



# QSG121: Getting Started with Silicon Labs RAIL on the Wireless Gecko (EFR32™) Portfolio

---

This quick start guide provides basic information on configuring, building, and installing applications for the Wireless Gecko (EFR32) portfolio using Silicon Labs RAIL (Radio Abstraction Interface Layer). Silicon Labs RAIL provides an intuitive, easily-customizable radio interface layer that is designed to support proprietary or standards-based wireless protocols. The RAIL API is documented in an online API reference as well as other documents installed with the stack installer, or available through Simplicity Studio. RAIL is delivered as a library that you can link to your applications. A description of library functions is provided in the development environment.

This guide is designed for new users of Silicon Labs RAIL on EFR32, and provides instructions to get started using one of the example applications included in the Simplicity Studio development environment.

## KEY FEATURES

- Overview of RAIL software components.
- Running Simplicity Studio for the first time.
- Building a sample application.
- Interacting with the sample application.

# 1 Overview

This guide is designed for new users of Silicon Labs RAIL on the EFR32, and provides instructions to get started using one of the example applications included in the RAIL SDK that can be loaded into the Simplicity Studio development environment. This guide assumes that you have already:

- Created an account on the Silicon Labs website, for access to the RAIL SDK.
- Purchased your Flex Gecko Wireless Starter Kit (WSTK).
- Downloaded the remaining software components. A card in your WSTK contains a link to a Getting Started page, which will direct you to links for the Silicon Labs software products. .

## 1.1 Software Components

To develop Silicon Labs RAIL-based applications for the EFR32 you will need the following components:

- The Simplicity Studio development environment.
- The Silicon Labs RAIL SDK, installed separately from Simplicity Studio
- (optional) A compiler other than the native Simplicity Studio compiler. You can use either:
  - IAR Embedded Workbench for ARM 7.30. You can download a trial of IAR EWARM 7.30 from the IAR website and use it with a 30 day trial: <http://www.iar.com/Products/IAR-Embedded-Workbench/ARM/>
  - GCC ARM Embedded Toolchain v. 4.9 (from <https://launchpad.net/gcc-arm-embedded>)

## 1.2 About the RAIL SDK

The Silicon Labs RAIL SDK consists of several components.

- The RAIL library: Provides a programming interface to radio functionality.
- The Radio Configurator: Part of Simplicity Studio's AppBuilder, the user interface that allows developers to configure static parameters of the radio physical layer.
- Sample applications: Can be used as is for evaluation and also serve as a starting point for application development. The RAILtest example application 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. RAILtest can be built with any PHY, including 802.15.4 and Bluetooth Smart. Many of the RAILtest serial commands can be used for lab evaluation.
- Documentation, including an API reference and application notes.

See UG103.13, *Application Development Fundamentals: RAIL*, for more information about RAIL and the RAIL SDK.

## 1.3 Support

Users can access the Silicon Labs support portal at <https://www.silabs.com/support>. Use the support portal to contact Customer Support for any questions you might have during the development process.

## 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 1.1, Software Components, for the list).

### 2.2 Install your Silicon Labs Stack or Software Development Kit

Install Silicon Labs software (see section 1, Overview, for more information).

### 2.3 Connect your Hardware

Connect your WSTK, with radio board mounted, to your PC using a USB cable.

**Note:** For best performance in Simplicity Studio please be sure that the power switch on your WSTK is in the Advanced Energy Monitoring or “AEM” position.

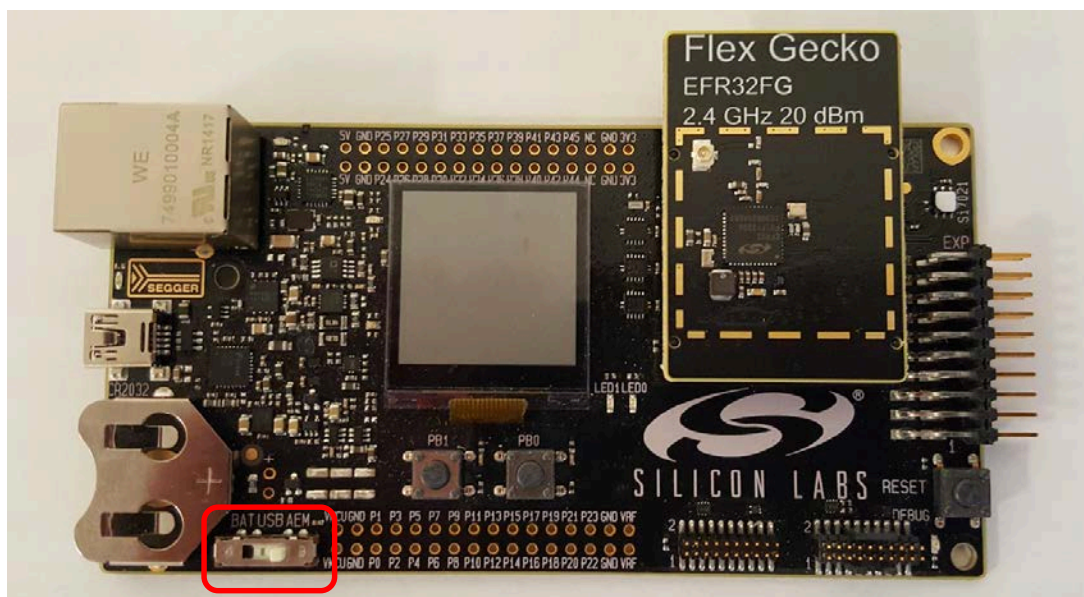


Figure 1. EFR32 Flex Gecko Hardware

**Note:** If you are going to use the WSTK as a mobile device to run Range Test or another sample test application, it is recommended that you connect the external AA battery pack to the WSTK board. The coin cell battery will not have enough power to do long-term testing using sample applications like Range Test and RAILtest.

## 2.4 Install Simplicity Studio

During installation, Simplicity Studio obtains updates and additional packages specific to your connected hardware.

1. As soon as Simplicity Studio launches, it searches for updates. This operation can take several minutes.

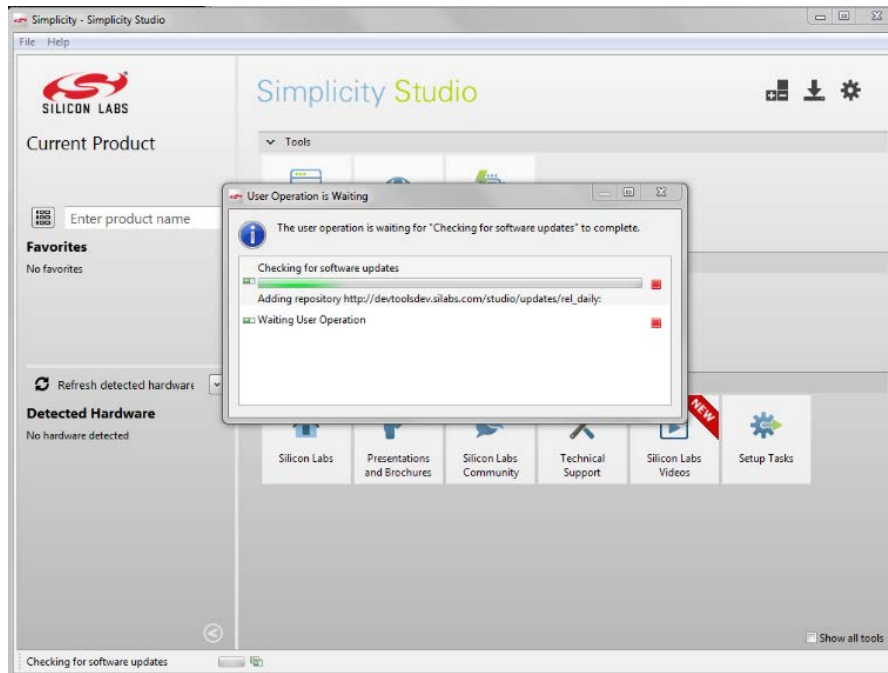


Figure 2. Checking for Software Updates

2. Once software update is complete, Simplicity Studio checks for connected hardware. If you have not connected your WSTK, you are prompted to do so (Figure 3). Connect your hardware and, when the screen changes to show that the hardware has been found, click **[Finish]**.

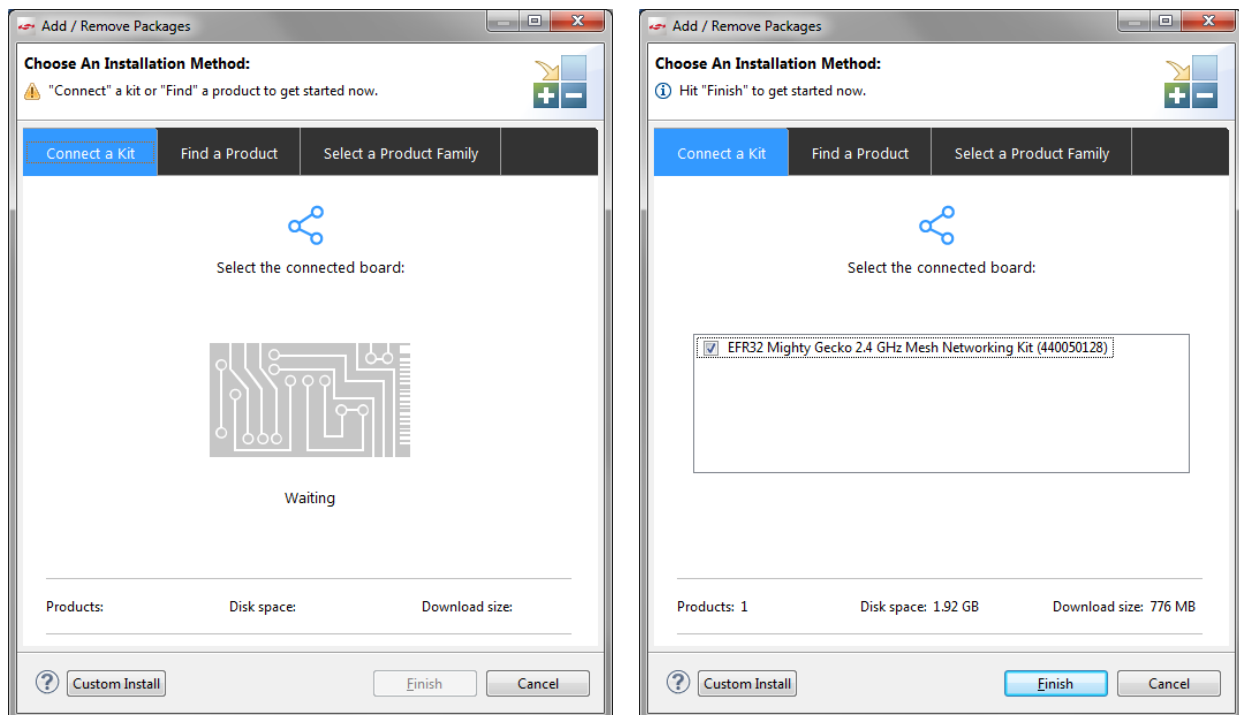
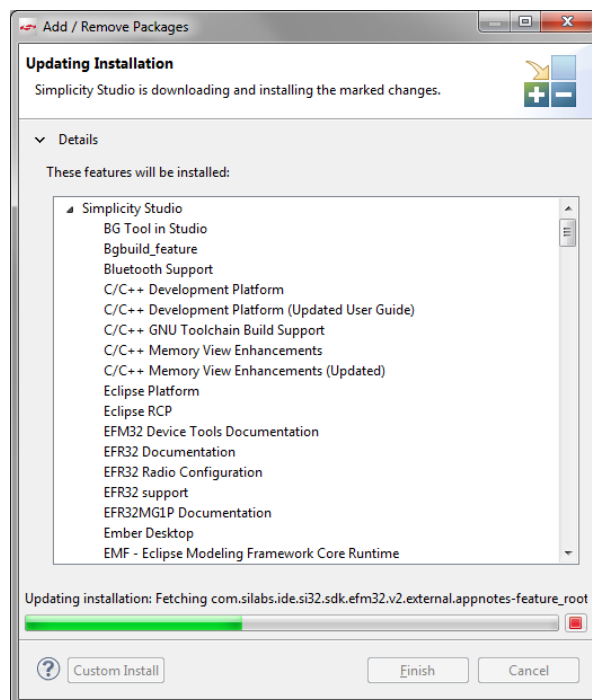


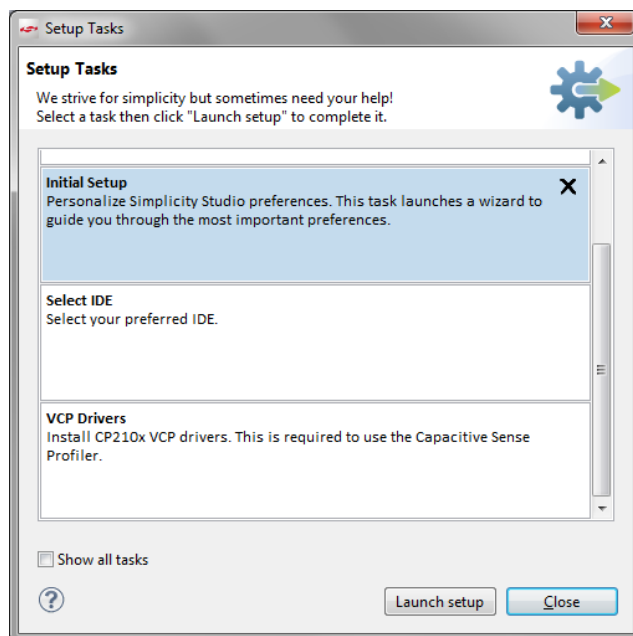
Figure 3. Connect a Kit Before and After Connection

3. Simplicity Studio then installs software packages related to your connected hardware (Figure 4). This procedure can take some time, during which the green progress indicator may appear stationary. However, the update steps above the progress indicator are continuously refreshed.



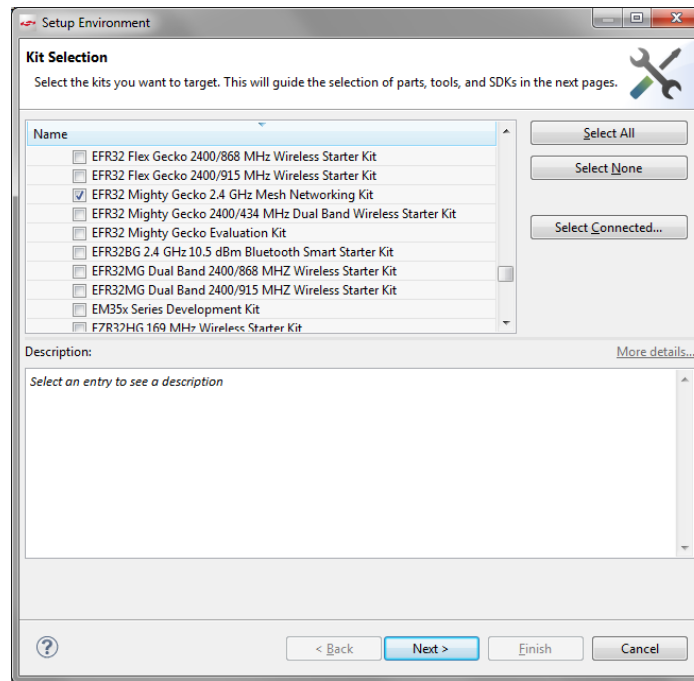
**Figure 4. Installation Update**

4. After update is complete, restart Simplicity Studio.
5. Once restart is complete, a menu of setup tasks is displayed (Figure 5). Select **Initial Setup** and then click **[Launch setup]**.



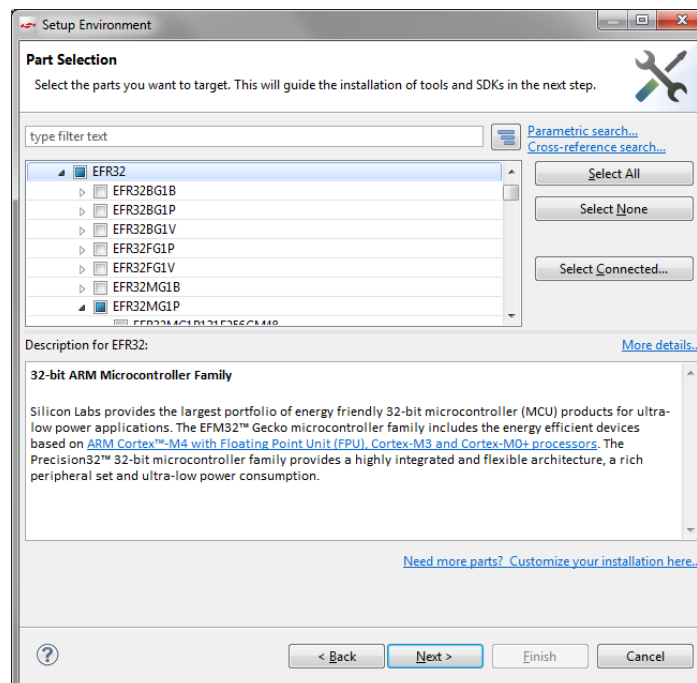
**Figure 5. Setup Tasks**

6. A kit selection dialog is displayed (Figure 6). Your connected kit should be selected. Click **[Next >]**.



**Figure 6. Kit Selection Dialog**

7. A part selection dialog is displayed (Figure 7). Your connected part should be selected. Click **[Next >]**.



**Figure 7. Part Selection Dialog**

8. A Build Environment dialog is displayed that shows detected items (Figure 8). If a Toolchain or SDK is not shown, you can click **Add...** to configure it now, or configure it later from the Settings control. Adding an SDK is described in section 3, Discovering the RAIL SDK, Click **[Finish]**. The Simplicity perspective, described in the next section, is displayed.

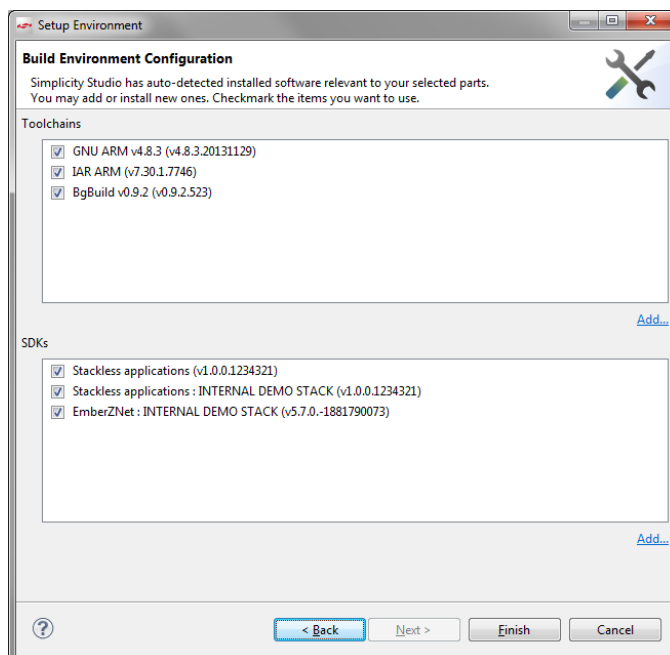


Figure 8. Build Environment Configuration

## 2.5 Navigation in Simplicity Studio

Simplicity Studio is built on the Eclipse platform. As such, it is broken up into different “perspectives,” each of which allows access to a specific set of functionality. Simplicity Studio starts up in the “Simplicity perspective,” sometimes referred to as the “Home Screen” (Figure 9).

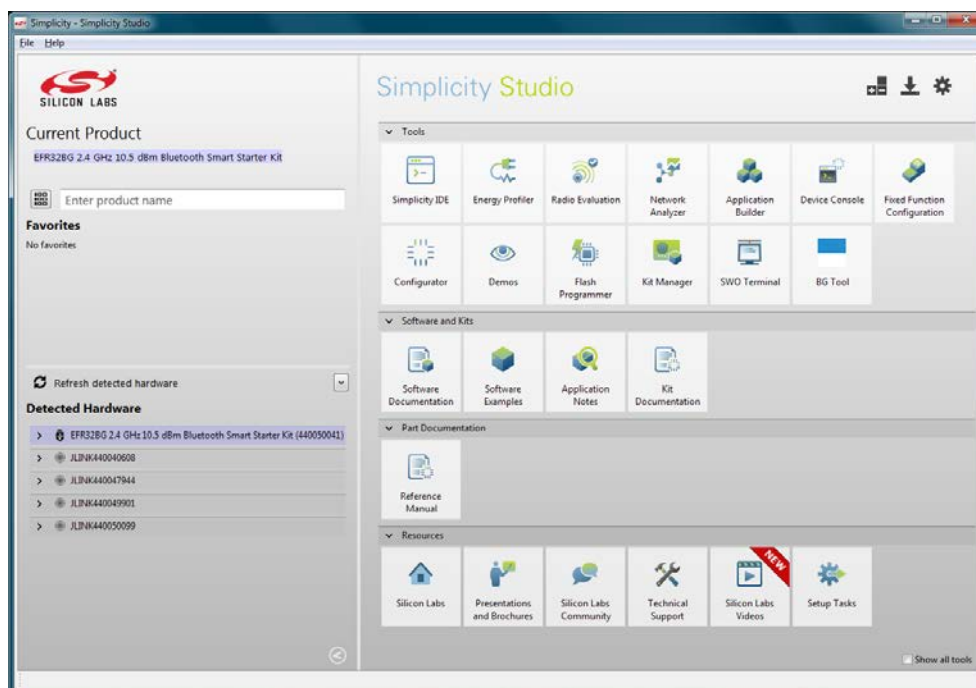
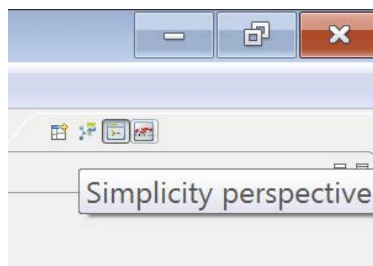


Figure 9. Simplicity Studio's Simplicity Perspective

From the Simplicity perspective, you can discover devices, configure Simplicity Studio, or navigate to another perspective for application development.

The Simplicity perspective shows large tile icons that represent the various sets of functionality within Simplicity Studio. Clicking a tile opens a different perspective. When you are in a different perspective, you can always return to the Simplicity perspective or any other perspective at any time by clicking on one of the tile icons in the top right-hand corner of your screen (Figure 10).



**Figure 10. Navigation Tile Icons**

Your detected hardware is shown in the **Detected Hardware** panel in the lower left of the Simplicity perspective. If you change connected hardware and it does not auto-detect, click **[Refresh Detected Hardware]**.

Three controls in the upper right (Figure 11) allow you to maintain part-specific packages, install updates to the core Simplicity Studio software, and change configuration settings.



**Figure 11. Maintenance Tools**



### 3 Discovering the RAIL SDK

If you are discovering from the initial setup process **Add ...** step, go to step 4. If you are discovering the SDK from the Settings icon, begin with step 1.

1. Click the Settings icon (Figure 12) on the top right-hand corner of the Simplicity perspective to open the Preferences window.

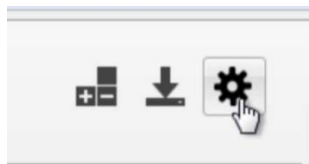


Figure 12. Settings Icon

2. In the Preferences window's left frame, click **Simplicity Studio > SDKs** to open the configuration dialog (Figure 13).

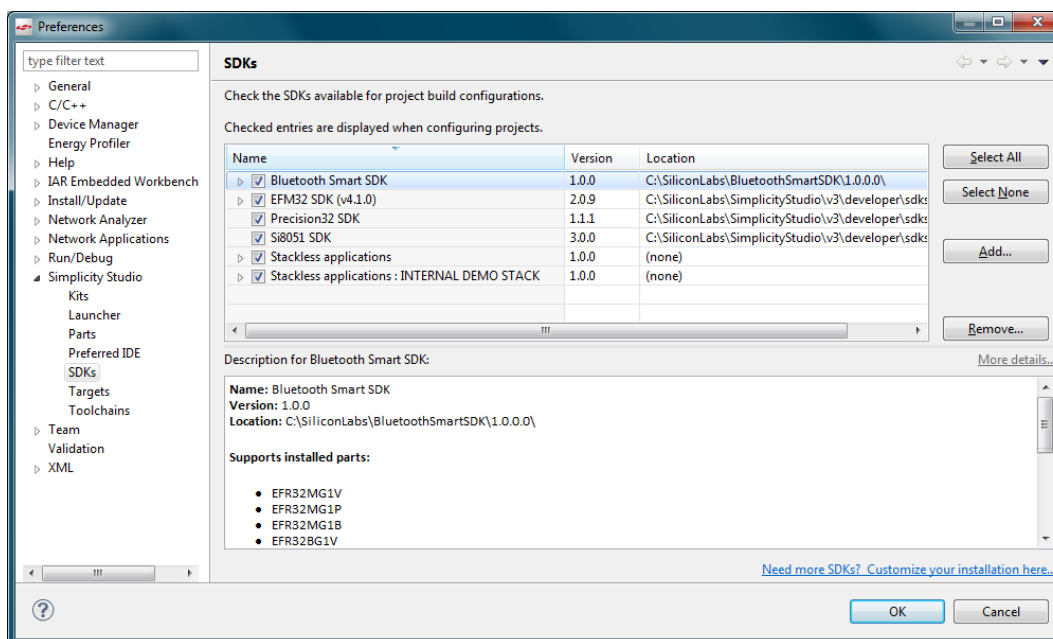


Figure 13. Silicon Labs SDK Configuration

3. Click **[Add]**.
4. Browse to the location where you have installed the SDK, and click **[OK]**.
5. Verify the software has detected the RAIL SDK (Figure 14) and click **[OK]**.

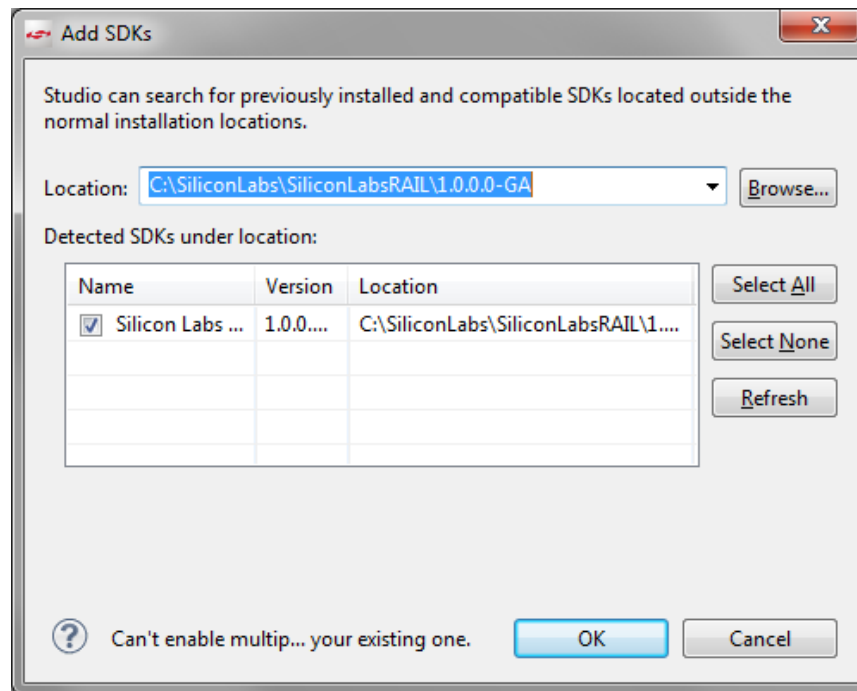


Figure 14. Add SDKs

- Click **[OK]** to return to the starting configuration page.
- Click **[Finish]** (from initial configuration) or **[OK]** (from the settings icon).

## 4 Building a Sample Application

The sample applications included in this release are:

**Silicon Labs RAIL Simple TRx Sample Application for EFR32:** This is a Silicon Labs Simple Transmit and Receive application demo. It is an example application that demonstrates the simplest transmit and receive sample application based on RAIL, with the following default PHY: frequency: 2.4 GHz modulation: 2 GFSK bitrate: 1 Mbps deviation: 500 kHz. There is no command line interface or LCD display associated with the Simple TRx sample application. The only interface for this sample is through the buttons on the WSTK. Both PB0 and PB1 will cause a message to be transmitted. When a message is received one of the LEDs will light.

**Silicon Labs RAIL Bidirectional Sample Application for EFR32:** This is a Silicon Labs Simple Transmit and ACK application demo. It is an example application which demonstrates the simplest transmit and ACK sample application based on RAIL with the following default PHY: frequency: 2.4 GHz modulation: 2 GFSK bitrate: 1 Mbps deviation: 500 kHz. There is no command line interface included in this application. Transmission is managed through the buttons on the WSTK. If you click on either button, a message will be sent and, if the receiving node has the same sample loaded, it will respond with an ACK. When a message is received one of the buttons will light up, when an ACK is received the other LED will light.

**Silicon Labs RAIL Energy Mode Sample Application for EFR32:** This is a Silicon Labs Energy Mode Sample Application that demonstrates the low power modes of the EFR32 with the following default PHY: frequency: 2.4 GHz modulation: 2 GFSK bitrate: 1 Mbps deviation: 500 kHz. The Energy Mode sample application displays the current status of the application on the WSTK's onboard LCD display. Use PB1 to move between energy modes and PB0 to transmit a packet from any energy mode. The performance of the EFR32 within each energy mode can be seen in Simplicity Studio's Energy Profiler.

**Silicon Labs RAIL RAILtest Sample Application for EFR32:** This is a Silicon Labs RAILtest application demo. It is an example application which demonstrates the features of RAIL through a command line interface with the following default PHY: frequency: 2.4 GHz modulation: 2 GFSK bitrate: 1 Mbps deviation: 500 kHz. For a complete list of commands type "help" on the command line through Simplicity Studio's Console interface. See document AN972, *EFR32 RF Evaluation Guide*, for more information on using RAILtest,

**Silicon Labs RAIL Range Test Sample Application for EFR32:** This is a customizable Range Test Sample Application that demonstrates over the air range of the EFR32. The default PHY configuration for this sample is: frequency: 2.4 GHz deviation, modulation and bitrate consistent with 802.15.4 specification. Users can change this PHY configuration to whatever settings they would like. The Range Test sample application uses the LCD display to show a partial list of PHY configuration settings. Use the buttons PB0 and PB1 to move through the interface on the LCD, set the device to receive or transmit mode, and start and end tests.

**Silicon Labs RAIL Range Test Sample Application for EFR32 - PHY: 250k, 2GFSK:** This is a customizable Range Test Sample Application that demonstrates over the air range of the EFR32 with the following PHY: frequency: 2.4 GHz modulation: 2 GFSK bitrate: 250 Kbps deviation: 125 kHz. Other than the default PHY configuration, this sample is identical to the Range Test Sample listed above.

**Silicon Labs RAIL Range Test Sample Application for EFR32 - PHY: 802.15.4:** This is a customizable Range Test Sample Application that demonstrates over-the-air range of the EFR32 with the following PHY: frequency: 2.4 GHz deviation, modulation and bitrate consistent with 802.15.4 specification. Other than the default PHY configuration, this sample is identical to the Range Test Sample listed above.

**Silicon Labs RAIL Duty Cycle Sample Application for EFR32:** The application has three modes of operation: Duty Cycle, Master, and Slave. Button1 is used to change modes. Application modes and states are displayed on the LCD of the WSTK. The application supports Duty Cycle mode testing, where both nodes are in Duty Cycle mode, and Master mode and Slave mode testing, where one node is in Master mode and the other in Slave mode.

- Duty Cycle mode: The application oscillates between EM1 and RX state and uses a long preamble length to catch every packet. Button0 transmits a single packet.
- Master mode: The application stays in IDLE state. Button0 blasts consecutive packets. The blast is followed by RX state to receive ACK from the Slave node.
- Slave mode: The application oscillates between EM2 and RX state. Once the Slave node finishes receiving a blast from the Master node, it sends an ACK and goes back to sleep. Button0 does not transmit in slave mode.

To build a sample application you must:

1. Create a new application based on a sample
2. Generate the application files
3. Build the application image using the IAR-EWARM or GCC compiler
4. Load the image onto your device

You can then interact with the application on the device to form a network or send data.

## 4.1 Selecting a sample application

1. In the Simplicity perspective, click the Software Examples tile (Figure 15).

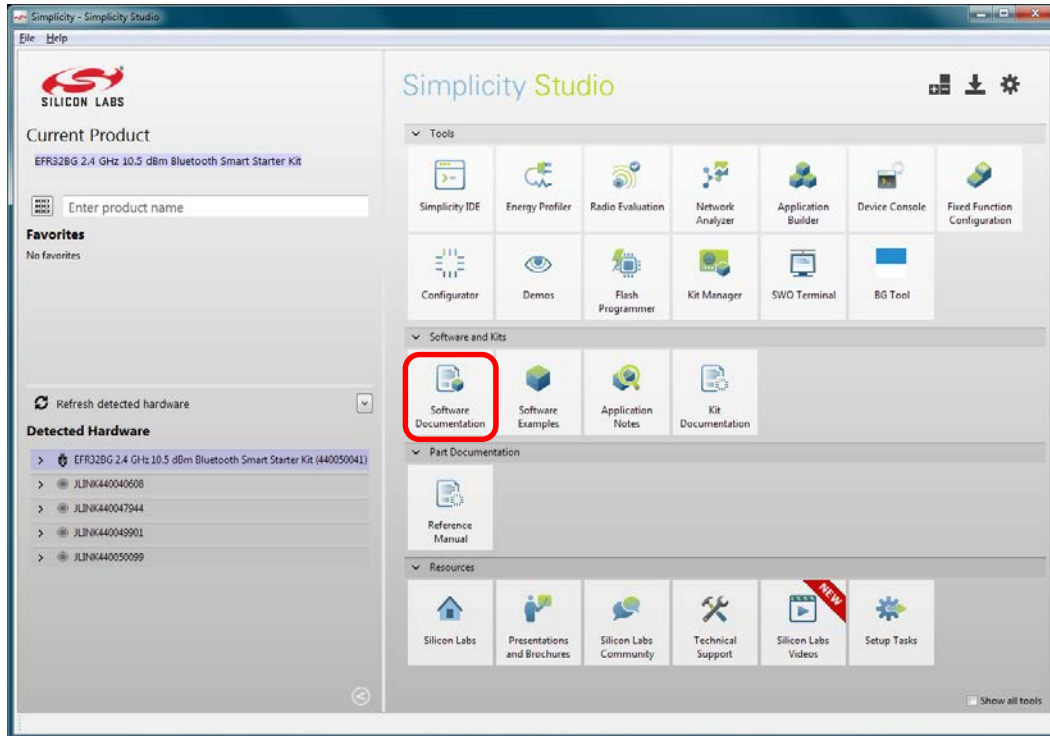


Figure 15. Software Examples Tile

2. In the New Example dialog (Figure 16), enter the kit, part, and SDK. Kit and part will be filled by Simplicity Studio if the hardware was detected and displayed in the Simplicity perspective (Figure 15).

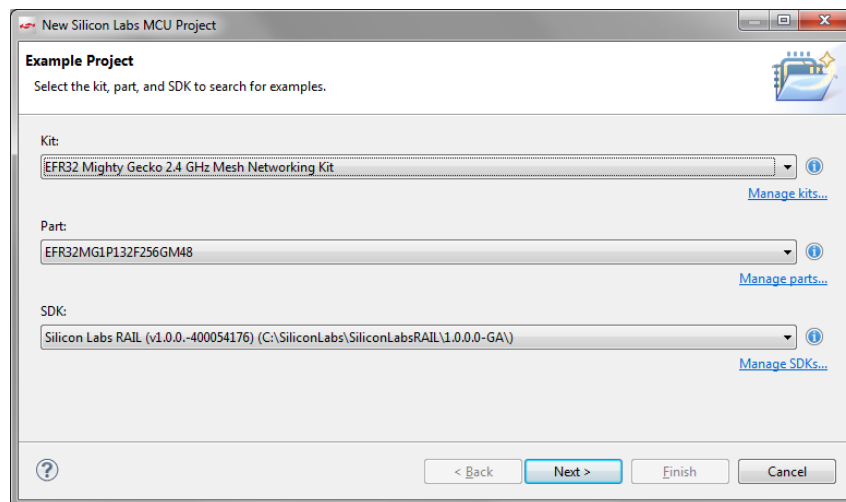


Figure 16. New Example Project Dialog

3. Click **[Next]**.

4. Select an example from the list (Figure 17).

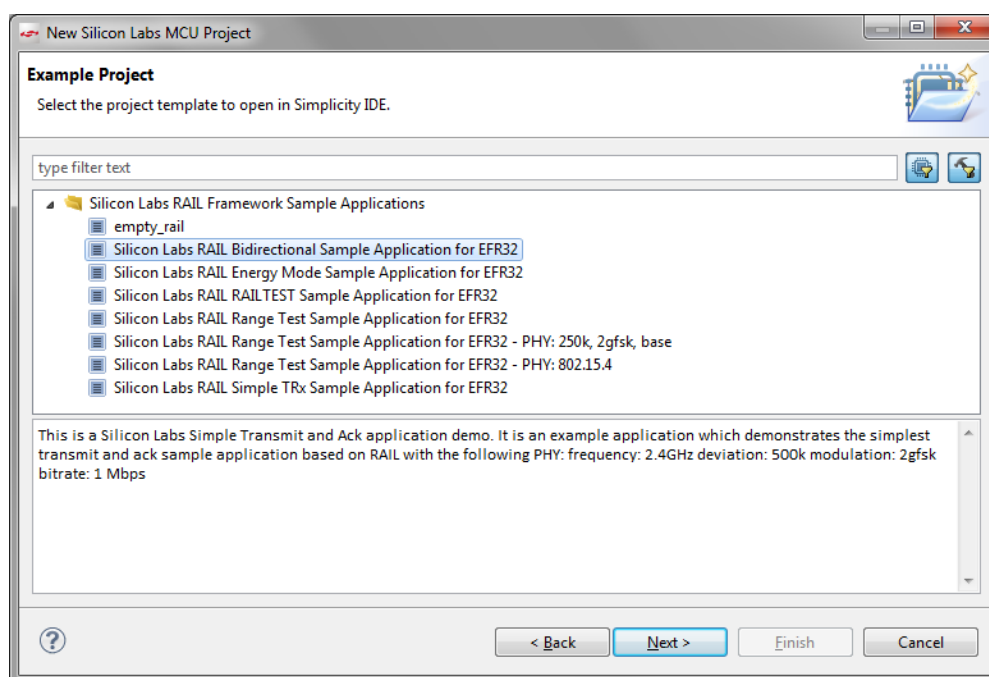


Figure 17. Sample Applications Dialog

5. Click **[Next >]**.
6. In the Project Configuration window (Figure 18), specify a location for your application.

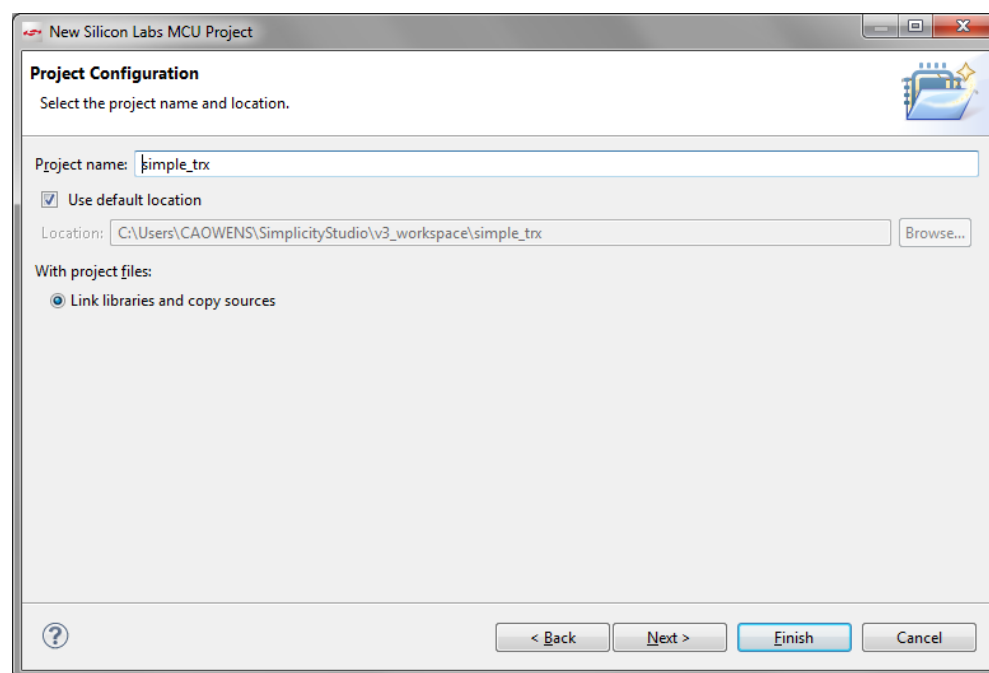


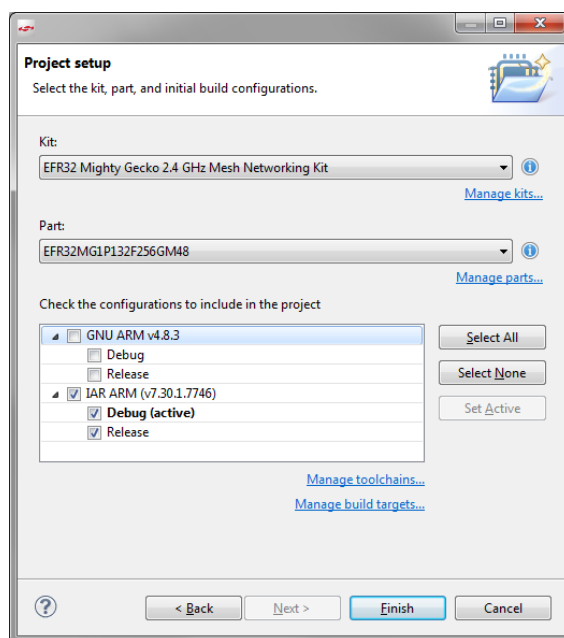
Figure 18. Project Configuration Window

**Note:** The location must be on the same Windows partition as the stack.

7. Specify a name for your application.
8. Click **[Next]**.

- In the Project Setup dialog you can select the platform and build configurations for your application. By default Simplicity Studio loads with the platform of your connected starter kit. You can modify it here if necessary. Click **[Finish]**.

**Note:** Make sure that IAR and GCC toolchains are not both selected. Due to an interaction between the GCC support and the IAR support, performing a 'generate' on a project which includes support for both compilers will break any subsequent use of GCC within the project. To work around this issue, you must create separate projects for IAR and GCC.

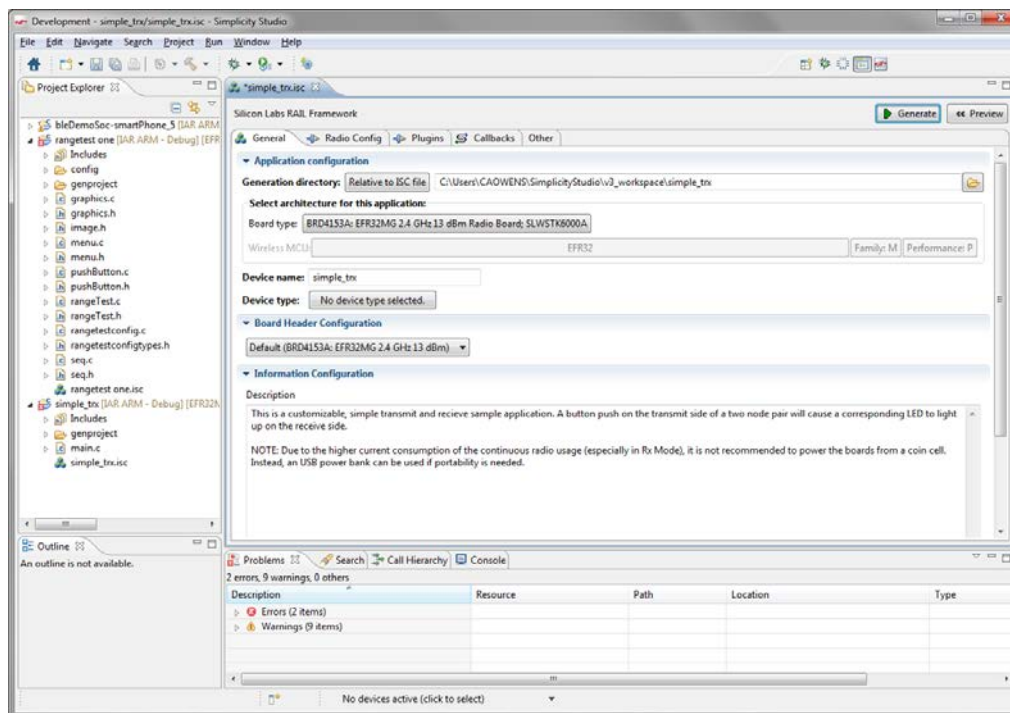


**Figure 19. Project Setup**

**Note:** You must have a Toolchain and Build target selected and configured for the **[Finish]** button to enable. If you do not see the **[Finish]** button enabled, check your Toolchains and Build targets by clicking on the links at the bottom of the dialog.

### 4.1.1 Generate the Application

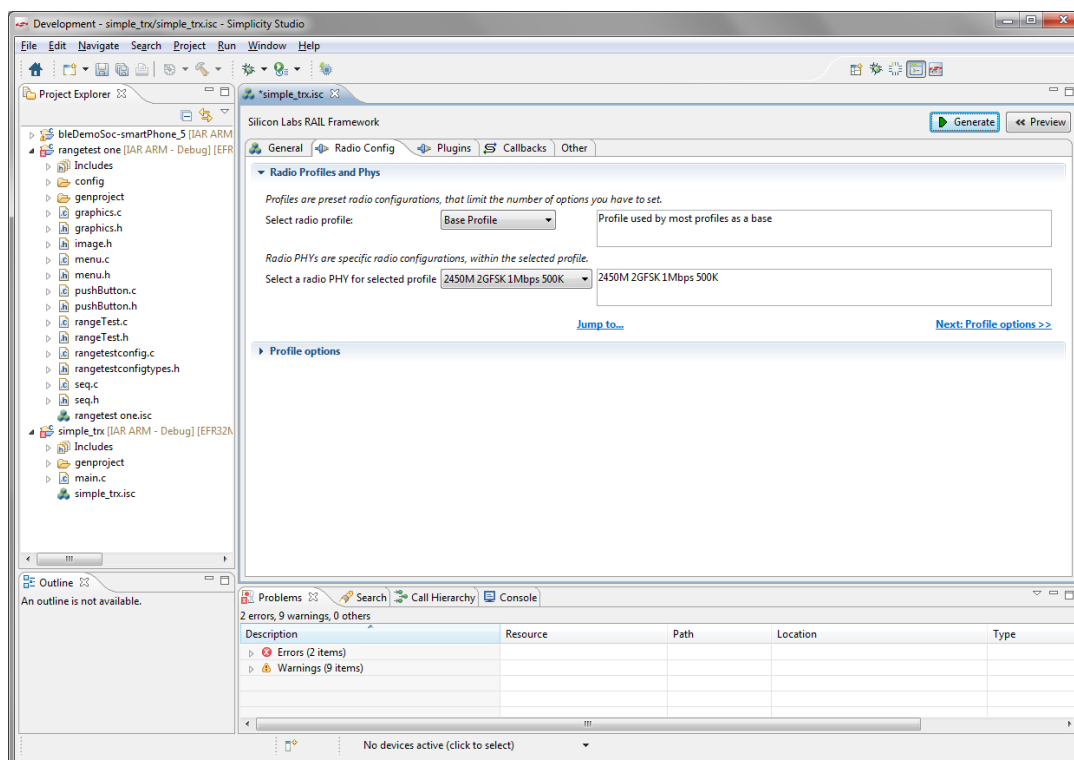
When you finish creating your sample application, an Application Builder General tab opens (Figure 20).



**Figure 20. Application Builder General Tab**

1. If the architecture parameters shown for MCU and Radio, and Board Type are not correct for your target device, change them by clicking on one of the configuration buttons and then selecting from its list. In general, the initial configuration settings for sample applications should be correct. The Device Type setting is not relevant for RAIL applications.
2. The RAIL application framework allows you to modify the PHY configuration for the application. Click the "Radio Config" tab to modify the PHY configuration for your application (Figure 21). This tab 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.



**Figure 21. Radio Config Tab**

For example, if the “Bluetooth LE Profile” is selected the user can either choose the default Bluetooth LE PHY or create a custom PHY configuration within the parameters defined by the Bluetooth LE Profile. See document AN971, *EFR32 Radio Configurator Guide*, for more information on using the radio configurator.



## 4.2 Generating the Application Source Files

1. Once you have completed your Radio Configuration, click **[Generate]** in the upper right corner of the tab in order to generate all of the source and headers associated with your application into your project (Figure 22).

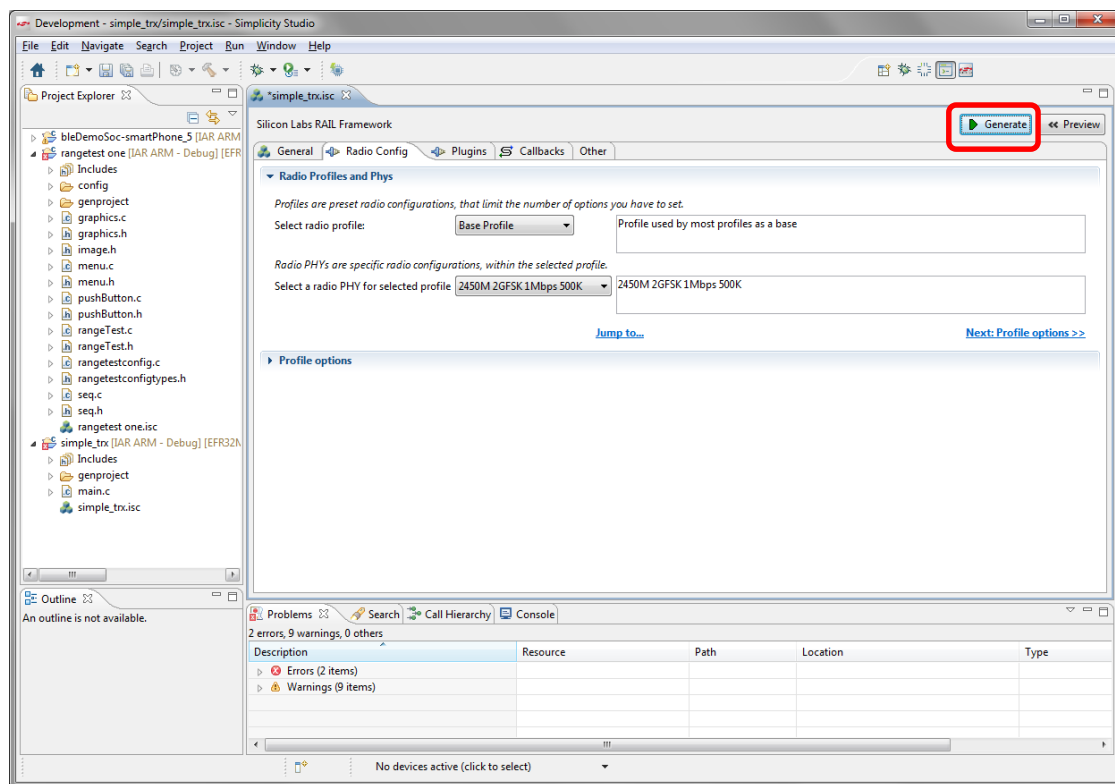


Figure 22. Generate Control

2. An overwrite dialog is presented (Figure 23). **CAUTION:** If you are starting from a sample application, uncheck the box for the callbacks file. The implementation of the application is contained in the callbacks file, and overwriting it will remove the sample functionality.

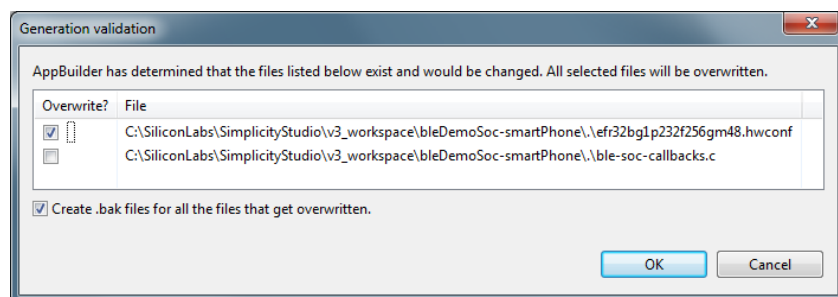


Figure 23. Generation Validation Dialog

- When generation is complete a dialog shows the generated files, one of which has the extension .eww (Figure 24). Click **[OK]**.

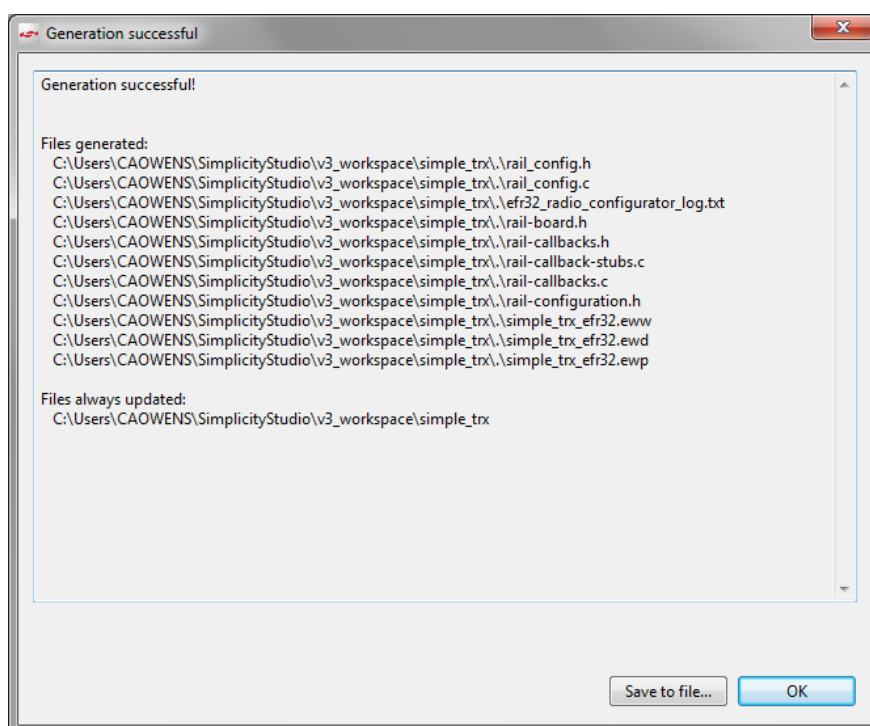


Figure 24 Generation Confirmation Dialog

### 4.3 Compiling and Flashing the Application

You can either automatically compile and flash the application by going to the Debug interface in AppBuilder, or you can manually compile it and then flash it.

#### 4.3.1 Automatically Compile and Flash

- After you click **[OK]** on the Generation Confirmation dialog, the AppBuilder interface returns. Click **[Debug]** to flash the compiled application to the EFR32 (Figure 25).

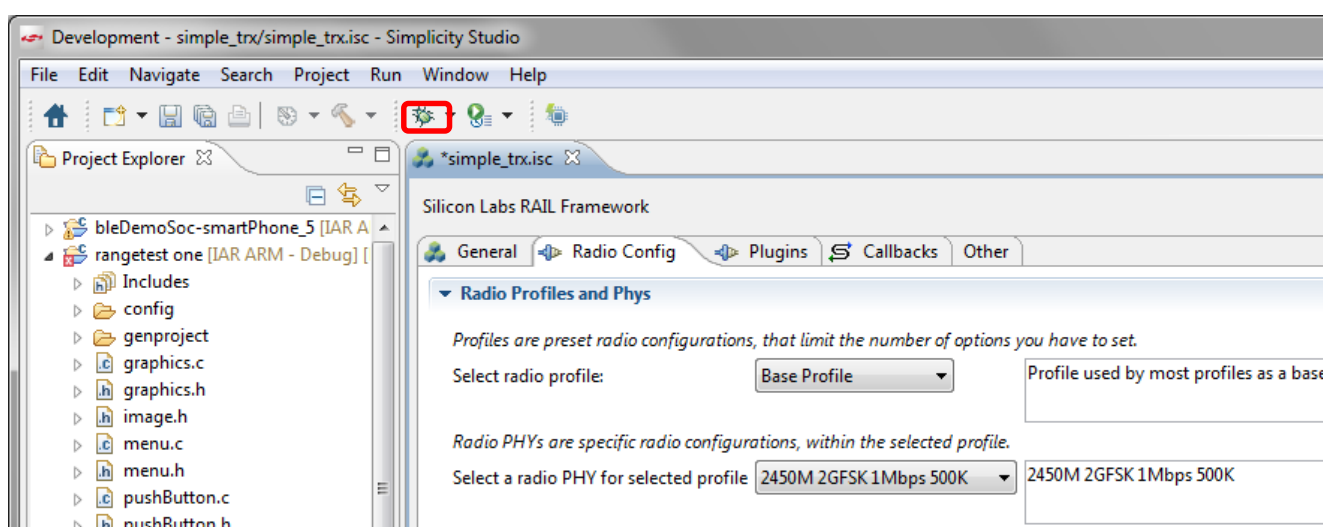


Figure 25. AppBuilder Debug Control

2. A dialog shows the procedure's progress (Figure 26). Wait until flashing has completed and a debug window is displayed.

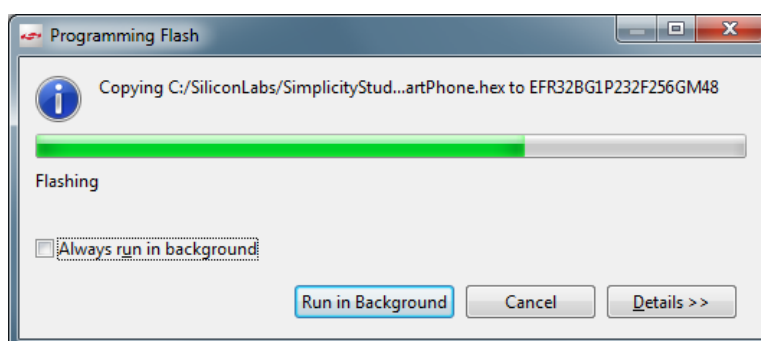


Figure 26. Flash Progress

3. In the Debug window, click **[Resume]** to start the application running on the WSTK (Figure 27).

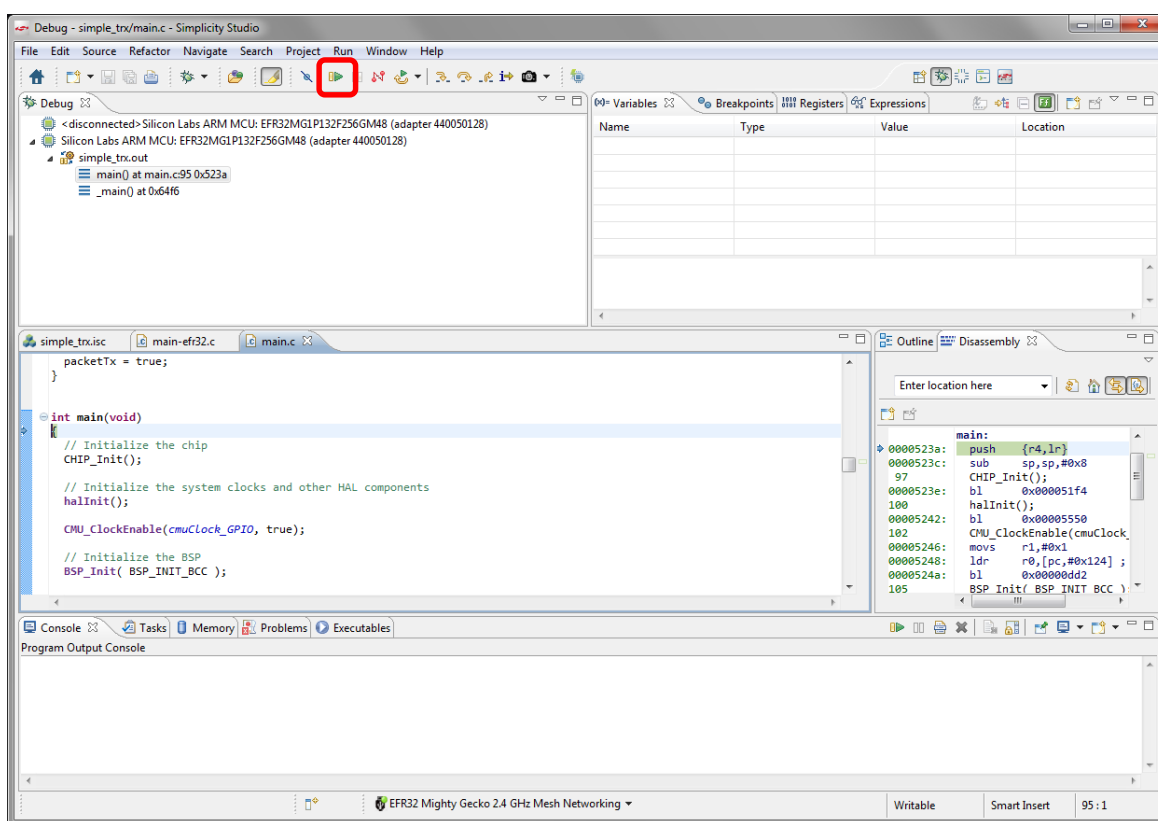



Figure 27. Debug Window

Next to the Resume control are Suspend, Disconnect, Reconnect, and Stepping controls. Click **[Disconnect]** (  ) when you are ready to exit Debug mode.

4. If the sample application you have loaded uses the WSTK LCD, the application UI should display on the WSTK LCD. If the sample application does not use the LCD, the LCD display will remain unchanged. Sample applications like RAILtest, EMode, and Range Test all use the LCD and will provide an interactive interface on the WSTK LCD. Simple sample applications like SimpleTRx and Bidirectional do not use the LCD and only provide interaction through the WSTK buttons PB0 and PB1.

### 4.3.2 Manually Compile and Flash



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

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

The Network Analyzer perspective provides an alternate method of loading any binary image onto your device through the Adapters View.

1. Connect your hardware and make sure it is detected in the Simplicity perspective.
2. Click the Network Analyzer tile on the right-hand side.

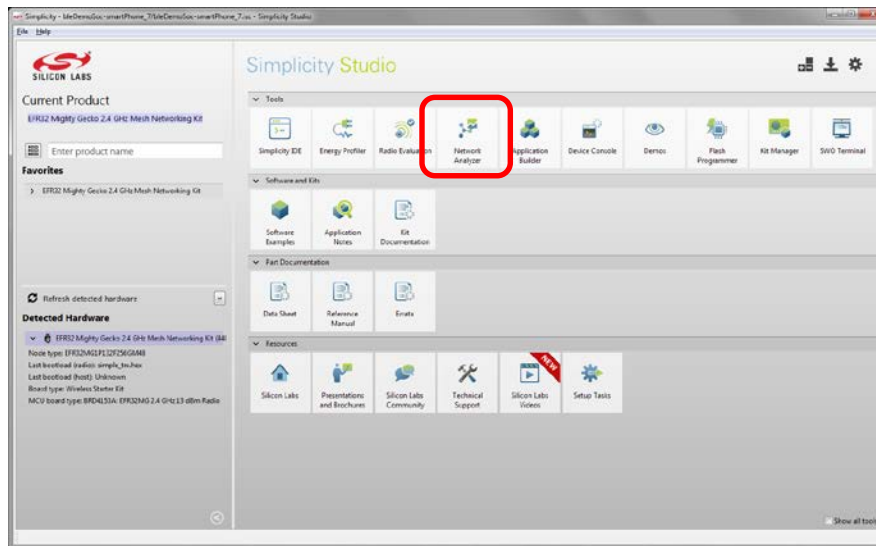


Figure 28. Network Analyzer Tile

3. The Network Analyzer perspective opens. This perspective includes an Adapters View on the left-hand side of the screen. Right-click on your device of choice, and select **Connect**.
4. Right-click on the device in the Adapters view again and select **Flash / Upload**. The Select Binary Image dialog is displayed (example shown in Figure 29).

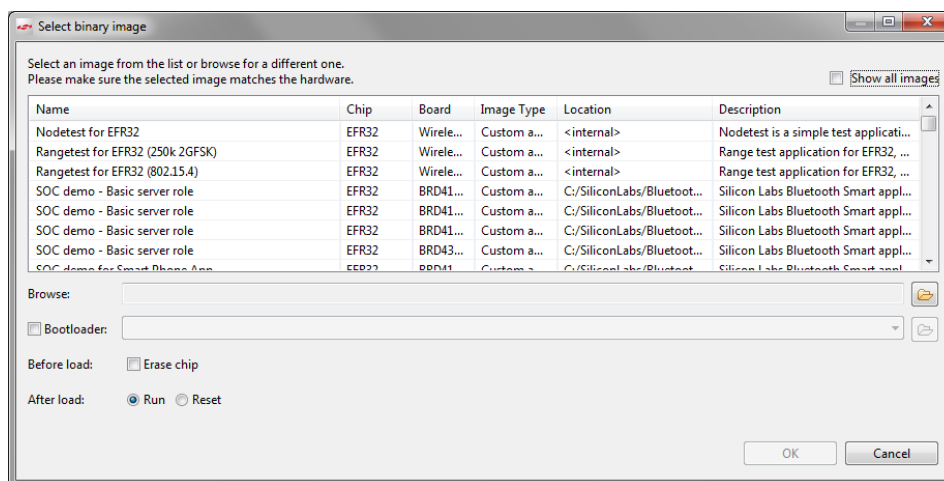


Figure 29. Select Binary Image

5. Navigate to the .s37, .bin or .hex image you wish to upload.
6. Check **Erase Chip**, to make sure that any previous bootloader or other non-volatile data is erased before your new image is uploaded.
7. 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**.
8. Click **[OK]**. You will be notified once the upload is complete.

## 5 Interacting with your Sample Application

Some of the sample applications included with the RAIL SDK allow you to interact with them over the Command Line Interface (CLI). If the sample application supports this you can interact with your loaded sample application through your development environment's Console interface using a CLI (command line interpreter). Many sample applications will provide a list of command options by typing "help" on the command line interface. Of the sample applications currently included in the RAIL SDK, only the RAILtest sample includes the CLI library and thus provides interaction over the Command Line Interface. If the sample application implements a CLI it will be available through Simplicity Studio's Console interface.

To launch the Console interface, click the Device Console tile in the Simplicity perspective. Alternatively, in the Network Analyzer perspective right-click on your device in the Adapters View. Choose **Connect** (if you are not already connected) and then **Launch Console** (Figure 30). To get a prompt on the Console Serial 1 tab, press Enter.

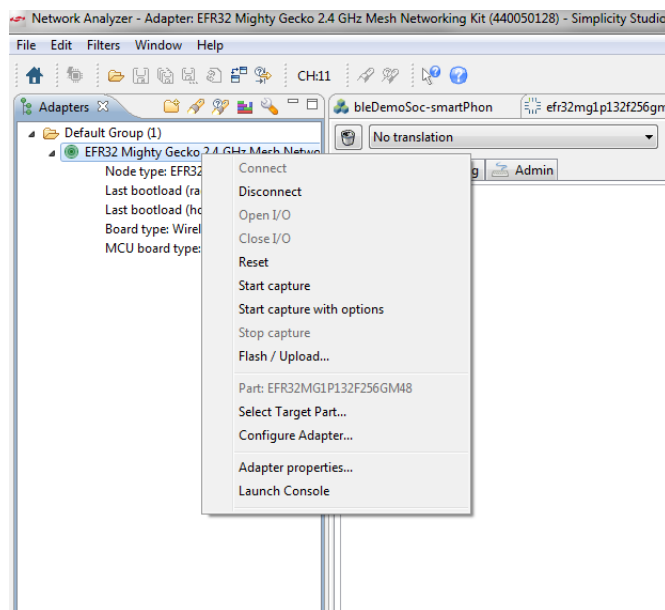
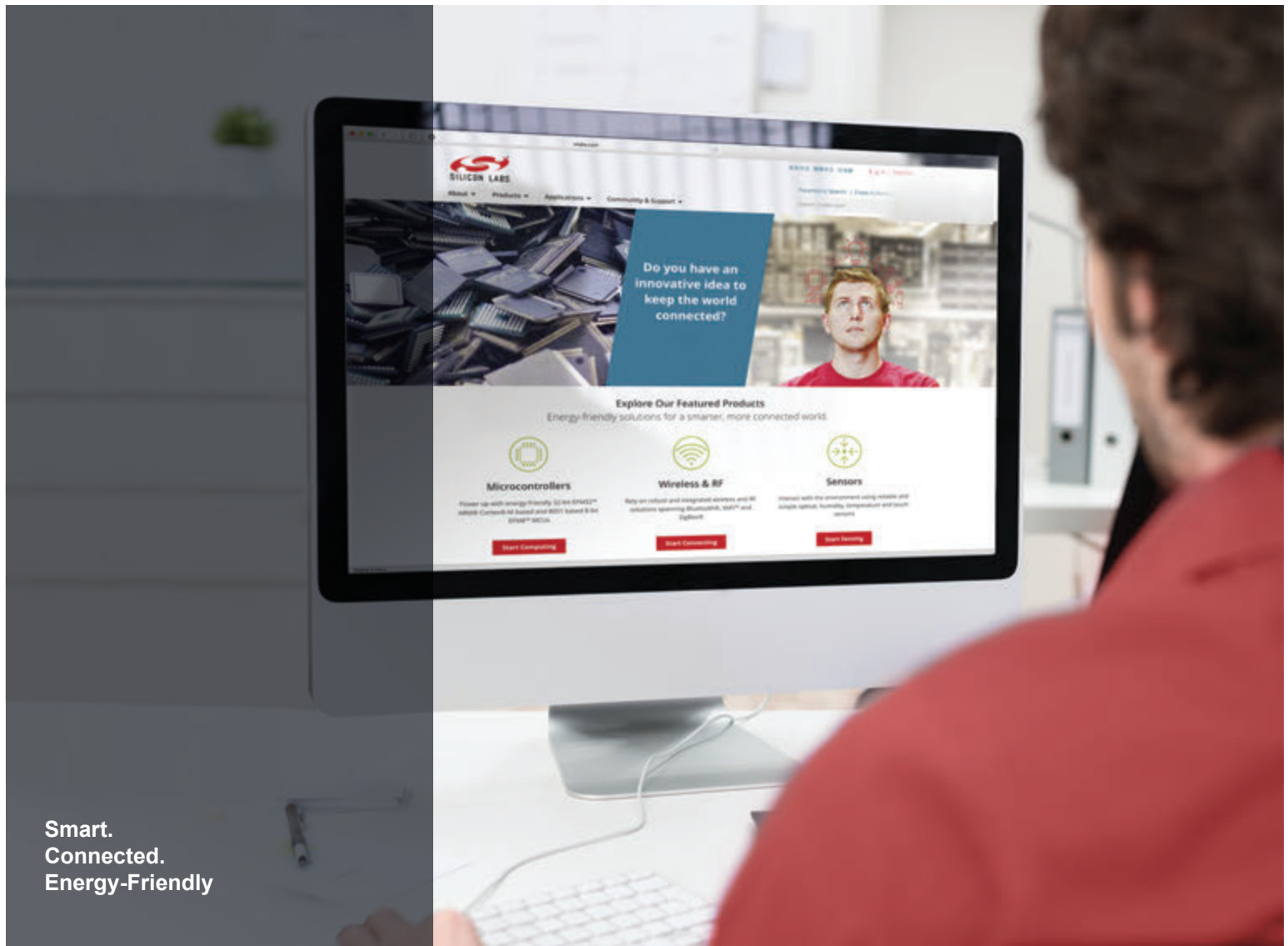


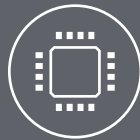
Figure 30. Launch Console



Smart.  
Connected.  
Energy-Friendly



**Products**  
[www.silabs.com/products](http://www.silabs.com/products)



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



**Support and Community**  
[community.silabs.com](http://community.silabs.com)

#### Disclaimer

Silicon Laboratories intends to provide customers with the latest, accurate, and in-depth documentation of all peripherals and modules available for system and software implementers using or intending to use the Silicon Laboratories products. Characterization data, available modules and peripherals, memory sizes and memory addresses refer to each specific device, and "Typical" parameters provided can and do vary in different applications. Application examples described herein are for illustrative purposes only. Silicon Laboratories reserves the right to make changes without further notice and limitation to product information, specifications, and descriptions herein, and does not give warranties as to the accuracy or completeness of the included information. Silicon Laboratories shall have no liability for the consequences of use of the information supplied herein. This document does not imply or express copyright licenses granted hereunder to design or fabricate any integrated circuits. The products are not designed or authorized to be used within any Life Support System without the specific written consent of Silicon Laboratories. A "Life Support System" is any product or system intended to support or sustain life and/or health, which, if it fails, can be reasonably expected to result in significant personal injury or death. Silicon Laboratories products are not designed or authorized for military applications. Silicon Laboratories products shall under no circumstances be used in weapons of mass destruction including (but not limited to) nuclear, biological or chemical weapons, or missiles capable of delivering such weapons.

#### Trademark Information

Silicon Laboratories Inc.®, Silicon Laboratories®, Silicon Labs®, SiLabs® and the Silicon Labs logo®, Bluegiga®, Bluegiga Logo®, Clockbuilder®, CMEEMS®, DSPLL®, EFM®, EFM32®, EFR®, Ember®, Energy Micro, Energy Micro logo and combinations thereof, "the world's most energy friendly microcontrollers", Ember®, EZLink®, EZRadio®, EZRadioPRO®, Gecko®, ISOModem®, Precision32®, ProSLIC®, Simplicity Studio®, SiPHY®, Telegesis, the Telegesis Logo®, USBXpress® and others are trademarks or registered trademarks of Silicon Laboratories Inc. ARM, CORTEX, Cortex-M3 and THUMB are trademarks or registered trademarks of ARM Holdings. Keil is a registered trademark of ARM Limited. All other products or brand names mentioned herein are trademarks of their respective holders.



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

<http://www.silabs.com>