QSG138: Getting Started with the Silicon Labs Flex Software Development Kit for the Wireless Gecko (EFR32™) Portfolio

This quick start guide provides basic information on configuring, building, and installing applications for the EFR32 using the Silicon Labs Flex SDK (Software Development Kit). The Flex SDK provides two paths to application development. The first uses Silicon Labs Connect, 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. The second begins with Silicon Labs RAIL (Radio Abstraction Interface Layer), which provides an intuitive, easily-customizable radio interface layer that is designed to support proprietary or standards-based wireless protocols.

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 using on the EFR32.

If you are working with the EZR32, see *QSG137: Getting Started with the Silicon Labs Flex Software Development Kit for the EZR32*. To see a table showing the Connect features supported on the EZR32 and the EFR32, see **Appendix: EFR32/EZR32 Comparison**. The EZR32 does not support RAIL development.

KEY FEATURES

- Product overview
- Setting up your development environment
- Discovering your SDK
- Working with example applications.
- Using the Hardware Configurator
- EFR32/EZR32 Flex Feature Comparison

1 Flex SDK Product Overview

The Silicon Labs 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 but develop custom lower-level radio and network protocols.

This section covers:

- Background information on RAIL and Connect, and the example applications included with the Flex SDK
- 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 uses Silicon Labs Connect, 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 plugins that can be turned on or off as desired. The second begins with Silicon Labs RAIL (Radio Abstraction Interface Layer), which provides an intuitive, easily-customizable radio interface layer that is designed to support proprietary or standards-based wireless protocols.

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)
- Host and NCP mode

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 description on the Simplicity Studio IDE General tab provides additional detail about the example and interfacing with it. Note that the empty-flex example has no plugins selected and therefore will not compile without additional configuration.

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 the following Connect example applications:

Silicon Labs Connect Sensor Sample Application and **Silicon Labs Connect Sink Sample Application**: 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.

Silicon Labs Connect Sensor Sample Application (Host) and **Silicon Labs Connect Sink Sample Application (Host)**: Both applications run on a UNIX UART Host, with EFR32NCP.

Silicon Labs Connect Wire-Replacement Sample Application: Demonstrates point-to-point bi-directional direct or indirect communication between two nodes.

Silicon Labs Connect Wire-Replacement Sample Application (Host): Runs on Unix UART Host, with an EFR32 NCP.

Silicon Labs Connect Commissioned Direct Device Sample Application: Demonstrates direct communication between nodes in range, whereas network parameters are commissioned by the application.

Silicon Labs Connect Commissioned Direct Device Sample Application (Host): Runs on Unix UART Host, with an EFR32 NCP.

Silicon Labs Connect NCP UART SW (Software Flow Control): This network coprocessor (NCP) application supports communication with a host application over a UART interface with software flow control. It runs on an EFR32. For more information about XON/XOFF software flow control, see *AN844: Guide to Host/Network Co-Processor Communications Using Silicon Labs Thread and Connect*.

Silicon Labs Connect NCP UART HW (Hardware Flow Control): This network coprocessor (NCP) application supports communication with a host application over a UART interface with hardware flow control. It runs on an EFR32.

1.1.2 Silicon Labs RAIL

Silicon Labs RAIL provides an intuitive, easily-customizable 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.

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 are included in the current release.

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.

RAILtest has the following PHY:

- frequency: 2.4 GHz
- deviation: 500 k
- modulation: 2 gfsk
- bitrate: 1 Mbps

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 the Silicon Labs WSTK hardware without the need for commands from a host computer. This capability allows for mobility during range testing activities.

Silicon Labs RAIL Range Test Sample Application for EFR32: A customizable example that demonstrates the over-the-air range of the EFR32. The default PHY configuration for this example is:

- frequency: 2.4 GHz
- deviation, modulation and bitrate consistent with 802.15.4 specification

Developers can change this PHY configuration.

Silicon Labs RAIL Range Test Sample Application for EFR32 - PHY: 802.15.4: Demonstrates the over-the-air range of the EFR32 with the following PHY:

- frequency: 2.4GHz
- deviation, modulation and bitrate consistent with 802.15.4 specification

Silicon Labs RAIL Range Test Sample Application for EFR32 - PHY: 250k, 2gfsk, base: A customizable example that demonstrates the over the air range of the EFR32 with the following PHY:

- frequency: 2.4 GHz
- deviation: 125 k
- modulation: 2 gfsk
- bitrate: 250 Kbps

Other Example Applications

Silicon Labs RAIL Simple TRx Sample Application for EFR32: Demonstrates the simplest transmit and receive example application based on RAIL with the following PHY:

- frequency: 2.4 GHz
- deviation: 500 k
- modulation: 2 gfsk
- bitrate: 1 Mbps

Silicon Labs RAIL Bidirectional Sample Application for EFR32: Demonstrates the simplest transmit and ack example application based on RAIL with the following PHY:

- frequency: 2.4 GHz
- deviation: 500 k
- modulation: 2 gfsk
- bitrate: 1 Mbps

Silicon Labs RAIL Energy Mode Sample Application for EFR32: Demonstrates the low power modes of the EFR32 with the following PHY:

- frequency: 2.4 GHz
- deviation: 500 k
- modulation: 2 gfsk
- bitrate: 1 Mbps

Silicon Labs RAIL Duty Cycle Sample Application for EFR32: Includes three modes of operation (Duty Cycle (both nodes are in the same mode), Master, and Slave) and demonstrates low power applications using the EFR32 with the following PHY:

- frequency: 2.4GHz
- deviation: 500k
- modulation: 2gfsk
- bitrate: 38.4 kbps

1.2 Prerequisites

Before following the procedures in this guide you must have

- Purchased one of the Wireless Gecko (EFR32) Portfolio Mesh Networking Kits (As of this writing, all EFR32FG kits and EFR32MG 434/868/915 are supported), although some features may only function on a subset of these.
- Downloaded the required software components. A card included in your development hardware kit contains a link to a Getting Started page, which will direct you to links for the Silicon Labs software products. See the Flex SDK release notes for version restrictions and compatibility constraints for Connect and RAIL and these components. To develop Silicon Labs Connect- or RAIL-based applications, you will need the following:
  - The Simplicity Studio v4 development environment, which incorporates AppBuilder (see **Installing Simplicity Studio)**. AppBuilder is an interactive GUI tool that allows you to configure a body of Silicon Labs-supplied code to implement applications. Online help for AppBuilder and other Simplicity Studio modules is provided.
  - The Silicon Labs Flex SDK, installed through Simplicity Studio.
  - (optional) IAR Embedded Workbench for ARM (IAR EWARM) 7.80.2. 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 of IAR from the Silicon Labs support portal. Refer to the “QuickStart Installation Information” section of the IAR installer for additional information about the installation process and how to configure your license.

Although you will not need it for the tasks in this Getting Started guide, you may wish to become familiar with the functionality available through Simplicity Commander. Simplicity Commander is installed along with Simplicity Studio in a subfolder of the install folder. See *UG162: Simplicity Commander Reference Guide* for more information.

1.3 Support

Users can access the Silicon Labs support portal at https://www.silabs.com/support or through the **Resources** tab in the Simplicity Studio Launcher perspective. Use the support portal to contact Customer Support for any questions you might have during the development process.

1.4 Documentation

Documentation provided with the Flex SDK can be accessed through Simplicity Studio. Simplicity Studio also provides links to hardware documentation and other appnotes, as described in section **Accessing Other Resources**. See the release notes for details on other documentation available.

2 Setting Up Your Development Environment

2.1 Install Third-Party Tools

Install third-party tools, such as IAR Embedded Workbench for ARM (see section **Software Components** for the list).

2.2 Connect your Hardware

Connect your WSTK, with radio board mounted, to the PC on which you will install Simplicity Studio. By having it connected when Simplicity Studio installs, Simplicity Studio will automatically obtain the relevant additional resources it needs.

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

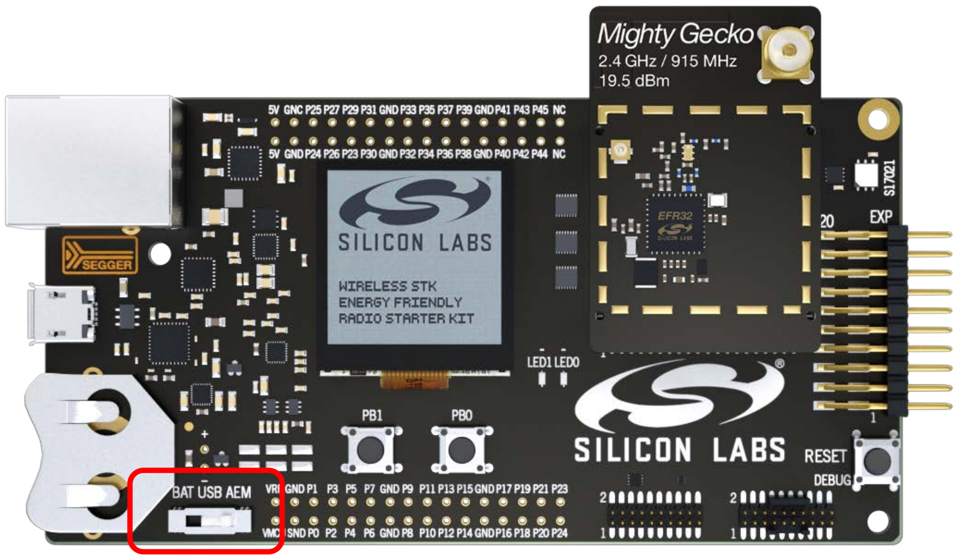
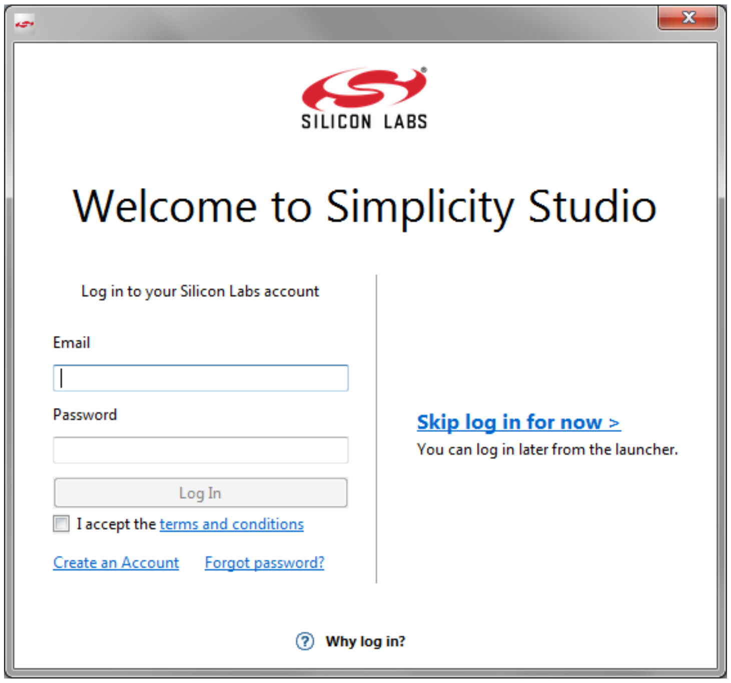


Figure 1. EFR32MG on a WSTK

2.3 Install Simplicity Studio and the Flex SDK

1. Run the Simplicity Studio installation application.
2. When Simplicity Studio first launches, you are invited to log in. If you have already created a user account on the support portal, the login username and password is the same, or you can create an account here.

Note: Although some Silicon Labs stacks require that you have a valid Silicon Labs user account before they can be installed, the Flex SDK does not.


Welcome to Simplicity Studio

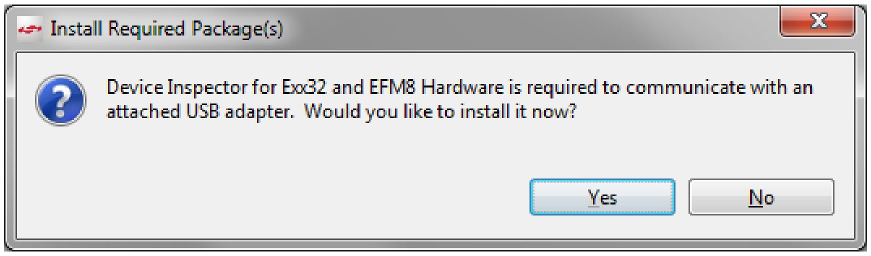
Log in to your Silicon Labs account

Email

Password

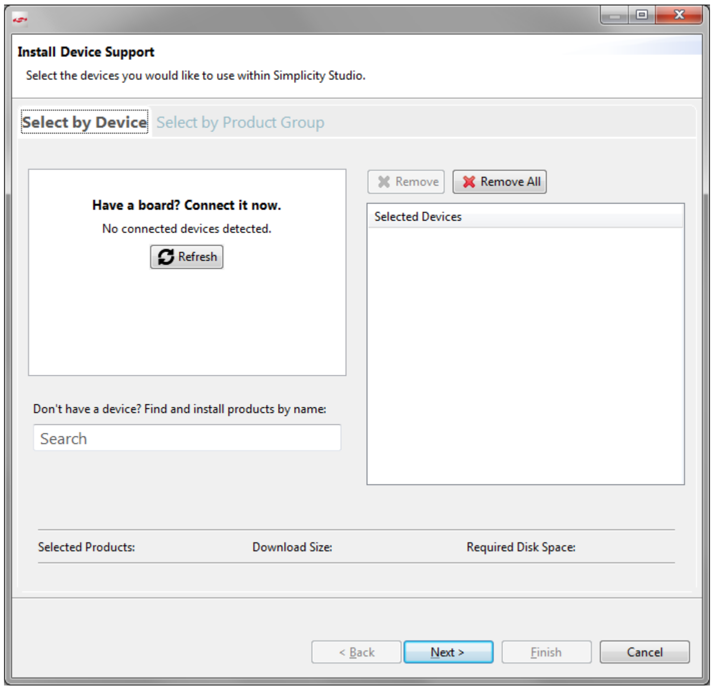
You can log in later from the launcher.

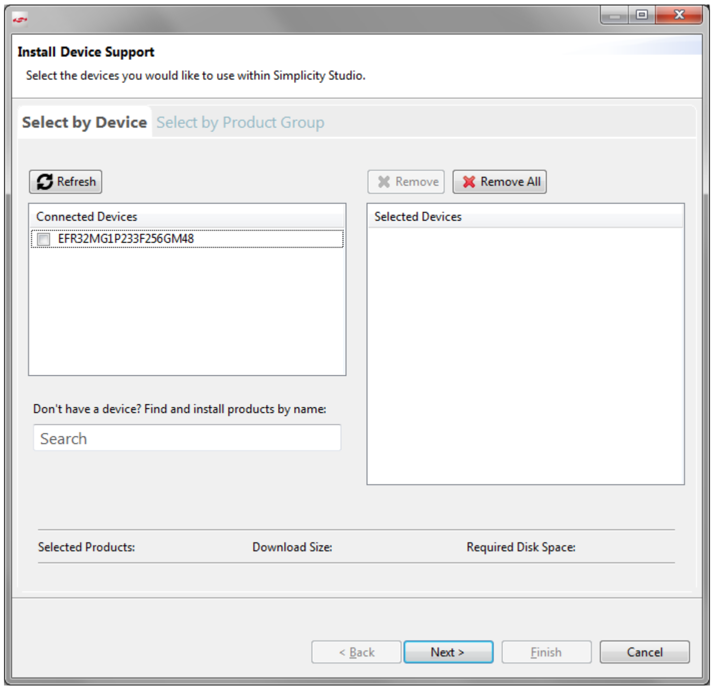
3. After login, Simplicity Studio adds software information. Once initial software installation is complete, Simplicity Studio checks for connected hardware. If you have the WSTK connected by USB cable, Simplicity Studio will detect the USB cable and prompt you to download a Device Inspector. Click **Yes**.


Install Required Package(s)

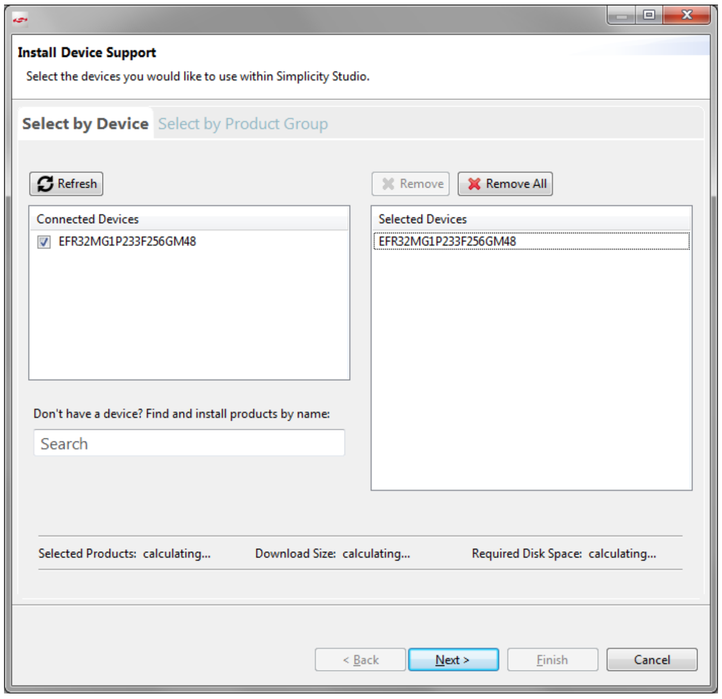
Device Inspector for Exx32 and EFM8 Hardware is required to communicate with an attached USB adapter. Would you like to install it now?

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



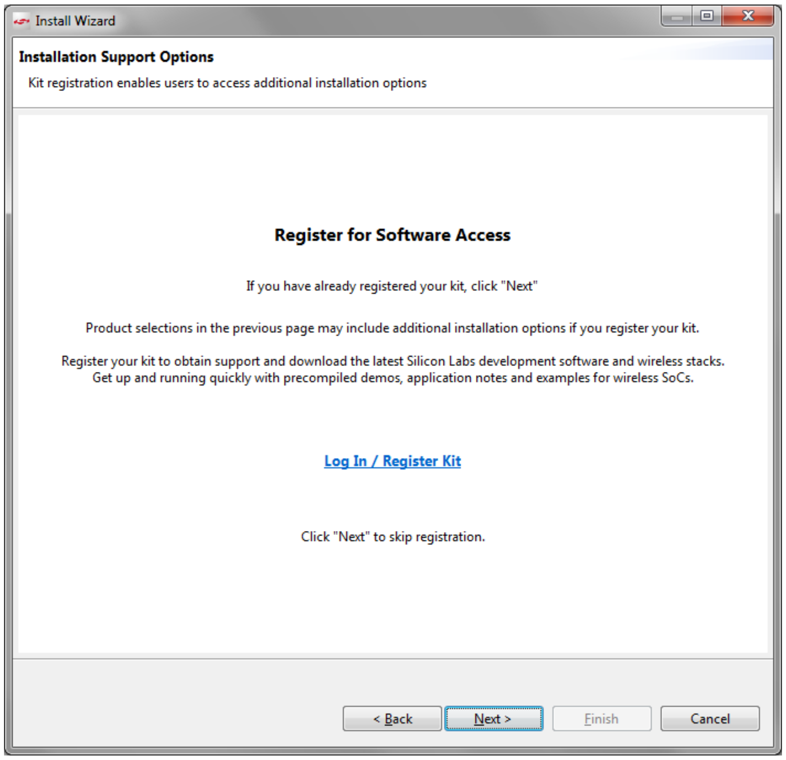


5. Click the checkbox next to the device to select it. Selecting the device allows Simplicity Studio to present the relevant software packages for you to install. Click **Next**.


Install Device Support
Select the devices you would like to use within Simplicity Studio.
Select by Device
Select by Product Group
Refresh
Remove
Remove All
Connected Devices
EFR32MG1P233F256GM48
Selected Devices
EFR32MG1P233F256GM48
Don't have a device? Find and install products by name:
Search
Selected Products: calculating...
Download Size: calculating...
Required Disk Space: calculating...
< Back
Next >
Finish
Cancel

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

6. A kit registration dialog is displayed. Unless you are also installing a stack, such as Silicon Labs, that requires registration, click **Next**.



Install Wizard

Installation Support Options

Kit registration enables users to access additional installation options

Register for Software Access

If you have already registered your kit, click "Next"

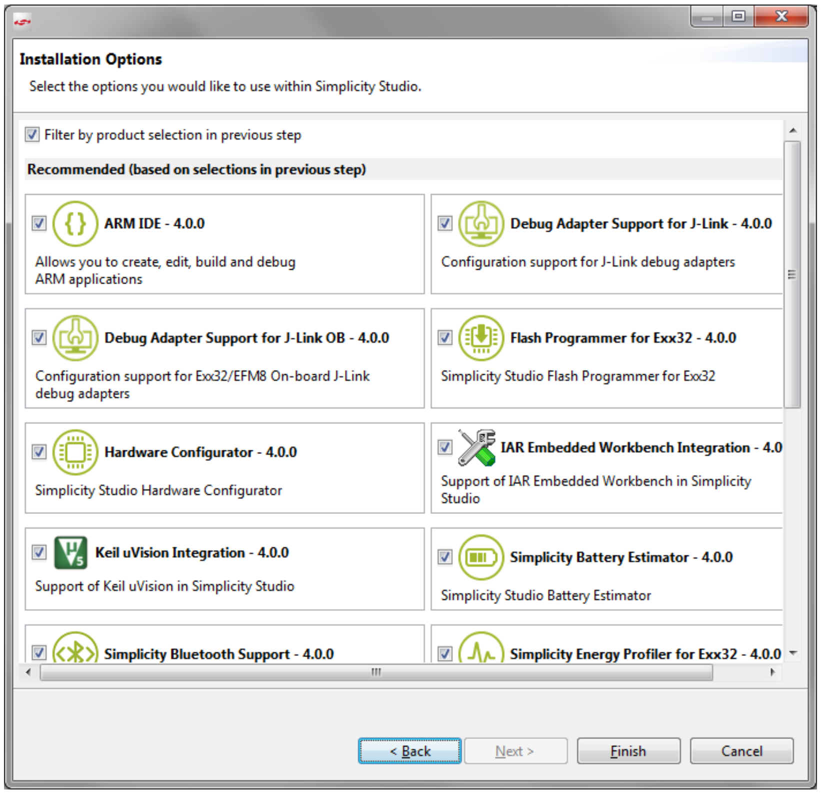
Product selections in the previous page may include additional installation options if you register your kit.

Register your kit to obtain support and download the latest Silicon Labs development software and wireless stacks.
Get up and running quickly with precompiled demos, application notes and examples for wireless SoCs.

Log In / Register Kit

Click "Next" to skip registration.

7. The **Installation Options** dialog shows the software packages that can be installed. By default the list is filtered by the product selection. To see all tools, uncheck the **Filter by product selection in previous step** box. In general, you will install all the selected tools. Also by default, the current versions of all Silicon Labs stacks that are compatible with this part are checked. Uncheck any you do not wish to install. Previous stack versions are shown under **Other Options**. Click **Finish**.



Installation Options

Select the options you would like to use within Simplicity Studio.

☑ Filter by product selection in previous step

Recommended (based on selections in previous step)

☑ **ARM IDE - 4.0.0**
Allows you to create, edit, build and debug ARM applications

☑ **Debug Adapter Support for J-Link - 4.0.0**
Configuration support for J-Link debug adapters

☑ **Debug Adapter Support for J-Link OB - 4.0.0**
Configuration support for Exx32/EFM8 On-board J-Link debug adapters

☑ **Flash Programmer for Exx32 - 4.0.0**
Simplicity Studio Flash Programmer for Exx32

☑ **Hardware Configurator - 4.0.0**
Simplicity Studio Hardware Configurator

☑ **IAR Embedded Workbench Integration - 4.0**
Support of IAR Embedded Workbench in Simplicity Studio

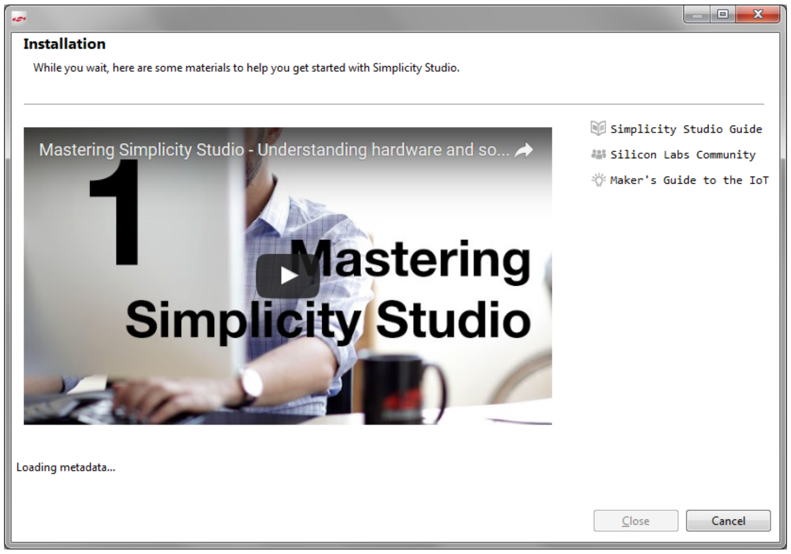
☑ **Keil uVision Integration - 4.0.0**
Support of Keil uVision in Simplicity Studio

☑ **Simplicity Battery Estimator - 4.0.0**
Simplicity Studio Battery Estimator

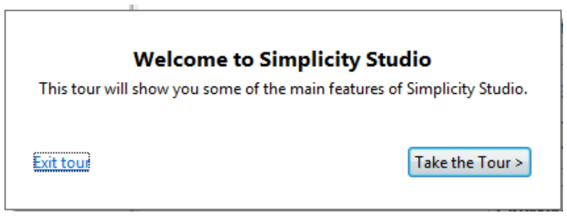
☑ **Simplicity Bluetooth Support - 4.0.0**

☑ **Simplicity Energy Profiler for Exx32 - 4.0.0**

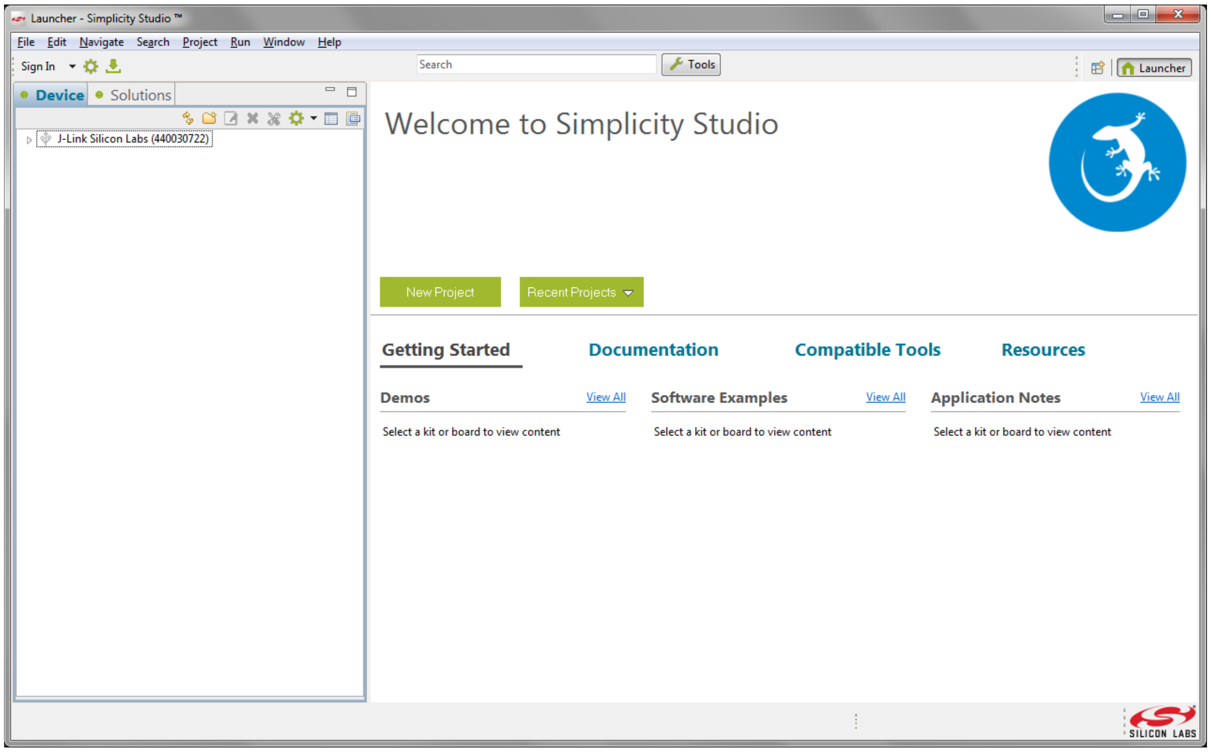
Installation will take several minutes. During installation, Simplicity Studio offers you viewing and reading options to learn more about the environment.


Installation
While you wait, here are some materials to help you get started with Simplicity Studio.
Mastering Simplicity Studio - Understanding hardware and so...
1 Mastering Simplicity Studio
Simplicity Studio Guide
Silicon Labs Community
Maker's Guide to the IoT
Loading metadata...
Close
Cancel

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


Welcome to Simplicity Studio
This tour will show you some of the main features of Simplicity Studio.
Exit tour
Take the Tour >

10. The Launcher perspective opens, but it is not yet fully populated. Expand the J-Link entry in the devices tab and click on the WSTK.


Launcher - Simplicity Studio ™
File Edit Navigate Search Project Run Window Help
Sign In
Search
Tools
Launcher
Device
Solutions
J-Link Silicon Labs (440030722)
Welcome to Simplicity Studio
New Project
Recent Projects
Getting Started
Documentation
Compatible Tools
Resources
Demos
View All
Select a kit or board to view content
Software Examples
View All
Select a kit or board to view content
Application Notes
View All
Select a kit or board to view content
SILICON LABS

The Launcher perspective is then populated with the software components and functionality associated with your WSTK and stack, as described in the next section.

Before continuing with the example application, you may need to update your device firmware or change your preferred WSTK. These functions are described in sections **Updating Adapter Firmware** and **Changing the Preferred SDK,** respectively.

Note: If you plan to work with NCP software communicating with a host MCU/PC, you should upgrade your adapter firmware to build 435 or later.

3 Functionality in the Launcher Perspective

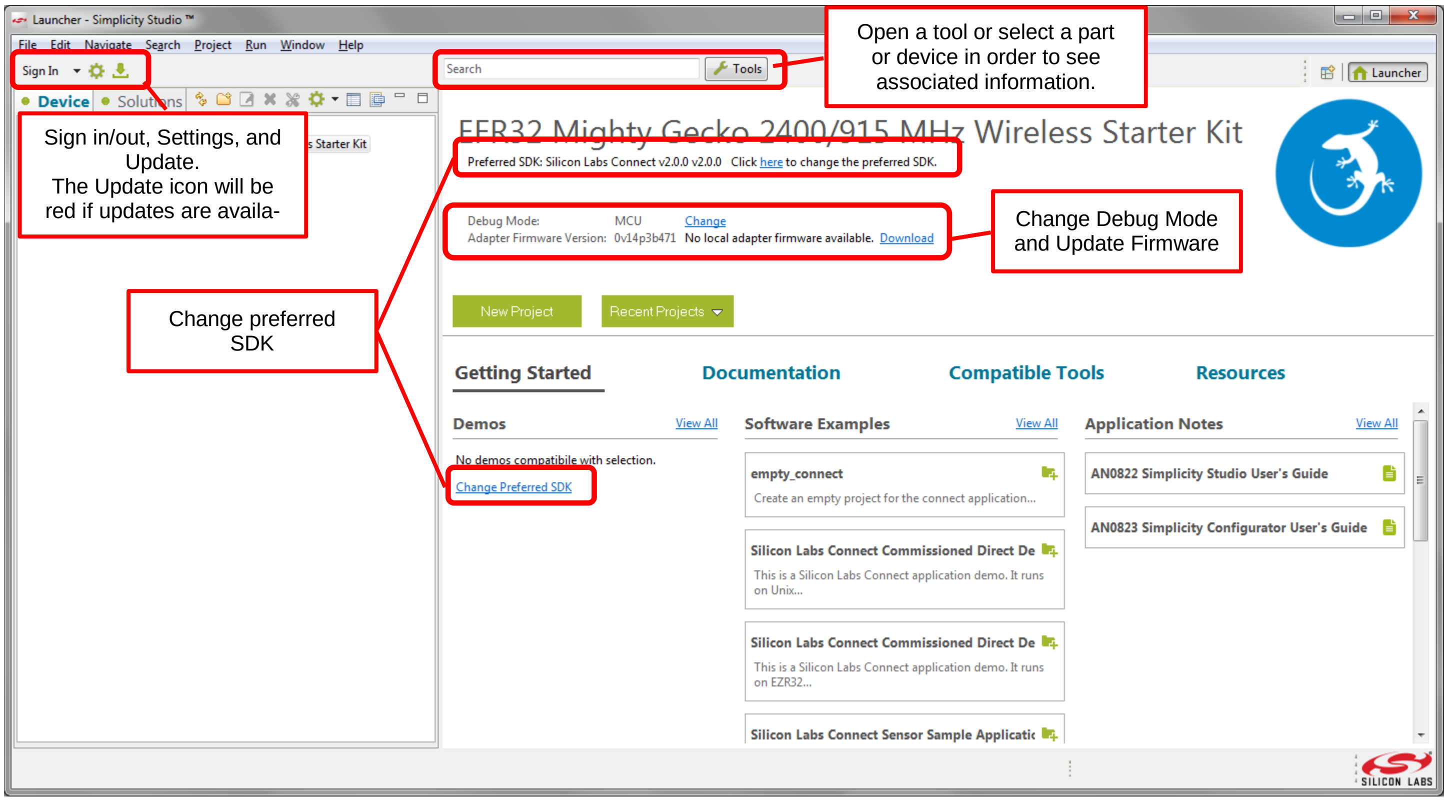
In the Launcher perspective you can perform a number of functions. Additional information on some of these is provided later in the section.

- Sign in (if you are not signed in already)
- Open application settings
- Update your software and firmware (see below for more information)
- Open tools or search

  As you open tools or perspectives, buttons for them are displayed in the upper right. You can use those buttons to easily navigate back to the Launcher perspective or to other perspectives.
- Change your preferred SDK (see below for more information)
- Change debug mode
- Update adapter firmware (see below for more information)
- Access demos, examples, documentation, and other resources (see below for more information)

Note: Perspectives are made up of a number of tiles or panes, called views, as well as the content in those views. You can change the layouts of various perspectives by expanding or relocating views, or adding or removing views. If you want to return to the default layout, right-click the perspective button in the upper right and select **Default**.

The Devices view on the left of the Launcher perspective also has a solutions tab. If you developing for complex networks with a number of different parts involved, you can add them all to the solution and then select the one you are working on from the list.



Open a tool or select a part or device in order to see associated information.

Sign in/out, Settings, and Update. The Update icon will be red if updates are availa-

Change preferred SDK

Change Debug Mode and Update Firmware

EFR32 Mighty Gecko 2400/915 MHz Wireless Starter Kit

Preferred SDK: Silicon Labs Connect v2.0.0 v2.0.0 Click here to change the preferred SDK.

Debug Mode: MCU Change
Adapter Firmware Version: 0v14p3b471 No local adapter firmware available. Download

Demos

No demos compatibile with selection.
Change Preferred SDK

Software Examples

empty_connect
Create an empty project for the connect application...

Silicon Labs Connect Commissioned Direct De
This is a Silicon Labs Connect application demo. It runs on Unix...

Silicon Labs Connect Commissioned Direct De
This is a Silicon Labs Connect application demo. It runs on EZR32...

Silicon Labs Connect Sensor Sample Applicatic

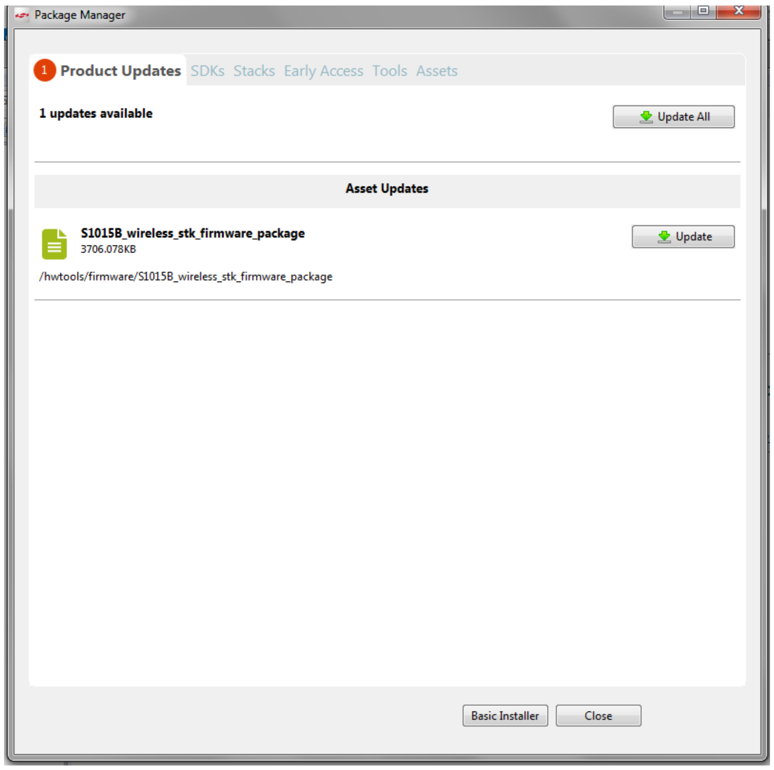
Application Notes

AN0822 Simplicity Studio User's Guide
AN0823 Simplicity Configurator User's Guide

3.1 Downloading Updates

The Download Update icon will be red if updates are available. If Simplicity Studio detects an available update, and you are in another perspective that does not display the update icon, you will be notified that an update is available.

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



1 updates available

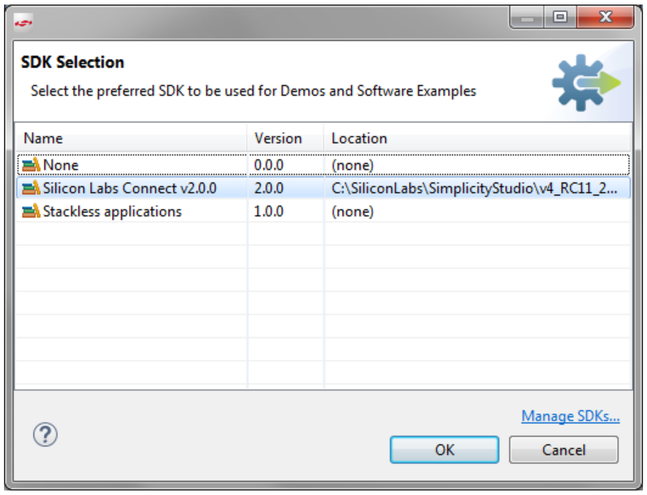
Asset Updates

S1015B_wireless_stk_firmware_package
3706.078KB
/hwtools/firmware/S1015B_wireless_stk_firmware_package

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

3.2 Changing the Preferred SDK

If you have more than one stack or SDK installed, you can change the preferred (or currently active) SDK by clicking one of the links. The SDK selection interface is displayed. Click an SDK, and then click **OK**.

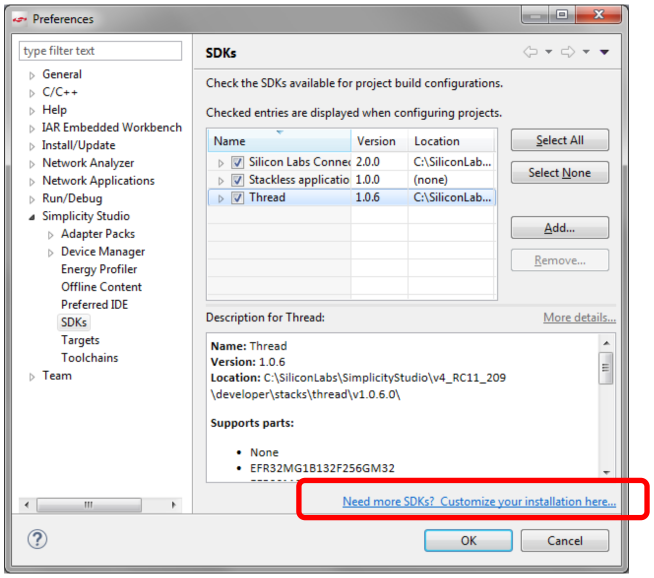


SDK Selection

Select the preferred SDK to be used for Demos and Software Examples

Name	Version	Location
None	0.0.0	(none)
Silicon Labs Connect v2.0.0	2.0.0	C:\SiliconLabs\SimplicityStudio\v4_RC11_2...
Stackless applications	1.0.0	(none)

If the SDK you want to use is not displayed, in the SDK Selection dialog click **Manage SDKs**. In the Preferences dialog, click **Need more SDKs …**



SDKs

Check the SDKs available for project build configurations.

Checked entries are displayed when configuring projects.

Name	Version	Location
☑ Silicon Labs Connec	2.0.0	C:\SiliconLab...
☑ Stackless applicatio	1.0.0	(none)
☑ Thread	1.0.6	C:\SiliconLab...

Description for Thread:

Name: Thread
Version: 1.0.6
Location: C:\SiliconLabs\SimplicityStudio\v4_RC11_209 \developer\stacks\thread\v1.0.6.0\

Supports parts:

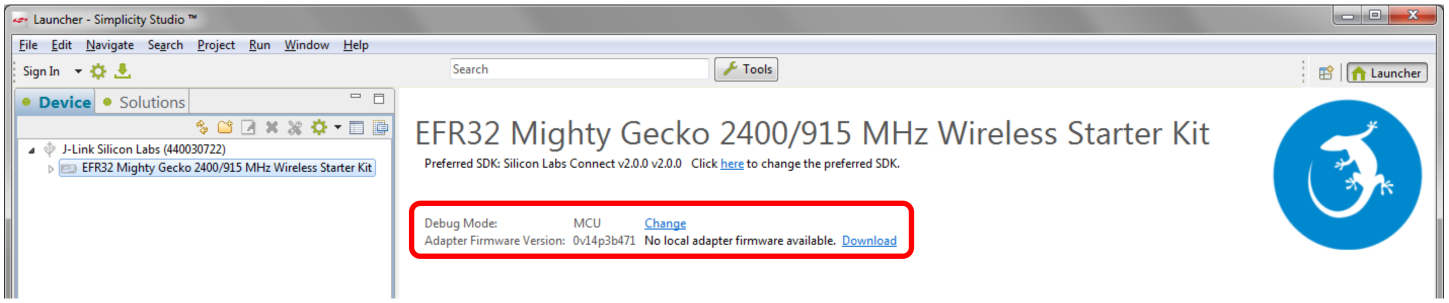
- None
- EFR32MG1B132F256GM32

This updates your system and opens the Package Manager interface, the same as if you clicked the Download Updates icon. Click on either the SDK or the Stack tabs to see and install a new item. Then return to the SDK Selection dialog and select the recently installed SDK.

Note: If you are changing to an SDK you have installed outside of Simplicity Studio, first add it, which takes you back to the Launcher perspective. Then click one of the links to change the preferred SDK, select the SDK, and click **OK**.

3.3 Updating Adapter Firmware

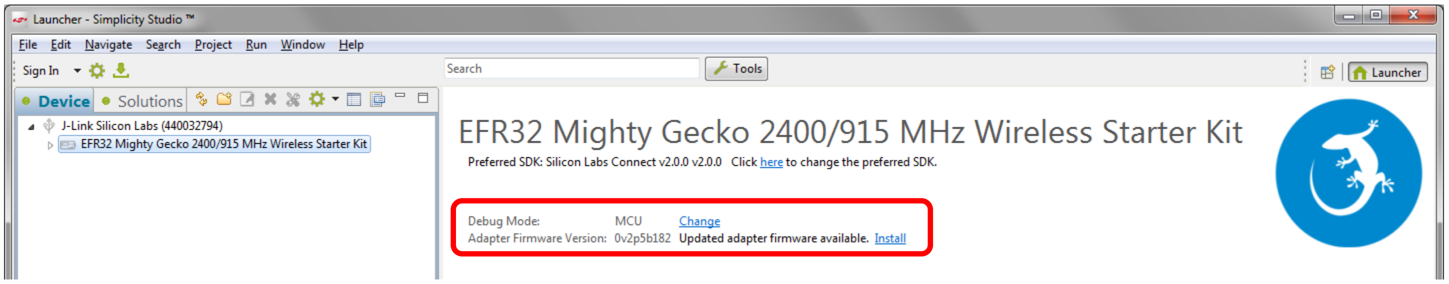
Initially the Launcher perspective may display “No local adapter firmware available.” Click **Download** to download any updates.


Launcher - Simplicity Studio ™
File Edit Navigate Search Project Run Window Help
Sign In
Search
Tools
Launcher
Device
Solutions
J-Link Silicon Labs (440030722)
EFR32 Mighty Gecko 2400/915 MHz Wireless Starter Kit
EFR32 Mighty Gecko 2400/915 MHz Wireless Starter Kit

Preferred SDK: Silicon Labs Connect v2.0.0 v2.0.0 Click here to change the preferred SDK.

Debug Mode: MCU Change
Adapter Firmware Version: 0v14p3b471 No local adapter firmware available. Download

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

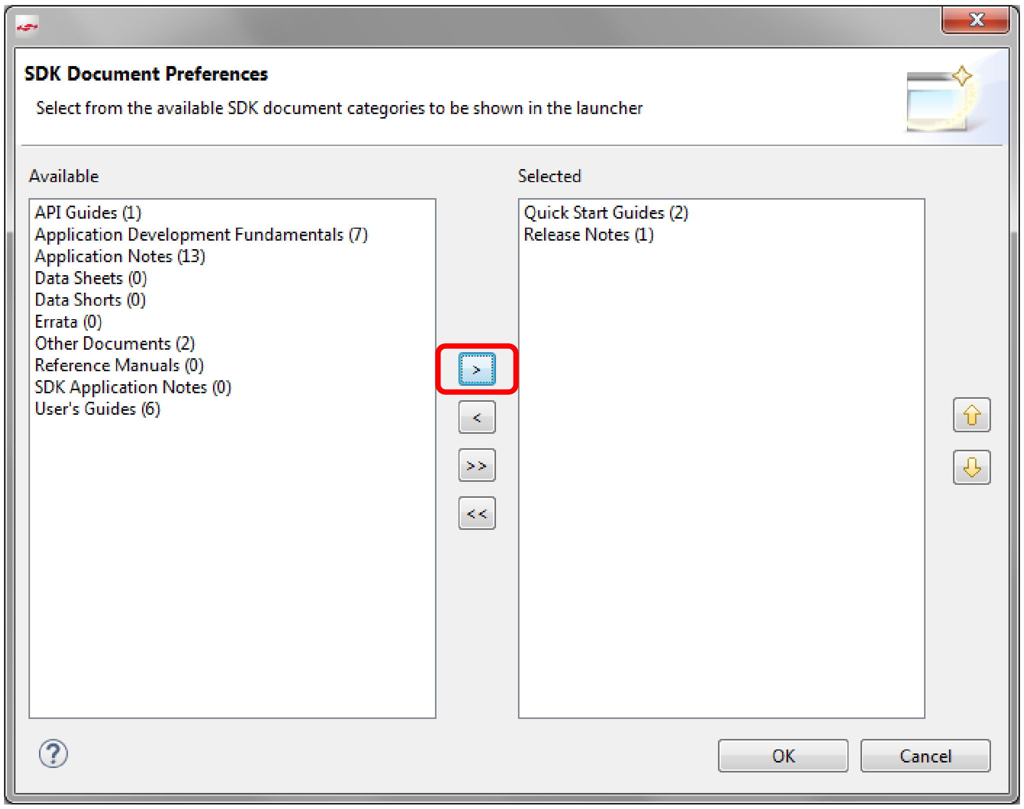

Launcher - Simplicity Studio ™
File Edit Navigate Search Project Run Window Help
Sign In
Search
Tools
Launcher
Device
Solutions
J-Link Silicon Labs (440032794)
EFR32 Mighty Gecko 2400/915 MHz Wireless Starter Kit
EFR32 Mighty Gecko 2400/915 MHz Wireless Starter Kit

Preferred SDK: Silicon Labs Connect v2.0.0 v2.0.0 Click here to change the preferred SDK.

Debug Mode: MCU Change
Adapter Firmware Version: 0v2p5b182 Updated adapter firmware available. Install

3.4 Accessing Documentation and Other Resources

The **Getting Started** tab provides access to demos, example applications, and application notes and stack documentation. To see stack documentation, drop down the list, and select **SDK Documents**. In the SDK Document Preferences dialog, select categories on the left and click the arrow to move them to the right. Only selected categories are displayed in the Launcher perspective. Click **OK**.


SDK Document Preferences

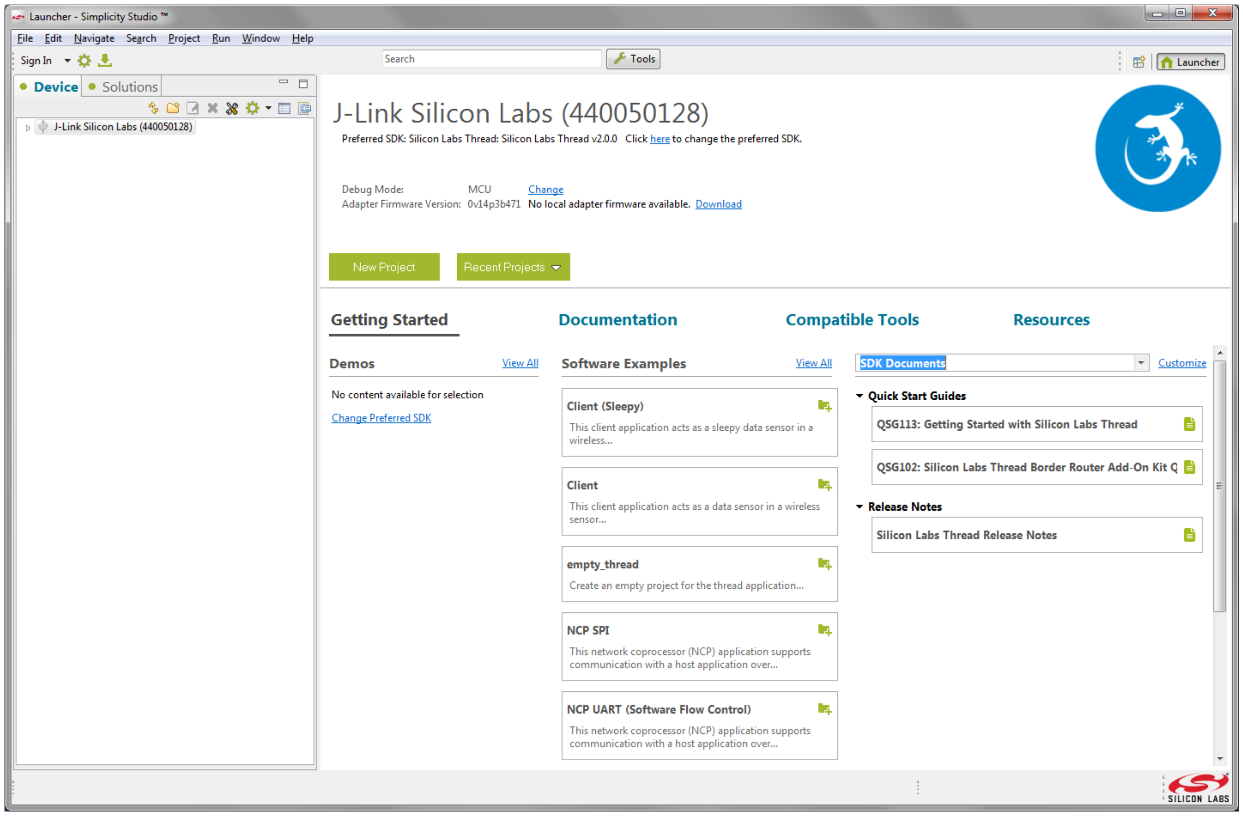
Select from the available SDK document categories to be shown in the launcher

Available

- API Guides (1)
- Application Development Fundamentals (7)
- Application Notes (13)
- Data Sheets (0)
- Data Shorts (0)
- Errata (0)
- Other Documents (2)
- Reference Manuals (0)
- SDK Application Notes (0)
- User's Guides (6)

Selected

- Quick Start Guides (2)
- Release Notes (1)



J-Link Silicon Labs (440050128)

J-Link Silicon Labs (440050128)

Preferred SDK: Silicon Labs Thread: Silicon Labs Thread v2.0.0 Click here to change the preferred SDK.

Debug Mode: MCU Change
Adapter Firmware Version: 0v14p3b471 No local adapter firmware available. Download

Demos

No content available for selection

Change Preferred SDK

Software Examples

Client (Sleepy)
This client application acts as a sleepy data sensor in a wireless...

Client
This client application acts as a data sensor in a wireless sensor...

empty_thread
Create an empty project for the thread application...

NCP SPI
This network coprocessor (NCP) application supports communication with a host application over...

NCP UART (Software Flow Control)
This network coprocessor (NCP) application supports communication with a host application over...

SDK Documents

Quick Start Guides
- QSG113: Getting Started with Silicon Labs Thread
- QSG102: Silicon Labs Thread Border Router Add-On Kit Q

Release Notes
- Silicon Labs Thread Release Notes

The **Documentation** tab lists schematics, specifications, datasheets and other documentation about the kit hardware.

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

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

4 Working with Example Applications

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

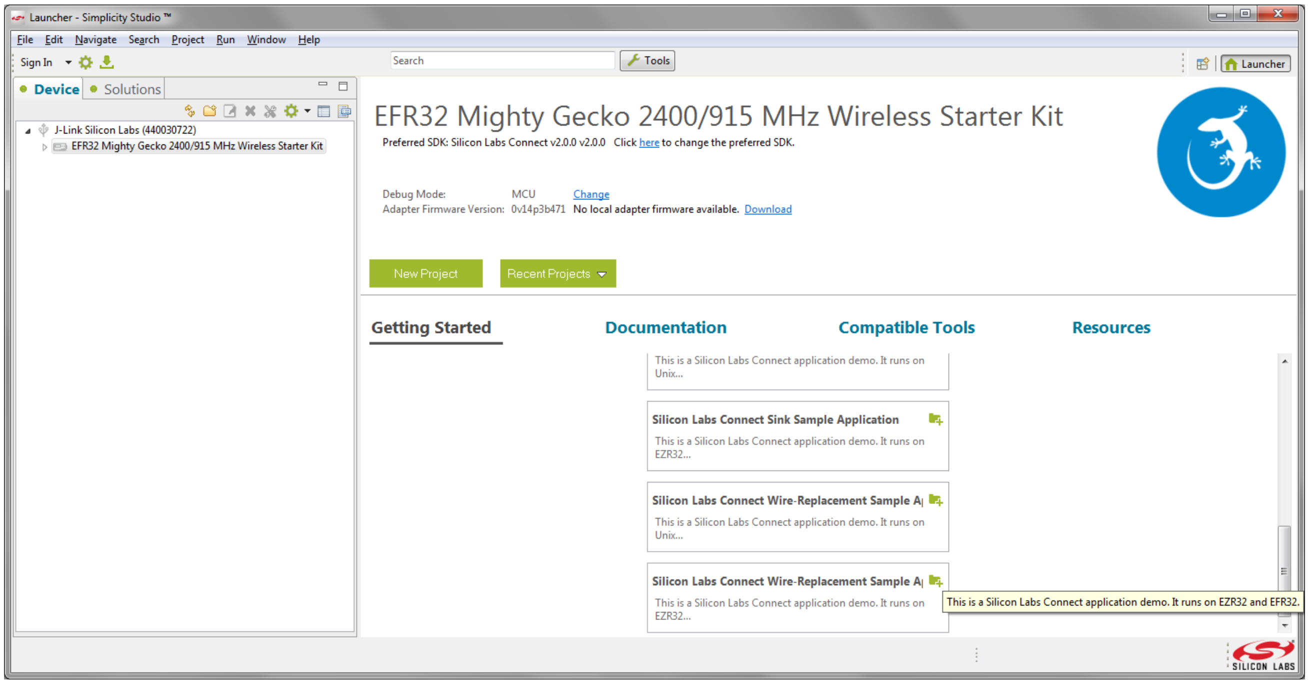
1. Select an example application.
2. Generate application files.
3. Compile and flash the application to the radio board.

These steps are described in detail in the following sections. If you are going to be working with Connect, follow the procedures in section **Working with Connect**. If you are going to be working with RAIL, follow the procedures in section **Working with RAIL Examples**.

4.1 Working with Connect

4.1.1 Selecting a Connect Example Application

1. In the Launcher perspective, click on aan example application, in this case the Wire Replacement Sample for the EZR32 and EFR32.



Launcher - Simplicity Studio ™
File Edit Navigate Search Project Run Window Help
Sign In
Search
Tools
Launcher
Device
Solutions
J-Link Silicon Labs (440030722)
EFR32 Mighty Gecko 2400/915 MHz Wireless Starter Kit
New Project
Recent Projects
Getting Started
Documentation
Compatible Tools
Resources
SILICON LABS

EFR32 Mighty Gecko 2400/915 MHz Wireless Starter Kit

Preferred SDK: Silicon Labs Connect v2.0.0 v2.0.0 Click here to change the preferred SDK.

Debug Mode: MCU Change
Adapter Firmware Version: 0v14p3b471 No local adapter firmware available. Download

This is a Silicon Labs Connect application demo. It runs on Unix...

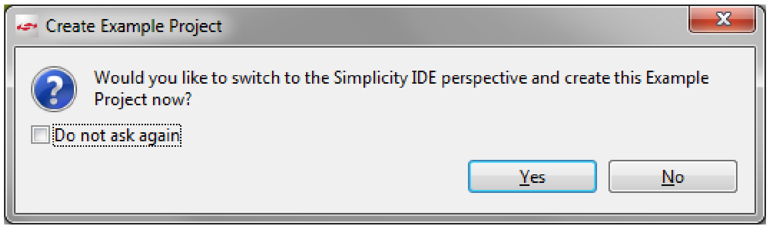
Silicon Labs Connect Sink Sample Application
This is a Silicon Labs Connect application demo. It runs on EZR32...

Silicon Labs Connect Wire-Replacement Sample A
This is a Silicon Labs Connect application demo. It runs on Unix...

Silicon Labs Connect Wire-Replacement Sample A
This is a Silicon Labs Connect application demo. It runs on EZR32...

This is a Silicon Labs Connect application demo. It runs on EZR32 and EFR32.

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

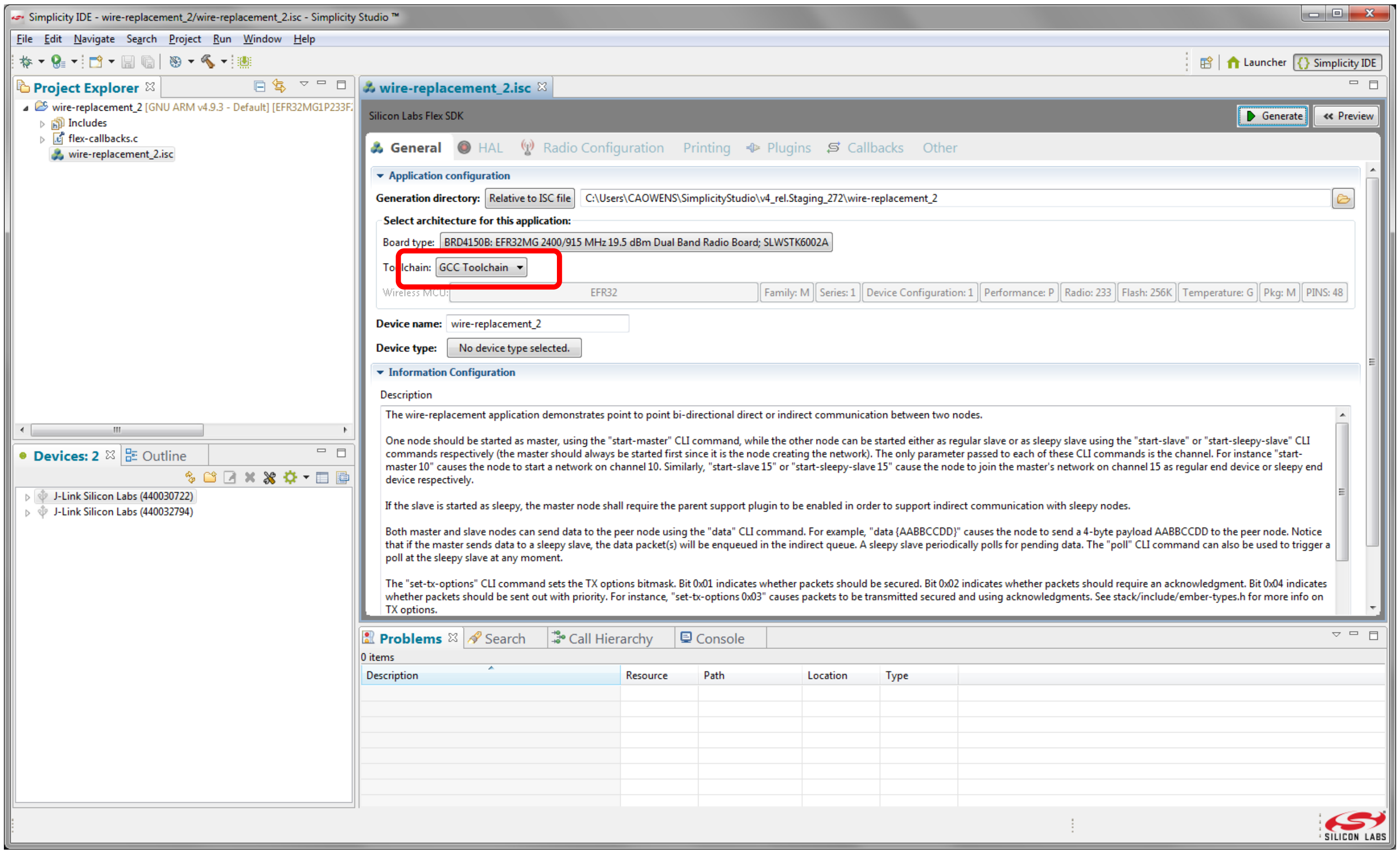


Create Example Project

Would you like to switch to the Simplicity IDE perspective and create this Example Project now?

☐ Do not ask again

3. Simplicity IDE opens with the new project in AppBuilder view. The default toolchain is GCC. If you are using IAR, change the Toolchain value in the drop down.


Simplicity IDE - wire-replacement_2/wire-replacement_2.isc - Simplicity Studio ™
File Edit Navigate Search Project Run Window Help
Launcher
Simplicity IDE
Project Explorer
wire-replacement_2 [GNU ARM v4.9.3 - Default] [EFR32MG1P233F
Includes
flex-callbacks.c
wire-replacement_2.isc
Devices: 2
Outline
J-Link Silicon Labs (440030722)
J-Link Silicon Labs (440032794)
wire-replacement_2.isc
Silicon Labs Flex SDK
Generate
Preview
General
HAL
Radio Configuration
Printing
Plugins
Callbacks
Other
Application configuration
Information Configuration
Problems
Search
Call Hierarchy
Console
0 items
SILICON LABS

Generation directory: Relative to ISC file C:\Users\CAOWENS\SimplicityStudio\v4_rel.Staging_272\wire-replacement_2

Select architecture for this application:

Board type: BRD4150B: EFR32MG 2400/915 MHz 19.5 dBm Dual Band Radio Board; SLWSTK6002A

Toolchain: GCC Toolchain

Wireless MCU: EFR32 | Family: M | Series: 1 | Device Configuration: 1 | Performance: P | Radio: 233 | Flash: 256K | Temperature: G | Pkg: M | PINS: 48

Device name: wire-replacement_2

Device type: No device type selected.

Description

The wire-replacement application demonstrates point to point bi-directional direct or indirect communication between two nodes.

One node should be started as master, using the "start-master" CLI command, while the other node can be started either as regular slave or as sleepy slave using the "start-slave" or "start-sleepy-slave" CLI commands respectively (the master should always be started first since it is the node creating the network). The only parameter passed to each of these CLI commands is the channel. For instance "start-master 10" causes the node to start a network on channel 10. Similarly, "start-slave 15" or "start-sleepy-slave 15" cause the node to join the master's network on channel 15 as regular end device or sleepy end device respectively.

If the slave is started as sleepy, the master node shall require the parent support plugin to be enabled in order to support indirect communication with sleepy nodes.

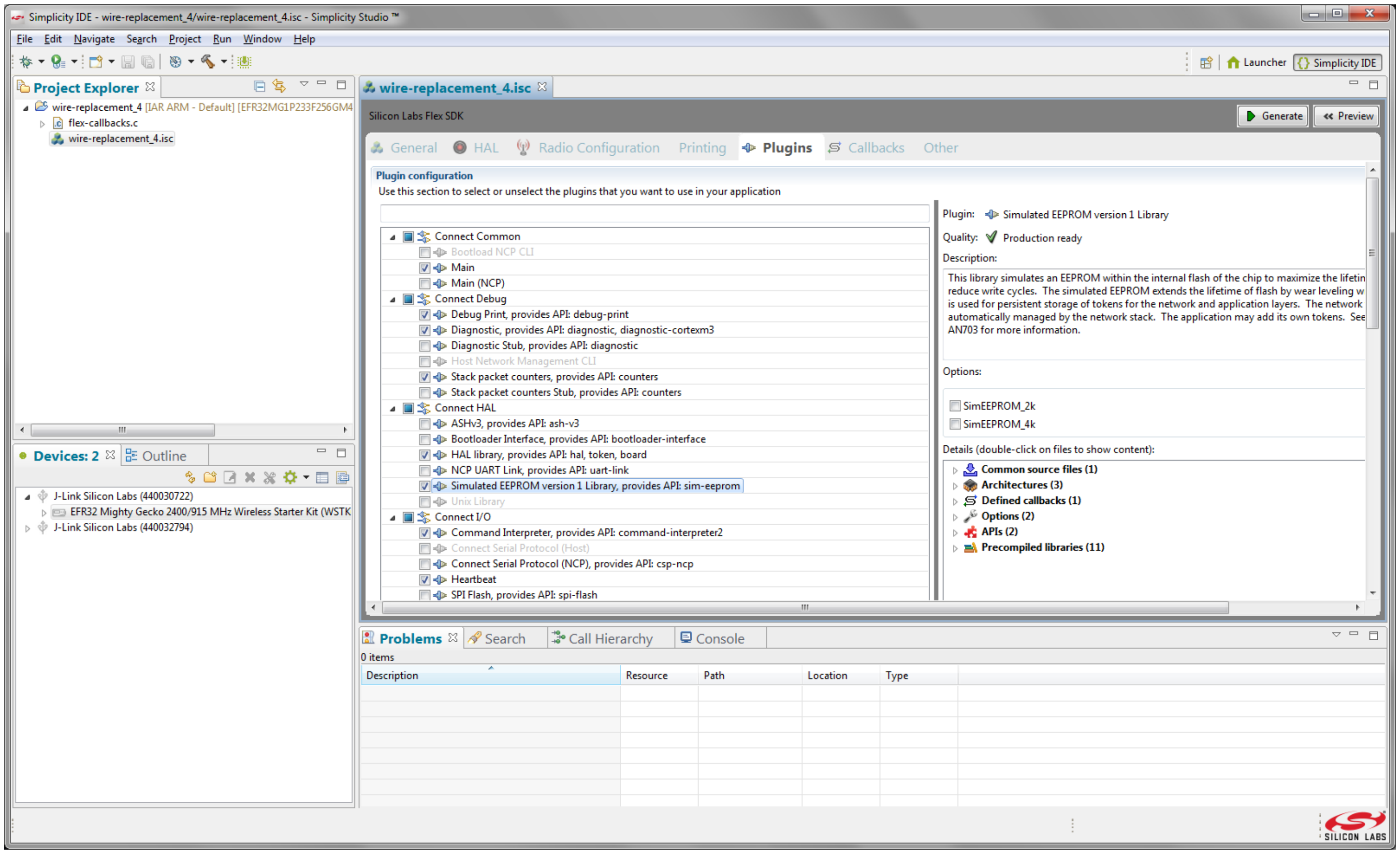
Both master and slave nodes can send data to the peer node using the "data" CLI command. For example, "data {AABBCCDD}" causes the node to send a 4-byte payload AABBCCDD to the peer node. Notice that if the master sends data to a sleepy slave, the data packet(s) will be enqueued in the indirect queue. A sleepy slave periodically polls for pending data. The "poll" CLI command can also be used to trigger a poll at the sleepy slave at any moment.

The "set-tx-options" CLI command sets the TX options bitmask. Bit 0x01 indicates whether packets should be secured. Bit 0x02 indicates whether packets should require an acknowledgment. Bit 0x04 indicates whether packets should be sent out with priority. For instance, "set-tx-options 0x03" causes packets to be transmitted secured and using acknowledgments. See stack/include/ember-types.h for more info on TX options.

Description	Resource	Path	Location	Type

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

The Plugins tab displays the various plugins that are preselected to enable the example's functionality. Click on a plugin to see its configurable options (if any), along with more information about the plugin.


Simplicity IDE - wire-replacement_4/wire-replacement_4.isc - Simplicity Studio ™
File Edit Navigate Search Project Run Window Help
Launcher
Simplicity IDE
Project Explorer
wire-replacement_4 [IAR ARM - Default] [EFR32MG1P233F256GM4
flex-callbacks.c
wire-replacement_4.isc
Devices: 2
Outline
J-Link Silicon Labs (440030722)
EFR32 Mighty Gecko 2400/915 MHz Wireless Starter Kit (WSTK
J-Link Silicon Labs (440032794)
wire-replacement_4.isc
Silicon Labs Flex SDK
Generate
Preview
General
HAL
Radio Configuration
Printing
Plugins
Callbacks
Other
Problems
Search
Call Hierarchy
Console

Plugin configuration
Use this section to select or unselect the plugins that you want to use in your application

- Connect Common
  - [ ] Bootload NCP CLI
  - [✓] Main
  - [ ] Main (NCP)
- Connect Debug
  - [✓] Debug Print, provides API: debug-print
  - [✓] Diagnostic, provides API: diagnostic, diagnostic-cortexm3
  - [ ] Diagnostic Stub, provides API: diagnostic
  - [ ] Host Network Management CLI
  - [✓] Stack packet counters, provides API: counters
  - [ ] Stack packet counters Stub, provides API: counters
- Connect HAL
  - [ ] ASHv3, provides API: ash-v3
  - [ ] Bootloader Interface, provides API: bootloader-interface
  - [✓] HAL library, provides API: hal, token, board
  - [ ] NCP UART Link, provides API: uart-link
  - [✓] Simulated EEPROM version 1 Library, provides API: sim-eeprom
  - [ ] Unix Library
- Connect I/O
  - [✓] Command Interpreter, provides API: command-interpreter2
  - [ ] Connect Serial Protocol (Host)
  - [ ] Connect Serial Protocol (NCP), provides API: csp-ncp
  - [✓] Heartbeat
  - [ ] SPI Flash, provides API: spi-flash

Plugin: Simulated EEPROM version 1 Library
Quality: ✓ Production ready
Description:
This library simulates an EEPROM within the internal flash of the chip to maximize the lifetin reduce write cycles. The simulated EEPROM extends the lifetime of flash by wear leveling w is used for persistent storage of tokens for the network and application layers. The network automatically managed by the network stack. The application may add its own tokens. See AN703 for more information.

Options:
- [ ] SimEEPROM_2k
- [ ] SimEEPROM_4k

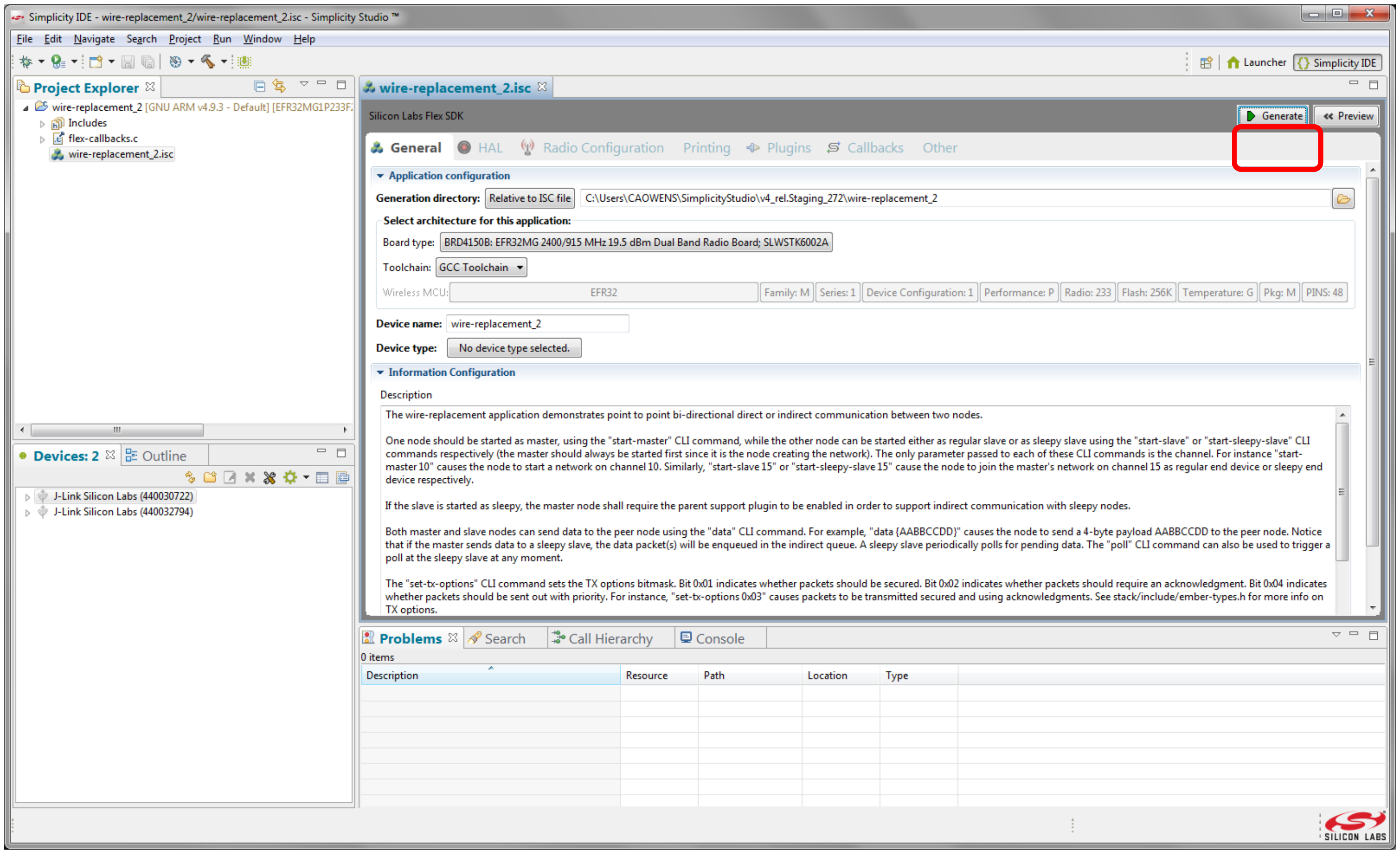
Details (double-click on files to show content):
- Common source files (1)
- Architectures (3)
- Defined callbacks (1)
- Options (2)
- APIs (2)
- Precompiled libraries (11)

0 items

Description	Resource	Path	Location	Type

4.1.2 Generating Connect Application Files

1. In the General tab, the Information Configuration block provides details about application setup and functionality (you may need to scroll down to see it). When you are ready to generate the sample project code, click **Generate.**


Simplicity IDE - wire-replacement_2/wire-replacement_2.isc - Simplicity Studio ™

Silicon Labs Flex SDK

Application configuration

Generation directory: Relative to ISC file C:\Users\CAOWENS\SimplicityStudio\v4_rel.Staging_272\wire-replacement_2

Select architecture for this application:

Board type: BRD4150B: EFR32MG 2400/915 MHz 19.5 dBm Dual Band Radio Board; SLWSTK6002A

Toolchain: GCC Toolchain

Wireless MCU: EFR32 Family: M Series: 1 Device Configuration: 1 Performance: P Radio: 233 Flash: 256K Temperature: G Pkg: M PINS: 48

Device name: wire-replacement_2

Device type: No device type selected.

Information Configuration

Description

The wire-replacement application demonstrates point to point bi-directional direct or indirect communication between two nodes.

One node should be started as master, using the "start-master" CLI command, while the other node can be started either as regular slave or as sleepy slave using the "start-slave" or "start-sleepy-slave" CLI commands respectively (the master should always be started first since it is the node creating the network). The only parameter passed to each of these CLI commands is the channel. For instance "start-master 10" causes the node to start a network on channel 10. Similarly, "start-slave 15" or "start-sleepy-slave 15" cause the node to join the master's network on channel 15 as regular end device or sleepy end device respectively.

If the slave is started as sleepy, the master node shall require the parent support plugin to be enabled in order to support indirect communication with sleepy nodes.

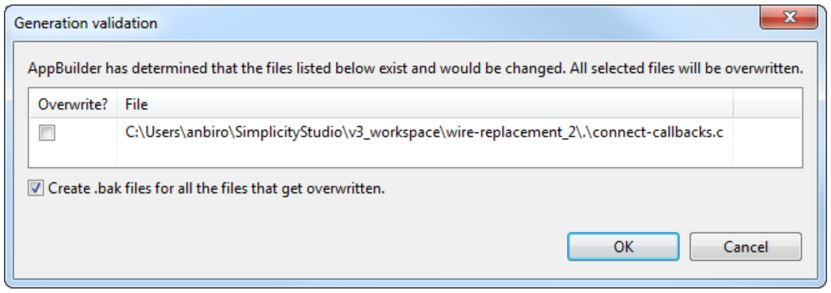
Both master and slave nodes can send data to the peer node using the "data" CLI command. For example, "data {AABBCCDD}" causes the node to send a 4-byte payload AABBCCDD to the peer node. Notice that if the master sends data to a sleepy slave, the data packet(s) will be enqueued in the indirect queue. A sleepy slave periodically polls for pending data. The "poll" CLI command can also be used to trigger a poll at the sleepy slave at any moment.

The "set-tx-options" CLI command sets the TX options bitmask. Bit 0x01 indicates whether packets should be secured. Bit 0x02 indicates whether packets should require an acknowledgment. Bit 0x04 indicates whether packets should be sent out with priority. For instance, "set-tx-options 0x03" causes packets to be transmitted secured and using acknowledgments. See stack/include/ember-types.h for more info on TX options.

Problems: 0 items

Description	Resource	Path	Location	Type

2. When asked about overwriting files, as long as you are working with examples do not check connect-callbacks.c. Only overwrite before starting your own coding, to create a file populated with stub callbacks for those you have indicated. Once you have made modifications to the callbacks in the file, be careful not to overwrite again.

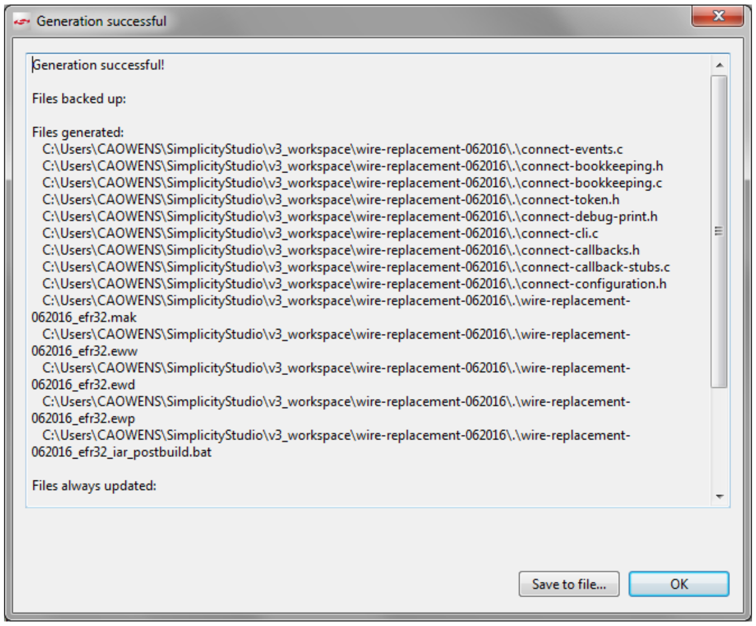

Generation validation

AppBuilder has determined that the files listed below exist and would be changed. All selected files will be overwritten.

Overwrite?	File
☐	C:\Users\anbiro\SimplicityStudio\v3_workspace\wire-replacement_2\.\connect-callbacks.c

☑ Create .bak files for all the files that get overwritten.

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



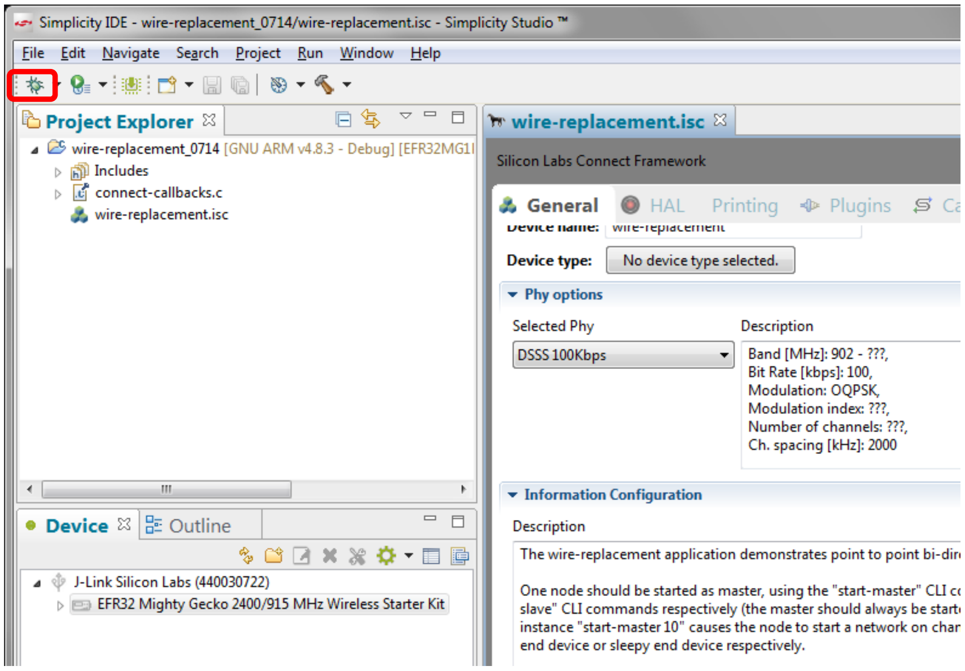
```
Generation successful!

Files backed up:

Files generated:
  C:\Users\CAOWENS\SimplicityStudio\v3_workspace\wire-replacement-062016\.\connect-events.c
  C:\Users\CAOWENS\SimplicityStudio\v3_workspace\wire-replacement-062016\.\connect-bookkeeping.h
  C:\Users\CAOWENS\SimplicityStudio\v3_workspace\wire-replacement-062016\.\connect-bookkeeping.c
  C:\Users\CAOWENS\SimplicityStudio\v3_workspace\wire-replacement-062016\.\connect-token.h
  C:\Users\CAOWENS\SimplicityStudio\v3_workspace\wire-replacement-062016\.\connect-debug-print.h
  C:\Users\CAOWENS\SimplicityStudio\v3_workspace\wire-replacement-062016\.\connect-cli.c
  C:\Users\CAOWENS\SimplicityStudio\v3_workspace\wire-replacement-062016\.\connect-callbacks.h
  C:\Users\CAOWENS\SimplicityStudio\v3_workspace\wire-replacement-062016\.\connect-callback-stubs.c
  C:\Users\CAOWENS\SimplicityStudio\v3_workspace\wire-replacement-062016\.\connect-configuration.h
  C:\Users\CAOWENS\SimplicityStudio\v3_workspace\wire-replacement-062016\.\wire-replacement-
062016_efr32.mak
  C:\Users\CAOWENS\SimplicityStudio\v3_workspace\wire-replacement-062016\.\wire-replacement-
062016_efr32.eww
  C:\Users\CAOWENS\SimplicityStudio\v3_workspace\wire-replacement-062016\.\wire-replacement-
062016_efr32.ewd
  C:\Users\CAOWENS\SimplicityStudio\v3_workspace\wire-replacement-062016\.\wire-replacement-
062016_efr32.ewp
  C:\Users\CAOWENS\SimplicityStudio\v3_workspace\wire-replacement-062016\.\wire-replacement-
062016_efr32_iar_postbuild.bat

Files always updated:
```

4.1.3 Compiling and Flashing the Connect Application

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

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.

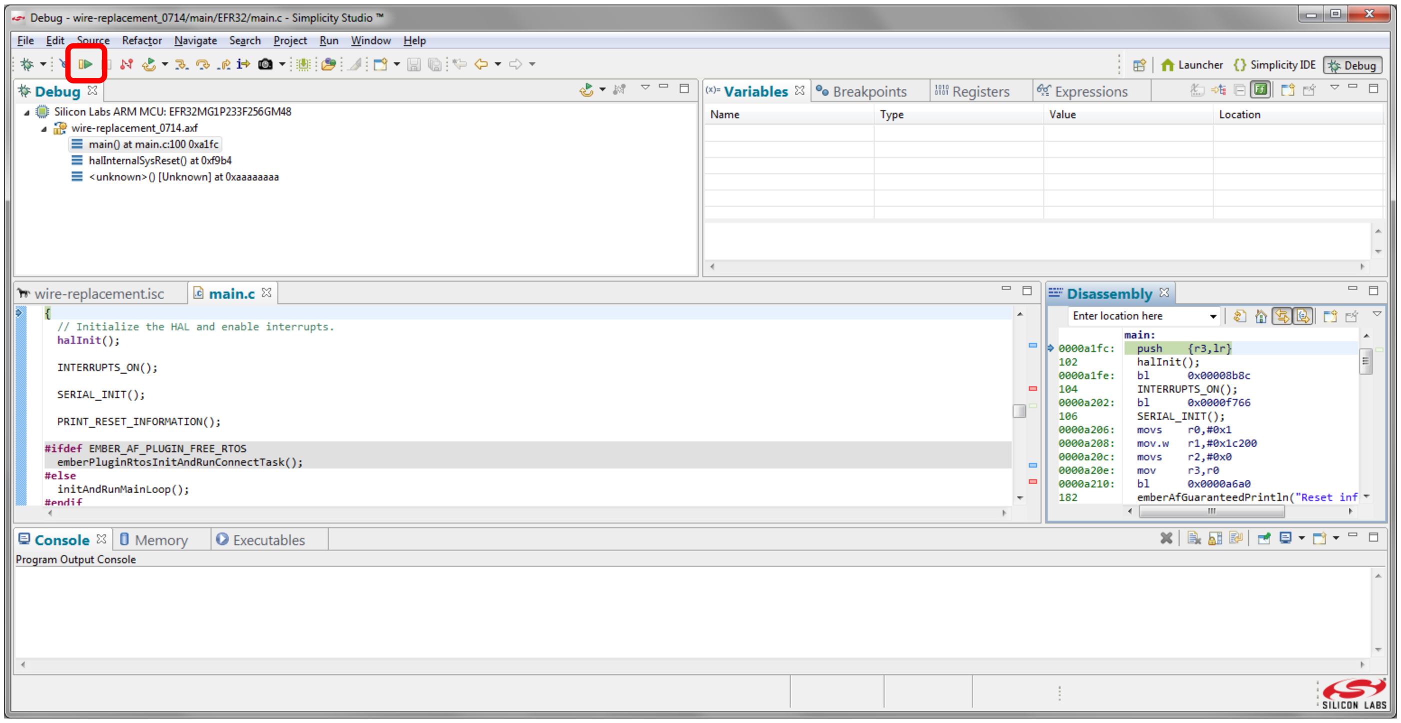

Simplicity IDE - wire-replacement_0714/wire-replacement.isc - Simplicity Studio ™
File Edit Navigate Search Project Run Window Help
Project Explorer
wire-replacement_0714 [GNU ARM v4.8.3 - Debug] [EFR32MG1
Includes
connect-callbacks.c
wire-replacement.isc
Device
Outline
J-Link Silicon Labs (440030722)
EFR32 Mighty Gecko 2400/915 MHz Wireless Starter Kit
wire-replacement.isc
Silicon Labs Connect Framework
General
HAL
Printing
Plugins

Device name: wire-replacement
Device type: No device type selected.

Phy options
Selected Phy: DSSS 100Kbps
Description: Band [MHz]: 902 - ???, Bit Rate [kbps]: 100, Modulation: OQPSK, Modulation index: ???, Number of channels: ???, Ch. spacing [kHz]: 2000

Information Configuration
Description
The wire-replacement application demonstrates point to point bi-dir
One node should be started as master, using the "start-master" CLI c
slave" CLI commands respectively (the master should always be start
instance "start-master 10" causes the node to start a network on char
end device or sleepy end device respectively.

2. Progress is displayed in the lower left. Wait until flashing has completed and a Debug perspective is displayed. Click the **Resume** control to start the application running on the WSTK.



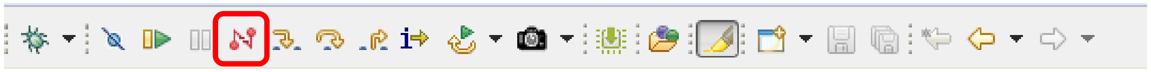
Debug
- Silicon Labs ARM MCU: EFR32MG1P233F256GM48
  - wire-replacement_0714.axf
    - main() at main.c:100 0xa1fc
    - halInternalSysReset() at 0xf9b4
    - <unknown>() [Unknown] at 0xaaaaaaaa

wire-replacement.isc | main.c

```
{
  // Initialize the HAL and enable interrupts.
  halInit();

  INTERRUPTS_ON();

  SERIAL_INIT();

  PRINT_RESET_INFORMATION();

#ifdef EMBER_AF_PLUGIN_FREE_RTOS
  emberPluginRtosInitAndRunConnectTask();
#else
  initAndRunMainLoop();
#endif
```

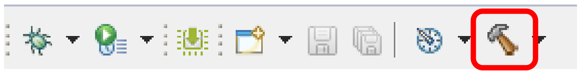
```
           main:
0000a1fc:    push    {r3,lr}
102          halInit();
0000a1fe:    bl      0x00008b8c
104          INTERRUPTS_ON();
0000a202:    bl      0x0000f766
106          SERIAL_INIT();
0000a206:    movs    r0,#0x1
0000a208:    mov.w   r1,#0x1c200
0000a20c:    movs    r2,#0x0
0000a20e:    mov     r3,r0
0000a210:    bl      0x0000a6a0
182          emberAfGuaranteedPrintln("Reset inf
```

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



Manually Compile and Flash

After you generate your project files, instead of clicking Debug in the Simplicity IDE, click the **Build** control in the top tool bar. Your sample application will compile based on its build configuration. You can change the build configuration at any time by right clicking on the project and going to **Build Configurations > Set Active**.

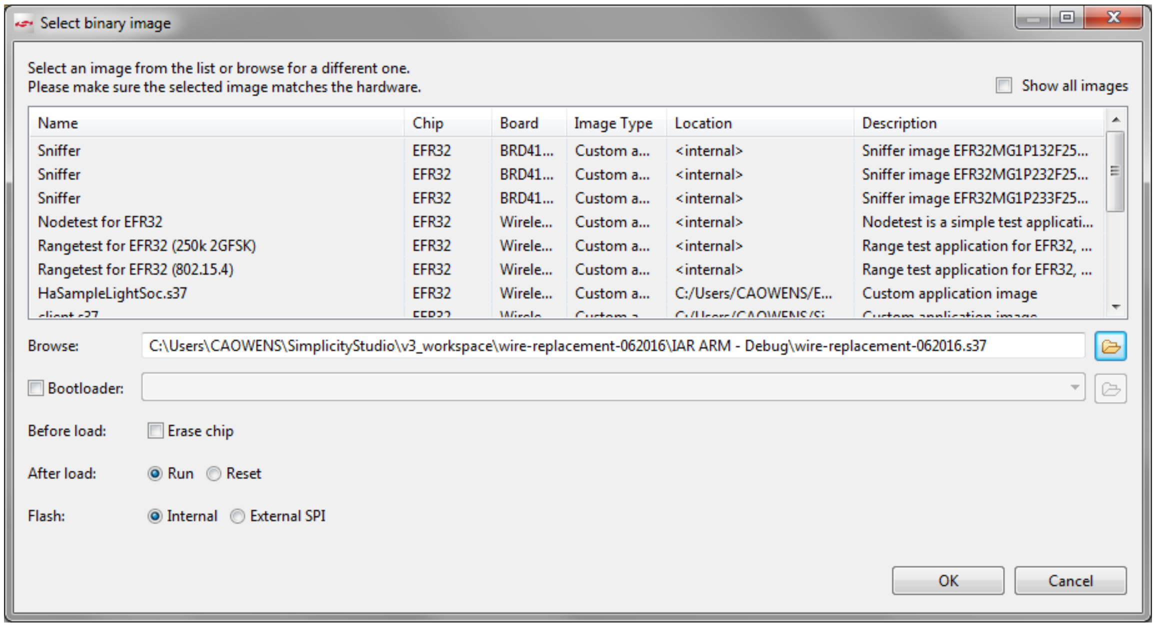


You can also build your application directly in IAR-EWARM by opening IAR-EWARM and opening the generated project file inside IAR.

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 the Devices view in the Simplicity IDE perspective.

1. In the Devices view, right-click on the J-Link device and select **Upload Application**. The Select Binary Image dialog is displayed.



Select binary image

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

Show all images

Name	Chip	Board	Image Type	Location	Description
Sniffer	EFR32	BRD41...	Custom a...	<internal>	Sniffer image EFR32MG1P132F25...
Sniffer	EFR32	BRD41...	Custom a...	<internal>	Sniffer image EFR32MG1P232F25...
Sniffer	EFR32	BRD41...	Custom a...	<internal>	Sniffer image EFR32MG1P233F25...
Nodetest for EFR32	EFR32	Wirele...	Custom a...	<internal>	Nodetest is a simple test applicati...
Rangetest for EFR32 (250k 2GFSK)	EFR32	Wirele...	Custom a...	<internal>	Range test application for EFR32, ...
Rangetest for EFR32 (802.15.4)	EFR32	Wirele...	Custom a...	<internal>	Range test application for EFR32, ...
HaSampleLightSoc.s37	EFR32	Wirele...	Custom a...	C:/Users/CAOWENS/E...	Custom application image

Browse: C:\Users\CAOWENS\SimplicityStudio\v3_workspace\wire-replacement-062016\IAR ARM - Debug\wire-replacement-062016.s37

Bootloader:

Before load: Erase chip

After load: [✓ Run] Reset

Flash: [✓ Internal] External SPI

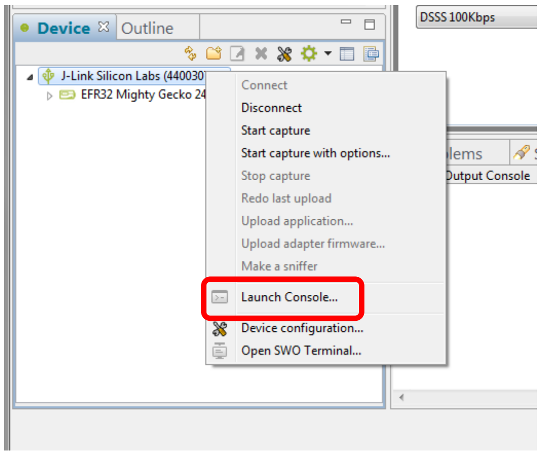
OK Cancel

2. Navigate to the .s37, .bin or .hex image you wish to upload.
3. Optionally, check **Erase Chip**, to make sure that any previous bootloader or other non-volatile data is erased before your new image is uploaded. New users will typically check this.
4. The **After Load** options are **Run** (begin executing the code immediately) and **Reset** (wait for an event, such as a debugger to connect or manual initiation of a boot sequence). During initial development you will typically leave this set to **Run**.
5. Click **OK.**

4.1.4 Interacting with Connect Examples

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. Choose **Connect** (if you are not already connected) and then **Launch Console**. Alternatively, from the Tools drop-down in the Launcher perspective select **Device Console**. To get a prompt on the Console Serial 1 tab, press Enter.



To create the network on the wire-replacement example (comments in parentheses), repeat the procedures in section **Selecting an Example Application** through **Compiling and Flashing the Application** to compile and load the Wire Replacement example on a second device.

Then use the following procedure.

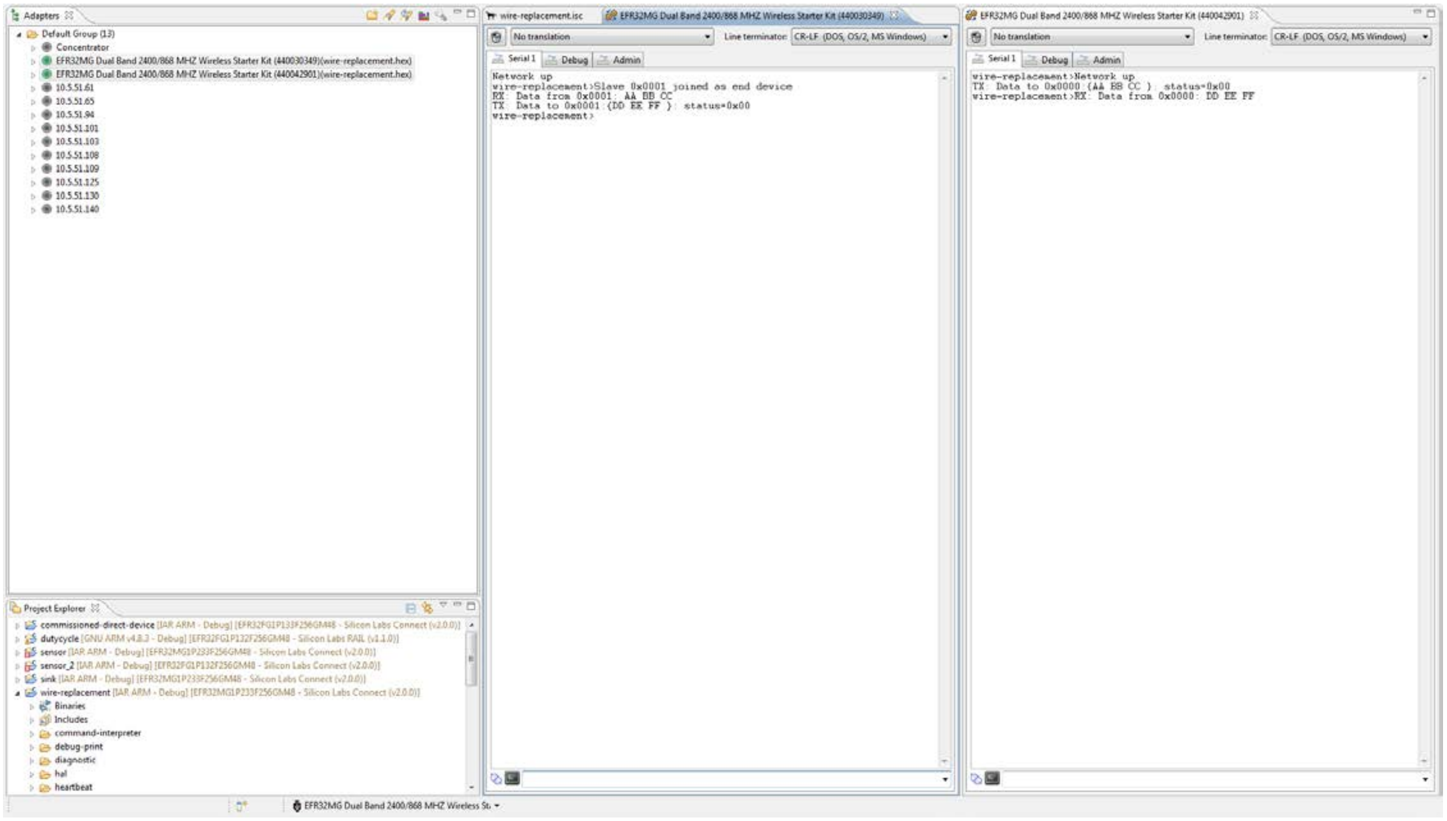
1. On Device A, enter `start-master 0` (where 0 is the channel number. You should see `Network Up`.
2. On Device B, enter `start-slave 0` (where 0 is the channel number). You should see:

   From Device B: `Network Up`
   From Device A: `Slave 0x0001 joined as end device from device A`
3. On device B, enter `data {AA BB CC}`

   From Device B: `TX: Data to 0x0000:{AA BB CC }: status=0x00`
   From Device A: `RX: Data from 0x0001: AA BB CC`
4. On Device A enter `data {DD EE FF}`

   From Device A: `TX: Data to 0x0001:{DD EE FF }: status=0x00`
   From Device B: `RX: Data from 0x0000: DD EE FF`

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



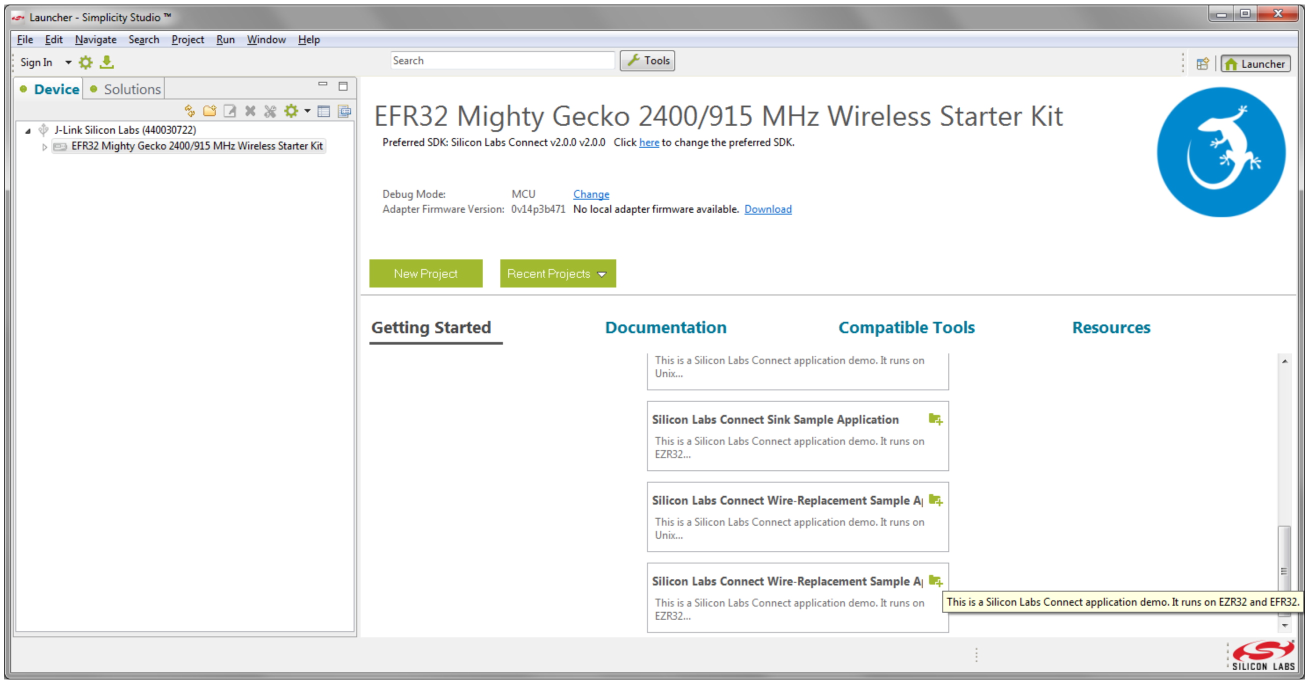
```
Network up
wire-replacement>Slave 0x0001 joined as end device
RX: Data from 0x0001: AA BB CC
TX: Data to 0x0001 {DD EE FF }: status=0x00
wire-replacement>
```

```
wire-replacement>Network up
TX: Data to 0x0000 {AA BB CC }: status=0x00
wire-replacement>RX: Data from 0x0000: DD EE FF
```

4.2 Working with RAIL

4.2.1 Selecting a RAIL Example Application

1. In the Launcher perspective, click on an example application. RF engineers may wish to start with RAILTest. Firmware engineers may wish to start with the simple_trx example. This section used the simple_trx example.


Launcher - Simplicity Studio ™
File Edit Navigate Search Project Run Window Help
Sign In
Search
Tools
Launcher
Device
Solutions
J-Link Silicon Labs (440030722)
EFR32 Mighty Gecko 2400/915 MHz Wireless Starter Kit
New Project
Recent Projects
Getting Started
Documentation
Compatible Tools
Resources
SILICON LABS

EFR32 Mighty Gecko 2400/915 MHz Wireless Starter Kit

Preferred SDK: Silicon Labs Connect v2.0.0 v2.0.0 Click here to change the preferred SDK.

Debug Mode: MCU Change

Adapter Firmware Version: 0v14p3b471 No local adapter firmware available. Download

This is a Silicon Labs Connect application demo. It runs on Unix...

Silicon Labs Connect Sink Sample Application

This is a Silicon Labs Connect application demo. It runs on EZR32...

Silicon Labs Connect Wire-Replacement Sample Ap

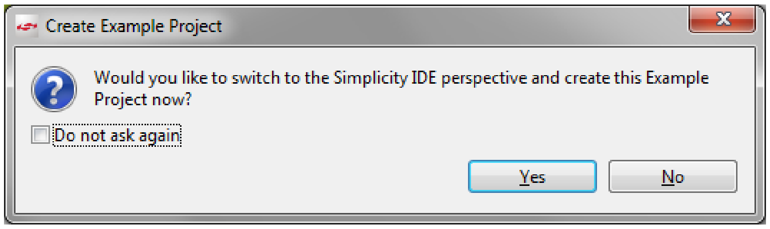
This is a Silicon Labs Connect application demo. It runs on Unix...

Silicon Labs Connect Wire-Replacement Sample Ap

This is a Silicon Labs Connect application demo. It runs on EZR32...

This is a Silicon Labs Connect application demo. It runs on EZR32 and EFR32.

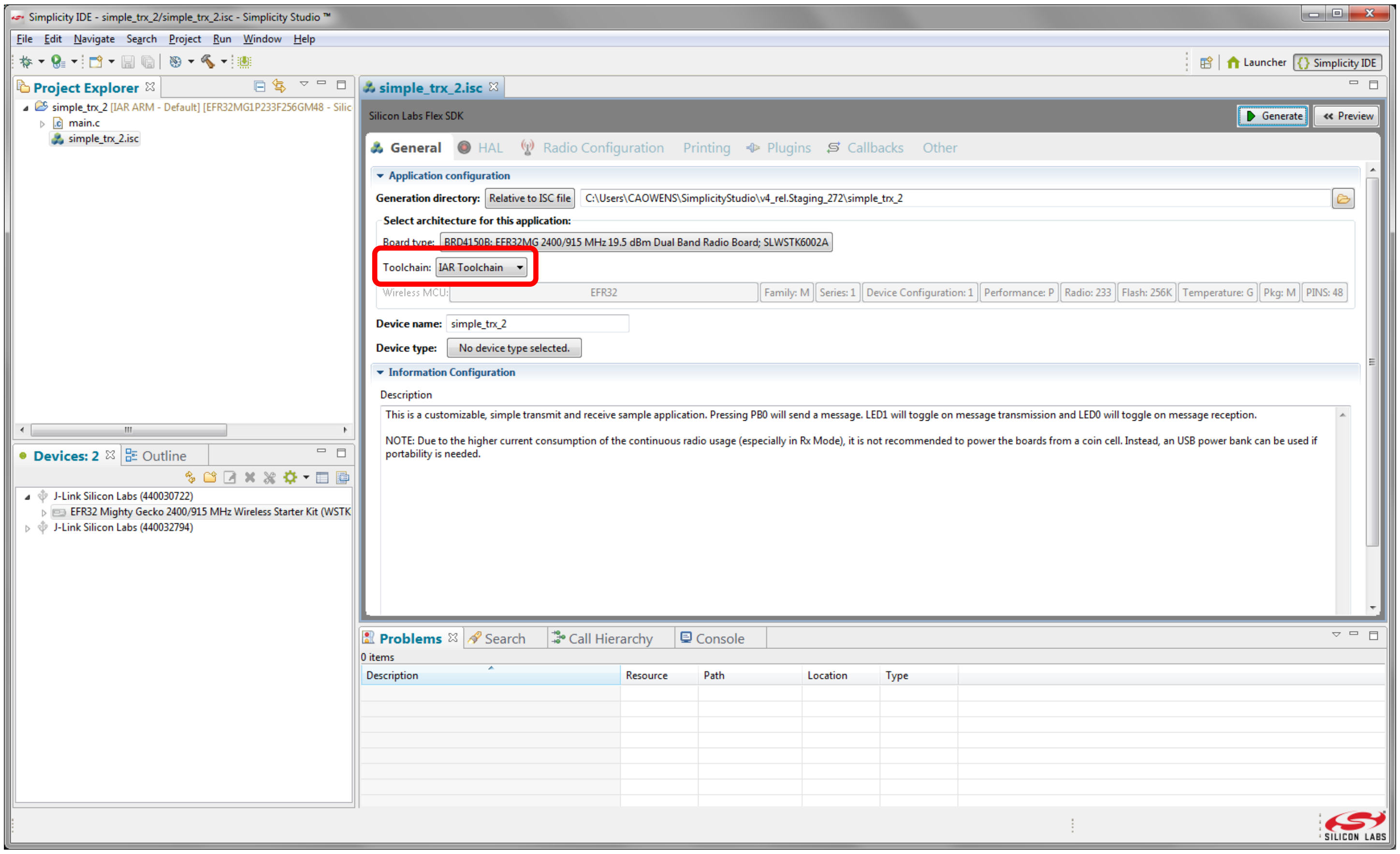
2. You are asked if you want to switch to the Simplicity IDE. Click **Yes**.



Would you like to switch to the Simplicity IDE perspective and create this Example Project now?

☐ Do not ask again

3. Simplicity IDE opens with the new project in AppBuilder view. The default toolchain is GCC. If you are using IAR, change the Toolchain value in the drop down.



simple_trx_2 [IAR ARM - Default] [EFR32MG1P233F256GM48 - Silic
main.c
simple_trx_2.isc

Application configuration

Generation directory: Relative to ISC file C:\Users\CAOWENS\SimplicityStudio\v4_rel.Staging_272\simple_trx_2

Select architecture for this application:

Board type: BRD4150B: EFR32MG 2400/915 MHz 19.5 dBm Dual Band Radio Board; SLWSTK6002A

Toolchain: IAR Toolchain

Wireless MCU: EFR32 | Family: M | Series: 1 | Device Configuration: 1 | Performance: P | Radio: 233 | Flash: 256K | Temperature: G | Pkg: M | PINS: 48

Device name: simple_trx_2

Device type: No device type selected.

Information Configuration

Description

This is a customizable, simple transmit and receive sample application. Pressing PB0 will send a message. LED1 will toggle on message transmission and LED0 will toggle on message reception.

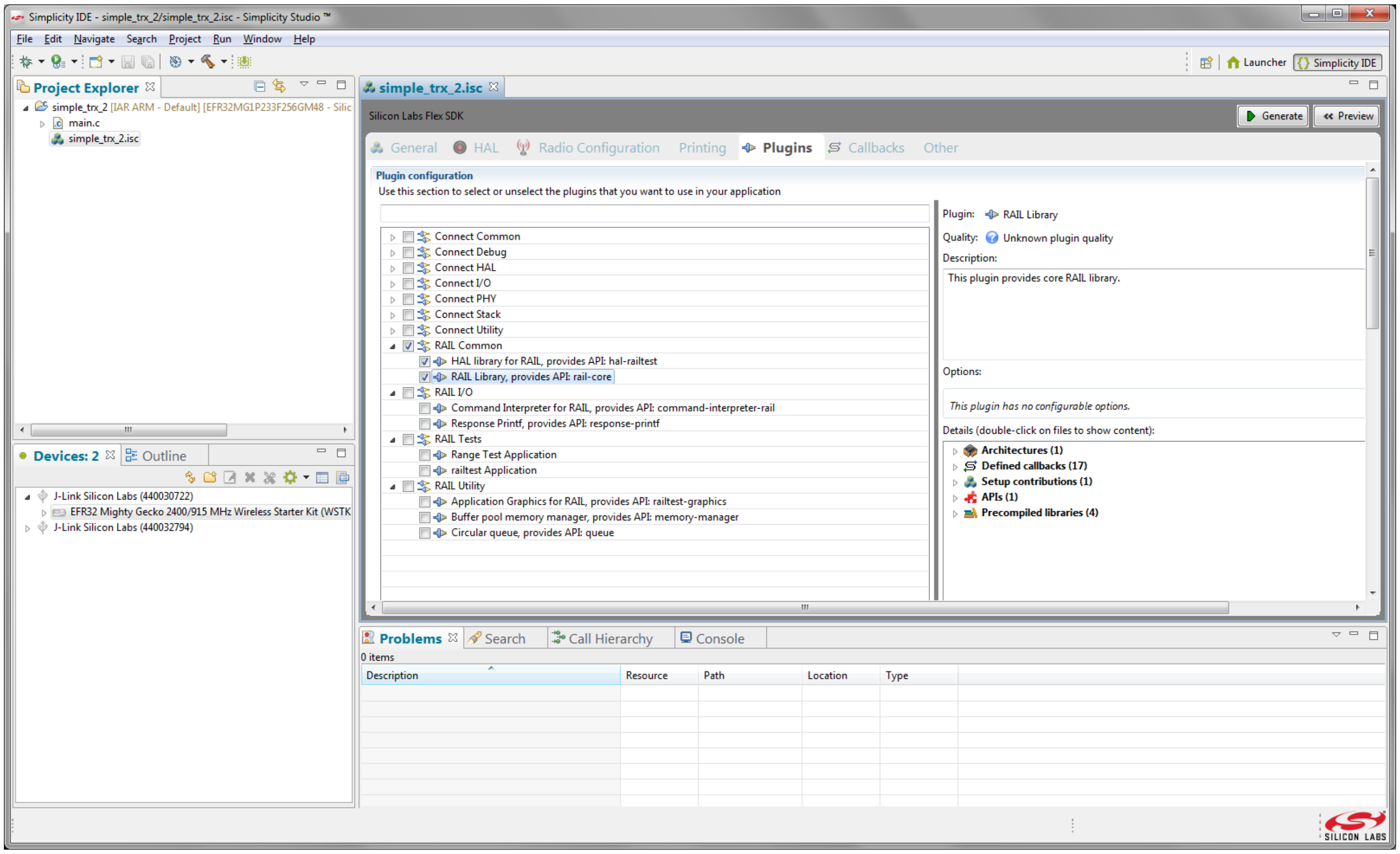
NOTE: Due to the higher current consumption of the continuous radio usage (especially in Rx Mode), it is not recommended to power the boards from a coin cell. Instead, an USB power bank can be used if portability is needed.

J-Link Silicon Labs (440030722)
EFR32 Mighty Gecko 2400/915 MHz Wireless Starter Kit (WSTK
J-Link Silicon Labs (440032794)

0 items

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

Compare the plugins tab for a RAIL application with the one shown for Connect. Only RAIL plugins are selected. Note that, in order to see information about the plugin, you either need to scroll up, or collapse the Connect plugins



Simplicity IDE - simple_trx_2/simple_trx_2.isc - Simplicity Studio ™

Project Explorer: simple_trx_2 [IAR ARM - Default] [EFR32MG1P233F256GM48 - Silic; main.c; simple_trx_2.isc

Devices: 2
- J-Link Silicon Labs (440030722)
  - EFR32 Mighty Gecko 2400/915 MHz Wireless Starter Kit (WSTK
- J-Link Silicon Labs (440032794)

simple_trx_2.isc — Silicon Labs Flex SDK

Plugin configuration
Use this section to select or unselect the plugins that you want to use in your application

- [ ] Connect Common
- [ ] Connect Debug
- [ ] Connect HAL
- [ ] Connect I/O
- [ ] Connect PHY
- [ ] Connect Stack
- [ ] Connect Utility
- [✓] RAIL Common
  - [✓] HAL library for RAIL, provides API: hal-railtest
  - [✓] RAIL Library, provides API: rail-core
- [ ] RAIL I/O
  - [ ] Command Interpreter for RAIL, provides API: command-interpreter-rail
  - [ ] Response Printf, provides API: response-printf
- [ ] RAIL Tests
  - [ ] Range Test Application
  - [ ] railtest Application
- [ ] RAIL Utility
  - [ ] Application Graphics for RAIL, provides API: railtest-graphics
  - [ ] Buffer pool memory manager, provides API: memory-manager
  - [ ] Circular queue, provides API: queue

Plugin: RAIL Library
Quality: Unknown plugin quality
Description:
This plugin provides core RAIL library.

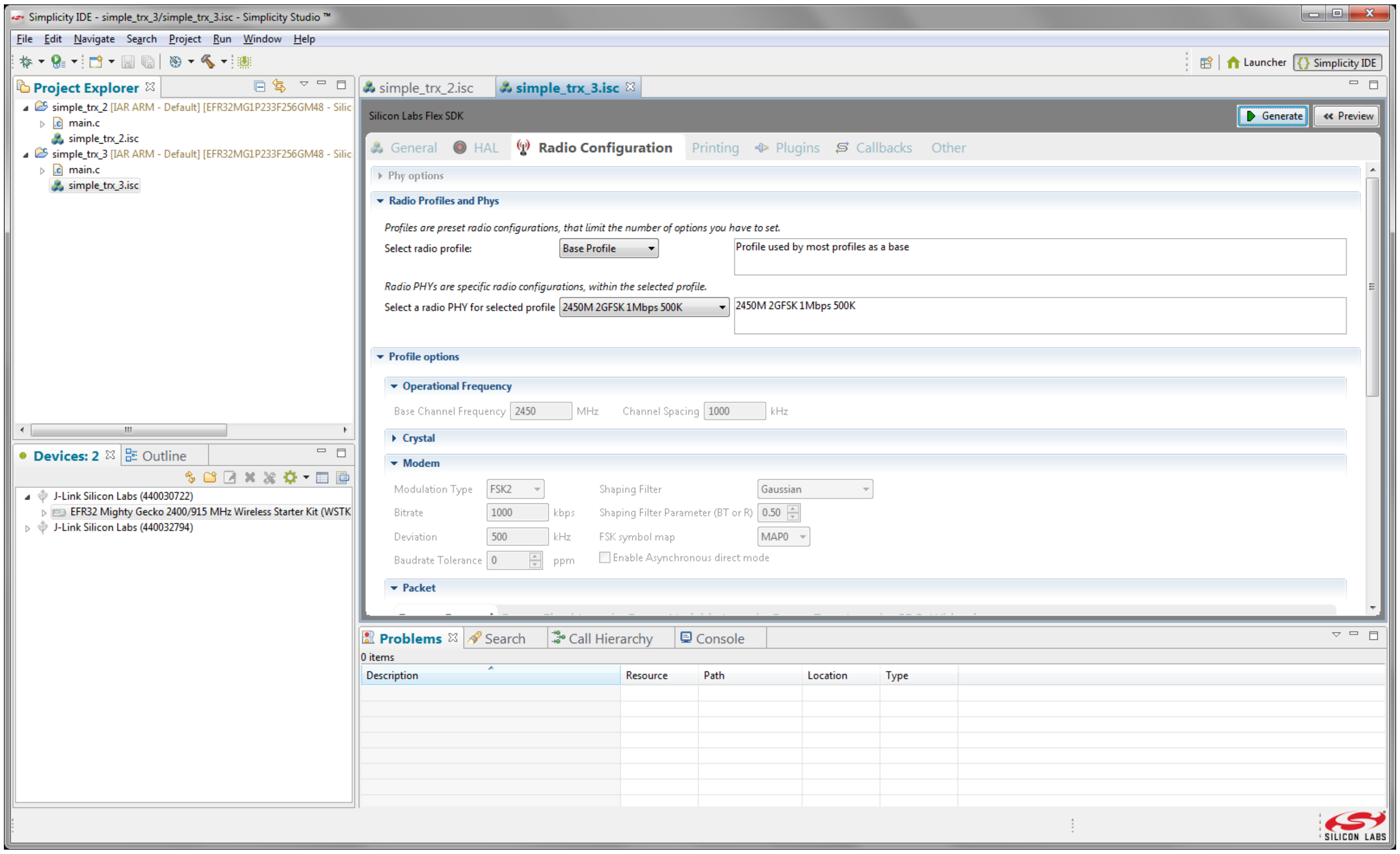
Options:
This plugin has no configurable options.

Details (double-click on files to show content):
- Architectures (1)
- Defined callbacks (17)
- Setup contributions (1)
- APIs (1)
- Precompiled libraries (4)

Problems
0 items

Description	Resource	Path	Location	Type

The RAIL application framework allows you to modify the PHY configuration for the application. Click the Radio Configuration tab, which allows you to select a preset radio configuration or build a custom radio configuration. It includes two main selections: Profile and Radio PHY. Profile gives you the ability to select a pre-defined set of PHY configurations. Once you have selected a profile, you can then select a Radio PHY within that profile.



simple_trx_2 [IAR ARM - Default] [EFR32MG1P233F256GM48 - Silic
main.c
simple_trx_2.isc
simple_trx_3 [IAR ARM - Default] [EFR32MG1P233F256GM48 - Silic
main.c
simple_trx_3.isc

J-Link Silicon Labs (440030722)
EFR32 Mighty Gecko 2400/915 MHz Wireless Starter Kit (WSTK
J-Link Silicon Labs (440032794)

Profiles are preset radio configurations, that limit the number of options you have to set.
Select radio profile: Base Profile — Profile used by most profiles as a base

Radio PHYs are specific radio configurations, within the selected profile.
Select a radio PHY for selected profile 2450M 2GFSK 1Mbps 500K — 2450M 2GFSK 1Mbps 500K

Base Channel Frequency 2450 MHz
Channel Spacing 1000 kHz

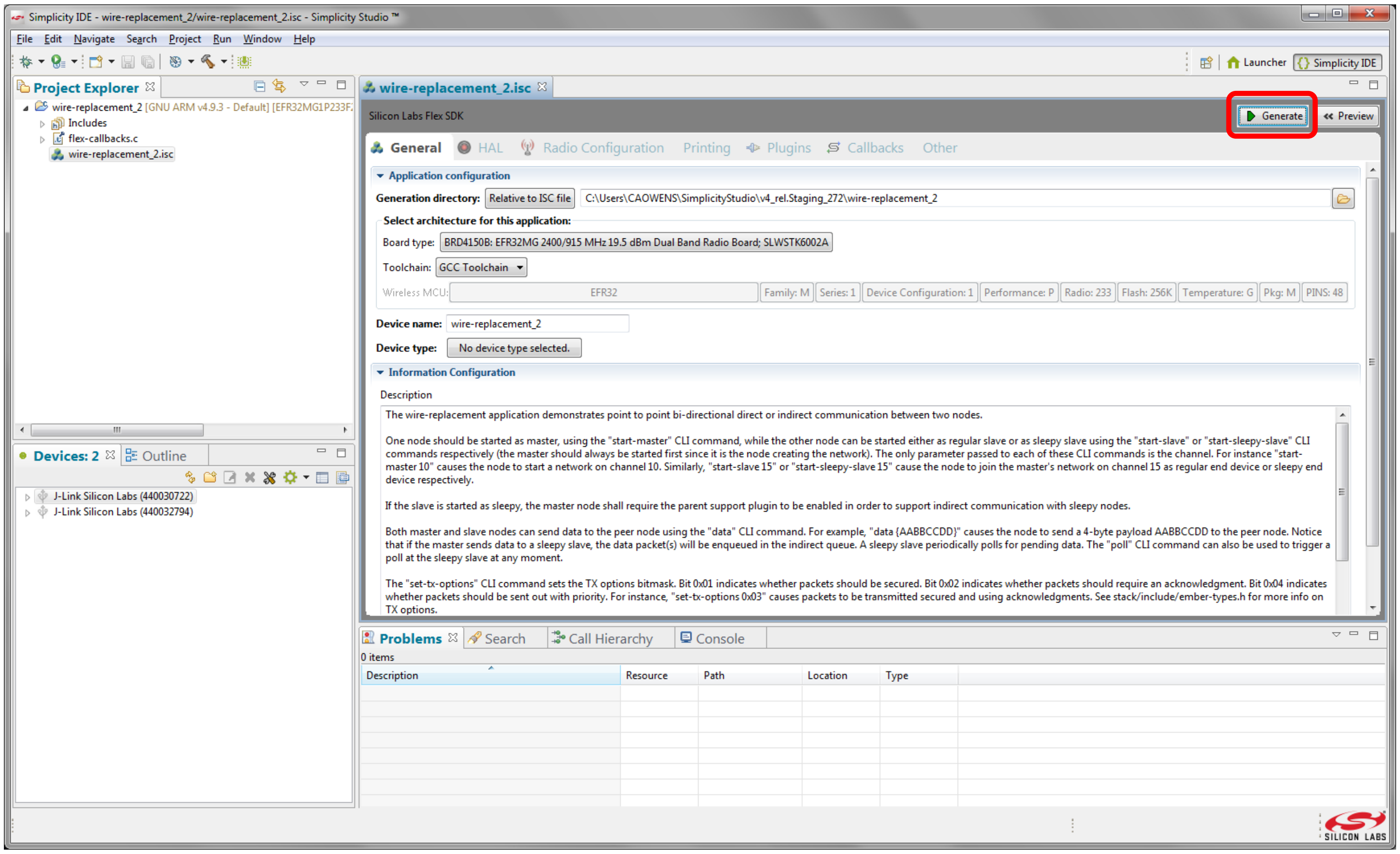
Modulation Type FSK2
Shaping Filter Gaussian
Bitrate 1000 kbps
Shaping Filter Parameter (BT or R) 0.50
Deviation 500 kHz
FSK symbol map MAP0
Baudrate Tolerance 0 ppm
Enable Asynchronous direct mode

0 items

For example, if you select the Basic Profile you can then select from a number of pre-defined PHY configurations, or create a custom PHY configuration. See *AN971: EFR32 Radio Configurator Guide* for more information on using the radio configurator.

4.2.2 Generating the RAIL Application

1. In the General tab, the Information Configuration block provides details about application setup and functionality (you may need to scroll down to see it). When you are ready to generate the sample project code, click **Generate.**



Application configuration

Generation directory: Relative to ISC file C:\Users\CAOWENS\SimplicityStudio\v4_rel.Staging_272\wire-replacement_2

Select architecture for this application:

Board type: BRD4150B: EFR32MG 2400/915 MHz 19.5 dBm Dual Band Radio Board; SLWSTK6002A

Toolchain: GCC Toolchain

Wireless MCU: EFR32 Family: M Series: 1 Device Configuration: 1 Performance: P Radio: 233 Flash: 256K Temperature: G Pkg: M PINS: 48

Device name: wire-replacement_2

Device type: No device type selected.

Information Configuration

Description

The wire-replacement application demonstrates point to point bi-directional direct or indirect communication between two nodes.

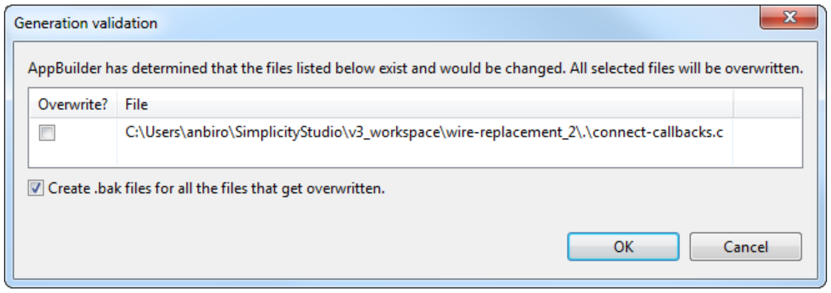
One node should be started as master, using the "start-master" CLI command, while the other node can be started either as regular slave or as sleepy slave using the "start-slave" or "start-sleepy-slave" CLI commands respectively (the master should always be started first since it is the node creating the network). The only parameter passed to each of these CLI commands is the channel. For instance "start-master 10" causes the node to start a network on channel 10. Similarly, "start-slave 15" or "start-sleepy-slave 15" cause the node to join the master's network on channel 15 as regular end device or sleepy end device respectively.

If the slave is started as sleepy, the master node shall require the parent support plugin to be enabled in order to support indirect communication with sleepy nodes.

Both master and slave nodes can send data to the peer node using the "data" CLI command. For example, "data {AABBCCDD}" causes the node to send a 4-byte payload AABBCCDD to the peer node. Notice that if the master sends data to a sleepy slave, the data packet(s) will be enqueued in the indirect queue. A sleepy slave periodically polls for pending data. The "poll" CLI command can also be used to trigger a poll at the sleepy slave at any moment.

The "set-tx-options" CLI command sets the TX options bitmask. Bit 0x01 indicates whether packets should be secured. Bit 0x02 indicates whether packets should require an acknowledgment. Bit 0x04 indicates whether packets should be sent out with priority. For instance, "set-tx-options 0x03" causes packets to be transmitted secured and using acknowledgments. See stack/include/ember-types.h for more info on TX options.

2. When asked about overwriting files, as long as you are working with examples do not check connect-callbacks.c. Only overwrite before starting your own coding, to create a file populated with stub callbacks for those you have indicated. Once you have made modifications to the callbacks in the file, be careful not to overwrite again.

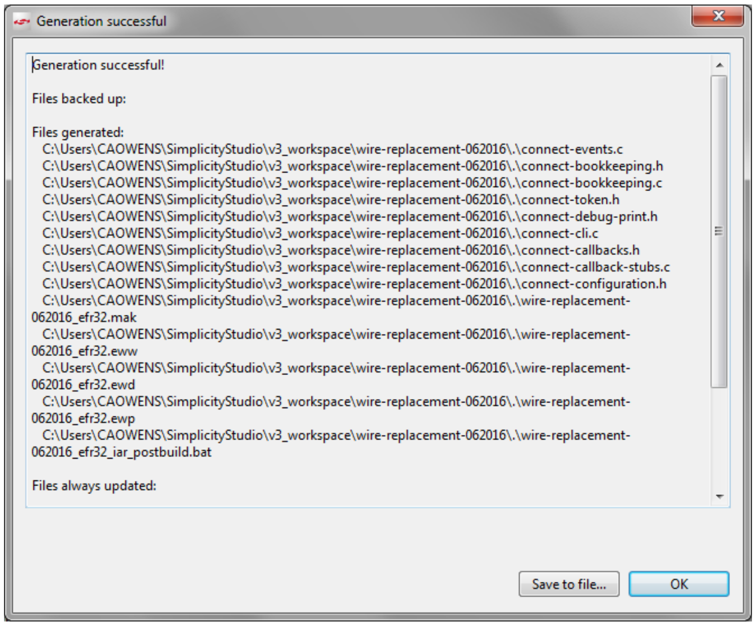


AppBuilder has determined that the files listed below exist and would be changed. All selected files will be overwritten.

Overwrite?	File
☐	C:\Users\anbiro\SimplicityStudio\v3_workspace\wire-replacement_2\.\connect-callbacks.c

☑ Create .bak files for all the files that get overwritten.

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



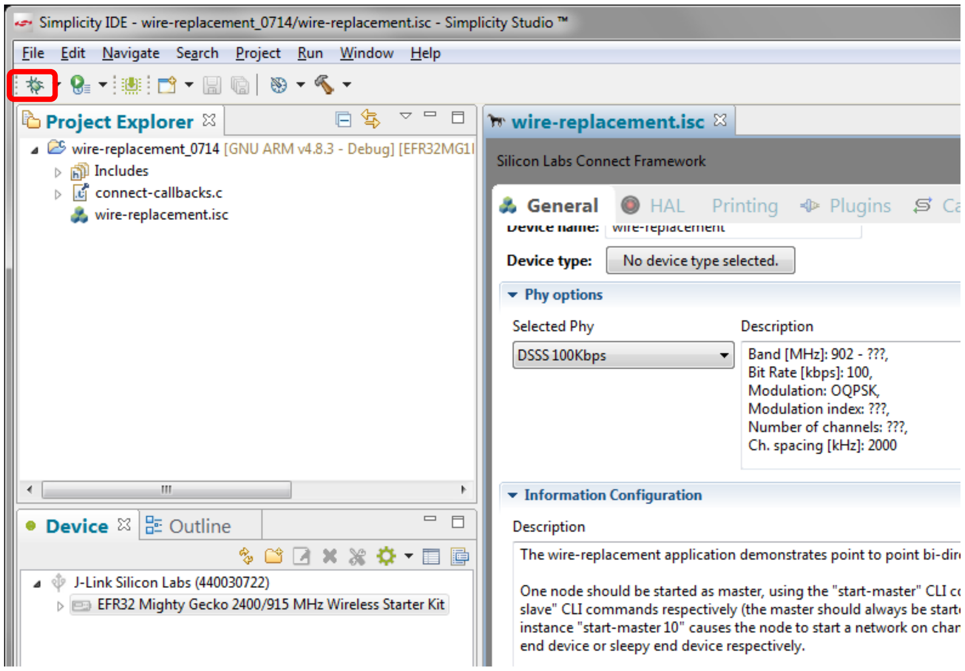
```
Generation successful!

Files backed up:

Files generated:
  C:\Users\CAOWENS\SimplicityStudio\v3_workspace\wire-replacement-062016\.\connect-events.c
  C:\Users\CAOWENS\SimplicityStudio\v3_workspace\wire-replacement-062016\.\connect-bookkeeping.h
  C:\Users\CAOWENS\SimplicityStudio\v3_workspace\wire-replacement-062016\.\connect-bookkeeping.c
  C:\Users\CAOWENS\SimplicityStudio\v3_workspace\wire-replacement-062016\.\connect-token.h
  C:\Users\CAOWENS\SimplicityStudio\v3_workspace\wire-replacement-062016\.\connect-debug-print.h
  C:\Users\CAOWENS\SimplicityStudio\v3_workspace\wire-replacement-062016\.\connect-cli.c
  C:\Users\CAOWENS\SimplicityStudio\v3_workspace\wire-replacement-062016\.\connect-callbacks.h
  C:\Users\CAOWENS\SimplicityStudio\v3_workspace\wire-replacement-062016\.\connect-callback-stubs.c
  C:\Users\CAOWENS\SimplicityStudio\v3_workspace\wire-replacement-062016\.\connect-configuration.h
  C:\Users\CAOWENS\SimplicityStudio\v3_workspace\wire-replacement-062016\.\wire-replacement-
062016_efr32.mak
  C:\Users\CAOWENS\SimplicityStudio\v3_workspace\wire-replacement-062016\.\wire-replacement-
062016_efr32.eww
  C:\Users\CAOWENS\SimplicityStudio\v3_workspace\wire-replacement-062016\.\wire-replacement-
062016_efr32.ewd
  C:\Users\CAOWENS\SimplicityStudio\v3_workspace\wire-replacement-062016\.\wire-replacement-
062016_efr32.ewp
  C:\Users\CAOWENS\SimplicityStudio\v3_workspace\wire-replacement-062016\.\wire-replacement-
062016_efr32_iar_postbuild.bat

Files always updated:
```

4.2.3 Compiling and Flashing the RAIL Example

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

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.


Simplicity IDE - wire-replacement_0714/wire-replacement.isc - Simplicity Studio ™
File
Edit
Navigate
Search
Project
Run
Window
Help
Project Explorer
wire-replacement_0714 [GNU ARM v4.8.3 - Debug] [EFR32MG1
Includes
connect-callbacks.c
wire-replacement.isc
Device
Outline
J-Link Silicon Labs (440030722)
EFR32 Mighty Gecko 2400/915 MHz Wireless Starter Kit
wire-replacement.isc
Silicon Labs Connect Framework
General
HAL
Printing
Plugins
Ca

Device name: wire-replacement
Device type: No device type selected.

Phy options

Selected Phy	Description
DSSS 100Kbps	Band [MHz]: 902 - ???, Bit Rate [kbps]: 100, Modulation: OQPSK, Modulation index: ???, Number of channels: ???, Ch. spacing [kHz]: 2000

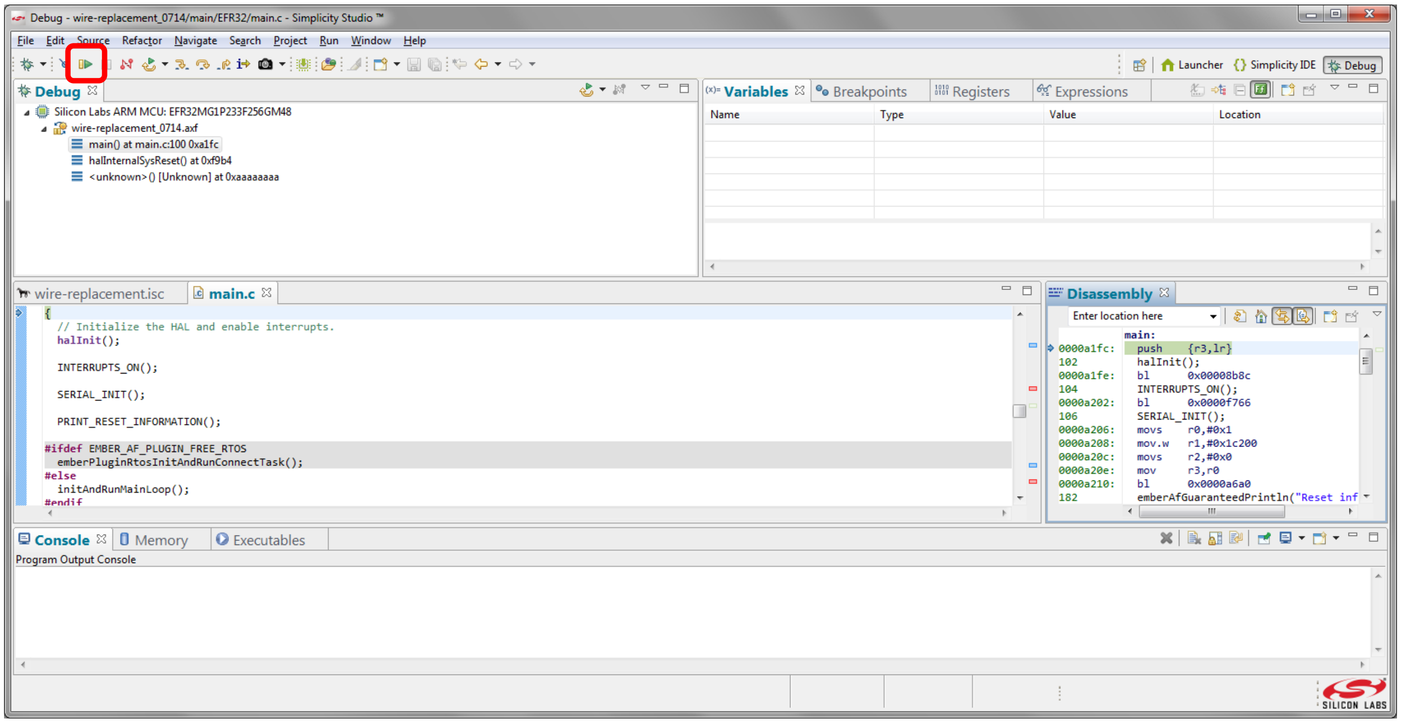
Information Configuration

Description

The wire-replacement application demonstrates point to point bi-dir

One node should be started as master, using the "start-master" CLI co
slave" CLI commands respectively (the master should always be start
instance "start-master 10" causes the node to start a network on char
end device or sleepy end device respectively.

2. Progress is displayed in the lower left. Wait until flashing has completed and a Debug perspective is displayed. Click the **Resume** control to start the application running on the WSTK.

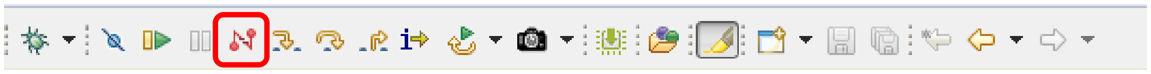


```
Silicon Labs ARM MCU: EFR32MG1P233F256GM48
  wire-replacement_0714.axf
    main() at main.c:100 0xa1fc
    halInternalSysReset() at 0xf9b4
    <unknown>() [Unknown] at 0xaaaaaaaa
```

```
{
  // Initialize the HAL and enable interrupts.
  halInit();

  INTERRUPTS_ON();

  SERIAL_INIT();

  PRINT_RESET_INFORMATION();

#ifdef EMBER_AF_PLUGIN_FREE_RTOS
  emberPluginRtosInitAndRunConnectTask();
#else
  initAndRunMainLoop();
#endif
```

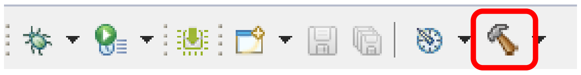
```
           main:
0000a1fc:    push    {r3,lr}
102          halInit();
0000a1fe:    bl      0x00008b8c
104          INTERRUPTS_ON();
0000a202:    bl      0x0000f766
106          SERIAL_INIT();
0000a206:    movs    r0,#0x1
0000a208:    mov.w   r1,#0x1c200
0000a20c:    movs    r2,#0x0
0000a20e:    mov     r3,r0
0000a210:    bl      0x0000a6a0
182          emberAfGuaranteedPrintln("Reset inf
```

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



Manually Compile and Flash

After you generate your project files, instead of clicking Debug in the Simplicity IDE, click the **Build** control in the top tool bar. Your sample application will compile based on its build configuration. You can change the build configuration at any time by right clicking on the project and going to **Build Configurations > Set Active**.

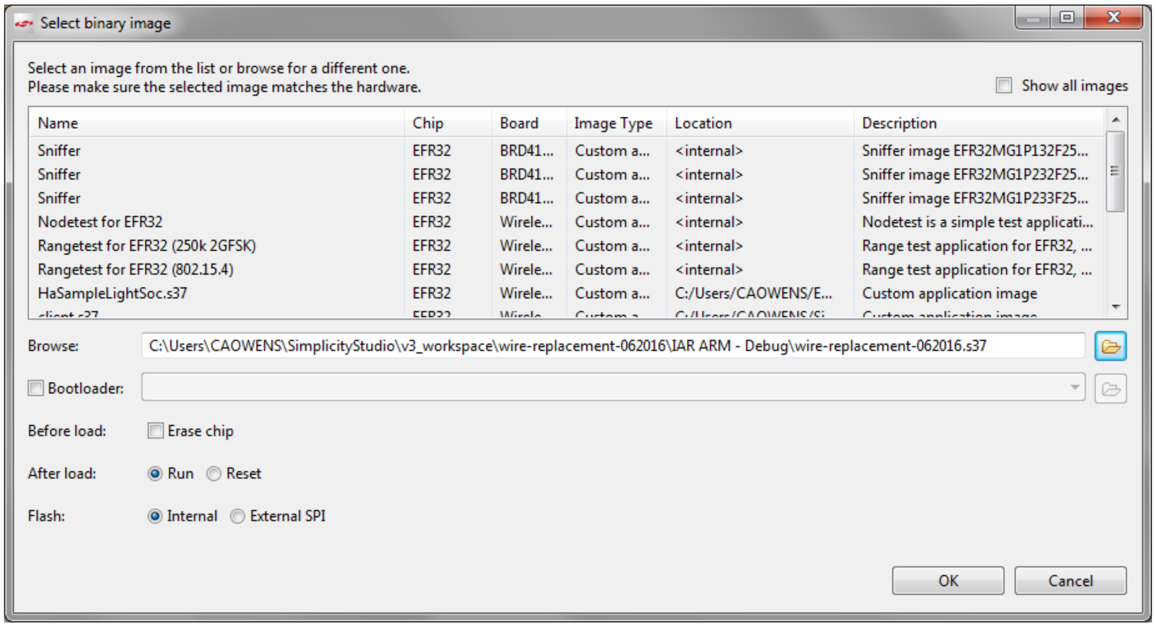


You can also build your application directly in IAR-EWARM by opening IAR-EWARM and opening the generated project file inside IAR.

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 the Devices view in the Simplicity IDE perspective.

1. In the Devices view, right-click on the J-Link device and select **Upload Application**. The Select Binary Image dialog is displayed.



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

☐ Show all images

Name	Chip	Board	Image Type	Location	Description
Sniffer	EFR32	BRD41...	Custom a...	<internal>	Sniffer image EFR32MG1P132F25...
Sniffer	EFR32	BRD41...	Custom a...	<internal>	Sniffer image EFR32MG1P232F25...
Sniffer	EFR32	BRD41...	Custom a...	<internal>	Sniffer image EFR32MG1P233F25...
Nodetest for EFR32	EFR32	Wirele...	Custom a...	<internal>	Nodetest is a simple test applicati...
Rangetest for EFR32 (250k 2GFSK)	EFR32	Wirele...	Custom a...	<internal>	Range test application for EFR32, ...
Rangetest for EFR32 (802.15.4)	EFR32	Wirele...	Custom a...	<internal>	Range test application for EFR32, ...
HaSampleLightSoc.s37	EFR32	Wirele...	Custom a...	C:/Users/CAOWENS/E...	Custom application image
client.s37	EFR32	Wirele...	Custom a...	C:/Users/CAOWENS/Si...	Custom application image

Browse: C:\Users\CAOWENS\SimplicityStudio\v3_workspace\wire-replacement-062016\IAR ARM - Debug\wire-replacement-062016.s37

☐ Bootloader:

Before load: ☐ Erase chip

After load: [✓ Run] Reset

Flash: [✓ Internal] External SPI

2. Navigate to the .s37, .bin or .hex image you wish to upload.
3. Optionally, check **Erase Chip**, to make sure that any previous bootloader or other non-volatile data is erased before your new image is uploaded. New users will typically check this.
4. The **After Load** options are **Run** (begin executing the code immediately) and **Reset** (wait for an event, such as a debugger to connect or manual initiation of a boot sequence). During initial development you will typically leave this set to **Run**.
5. Click **OK.**

4.2.4 Interacting with the RAIL Example

Interaction in the simple TRX example is through buttons on the WSTK

Repeat the procedures in section **Selecting a RAIL Example Application** through **Compiling and Flashing the RAIL Example** to compile and load the simple TRX example on a second device.

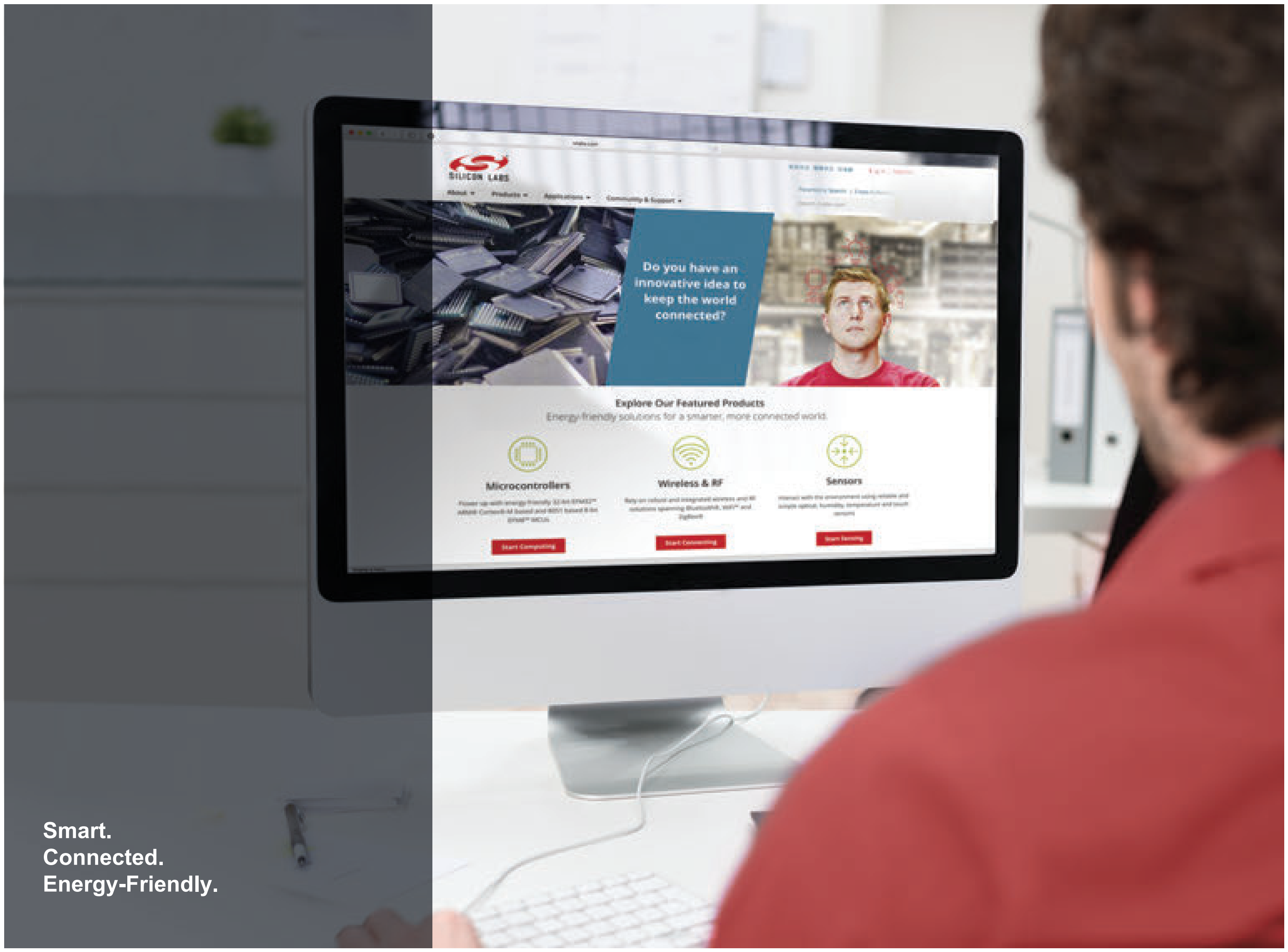
Press PB0 on Device A sends a message. LED1 on Device A toggles on message transmission and LED0 on Device B toggles on message reception.

5 Appendix: EFR32/EZR32 Comparison

The following table compares the Connect features for the EZR32 with those for the EFR32.

Table 1. EZR32/EFR32 Connect Feature Comparison

Connect Feature	EFR32	EZR32
PHY Support (FCC, ETSI, Anatel, China, ARIB, KCC)	Yes	Yes
Networking Stack Functions (Network formation, Scanning, Association, CSMA-CA, Filtering, Retries, Auto-Acks)	Yes	Yes
Node Types (End Nodes, Sleepy End Nodes, Coordinator, Range Extender)	Yes	Yes
Network Topologies (Point-to-Point, Star, Sleepy Star, Extended Star)	Yes	Yes
DSSS	Yes	No
NCP Mode	Yes	No
Security (XTEA, AES)	Yes	Yes
AppBuilder Support in Simplicity Studio	Yes	Yes
Hardware Configuration in Simplicity Studio	No (Future Release)	No (Future Release)
Frequency Hopping	FCC - Yes (ETSI - Future Release)	No
Secure Bootloader	No (Future Release)	No
OTA Updates (Broadcast/Unicast)	Yes	No
FreeRTOS Support	No	No
Radio Configuration	Yes	No (Pre-Set Config.)
Compilers	IAR, GCC	IAR
Multi-Address Filtering	Yes	No
Antenna Diversity	No (Future Release)	No
Sample Applications	Point-to-Point Sensor/Sink	Point-to-Point Sensor/Sink


Smart.
Connected.
Energy-Friendly.
SILICON LABS
Do you have an innovative idea to keep the world connected?
Explore Our Featured Products
Energy-friendly solutions for a smarter, more connected world.
Microcontrollers
Wireless & RF
Sensors

Products
www.silabs.com/products

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

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

http://www.silabs.com