

AN888: EZR32 Quick Start Guide

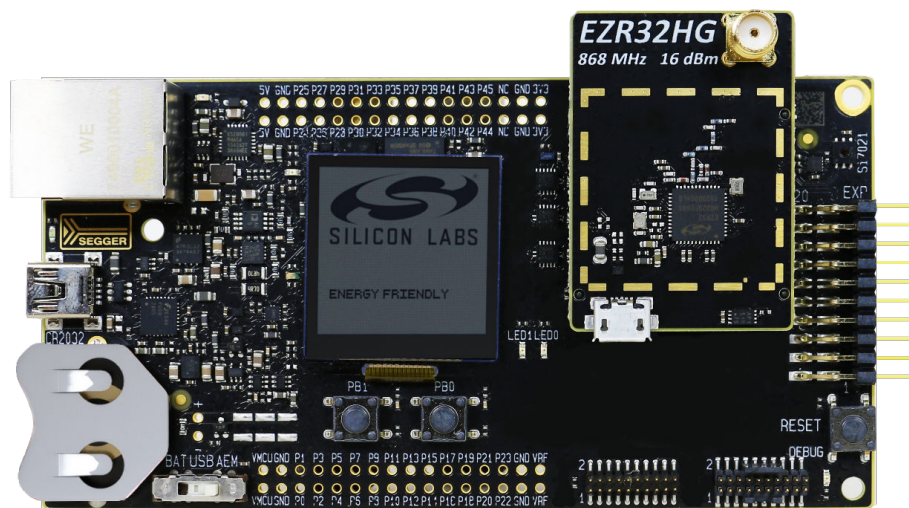


The EZR32LG, EZR32WG, and EZR32HG Wireless MCUs deliver a high-performance, low-energy wireless solution integrated into a small form factor package. By combining a high-performance Silicon Labs EZRadio or EZRadioPRO sub-GHz RF transceiver with an energy-efficient 32-bit MCU, the EZR32LG and EZR32WG family provides designers the ultimate in flexibility with a family of pin-compatible devices that scale with 64/128/256 kB of flash and support transceivers. The EZR32HG family provides the high energy efficiency of the Cortex-M0+ core in a smaller form factor with 32/64 kB flash. The ultra-low power operating modes and fast wake-up times of the Silicon Labs energy friendly 32-bit MCUs, combined with the low transmit and receive power consumption of the sub-GHz radio, result in a solution optimized for battery-powered applications.

The EZR32LG, EZR32WG, and EZR32HG Wireless Starter Kit (WSTK) is a low-cost, fully featured starter kit that allows designers to evaluate the performance and features of the EZR32LG/WG/HG device family. The kit will provide all hardware and software tools needed to develop and debug applications using the EZR32LG, WG or HG low-energy Wireless MCUs. Hardware tools include an on-board debugger, advanced energy monitoring, and integrated packet trace, which are all available either over USB or Ethernet interfaces. The kit is fully supported by Silicon Labs' software libraries and the kit board support package (BSP).

KEY FEATURES

- How to install EZR32 SDK in Simplicity Studio
- Shows where to find documentation
- How to use the SimpleTRX sample application
- How to modify the Radio configuration
- How to use radio GPIOs for debugging



1. Getting Started

1.1 Hardware

- BRD4001A — Wireless Starter Kit Mainboard
- One set of radio boards from the table below.

Table 1.1. Radio Board Options

Kit ID	Radio Board ID	Part Number	Radio Part Number	Matching	Frequency	PA Level
SLWSTK6200A	BRD4502A	EZR32LG330F256G60G	Si4460revC	Direct Tie	868 MHz	+13 dBm
SLWSTK6201A	BRD4502B	EZR32LG330F256G55G	Si4455revC	Direct Tie	434 MHz	+10 dBm
SLWSTK6202A	BRD4503A	EZR32LG330F256G63G	Si4463revC	Switched	915 MHz	+20 dBm
SLWSTK6220A	BRD4502C	EZR32WG330F256G60G	Si4460revC	Direct Tie	868 MHz	+13 dBm
SLWSTK6221A	BRD4502D	EZR32WG330F256G55G	Si4455revC	Direct Tie	434 MHz	+10 dBm
SLWSTK6222A	BRD4503B	EZR32WG330F256G63G	Si4463revC	Switched	915 MHz	+20 dBm
SLWSTK6223A	BRD4505B	EZR32WG330F256G63G	Si4463revC	Switched	490 MHz	+20 dBm
SLWSTK6224A	BRD4504B	EZR32WG330F256G63G	Si4463revC	Direct Tie	169 MHz	+20 dBm
SLWSTK6240A	BRD4546A	EZR32HG320F64G68G	Si4468revA	Direct Tie	868 MHz	+16 dBm
SLWSTK6241A	BRD4542B	EZR32HG320F64G55G	Si4455revC	Direct Tie	434 MHz	+10 dBm
SLWSTK6242A	BRD4543B	EZR32HG320F64G68G	Si4468revA	Switched	915 MHz	+20 dBm
SLWSTK6243A	BRD4545A	EZR32HG320F64G63G	Si4463revC	Switched	490 MHz	+20 dBm
SLWSTK6244A	BRD4544A	EZR32HG320F64G63G	Si4463revC	Direct Tie	169 MHz	+20 dBm

1.2 Software

Required software:

- Simplicity Studio 3.0 + EFM and Wireless Packages.
- WSTK Firmware Version 2.4 or later. (Update available within Simplicity Studio through the Kit Manager.)

Software Dependencies:

- Simplicity Studio v3

1.3 Installing All Required Software

Install all of the software listed above. All of the EZR32 development kit software and prerequisite software must be installed on your machine for the development kit software to work correctly. The order in which the software is installed is not important, but make sure you have it all installed before you launch Simplicity Studio and begin working on your first project.

1.4 Connecting Your Hardware

Once all required software is installed, connect the EZR32LG/WG/HG development kit hardware to your PC. It is simply connected using a USB cable, as shown below:

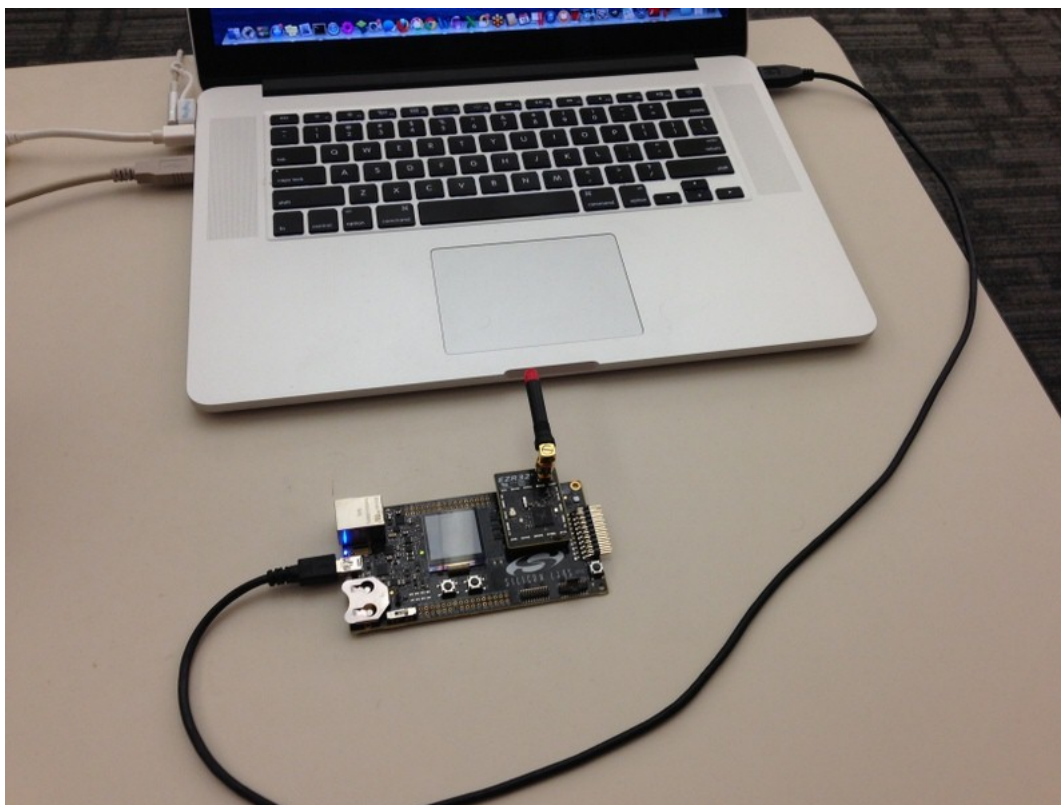


Figure 1.1. WSTK Board Connected to PC with USB

Note: Set the switch on the WSTK to AEM position to enable programming of the MCU.

2. Running Simplicity Studio for the First Time

2.1 Navigation in Simplicity Studio

Simplicity Studio is built on the Eclipse platform. As such, it is broken up into different “perspectives”, with each perspective allowing access to a specific set of functionalities. Simplicity Studio starts in the “Simplicity Perspective,” sometimes referred to as the “Home Screen.”

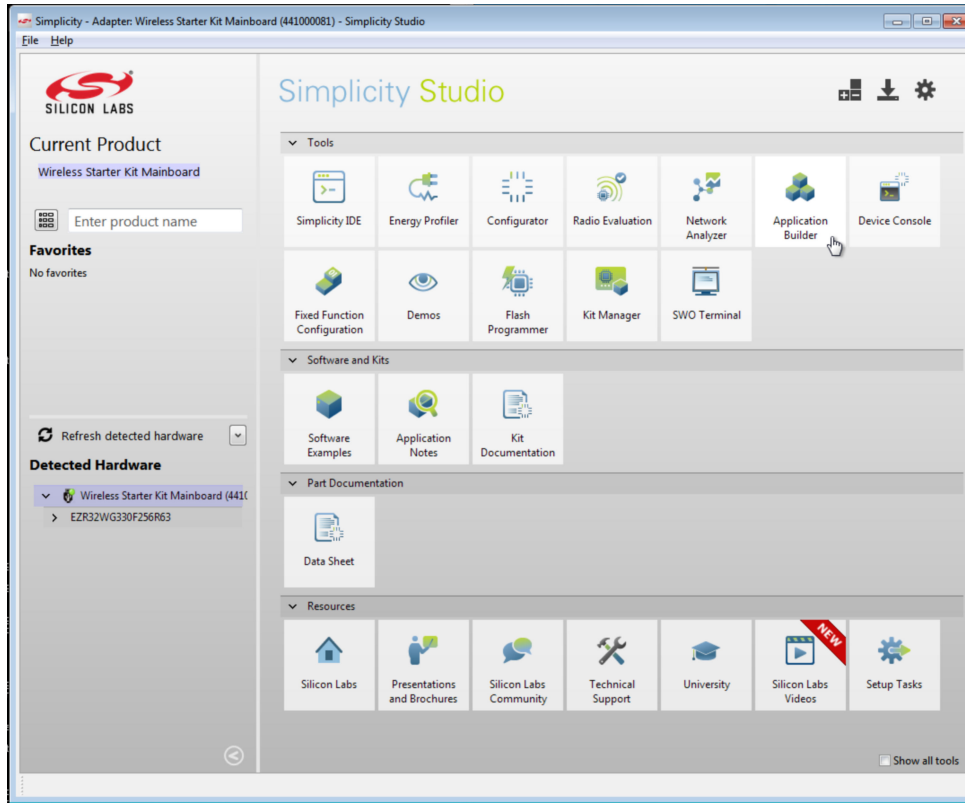


Figure 2.1. Simplicity Studio's Simplicity Perspective

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 pieces of functionality within Simplicity Studio. When you are in a different perspective, you will see smaller tile icons in the top right-hand corner of your screen. You can navigate back 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.

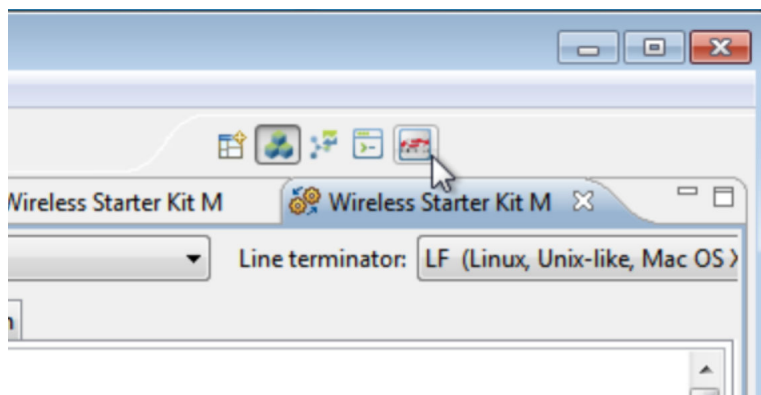


Figure 2.2. Navigation Tile Icons

2.2 Downloading Packages for Simplicity Studio

By default, Simplicity Studio comes with some core components necessary for development. The real substance of the Simplicity Studio application is provided in the “Packages” available from the Simplicity Studio update site. Before you begin working with Simplicity Studio, make sure that you have all packages installed that you need for your development platform. In particular, for the EZR32 family of products, you will need both the “Wireless” and “EFM Product” packages.

To download packages for Simplicity Studio, click on the **[Package Management]** icon in the top right-hand corner of the Simplicity Perspective.

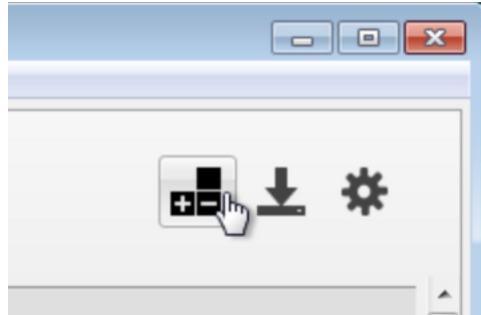


Figure 2.3. Packages Management

The Add/Remove Packages window opens and will guide you through the process of updating Simplicity Studio’s packages. You will need both the wireless package and the EFM SDK package for development on the EZR32 product.

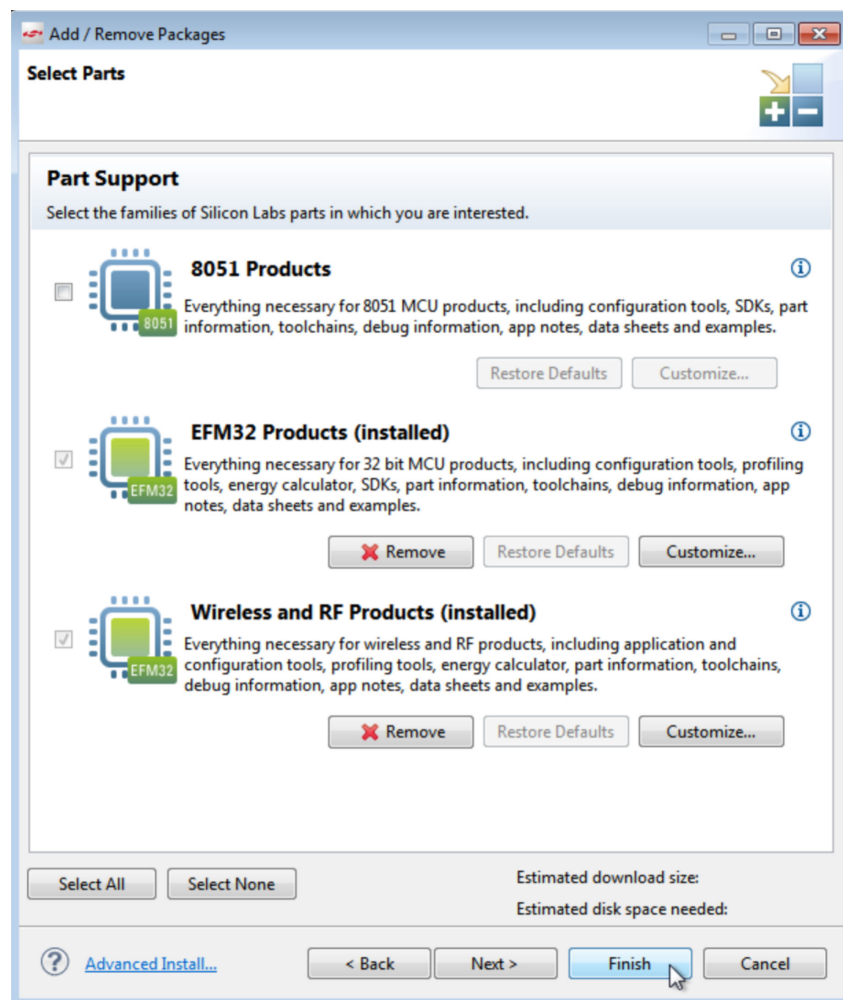


Figure 2.4. Add/Remove Packages Window

2.3 Updating Simplicity Studio

In addition to downloading packages, you can also update the Simplicity Studio core software at any time by clicking the **[Update Software]** icon in the top right-hand corner of the Simplicity Perspective.



Figure 2.5. Updating Simplicity Studio

2.4 Discovering Your Development Kit Hardware in Simplicity Studio

When you launch Simplicity Studio, it should automatically discover your USB connected Wireless Starter Kit (WSTK) hardware and display it in the Simplicity Perspective, as shown below:

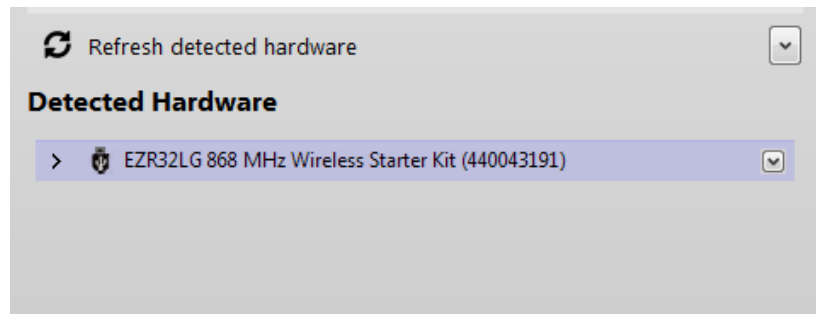


Figure 2.6. Detected Hardware

The WSTK board will come up with an identifier that includes the serial number of the WSTK connected. The serial number displayed in Simplicity Studio should match the number displayed on the WSTK board's LCD.

2.5 Upgrading the WSTK Boards to the Latest Firmware

Once you have connected the WSTK boards to the computer, make sure that the boards have the latest WSTK firmware loaded. This is important, as it ensures that the WSTK boards will correctly communicate with the Simplicity Studio software.

Simplicity Studio includes a bundled Kit Manager that can be used to upgrade your WSTK boards. To access the Kit Manager within Simplicity Studio, click on the WSTK board that needs to be upgraded so that it shows as selected, and then click the **[Kit Manager]** tile on the right-hand side of the Simplicity Perspective.

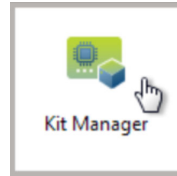


Figure 2.7. Kit Manager

Clicking on the Kit Manager tile will open the Kit Manager dialog window, through which you can load a new firmware image onto your WSTK main board. The Kit Manager has a bundled version of the latest WSTK software that it will ask to load onto your device. (If your device doesn't already have the latest WSTK software.)

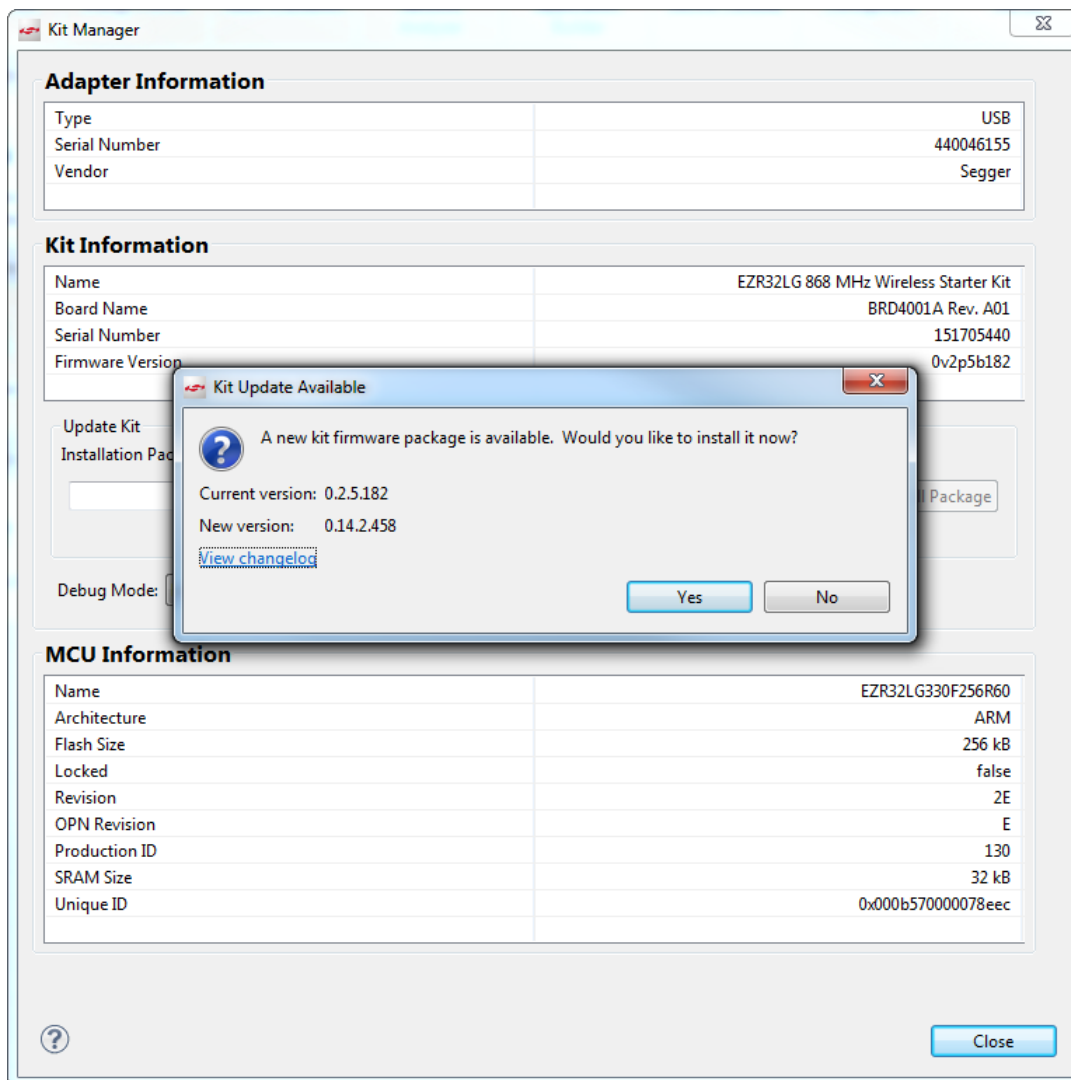


Figure 2.8. Kit Manager Dialog

3. Documentation

The EZR32 parts are combined from an EFM32 MCU and an EZRadio or EZRadioPro radio, so there are two sets of documentation.

3.1 MCU Related Documentation

MCU-related documentation is installed with Simplicity Studio. Kit documentation, data sheets, and reference manuals can be accessed by selecting the following tiles:

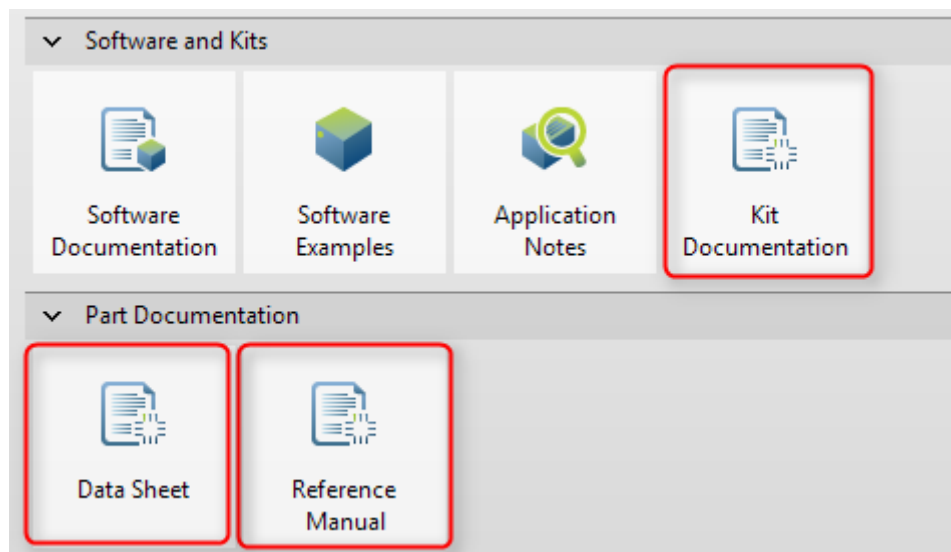


Figure 3.1. MCU Related Documentation

The connection between the MCU and the Radio is described in the data sheet under the chapter “EZRadio and EZRadioPro Transceivers”.

3.2 Radio Related Documentation

There's no specific radio documentation for EZR32, however, most of the documentation for EZRadio or EZRadioPro complies with these parts. This documentation can be found on our website under <http://www.silabs.com/EZRadio> and <http://www.silabs.com/EZRadioPro>.

3.3 Software Documentation

Software documentation is accessible through the Software Documentation tile:

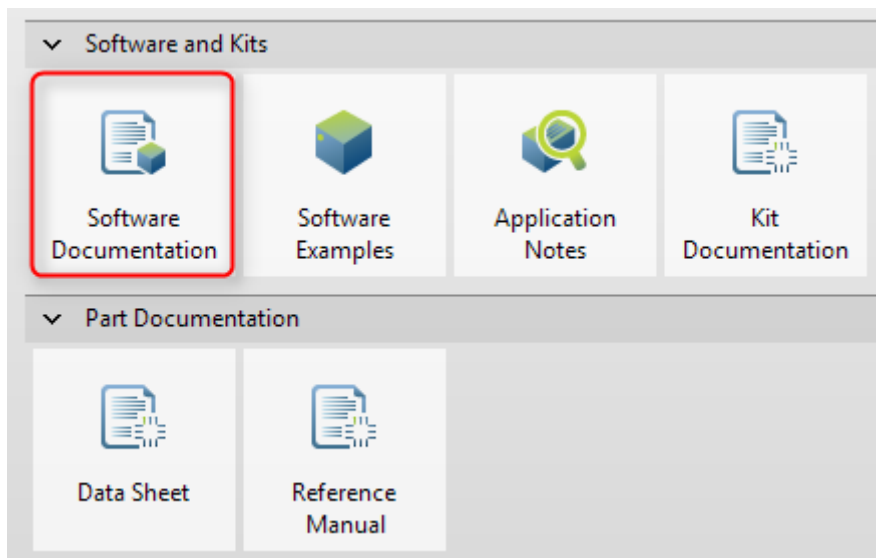


Figure 3.2. Software Documentation

The documentation of the open source radio driver is located under any EZR32, EnergyAware Driver, EZRADIODRV:

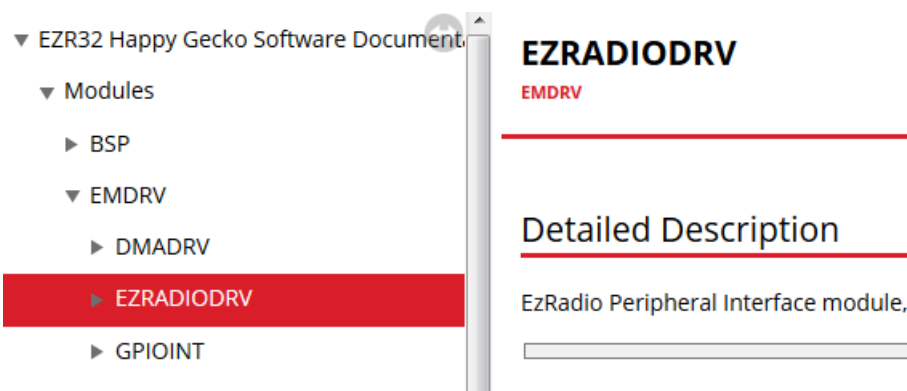


Figure 3.3. Open Source Radio Driver Documentation

This driver adds a user-friendly interface layer over the EZRadio/EZRadioPro SPI API. However, not every feature is implemented, see “EZRadioPRO API Rev C2A-A2A” and “EZRadio API Rev C2A” under <http://www.silabs.com/EZRadioPro> and <http://www.silabs.com/EZRadio> for the full SPI API.

4. The Simple TRX Sample Application

Simplicity Studio comes bundled with an SDK for the EZR32 as part of the EFM packaged sample applications. One of the sample applications, “Simple TRX” can be used to test and work with the radio on the EZR32. This is a simple application that can be programmed on two devices and send packets from one to the other by pressing the buttons on the WSTK.

To access the samples available for the EZR32, first select your USB connected device on the left-hand side under discovered devices in the Simplicity Perspective. With the device selected, click on the **[Software Examples]** tile.

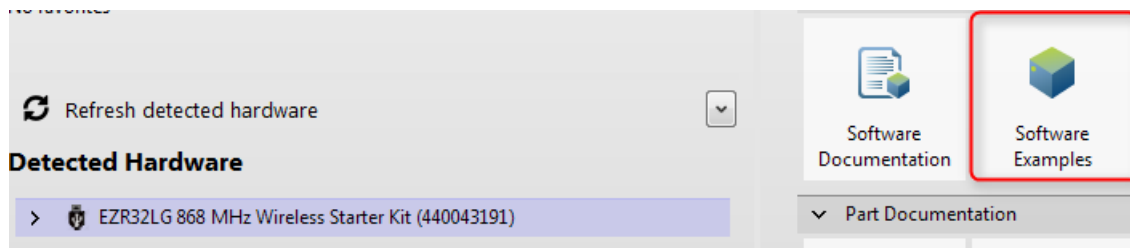


Figure 4.1. Software Example Tiles

This will launch the Example Project dialog window. If you have downloaded the EFM32 SDK, you should see it as an option in the Example Project window.

The Example Project dialog allows you to find the available sample applications for your connected part.

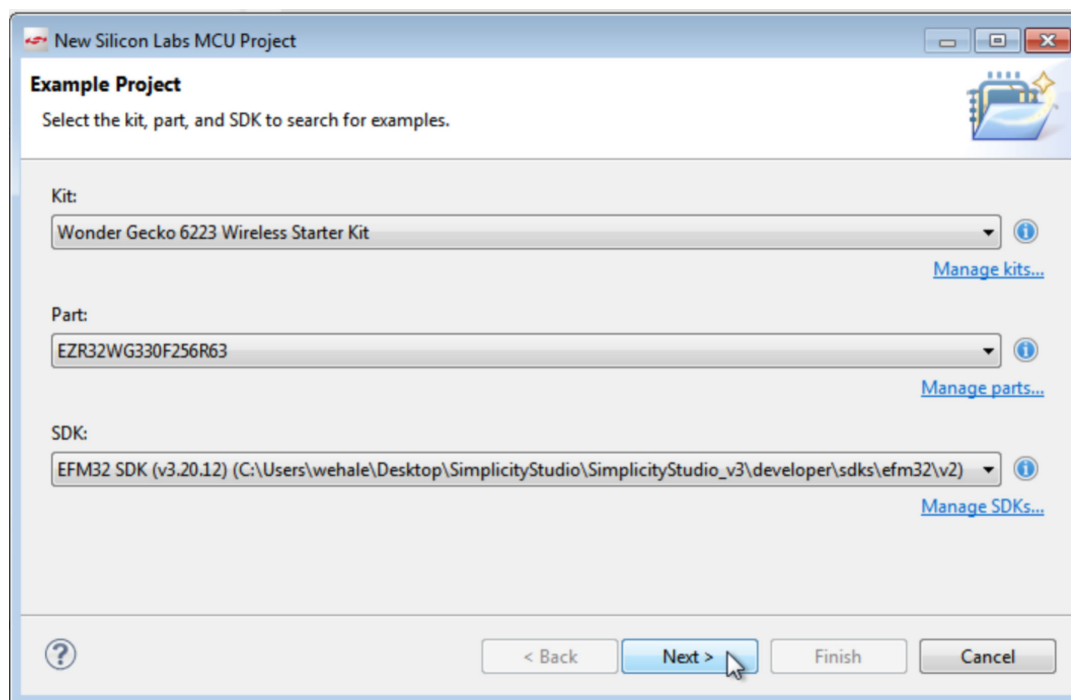


Figure 4.2. Example Project Window

Note: Make sure you have a Kit selected based on the type of EZR32 you are using. For instance, if you are using a Wonder Gecko EZR32, you will want to select one of the Wonder Gecko Wireless starter kits. The sample applications are associated with a specific kit, so a real kit must be selected in order for the sample applications to appear as options.

Click the **[Next]** button. In the next screen you should see all of the sample applications available for your part. Choose the **[simple_trx]** sample under **[Radio Examples]** and click **[Next]**.

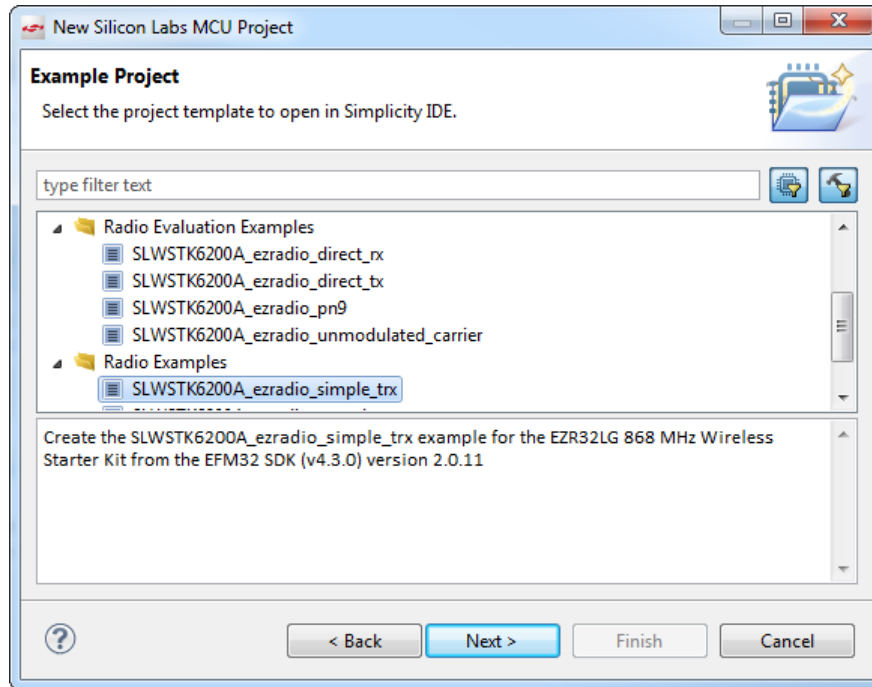


Figure 4.3. Select Simple TRX Sample Application

The next screen will allow you to select a name for your application. Choose the name you want and select **[Next]** again. The next screen allows you to configure your Build Configurations for this project. For the Simple TRX sample you have the option of building with IAR (if you have IAR installed) or GCC. Select all the build configuration options you want and click **[Finish]**.

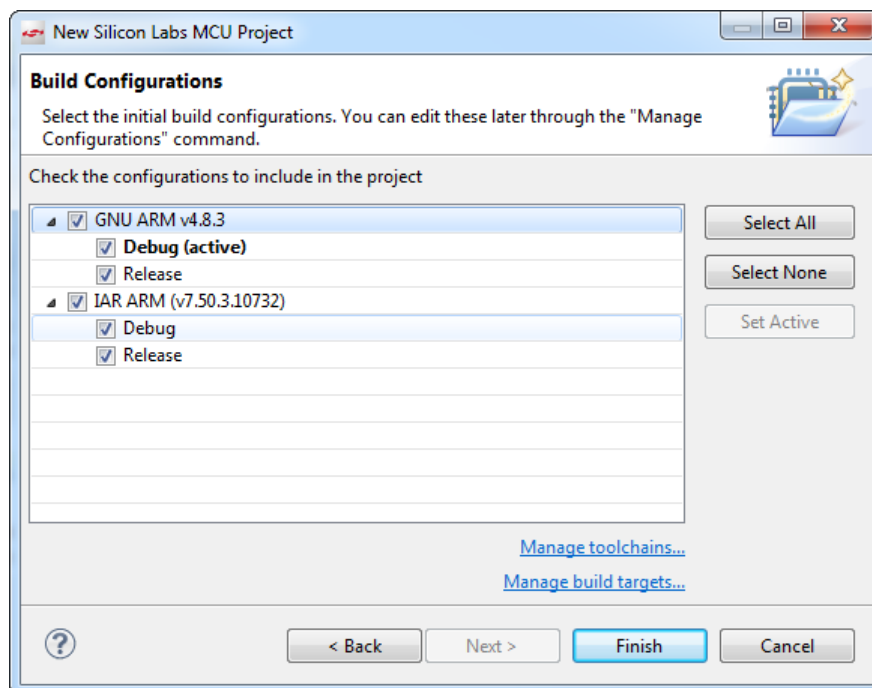


Figure 4.4. Build Configurations

The Simple TRX sample is developed in the Development Perspective. Clicking on the **[Finish]** button will automatically launch the Development Perspective in which you can edit and build your application. For more information on how to build, load, and interact with your application, see [5. Building and Uploading Your Sample Application](#).

For more information on how to use the Simple Rx/Tx sample application, see the `readme.txt` included with the sample and available in your generated application.

All sample applications include a `readme.txt` file that explains the sample and its operation.

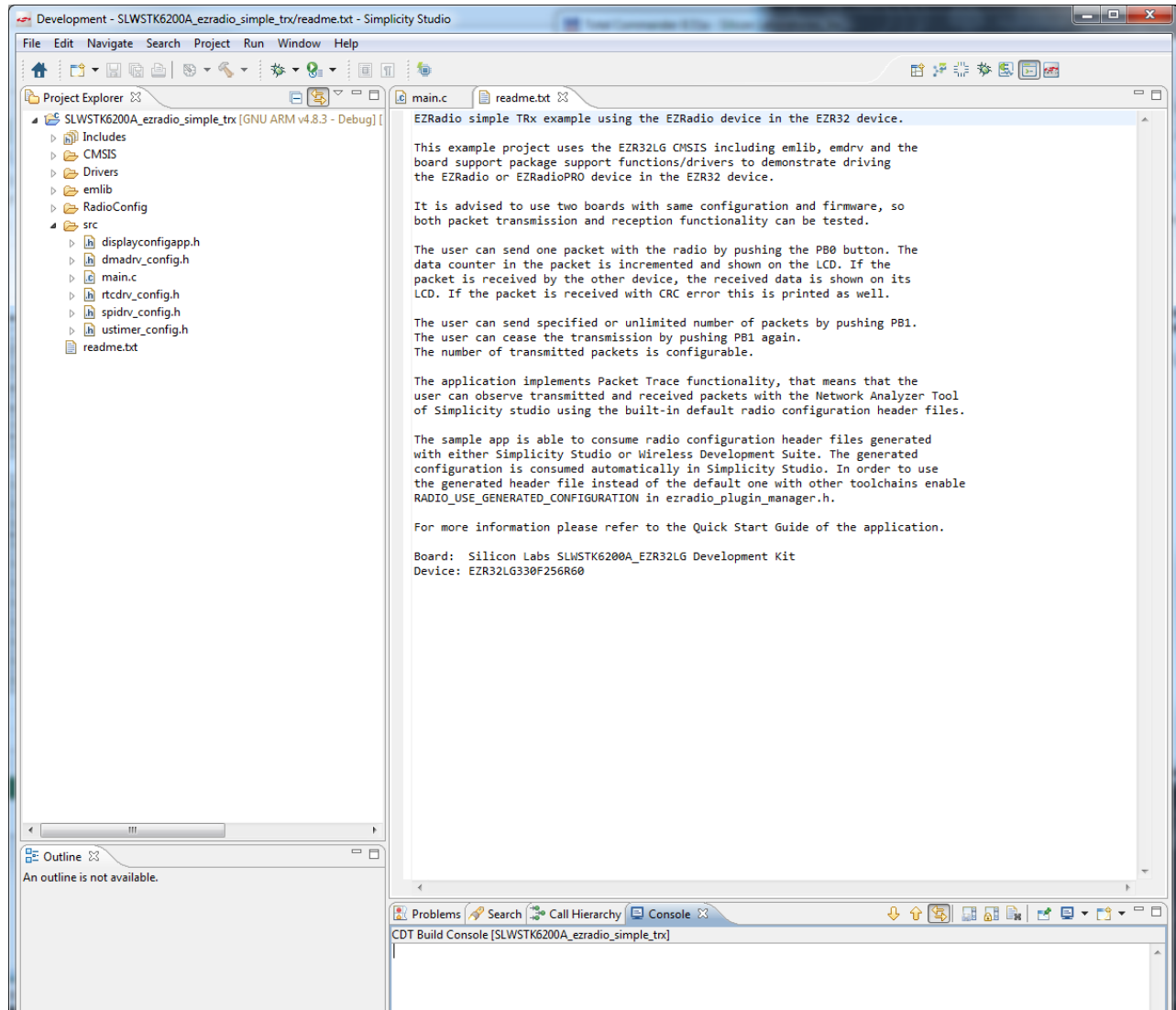


Figure 4.5. Readme.txt File

For more information on how to load and operate the Simple Rx/Tx sample application, continue to Sections [5.1 Building](#) and [5.2 Uploading](#).

5. Building and Uploading Your Sample Application

5.1 Building

Once your application is generated, you have the ability to compile it using Simplicity Studio. In order to compile your application, simply click on the **[Build]** icon in the top tool bar.

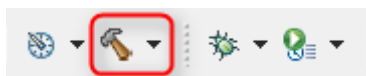


Figure 5.1. Build Button

Your sample application will compile based on its build configuration. You may 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]**

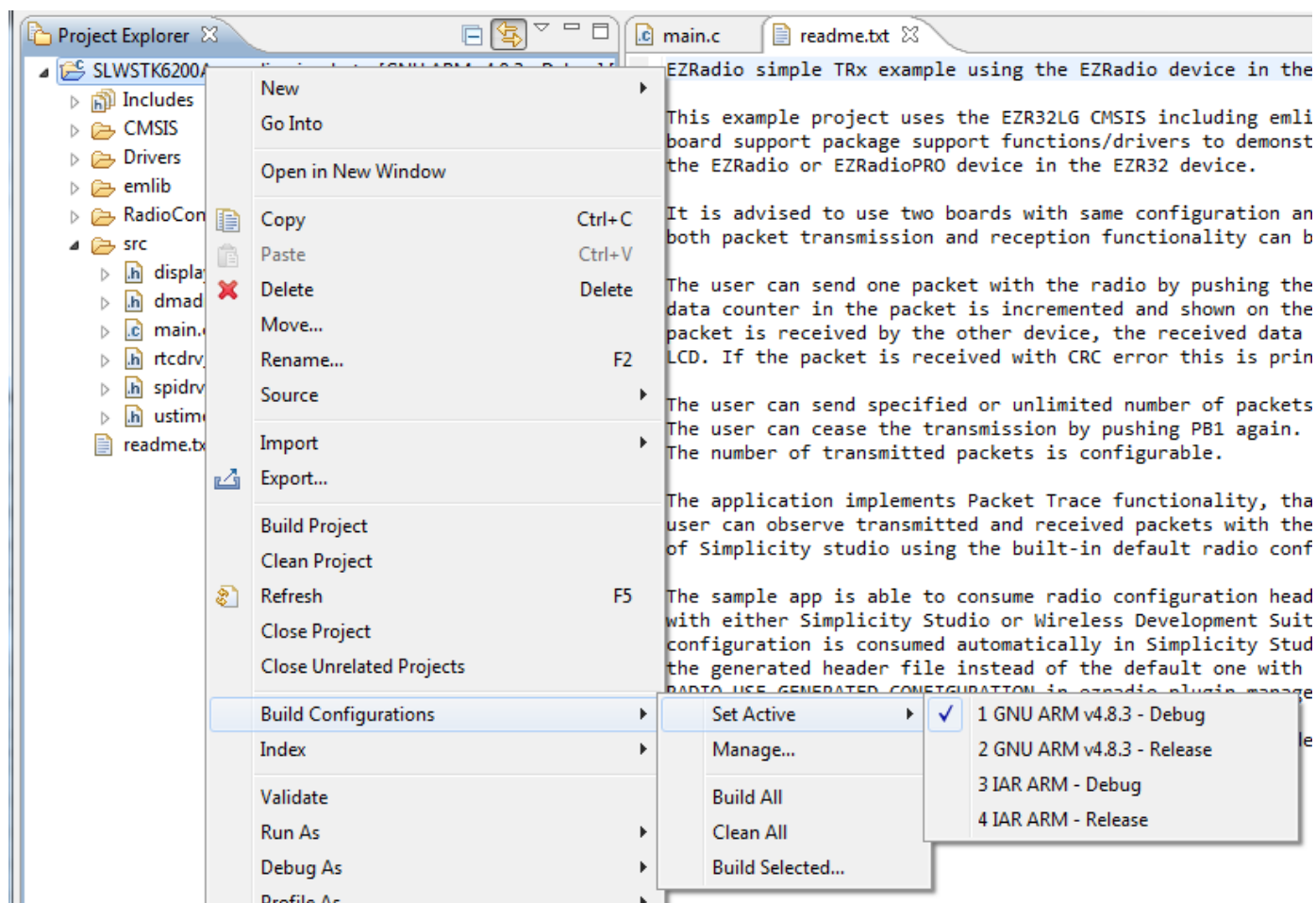


Figure 5.2. Change Build Configuration

5.2 Uploading

The easiest way to upload your application is to start debugging it. To do so, click the **[Debug]** button. This will also build your application if it has not been built yet.



Figure 5.3. Debug Button

After successfully uploading your application, Studio will automatically switch to Debug perspective, suspended before the first line of the main function. To start the program, click the **[Resume]** button.



Figure 5.4. Resume Button

To exit from debug mode, click the **[Disconnect]** button.



Figure 5.5. Disconnect Button

6. Modifying the Radio Configuration

Every application comes with a default configuration, which is compatible between kits for the same frequency band (i.e. the SimpleTRX for SLWSTK6200 can communicate with the SimpleTRX for SLWSTK6220). To modify the configuration, use the `radio-configurator_SLWSTK62xxA.isc` under the RadioConfig folder of your project.

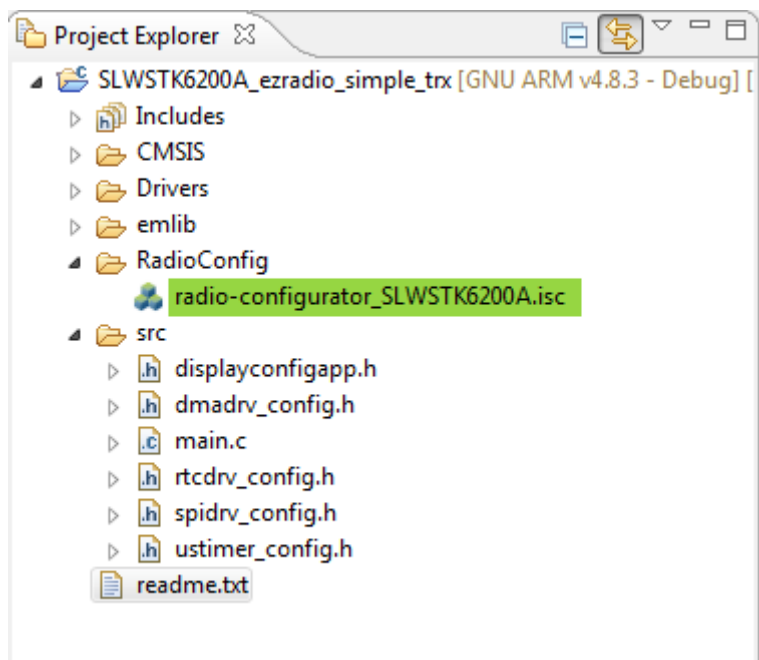


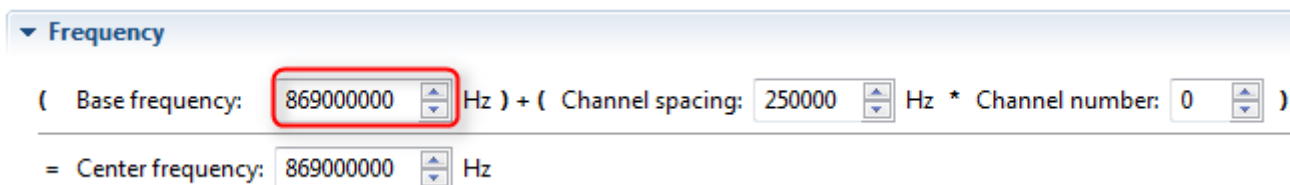
Figure 6.1. Radio Configurator

The radio configurator is build up from sections. Some of the important sections:

- Radio profiles: Modify the driver's behavior.
- Frequency: Modify the carrier frequency.
- Crystal: Sets the crystal parameters and the load capacitance.
- RF parameters: Modify the modulation parameters.
- Packet configuration: Sets the packet configuration and the per-field modulation.
- GPIOs: Set the behavior of the four radio GPIOs.

6.1 Change the Carrier Frequency

Open the radio configurator, expand the Frequency section, and change the base frequency by a few MHz.



▼ Frequency

(Base frequency: 869000000 Hz) + (Channel spacing: 250000 Hz * Channel number: 0)

= Center frequency: 869000000 Hz

Figure 6.2. Frequency Section

To use the modified configuration, click on the Generate button in the upper-right corner:

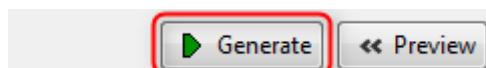


Figure 6.3. Generate Button

A notification window with the generated files will appear (if this is not the first time you generate headers, it will ask permission to overwrite files).

The most important generated file is `radio-config-wds-gen.h`, which holds the actual configuration for the radio. The `app-config.h` is the configuration for the EZRadio driver. The header `radio-config-user-input.h` can be used to access the configuration input from code.

The generator also adds the global constant definition `RADIO_USE_GENERATED_CONFIGURATION` with the value of 1. This will tell the driver to use the generated configuration instead of the pre-generated files from the SDK.

After compiling and uploading, the program should use the modified configuration.

6.2 Configuring the GPIOs

The GPIO configuration can be found under the GPIOs section. By default, GPIO0 and GPIO1 are used by the Packet Trace interface on packet-based examples. To configure GPIO0 and 1, turn on this profile.

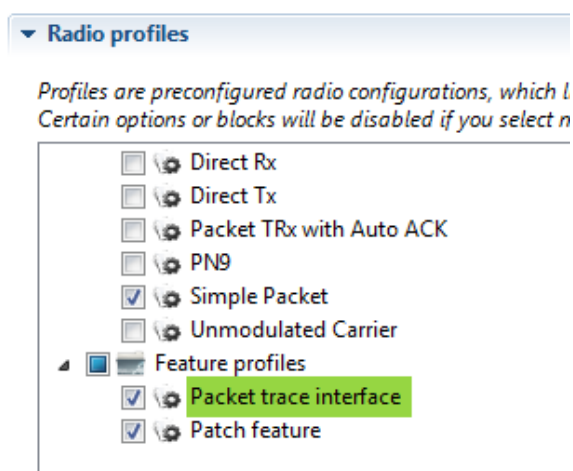


Figure 6.4. Packet Trace Interface Plugin

Note: On high-power designs using switched matching, it is not recommended to change GPIO2 and GPIO3 since they're controlling the RF switch.

Radio GPIOs are very useful for debugging problems. Some useful settings:

- Outputs TX Data bits pulled from the TX FIFO
- Outputs the demodulated RX Data stream
- Outputs TX or RX data CLK
- This output is set high while in TX state
- This output is set high while in RX state
- This output goes high when a valid preamble is detected and returns low after the packet is received
- This output goes high when the chip is NOT in SLEEP state

For example, the following profile can be used to optimize for power consumption:

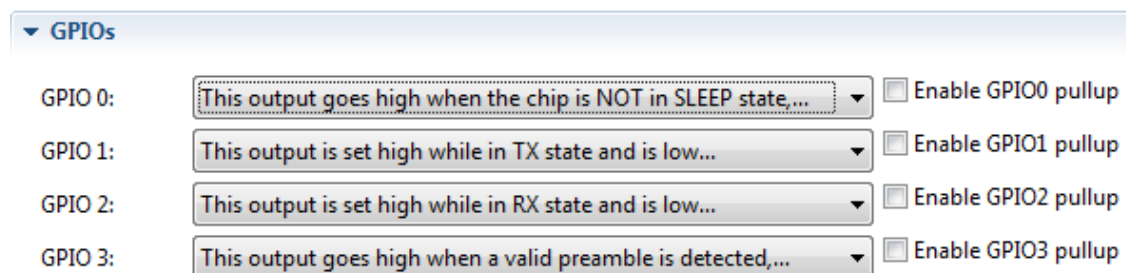
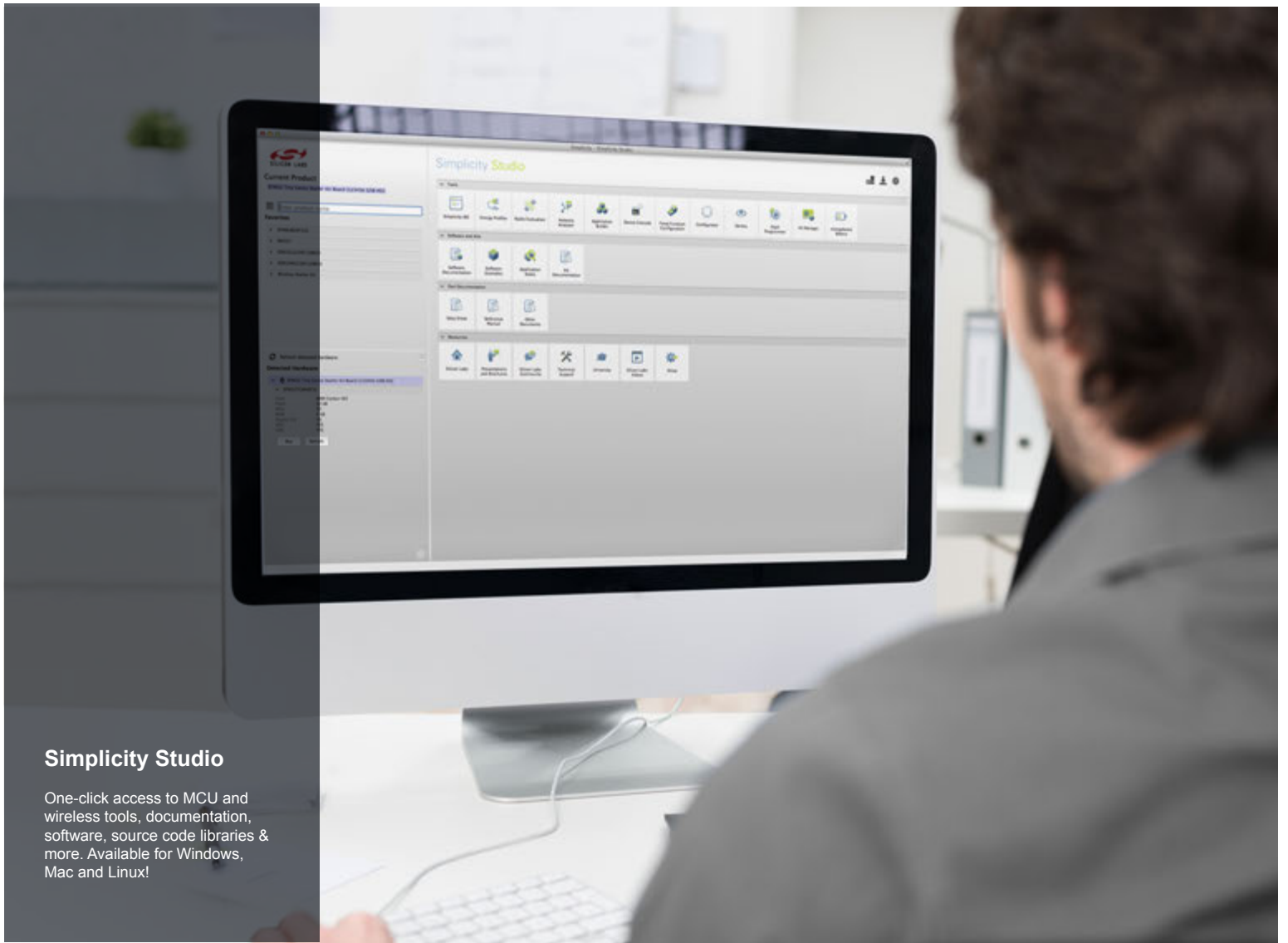


Figure 6.5. GPIO Configuration for Power Optimization

- GPIO0 is high when the radio is not sleeping
- GPIO1 is high when transmitting
- GPIO2 is high when the radio is in receive mode
- GPIO3 is high when the radio is receiving a packet

After compiling and uploading, the program should use the modified configuration. The GPIOs can be found on P30, P31, P32 and P33 on the WSTK. GPIO2 and GPIO3 are directly wired on all kits, GPIO0 and GPIO1 is directly wired on the EZR32HG kits, but only usable through PRS channels on EZR32LG and EZR32WG kits. To configure the PRS channels, see the “EZRadio and EZRadioPRO Transceivers GPIO Configuration” in the data sheet.



Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!



IoT Portfolio
www.silabs.com/iot



SW/HW
www.silabs.com/simplicity



Quality
www.silabs.com/quality



Support and Community
community.silabs.com

Disclaimer

Silicon 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®, CMEMS®, DSPLL®, EFM®, EFM32®, EFR®, Ember®, Energy Micro, Energy Micro logo and combinations thereof, "the world's most energy friendly microcontrollers", Ember®, EZLink®, EZRadio®, EZRadioPRO®, Gecko®, ISOModem®, 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>