 5 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. Installation instructions are provided in the online Simplicity Studio 5 User's Guide. Alternatively, Gecko SDK may be installed manually by downloading or cloning the latest from GitHub. See https://github.com/SiliconLabs/gecko_sdk for more information.

  See the Flex SDK v3.x release notes for version restrictions and compatibility constraints for Connect and RAIL and these components.

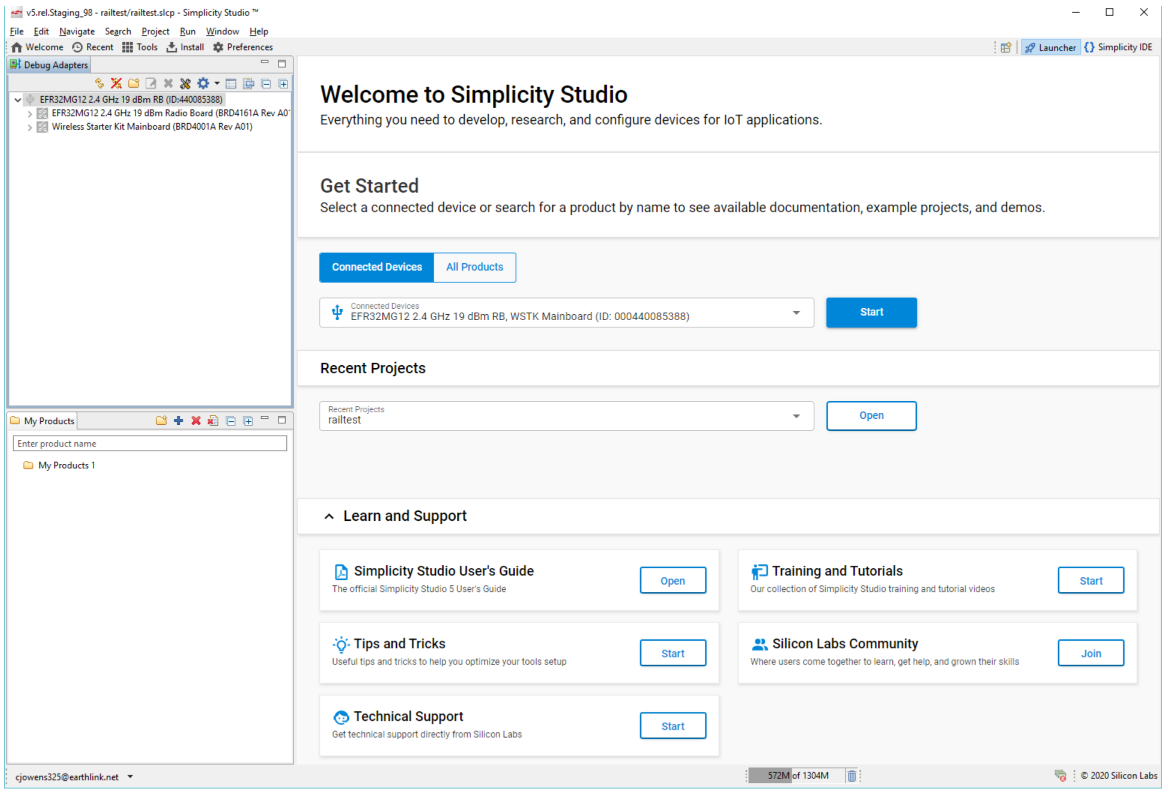
  Simplicity Commander is installed along with Simplicity Studio. An application with limited functionality can be accessed through Simplicity Studio's Tools menu. Most functions are accessible through a CLI invoked by opening a command prompt in the Simplicity Commander directory. See *UG162: Simplicity Commander Reference Guide* for more information.
- (optional) Installed IAR Embedded Workbench for ARM (IAR EWARM). See the Release Notes for the IAR version supported by this version of the Flex SDK. This can 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. Download the supported version from the Silicon Labs Support Portal, as described below. Refer to the "QuickStart Installation Information" section of the IAR installer for additional information about the installation process and how to configure your license.

To get a 30-day evaluation license for IAR:

- Go to the Silicon Labs support portal at https://www.silabs.com/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 **Development Tools**. Click **Go**. In the results is a link to the IAR-EWARM version named in the release notes.
- 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.
- Once IAR-EWARM is installed, the next time Simplicity Studio starts it will automatically detect and configure the IDE to use IAR-EWARM.

1.5 Technical Support

You can access the Silicon Labs support portal at https://www.silabs.com/support through Simplicity Studio 5's Welcome view under Learn and Support. Use the support portal to contact Customer Support for any technical questions you might have during the development process.


v5.rel.Staging_98 - railtest/railtest.slcp - Simplicity Studio ™
File Edit Navigate Search Project Run Window Help
Welcome Recent Tools Install Preferences
Launcher Simplicity IDE
Debug Adapters
EFR32MG12 2.4 GHz 19 dBm RB (ID:440085388)
EFR32MG12 2.4 GHz 19 dBm Radio Board (BRD4161A Rev A0
Wireless Starter Kit Mainboard (BRD4001A Rev A01)
My Products
Enter product name
My Products 1
Connected Devices
All Products
Start
Open
Join
cjowens325@earthlink.net
572M of 1304M
© 2020 Silicon Labs

Welcome to Simplicity Studio

Everything you need to develop, research, and configure devices for IoT applications.

Get Started

Select a connected device or search for a product by name to see available documentation, example projects, and demos.

Connected Devices: EFR32MG12 2.4 GHz 19 dBm RB, WSTK Mainboard (ID: 000440085388)

Recent Projects

Recent Projects: railtest

Learn and Support

Simplicity Studio User's Guide — The official Simplicity Studio 5 User's Guide

Training and Tutorials — Our collection of Simplicity Studio training and tutorial videos

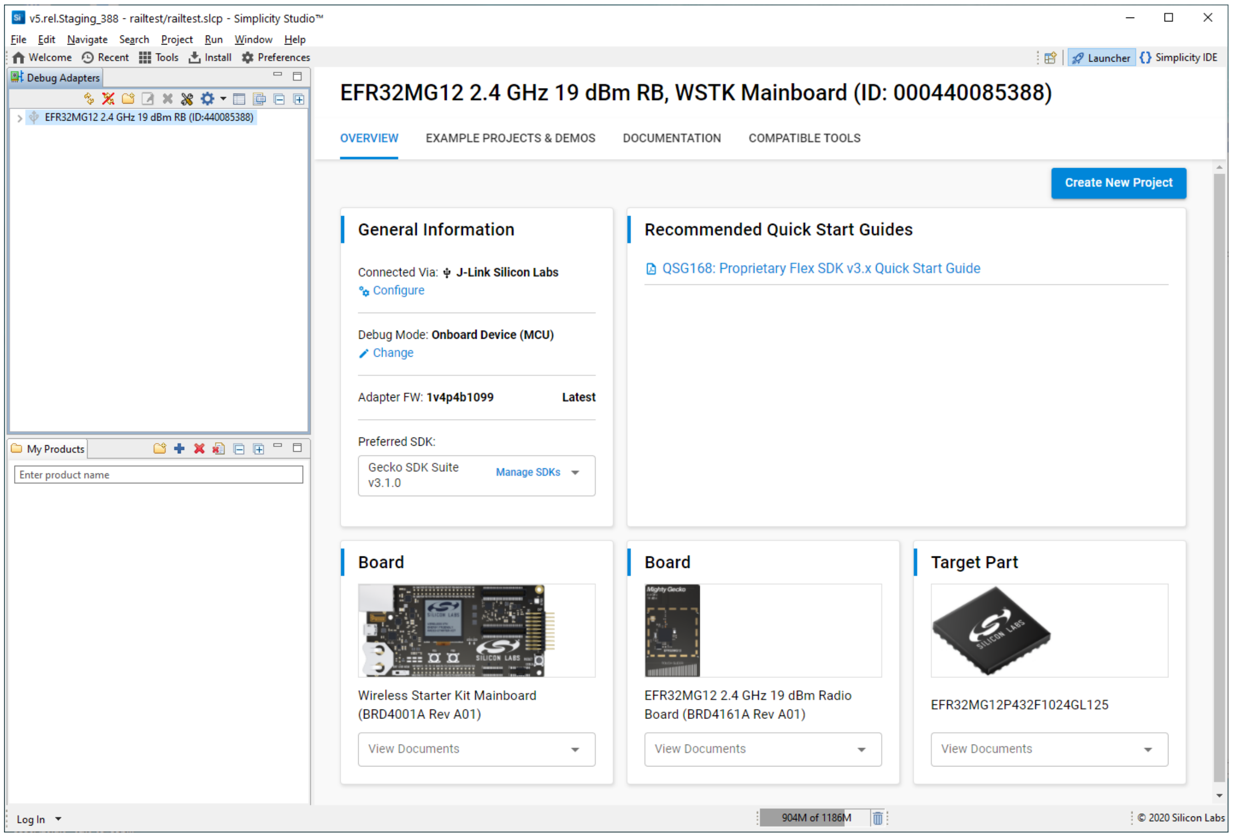
Tips and Tricks — Useful tips and tricks to help you optimize your tools setup

Silicon Labs Community — Where users come together to learn, get help, and grown their skills

Technical Support — Get technical support directly from Silicon Labs

1.6 Documentation

Hardware-specific documentation may be accessed through links on the part OVERVIEW tab in Simplicity Studio 5.


v5.rel.Staging_388 - railtest/railtest.slcp - Simplicity Studio™
File Edit Navigate Search Project Run Window Help
Welcome Recent Tools Install Preferences
Launcher Simplicity IDE
Debug Adapters
EFR32MG12 2.4 GHz 19 dBm RB (ID:440085388)
My Products
Enter product name
Log In
OVERVIEW
EXAMPLE PROJECTS & DEMOS
DOCUMENTATION
COMPATIBLE TOOLS
Create New Project
904M of 1186M
© 2020 Silicon Labs

EFR32MG12 2.4 GHz 19 dBm RB, WSTK Mainboard (ID: 000440085388)

General Information

Connected Via: **J-Link Silicon Labs**
Configure

Debug Mode: **Onboard Device (MCU)**
Change

Adapter FW: **1v4p4b1099** **Latest**

Preferred SDK:
Gecko SDK Suite v3.1.0 Manage SDKs

Recommended Quick Start Guides

QSG168: Proprietary Flex SDK v3.x Quick Start Guide

Board

Wireless Starter Kit Mainboard (BRD4001A Rev A01)
View Documents

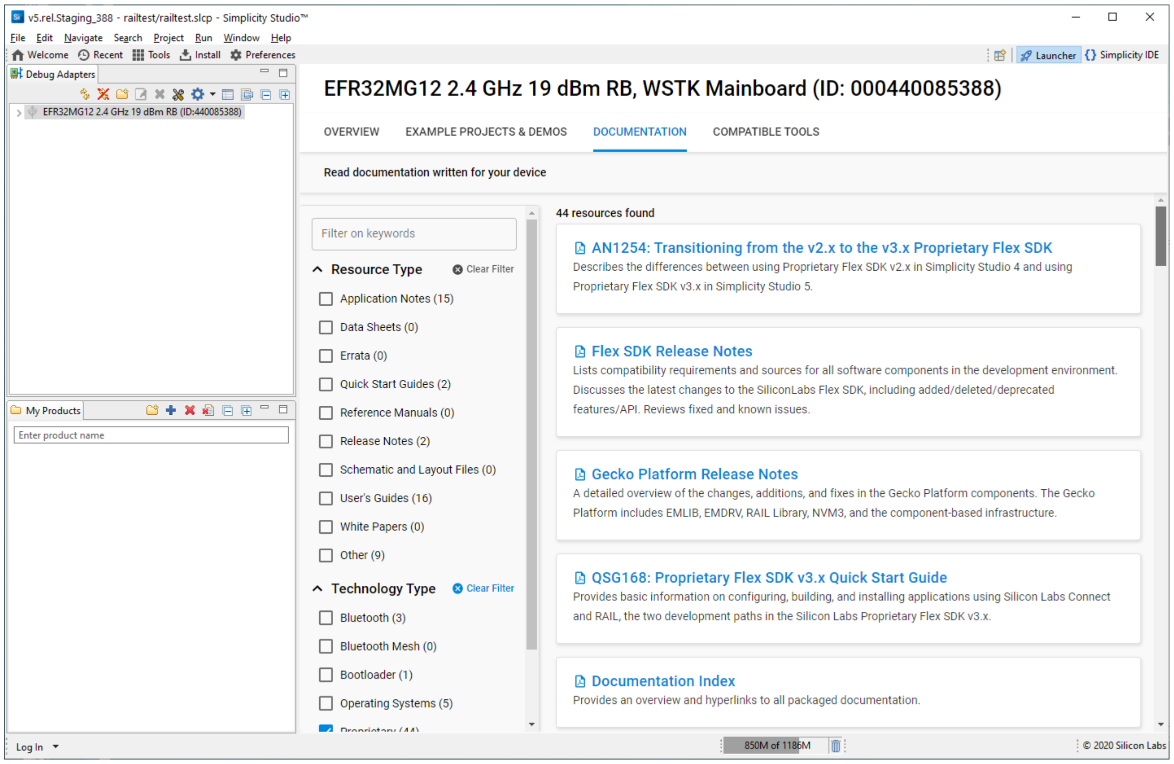
Board

EFR32MG12 2.4 GHz 19 dBm Radio Board (BRD4161A Rev A01)
View Documents

Target Part

EFR32MG12P432F1024GL125
View Documents

SDK documentation and other references are available through the Documentation tab.


v5.rel.Staging_388 - railtest/railtest.slcp - Simplicity Studio™
File Edit Navigate Search Project Run Window Help
Welcome Recent Tools Install Preferences
Launcher Simplicity IDE
Debug Adapters
EFR32MG12 2.4 GHz 19 dBm RB (ID:440085388)
My Products
Enter product name
Log In
OVERVIEW
EXAMPLE PROJECTS & DEMOS
DOCUMENTATION
COMPATIBLE TOOLS
850M of 1186M
© 2020 Silicon Labs

EFR32MG12 2.4 GHz 19 dBm RB, WSTK Mainboard (ID: 000440085388)

Read documentation written for your device

Filter on keywords

Resource Type Clear Filter
- [ ] Application Notes (15)
- [ ] Data Sheets (0)
- [ ] Errata (0)
- [ ] Quick Start Guides (2)
- [ ] Reference Manuals (0)
- [ ] Release Notes (2)
- [ ] Schematic and Layout Files (0)
- [ ] User's Guides (16)
- [ ] White Papers (0)
- [ ] Other (9)

Technology Type Clear Filter
- [ ] Bluetooth (3)
- [ ] Bluetooth Mesh (0)
- [ ] Bootloader (1)
- [ ] Operating Systems (5)
- [x] Proprietary (44)

44 resources found

AN1254: Transitioning from the v2.x to the v3.x Proprietary Flex SDK
Describes the differences between using Proprietary Flex SDK v2.x in Simplicity Studio 4 and using Proprietary Flex SDK v3.x in Simplicity Studio 5.

Flex SDK Release Notes
Lists compatibility requirements and sources for all software components in the development environment. Discusses the latest changes to the SiliconLabs Flex SDK, including added/deleted/deprecated features/API. Reviews fixed and known issues.

Gecko Platform Release Notes
A detailed overview of the changes, additions, and fixes in the Gecko Platform components. The Gecko Platform includes EMLIB, EMDRV, RAIL Library, NVM3, and the component-based infrastructure.

QSG168: Proprietary Flex SDK v3.x Quick Start Guide
Provides basic information on configuring, building, and installing applications using Silicon Labs Connect and RAIL, the two development paths in the Silicon Labs Proprietary Flex SDK v3.x.

Documentation Index
Provides an overview and hyperlinks to all packaged documentation.

2 Getting Started with Simplicity Studio 5 and the Proprietary Flex SDK

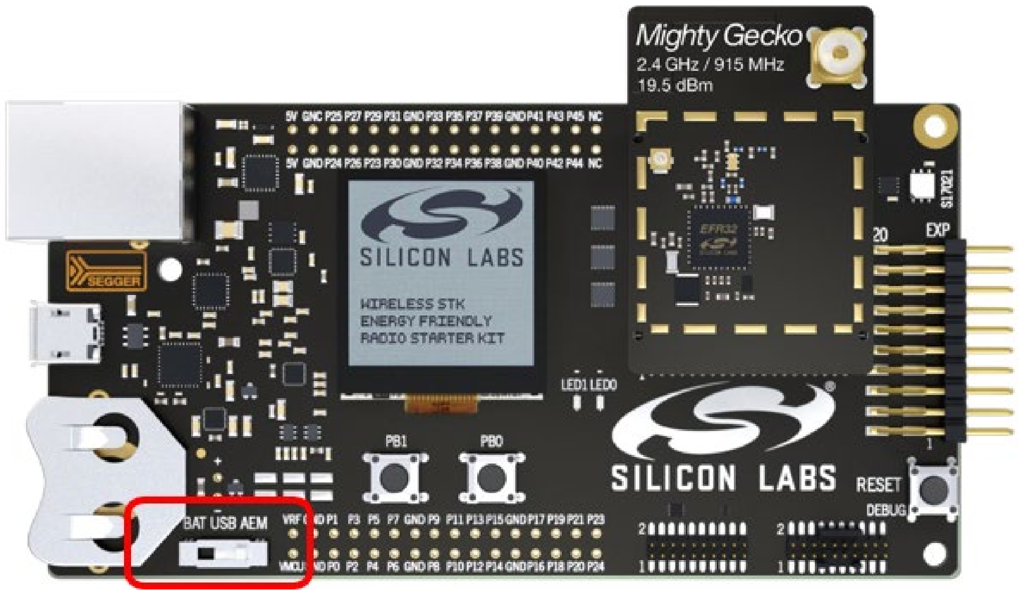
Simplicity Studio 5 (SSv5) is a free Eclipse-based Integrated Development Environment (IDE) and a collection of value-add tools provided by Silicon Labs. Developers can use SSv5 to develop, debug and analyze their Flex and other Silicon Labs SDK-based applications. Its main goal is to reduce development time so that you can focus on your application instead of researching the hardware reference manuals.

The online Simplicity Studio 5 User's Guide provides both a summary overview of SSv5 functions and how to use it with Flex SDK, as well as detailed references to the various SSv5 tools and perspectives.

This guide assumes that you have downloaded SSv5 and the Gecko SDK, which includes the Flex 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.



3 Working with Example Applications

This chapter offers instructions on working with RAIL and Connect SoC examples.

Simplicity Studio offers a variety of ways to begin a project using an example application. The online Simplicity Studio 5 User's Guide 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.

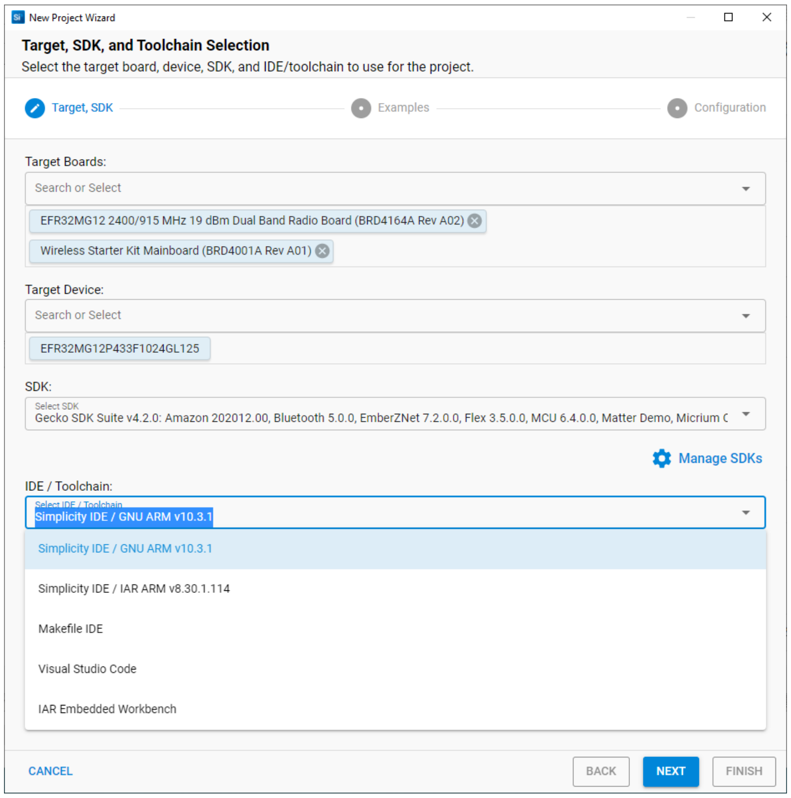
Regardless of how you begin, when working with example applications in Simplicity Studio, you will execute the following steps:

- Select an example application and create a project
- Modify and generate radio configuration.
- Modify code if necessary
- Compile and flash the application to the radio board.

These steps are described in the following sections. Note: Your SDK version may be later than the version shown in the procedure illustrations.

3.1 Creating a Project Based on an Example

1. Open SSv5's File menu and select **New > Silicon Labs Project Wizard**. The Target, SDK, and Toolchain Selection dialog opens. If you want to change the toolchain from the default GCC to IAR, do so here. Click **NEXT**.



New Project Wizard

Target, SDK, and Toolchain Selection

Select the target board, device, SDK, and IDE/toolchain to use for the project.

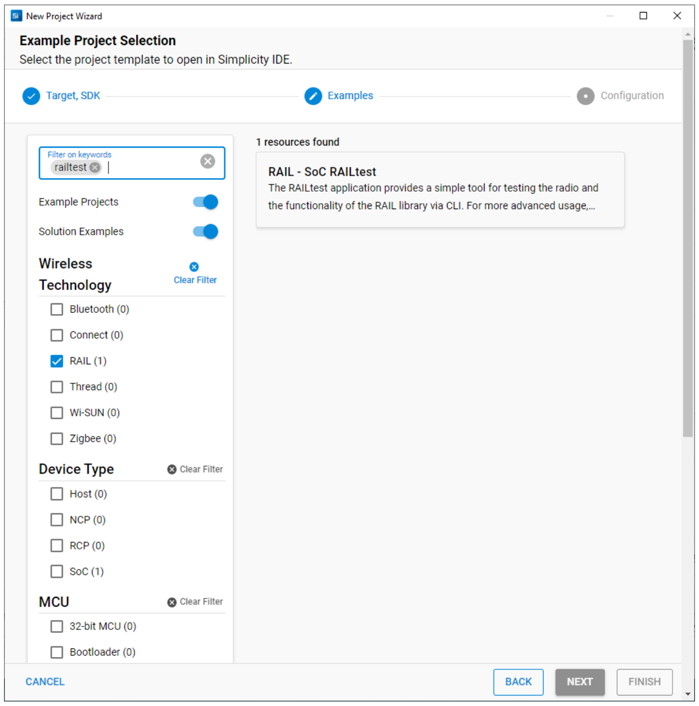
Target Boards: EFR32MG12 2400/915 MHz 19 dBm Dual Band Radio Board (BRD4164A Rev A02); Wireless Starter Kit Mainboard (BRD4001A Rev A01)

Target Device: EFR32MG12P433F1024GL125

SDK: Gecko SDK Suite v4.2.0: Amazon 202012.00, Bluetooth 5.0.0, EmberZNet 7.2.0.0, Flex 3.5.0.0, MCU 6.4.0.0, Matter Demo, Micrium C

IDE / Toolchain: Simplicity IDE / GNU ARM v10.3.1

2. The Example Project Selection dialog opens. A variety of filters are available to help you find a specific example, such as 'railt' and Technology Type: Proprietary in the following figure. Select the example, in this case **RAIL – SoC RAILtest**. Select it and click **NEXT**.


New Project Wizard

Example Project Selection

Select the project template to open in Simplicity IDE.

Filter on keywords: railtest

Example Projects [✓]

Solution Examples [✓]

Wireless Technology — Clear Filter
- [ ] Bluetooth (0)
- [ ] Connect (0)
- [x] RAIL (1)
- [ ] Thread (0)
- [ ] Wi-SUN (0)
- [ ] Zigbee (0)

Device Type — Clear Filter
- [ ] Host (0)
- [ ] NCP (0)
- [ ] RCP (0)
- [ ] SoC (1)

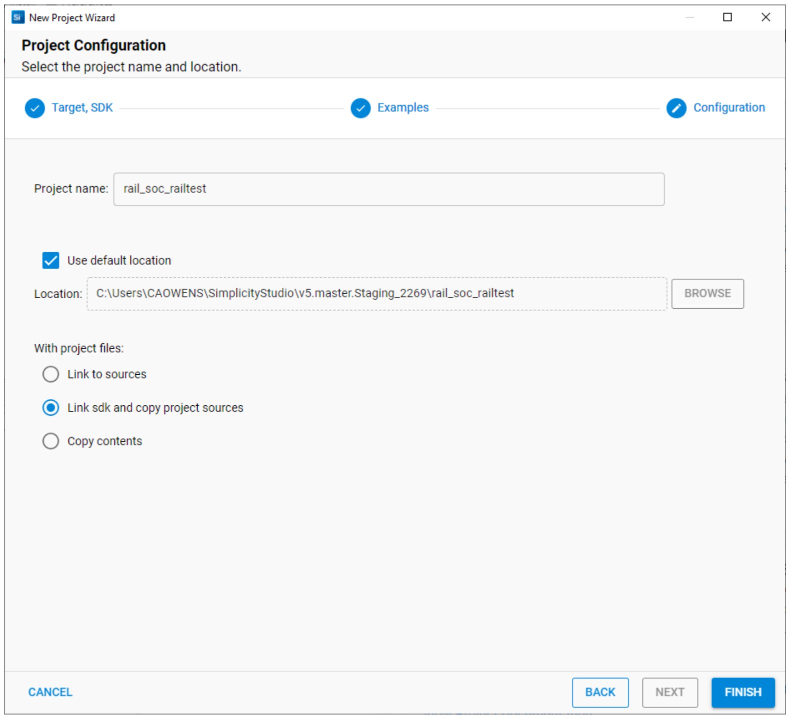
MCU — Clear Filter
- [ ] 32-bit MCU (0)
- [ ] Bootloader (0)

1 resources found

RAIL - SoC RAILtest

The RAILtest application provides a simple tool for testing the radio and the functionality of the RAIL library via CLI. For more advanced usage,...

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** and project generation will start.


New Project Wizard

Project Configuration

Select the project name and location.

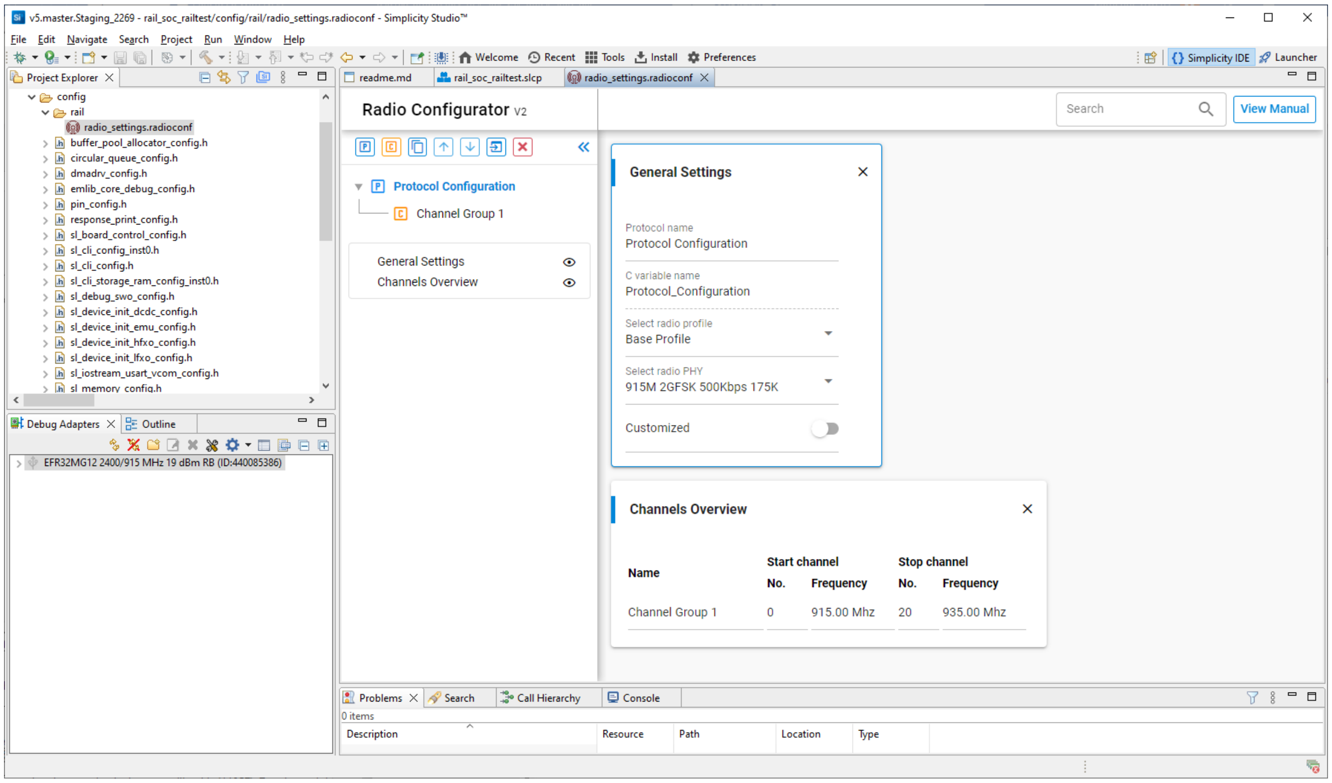
Project name: rail_soc_railtest

[✓] Use default location

Location: C:\Users\CAOWENS\SimplicityStudio\v5.master.Staging_2269\rail_soc_railtest

With project files:
- ○ Link to sources
- ● Link sdk and copy project sources
- ○ Copy contents

4. The new project opens in the Simplicity IDE Perspective with the Radio Configurator tab (*radio_settings.radioconf*) selected. The Project Configurator tab (*<projectname>.slcp*) and a description of the project (*readme.md*) are also available.


v5.master.Staging_2269 - rail_soc_railtest/config/rail/radio_settings.radioconf - Simplicity Studio™
File Edit Navigate Search Project Run Window Help
Welcome Recent Tools Install Preferences
Simplicity IDE Launcher
Project Explorer
config
rail
radio_settings.radioconf
buffer_pool_allocator_config.h
circular_queue_config.h
dmadrv_config.h
emlib_core_debug_config.h
pin_config.h
response_print_config.h
sl_board_control_config.h
sl_cli_config_inst0.h
sl_cli_config.h
sl_cli_storage_ram_config_inst0.h
sl_debug_swo_config.h
sl_device_init_dcdc_config.h
sl_device_init_emu_config.h
sl_device_init_hfxo_config.h
sl_device_init_lfxo_config.h
sl_iostream_usart_vcom_config.h
sl_memory_config.h
Debug Adapters
Outline
EFR32MG12 2400/915 MHz 19 dBm RB (ID:440085386)
readme.md
rail_soc_railtest.slcp
radio_settings.radioconf
Radio Configurator v2
Search
View Manual
Protocol Configuration
Channel Group 1
General Settings
Channels Overview
Problems
Search
Call Hierarchy
Console
0 items
Description
Resource
Path
Location
Type

General Settings

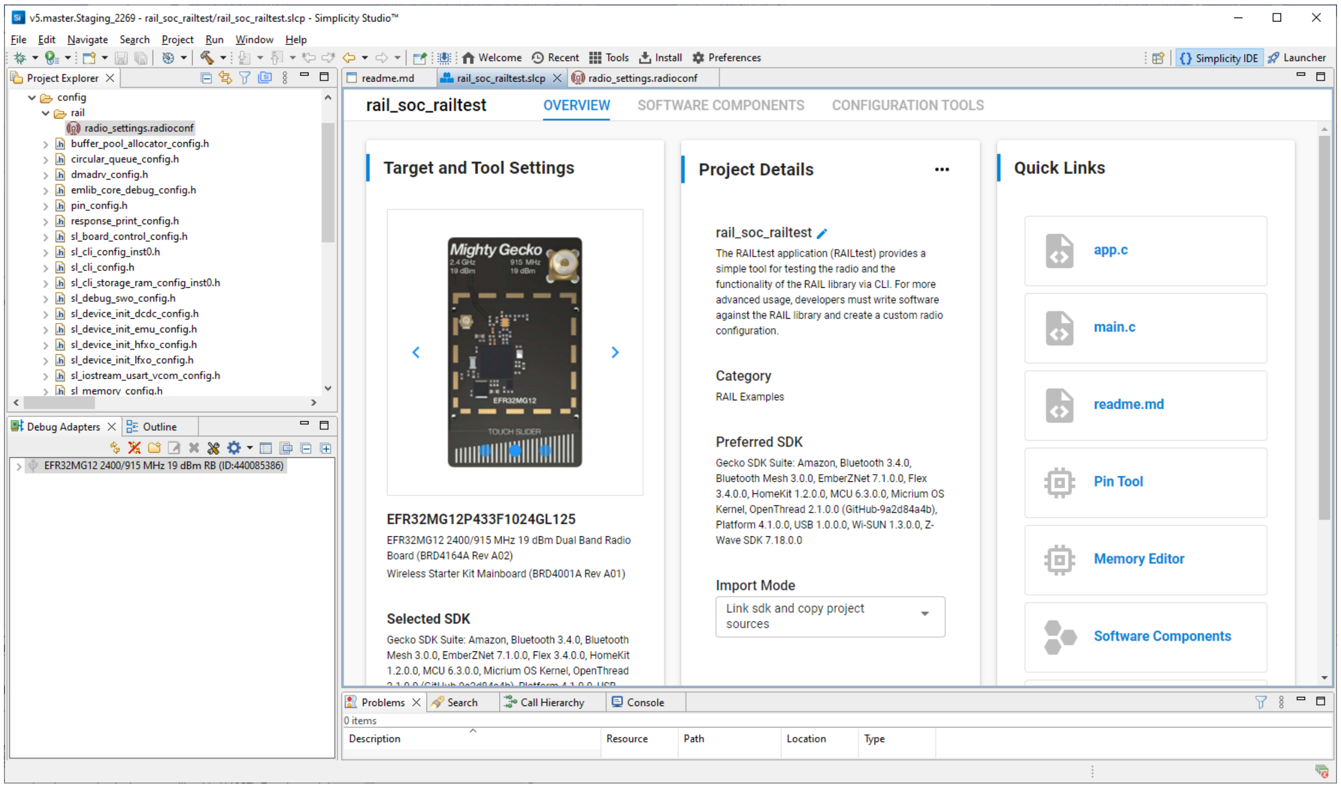
Protocol name: Protocol Configuration
C variable name: Protocol_Configuration
Select radio profile: Base Profile
Select radio PHY: 915M 2GFSK 500Kbps 175K
Customized

Channels Overview

Name	Start channel		Stop channel	
	No.	Frequency	No.	Frequency
Channel Group 1	0	915.00 Mhz	20	935.00 Mhz

3.2 Component Configuration

Click the <projectname>.slcp tab to open the Project Configurator. The default configuration settings will work on the connected development board so, when first experimenting with a development kit, keeping them unchanged and going straight to compiling and flashing is recommended.



EFR32MG12P433F1024GL125
EFR32MG12 2400/915 MHz 19 dBm Dual Band Radio Board (BRD4164A Rev A02)
Wireless Starter Kit Mainboard (BRD4001A Rev A01)

Selected SDK
Gecko SDK Suite: Amazon, Bluetooth 3.4.0, Bluetooth Mesh 3.0.0, EmberZNet 7.1.0.0, Flex 3.4.0.0, HomeKit 1.2.0.0, MCU 6.3.0.0, Micrium OS Kernel, OpenThread

rail_soc_railtest
The RAILtest application (RAILtest) provides a simple tool for testing the radio and the functionality of the RAIL library via CLI. For more advanced usage, developers must write software against the RAIL library and create a custom radio configuration.

Category
RAIL Examples

Preferred SDK
Gecko SDK Suite: Amazon, Bluetooth 3.4.0, Bluetooth Mesh 3.0.0, EmberZNet 7.1.0.0, Flex 3.4.0.0, HomeKit 1.2.0.0, MCU 6.3.0.0, Micrium OS Kernel, OpenThread 2.1.0.0 (GitHub-9a2d84a4b), Platform 4.1.0.0, USB 1.0.0.0, Wi-SUN 1.3.0.0, Z-Wave SDK 7.18.0.0

Import Mode
Link sdk and copy project sources

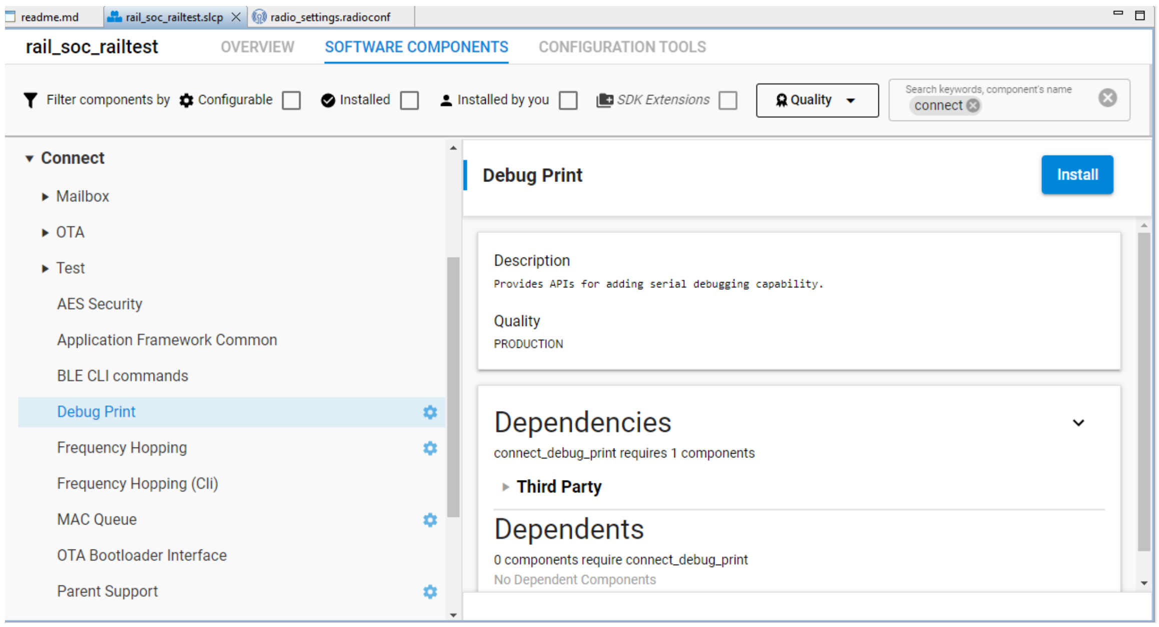
Flex SDK v3.x projects incorporate a Gecko Platform component-based architecture. Software features and functions can be installed and configured through the Component Library. The installation process will:

1. Copy the corresponding SDK files from the SDK folder into the project folder.
2. Copy all the dependencies of the given component into the project folder.
3. Add new include directories to the project settings.
4. Copy the configurations files into the */config* folder.
5. Modify the corresponding auto-generated files to integrate the component into the application.

Additionally, “init” type software components will implement the initialization code for a given component, utilizing their corresponding configuration file as input.

Some software components (like Parent support) will fully integrate into the application to perform a specific task without the need of any additional code, while other components provide an API to be used in the application.

To see the component library, go to the SOFTWARE COMPONENTS tab.



Debug Print

Description
Provides APIs for adding serial debugging capability.

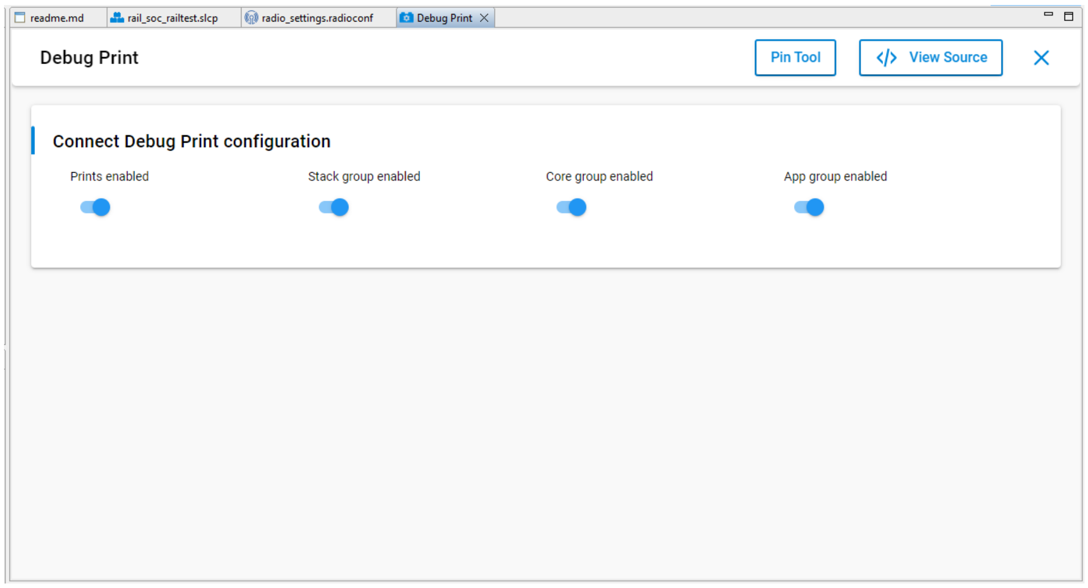
Quality
PRODUCTION

Dependencies
connect_debug_print requires 1 components
Third Party

Dependents
0 components require connect_debug_print
No Dependent Components

The project is configured by installing and uninstalling components, and configuring installed components. Installed components are checked. Configurable components have a gear symbol. Select a component to see information about it. A number of filters as well as a keyword search are available to help you explore the various component categories. Note that components for all installed SDKs are presented. See the online *Simplicity Studio 5 User's Guide* for details about the functionality available through the Simplicity IDE perspective and the Project Configurator.

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

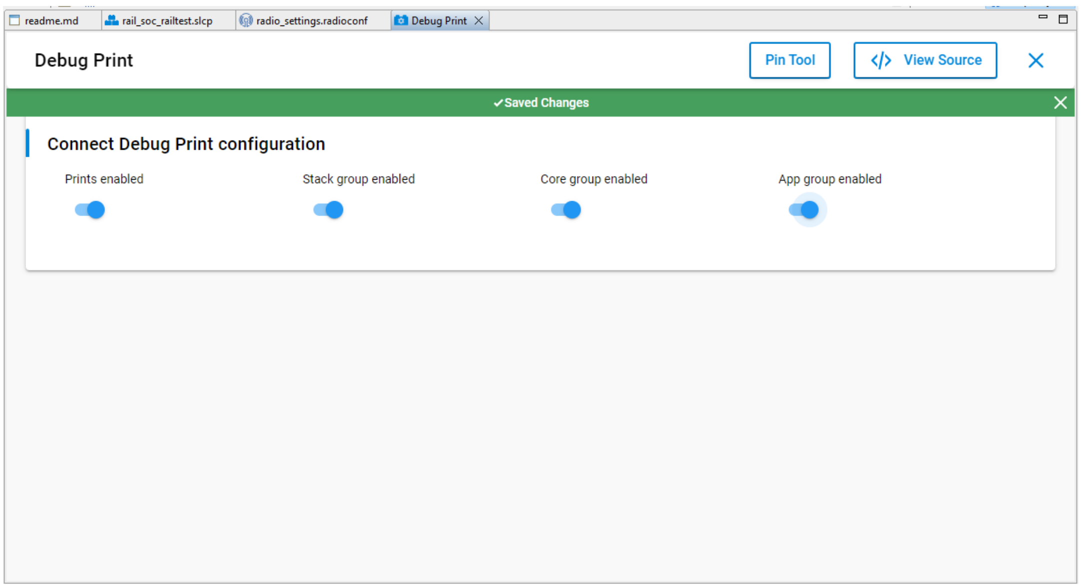


Debug Print

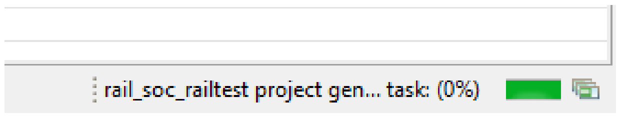
Connect Debug Print configuration

Prints enabled: on
Stack group enabled: on
Core group enabled: on
App group enabled: on

Any changes you make in the Component Editor are autosaved.

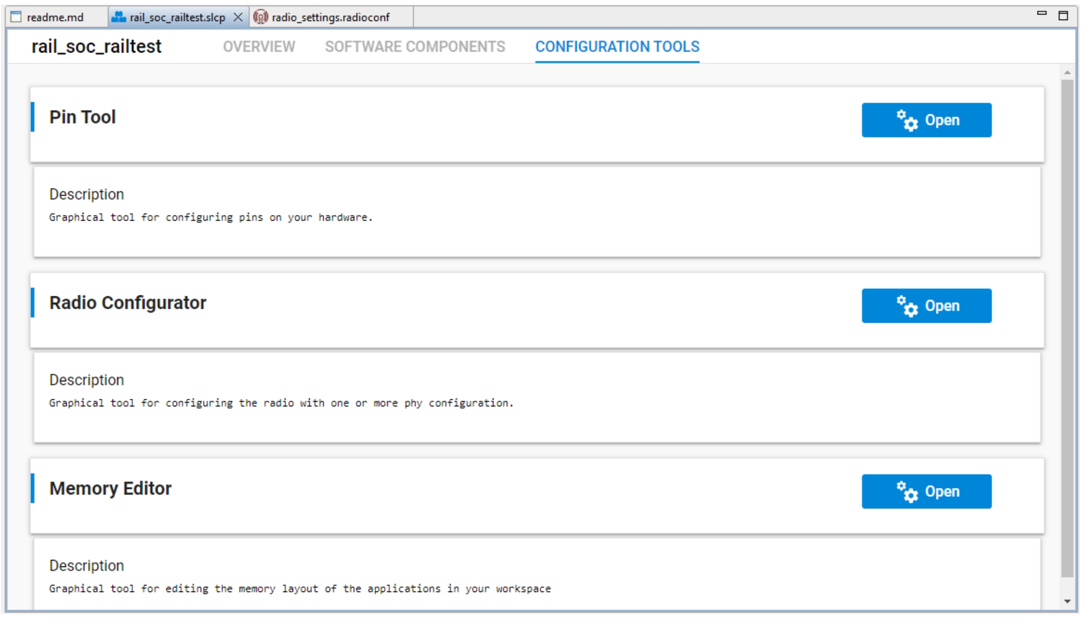


As you install and otherwise make changes in the Project Configurator, project files are autogenerated. Progress is shown in the lower right corner of the Simplicity IDE perspective.

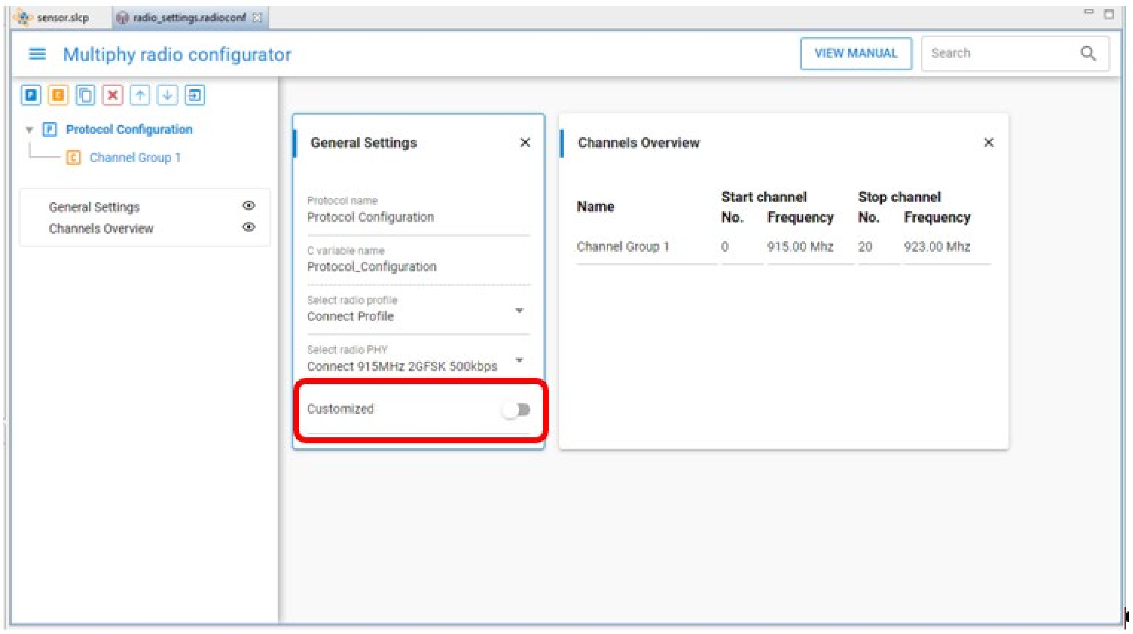


3.3 Radio Configuration

If the *radio_settings.radioconf* tab is not present, click the Project Configurator CONFIGURATION TOOLS tab. Click **Open** next to the Radio Configurator. It will open the *radio_settings.radioconf* file of the **/config/rai**l folder in the Radio Configurator.


readme.md
rail_soc_railtest.slcp
radio_settings.radioconf
rail_soc_railtest
OVERVIEW
SOFTWARE COMPONENTS
CONFIGURATION TOOLS
Pin Tool
Open
Description
Graphical tool for configuring pins on your hardware.
Radio Configurator
Open
Description
Graphical tool for configuring the radio with one or more phy configuration.
Memory Editor
Open
Description
Graphical tool for editing the memory layout of the applications in your workspace

Select a radio profile, and a pre-defined radio PHY of the selected profile. You can then customize the PHY by turning on **Customized**. Changes made in the Radio Configurator are not autosaved. When saving the changes to *radio_settings.radioconf*, the radio configuration will be autogenerated and saved in the */autogen/rail_config.h* and */autogen/rail_config.c* files. See *AN1253: EFR32 Radio Configurator Guide for Simplicity Studio 5* for more information on using the Radio Configurator.



General Settings

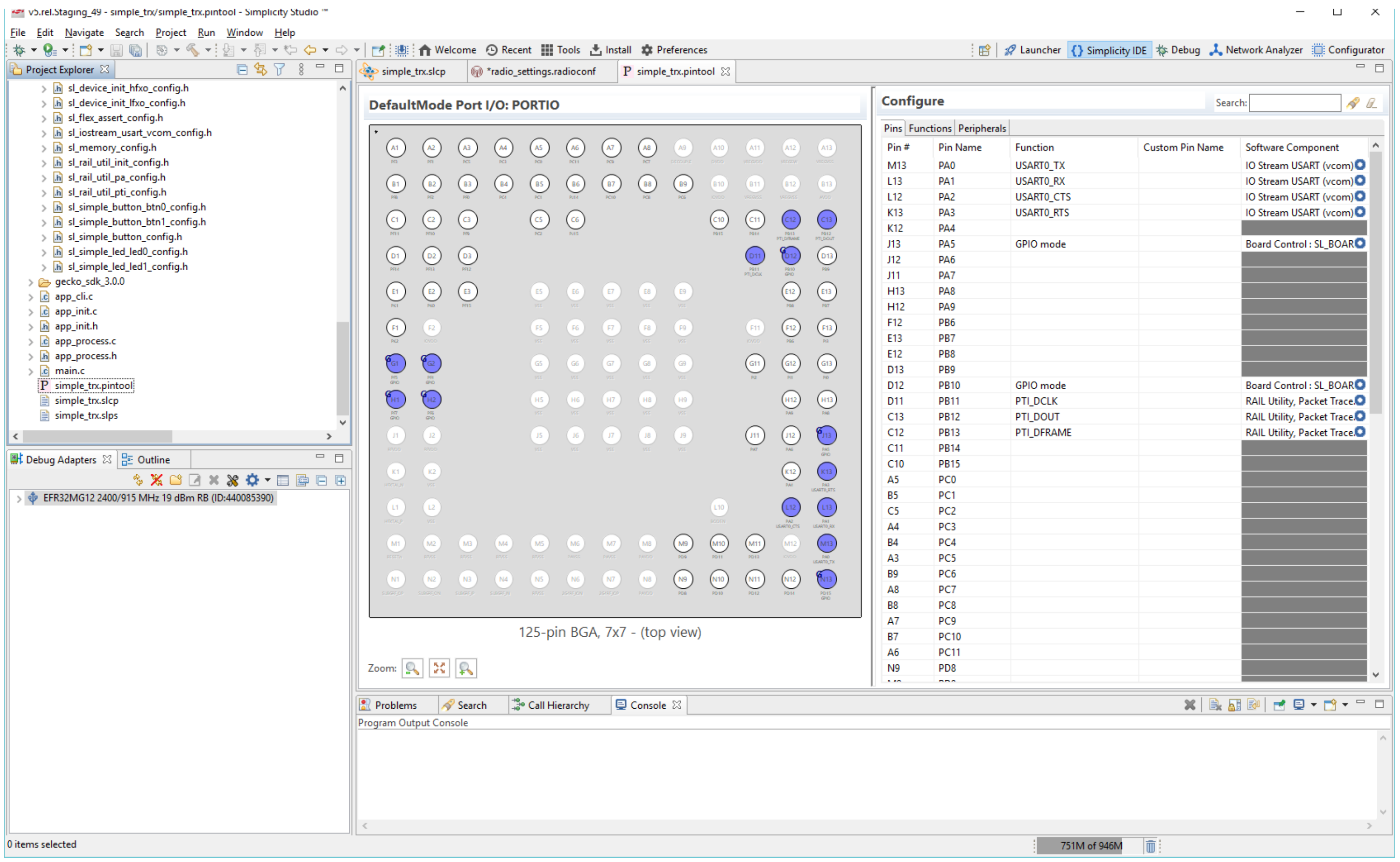
Protocol name: Protocol Configuration
C variable name: Protocol_Configuration
Select radio profile: Connect Profile
Select radio PHY: Connect 915MHz 2GFSK 500kbps
Customized

Channels Overview

Name	Start channel No.	Start channel Frequency	Stop channel No.	Stop channel Frequency
Channel Group 1	0	915.00 Mhz	20	923.00 Mhz

3.4 Pin Tool

On the CONFIGURATION TOOLS tab, click **Open** next to Pin Tool. This will open the file *<project>.pintool* of the project in the Pin Tool, which allows you to easily configure new peripherals or change the properties of existing ones. You can also double-click the file *<project>.pintool* in the Project Explorer view to open it.


v5.rel.Staging_49 - simple_trx/simple_trx.pintool - Simplicity Studio ™

Project Explorer:

```
sl_device_init_hfxo_config.h
sl_device_init_lfxo_config.h
sl_flex_assert_config.h
sl_iostream_usart_vcom_config.h
sl_memory_config.h
sl_rail_util_init_config.h
sl_rail_util_pa_config.h
sl_rail_util_pti_config.h
sl_simple_button_btn0_config.h
sl_simple_button_btn1_config.h
sl_simple_button_config.h
sl_simple_led_led0_config.h
sl_simple_led_led1_config.h
gecko_sdk_3.0.0
app_cli.c
app_init.c
app_init.h
app_process.c
app_process.h
main.c
simple_trx.pintool
simple_trx.slcp
simple_trx.slps
```

Debug Adapters: EFR32MG12 2400/915 MHz 19 dBm RB (ID:440085390)

DefaultMode Port I/O: PORTIO

125-pin BGA, 7x7 - (top view)

Configure

Pin #	Pin Name	Function	Custom Pin Name	Software Component
M13	PA0	USART0_TX		IO Stream USART (vcom)
L13	PA1	USART0_RX		IO Stream USART (vcom)
L12	PA2	USART0_CTS		IO Stream USART (vcom)
K13	PA3	USART0_RTS		IO Stream USART (vcom)
K12	PA4			
J13	PA5	GPIO mode		Board Control : SL_BOAR
J12	PA6			
J11	PA7			
H13	PA8			
H12	PA9			
F12	PB6			
E13	PB7			
E12	PB8			
D13	PB9			
D12	PB10	GPIO mode		Board Control : SL_BOAR
D11	PB11	PTI_DCLK		RAIL Utility, Packet Trace
C13	PB12	PTI_DOUT		RAIL Utility, Packet Trace
C12	PB13	PTI_DFRAME		RAIL Utility, Packet Trace
C11	PB14			
C10	PB15			
A5	PC0			
B5	PC1			
C5	PC2			
A4	PC3			
B4	PC4			
A3	PC5			
B9	PC6			
A8	PC7			
B8	PC8			
A7	PC9			
B7	PC10			
A6	PC11			
N9	PD8			

Program Output Console

0 items selected

751M of 946M

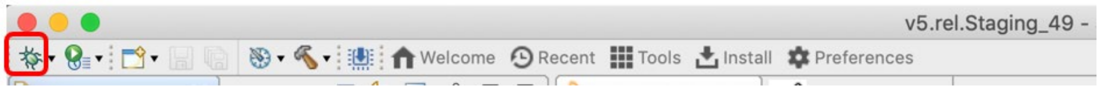
Double-click a Software Component to open the Component Editor and configure that function. Pin Tool does not autosave.

3.5 Compiling and Flashing the Application

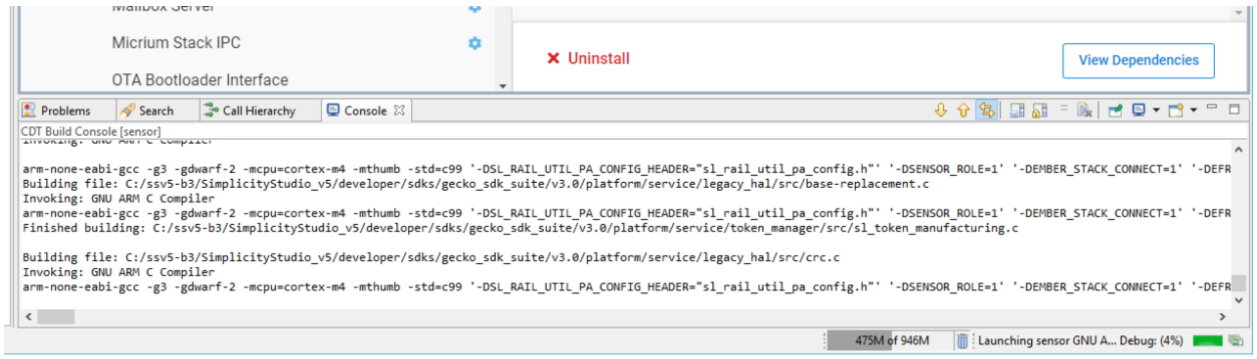
You can either compile and flash the application automatically, or manually compile it and then flash it.

3.5.1 Automatically Compile and Flash

You can automatically compile and flash the application to your connected development hardware in the Simplicity IDE. Click the **Debug** control.



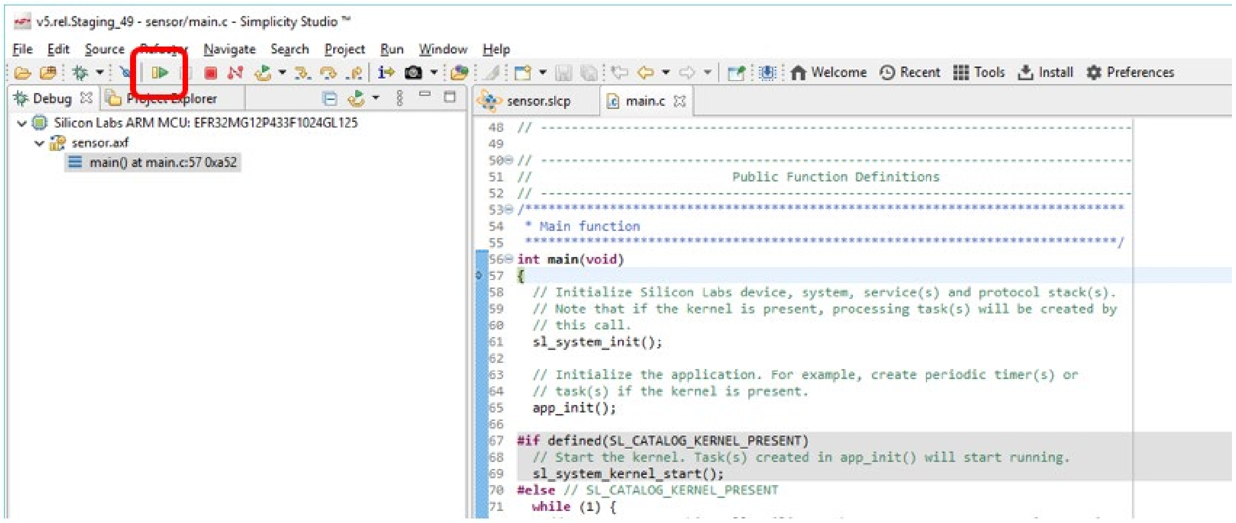
Progress is displayed in the console and a progress bar in the lower right.



```
arm-none-eabi-gcc -g3 -gdwarf-2 -mcpu=cortex-m4 -mthumb -std=c99 '-DSL_RAIL_UTIL_PA_CONFIG_HEADER="sl_rail_util_pa_config.h"' '-DSENSOR_ROLE=1' '-DEMBER_STACK_CONNECT=1' '-DEFR
Building file: C:/ssv5-b3/SimplicityStudio_v5/developer/sdks/gecko_sdk_suite/v3.0/platform/service/legacy_hal/src/base-replacement.c
Invoking: GNU ARM C Compiler
arm-none-eabi-gcc -g3 -gdwarf-2 -mcpu=cortex-m4 -mthumb -std=c99 '-DSL_RAIL_UTIL_PA_CONFIG_HEADER="sl_rail_util_pa_config.h"' '-DSENSOR_ROLE=1' '-DEMBER_STACK_CONNECT=1' '-DEFR
Finished building: C:/ssv5-b3/SimplicityStudio_v5/developer/sdks/gecko_sdk_suite/v3.0/platform/service/token_manager/src/sl_token_manufacturing.c

Building file: C:/ssv5-b3/SimplicityStudio_v5/developer/sdks/gecko_sdk_suite/v3.0/platform/service/legacy_hal/src/crc.c
Invoking: GNU ARM C Compiler
arm-none-eabi-gcc -g3 -gdwarf-2 -mcpu=cortex-m4 -mthumb -std=c99 '-DSL_RAIL_UTIL_PA_CONFIG_HEADER="sl_rail_util_pa_config.h"' '-DSENSOR_ROLE=1' '-DEMBER_STACK_CONNECT=1' '-DEFR
```

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



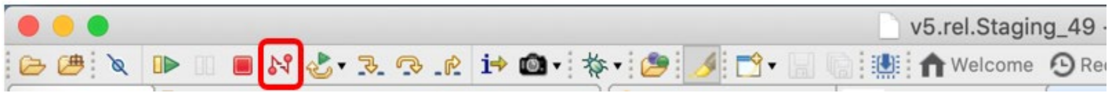
```
// ---------------------------------------------------------------------------

// ---------------------------------------------------------------------------
//                          Public Function Definitions
// ---------------------------------------------------------------------------
/*****************************************************************************
 * Main function
 *****************************************************************************/
int main(void)
{
  // Initialize Silicon Labs device, system, service(s) and protocol stack(s).
  // Note that if the kernel is present, processing task(s) will be created by
  // this call.
  sl_system_init();

  // Initialize the application. For example, create periodic timer(s) or
  // task(s) if the kernel is present.
  app_init();

#if defined(SL_CATALOG_KERNEL_PRESENT)
  // Start the kernel. Task(s) created in app_init() will start running.
  sl_system_kernel_start();
#else // SL_CATALOG_KERNEL_PRESENT
  while (1) {
```

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

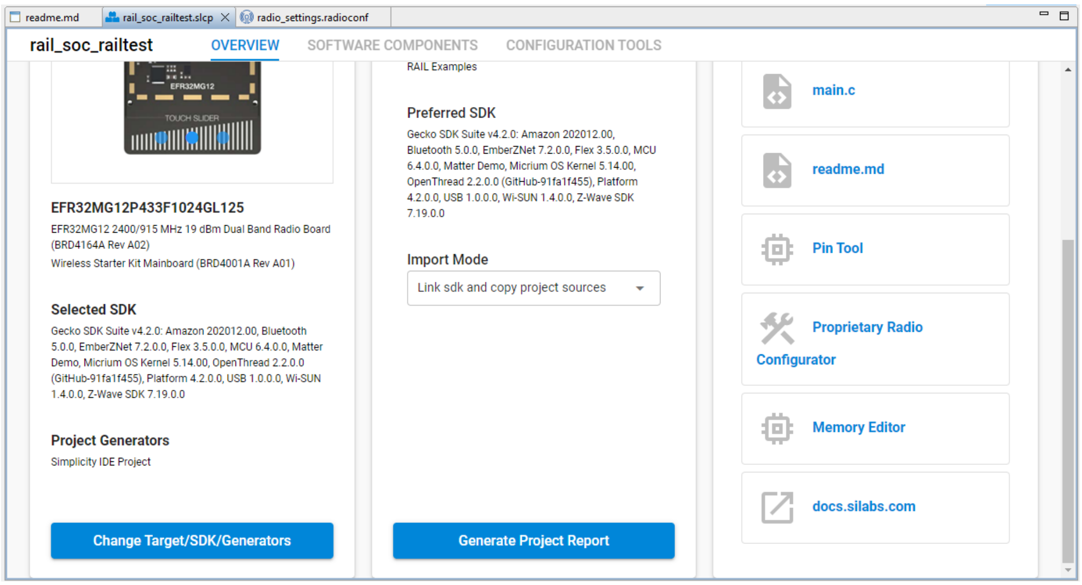


3.5.2 Manually Compile and Flash

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

The sample application will compile based on its build configuration. You can change the build configuration at any time by right-clicking the project directory in Project Explorer view and opening **Build Configurations > Set Active**.

You can also build your application directly in IAR-EWARM by opening IAR-EWARM and opening the generated project file inside IAR. To generate the file for IAR EWARM, go to the OVERVIEW tab of the Project Configurator, scroll down, and click **Change Target/SDK/Generators** on the Target and Tool Settings card, drop down the list under CHANGE PROJECT GENERATORES, and include IAR EMBEDDED WORKBENCH PROJECT. The .eww file is created after you save the change.



EFR32MG12P433F1024GL125
EFR32MG12 2400/915 MHz 19 dBm Dual Band Radio Board (BRD4164A Rev A02)
Wireless Starter Kit Mainboard (BRD4001A Rev A01)

Selected SDK
Gecko SDK Suite v4.2.0: Amazon 202012.00, Bluetooth 5.0.0, EmberZNet 7.2.0.0, Flex 3.5.0.0, MCU 6.4.0.0, Matter Demo, Micrium OS Kernel 5.14.00, OpenThread 2.2.0.0 (GitHub-91fa1f455), Platform 4.2.0.0, USB 1.0.0.0, Wi-SUN 1.4.0.0, Z-Wave SDK 7.19.0.0

Project Generators
Simplicity IDE Project

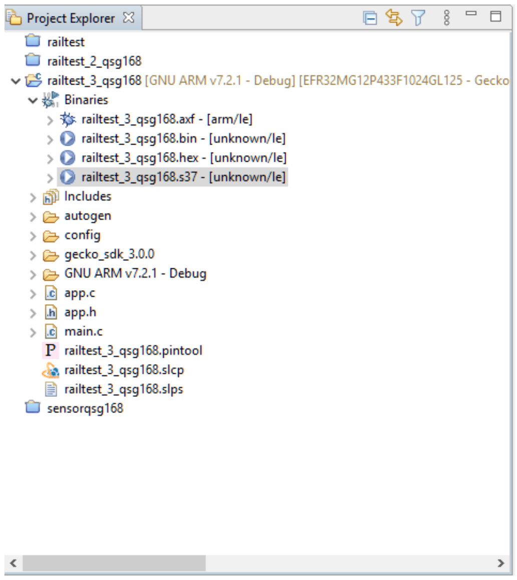
RAIL Examples

Preferred SDK
Gecko SDK Suite v4.2.0: Amazon 202012.00, Bluetooth 5.0.0, EmberZNet 7.2.0.0, Flex 3.5.0.0, MCU 6.4.0.0, Matter Demo, Micrium OS Kernel 5.14.00, OpenThread 2.2.0.0 (GitHub-91fa1f455), Platform 4.2.0.0, USB 1.0.0.0, Wi-SUN 1.4.0.0, Z-Wave SDK 7.19.0.0

Import Mode
Link sdk and copy project sources

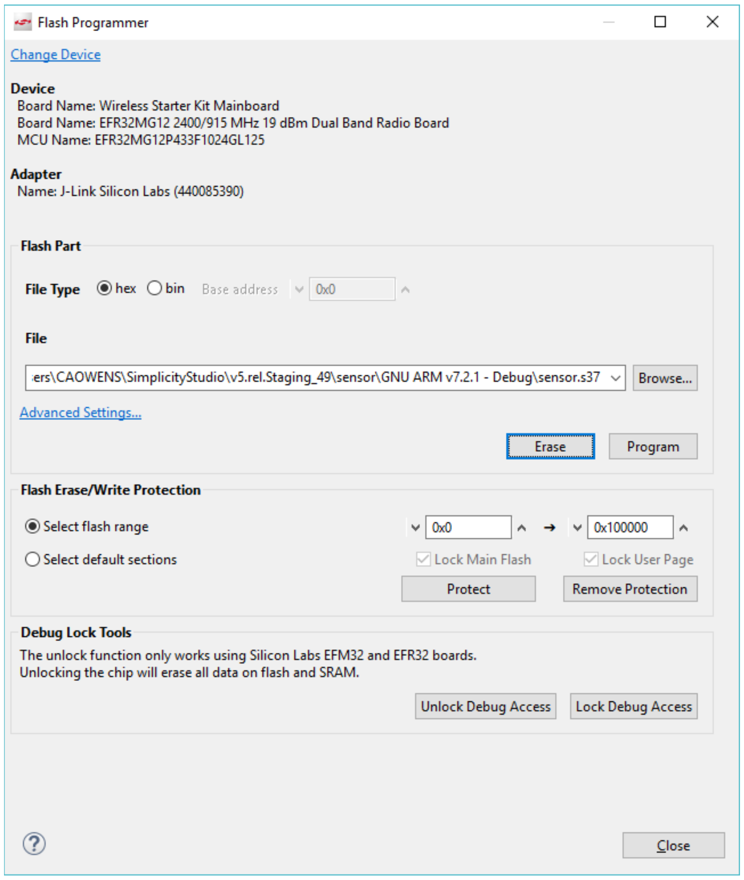
1. Open IAR-EWARM.
2. Select **File > Open > Workspace** and navigate to the location you selected for your sample application.
3. Select the application *.eww* file and click **Open**.
4. Select **Project > Make** or press F7. If the application builds without errors, you are ready to install the image on a device.

You can load the binary image through Project Explorer view. Locate the <project>.bin, .hex, or .s37 file in the **Binaries** subdirectory.



```
Project Explorer
  railtest
  railtest_2_qsg168
v railtest_3_qsg168 [GNU ARM v7.2.1 - Debug] [EFR32MG12P433F1024GL125 - Gecko
  v Binaries
    > railtest_3_qsg168.axf - [arm/le]
    > railtest_3_qsg168.bin - [unknown/le]
    > railtest_3_qsg168.hex - [unknown/le]
    > railtest_3_qsg168.s37 - [unknown/le]
  > Includes
  > autogen
  > config
  > gecko_sdk_3.0.0
  > GNU ARM v7.2.1 - Debug
  > app.c
  > app.h
  > main.c
    railtest_3_qsg168.pintool
    railtest_3_qsg168.slcp
    railtest_3_qsg168.slps
  sensorqsg168
```

Right-click the file and select **Flash Programmer**. The Flash Programmer opens with the file path populated. Click **Program.**



Flash Programmer

Change Device

Device
Board Name: Wireless Starter Kit Mainboard
Board Name: EFR32MG12 2400/915 MHz 19 dBm Dual Band Radio Board
MCU Name: EFR32MG12P433F1024GL125

Adapter
Name: J-Link Silicon Labs (440085390)

Flash Part

File Type (●) hex ( ) bin Base address 0x0

File

ers\CAOWENS\SimplicityStudio\v5.rel.Staging_49\sensor\GNU ARM v7.2.1 - Debug\sensor.s37 Browse...

Advanced Settings...

Erase Program

Flash Erase/Write Protection

(●) Select flash range 0x0 → 0x100000

( ) Select default sections ☑ Lock Main Flash ☑ Lock User Page

Protect Remove Protection

Debug Lock Tools

The unlock function only works using Silicon Labs EFM32 and EFR32 boards.
Unlocking the chip will erase all data on flash and SRAM.

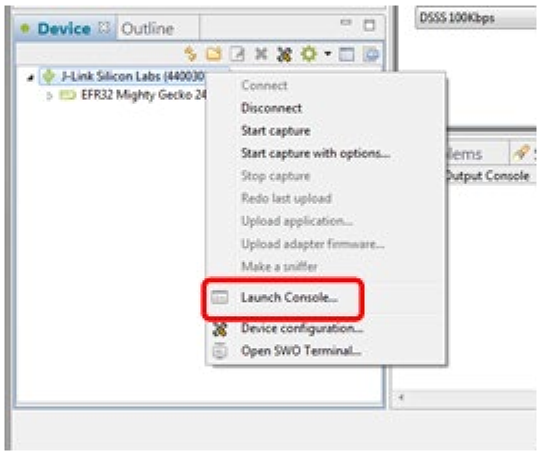
Unlock Debug Access Lock Debug Access

Close

3.6 Interacting with Examples

Depending on the example application, you may be able to interact with it through buttons, LEDs and the LCD of the mainboard and through your development environment's Console interface using a CLI (command line interpreter).

To launch the Console interface, in the Simplicity IDE perspective right-click on the debug adapter in the Debug Adapters view. Choose **Launch Console**. Alternatively, from the Tools icon in the Simplicity IDE toolbar, or on the COMPATIBLE TOOLS tab in the Launcher perspective, select **Device Console**. Select the Serial 1 tab and click Enter to get a prompt from the sample app CLI.



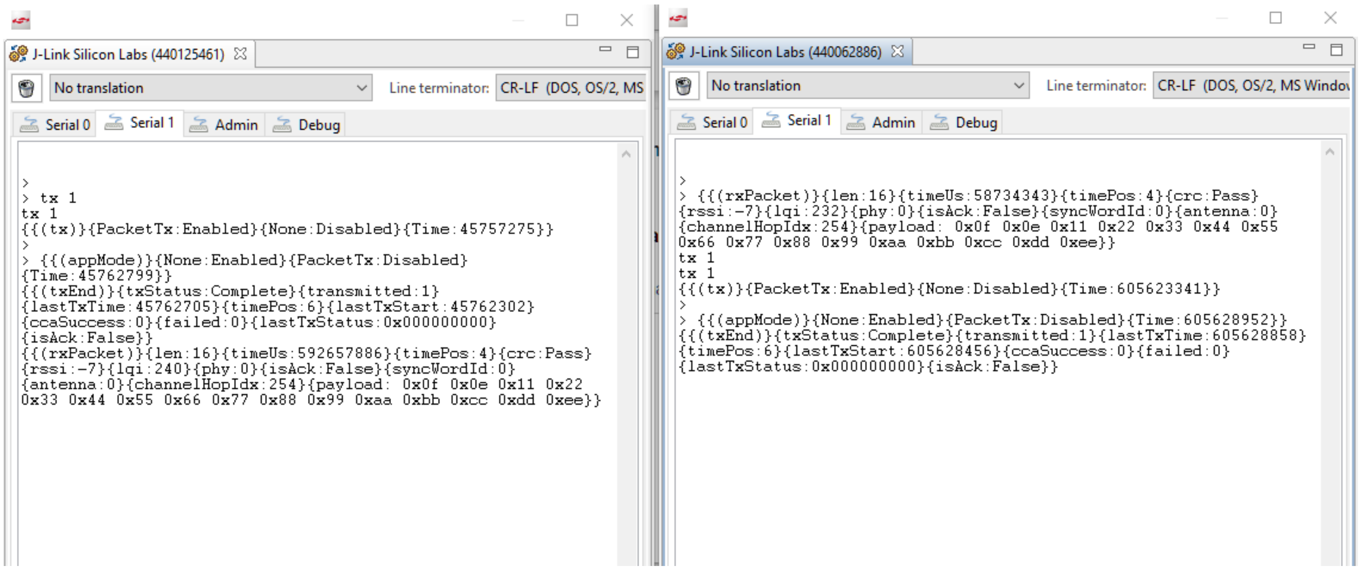
In the RAILtest application the console interface allows testing of any RAIL feature. Enter "help" to get a list of the available CLI commands.

For a simple radio functionality test flash the same RAILtest binary on two radio boards and open a console for each of them. Make sure an antenna is connected to each radio board if applicable. RAILtest starts in RX state on both boards.

Type `tx 1` in the first console window. The first radio node will transmit one packet and then return to RX state. A received packet will be reported in the second console window by the second radio node.

Type `tx 1` in the second console window. The second radio node will transmit one packet and then return to RX state. A received packet will be reported in the first console window by the first radio node.

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


J-Link Silicon Labs (440125461)
No translation
Line terminator: CR-LF (DOS, OS/2, MS
Serial 0
Serial 1
Admin
Debug

```
>
> tx 1
tx 1
{{(tx)}{PacketTx:Enabled}{None:Disabled}{Time:45757275}}
>
> {{(appMode)}{None:Enabled}{PacketTx:Disabled}
{Time:45762799}}
{{(txEnd)}{txStatus:Complete}{transmitted:1}
{lastTxTime:45762705}{timePos:6}{lastTxStart:45762302}
{ccaSuccess:0}{failed:0}{lastTxStatus:0x000000000}
{isAck:False}}
{{(rxPacket)}{len:16}{timeUs:592657886}{timePos:4}{crc:Pass}
{rssi:-7}{lqi:240}{phy:0}{isAck:False}{syncWordId:0}
{antenna:0}{channelHopIdx:254}{payload: 0x0f 0x0e 0x11 0x22
0x33 0x44 0x55 0x66 0x77 0x88 0x99 0xaa 0xbb 0xcc 0xdd 0xee}}
```

J-Link Silicon Labs (440062886)
No translation
Line terminator: CR-LF (DOS, OS/2, MS Windov
Serial 0
Serial 1
Admin
Debug

```
>
> {{(rxPacket)}{len:16}{timeUs:58734343}{timePos:4}{crc:Pass}
{rssi:-7}{lqi:232}{phy:0}{isAck:False}{syncWordId:0}{antenna:0}
{channelHopIdx:254}{payload: 0x0f 0x0e 0x11 0x22 0x33 0x44 0x55
0x66 0x77 0x88 0x99 0xaa 0xbb 0xcc 0xdd 0xee}}
tx 1
tx 1
{{(tx)}{PacketTx:Enabled}{None:Disabled}{Time:605623341}}
>
> {{(appMode)}{None:Enabled}{PacketTx:Disabled}{Time:605628952}}
{{(txEnd)}{txStatus:Complete}{transmitted:1}{lastTxTime:605628858}
{timePos:6}{lastTxStart:605628456}{ccaSuccess:0}{failed:0}
{lastTxStatus:0x000000000}{isAck:False}}
```

3.7 Working with Custom Boards

When working with custom boards, two methods are available, depending on your needs:

- Create an example on the mainboard, then remove the mainboard on the Project Configurator OVERVIEW tab using the **Change Target/SDK** control. This way, the wiring of the kit is kept, but you can change it using the Pin Tool.
- Create an example with just the part, no mainboard, and set up the wiring for all required peripherals of the example (like buttons or pins) using the Pin Tool.

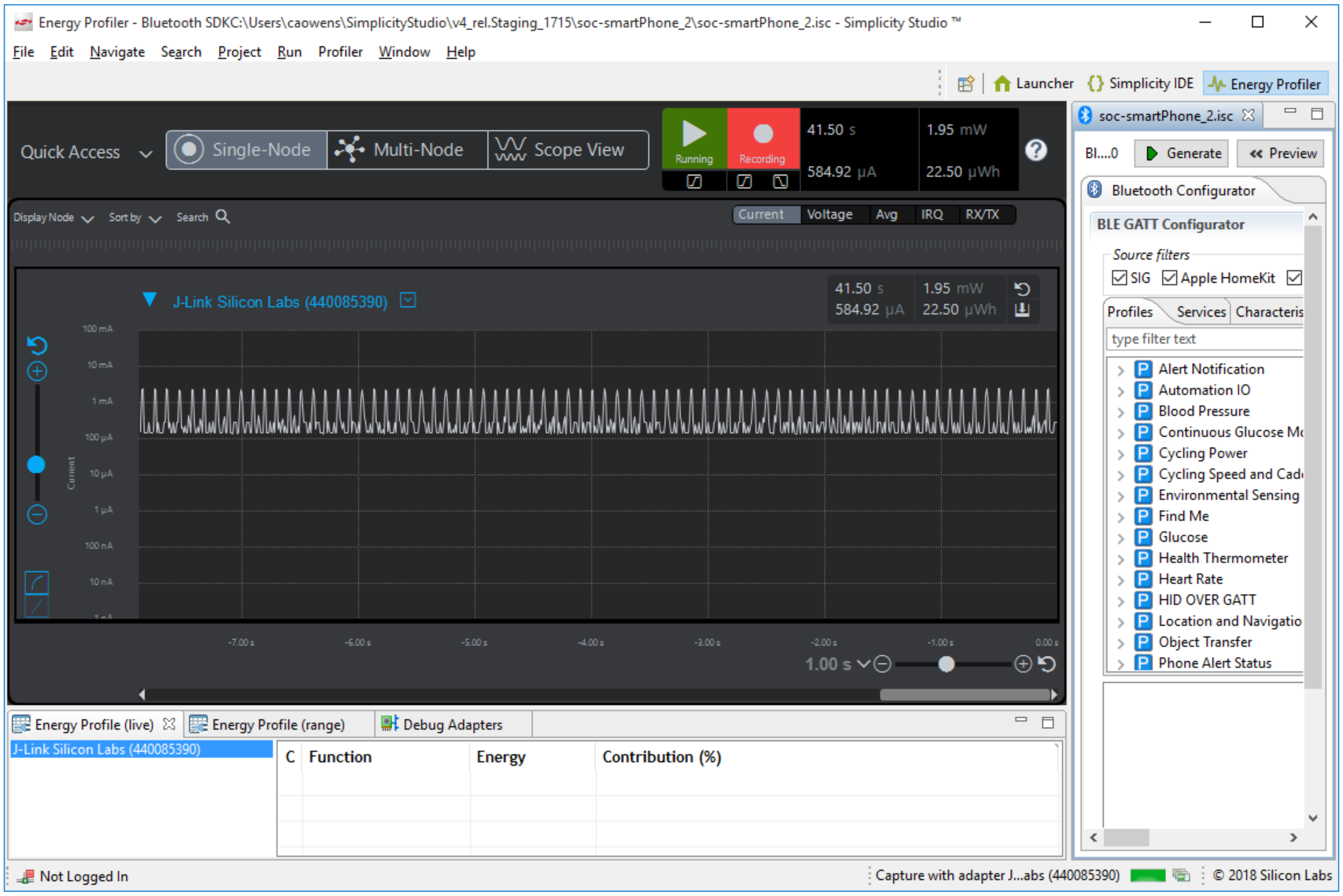
Note that a mainboard can also depend on components (typically peripheral init components) that are removed if you remove the main-board from the project.

4 Next Steps

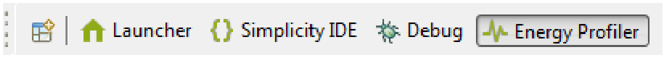
4.1 Multi-Node Energy Profiler

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

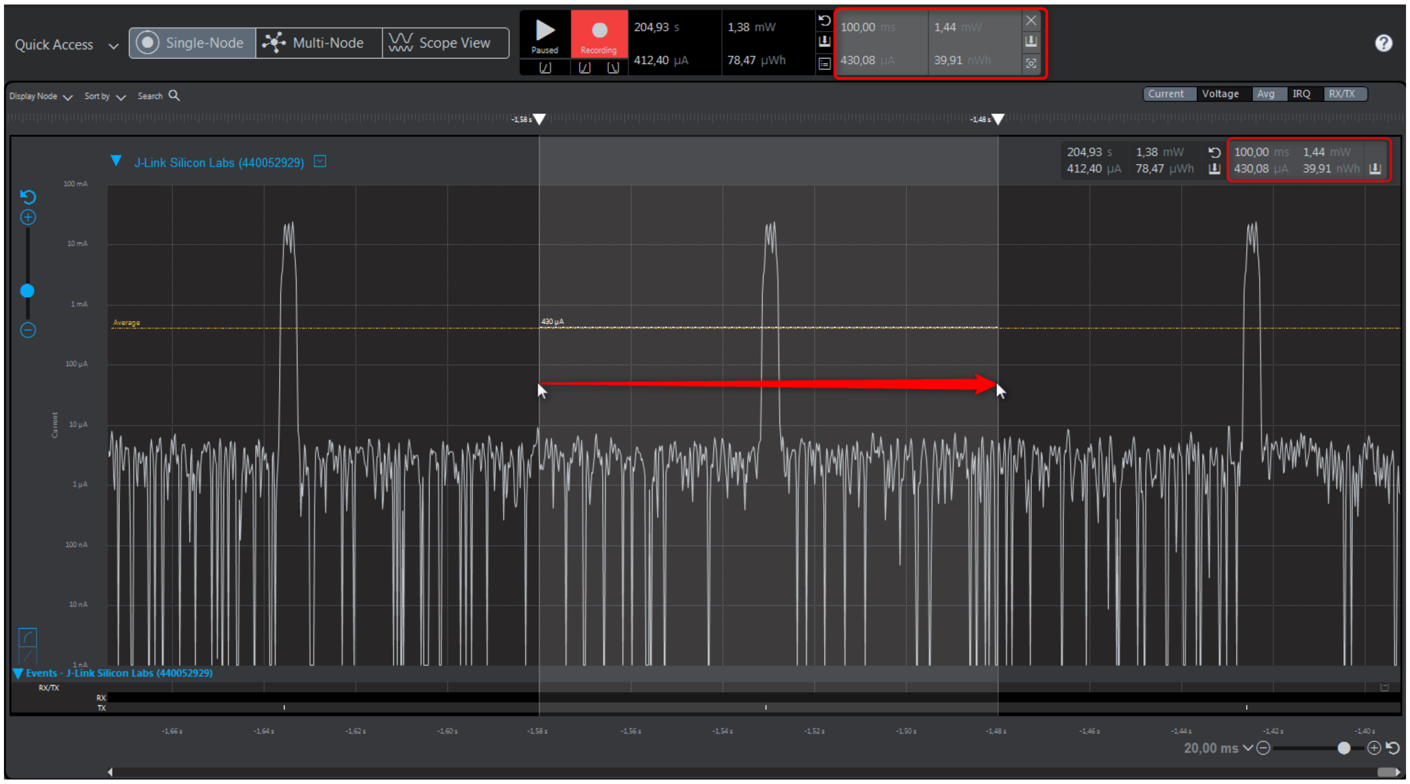
To profile the current project, drop down the Profile as menu in the Simplicity IDE perspective and select **Profile as / Simplicity Energy Profiler target**. This automatically builds your project, uploads it to the device, and starts Energy Profiler. A new Energy Profiler perspective appears, shown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



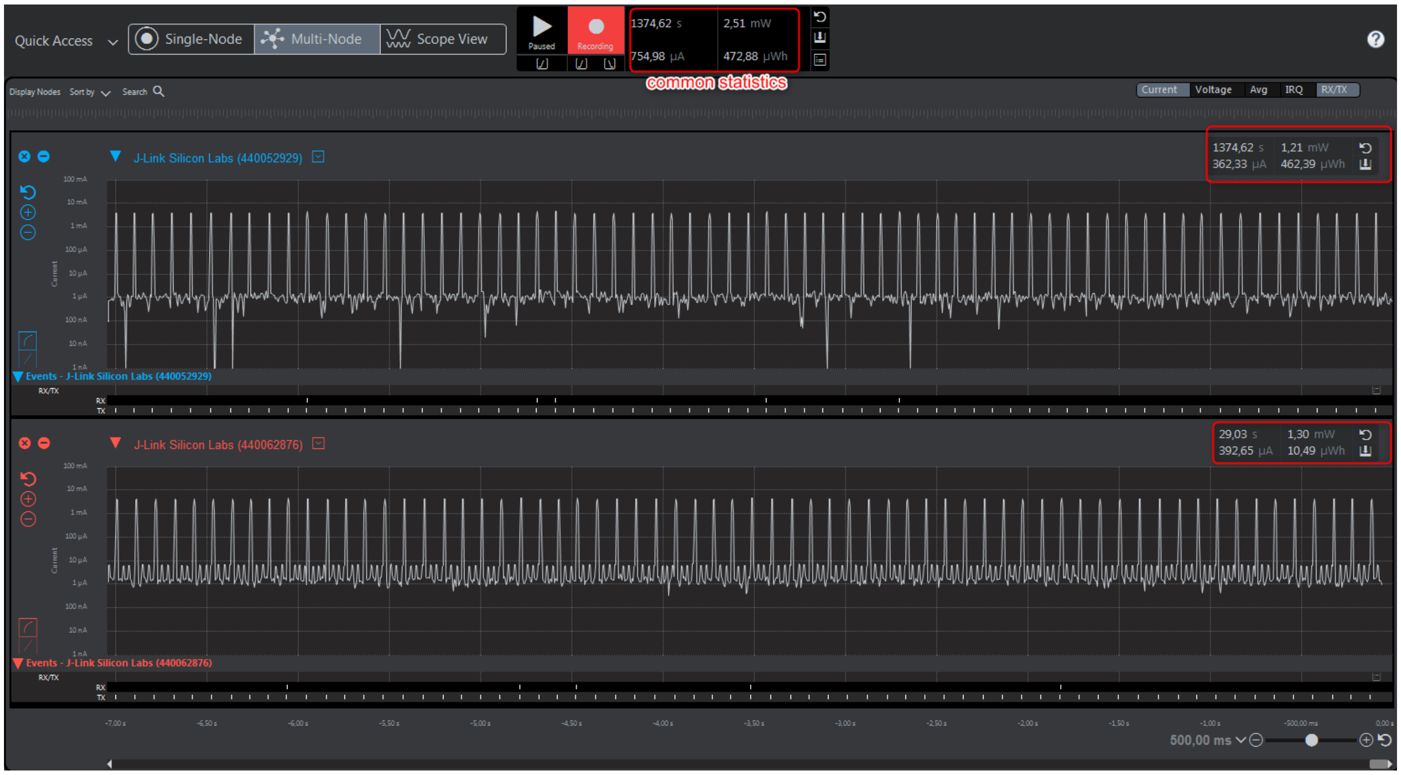
See *UG343: Multi-Node Energy Profiler User's Guide* for details on how to use this tool. You can switch easily between Simplicity IDE and Energy Profiler perspectives using the Perspective buttons in the upper right corner of your current perspective.



You can see peaks in the energy consumption diagram. Pause profiling by clicking the green Play button, click one of the peaks, and zoom in with time axis (y-axis) zoom until you see sufficient details. These usually represent packet transmissions . You can also see the corresponding Tx events in the Rx/Tx bar below, provided that you enabled Rx/Tx view in the upper right corner. 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, 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.. Overall average is measured as well, but this is influenced by transient events.



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) and select **Start Energy Capture**. To stop measuring, click the Quick Access menu, and select **End/Save session**.

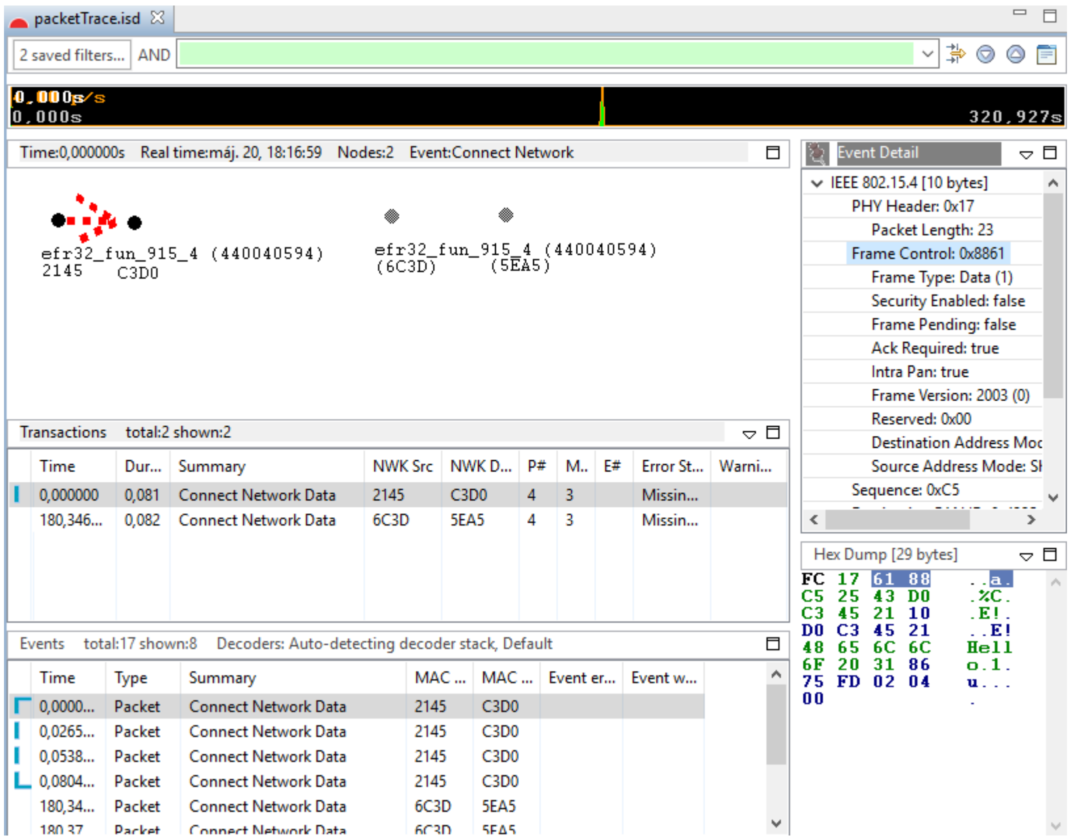


To learn more about how to use this tool, see *UG343: Multi-Node Energy Profiler User's Guide*.

4.2 Network Analyzer

Silicon Labs Network Analyzer is a packet capture and debugging tool that can be used to debug connectivity between Wireless Geckos and other devices. It significantly accelerates the network and application development process with graphical views of network traffic, activity, and duration.

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.



Time:0,000000s Real time:máj. 20, 18:16:59 Nodes:2 Event:Connect Network

Transactions total:2 shown:2

Time	Dur...	Summary	NWK Src	NWK D...	P#	M..	E#	Error St...	Warni...
0,000000	0,081	Connect Network Data	2145	C3D0	4	3		Missin...	
180,346...	0,082	Connect Network Data	6C3D	5EA5	4	3		Missin...	

Events total:17 shown:8 Decoders: Auto-detecting decoder stack, Default

Time	Type	Summary	MAC ...	MAC ...	Event er...	Event w...
0,0000...	Packet	Connect Network Data	2145	C3D0		
0,0265...	Packet	Connect Network Data	2145	C3D0		
0,0538...	Packet	Connect Network Data	2145	C3D0		
0,0804...	Packet	Connect Network Data	2145	C3D0		
180,34...	Packet	Connect Network Data	6C3D	5EA5		
180,37	Packet	Connect Network Data	6C3D	5EA5		

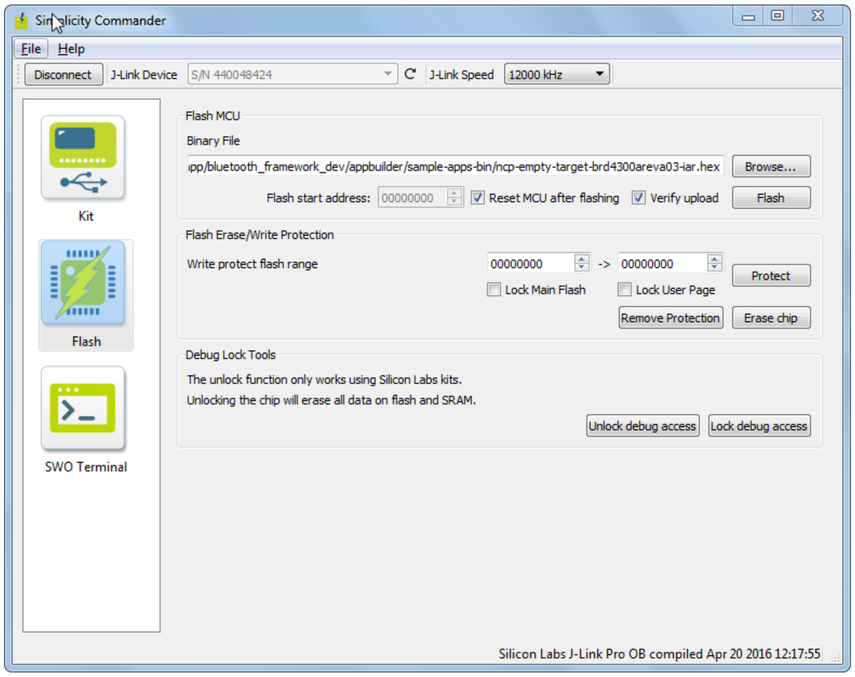
IEEE 802.15.4 [10 bytes]
- PHY Header: 0x17
  - Packet Length: 23
- Frame Control: 0x8861
  - Frame Type: Data (1)
  - Security Enabled: false
  - Frame Pending: false
  - Ack Required: true
  - Intra Pan: true
  - Frame Version: 2003 (0)
  - Reserved: 0x00
  - Destination Address Moc
  - Source Address Mode: Sh
- Sequence: 0xC5

```
FC 17 61 88   ..a.
C5 25 43 D0   .%C.
C3 45 21 10   .E!.
D0 C3 45 21   ..E!
48 65 6C 6C   Hell
6F 20 31 86   o.1.
75 FD 02 04   u...
00            .
```

Figure 4-1. Connect Traffic Capture with Packet Trace

4.3 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.



S/N 440048424
12000 kHz
Binary File: ıpp/bluetooth_framework_dev/appbuilder/sample-apps-bin/ncp-empty-target-brd4300areva03-iar.hex
Flash start address: 00000000 ☑ Reset MCU after flashing ☑ Verify upload
Write protect flash range 00000000 -> 00000000
☐ Lock Main Flash ☐ Lock User Page
The unlock function only works using Silicon Labs kits.
Unlocking the chip will erase all data on flash and SRAM.
Silicon Labs J-Link Pro OB compiled Apr 20 2016 12:17:55

Figure 4-2. Simplicity Commander