Instruction

Z-Wave Getting Started for End Devices

Document No.:	INS14280
Version:	15
Description:	-
Written By:	ANERDO;LAURGE;BSANTOK;PEBALOGH;MAMIHALI
Date:	2025-01-07
Reviewed By:	BSANTOK;ANERDO;MAMIHALI
Restrictions:	Public

Approved by:

Date	CET	Initials Name	Justification
2025-01-29	13:50:00	ANERDO	András Erdo

REVISION RECORD

Doc. Rev	Date	By	Pages affected	Brief description of changes
1	20181206	AMUNKHAUS	ALL	Initial release of document.
2-4	20190829	COLSEN JFR	ALL	Minor typos.
4	20190904	SCBROWNI	ALL	Minor typos.
5	20201125	JFR	Section 4	Added Z-Wave Long Range.
6	20220115	JFR	Front page	Removed 700
7	20220603	ZOJANOSI	Section 6 & 7	Added necessity of bootloader, serial debug, region change
8	20220622	GEKOCZIA	Section 4.1	Added Z-Wave 800 Pro Kit
9	20220725	MNPALANI	Section 5 & 6	Updated Pictures
10	20220926	LAURGE	Section 8.2	Updated pictures and component name
10	20221024	BSANTOK	Sections 4–11	Brought guides up to date, moved PC Controller and Zniffer after Sample Application, moved Energy Profiler to a separate document, added instructions for configuring a development kit as a network sniffer, improved wording, improved accessibility of some figures
10	20221201	BSANTOK	Sections 3, 4, 6, 7.1, 9.2 & References	Added references to supporting documents (mentioning their availability in Simplicity Studio, [13] & [14]), split Section 6 into subsections, added instructions for flashing the bootloader and performing an S2 inclusion, updated instructions for changing the RF region, improved wording
11	20230303	BSANTOK	Sections 5.2.3 & 6	Added information about solutions to Section 6.2, fixed a typo, fixed a mistake in Figure 7, updated Figure 9 to highlight 'Example Projects' and adjusted its size
12	20230322	BSANTOK	Section 11	Removed redundant instructions for flashing the bootloader, clarified connection requirement
13	20230627	BSANTOK	Section 8.1	Add instructions to enable zw_debug
14	20240527	PEBALOGH BSANTOK	Section 8.3 Sections 2.2, 4.1, 5.2.1, 6.4, 8.2, 11	Introduced ZWave CLI component, improved wording, added description of Thunderboard Kits, recommended unplugging radio board when updating adapter FW, updated table of regions, called out renaming of app.c, clarified difference between Zniffer variants
15	20241211	BSANTOK	Sections 4.1, 5.2 6, 7, 8, 11 & 12	Add information about Margay Dev Kits and Solution Examples, recommend ways to gain flash space, make guide consistent with SDK version 7.23 (remapped buttons and LEDs, renamed items)

Table of Contents

Table of Figures

Table of Tables

1 Abbreviations

Abbreviation	Explanation
AEM	Advanced Energy Monitoring
DSK	Device Specific Key
IDE	Integrated Development Environment
OTA	Over The Air
SDK	Software Development Kit
WSTK	Wireless Starter Kit (mainboard)
CLI	Command Line Interface

2 Introduction

2.1 Purpose

This document describes how to get started with Z-Wave development for end devices using Simplicity Studio.

2.2 Audience and Prerequisites

Developers new to Z-Wave will get an introduction to the Z-Wave Development Kit. Developers already familiar with Z-Wave will still benefit from reading this guide, as it demonstrates the new development tools. Common for all developers is that everyone will get a smooth and quick start with Z-Wave development.

There are no prerequisites. It is, however, strongly recommended that one purchase the Z-Wave Development Kit as the development environment auto-discovers hardware and sets itself up accordingly. However, it is possible to have a look and feel of the software package without buying the Development Kit.

3 Welcome to Z-Wave

Z-Wave is a reliable and robust wireless technology particularly designed and developed for Home Automation. Unlike other standards, which rely on heavily congested 2.4 GHz and 5 GHz network where WLAN devices reside, Z-Wave uses Sub-GHz frequency. The chances of interference in Z-Wave networks are much less than other Home Automation standards. Advantages of Z-Wave include:

- Z-Wave uses sub 1 GHz frequency avoiding the heavily congested 2.4 GHz and 5 GHz bands.
- Z-Wave offers secure and reliable two-way communication using message acknowledgement and mesh networking.
- Z-Wave ensures 100% interoperability.

Many people understand different things for the word ‘Interoperability’. Interoperability is *not* just having various nodes joining one network. A Wi-Fi thermostat and printer may operate on the same home network, but they don’t talk to each other. When a consumer leaves his or her home, they want to push one button that will lock all doors, arm the alarm system, set the temperature, and switch off all the lights. The products are likely from different vendors, so just having them join the same network is not interoperability. They all need to speak the exact same language. This is called Application Layer Interoperability.

Z-Wave requires application interoperability in all products and has put in place a stringent certification program that all products go through to ensure correct commands are being used in products.

The certification program is executed by independent certification laboratories, and the process is simple:

1) Join the Z-Wave Alliance.
2) Download the Self-Certification Tool on silabs.com.
3) Access and fill out the online form and documentation.
4) Submit the product to the third-party test house.
5) Once a product passes certification and product and packaging labels are approved, then the Z-Wave interoperability logo is awarded, and the product can go to market.

These steps are further described in [11].

Consumers and channel partners will recognize that the product is Z-Wave certified and will work seamlessly with other Z-Wave products. Z-Wave gives the freedom to choose devices from different vendors, giving end users a choice and enabling manufacturers to leverage the ecosystem of devices.

Z-Wave is interoperability.

4 The Z-Wave Development Kits

The Z-Wave Development Kit is meant to help you evaluate Silicon Labs' Z-Wave modules and get you started with developing your own Z-Wave product.

The Z-Wave Development Kit is designed especially for embedded Z-Wave software and hardware development. The kit includes sample embedded applications for quick prototyping, Z-Wave protocol sniffer tools for analyzing and resolving issues, and Z-Wave RF modules for building prototypes.

User's Guides for the development kits are also available through Simplicity Studio (which will be discussed in Section 5.2).

4.1 Development Kit Hardware by 800 Series

For the latest news of the Z-Wave 800 Series hardware, refer to the following pages:

- 800 Series modules:
  https://www.silabs.com/wireless/z-wave/800-series-modules
- 800 Series EFR32ZG23 SoCs:
  https://www.silabs.com/wireless/z-wave/800-series-modem-soc
- 800 Series EFR32ZG28 SoCs:
  https://www.silabs.com/wireless/z-wave/efr32zg28-z-wave-800-socs

4.1.1 Z-Wave 800 Series Pro Kit

The **Z-Wave 800 Pro Kit** contains the following:

- 2 pcs. BRD4002A – Wireless Starter Kit Mainboard (WSTK).
- 1 pc. BRD4204D – Z-Wave and Z-Wave Long Range EFR32ZG23 868-915 MHz 14 dBm Radio Board.
- 1 pc. BRD4205B – Z-Wave and Z-Wave Long Range ZGM230S Radio Board.
- 1 pc. BRD2603A – Z-Wave and Z-Wave Long Range ZGM230 +14 dBm Dev Kit Board.
- 2 pcs. BRD8029A – Buttons and LEDs Expansion Board (EXP board).
  Note: the Expansion Board is no longer used in SDK versions 7.23 and above. Functionalities previously available on the Expansion Board have been mapped to hardware on the WSTK, with some being only accessible via the Command Line Interface. For details, refer to the README file in each sample application's directory.
- 2 pcs. ANT-SS900 868-915MHz Compressed Whip Antenna.

Note: the UZB-S USB Stick Network Sniffer and the UZB-7 Z-Wave 700 Stick Bridge Module are not part of the Z-Wave 800 Pro Kit. A second Kit can be configured for use as a Z-Wave network sniffer or Serial API interface. Section 11 of this document describes how to configure a Kit as a Network Sniffer. Configuring a Kit for use with the Z-Wave PC Controller Host Application can be done in a similar manner, by flashing the Z-Wave – NCP Serial API Controller application.

Other Pro Kits featuring EFR32ZG28 hardware (with an increased flash size) are also available:

- With a 14 dBm radio board:
  https://www.silabs.com/development-tools/wireless/efr32xg28-pro-kit-14-dbm
- With a 20 dBm radio board:
  https://www.silabs.com/development-tools/wireless/efr32xg28-pro-kit-20-dbm

4.1.1.1 Prepare the Hardware

Before installing any software or before powering any of the hardware, start by preparing the needed hardware.

1) Connect the radio board BRD4204D/BRD4205B to the mainboard BRD4002A.
2) If using SDK version 7.22 or below, connect the EXP board BRD8029A to the extension port of the mainboard BRD4002A.
3) Set the Power switch of the WSTK mainboard in AEM position.
4) Connect the mainboard BRD4002A using a USB cable to the PC.

Familiarize yourself with the hardware by locating the push buttons, LEDs, etc.

4.1.2 Z-Wave Thunderboard Kits

The Z-Wave Thunderboard Kits are small form factor, cost-effective, all-in-one development platforms equipped with a USB Type-C connector.

These development boards can be configured to act as end devices, network controllers or network sniffers.

Note: some sample applications support only a different or reduced set of functionalities on these development boards due to hardware limitations.

4.1.2.1 Z-Wave 800 Series Development Kit

The contents of the **Z-Wave 800 Series Development Kit** are the following:

- 1 pc. BRD2603A – ZGM230s +14 dBm Dev Kit Board.
- 1 pc. ANT SS900 – 868-915 MHz Antenna

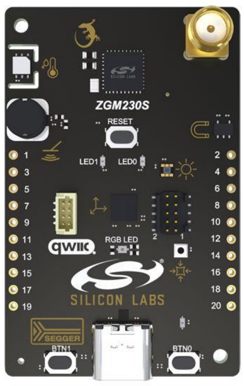


Figure 1: BRD2603A - ZGM230s +14 dBm Dev Kit Board

4.1.2.2 EFR32xG28 Explorer Kit

The **EFR32xG28 Explorer Kit** contains a single xG28 Explorer board (BRD2705A) with an integrated antenna.

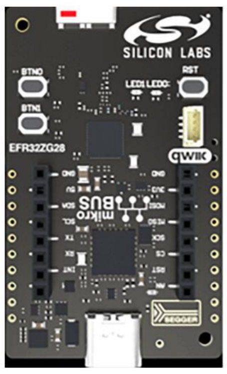


Figure 2: EFR32xG28 Explorer Kit

4.2 Development Kit Hardware by 700 Series

To get the latest news of the Z-Wave 700, refer to:
https://www.silabs.com/wireless/z-wave/700-series-modules
and
https://www.silabs.com/wireless/z-wave/700-series-modem-soc

The **Z-Wave Long Range 700 Starter Kit** contains the following:

- 2 pcs. BRD4001A – Wireless Starter Kit Mainboard (WSTK).
- 2 pcs. BRD4207A – Z-Wave and Z-Wave Long Range ZGM130S Radio Board intended end device development.
- 2 pcs. BRD8029A – Buttons and LEDs Expansion Board (EXP board).
- 1 pc. SLUSB001A – UZB7 – Controller USB stick.
- 1 pc. UZB-S – (ACC-UZB3-S) UZB-S USB stick network sniffer.
- 2 pcs. ANT-SS900 – 868-915 MHz Compressed Whip Antenna.

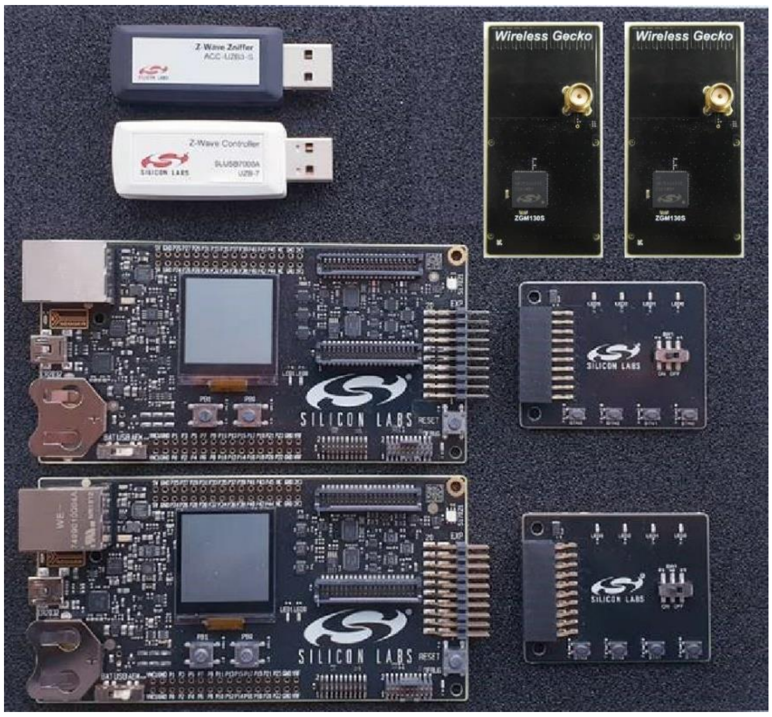


Figure 3: Content of the Z-Wave Development Kit

For a more in-depth description of the various hardware components, refer to 'How to Use Z-Wave Pre-Certified Apps' [1].

4.2.1 Prepare the Hardware

Before installing any software or before powering any of the hardware, start by preparing the needed hardware.

1) Connect the radio board BRD4207A to the mainboard BRD4001A.
2) Connect the EXP board BRD8029A to the extension port of the mainboard BRD4001A.
3) Set the Power switch of the WSTK mainboard in AEM position.
4) Connect the mainboard BRD4001A using a USB cable to the PC.

Familiarize yourself with the hardware by locating the various push buttons, LEDs, etc.

- BRD4207A Radio Board with ZGM130S used for end device development
- BRD8029A EXP Board

5 Install the Z-Wave SDK

Simplicity Studio is a free Eclipse-based Integrated Development Environment and a collection of value-add tools provided by Silicon Labs. Developers can use Simplicity Studio to develop, debug, and analyze their Z-Wave and other Silicon Labs SDK applications. Its main goal is to reduce development time so that you can focus on your application.

Before proceeding, you need an account for silabs.com. You can register at:
https://siliconlabs.force.com/apex/SL_CommunitiesSelfReg?form=short

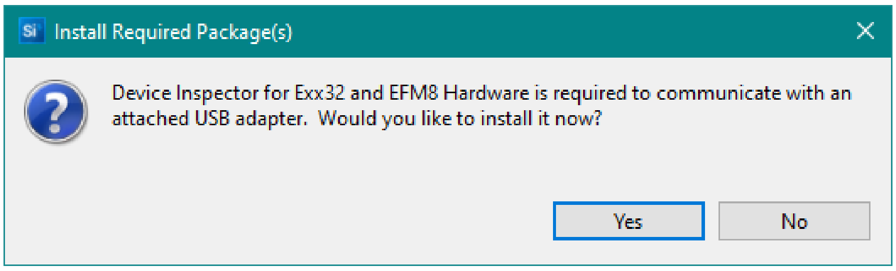
5.1 Connect your Hardware

Make sure you have set up and connected the hardware as described in Section 4.1.1.1 or 4.2.1.

5.2 Install Simplicity Studio

1) Download the latest version of Simplicity Studio from:
https://www.silabs.com/products/development-tools/software/simplicity-studio
2) When the download is complete, run the Simplicity Studio installer. The installer will present a License Agreement dialog. Accept the terms of the agreement and click *Next*.
3) Choose the installation destination. You can use the default location. Click *Next* and then click *Install*.
4) When the application launches, you will be presented with a License Agreement dialog. Accept the agreements and click *Done*.
5) Log in with your Silicon Labs Community account when prompted.

6) After logging in, Simplicity Studio installs additional software components. Allow the program to install the required drivers when prompted.
   Once the 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 device and prompt you to install a Device Inspector. Click *Yes*. See Figure 4.

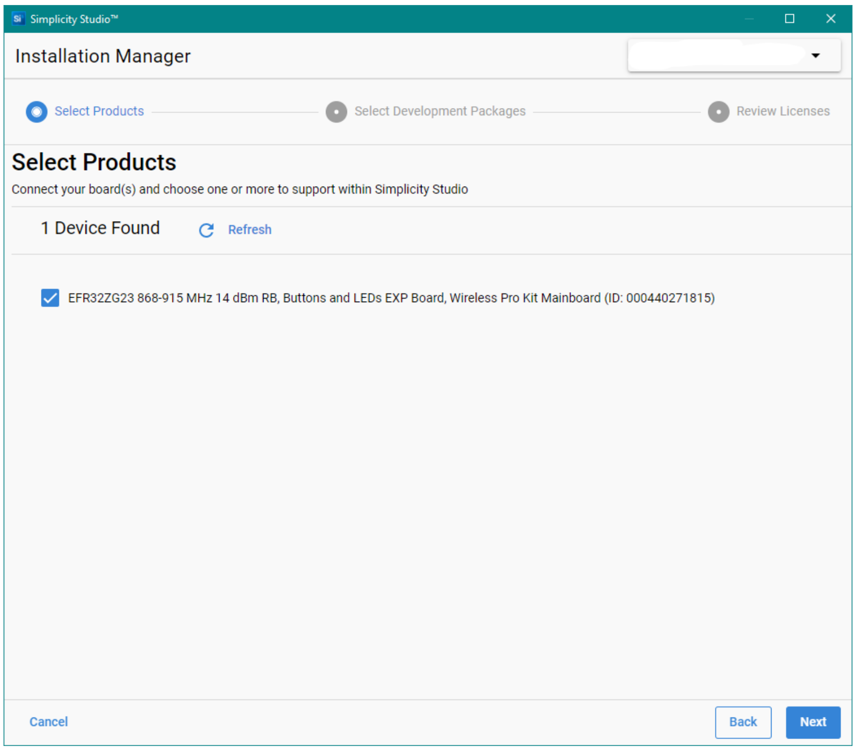


Install Required Package(s)

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

Figure 4: Install Required Device Inspector

7) After some additional items are installed, you are offered the option of installing by device or installing by product group. This guide will be using the '*Install by connecting device(s)'* option, which will install the relevant software based on the connected hardware (if you do not have any hardware connected, you can browse for possible hardware kits, and still get the same easy install experience). See Figure 5. Select your products and click '*Next*'.



Simplicity Studio™

Installation Manager

Select Products

Connect your board(s) and choose one or more to support within Simplicity Studio

1 Device Found

☑ EFR32ZG23 868-915 MHz 14 dBm RB, Buttons and LEDs EXP Board, Wireless Pro Kit Mainboard (ID: 000440271815)

Figure 5: Select Devices to Install the Needed Software Packages

8) You will be presented with the Package Installation Options. You can choose '*Advanced'* and uncheck any packages that you do not want to install, but it is recommended to select the '*Auto'* option here.

9) Next, the studio displays a Review Licenses dialog. Accept the licenses shown and click *Finish*.

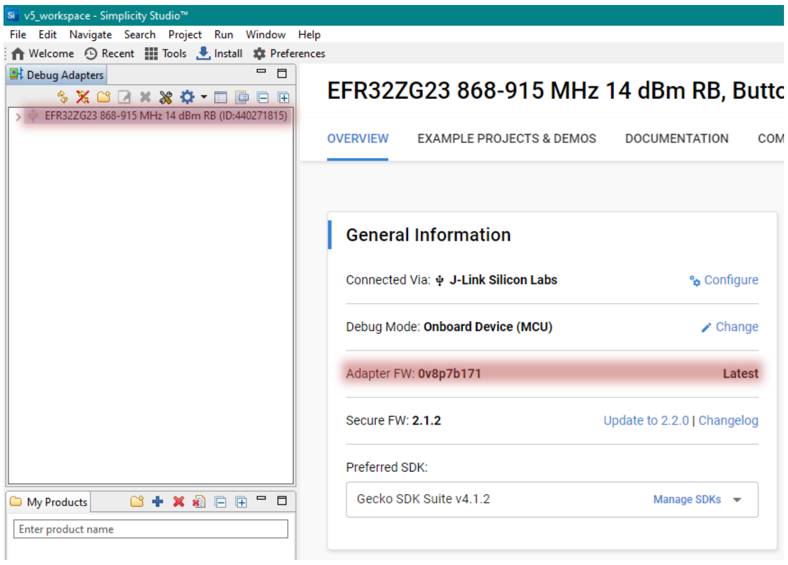
10) Installation takes several minutes. During installation, Simplicity Studio offers you viewing and reading options to learn more about the environment. After installation is complete, restart Simplicity Studio.

11) When Simplicity Studio restarts, you may be presented with another Review Licenses dialog. Accept the license and click '*Done'*.

5.2.1 Updating Adapter Firmware

The final step before proceeding is to update the device firmware. Make sure you have selected a device from the '*Debug Adapters*' view on the left, and then click '*Download'* and/or '*Install'* if a new version is available.

Note: It is recommended that you unplug the radio board from the mainboard while updating the adapter firmware.


v5_workspace - Simplicity Studio™
File Edit Navigate Search Project Run Window Help
Welcome Recent Tools Install Preferences
Debug Adapters
EFR32ZG23 868-915 MHz 14 dBm RB (ID:440271815)
My Products
Enter product name
EFR32ZG23 868-915 MHz 14 dBm RB, Buttc
OVERVIEW
EXAMPLE PROJECTS & DEMOS
DOCUMENTATION
COM

General Information

Connected Via: J-Link Silicon Labs — Configure

Debug Mode: Onboard Device (MCU) — Change

Adapter FW: 0v8p7b171 — Latest

Secure FW: 2.1.2 — Update to 2.2.0 | Changelog

Preferred SDK: Gecko SDK Suite v4.1.2 — Manage SDKs

Figure 6: Device Firmware Update

5.2.2 Associate Simplicity Studio with the Hardware

The default view when opening Simplicity Studio is the Launcher perspective. In this view, click the connection entry in the '*Debug Adapters*' view. The Launcher perspective is then populated with the software components and functionality associated with your hardware and installed SDKs.

5.2.3 Functionality in the Launcher Perspective

Simplicity Studio is now fully configured and ready to use. This section introduces the features of the welcome screen called the Launcher Perspective.

Perspectives are made up of several tiles or panes, called views, as well as the content in those views. You can perform several functions in the Launcher Perspective, as shown in Figure 7.

You can also refer to https://docs.silabs.com/simplicity-studio-5-users-guide/latest/ss-5-users-guide-overview/ for detailed documentation on Simplicity Studio.

1) In the **Toolbar**, you can:

- Open application settings
- Update your software and firmware
- Open the Tools menu

2) Change perspectives

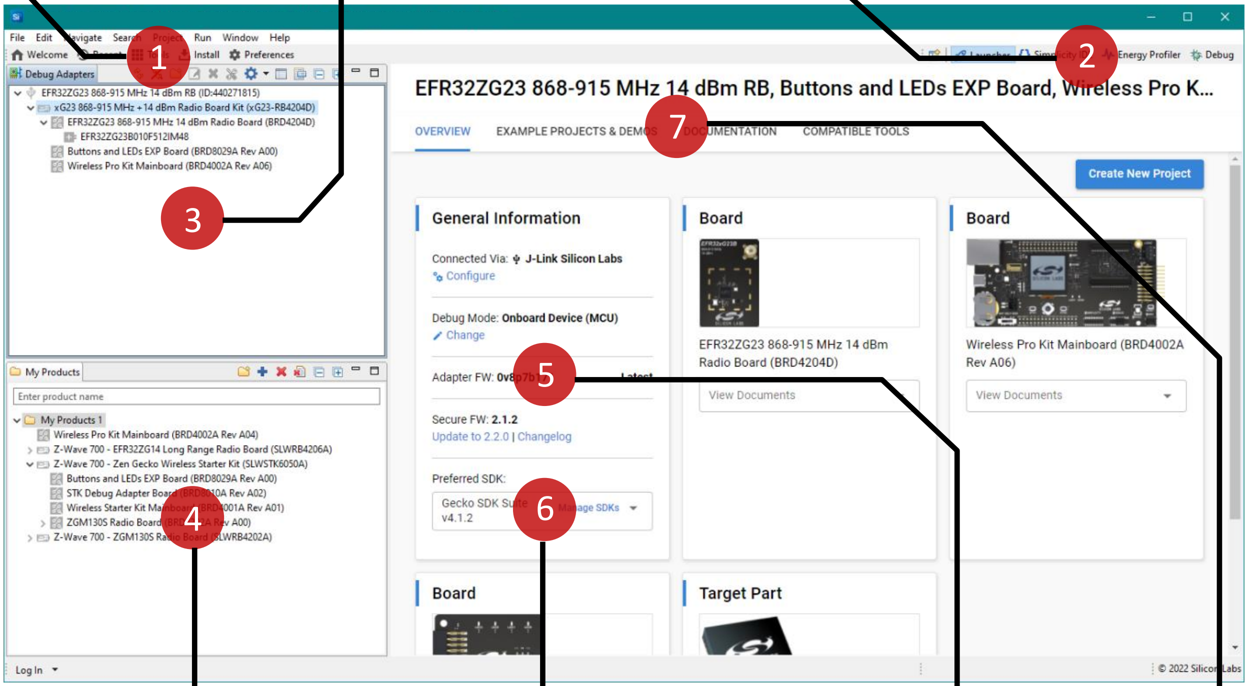
As you open the Simplicity IDE or other tools, buttons for different perspectives are displayed in the upper right corner. Use these buttons to easily navigate back to the Launcher perspective or to other perspectives.

You can change the layouts of various perspectives by expanding or relocating views or adding or removing views.

To return to the default layout, right-click the perspective button in the upper right and select Reset.

3) The **Debug Adapters** view lists all the connected hardware.

Devices are organized in a hierarchical fashion, showing the adapter at the top, which can then be expanded into the development board, the IC on the board, the expansion board, etc.



EFR32ZG23 868-915 MHz 14 dBm RB (ID:440271815)
xG23 868-915 MHz +14 dBm Radio Board Kit (xG23-RB4204D)
EFR32ZG23 868-915 MHz 14 dBm Radio Board (BRD4204D)
EFR32ZG23B010F512IM48
Buttons and LEDs EXP Board (BRD8029A Rev A00)
Wireless Pro Kit Mainboard (BRD4002A Rev A06)

My Products 1
Wireless Pro Kit Mainboard (BRD4002A Rev A04)
Z-Wave 700 - EFR32ZG14 Long Range Radio Board (SLWRB4206A)
Z-Wave 700 - Zen Gecko Wireless Starter Kit (SLWSTK6050A)
Buttons and LEDs EXP Board (BRD8029A Rev A00)
STK Debug Adapter Board (BRD8010A Rev A02)
Wireless Starter Kit Mainboard (BRD4001A Rev A01)
ZGM130S Radio Board (BRD4202A Rev A00)
Z-Wave 700 - ZGM130S Radio Board (SLWRB4202A)

EFR32ZG23 868-915 MHz 14 dBm RB, Buttons and LEDs EXP Board, Wireless Pro K...

General Information

Connected Via: J-Link Silicon Labs

Debug Mode: Onboard Device (MCU)

Adapter FW: 0v8p7b1

Secure FW: 2.1.2
Update to 2.2.0 | Changelog

Preferred SDK:
Gecko SDK Suite v4.1.2

Board
EFR32ZG23 868-915 MHz 14 dBm Radio Board (BRD4204D)

Board
Wireless Pro Kit Mainboard (BRD4002A Rev A06)

Board

Target Part

4) The **My Products** view allows you create solutions comprised of multiple parts.

If you are developing for complex networks with several different parts involved, you can add them all to the solution and then select the one you are working on from the list.

You do not need to have the hardware connected.

6) Your **preferred SDK.** When working with Z-Wave devices, the Simplicity SDK Suite should be selected.

5) You can update the **adapter firmware** here (displayed only if a device is connected).

7) Documentation and Other Resources

- The Example Projects & Demos tab provides access to demos, example applications and solution examples.
- The Documentation tab lists documentation about the stack and about the hardware.
- The Compatible Tools tab is an alternative way to access the tools available through the Show Compatible Tools dialog.

Figure 7: The Launcher Perspective

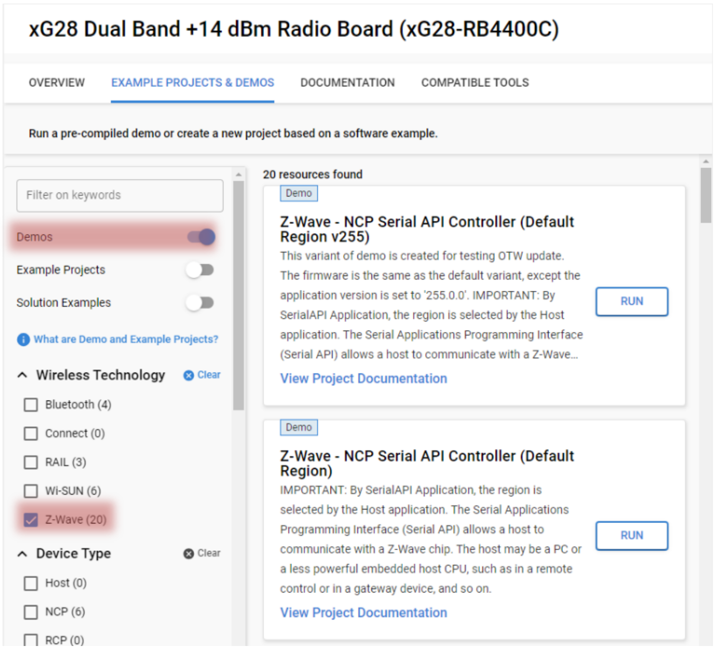
6 Prepare Your First Z-Wave Sample Application

Having all the hardware and software set up, you are finally ready to build and run the sample applications.

In the Launcher Perspective, make sure the hardware is selected and that the preferred SDK is set to the Simplicity SDK Suite.

The *Example Projects & Demos* tab contains all the available Z-Wave sample applications, with different features and levels of customizability.

Pre-built *Demos* are binary files either containing a combination of a bootloader and a sample application in its default configuration or only a bootloader for a specific device.



xG28 Dual Band +14 dBm Radio Board (xG28-RB4400C)

OVERVIEW | EXAMPLE PROJECTS & DEMOS | DOCUMENTATION | COMPATIBLE TOOLS

Run a pre-compiled demo or create a new project based on a software example.

Filter on keywords

Demos [✓ on]
Example Projects
Solution Examples

What are Demo and Example Projects?

Wireless Technology — Clear
- Bluetooth (4)
- Connect (0)
- RAIL (3)
- Wi-SUN (6)
- [✓] Z-Wave (20)

Device Type — Clear
- Host (0)
- NCP (6)
- RCP (0)

20 resources found

Demo

Z-Wave - NCP Serial API Controller (Default Region v255)

This variant of demo is created for testing OTW update. The firmware is the same as the default variant, except the application version is set to '255.0.0'. IMPORTANT: By SerialAPI Application, the region is selected by the Host application. The Serial Applications Programming Interface (Serial API) allows a host to communicate with a Z-Wave...

RUN

View Project Documentation

Demo

Z-Wave - NCP Serial API Controller (Default Region)

IMPORTANT: By SerialAPI Application, the region is selected by the Host application. The Serial Applications Programming Interface (Serial API) allows a host to communicate with a Z-Wave chip. The host may be a PC or a less powerful embedded host CPU, such as in a remote control or in a gateway device, and so on.

RUN

View Project Documentation

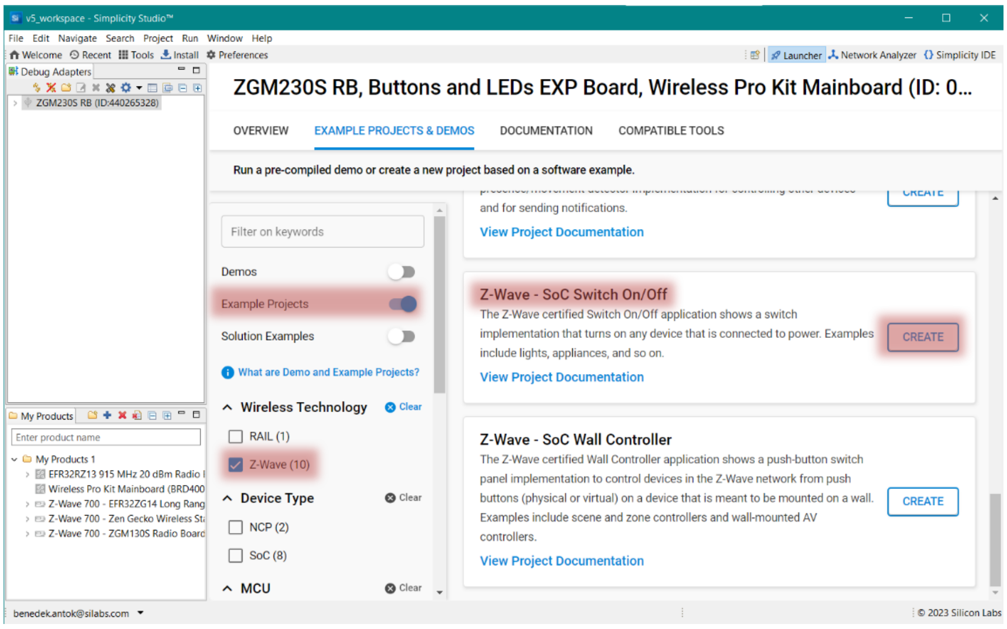
Figure 8: Simplicity Studio Snapshot Listing Pre-Built Z-Wave Demos

These demo binaries are ready to be flashed to a compatible Z-Wave development board using the *'RUN'* button.

They come in regular and *'v255'* variants. The latter is identical to the regular version, except that it has its major version number set to the highest possible value to facilitate performing an OTA update.

The demo binaries are configured to use the default Z-Wave region (Europe), unless otherwise configured via a manufacturing token. To set a different region, refer to Section 6.4.

Example Projects are sample applications that can be used as a basis for custom application development. These projects do not contain a bootloader; a pre-built bootloader demo may be used in conjunction with them.


v5_workspace - Simplicity Studio™
File Edit Navigate Search Project Run Window Help
Welcome Recent Tools Install Preferences
Launcher Network Analyzer Simplicity IDE
Debug Adapters
ZGM230S RB (ID:440265328)
My Products
Enter product name
My Products 1
EFR32RZ13 915 MHz 20 dBm Radio I
Wireless Pro Kit Mainboard (BRD400
Z-Wave 700 - EFR32ZG14 Long Rang
Z-Wave 700 - Zen Gecko Wireless Sta
Z-Wave 700 - ZGM130S Radio Board
benedek.antok@silabs.com
ZGM230S RB, Buttons and LEDs EXP Board, Wireless Pro Kit Mainboard (ID: 0...
OVERVIEW
EXAMPLE PROJECTS & DEMOS
DOCUMENTATION
COMPATIBLE TOOLS
Filter on keywords
Demos
Example Projects
Solution Examples
What are Demo and Example Projects?
Wireless Technology
Clear
RAIL (1)
Z-Wave (10)
Device Type
Clear
NCP (2)
SoC (8)
MCU
Clear
CREATE
View Project Documentation
CREATE
View Project Documentation
CREATE
View Project Documentation
© 2023 Silicon Labs

Run a pre-compiled demo or create a new project based on a software example.

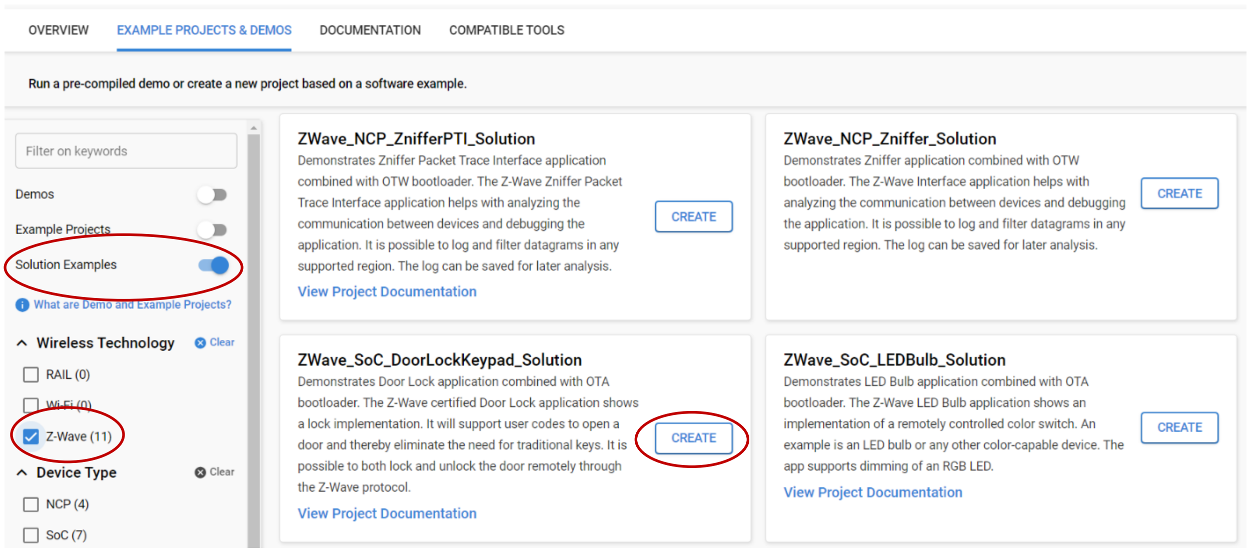
and for sending notifications.

Z-Wave - SoC Switch On/Off
The Z-Wave certified Switch On/Off application shows a switch implementation that turns on any device that is connected to power. Examples include lights, appliances, and so on.

Z-Wave - SoC Wall Controller
The Z-Wave certified Wall Controller application shows a push-button switch panel implementation to control devices in the Z-Wave network from push buttons (physical or virtual) on a device that is meant to be mounted on a wall. Examples include scene and zone controllers and wall-mounted AV controllers.

Figure 9: Simplicity Studio Snapshot Listing Z-Wave Sample Application Example Projects

Z-Wave *Solution Examples*, sometimes referred to as *'workspaces'*, are combinations of a bootloader project and a Z-Wave Example Project.



Run a pre-compiled demo or create a new project based on a software example.

ZWave_NCP_ZnifferPTI_Solution
Demonstrates Zniffer Packet Trace Interface application combined with OTW bootloader. The Z-Wave Zniffer Packet Trace Interface application helps with analyzing the communication between devices and debugging the application. It is possible to log and filter datagrams in any supported region. The log can be saved for later analysis.

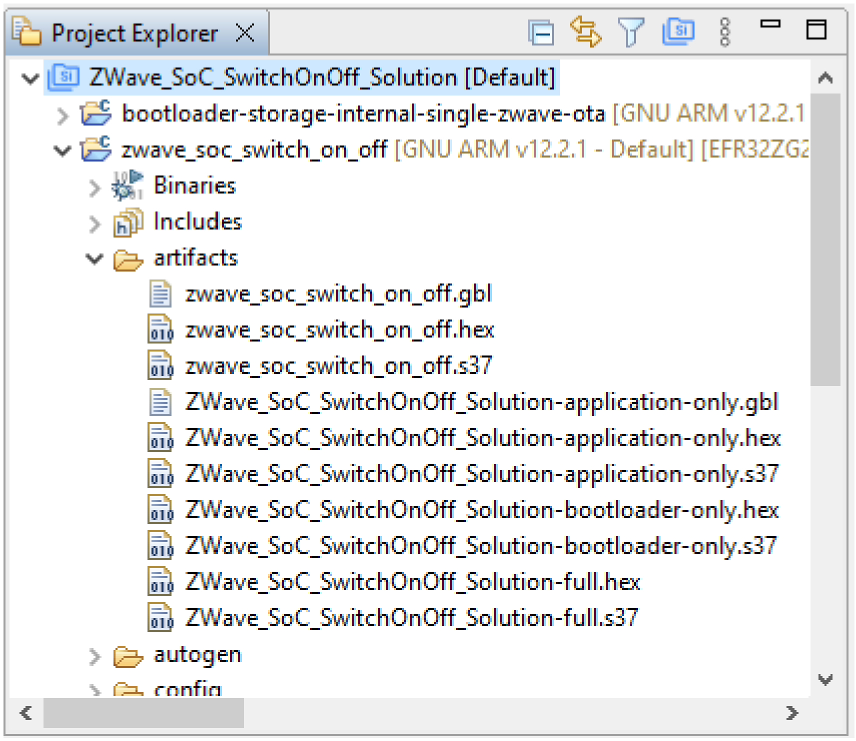
ZWave_NCP_Zniffer_Solution
Demonstrates Zniffer application combined with OTW bootloader. The Z-Wave Interface application helps with analyzing the communication between devices and debugging the application. It is possible to log and filter datagrams in any supported region. The log can be saved for later analysis.

ZWave_SoC_DoorLockKeypad_Solution
Demonstrates Door Lock application combined with OTA bootloader. The Z-Wave certified Door Lock application shows a lock implementation. It will support user codes to open a door and thereby eliminate the need for traditional keys. It is possible to both lock and unlock the door remotely through the Z-Wave protocol.

ZWave_SoC_LEDBulb_Solution
Demonstrates LED Bulb application combined with OTA bootloader. The Z-Wave LED Bulb application shows an implementation of a remotely controlled color switch. An example is an LED bulb or any other color-capable device. The app supports dimming of an RGB LED.

Figure 10: Simplicity Studio Snapshot Listing Z-Wave Solution Examples

In Solution Examples, a binary combining the bootloader and the sample application is created at the end of the build process. This binary can be found in the artifacts folder of the application project, with a '_full' postfix.
Other binaries containing only the bootloader and only the application can also be found in this folder.



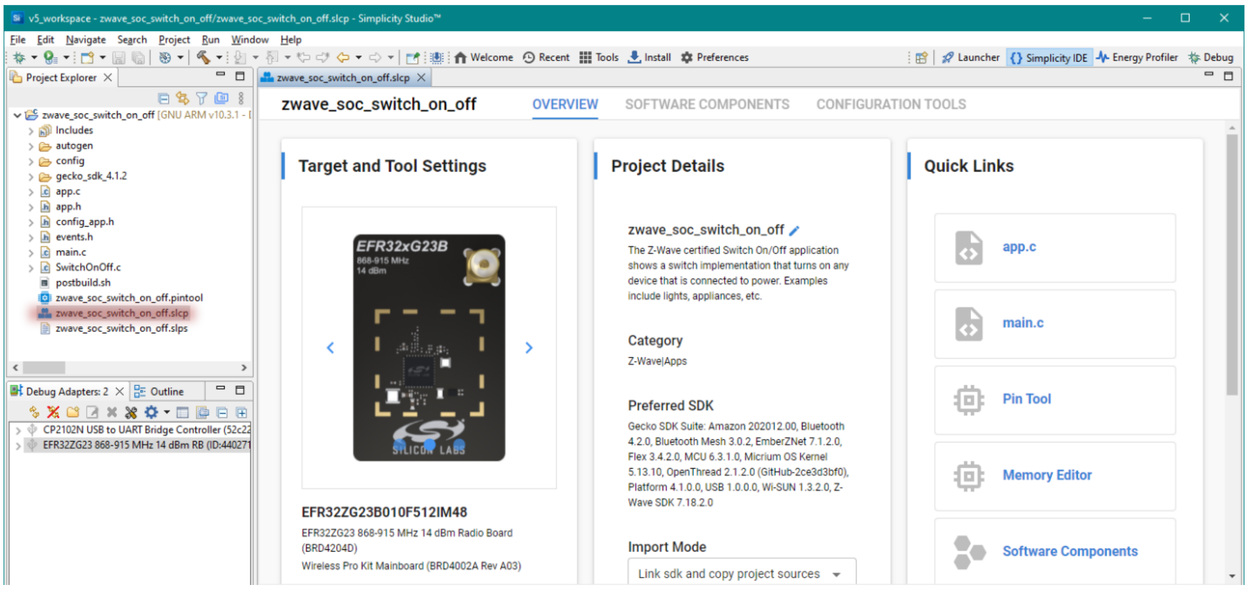
```
ZWave_SoC_SwitchOnOff_Solution [Default]
  bootloader-storage-internal-single-zwave-ota [GNU ARM v12.2.1
  zwave_soc_switch_on_off [GNU ARM v12.2.1 - Default] [EFR32ZG2
    Binaries
    Includes
    artifacts
      zwave_soc_switch_on_off.gbl
      zwave_soc_switch_on_off.hex
      zwave_soc_switch_on_off.s37
      ZWave_SoC_SwitchOnOff_Solution-application-only.gbl
      ZWave_SoC_SwitchOnOff_Solution-application-only.hex
      ZWave_SoC_SwitchOnOff_Solution-application-only.s37
      ZWave_SoC_SwitchOnOff_Solution-bootloader-only.hex
      ZWave_SoC_SwitchOnOff_Solution-bootloader-only.s37
      ZWave_SoC_SwitchOnOff_Solution-full.hex
      ZWave_SoC_SwitchOnOff_Solution-full.s37
    autogen
    config
```

Figure 11: Artifacts Generated for a Solution Example in Simplicity Studio

This guide will demonstrate how to open, compile, and program a device with the Switch On/Off Sample Application, using an Example Project.

6.1 Configuring a Sample Application

Click on ‘*Create*’ in the Launcher Perspective, then click on ‘*Finish*’ in the Project Configuration dialog.
Simplicity Studio will generate a local copy of the example project and open the Simplicity IDE Perspective, as shown in Figure 12.



v5_workspace - zwave_soc_switch_on_off/zwave_soc_switch_on_off.slcp - Simplicity Studio™

File Edit Navigate Search Project Run Window Help

Welcome Recent Tools Install Preferences Launcher Simplicity IDE Energy Profiler Debug

Project Explorer

- zwave_soc_switch_on_off [GNU ARM v10.3.1 -
  - Includes
  - autogen
  - config
  - gecko_sdk_4.1.2
  - app.c
  - app.h
  - config_app.h
  - events.h
  - main.c
  - SwitchOnOff.c
  - postbuild.sh
  - zwave_soc_switch_on_off.pintool
  - zwave_soc_switch_on_off.slcp
  - zwave_soc_switch_on_off.slps

Debug Adapters: 2 Outline

- CP2102N USB to UART Bridge Controller (52c22
- EFR32ZG23 868-915 MHz 14 dBm RB (ID:440271

zwave_soc_switch_on_off.slcp

zwave_soc_switch_on_off

OVERVIEW SOFTWARE COMPONENTS CONFIGURATION TOOLS

Target and Tool Settings

EFR32ZG23B010F512IM48
EFR32ZG23 868-915 MHz 14 dBm Radio Board (BRD4204D)
Wireless Pro Kit Mainboard (BRD4002A Rev A03)

Project Details

zwave_soc_switch_on_off
The Z-Wave certified Switch On/Off application shows a switch implementation that turns on any device that is connected to power. Examples include lights, appliances, etc.

Category
Z-Wave|Apps

Preferred SDK
Gecko SDK Suite: Amazon 202012.00, Bluetooth 4.2.0, Bluetooth Mesh 3.0.2, EmberZNet 7.1.2.0, Flex 3.4.2.0, MCU 6.3.1.0, Micrium OS Kernel 5.13.10, OpenThread 2.1.2.0 (GitHub-2ce3d3bf0), Platform 4.1.0.0, USB 1.0.0.0, Wi-SUN 1.3.2.0, Z-Wave SDK 7.18.2.0

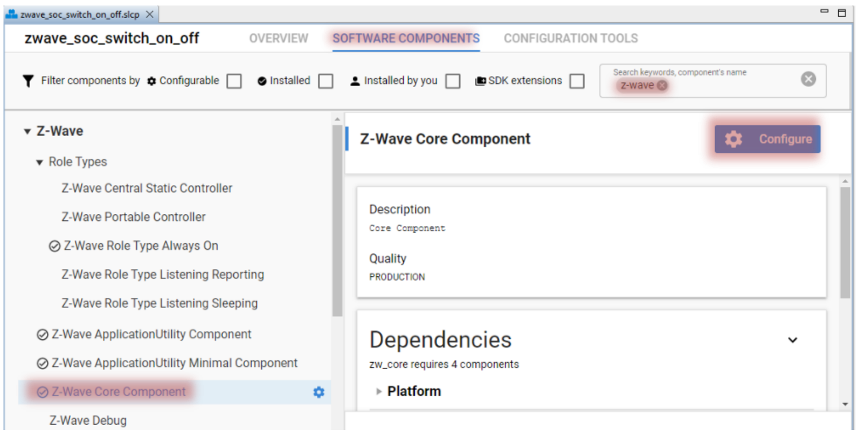
Import Mode
Link sdk and copy project sources

Quick Links

app.c
main.c
Pin Tool
Memory Editor
Software Components

Figure 12: The Simplicity IDE Perspective

The main logic of this sample application is in the file called ‘app.*c*’ (or *‘SwitchOnOff.c’* prior to SDK version 7.20).
Before building the application, the desired frequency for the sample application can be configured by going to the ‘*Software Components*’ tab, selecting the ‘*Z-Wave Core Component’,* and clicking on ‘*Configure*’ in as highlighted in Figure 14.


zwave_soc_switch_on_off.slcp
zwave_soc_switch_on_off
OVERVIEW
SOFTWARE COMPONENTS
CONFIGURATION TOOLS
Filter components by
Configurable
Installed
Installed by you
SDK extensions
Search keywords, component's name
z-wave
Z-Wave
Role Types
Z-Wave Central Static Controller
Z-Wave Portable Controller
Z-Wave Role Type Always On
Z-Wave Role Type Listening Reporting
Z-Wave Role Type Listening Sleeping
Z-Wave ApplicationUtility Component
Z-Wave ApplicationUtility Minimal Component
Z-Wave Core Component
Z-Wave Debug
Configure
Platform

Z-Wave Core Component

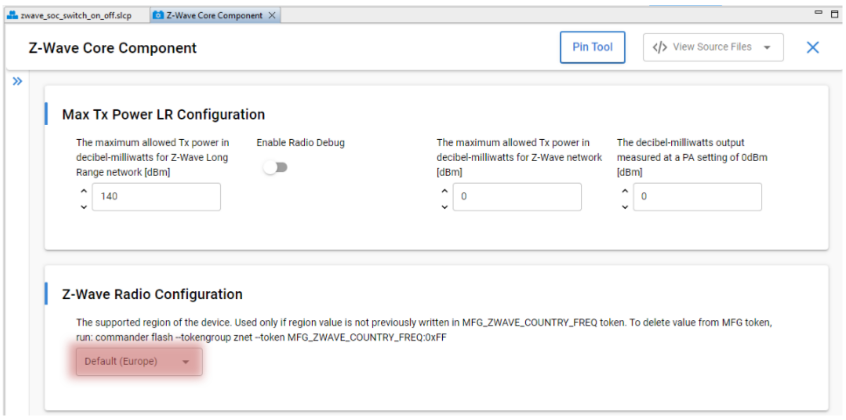
Description
Core Component

Quality
PRODUCTION

Dependencies
zw_core requires 4 components

Figure 13: Application Configuration Window

From here, select the desired frequency as shown in Figure 14.



Z-Wave Core Component

Max Tx Power LR Configuration

The maximum allowed Tx power in decibel-milliwatts for Z-Wave Long Range network [dBm]: 140

Enable Radio Debug

The maximum allowed Tx power in decibel-milliwatts for Z-Wave network [dBm]: 0

The decibel-milliwatts output measured at a PA setting of 0dBm [dBm]: 0

Z-Wave Radio Configuration

The supported region of the device. Used only if region value is not previously written in MFG_ZWAVE_COUNTRY_FREQ token. To delete value from MFG token, run: commander flash --tokengroup znet --token MFG_ZWAVE_COUNTRY_FREQ:0xFF

Default (Europe)

Figure 14: Application Configuration Parameter for Selecting Region-Frequency

The default frequency is set to EU region. Navigate to the Silicon Labs website, to see where Z-Wave RF has been approved.
Refer to Table 1 for a complete list of frequencies supported by the SDK.

Frequency Region	Variable to use	Frequency Region	Variable to use
Europe	REGION_EU	Russia	REGION_RU
United States of America	REGION_US	China	REGION_CN
Australia/New Zealand	REGION_ANZ	USA – Long Range	REGION_US_LR
Hong Kong	REGION_HK	Europe – Long Range	REGION_EU_LR
India	REGION_IN	Japan	REGION_JP
Israel	REGION_IL	Korea	REGION_KR

Table 1: Overview of Selectable Frequencies

Click on Build, as shown in Figure 15.

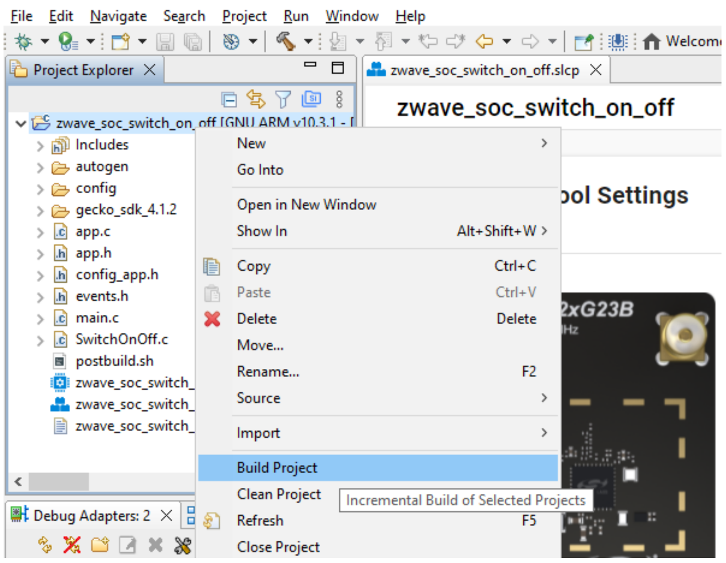
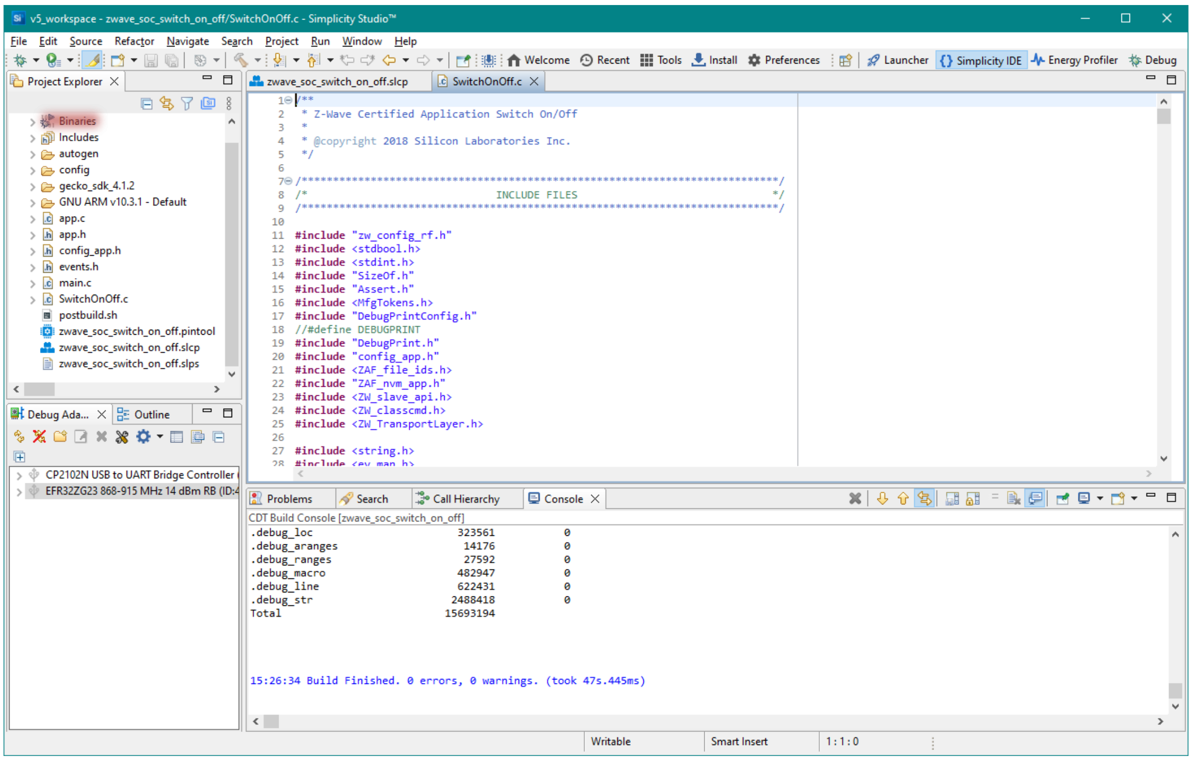


Figure 15: Sample Application - Build Step

Once the build finishes (Figure 16), after a short while, a new folder named ‘*Binaries*’ shows up in the Project Explorer view.



v5_workspace - zwave_soc_switch_on_off/SwitchOnOff.c - Simplicity Studio™
File Edit Source Refactor Navigate Search Project Run Window Help
Welcome Recent Tools Install Preferences Launcher Simplicity IDE Energy Profiler Debug
Project Explorer
Binaries
Includes
autogen
config
gecko_sdk_4.1.2
GNU ARM v10.3.1 - Default
app.c
app.h
config_app.h
events.h
main.c
SwitchOnOff.c
postbuild.sh
zwave_soc_switch_on_off.pintool
zwave_soc_switch_on_off.slcp
zwave_soc_switch_on_off.slps
zwave_soc_switch_on_off.slcp
SwitchOnOff.c
Debug Ada...
Outline
CP2102N USB to UART Bridge Controller
EFR32ZG23 868-915 MHz 14 dBm RB (ID:4
Problems
Search
Call Hierarchy
Console
Writable
Smart Insert
1 : 1 : 0

```
/**
 * Z-Wave Certified Application Switch On/Off
 *
 * @copyright 2018 Silicon Laboratories Inc.
 */

/****************************************************************************/
/*                              INCLUDE FILES                               */
/****************************************************************************/

#include "zw_config_rf.h"
#include <stdbool.h>
#include <stdint.h>
#include "SizeOf.h"
#include "Assert.h"
#include <MfgTokens.h>
#include "DebugPrintConfig.h"
//#define DEBUGPRINT
#include "DebugPrint.h"
#include "config_app.h"
#include <ZAF_file_ids.h>
#include "ZAF_nvm_app.h"
#include <ZW_slave_api.h>
#include <ZW_classcmd.h>
#include <ZW_TransportLayer.h>

#include <string.h>
#include <ev_man.h>
```

CDT Build Console [zwave_soc_switch_on_off]

```
.debug_loc                  323561         0
.debug_aranges               14176         0
.debug_ranges                27592         0
.debug_macro                482947         0
.debug_line                 622431         0
.debug_str                 2488418         0
Total                     15693194




15:26:34 Build Finished. 0 errors, 0 warnings. (took 47s.445ms)
```

Figure 16: Sample Application Build Completed and Binaries Are Generated

6.2 Flashing the Bootloader

If you are using 700 Series hardware, skip ahead to Section 6.3.

800 Series radio boards require a Z-Wave specific bootloader before flashing an application. *Note:* this does not apply to demos, as the bootloader is included in the demo binaries.

To flash a bootloader to your 800 Series device, perform the following steps, as shown in Figure 17:

1) Navigate to the Launcher Perspective.
2) Select your device in the Debug Adapters view
3) Go to the '*Example Projects & Demos*' tab.
4) Find the Z-Wave OTA bootloader. Use the filtering options to speed up this process.
5) Click '*Run*' and wait until the bootloader is uploaded to the device.

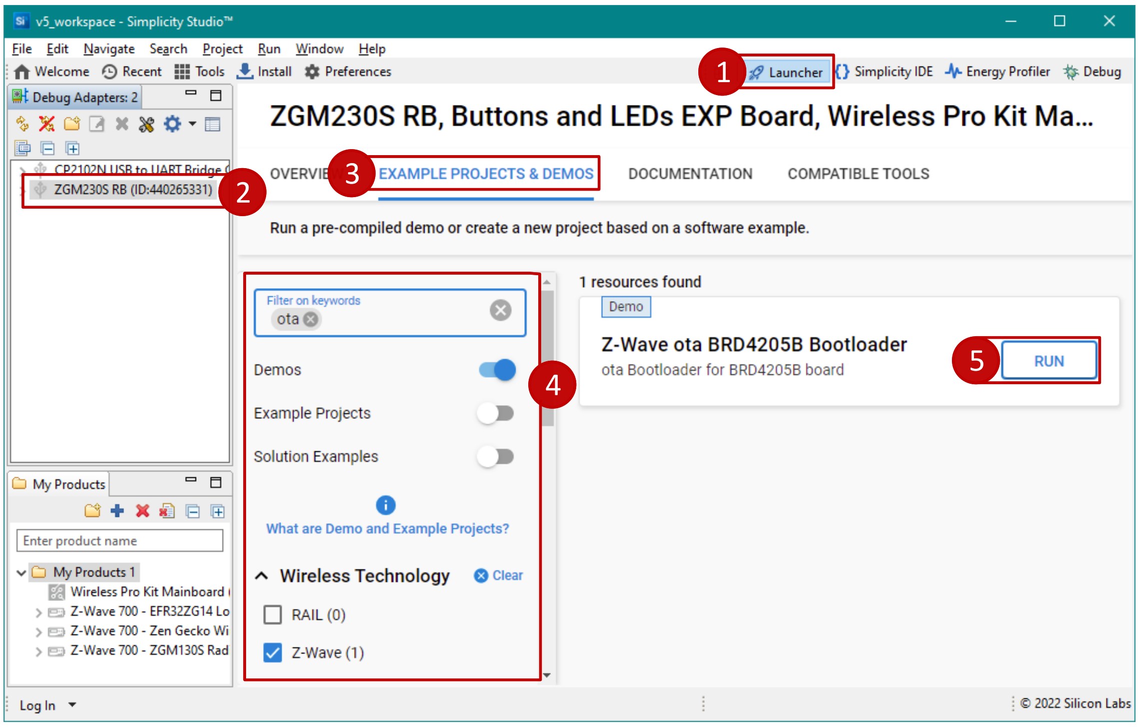

v5_workspace - Simplicity Studio™
File Edit Navigate Search Project Run Window Help
Welcome Recent Tools Install Preferences
1
Launcher
Simplicity IDE
Energy Profiler
Debug
Debug Adapters: 2
CP2102N USB to UART Bridge
ZGM230S RB (ID:440265331)
2
ZGM230S RB, Buttons and LEDs EXP Board, Wireless Pro Kit Ma...
OVERVIEW
3
EXAMPLE PROJECTS & DEMOS
DOCUMENTATION
COMPATIBLE TOOLS
Run a pre-compiled demo or create a new project based on a software example.
Filter on keywords
ota
Demos
Example Projects
Solution Examples
4
What are Demo and Example Projects?
Wireless Technology
Clear
RAIL (0)
Z-Wave (1)
1 resources found
Demo
Z-Wave ota BRD4205B Bootloader
ota Bootloader for BRD4205B board
5
RUN
My Products
Enter product name
My Products 1
Wireless Pro Kit Mainboard
Z-Wave 700 - EFR32ZG14 Lo
Z-Wave 700 - Zen Gecko Wi
Z-Wave 700 - ZGM130S Rad
Log In
© 2022 Silicon Labs

Figure 17: Flashing the Bootloader

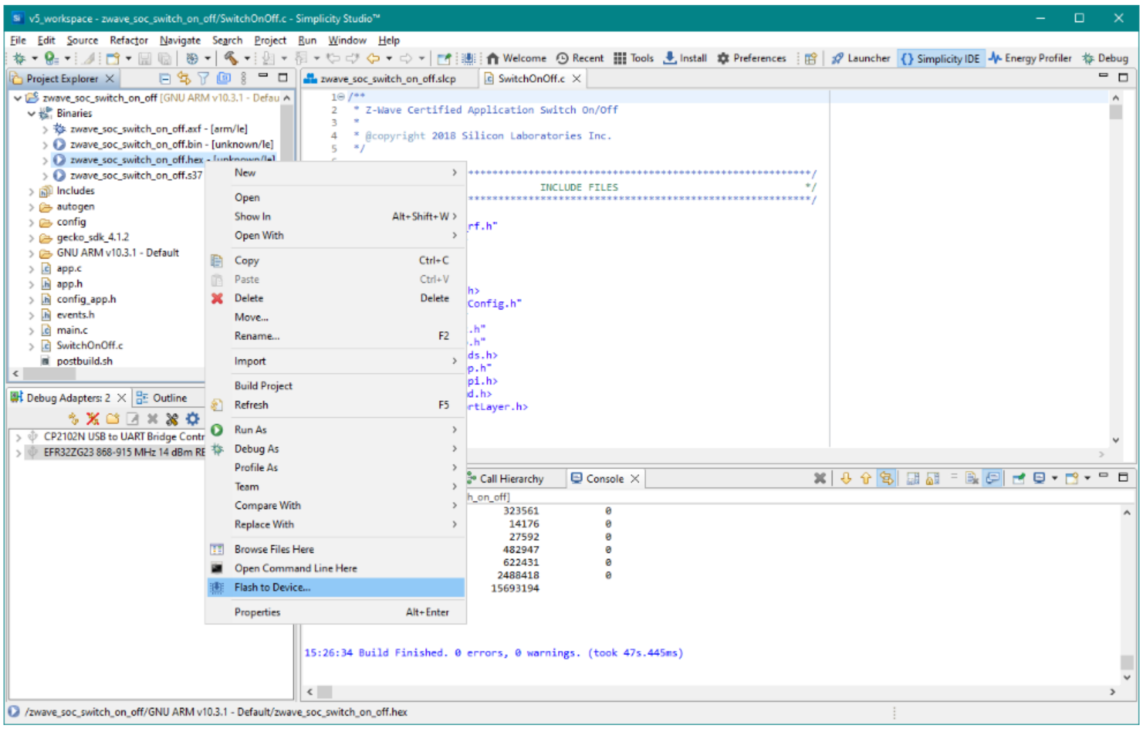
To avoid having to flash the bootloader manually, you can create a *Solution Example* instead of an *Example Project* (see the beginning of Section 6 for more details about Solution Examples).

For further details about the bootloader, please refer to [13].

Continue by returning to the Simplicity IDE perspective.

6.3 Flashing the Application

Expand the ‘*Binaries*’ folder and right click on the *.hex file, then select ‘*Flash to Device…*’ (see Figure 18).


v5_workspace - zwave_soc_switch_on_off/SwitchOnOff.c - Simplicity Studio™
File Edit Source Refactor Navigate Search Project Run Window Help
Welcome Recent Tools Install Preferences Launcher Simplicity IDE Energy Profiler Debug
Project Explorer
zwave_soc_switch_on_off [GNU ARM v10.3.1 - Defau
Binaries
zwave_soc_switch_on_off.axf - [arm/le]
zwave_soc_switch_on_off.bin - [unknown/le]
zwave_soc_switch_on_off.hex - [unknown/le]
zwave_soc_switch_on_off.s37
Includes
autogen
config
gecko_sdk_4.1.2
GNU ARM v10.3.1 - Default
app.c
app.h
config_app.h
events.h
main.c
SwitchOnOff.c
postbuild.sh
Debug Adapters: 2
Outline
CP2102N USB to UART Bridge Contr
EFR32ZG23 868-915 MHz 14 dBm RE
New
Open
Show In Alt+Shift+W
Open With
Copy Ctrl+C
Paste Ctrl+V
Delete Delete
Move...
Rename... F2
Import
Build Project
Refresh F5
Run As
Debug As
Profile As
Team
Compare With
Replace With
Browse Files Here
Open Command Line Here
Flash to Device...
Properties Alt+Enter
zwave_soc_switch_on_off.slcp
SwitchOnOff.c
Call Hierarchy
Console
/zwave_soc_switch_on_off/GNU ARM v10.3.1 - Default/zwave_soc_switch_on_off.hex

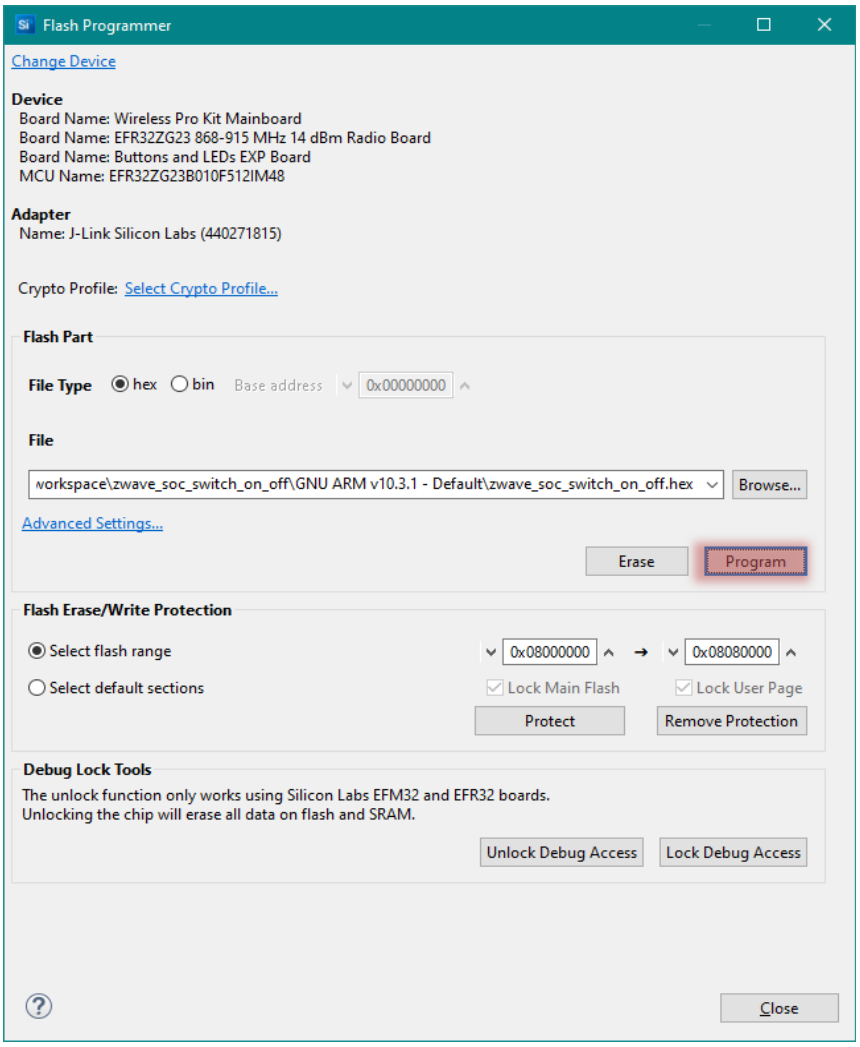
```
/**
 * Z-Wave Certified Application Switch On/Off
 *
 * @copyright 2018 Silicon Laboratories Inc.
 */
```

```
h_on_off]
    323561        0
     14176        0
     27592        0
    482947        0
    622431        0
   2488418        0
  15693194

15:26:34 Build Finished. 0 errors, 0 warnings. (took 47s.445ms)
```

Figure 18: Select Image to Flash

Select the connected hardware in the pop-up window. If you followed the instructions in this guide, the ‘*Flash Programmer*’ should now be pre-filled with all the needed data, and you are ready to click on ‘*Program*’. Refer to Figure 19.



Device
Board Name: Wireless Pro Kit Mainboard
Board Name: EFR32ZG23 868-915 MHz 14 dBm Radio Board
Board Name: Buttons and LEDs EXP Board
MCU Name: EFR32ZG23B010F512IM48

Adapter
Name: J-Link Silicon Labs (440271815)

Crypto Profile: Select Crypto Profile...

Flash Part
File Type [✓ hex] bin Base address 0x00000000
File
workspace\zwave_soc_switch_on_off\GNU ARM v10.3.1 - Default\zwave_soc_switch_on_off.hex

Flash Erase/Write Protection
[✓ Select flash range] 0x08000000 → 0x08080000
Select default sections
Lock Main Flash ✓ Lock User Page ✓

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

Figure 19: Flash Programmer with Selected Device and File to Be Flashed

After a short while, the Z-Wave sample application will be flashed to your end device. Include it in a Z-Wave network using the PC Controller to start testing the functionalities (See Section 7 of this document).

Refer to [1] for instructions on how to operate the sample application.
Refer to [10] for instructions on how to read out the DSK value.

The following sections (6.4 & 6.5) describe procedures that you do not normally have to perform, but of which you should be aware.

6.4 Changing the Region

If you need to change the frequency at which the module operates, you will need to set a special token on your device using Simplicity Commander.

1. Locate your Simplicity Commander installation in Simplicity Studio by searching for commander.exe file, which is typically in C:\SiliconLabs\SimplicityStudio\v5\developer\adapter_packs\commander
2. Open your terminal of choice inside the folder and execute the following command, **substituting your device's serial number in the '*serialno*' option**:

```
commander.exe flash --serialno 000000000 --tokengroup znet --token
MFG_ZWAVE_COUNTRY_FREQ:!ERASE!
```

Note: For running Z-Wave applications in a custom region, i.e. to run applications that do not have corresponding demos shipped with Simplicity Studio, please write the region value to MFG Token.
For example, for ANZ, use

```
commander.exe flash --serialno 000000000 --tokengroup znet --token
MFG_ZWAVE_COUNTRY_FREQ:0x02
```

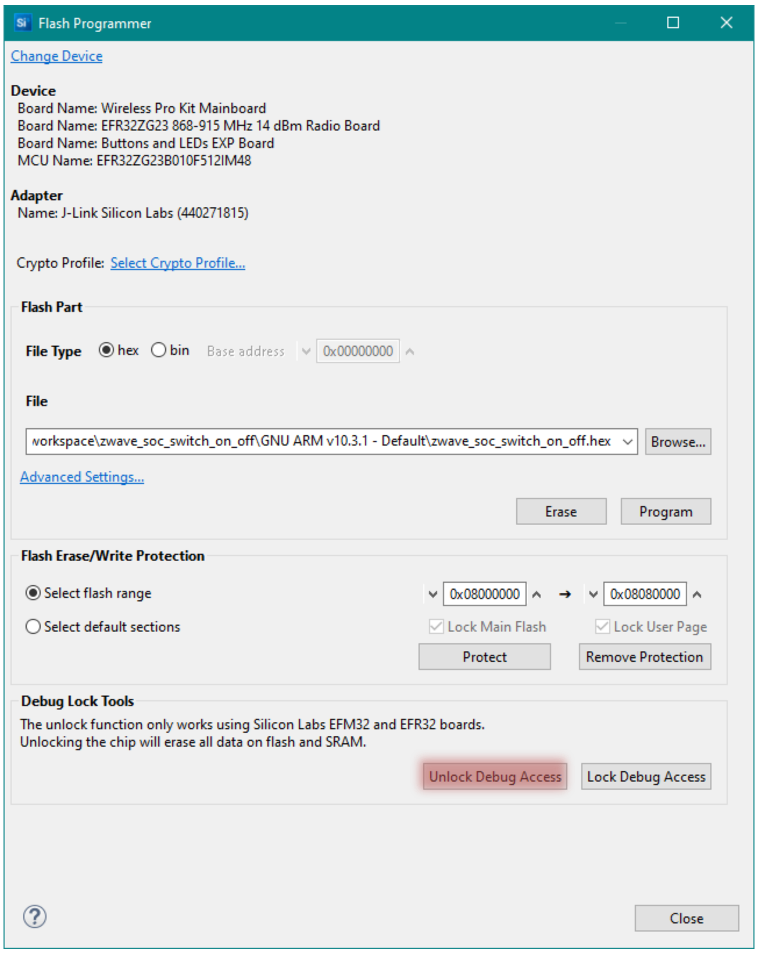
The hexadecimal values for each of the regions are defined here:
`<sdk folder>\protocol\z-wave\PAL\inc\zpal_radio.h`.

6.5 Reprogramming a Sleeping Device

After a device has been programmed with an application capable of deep sleep (such as '*Sensor PIR*') the programmer will throw an error when you attempt to flash it again. The reason for this is because the processor has been put into deep sleep mode (EM4) and the programmer cannot wake it.

You can recover the device by performing what's called a *'Debug Unlock'*. This will completely erase the device and allow you to connect to it again. This process can be performed by either Simplicity Commander or the Flash Programmer within Simplicity Studio.

Using the Flash Programmer, click on *'Unlock Debug Access'* and then flash the device with the new firmware.



Device
Board Name: Wireless Pro Kit Mainboard
Board Name: EFR32ZG23 868-915 MHz 14 dBm Radio Board
Board Name: Buttons and LEDs EXP Board
MCU Name: EFR32ZG23B010F512IM48

Adapter
Name: J-Link Silicon Labs (440271815)

Crypto Profile: Select Crypto Profile...

Flash Part
File Type [✓ hex] bin Base address 0x00000000
File
workspace\zwave_soc_switch_on_off\GNU ARM v10.3.1 - Default\zwave_soc_switch_on_off.hex

Flash Erase/Write Protection
[✓ Select flash range] 0x08000000 → 0x08080000
Select default sections — Lock Main Flash ✓, Lock User Page ✓

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

Figure 20: Recover a Sleeping Device With 'Unlock Debug Access'

7 Run Your First Z-Wave Sample Application

This section describes how to run and test the sample application you have just configured and flashed with the help of two commonly used Z-Wave development tools: the **Z-Wave PC Controller** and the **Z-Wave Zniffer**. These come as part of the Simplicity SDK suite (described in Section 5).

7.1 Z-Wave PC Controller

The Z-Wave PC Controller is a client application for communicating with Z-Wave and Z-Wave Long Range nodes, like switches and sensors. The application requires a device running the *Z-Wave Serial API Controller* firmware (either a dedicated USB stick included in your Development Kit or a compatible Radio Board) connected to a USB port on the PC.

If you have a dedicated Z-Wave Serial API Controller device, such as the UZB-7 Z-Wave 700 Stick Bridge Module, connect it to your computer.
Otherwise, configure a spare Radio Board for use with the Z-Wave PC Controller by connecting it to your computer and flashing the *'Z-Wave – NCP Serial API Controller'* demo firmware onto it (see Section 6 for instructions on how to flash a pre-built demo).

The PC Controller is often used in the development of a new Z-Wave end device. The device can be included in the PC Controller's Z-Wave network, which can then be used to test the end device by sending various commands to test the implemented functionality of the end device.

This getting started guide will only cover the very basics of this tool. Refer to the manual [2] to learn about all the features of this tool.

1) Start by connecting the hardware running the Z-Wave Serial API Controller firmware to your computer.
2) Open the PC Controller from the *'Show Compatible Tools'* dialog in the Toolbar.

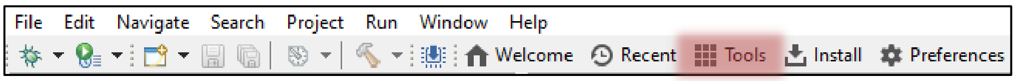
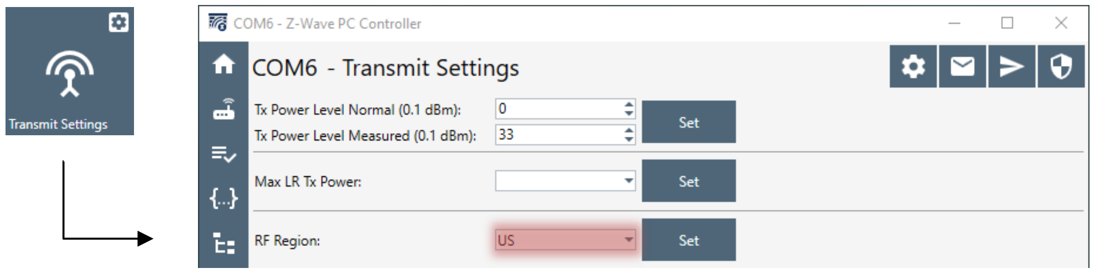


Figure 21: The Location of the Show Compatible Tools Dialog in the Toolbar

3) Click on the *'Settings'* ⚙ to select the correct COM port. You can click the *'Detect'* button to have the program automatically find the Controller. Confirm your selection by pressing *'OK'*.

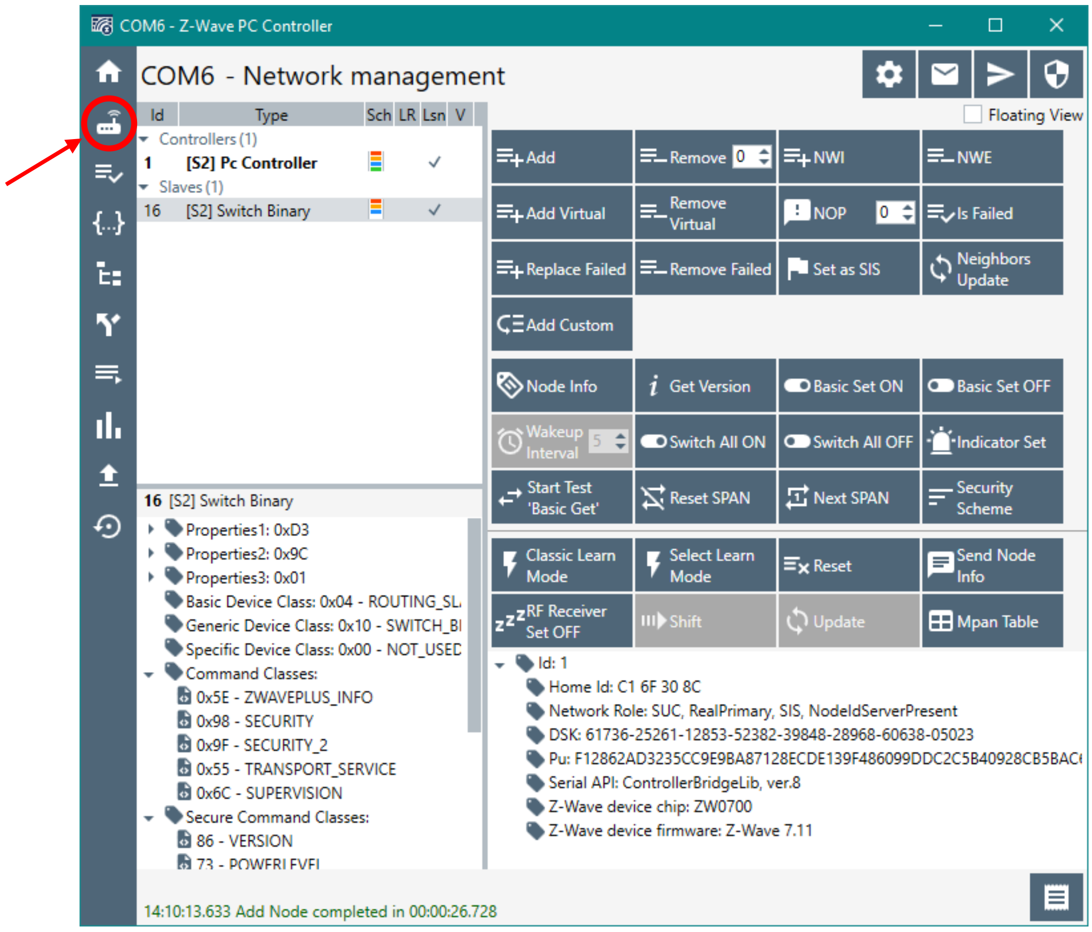
4) The PC Controller might be set to use an incompatible RF region by default. If your end device is programmed for another region, change the PC Controller's RF Region at the Transmit Settings.



Tx Power Level Normal (0.1 dBm): 0
Tx Power Level Measured (0.1 dBm): 33
Max LR Tx Power:
RF Region: US

Figure 22: Setting the RF Region in the Z-Wave PC Controller

5) Click on Network Management when connection to the UZB Controller has been established.



Id	Type	Sch	LR	Lsn	V
Controllers (1)					
1	[S2] Pc Controller			✓	
Slaves (1)					
16	[S2] Switch Binary			✓	

16 [S2] Switch Binary
- Properties1: 0xD3
- Properties2: 0x9C
- Properties3: 0x01
- Basic Device Class: 0x04 - ROUTING_SL
- Generic Device Class: 0x10 - SWITCH_BI
- Specific Device Class: 0x00 - NOT_USED
- Command Classes:
  - 0x5E - ZWAVEPLUS_INFO
  - 0x98 - SECURITY
  - 0x9F - SECURITY_2
  - 0x55 - TRANSPORT_SERVICE
  - 0x6C - SUPERVISION
- Secure Command Classes:
  - 86 - VERSION
  - 73 - POWERLEVEL

- Id: 1
  - Home Id: C1 6F 30 8C
  - Network Role: SUC, RealPrimary, SIS, NodeIdServerPresent
  - DSK: 61736-25261-12853-52382-39848-28968-60638-05023
  - Pu: F12862AD3235CC9E9BA87128ECDE139F486099DDC2C5B40928CB5BAC
  - Serial API: ControllerBridgeLib, ver.8
  - Z-Wave device chip: ZW0700
  - Z-Wave device firmware: Z-Wave 7.11

14:10:13.633 Add Node completed in 00:00:26.728

Figure 23: PC Controller With 1 End Device Added

6) From this view you can add, remove, and send basic commands to an end device. Refer to the following sections in the PC Controller Manual [2]:

   3.2 Network Management View

   4.2.1 How to Add a Node

   4.2.3 How to Remove a Node

   For adding a node, click on ‘*Add*’ and activate learn mode on your end device.
   On your end device, press the button labeled ‘*BTN1’* on the WSTK (for SDK versions 7.23 and above) or on the Expansion Board (for SDK versions below 7.23).
   Refer to [1] for instructions on how to use the buttons.

   A pop-up window will appear listing the available security classes. Click ‘*OK*’.

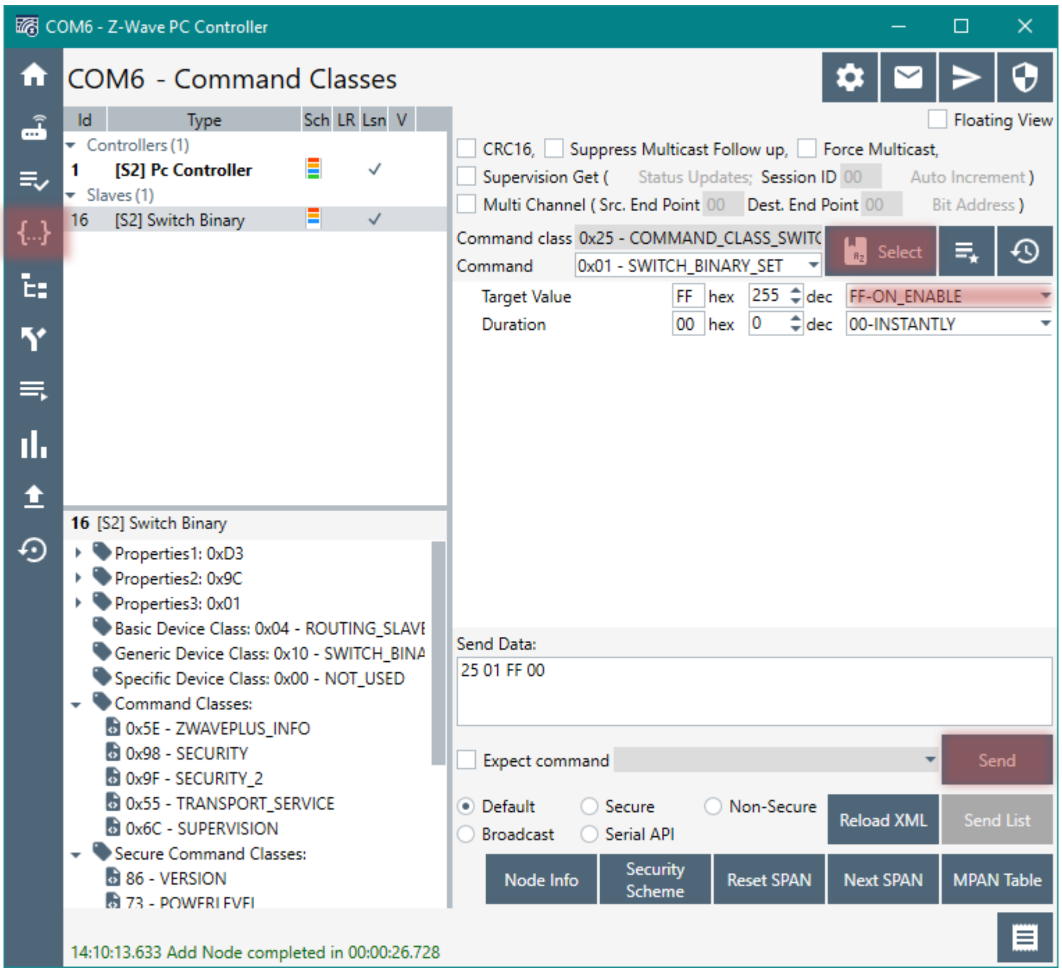
   For the next step, you will need the end device’s ‘*Device Specific Key*’, or ‘*DSK*’.
   Refer to Section 10 of this document for instructions on how to obtain this key.
   Enter the first five digits of the DSK and click ‘*OK*’.

   The included nodes will appear in the *‘End Devices’* section with a Node ID.

7) In addition to the basic functionality, it is also worth knowing from the beginning how to send various commands using the Command Class View. Refer to the following section in the PC Controller Manual [2]:

   4.4 Command Class View

   For sending a command to turn on the included switch, go to *Command Classes* view.
   Then click ‘Select’ and pick a supported command class, in this case
   ver.2 – Binary Switch » Switch Binary Set. Refer to Figure 24.
   Set the value and click on *Send*. The value on the switch should now have been set accordingly, which is indicated by *LED0* on the WSTK (or on the EXP board for SDK versions below 7.23).



COM6 - Z-Wave PC Controller

COM6 - Command Classes

Id	Type	Sch	LR	Lsn	V
Controllers (1)					
1	[S2] Pc Controller				✓
Slaves (1)					
16	[S2] Switch Binary				✓

16 [S2] Switch Binary
- Properties1: 0xD3
- Properties2: 0x9C
- Properties3: 0x01
- Basic Device Class: 0x04 - ROUTING_SLAVE
- Generic Device Class: 0x10 - SWITCH_BINA
- Specific Device Class: 0x00 - NOT_USED
- Command Classes:
  - 0x5E - ZWAVEPLUS_INFO
  - 0x98 - SECURITY
  - 0x9F - SECURITY_2
  - 0x55 - TRANSPORT_SERVICE
  - 0x6C - SUPERVISION
- Secure Command Classes:
  - 86 - VERSION
  - 73 - POWERLEVEL

Floating View

CRC16, Suppress Multicast Follow up, Force Multicast,
Supervision Get (Status Updates; Session ID 00 Auto Increment)
Multi Channel (Src. End Point 00 Dest. End Point 00 Bit Address)

Command class 0x25 - COMMAND_CLASS_SWITC
Command 0x01 - SWITCH_BINARY_SET

Target Value	FF	hex	255	dec	FF-ON_ENABLE
Duration	00	hex	0	dec	00-INSTANTLY

Send Data:
25 01 FF 00

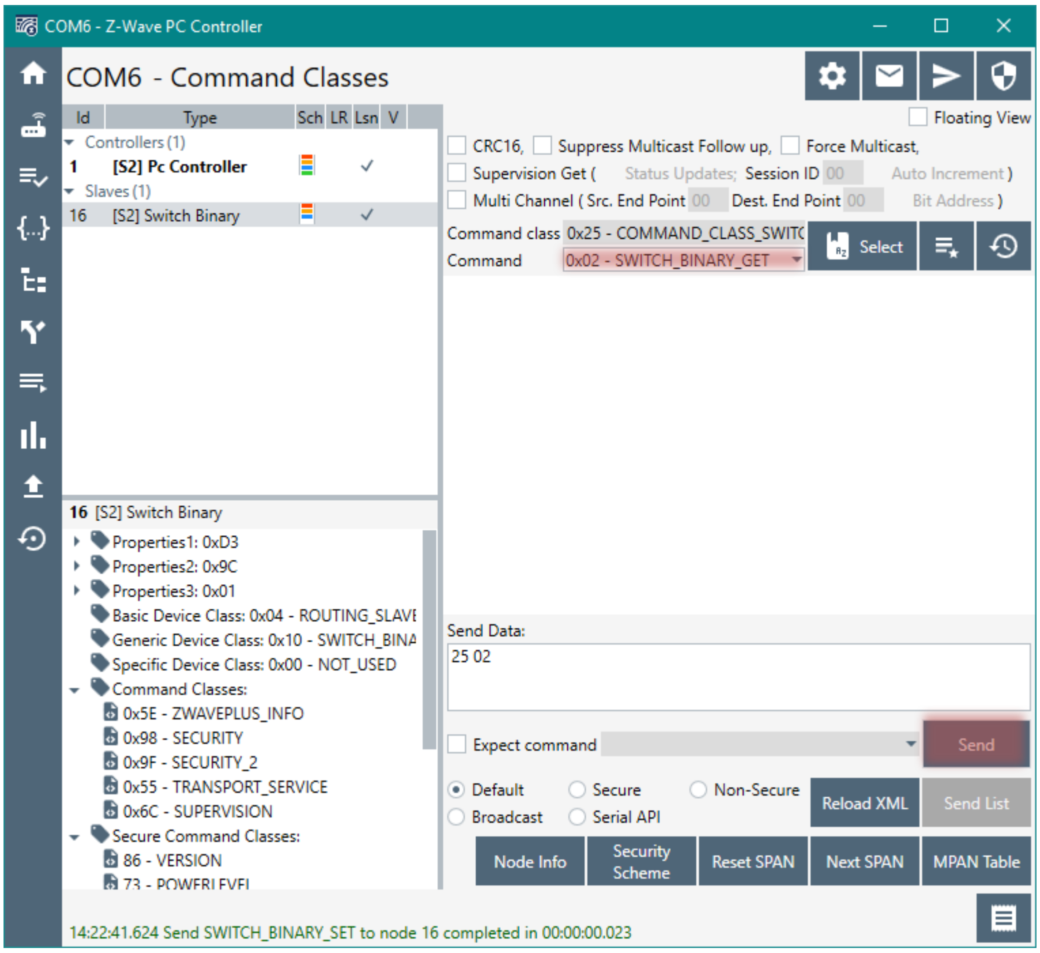
Expect command

[✓ Default] Secure Non-Secure
Broadcast Serial API

14:10:13.633 Add Node completed in 00:00:26.728

Figure 24: PC Controller, Command Classes View—Send Switch On Command

8) Send a Get Command to verify the switch was set accordingly. Refer to Figure 25.



```
14:22:41.624 Send SWITCH_BINARY_SET to node 16 completed in 00:00:00.023
```

Figure 25: PC Controller, Command Classes View—Send Switch Get Command

Use Zniffer to view the response returned by the end device for the Get Command – see Section ‘7.2 Z-Wave Zniffer’.

7.2 Z-Wave Zniffer

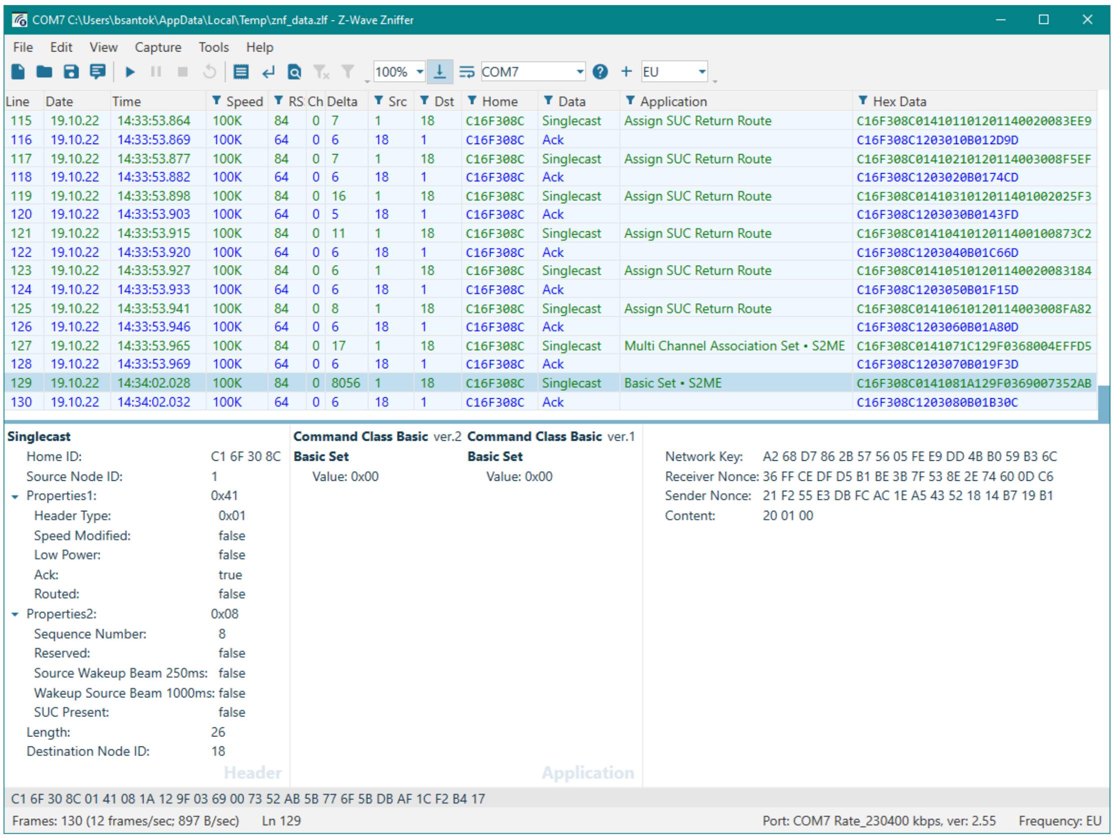
The Z-Wave Zniffer application is a development tool for capturing Z-Wave RF communication and presenting the frames in a graphical user interface on a PC.

The Zniffer tool is a passive ‘listener’ to the Z-Wave network traffic and will only display the RF communications taking place within direct RF range.

The tool shows the node ID of the Source and Destination for the communication, the type of frame being sent, and the application content, i.e., the specific command that is being sent.

This getting started guide will only cover the very basics of this tool. Refer to the manual [3] to learn about all the features of this tool.

1) Start by connecting the Zniffer USB to your computer. For details, see Appendix B: Configuring the Development Kit as a Network Sniffer.
2) Install the driver when prompted.
3) Open Zniffer from the ‘*Show Compatible Tools’* dialog (see Section 7.1, step 2).
4) Click on ‘*Detect Zniffer Modules*’ in the tools bar. Then select the correct COM Port and frequency to listen in the dropdown menu.
5) When ready, click on ‘*Start capturing frames*’ .


COM7 C:\Users\bsantok\AppData\Local\Temp\znf_data.zlf - Z-Wave Zniffer
File
Edit
View
Capture
Tools
Help
100%
COM7
EU

Line	Date	Time	Speed	RS	Ch	Delta	Src	Dst	Home	Data	Application	Hex Data
115	19.10.22	14:33:53.864	100K	84	0	7	1	18	C16F308C	Singlecast	Assign SUC Return Route	C16F308C01410110120114002008 3EE9
116	19.10.22	14:33:53.869	100K	64	0	6	18	1	C16F308C	Ack		C16F308C1203010B012D9D
117	19.10.22	14:33:53.877	100K	84	0	7	1	18	C16F308C	Singlecast	Assign SUC Return Route	C16F308C014102101201140030 08F5EF
118	19.10.22	14:33:53.882	100K	64	0	6	18	1	C16F308C	Ack		C16F308C1203020B0174CD
119	19.10.22	14:33:53.898	100K	84	0	16	1	18	C16F308C	Singlecast	Assign SUC Return Route	C16F308C01410310120114010020 25F3
120	19.10.22	14:33:53.903	100K	64	0	5	18	1	C16F308C	Ack		C16F308C1203030B0143FD
121	19.10.22	14:33:53.915	100K	84	0	11	1	18	C16F308C	Singlecast	Assign SUC Return Route	C16F308C01410410120114001008 73C2
122	19.10.22	14:33:53.920	100K	64	0	6	18	1	C16F308C	Ack		C16F308C1203040B01C66D
123	19.10.22	14:33:53.927	100K	84	0	6	1	18	C16F308C	Singlecast	Assign SUC Return Route	C16F308C01410510120114002008 3184
124	19.10.22	14:33:53.933	100K	64	0	6	18	1	C16F308C	Ack		C16F308C1203050B01F15D
125	19.10.22	14:33:53.941	100K	84	0	8	1	18	C16F308C	Singlecast	Assign SUC Return Route	C16F308C01410610120114003008 FA82
126	19.10.22	14:33:53.946	100K	64	0	6	18	1	C16F308C	Ack		C16F308C1203060B01A80D
127	19.10.22	14:33:53.965	100K	84	0	17	1	18	C16F308C	Singlecast	Multi Channel Association Set • S2ME	C16F308C0141071C129F036800 4EFFD5
128	19.10.22	14:33:53.969	100K	64	0	6	18	1	C16F308C	Ack		C16F308C1203070B019F3D
129	19.10.22	14:34:02.028	100K	84	0	8056	1	18	C16F308C	Singlecast	Basic Set • S2ME	C16F308C0141081A129F036900 7352AB
130	19.10.22	14:34:02.032	100K	64	0	6	18	1	C16F308C	Ack		C16F308C1203080B01B30C

Singlecast

Home ID:	C1 6F 30 8C
Source Node ID:	1
Properties1:	0x41
Header Type:	0x01
Speed Modified:	false
Low Power:	false
Ack:	true
Routed:	false
Properties2:	0x08
Sequence Number:	8
Reserved:	false
Source Wakeup Beam 250ms:	false
Wakeup Source Beam 1000ms:	false
SUC Present:	false
Length:	26
Destination Node ID:	18

Header

Command Class Basic ver.2
Basic Set
Value: 0x00

Command Class Basic ver.1
Basic Set
Value: 0x00

Application

Network Key:	A2 68 D7 86 2B 57 56 05 FE E9 DD 4B B0 59 B3 6C
Receiver Nonce:	36 FF CE DF D5 B1 BE 3B 7F 53 8E 2E 74 60 0D C6
Sender Nonce:	21 F2 55 E3 DB FC AC 1E A5 43 52 18 14 B7 19 B1
Content:	20 01 00

```
C1 6F 30 8C 01 41 08 1A 12 9F 03 69 00 73 52 AB 5B 77 6F 5B DB AF 1C F2 B4 17
Frames: 130 (12 frames/sec; 897 B/sec)    Ln 129    Port: COM7 Rate_230400 kbps, ver: 2.55    Frequency: EU
```

Figure 26: Zniffer Trace Showing a Decrypted S2 Frame for a 'Switch Binary Get'

6) From this view you can now see the Z-Wave frames being sent/received in the network. Refer to the following sections in the Zniffer Manual [3] for a suggested next step:

    4.1 Layout of the Zniffer Main Window

    5 Capturing Live RF Traffic

    7.5 Working with Encrypted Frames

    By clicking on '*Home ID*' you can filter the trace to only show the frames of interest, in case of multiple networks.

    To decrypt a message, you must enter the Security keys, which can be found in the PC Controller under '*Security Settings*' 🛡.
    The initial handshake during the inclusion process must be captured in the Zniffer for it to be able to decrypt the messages.

8 Debug a Z-Wave Application

8.1 Debugger

The debugger supplied with Simplicity Studio is based on the Eclipse debugger. It is a fully fledged debugger that offers the ability to step through code, set breakpoints, and examine memory, variables, and registers.

Install the '*Z-Wave Debug*' component for your project, as shown in Figure 27.

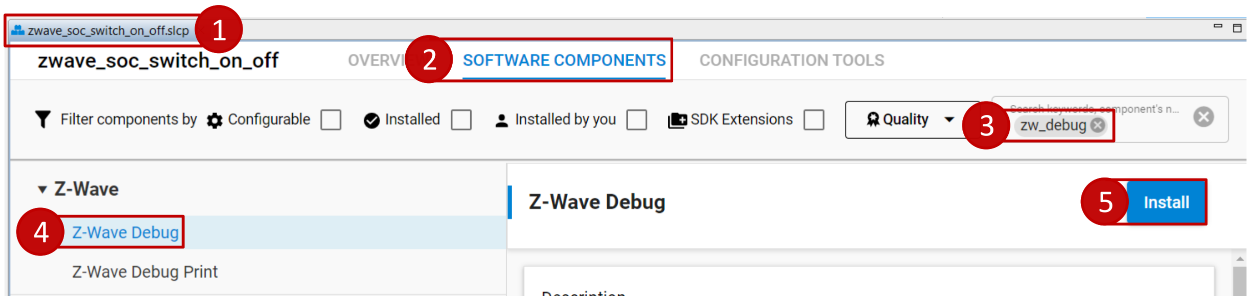


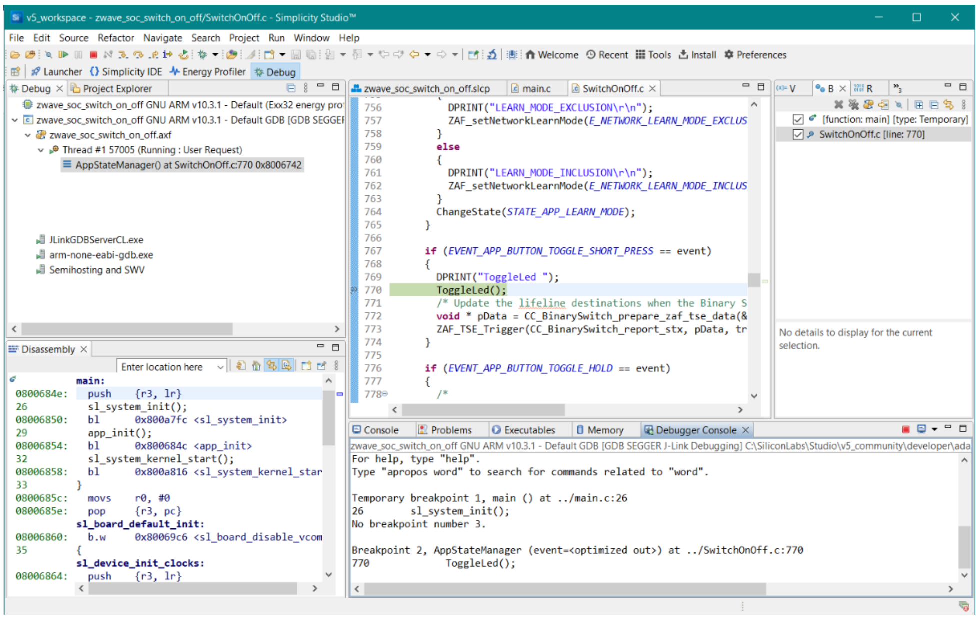
Figure 27: Installing the Z-Wave Debug component

Make sure your application can be built without errors.
If the firmware's size becomes too large with the component installed, refer to Appendix C: Decreasing the firmware's size.

From the Toolbar, click on the '*Debug*' button. The code will now be built with debugging enabled and automatically flashed to your device. When finished, the Debug perspective will open.

To set a breakpoint, double-click in the blue bar to the left of the code editor or right-click on a line of code and select '*Add Breakpoint*'. Breakpoints can be managed using the *Breakpoints* view in the Debug perspective. Register contents are viewable and editable using the *Registers* view. Memory can be accessed using the *Memory* view.

In Figure 28, the sample application ‘Switch On/Off’ is flashed to the device in debug mode. A breakpoint is then set near the ‘*ToggleLed*’ function. When the user presses on the button *‘BTN0’*, the code will halt execution at the breakpoint and show the relevant information.


v5_workspace - zwave_soc_switch_on_off/SwitchOnOff.c - Simplicity Studio™
File Edit Source Refactor Navigate Search Project Run Window Help
Welcome Recent Tools Install Preferences
Launcher Simplicity IDE Energy Profiler Debug
Debug Project Explorer
zwave_soc_switch_on_off.slcp main.c SwitchOnOff.c
Disassembly
Enter location here
Console Problems Executables Memory Debugger Console

```
zwave_soc_switch_on_off GNU ARM v10.3.1 - Default (Exx32 energy pro
zwave_soc_switch_on_off GNU ARM v10.3.1 - Default GDB [GDB SEGGEF
  zwave_soc_switch_on_off.axf
    Thread #1 57005 (Running : User Request)
      AppStateManager() at SwitchOnOff.c:770 0x8006742
  JLinkGDBServerCL.exe
  arm-none-eabi-gdb.exe
  Semihosting and SWV
```

```
          main:
0800684e:   push    {r3, lr}
26          sl_system_init();
08006850:   bl      0x800a7fc <sl_system_init>
29          app_init();
08006854:   bl      0x800684c <app_init>
32          sl_system_kernel_start();
08006858:   bl      0x800a816 <sl_system_kernel_star
33        }
0800685c:   movs    r0, #0
0800685e:   pop     {r3, pc}
          sl_board_default_init:
08006860:   b.w     0x80069c6 <sl_board_disable_vcom
35        {
          sl_device_init_clocks:
08006864:   push    {r3, lr}
```

```
      DPRINT("LEARN_MODE_EXCLUSION\r\n");
      ZAF_setNetworkLearnMode(E_NETWORK_LEARN_MODE_EXCLUS
    }
    else
    {
      DPRINT("LEARN_MODE_INCLUSION\r\n");
      ZAF_setNetworkLearnMode(E_NETWORK_LEARN_MODE_INCLUS
    }
    ChangeState(STATE_APP_LEARN_MODE);
  }

  if (EVENT_APP_BUTTON_TOGGLE_SHORT_PRESS == event)
  {
    DPRINT("ToggleLed ");
    ToggleLed();
    /* Update the lifeline destinations when the Binary S
    void * pData = CC_BinarySwitch_prepare_zaf_tse_data(&
    ZAF_TSE_Trigger(CC_BinarySwitch_report_stx, pData, tr
  }

  if (EVENT_APP_BUTTON_TOGGLE_HOLD == event)
  {
    /*
```

```
[function: main] [type: Temporary]
SwitchOnOff.c [line: 770]
No details to display for the current selection.
```

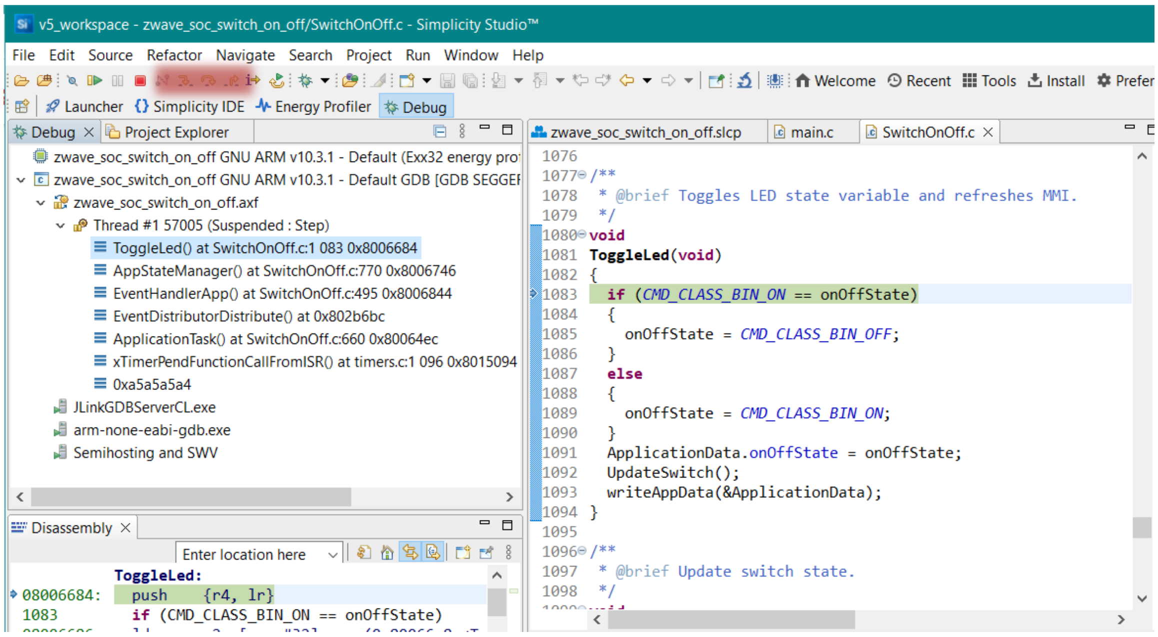
```
zwave_soc_switch_on_off GNU ARM v10.3.1 - Default GDB [GDB SEGGER J-Link Debugging] C:\SiliconLabs\Studio\v5_community\developer\ada
For help, type "help".
Type "apropos word" to search for commands related to "word".

Temporary breakpoint 1, main () at ../main.c:26
26          sl_system_init();
No breakpoint number 3.

Breakpoint 2, AppStateManager (event=<optimized out>) at ../SwitchOnOff.c:770
770             ToggleLed();
```

Figure 28: Debugging Switch On/Off

By using the ‘Step Over’ and ‘Step Into’ functionalities of the debugger, it is possible to follow the flow of the application.


v5_workspace - zwave_soc_switch_on_off/SwitchOnOff.c - Simplicity Studio™
File Edit Source Refactor Navigate Search Project Run Window Help
Welcome Recent Tools Install Prefer
Launcher Simplicity IDE Energy Profiler Debug
Debug Project Explorer
zwave_soc_switch_on_off.slcp main.c SwitchOnOff.c
Disassembly
Enter location here

```
zwave_soc_switch_on_off GNU ARM v10.3.1 - Default (Exx32 energy pro
zwave_soc_switch_on_off GNU ARM v10.3.1 - Default GDB [GDB SEGGEF
  zwave_soc_switch_on_off.axf
    Thread #1 57005 (Suspended : Step)
      ToggleLed() at SwitchOnOff.c:1 083 0x8006684
      AppStateManager() at SwitchOnOff.c:770 0x8006746
      EventHandlerApp() at SwitchOnOff.c:495 0x8006844
      EventDistributorDistribute() at 0x802b6bc
      ApplicationTask() at SwitchOnOff.c:660 0x80064ec
      xTimerPendFunctionCallFromISR() at timers.c:1 096 0x8015094
      0xa5a5a5a4
  JLinkGDBServerCL.exe
  arm-none-eabi-gdb.exe
  Semihosting and SWV
```

```
/**
 * @brief Toggles LED state variable and refreshes MMI.
 */
void
ToggleLed(void)
{
  if (CMD_CLASS_BIN_ON == onOffState)
  {
    onOffState = CMD_CLASS_BIN_OFF;
  }
  else
  {
    onOffState = CMD_CLASS_BIN_ON;
  }
  ApplicationData.onOffState = onOffState;
  UpdateSwitch();
  writeAppData(&ApplicationData);
}

/**
 * @brief Update switch state.
 */
```

```
          ToggleLed:
08006684:   push    {r4, lr}
1083        if (CMD_CLASS_BIN_ON == onOffState)
```

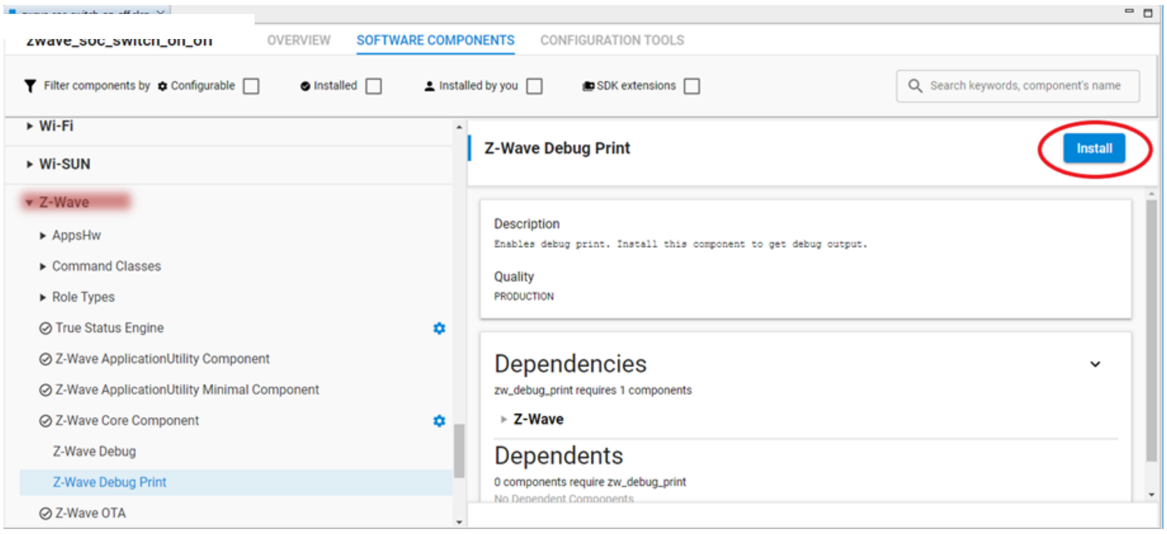
Figure 29: Using Step Into to Debug the Functionalities of ToggleLed

Note: This feature does not work on sleeping devices using deep sleep EM4 mode (Sensor PIR Sample App).

8.2 Serial Debug

The Application Framework also features a serial debug connection, enabling you to write statuses, states, and values to a terminal.

To enable the serial debugger, first install the '*Z-Wave Debug Print*' software component as shown in Figure 30.



Description
Enables debug print. Install this component to get debug output.

Quality
PRODUCTION

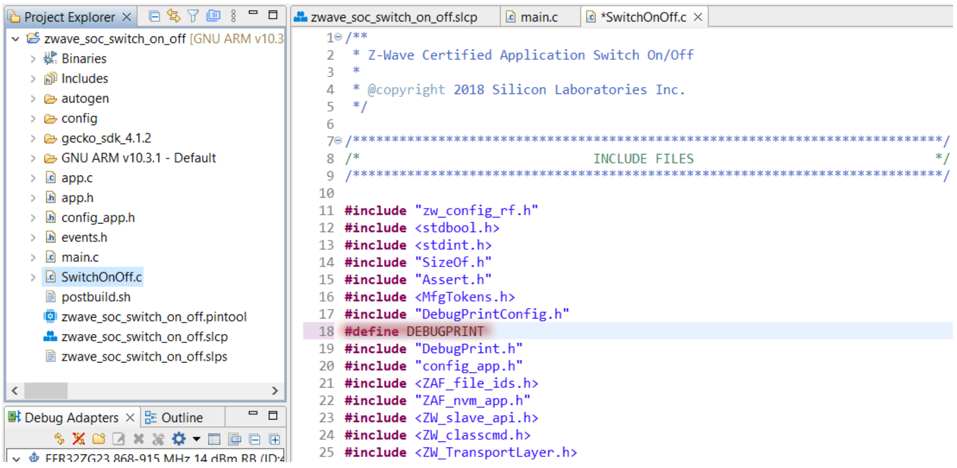
Dependencies
zw_debug_print requires 1 components
Z-Wave

Dependents
0 components require zw_debug_print
No Dependent Components

Figure 30: Install the Z-Wave Debug Print Component

Then uncomment the *define* line in the Include-section, as shown in Figure 31.

Note: SwitchOnOff.c was renamed to app.c in version 7.20 of the SDK.


Project Explorer
zwave_soc_switch_on_off [GNU ARM v10.3
Binaries
Includes
autogen
config
gecko_sdk_4.1.2
GNU ARM v10.3.1 - Default
app.c
app.h
config_app.h
events.h
main.c
SwitchOnOff.c
postbuild.sh
zwave_soc_switch_on_off.pintool
zwave_soc_switch_on_off.slcp
zwave_soc_switch_on_off.slps
Debug Adapters
Outline
EFR32ZG23 868-915 MHz 14 dBm RB (ID:
zwave_soc_switch_on_off.slcp
main.c
*SwitchOnOff.c

```
/**
 * Z-Wave Certified Application Switch On/Off
 *
 * @copyright 2018 Silicon Laboratories Inc.
 */

/****************************************************************************/
/*                              INCLUDE FILES                               */
/****************************************************************************/

#include "zw_config_rf.h"
#include <stdbool.h>
#include <stdint.h>
#include "SizeOf.h"
#include "Assert.h"
#include <MfgTokens.h>
#include "DebugPrintConfig.h"
#define DEBUGPRINT
#include "DebugPrint.h"
#include "config_app.h"
#include <ZAF_file_ids.h>
#include "ZAF_nvm_app.h"
#include <ZW_slave_api.h>
#include <ZW_classcmd.h>
#include <ZW_TransportLayer.h>
```

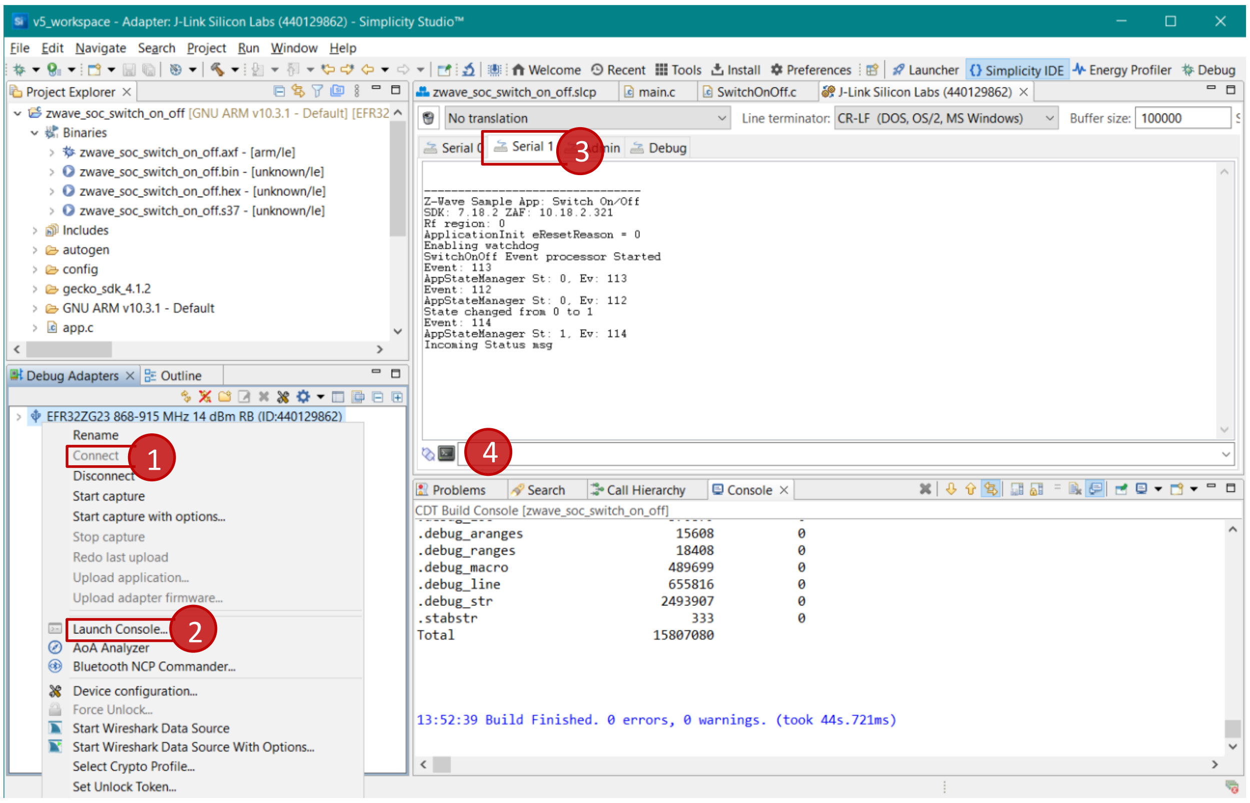
Figure 31: Enable Serial Connection for Debugging

Finally, build the project again and flash the device.

Once the device has been flashed, right click on the adapter, click '*Connect'*, and then '*Launch Console…*'.

In the Console view, select the '*Serial 1'* tab, click in the input field and press *Enter*. The small connection icon to the left of the input field should now show as connected. The console is now ready to display debug input. Refer to Figure 32.

Try to press the '*RESET'* button on the board to see the welcome message. For the Switch On/Off sample application, a new message will be displayed every time the LED is toggled.


v5_workspace - Adapter: J-Link Silicon Labs (440129862) - Simplicity Studio™
File Edit Navigate Search Project Run Window Help
Welcome Recent Tools Install Preferences Launcher Simplicity IDE Energy Profiler Debug
Project Explorer
zwave_soc_switch_on_off [GNU ARM v10.3.1 - Default] [EFR32
Binaries
zwave_soc_switch_on_off.axf - [arm/le]
zwave_soc_switch_on_off.bin - [unknown/le]
zwave_soc_switch_on_off.hex - [unknown/le]
zwave_soc_switch_on_off.s37 - [unknown/le]
Includes
autogen
config
gecko_sdk_4.1.2
GNU ARM v10.3.1 - Default
app.c
Debug Adapters
Outline
EFR32ZG23 868-915 MHz 14 dBm RB (ID:440129862)
Rename
Connect
1
Disconnect
Start capture
Start capture with options...
Stop capture
Redo last upload
Upload application...
Upload adapter firmware...
Launch Console...
2
AoA Analyzer
Bluetooth NCP Commander...
Device configuration...
Force Unlock...
Start Wireshark Data Source
Start Wireshark Data Source With Options...
Select Crypto Profile...
Set Unlock Token...
zwave_soc_switch_on_off.slcp
main.c
SwitchOnOff.c
J-Link Silicon Labs (440129862)
No translation
Line terminator: CR-LF (DOS, OS/2, MS Windows)
Buffer size: 100000
Serial 0
Serial 1
3
Admin
Debug
4
Problems
Search
Call Hierarchy
Console
CDT Build Console [zwave_soc_switch_on_off]

```
----------------------------------------
Z-Wave Sample App: Switch On/Off
SDK: 7.18.2 ZAF: 10.18.2.321
Rf region: 0
ApplicationInit eResetReason = 0
Enabling watchdog
SwitchOnOff Event processor Started
Event: 113
AppStateManager St: 0, Ev: 113
Event: 112
AppStateManager St: 0, Ev: 112
State changed from 0 to 1
Event: 114
AppStateManager St: 1, Ev: 114
Incoming Status msg
```

```
.debug_aranges                15608          0
.debug_ranges                 18408          0
.debug_macro                 489699          0
.debug_line                  655816          0
.debug_str                  2493907          0
.stabstr                        333          0
Total                      15807080


13:52:39 Build Finished. 0 errors, 0 warnings. (took 44s.721ms)
```

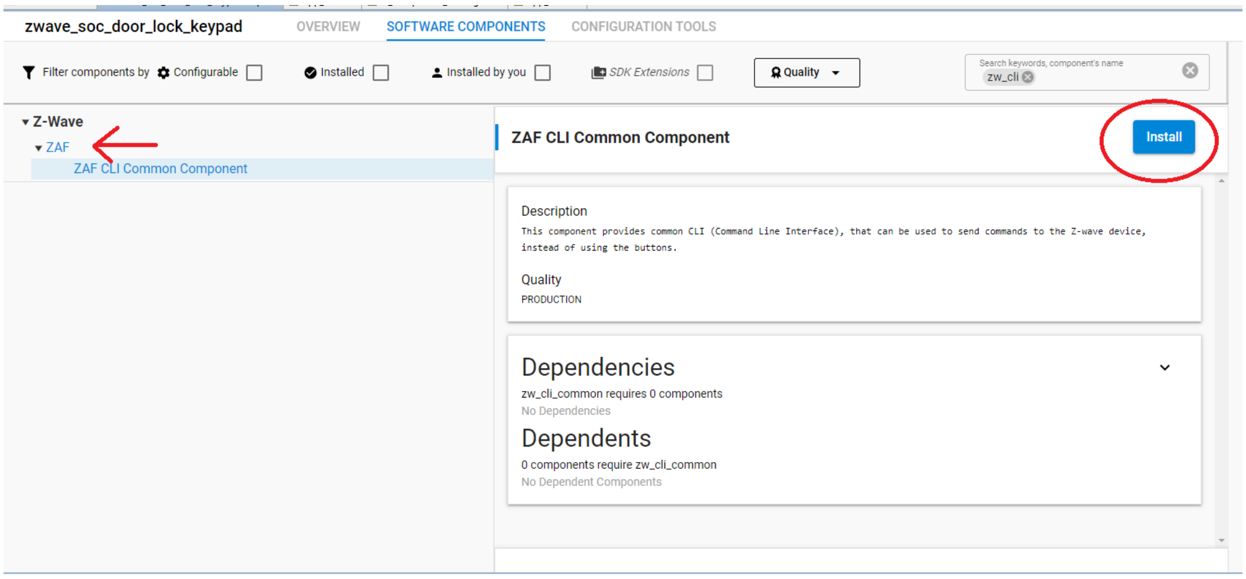
Figure 32: Debug Using Serial Connection

8.3 Command Line Interface

The Z-Wave SDK has a Command Line Interface (CLI) component that allows you to control a device without the use of physical buttons.

The Z-Wave CLI component includes common commands such as factory reset and starting or stopping learn mode. Most of the sample applications have additional commands implemented, based on their specific behavior.

This component is installed by default, unless the *Z-Wave Debug* component is also installed.



Description
This component provides common CLI (Command Line Interface), that can be used to send commands to the Z-wave device, instead of using the buttons.

Quality
PRODUCTION

Dependencies
zw_cli_common requires 0 components
No Dependencies

Dependents
0 components require zw_cli_common
No Dependent Components

Figure 33: Installing ZAF CLI Common Component

To connect to the CLI, the same procedure can be used as the one for the Serial Debug, described on the previous page. Once the connection is established, commands can be issued from the input field of the serial terminal.

Note: The CLI is halted while the application is in sleep mode. To disable sleeping, wake up the device (e.g. by pressing the Learn Mode button) and issue the *'sleeping disable'* command.

9 Next Steps in Developing

Congratulations! You have now completed the Z-Wave Getting Started Guide for developing end devices. You should now have a broad understanding of the contents of the development kit, as well as the development environment.

It is recommended to investigate the following documents, as a natural next step in your learning curve.

9.1 Next Steps for Software Developers

INS14278 - How to Use Certified Apps in Z-Wave 700/800 [1]

The purpose of this document is to describe how to use the sample applications which come as part of the Z-Wave SDK.

The document also describes the most commonly used Z-Wave terms, such as Association Groups, Endpoint, and Security. It also covers the basics of the Z-Wave SDK Framework and the libraries.

INS14914 - Energy Profiler for Z-Wave Applications [12]

This document describes how to use the Energy Profiler perspective in Simplicity Studio.

The Energy Profiler enables you to visualize the energy consumption of individual devices, multiple devices on one target system, or a network of interacting wireless devices to analyze and improve the power performance of these systems.

INS14259 - Z-Wave Plus V2 Application Framework SDK7 [4]

While previously mentioned guides give you a good start and introduce you to all the basics of the world of Z-Wave, this document contains all the details of the embedded SDK. Familiarize yourself with this document before developing your own application.

Simplicity Studio Training

For more training and in-depth information about 'Simplicity Studio', navigate to the homepage of the IDE: https://www.silabs.com/products/development-tools/software/simplicity-studio

9.2 Next Steps for Hardware Developers

UG517: Z-Wave 800 Series Integration Guide [14]

This document provides an implementation guide for integrating Z-Wave 800 devices into product designs. It is intended for product design engineers who aim for a fast integration of Z-Wave 800 devices.

INS14487 - Z-Wave 700 Series Integration Guide [5]

This document provides an implementation guide for integrating Z-Wave 700 devices into product designs. It is intended for product design engineers who aim for a fast integration of Z-Wave 700 devices.

INS14283 - Bring-up/test HW development [6]

This document describes how to use Silicon Labs development tools to bring up and validate hardware based on the Z-Wave 700/800 devices, including instructions in how to test the RF Performance of a device, without the overhead of the Z-Wave protocol.

INS14285 - Manufacture Z-Wave 700/800 product in volume [7]

Security-2 configuration in production environment, e.g., handling of QR codes.

NB: This document is not part of the Z-Wave 700 Beta release

INS14498 - Mandatory crystal adjustment for EFR32ZG14 based products [8]

This document describes the mandatory adjustment of the system crystal which must be performed on a product based on the EFR32ZG14 Gateway device

9.3 Certification

Each product must follow the Z-Wave Plus v2 specification to be able to pass the certification program and ensure interoperability in the ecosystem of existing products.

As the product must meet specific certification requirements it is strongly recommended that you consider the technical requirements for your device ***early in the process***, refer to [9] and [10].

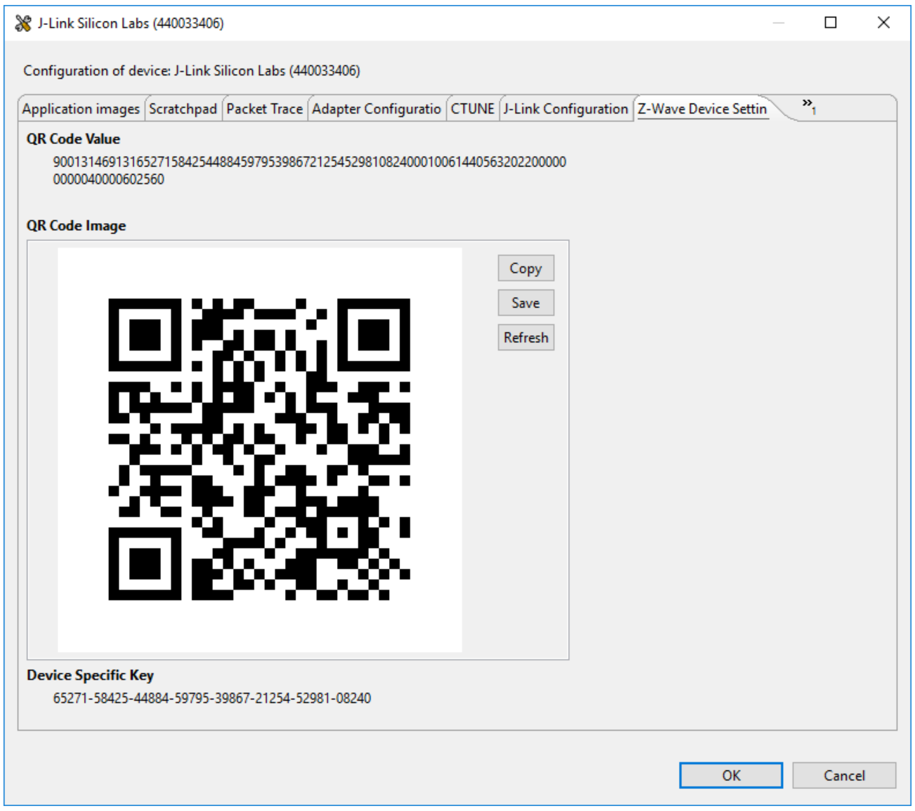
INS14284 - Prepare for Z-Wave Certification Tests [11]

Read this ‘Certification Overview’ document to understand the process and receive guidance in how to start developing your product to pass the certification.

10 Appendix A: Reading Out QR Code and DSK

Z-Wave chips generate their own S2 Device Specific Key when they have been programmed for the first time. S2 controllers require this DSK to be verified by the user, and in some cases the first part must even be typed in when the node is included. This section describes how to read out the DSK from the chip.

Using Simplicity Studio, right click on your connected hardware in the '*Debug Adapters*' view. See Figure 7**Error! Reference source not found.**, label 3. Select '*Device Configuration*' from the context menu.
From this menu, navigate to the 'Z-Wave Device Settings' tab (click the '*Show list'* » if it is not visible).



Configuration of device: J-Link Silicon Labs (440033406)

QR Code Value

```
900131469131652715842544884597953986721254529810824000100614405632022000000
000040000602560
```

QR Code Image

Device Specific Key

```
65271-58425-44884-59795-39867-21254-52981-08240
```

Figure 34: Device Configuration Dialog Showing Z-Wave Device Settings

In this view the entire QR Code Value and the corresponding QR Code Image is shown. In addition, the Device Specific Key (DSK) is shown.

This DSK can be compared against the Z-Ware UI, PC Controller dialog box or other Controller UI. If needed, the first decimal group can be typed in for S2 secure inclusion.

11 Appendix B: Configuring the Development Kit as a Network Sniffer

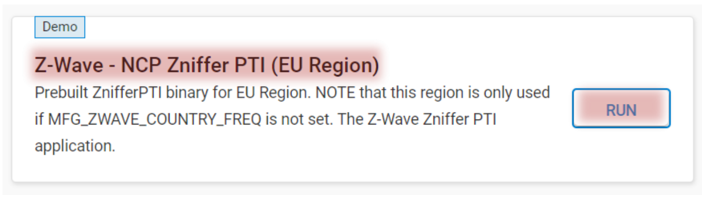
The Z-Wave 800 Pro Kit does not contain a UZB-S USB Stick Z-Wave Network Sniffer. However, a second development kit can be configured as a network sniffer with identical functionality. This section describes how to program a development kit to enable it to be used as a Z-Wave network sniffer.

In Simplicity Studio, navigate to the Launcher Perspective (refer to Figure 7).

Select the device that you would like to use as the network sniffer, then go to the '*Example Projects & Demos*' tab.
Here, choose a suitable demo variant for your hardware and region, then click '*Run*', as seen in Figure 35.

Both variants of the Zniffer application can capture and forward Z-Wave network traffic to the host computer for analysis. However, they use different interfaces for connection:

- The **Z-Wave - NCP Zniffer** uses a **USB** connection.
- The **Z-Wave - NCP Zniffer PTI** requires an **Ethernet** connection between the WSTK and the host computer. As such, it is only supported on kits with a mainboard and a radio board.
  For instructions on how to establish this connection, refer to [3].



Z-Wave - NCP Zniffer PTI (EU Region)

Prebuilt ZnifferPTI binary for EU Region. NOTE that this region is only used if MFG_ZWAVE_COUNTRY_FREQ is not set. The Z-Wave Zniffer PTI application.

Figure 35: Flashing the Z-Wave - NCP Zniffer PTI Application

After the program has been flashed, the development kit is ready for use as a Z-Wave network sniffer.

For instructions on how to use the Z-Wave Zniffer application, refer to Section 7.2 of this document.

12 Appendix C: Decreasing the firmware's size

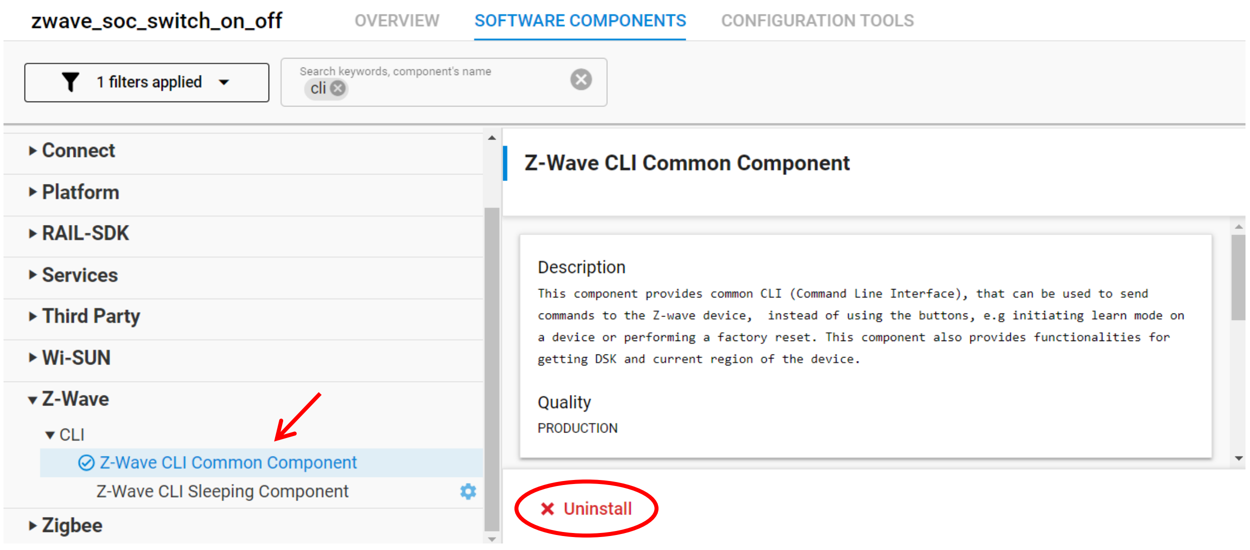
In case you run out of application flash memory, there are two recommended options to gain space by disabling certain application features:

1. **Disabling the Command Line Interface component**
   (recommended when the CLI functionality is not needed).
2. **Removing the OTA bootloader slot**
   (recommended when debugging the application, as a temporary measure).

12.1 Disabling the Command Line Interface component

Uninstall the *'Z-Wave CLI Common Component'* from your project (see Figure 36).

This will save about 25 kilobytes of space.



Description

This component provides common CLI (Command Line Interface), that can be used to send commands to the Z-wave device, instead of using the buttons, e.g initiating learn mode on a device or performing a factory reset. This component also provides functionalities for getting DSK and current region of the device.

Quality

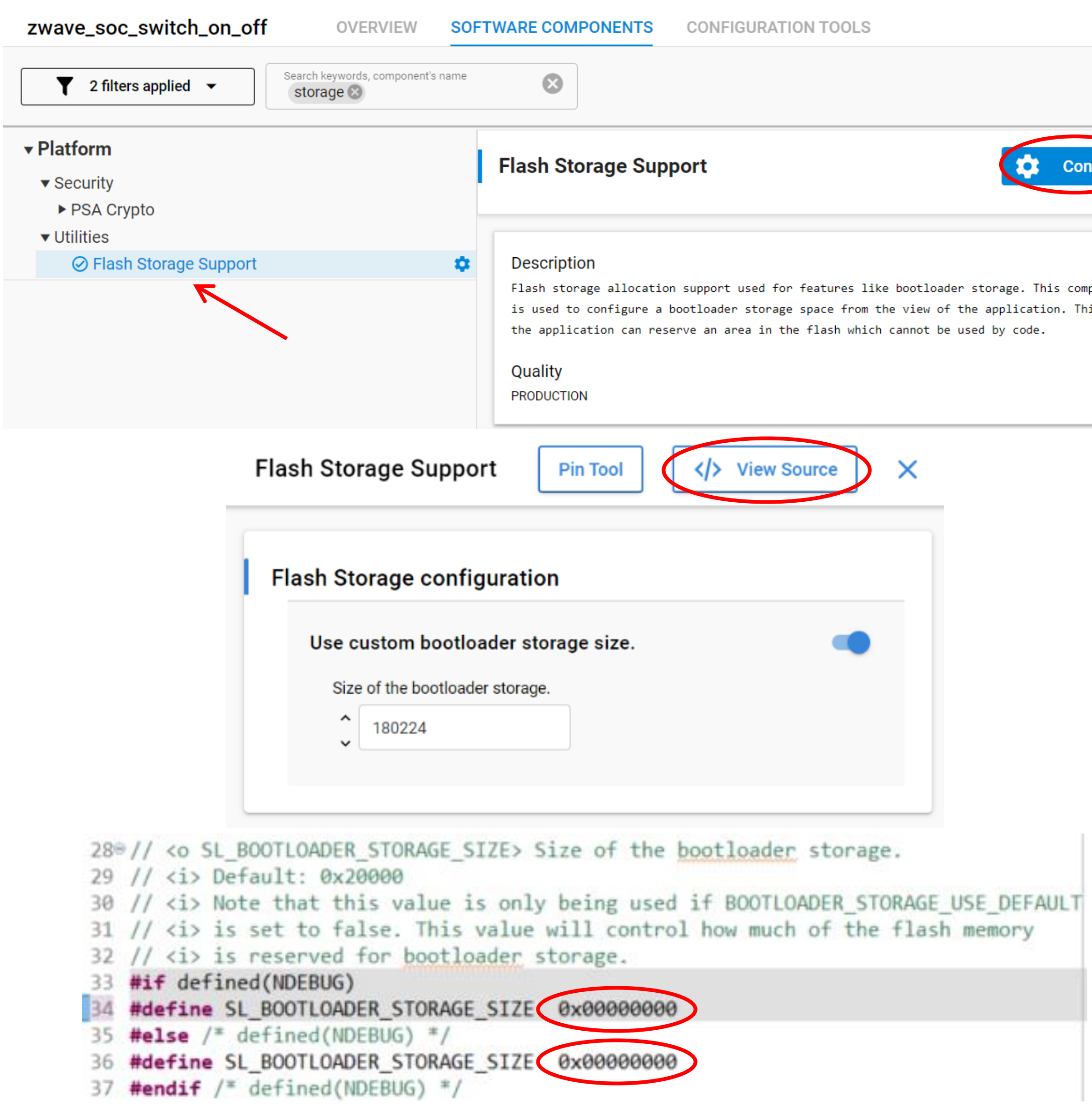
PRODUCTION

Figure 36: Uninstalling the Z-Wave CLI Common Component

12.2 Removing the OTA bootloader slot

Find the *'Flash Storage Support'* component and click *'Configure'*, then click on *'View Source'* (or open *'<Project Folder>/config/sl_storage_config.h'*).

Change the value of *'SL_BOOTLOADER_STORAGE_SIZE'* to 0 (the graphical configuration interface should not be used for this).
The first instance of this define is for release builds (this is the default) and the second one applies when the *'Z-Wave Debug'* component is installed.



Description
Flash storage allocation support used for features like bootloader storage. This component is used to configure a bootloader storage space from the view of the application. This way the application can reserve an area in the flash which cannot be used by code.

Quality
PRODUCTION

Flash Storage configuration

Use custom bootloader storage size.

Size of the bootloader storage.
180224

```
// <o SL_BOOTLOADER_STORAGE_SIZE> Size of the bootloader storage.
// <i> Default: 0x20000
// <i> Note that this value is only being used if BOOTLOADER_STORAGE_USE_DEFAULT
// <i> is set to false. This value will control how much of the flash memory
// <i> is reserved for bootloader storage.
#if defined(NDEBUG)
#define SL_BOOTLOADER_STORAGE_SIZE  0x00000000
#else /* defined(NDEBUG) */
#define SL_BOOTLOADER_STORAGE_SIZE  0x00000000
#endif /* defined(NDEBUG) */
```

Figure 37: Setting the custom bootloader storage size to 0

This way, about 200 KB of space can be gained.

Note: after this modification, updating the application via OTA will not be possible.

References

[1] Silabs, INS14278, Instruction, How to Use Z-Wave Pre-Certified Apps.
[2] Silabs, INS13114, Instruction, Z-Wave PC Based Controller v5 User Guide.
[3] Silabs, INS10249, Instruction, Z-Wave Zniffer User Guide.
[4] Silabs, INS14259, Instruction, Z-Wave Plus V2 Application Framework GSDK.
[5] Silabs, INS14487, Instruction, 700 Integration Guide.
[6] Silabs, INS14283, Instruction, Bring-up/test HW development.
[7] Silabs, INS14285, Instruction, Manufacture Z-Wave product in volume.
[8] Silabs, INS14498, Instruction, Mandatory crystal adjustment for EFR32ZG14 based products.
[9] Silabs, SDS14223, Software Design Specification, Z-Wave Command Class Control Specification.
[10] Silabs, SDS14224, Software Design Specification, Z-Wave Plus v2 Device Types Specification.
[11] Silabs, INS14284, Instruction, Prepare for Z-Wave Certification Tests.
[12] Silabs, INS14914, Instruction, Energy Profiler for Z-Wave Applications.
[13] Silabs, UG103.06, User’s Guide, Bootloader Fundamentals
[14] Silabs, UG517, User’s Guide, Z-Wave 800 Series Integration Guide