



Instruction

Z-Wave Getting Started for End Devices

Document No.:	INS14280
Version:	14
Description:	-
Written By:	ANERDO;LAURGE;BSANTOK;PEBALOGH
Date:	2024-05-30
Reviewed By:	BSANTOK;ANERDO
Restrictions:	Public

Approved by:

Date	CET	Initials	Name	Justification
2024-05-30	15:35:46	ANERDO		András Erdo

This document is the property of Silicon Labs. The data contained herein, in whole or in part, may not be duplicated, used or disclosed outside the recipient for any purpose. This restriction does not limit the recipient's right to use information contained in the data if it is obtained from another source without restriction.



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 Ziffer 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 8 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 Ziffer variants

Table of Contents

1	ABBREVIATIONS.....	6
2	INTRODUCTION	6
2.1	Purpose.....	6
2.2	Audience and Prerequisites.....	6
3	WELCOME TO Z-WAVE	7
4	THE Z-WAVE DEVELOPMENT KITS	8
4.1	Development Kit Hardware by 800 Series.....	8
4.1.1	Z-Wave 800 Series Pro Kit	8
4.1.1.1	Prepare the Hardware	8
4.1.2	Z-Wave Thunderboard Kits.....	9
4.1.2.1	Z-Wave 800 Series Development Kit	9
4.1.2.2	EFR32xG28 Explorer Kit	9
4.2	Development Kit Hardware by 700 Series.....	10
4.2.1	Prepare the Hardware.....	11
5	INSTALL THE Z-WAVE SDK	12
5.1	Connect your Hardware	12
5.2	Install Simplicity Studio	12
5.2.1	Updating Adapter Firmware.....	14
5.2.2	Associate Simplicity Studio with the Hardware.....	14
5.2.3	Functionality in the Launcher Perspective	14
6	PREPARE YOUR FIRST Z-WAVE SAMPLE APPLICATION	16
6.1	Configuring a Sample Application	17
6.2	Flashing the Bootloader	20
6.3	Flashing the Application	21
6.4	Changing the Region.....	22
6.5	Reprogramming a Sleeping Device.....	23
7	RUN YOUR FIRST Z-WAVE SAMPLE APPLICATION	24
7.1	Z-Wave PC Controller	24
7.2	Z-Wave Zniffer	28
8	DEBUG A Z-WAVE APPLICATION.....	30
8.1	Debugger	30
8.2	Serial Debug.....	32
8.3	Command Line Interface	34
9	NEXT STEPS IN DEVELOPING.....	35
9.1	Next Steps for Software Developers	35

9.2	Next Steps for Hardware Developers	36
9.3	Certification	36
10	APPENDIX A: READING OUT QR CODE AND DSK	37
11	APPENDIX B: CONFIGURING THE DEVELOPMENT KIT AS A NETWORK SNIFFER	38
	REFERENCES.....	39

Table of Figures

Figure 1: BRD2603A - ZGM230s +14 dBm Dev Kit Board	9
Figure 2: EFR32xG28 Explorer Kit.....	9
Figure 3: Content of the Z-Wave Development Kit	10
Figure 4: Install Required Device Inspector	12
Figure 5: Select Devices to Install the Needed Software Packages	13
Figure 6: Device Firmware Update.....	14
Figure 7: The Launcher Perspective	15
Figure 8: Simplicity Studio Snapshot Listing Z-Wave Sample Application Projects	16
Figure 9: Simplicity Studio Snapshot Listing Pre-Built Z-Wave Demos.....	16
Figure 10: The Simplicity IDE Perspective.....	17
Figure 11: Application Configuration Window.....	17
Figure 12: Application Configuration Parameter for Selecting Region-Frequency	18
Figure 13: Sample Application - Build Step	19
Figure 14: Sample Application Build Completed and Binaries Are Generated.....	19
Figure 15: Flashing the Bootloader	20
Figure 16: Select Image to Flash.....	21
Figure 17: Flash Programmer With Selected Device and File to Be Flashed.....	21
Figure 18: Recover a Sleeping Device With 'Unlock Debug Access'.....	23
Figure 19: The Location of the Show Compatible Tools Dialog in the Toolbar	24
Figure 20: Setting the RF Region in the Z-Wave PC Controller	24
Figure 21: PC Controller With 1 End Device Added	25
Figure 22: PC Controller, Command Classes View—Send Switch On Command	26
Figure 23: PC Controller, Command Classes View—Send Switch Get Command	27
Figure 24: Zniiffer Trace Showing a Decrypted S2 Frame for a 'Switch Binary Get'	28
Figure 25: Installing the Z-Wave Debug component	30
Figure 26: Debugging Switch On/Off	31
Figure 27: Using Step Into to Debug the Functionalities of ToggleLed	31
Figure 28: Install the Z-Wave Debug Print Component	32
Figure 29: Enable Serial Connection for Debugging	32
Figure 30: Debug Using Serial Connection.....	33
Figure 31: Installing ZAF CLI Common Component.....	34
Figure 32: Device Configuration Dialog Showing Z-Wave Device Settings	37
Figure 33: Flashing the Z-Wave - NCP Zniiffer PTI Application	38

Table of Tables

Table 1: Overview of Selectable Frequencies18

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 latest news of the Z-Wave 800, refer to:

<https://www.silabs.com/wireless/z-wave/800-series-modules>

and

<https://www.silabs.com/wireless/z-wave/800-series-modem-soc>

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.
- 2 pcs. BRD8029A – Buttons and LEDs Expansion Board (EXP board).
- 1 pc. SLUSB7000A – UZB7 – Controller USB stick.
- 2 pcs. ANT-SS900 868-915MHz Compressed Whip Antenna.
- 1 pc. 10-pin 1.27 mm IDC debug cable.

Note: the UZB-S USB stick network sniffer is not part of the Z-Wave 800 Pro Kit. A second Kit can be configured for use as a Z-Wave network sniffer. For instructions, refer to Section 11 of this document.

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) 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 various 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 are either not available or supported only with 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

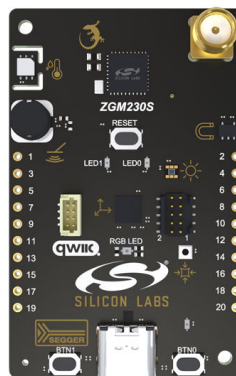


Figure 1: BRD2603A - ZGM230s +14 dBm Dev Kit Board

4.1.2.2 EFR32xG28 Explorer Kit

The **EFR32xG28 Explorer Kit** contains a single xG28 Explorer board (BRD2705A).

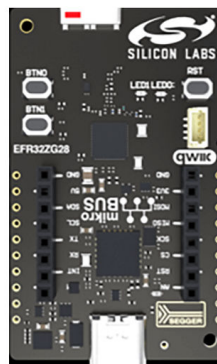


Figure 2: EFR32xG28 Explorer Kit

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.
- 2 pcs. ENRM002 – 1m USB A<-> USB Mini B cable.

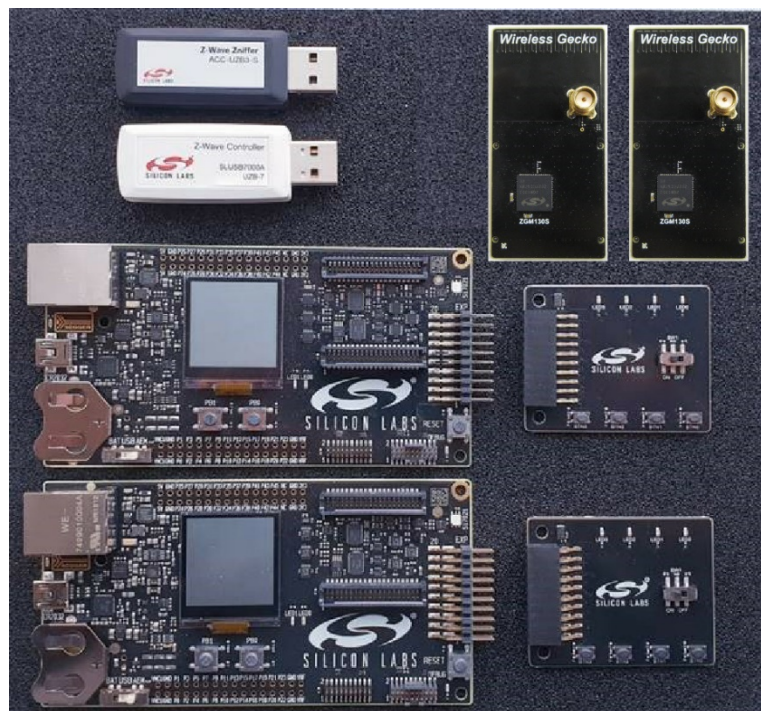


Figure 3: Content of the Z-Wave Development Kit

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.

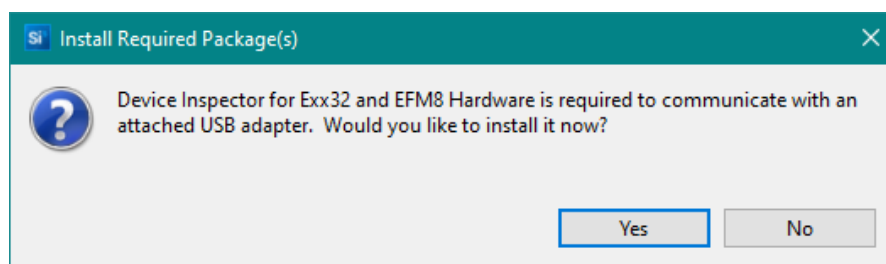


Figure 4: Install Required Device Inspector

- 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'*.

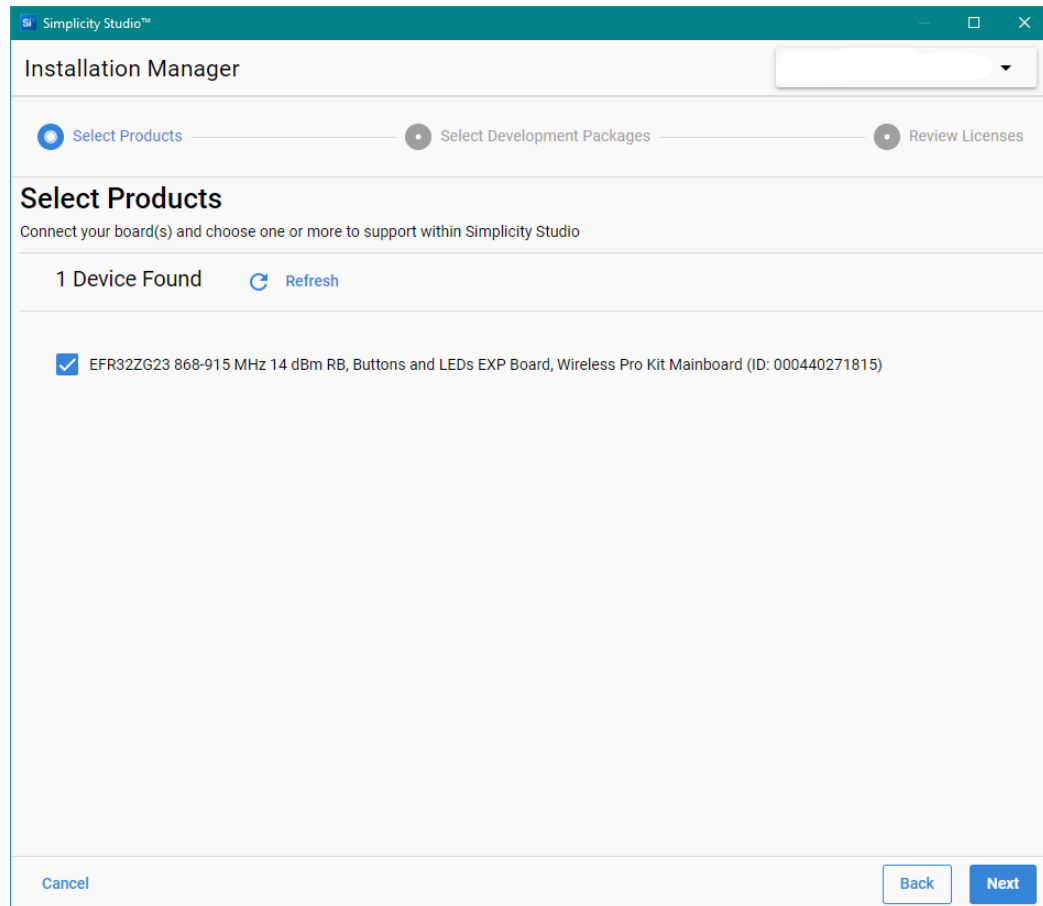


Figure 5: Select Devices to Install the Needed Software Packages

- 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.

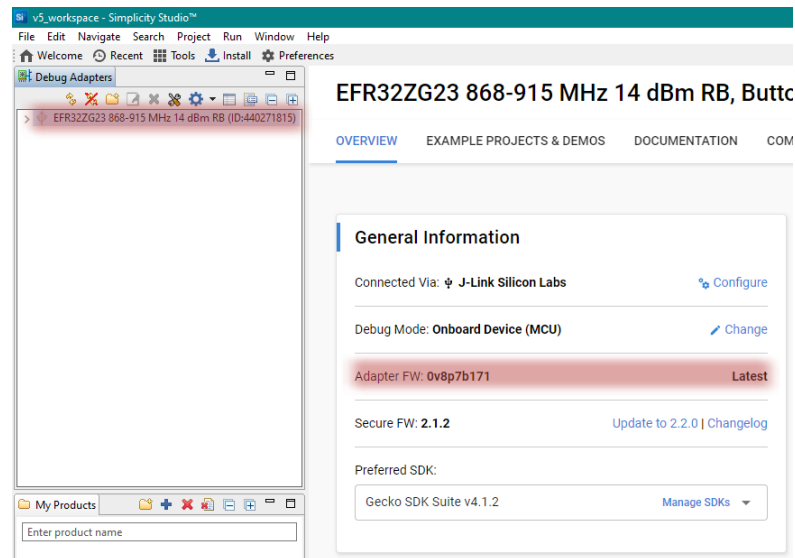


Figure 6: Device Firmware Update

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 then is 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/5.4.0/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. 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.

The screenshot shows the Simplicity Launcher interface with several views and components highlighted by numbered red circles:

- 1**: Points to the **Toolbar** at the top of the window.
- 2**: Points to the **Perspective Buttons** in the top right corner.
- 3**: Points to the **Debug Adapters** view on the left side.
- 4**: Points to the **My Products** view at the bottom left.
- 5**: Points to the **Adapter FW** section in the **General Information** panel.
- 6**: Points to the **Preferred SDK** dropdown menu in the **General Information** panel.
- 7**: Points to the **Documentation** tab in the top navigation bar.

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 Gecko 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 and example applications.
- 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 run the sample applications.

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

Figure 8 and Figure 9 below depict a snapshot list of some sample applications and pre-built demos for the selected hardware.

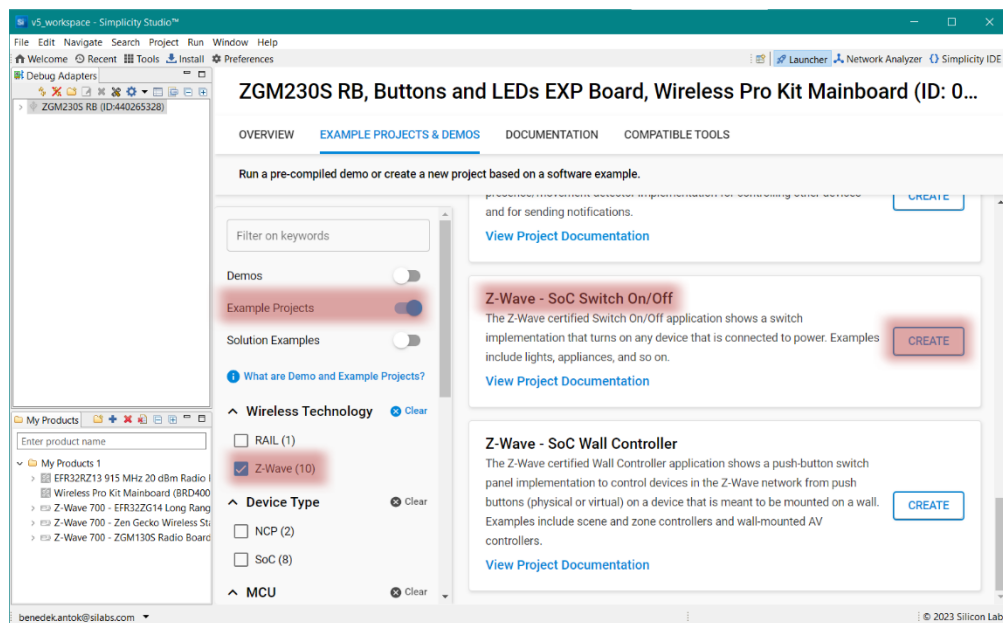


Figure 8: Simplicity Studio Snapshot Listing Z-Wave Sample Application Projects

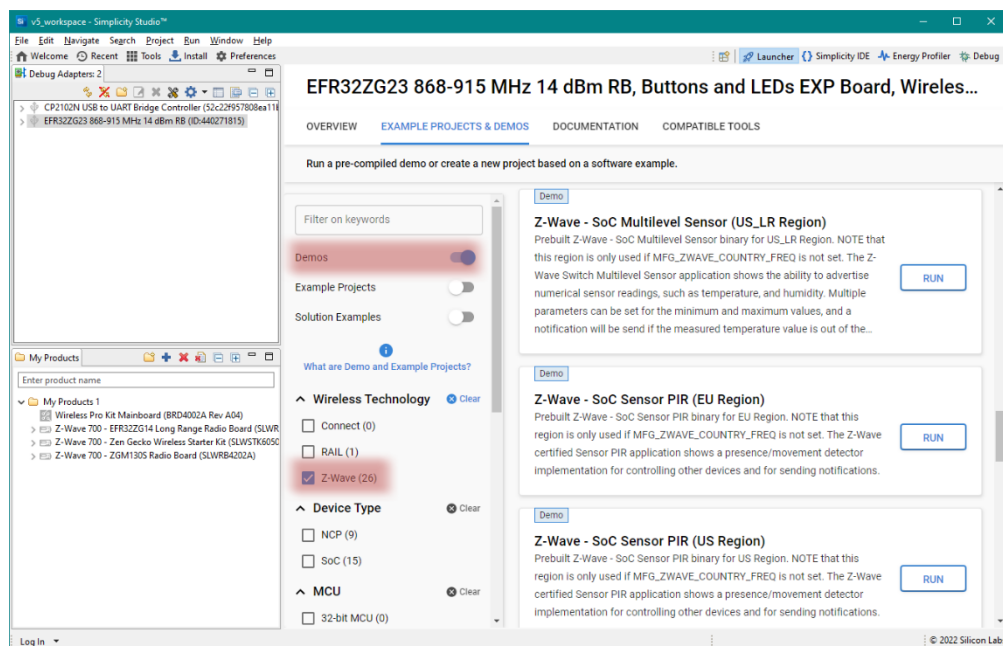


Figure 9: Simplicity Studio Snapshot Listing Pre-Built Z-Wave Demos

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

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 10.

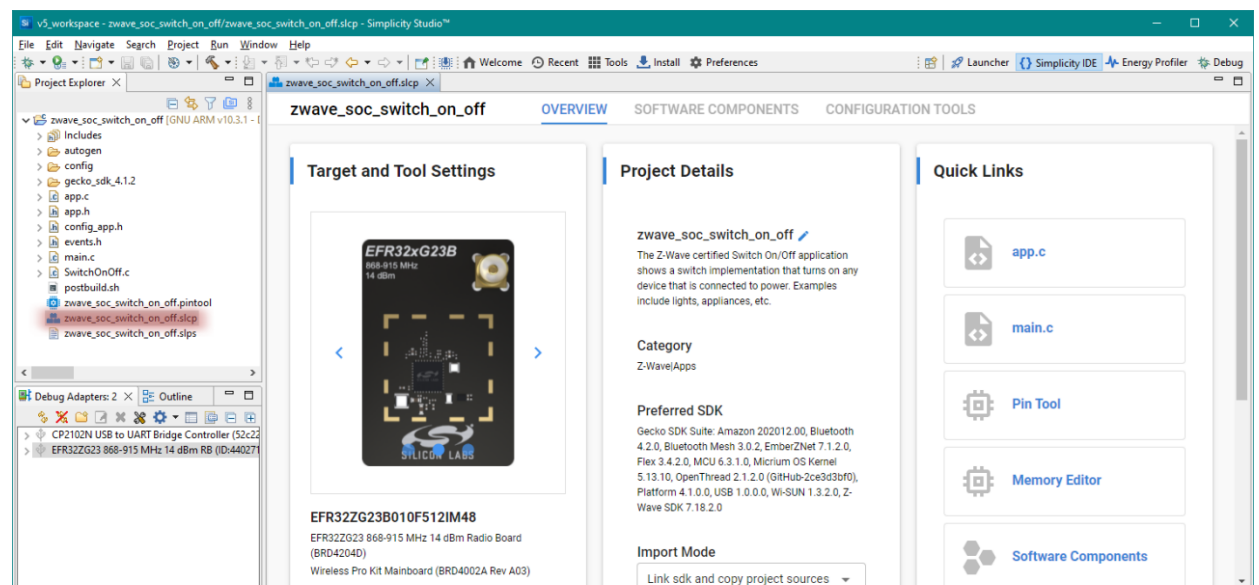


Figure 10: The Simplicity IDE Perspective

The main logic of this sample application is in the file called 'SwitchOnOff.c'. 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 12.

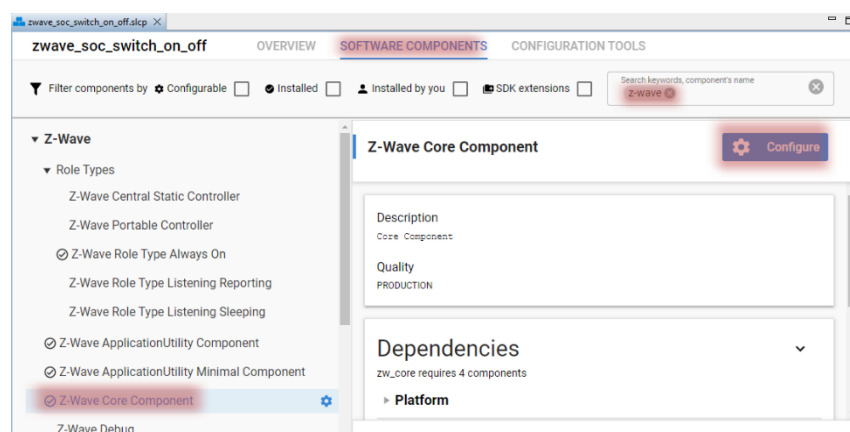


Figure 11: Application Configuration Window

From here, select the desired frequency as shown in Figure 12.

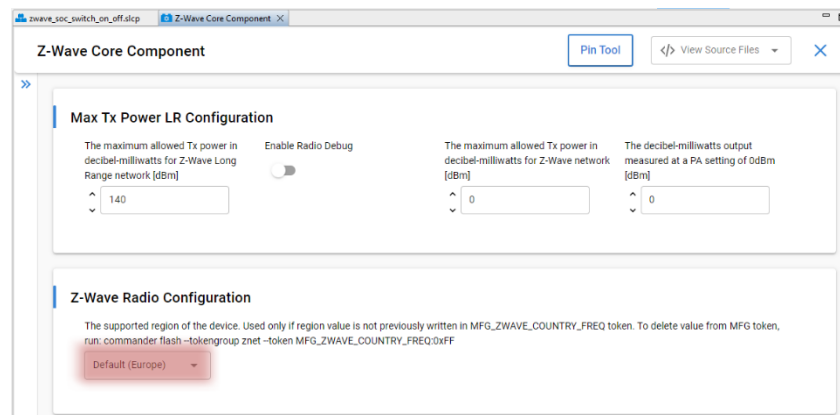


Figure 12: Application Configuration Parameter for Selecting Region-Frequency

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.

Table 1: Overview of Selectable Frequencies

Frequency Region	Variable to use
Europe	REGION_EU
United States of America	REGION_US
Australia/New Zealand	REGION_ANZ
Hong Kong	REGION_HK
India	REGION_IN
Israel	REGION_IL

Frequency Region	Variable to use
Russia	REGION_RU
China	REGION_CN
USA – Long Range	REGION_US_LR
Europe – Long Range	REGION_EU_LR
Japan	REGION_JP
Korea	REGION_KR

Click on Build, as shown in Figure 13.

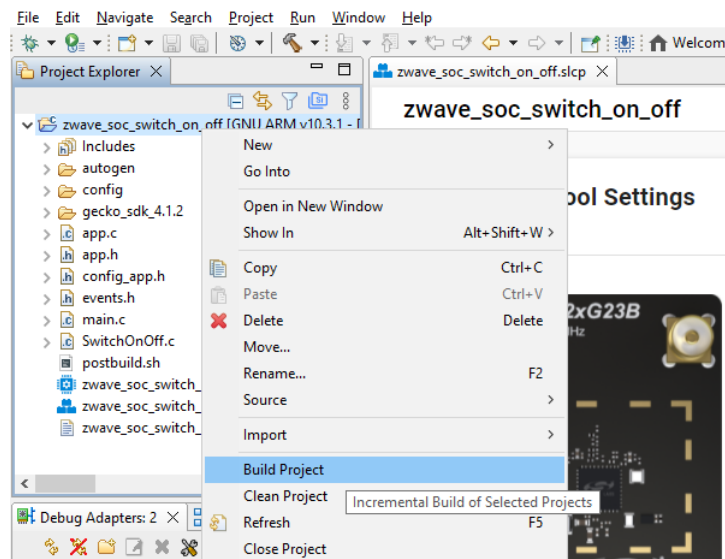


Figure 13: Sample Application - Build Step

Once the build finishes (Figure 14), after a short while, a new folder named '*Binaries*' shows up in the Project Explorer view.

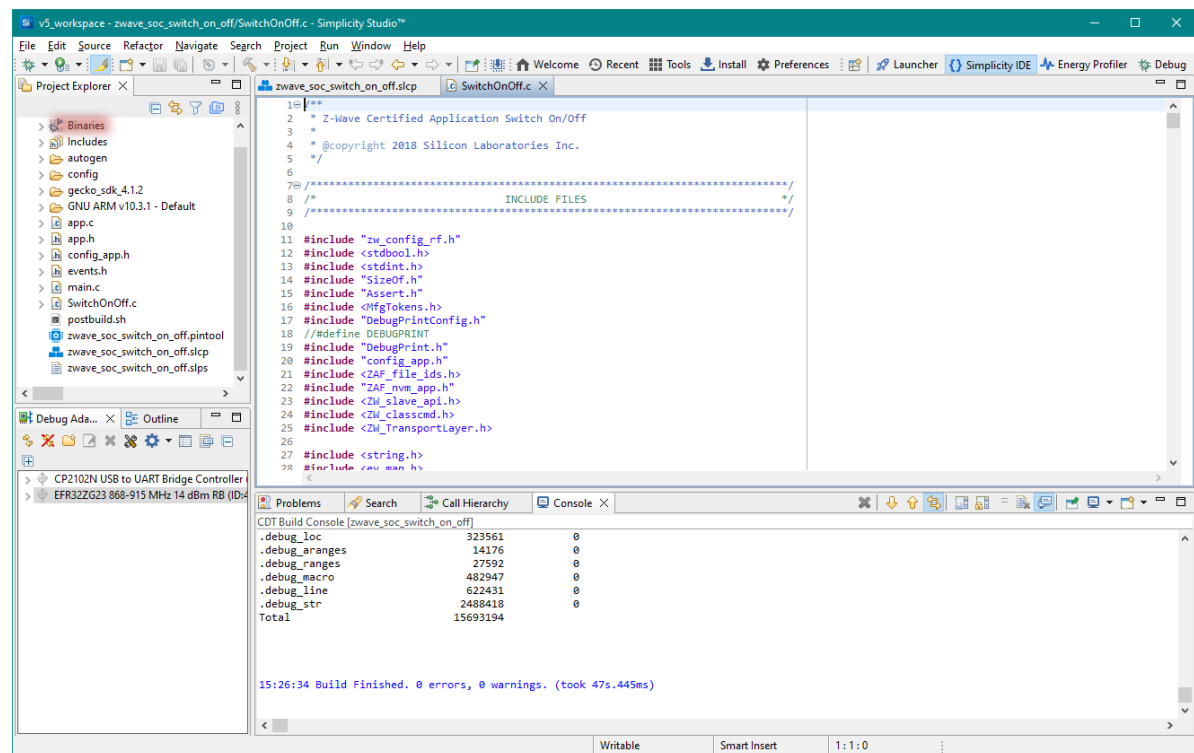


Figure 14: Sample Application Build Completed and Binaries Are Generated

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 that 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 15:

- 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.

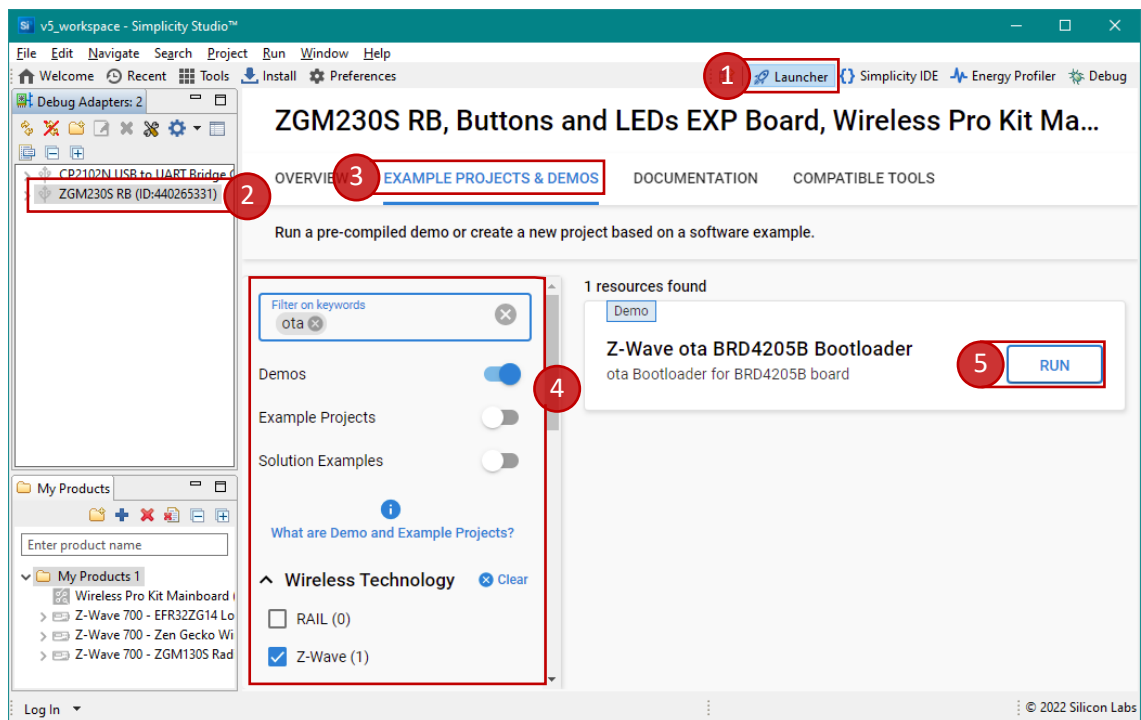


Figure 15: Flashing the Bootloader

To avoid having to flash the bootloader manually, you can create a solution instead of a single project. To see the list of available solutions, use the 'Solution Examples' filter in the Launcher perspective. When you build such a solution, a combined binary containing both the bootloader and the application will be created in the 'artifact' folder of your application project.

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 16).

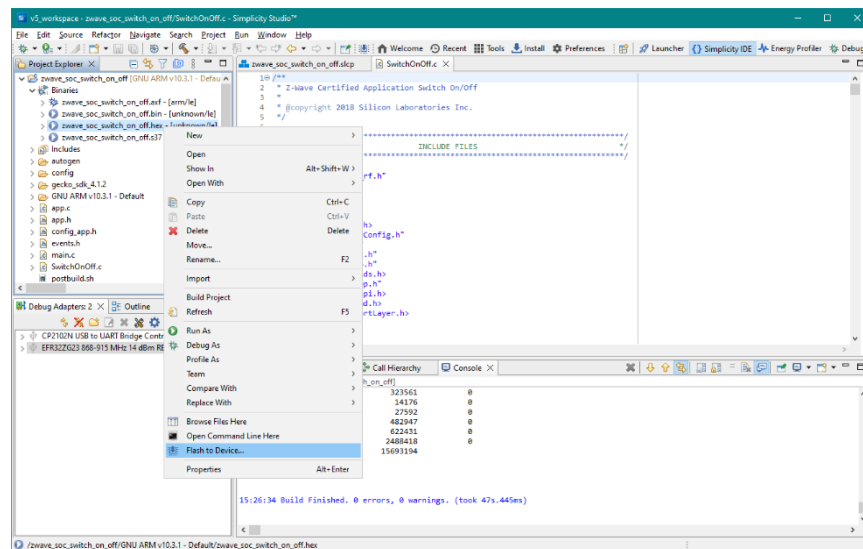


Figure 16: Select Image to Flash

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 17.

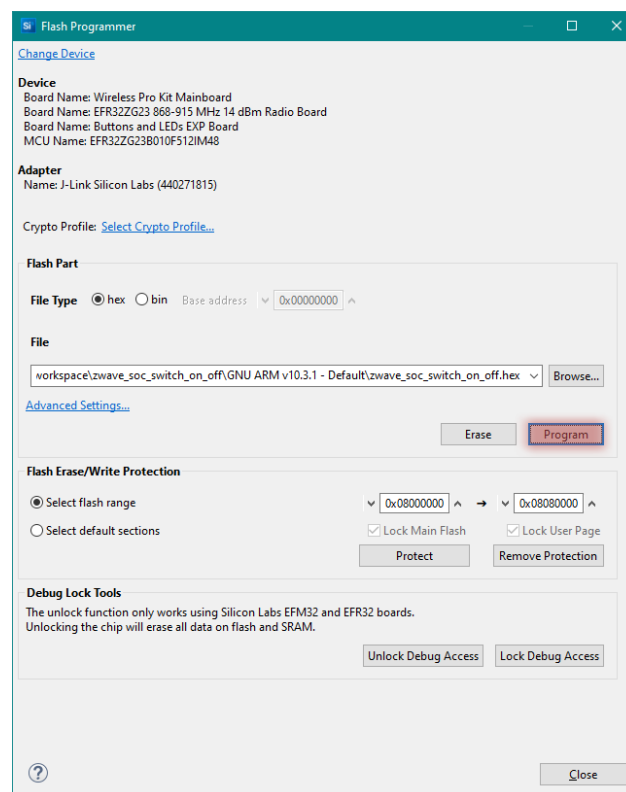


Figure 17: Flash Programmer With Selected Device and File to Be Flashed

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.

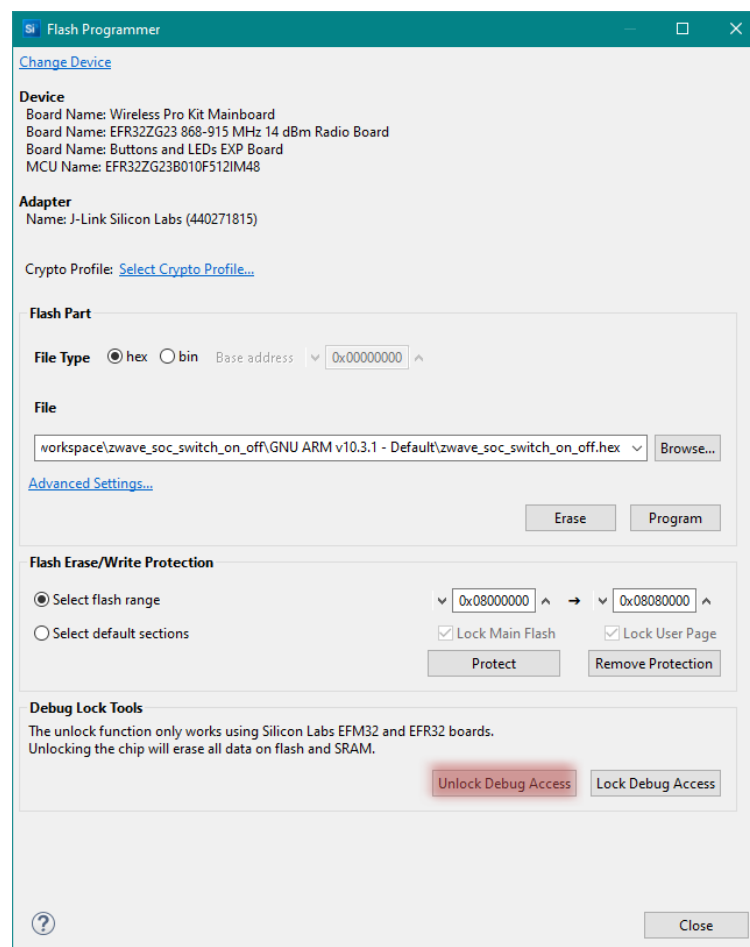


Figure 18: Recover a Sleeping Device With 'Unlock Debug Access'

7 Run Your First Z-Wave Sample Application

This section describes how 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 Gecko SDK suite (which you have installed in Section 5).

7.1 Z-Wave PC Controller

The Z-Wave PC Controller is an application for communicating with Z-Wave and Z-Wave Long Range nodes, like switches and sensors, through a USB Controller (which comes as part of the Developer Kit) connected to a USB port on the PC.


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 UZB Controller to your computer.
- 2) Open the PC Controller from the '*Show Compatible Tools*' dialog in the Toolbar.



Figure 19: 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 uses the US RF region by default. If your end device is programmed for another region, change the PC Controller's RF Region at the Transmit Settings.

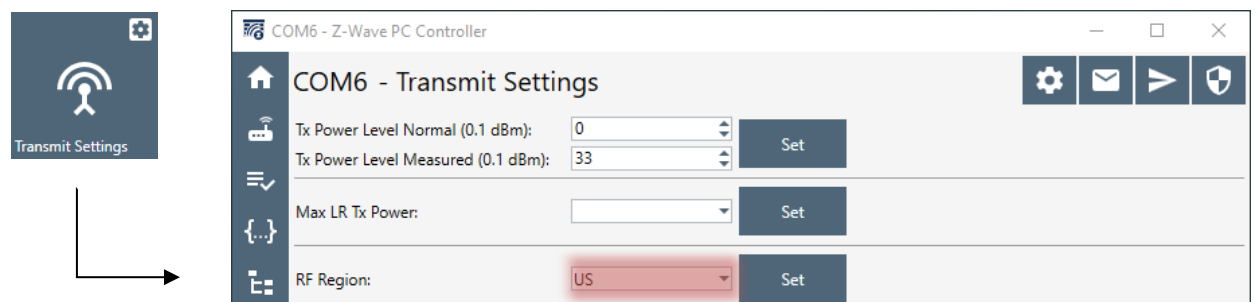


Figure 20: Setting the RF Region in the Z-Wave PC Controller

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

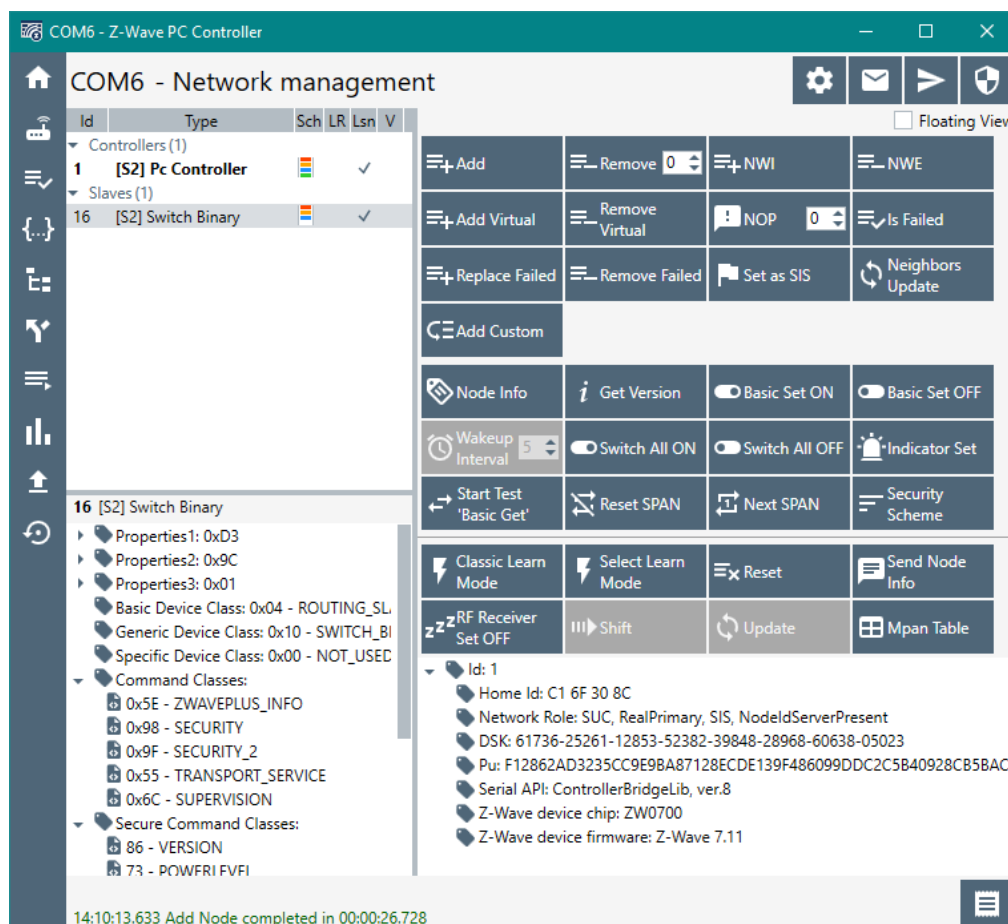


Figure 21: PC Controller With 1 End Device Added

- 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. For the sample applications, press the button labeled 'BTN1' on the Expansion Board. (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 'Slaves' 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 22.

Set the value and click on '*Send*'. The value on the switch should now have been set accordingly, which is indicated by '*LEDO*' on the EXP board.

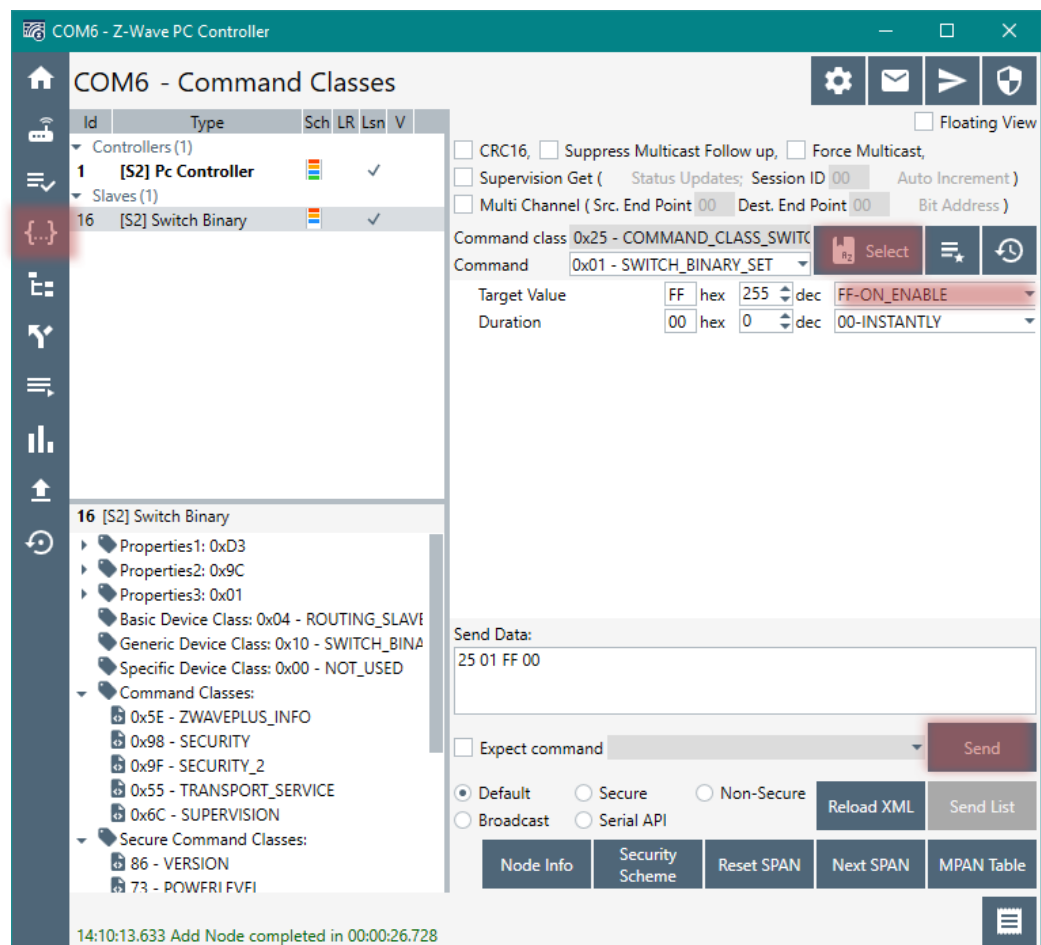


Figure 22: PC Controller, Command Classes View—Send Switch On Command

- 8) Send a Get Command to verify the switch was set accordingly. Refer to Figure 23.

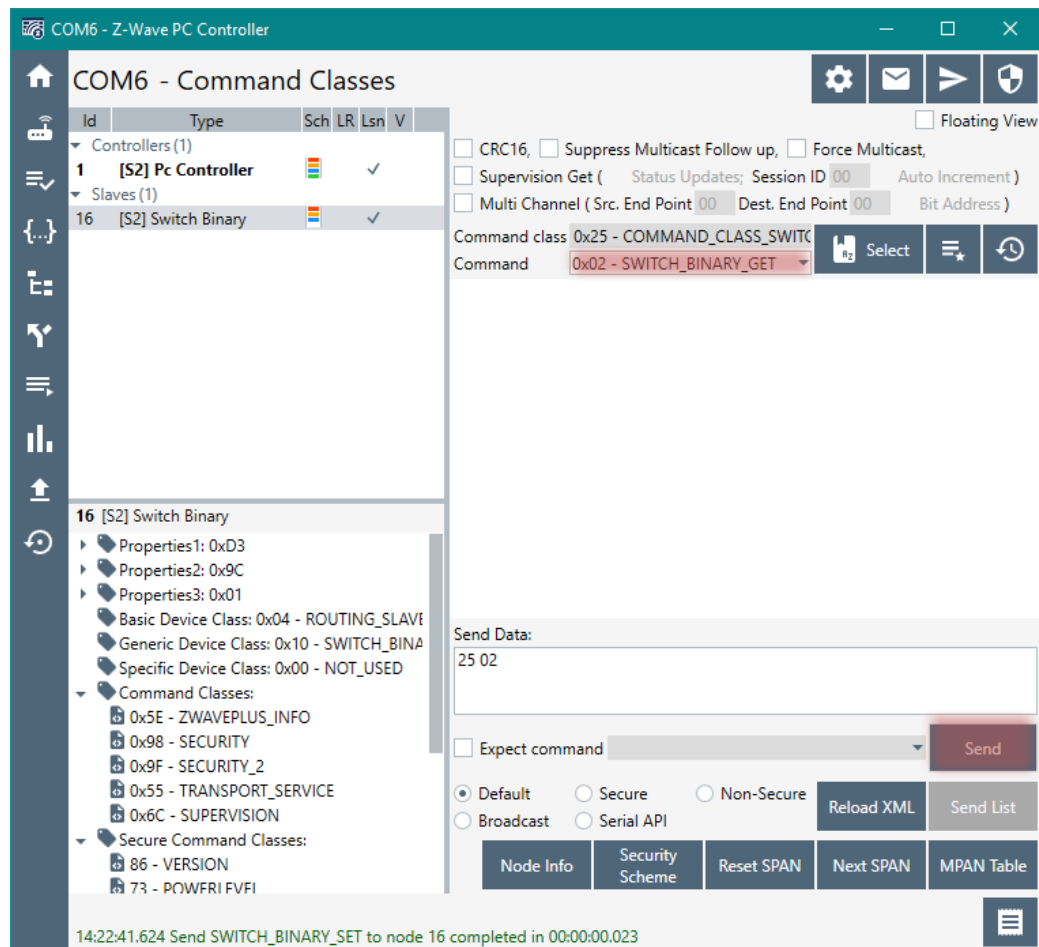


Figure 23: PC Controller, Command Classes View—Send Switch Get Command

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, which 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.
- 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.
- 5) When ready, click on 'Start capturing frames' .

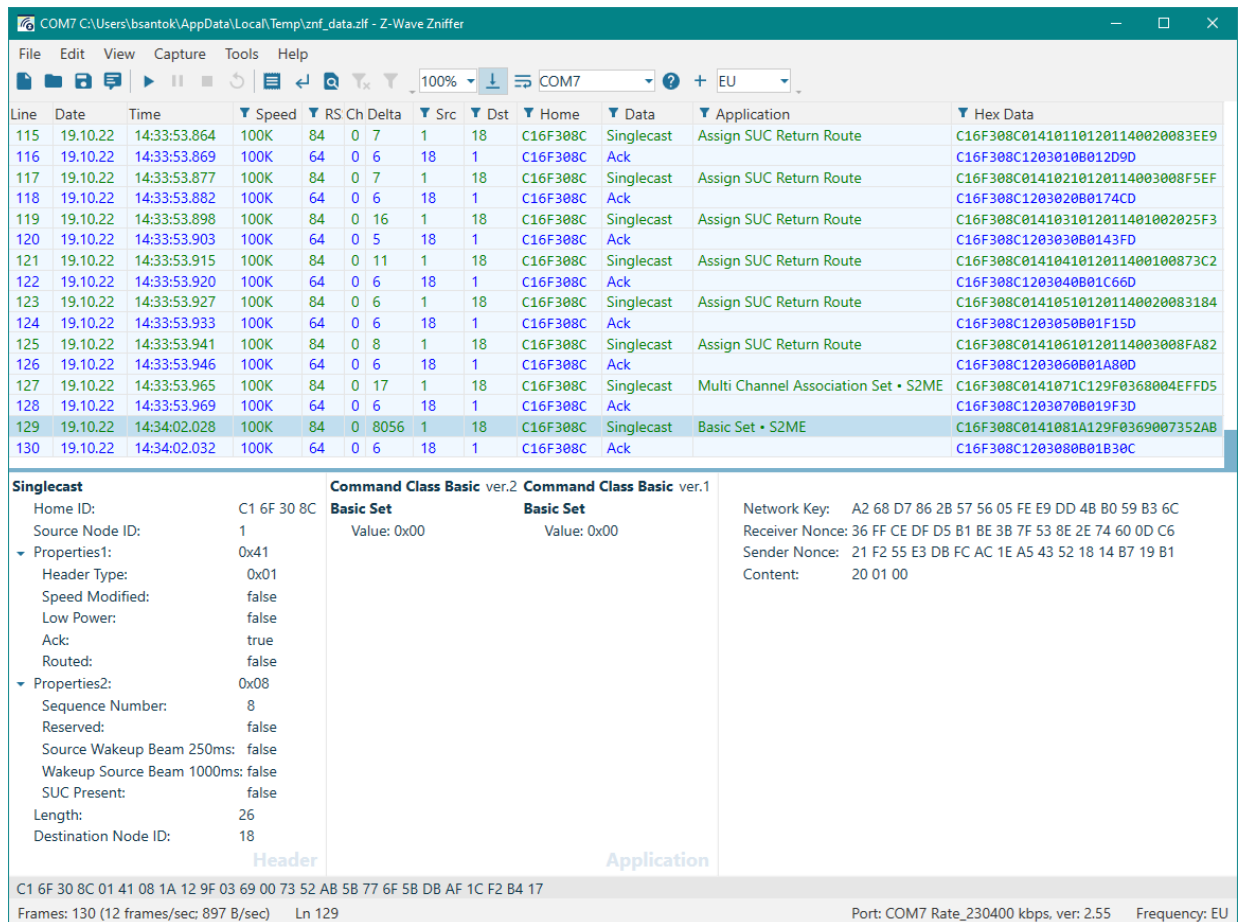


Figure 24: Zniffer Trace Showing a Decrypted S2 Frame for a 'Switch Binary Get'


- 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 25.

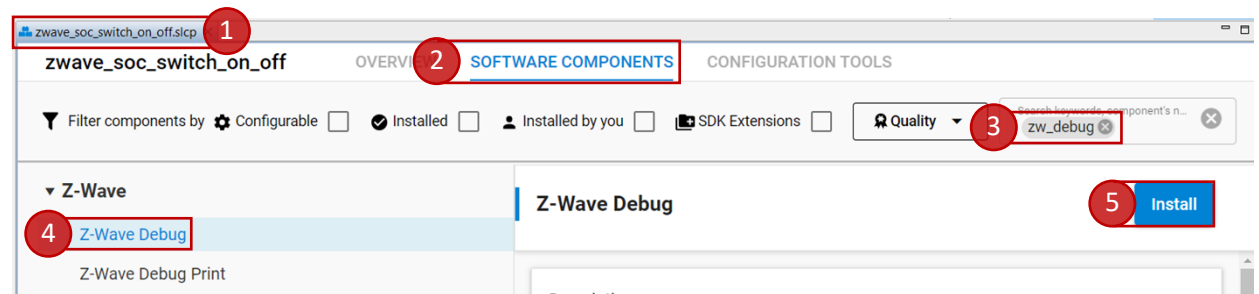



Figure 25: Installing the Z-Wave Debug component

Make sure your application can be built without errors.

From the Toolbar, click on the 'Debug'  button. The code will now be built with debugging enabled and automatically be 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 26, 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.

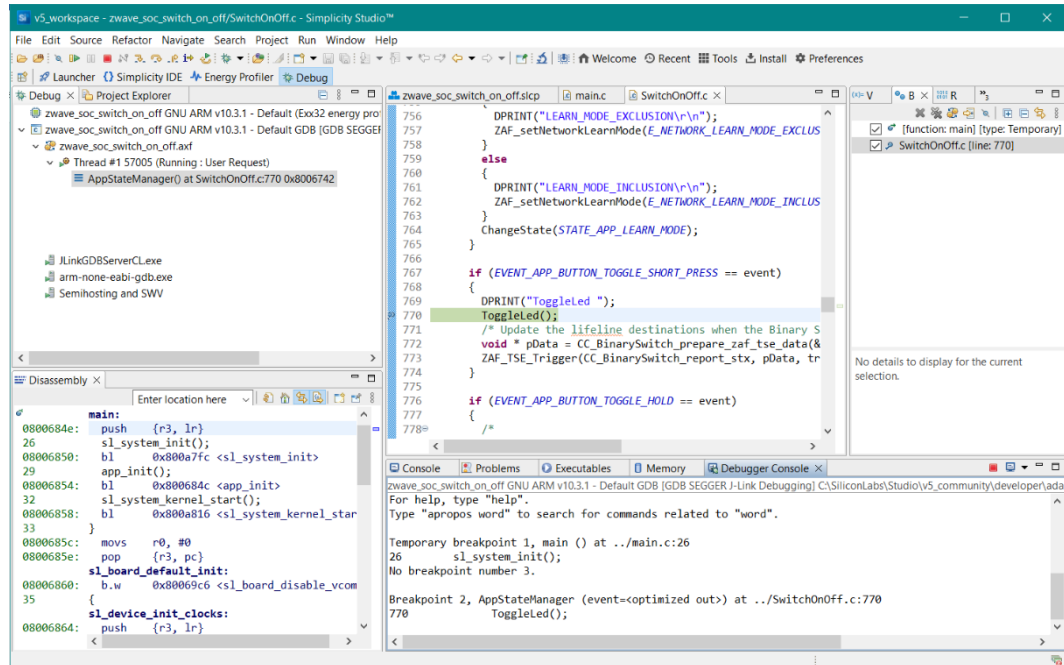


Figure 26: Debugging Switch On/Off

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

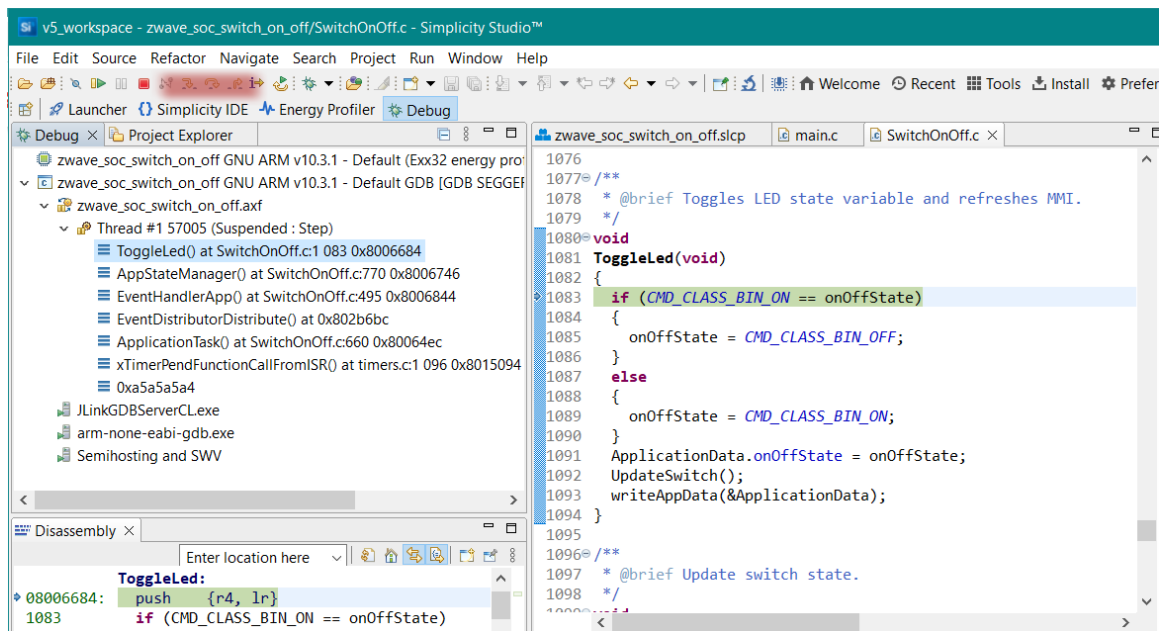


Figure 27: Using Step Into to Debug the Functionalities of ToggleLed

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 28.

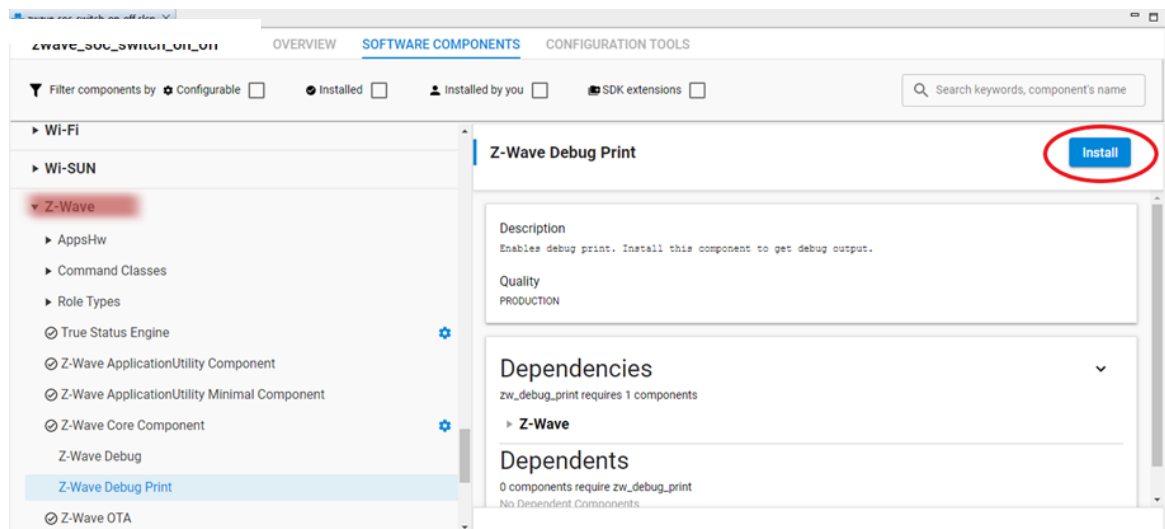


Figure 28: Install the Z-Wave Debug Print Component

Then uncomment the *define* line in the Include-section, as shown in Figure 29.

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

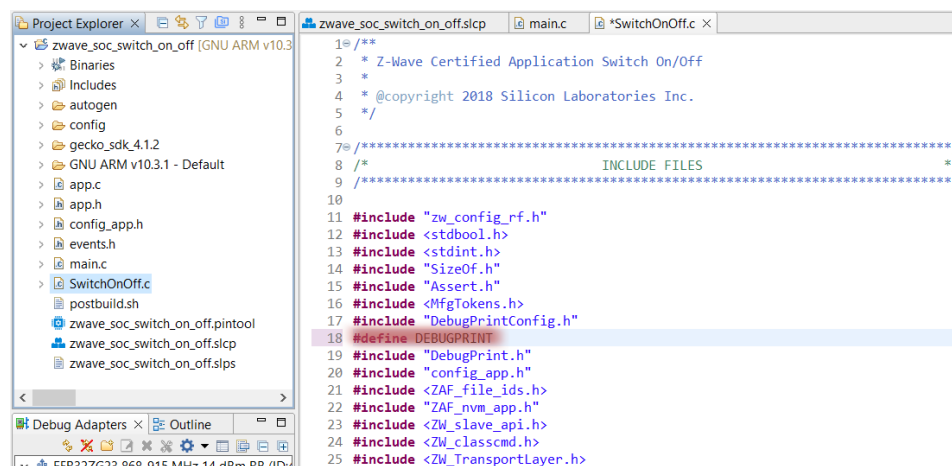


Figure 29: Enable Serial Connection for Debugging

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 30.

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.

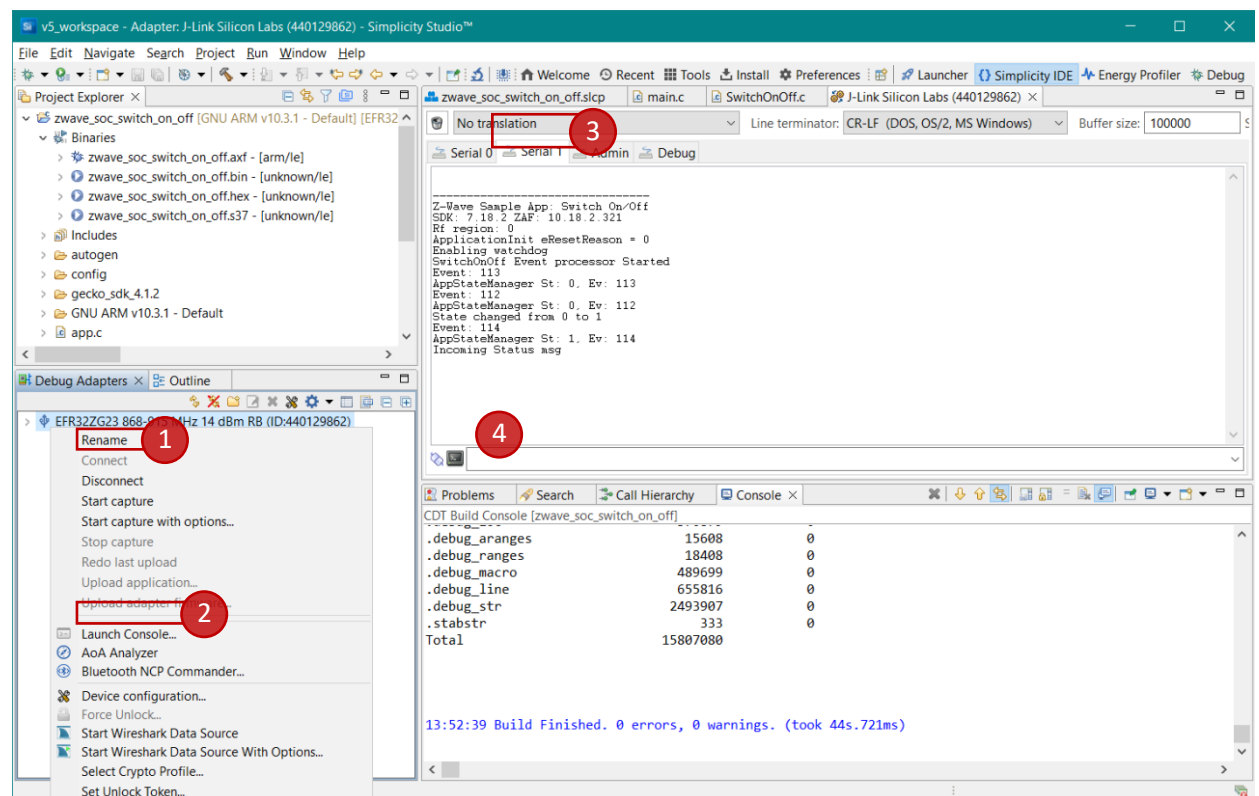


Figure 30: Debug Using Serial Connection

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 CLI component modifies the power consumption for sleeping applications.

By default, the sleeping sample applications, such as the Door Lock Keypad, Multilevel Sensor and Sensor PIR do not include the CLI component.

The ZWave CLI component includes common commands such as factory reset, learn mode start or stop. Most of the sample applications have additional commands implemented, based on their specific behavior.

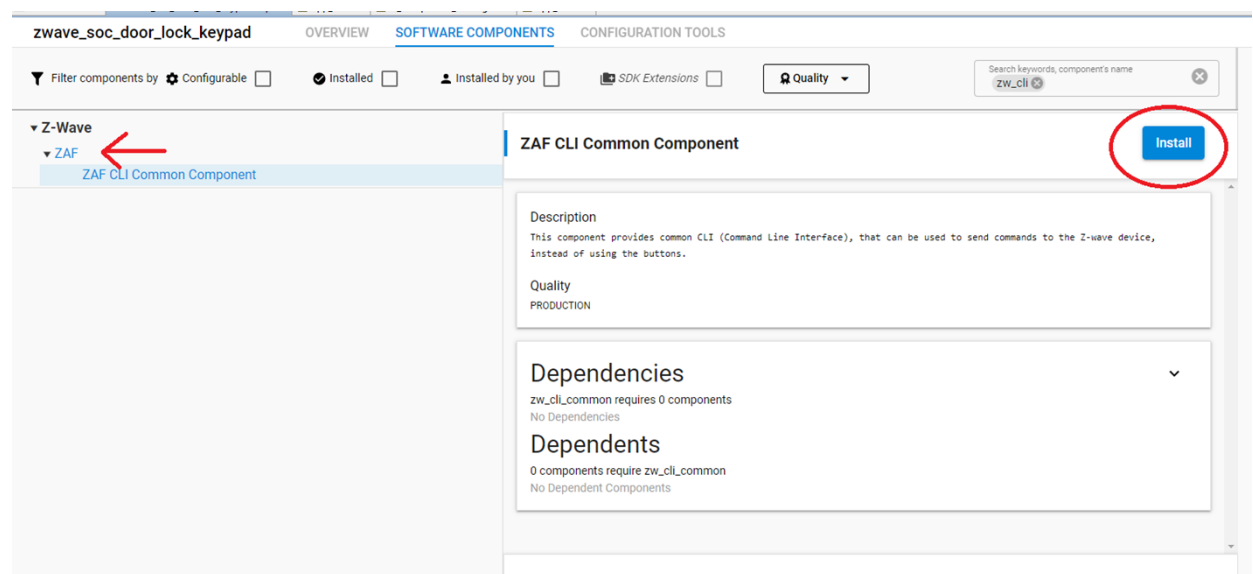


Figure 31: Installing ZAF CLI Common Component

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 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, 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).

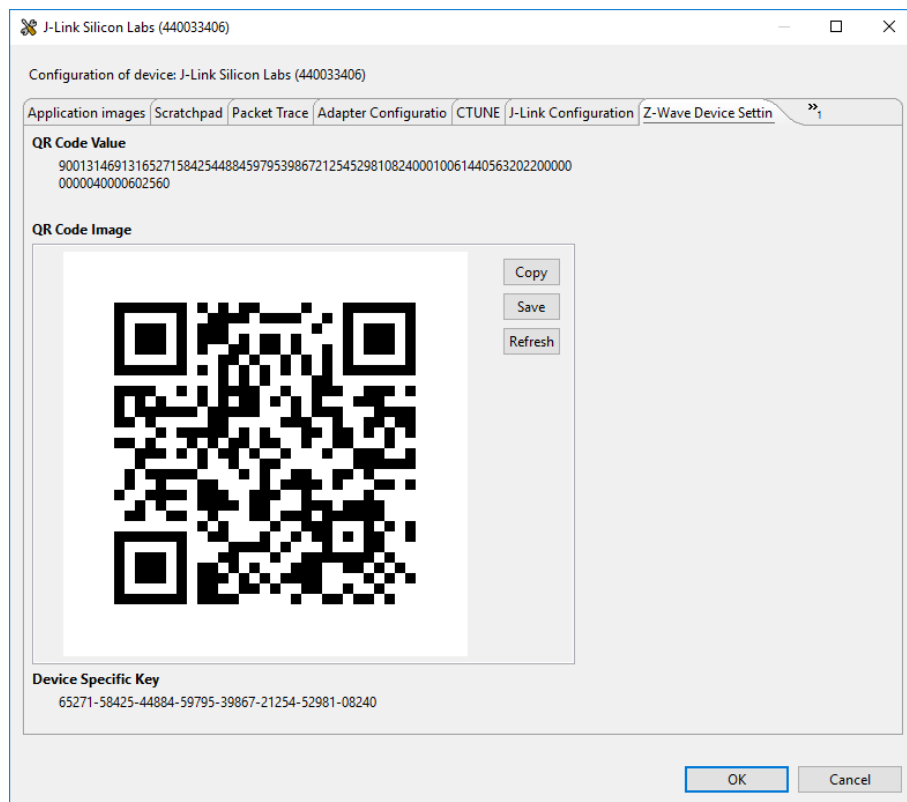


Figure 32: Device Configuration Dialog Showing Z-Wave Device Settings

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-Wave 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 33.

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 Beta** uses a **USB** connection. This is the only available option for the *BRD2603A Development Kit* and the *BRD2705A Explorer Kit*.
- 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].

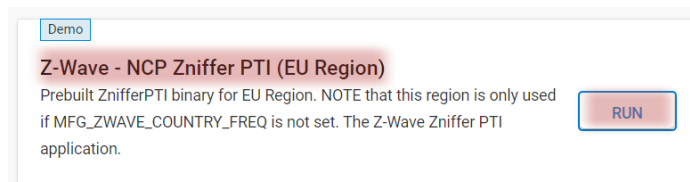


Figure 33: Flashing the Z-Wave - NCP Zniffer PTI Application

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.

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