Instruction

Z-Wave 700 Getting Started for End Devices

Document No.:	INS14280
Version:	4
Description:	-
Written By:	COLSEN;JFR
Date:	2019-09-12
Reviewed By:	JFR;NOBRIOT;JCC;SCBROWNI;NTJ;COLSEN;JSMILJANIC
Restrictions:	Public

Approved by:

Date	CET	Initials	Name	Justification
2019-09-12	05:02:46	NTJ	Niels Johansen	

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

Table of Contents

1 ABBREVIATIONS....1

2 INTRODUCTION....1

2.1 Purpose....1
2.2 Audience and Prerequisites....1

3 WELCOME TO Z-WAVE....2

4 THE Z-WAVE DEVELOPMENT KIT....3

4.1 Development Kit Hardware....3
4.2 Prepare the Hardware....4

5 INSTALL THE Z-WAVE SDK....5

5.1 Connect your Hardware....5
5.2 Install Simplicity Studio....5
5.2.1 Updating Adapter Firmware....7
5.2.2 Associate Simplicity Studio with the Hardware....7
5.2.3 Functionality in the Launcher Perspective....8
5.3 Additional Tools Needed for Z-Wave Development....10
5.3.1 Z-Wave PC Controller....10
5.3.2 Z-Wave Zniffer....13

6 RUN YOUR FIRST Z-WAVE SAMPLE APPLICATION....15

6.1 Changing the Frequency....17
6.2 Reprogramming a Sleeping Device....18

7 DEBUG A Z-WAVE APPLICATION....19

7.1 Debugger....19
7.2 Serial Debug....20

8 MEASURE THE POWER CONSUMPTION USING THE ENERGY PROFILER....22

8.1 Accuracy and Performance....24
8.2 Code Correlation....25

9 NEXT STEP IN DEVELOPING....27

9.1 Next Step for Software Developers....27
9.2 Next Step for Hardware Developers....28
9.3 Certification....28

10 APPENDIX A: READING OUT QR CODE AND DSK....29

REFERENCES....30

Table of Figures

Figure 1: Content of the Z-Wave Development Kit 3
Figure 2: Install Required Device Inspector 6
Figure 3: Select a Device to Install the Needed Software Packages 6
Figure 4: Device Firmware Update 7
Figure 5: Launcher Perspective Associated with Hardware 8
Figure 6: Launcher Perspective (Your Installed Version May Be Different) 9
Figure 7: PC Controller with 1 End Device Added 11
Figure 8: PC Controller, Command Classes View—Send Switch On Command 12
Figure 9: PC Controller, Command Classes View—Send Switch Get Command 13
Figure 10: Zniffer Trace Showing a Decrypted S2 Frame for a 'Switch Binary Get' 14
Figure 11: Sample Application Build Completed and Binaries are Generated 15
Figure 12: Flash Programmer with Selected Device and File to be Flashed 17
Figure 13: Recover a Sleeping Device by 'Unlock Debug Access' 18
Figure 14: Debugging Switch On / Off 19
Figure 15: Using Step-Into to Debug the Functionalities of ToggleLed 20
Figure 16: Enable Serial Connection for Debugging 21
Figure 17: Debug Using Serial Connection 21
Figure 18: Energy Profiler Showing the Consumption of SensorPIR Sample Application 22
Figure 19: Measure Average Consumption 23
Figure 20: Simultaneously Measuring the Consumption of Multiple Devices 23
Figure 21: Scope View 24
Figure 22: Start Energy Profiler with Code Correlation 25
Figure 23: Code Correlation Associates High Energy Consumption with Source Code 26

Table of Tables

Table 1: Overview of a Possible Frequencies 16

1 Abbreviations

Abbreviation	Explanation
DSK	Device Specific Key
IDE	Integrated Development Environment
SDK	Software Development Kit

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 700 development.

There are no prerequisites. It is, however, strongly recommended that one purchase the Z-Wave Development Kit as the development environment auto-discover hardware and setup accordingly. But 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 product to the third-party test house.
5) Once a product passes certification and product and packaging labels are approved, then the Z-Wave Cert 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 Kit

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.

To get the latest news of the Z-Wave 700, refer to:
https://www.silabs.com/products/wireless/mesh-networking/z-wave/700-platform

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.

4.1 Development Kit Hardware

The Z-Wave development kit contains the following:

- WSTK Main Development Board, 2 pcs.
- BRD4200A Radio Board with ZGM130S intended end device development, 2 pcs.
- BRD8029A EXP Board, 2 pcs.
- UZB7 Controller USB Dongle.
- Zniffer USB Dongle.

Figure 1: Content of the Z-Wave Development Kit

For a more in-depth description of the various hardware components, refer to "How to Use Certified Apps in Z-Wave 700" [1].

4.2 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 BRD4200A to the WSTK Main Board.
2) Connect the EXP board to the extension port of the WSTK Main Board.
3) Set the Power switch in AEM position.
4) Connect the WSTK Main Board using a USB cable to the PC.

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

- BRD4200A 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 setup and connected the hardware as described in section 4.2.

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 installation application. When Simplicity Studio first launches, it presents a License Agreement dialog. Accept the terms of the agreement and click *Next*.

3) Choose a destination location. You can leave it at the default location. Click *Next* and then lick *install*.
This will install Simplicity Studio; however, you still need to install the Z-Wave SDK. See next step.

4) When the application launches, use your Silabs.com account to login to get access to the SDKs.

5) After logging in, Simplicity Studio adds software information. 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 cable and prompt you to download a Device Inspector. Click *Yes*. See Figure 2.

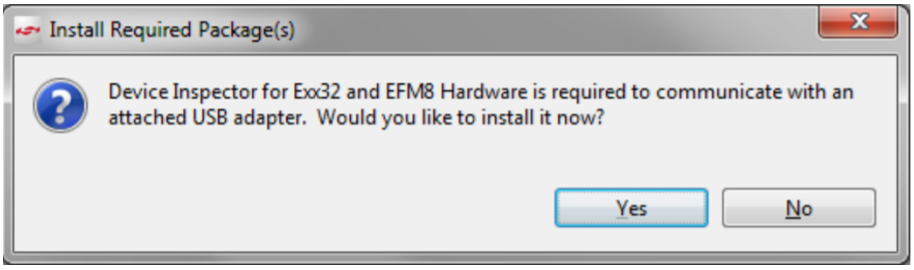


Figure 2: Install Required Device Inspector

6) After some additional items are installed, you are offered the option of installing by device or installing by product group. In this guide we will be using the '*Install by Device*' 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 3.

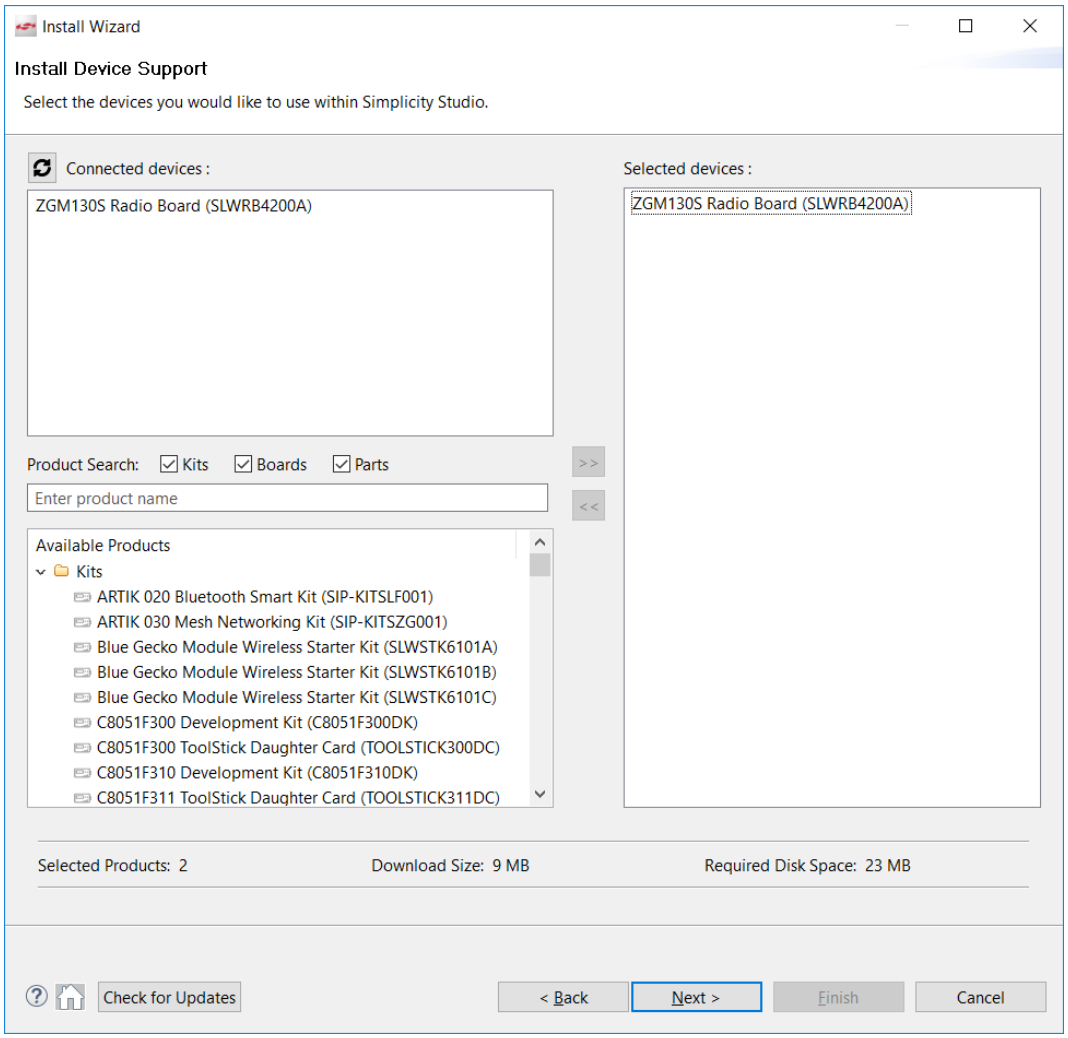


Figure 3: Select a Device to Install the Needed Software Packages

7) The next dialog will show the available software. You must be logged in to download the available software.

8) The installer options dialog shows the tools and software packages that can be installed. You can uncheck anything you do not want to install, but it is recommended that you leave everything as is.

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 are invited to take a tour. If you are not already familiar with Simplicity Studio, take the short tour to see the most frequently used features.

5.2.1 Updating Adapter Firmware

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

Figure 4: 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. See Figure 5 for a correct setup.

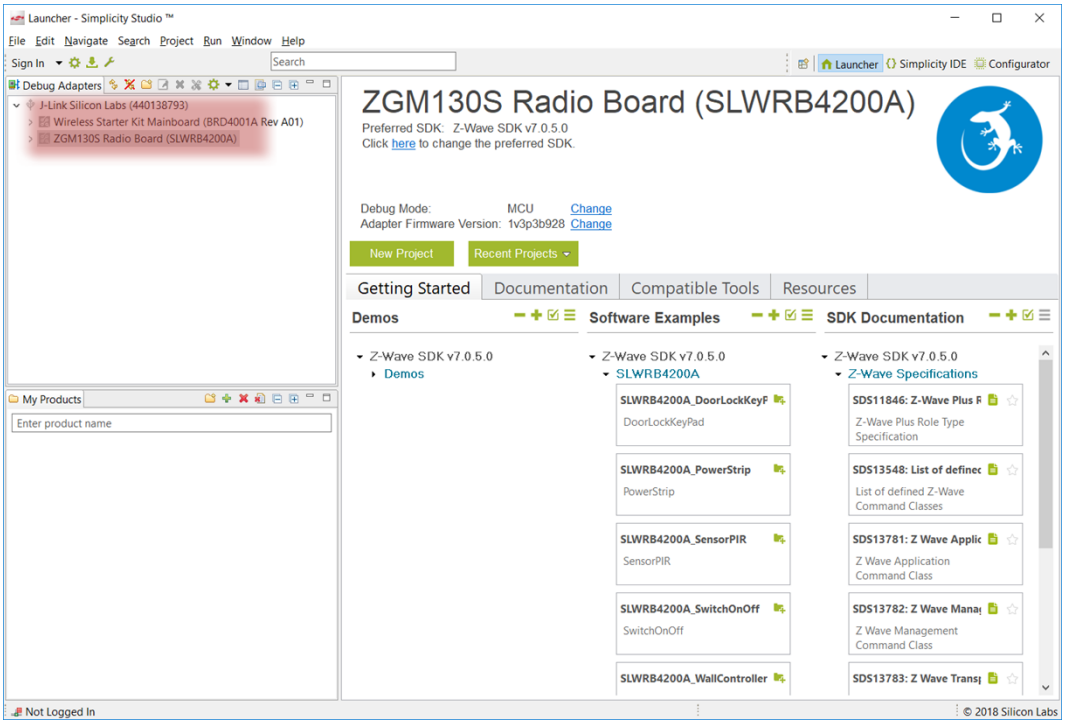


Figure 5: Launcher Perspective Associated with Hardware

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 a number of tiles or panes, called views, as well as the content in those views. You can perform a number of functions in the Launcher Perspective, as shown in Figure 6.

Some of the most important functions to know about are the following:

1) In the toolbar, you can
   - Sign in or out
   - Open application settings
   - Update your software and firmware
   - Open the Tools menu
   - Search for information including entries in the Community forums.

2) Change perspectives
   - As you open the Simplicity IDE or other tools, buttons for their perspectives are displayed in the upper right. Use those 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) Change your preferred SDK. When working with Z-Wave devices, the preferred SDK should be Z-Wave SDK.

4) Update adaptor firmware (displayed only if a device is connected).

5) 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 and the IC on the board.

6) Create solutions of multiple parts. If you are developing for complex networks with a number of different parts involved, you can add them all to the solution and then select the one you are working on from the list. You do not need to have the hardware connected to your computer.

7) Accessing Documentation and Other Resources
   - The Getting Started tab provides access to demos, example applications, and stack-related documentation.
   - The Documentation tab lists documentation about the stack and about the hardware on the right, and documents you selected as favorites on the left. Click the star icon on any document to show it in the My Favorite Documents list.
   - The Compatible Tools tab is an alternative way to access the tools available through the Tools dropdown.
   - The Resources tab provides access to support, marketing collateral, and the Silicon Labs community.

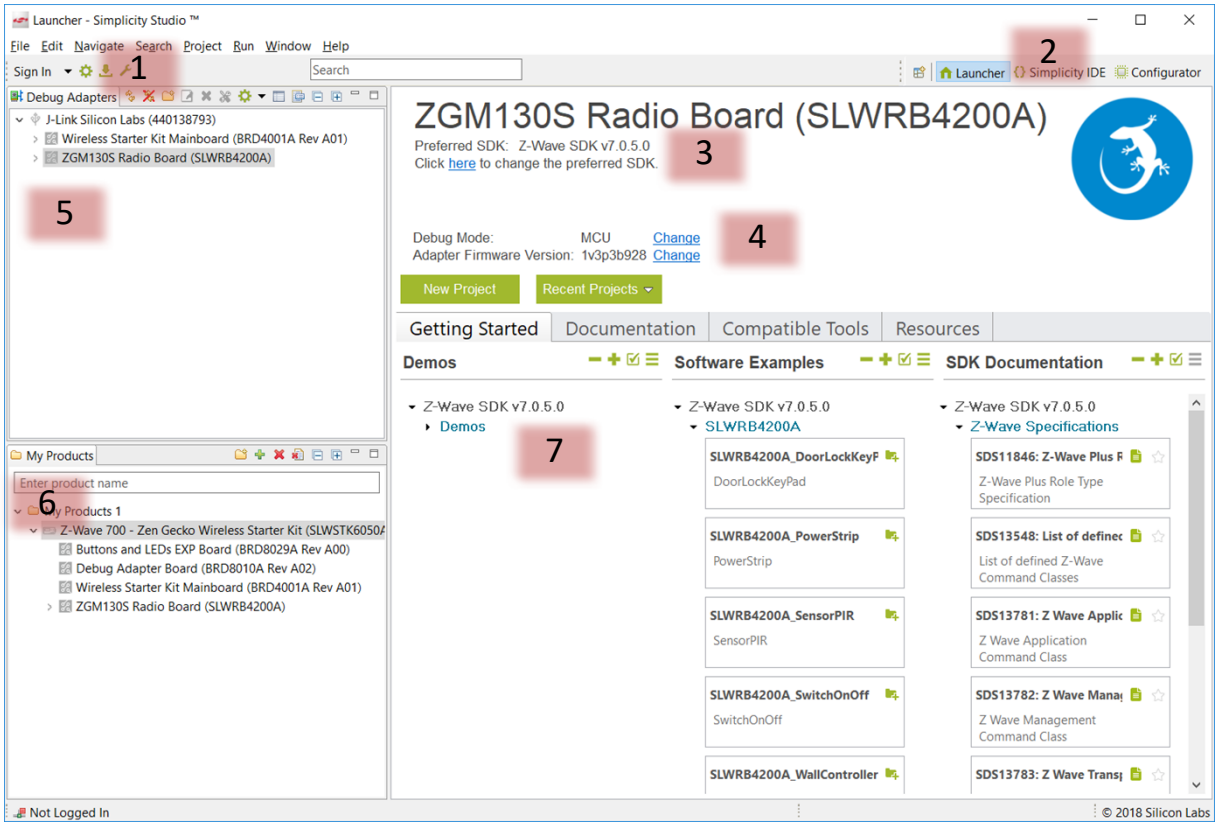


Figure 6: Launcher Perspective (Your Installed Version May Be Different)

5.3 Additional Tools Needed for Z-Wave Development

A couple of additional tools are needed for developing and debugging Z-Wave applications. These tools can be downloaded from this link:
https://www.silabs.com/products/development-tools/software/z-wave

5.3.1 Z-Wave PC Controller

The Z-Wave PC Controller is an application for communicating with Z-Wave 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, 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) Install driver.
3) Open PC Controller and click on the ‘Settings-wheel’ to select the correct COM port.
4) Click on Network Management when connection to the UZB Controller has been established.

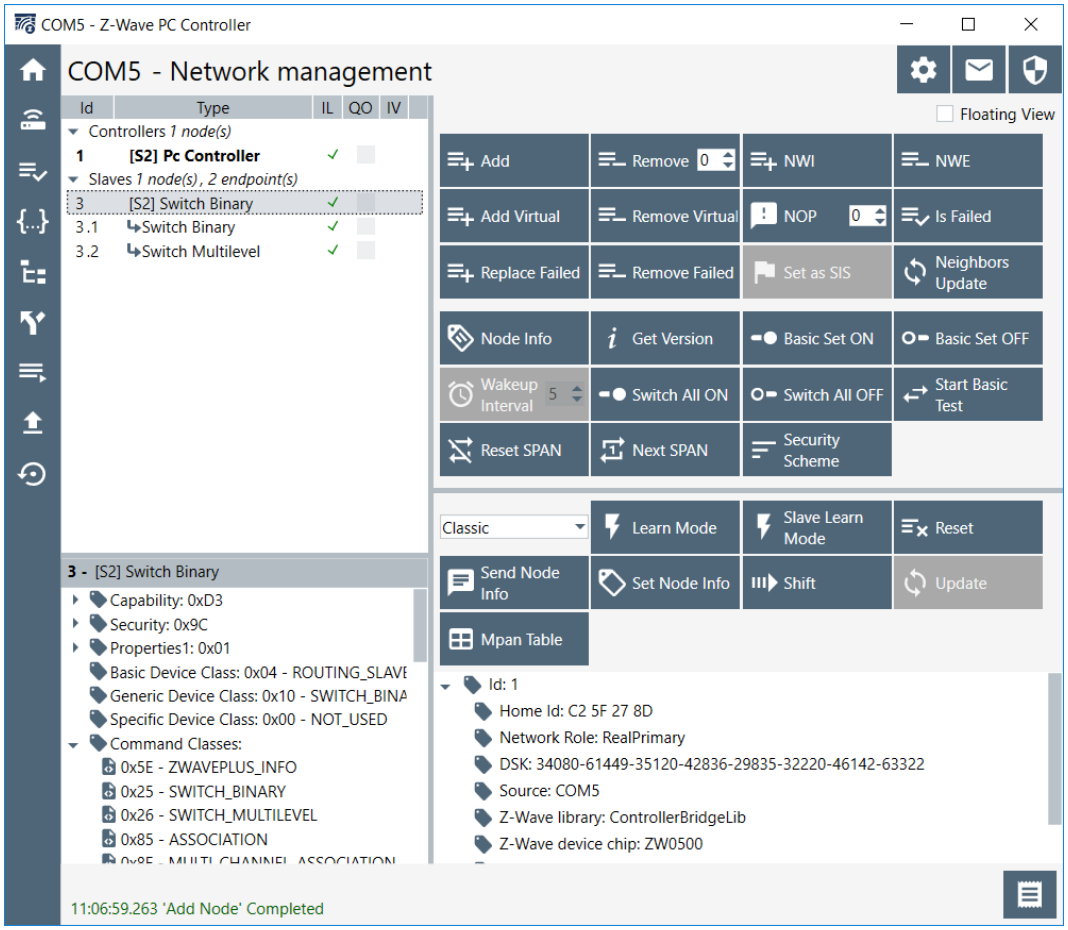


Figure 7: PC Controller with 1 End Device Added

5) 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]:

   4.2 Network Management View

   5.2.1 How to Add a Node

   5.2.3 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 button *S4*. The included nodes will appear in the 'Slaves' section with a Node ID.

   Refer to 10 for instructions on how to read out the DSK key needed for S2-authenticated inclusion.

6) 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]:

5.4 Command Class View

For sending a command to turn on the included switch, go to '*Command Classes*' view. Then select a supported command class, in this case the '*Switch_Binary*'. Set the value and click on '*Send*'. The value on the switch should now have been set accordingly. Refer to Figure 8.

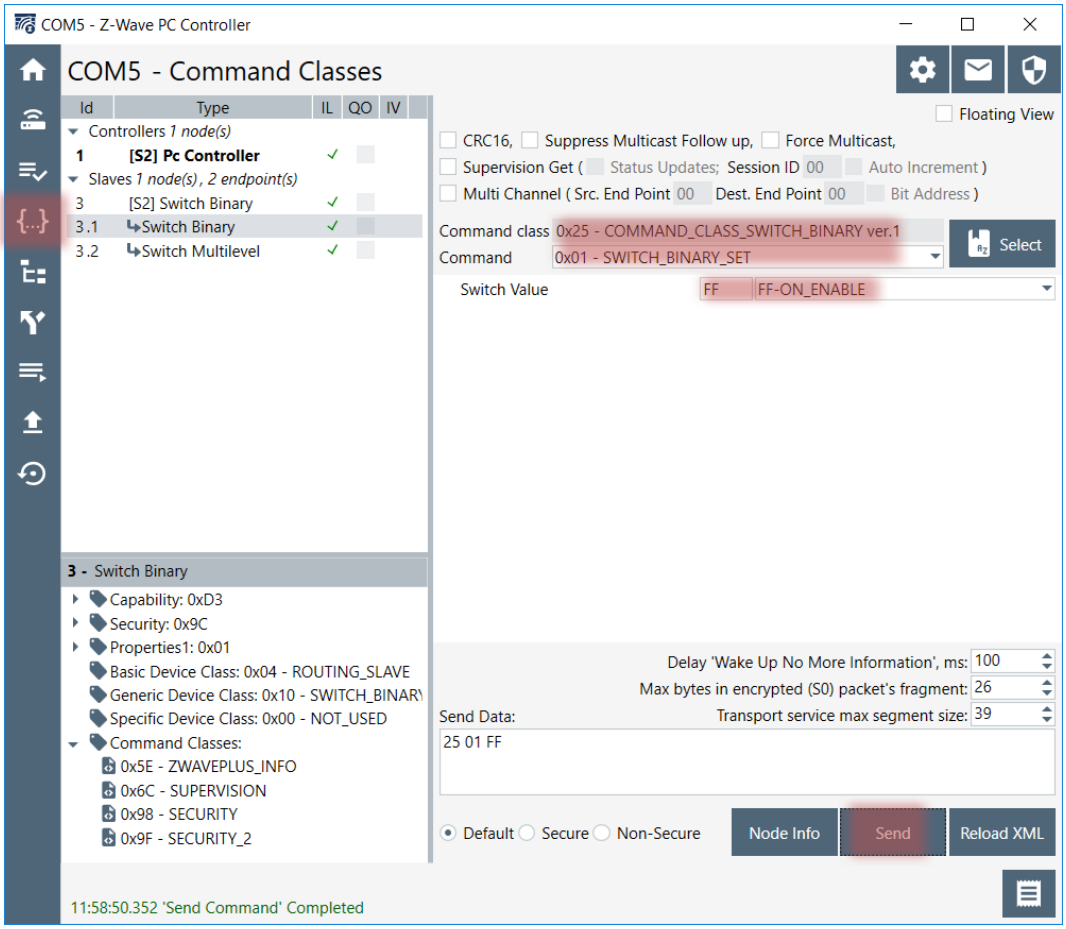


Figure 8: PC Controller, Command Classes View—Send Switch On Command

7) Send a Get Command to verify the switch was set accordingly. Refer to Figure 9.

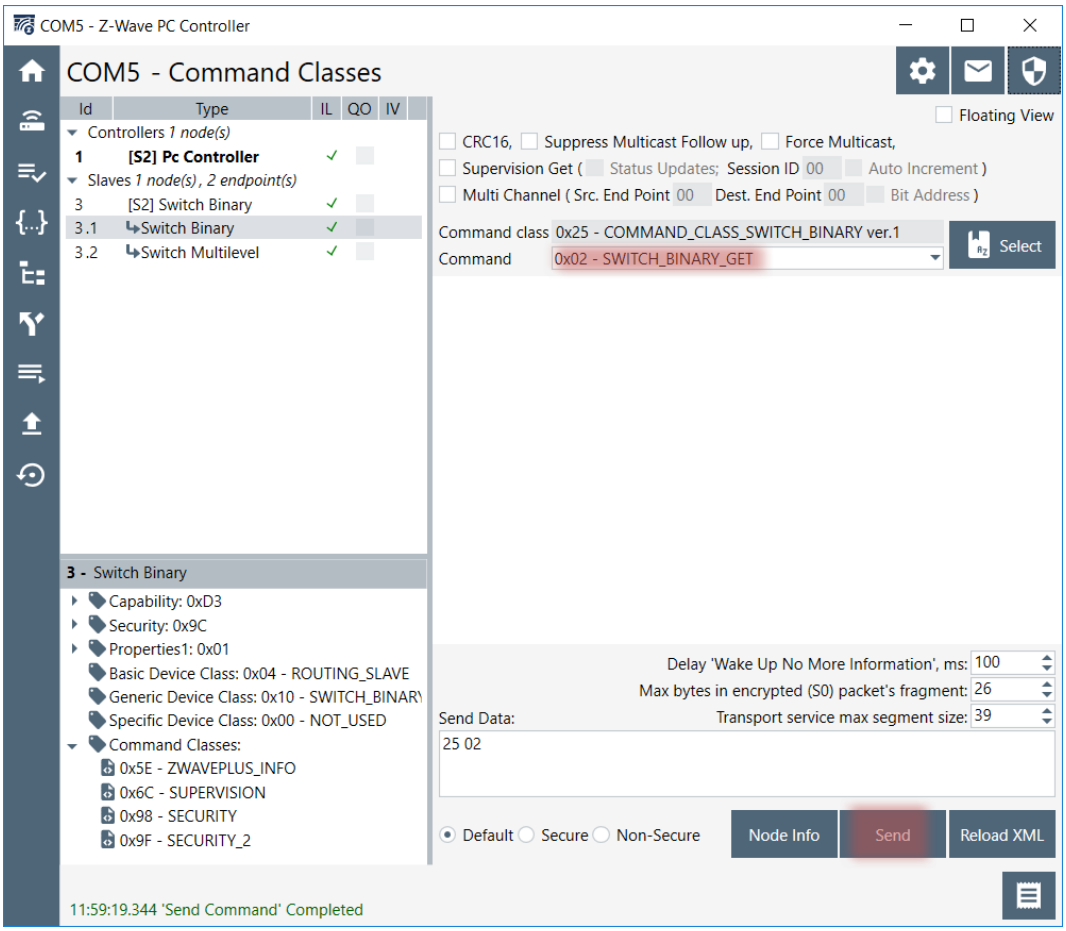


Figure 9: PC Controller, Command Classes View—Send Switch Get Command

Now verify using the Zniffer – see section “5.3.2 Z-Wave Zniffer”.

5.3.2 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.

3) Open Zniffer and click on '*Detect Zniffer Modules*'  in the tools bar. Then select the correct COM Port and frequency to listen in.
4) When ready, click the '*Start*' button to capture Z-Wave frames.

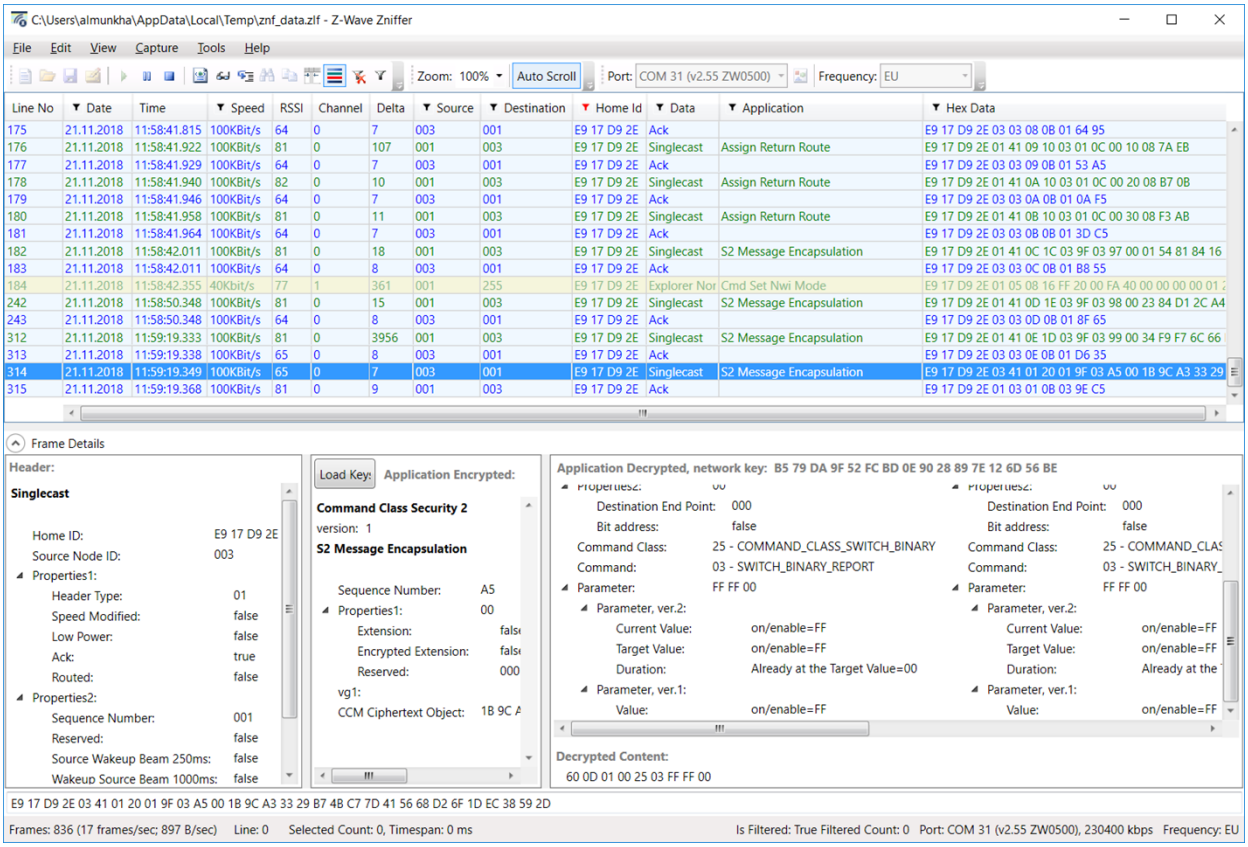


Figure 10: Zniffer Trace Showing a Decrypted S2 Frame for a 'Switch Binary Get'

5) From this view you can now see the Z-Wave frames being send / 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"  .

6 Run Your First Z-Wave Sample Application

Having all the hardware and software setup, 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 Z-Wave.

Then, in the Software Examples, you will see all the available Z-Wave sample applications. This guide will open, compile, and program a device with the SensorPIR Sample Application.

When you click on SensorPIR in the Launcher Perspective, the studio will automatically open a local copy of the example and open the Simplicity IDE Perspective. See Figure 11.

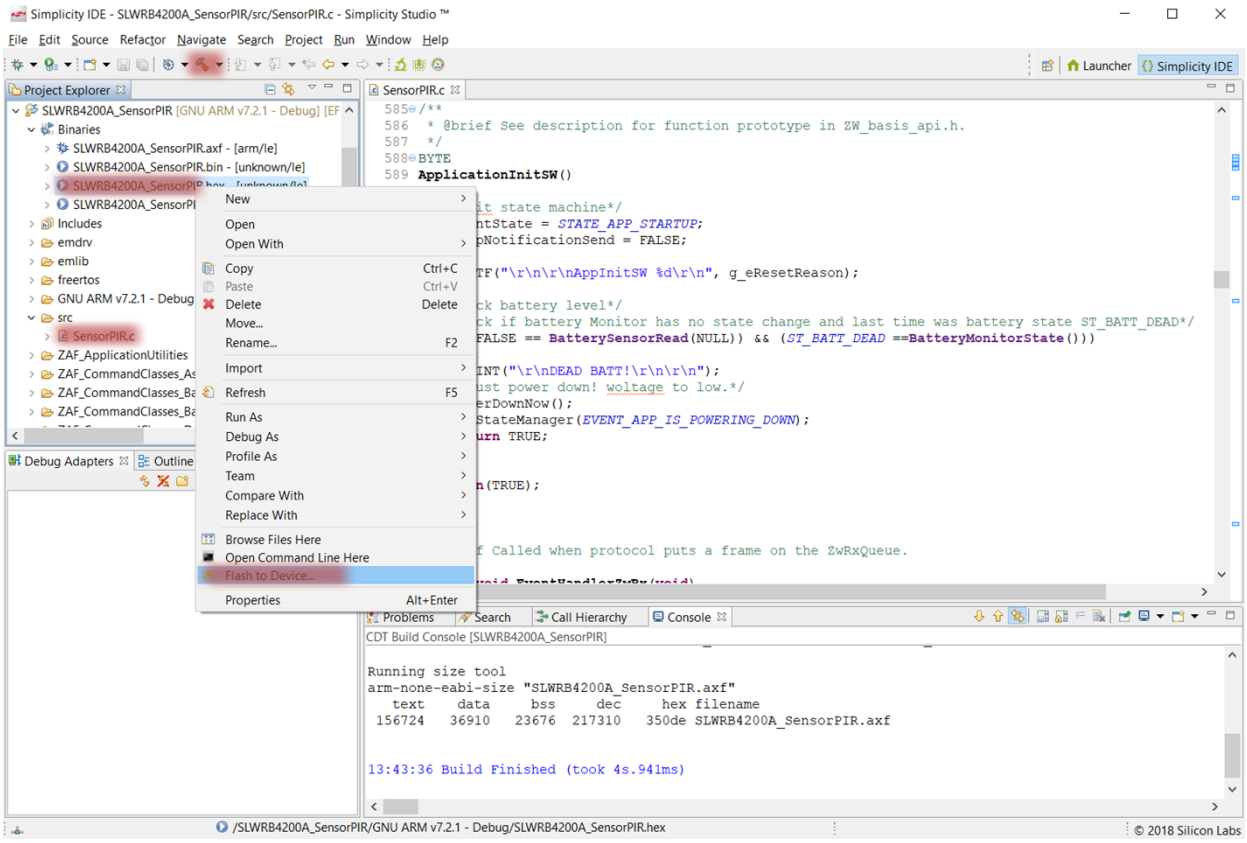


Figure 11: Sample Application Build Completed and Binaries are Generated

In the folder named '*src*' you will find the main file of the sample application, in this example called *SensorPIR.c*.

In the source code, search for '*APP_FREQ*' in order to set the desired frequency. In this guide we will be using the European frequency, thus we enter '*REGION_EU*'. Click on the '*Build*' button to start building the project.

Navigate to Silicon Labs website, to see which countries has been approved for the Z-Wave RF. Refer to Table 1 for a complete list of supported frequencies by the SDK.

Table 1: Overview of a Possible Frequencies

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

When the build finishes after a short while, a new folder named '*Binaries*' is shown in the Project Explorer window. Expand the folder and right click on the *.hex file to select '*Flash to Device..*'. Refer to Figure 11 again.

Select the connected hardware in the pop-up window. If you followed the instructions in this guide, the 'Flash Programmer' should now be prefilled with all the needed data, and you are ready to click on 'Program'. Refer to Figure 12.

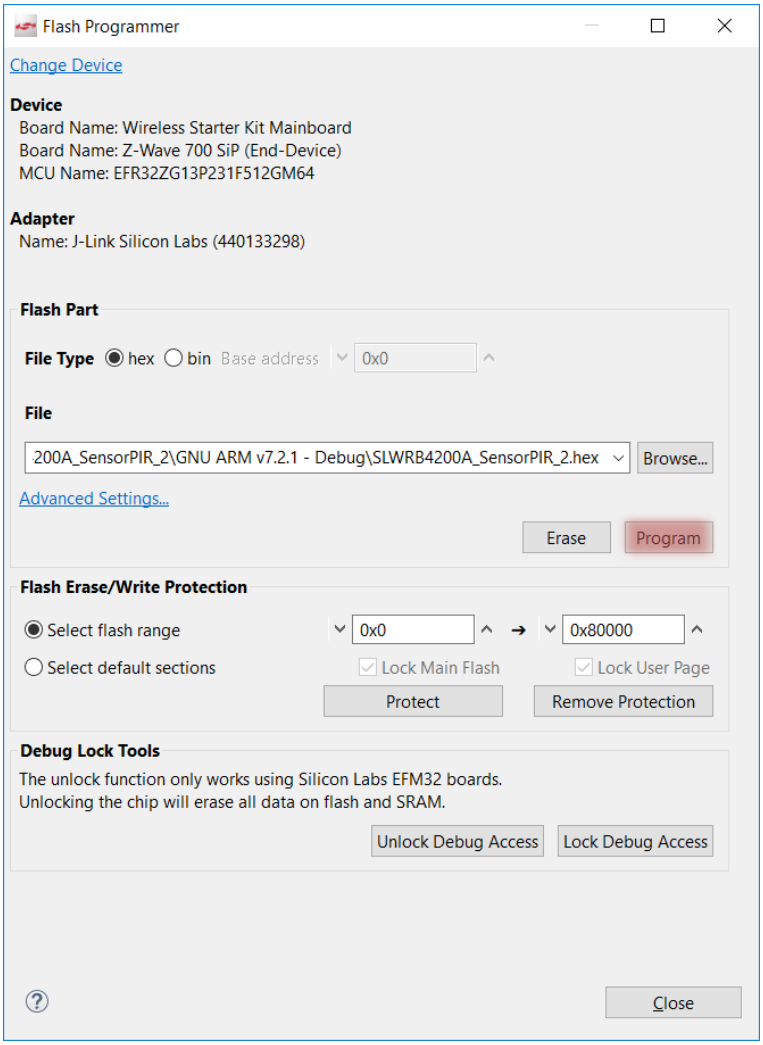


Figure 12: Flash Programmer with Selected Device and File to be Flashed

After a short while the programming finishes, and your end device is now flashed with a Z-Wave sample application. Include it in a Z-Wave network using the PC Controller to start testing the functionalities!

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

6.1 Changing the Frequency

The internal NVM is not erased between reprogramming. This allows a node to stay in a network and keep the same network keys when you reprogram it.

If you need to change the frequency at which the module operates, you need to “Erase” the chip before the new frequency will be written to the internal NVM. The “Erase” button is also seen in Figure 12.

6.2 Reprogramming a Sleeping Device

After a device has been programmed with SensorPIR – or any other deep sleeping application – it will throw an error when trying 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.

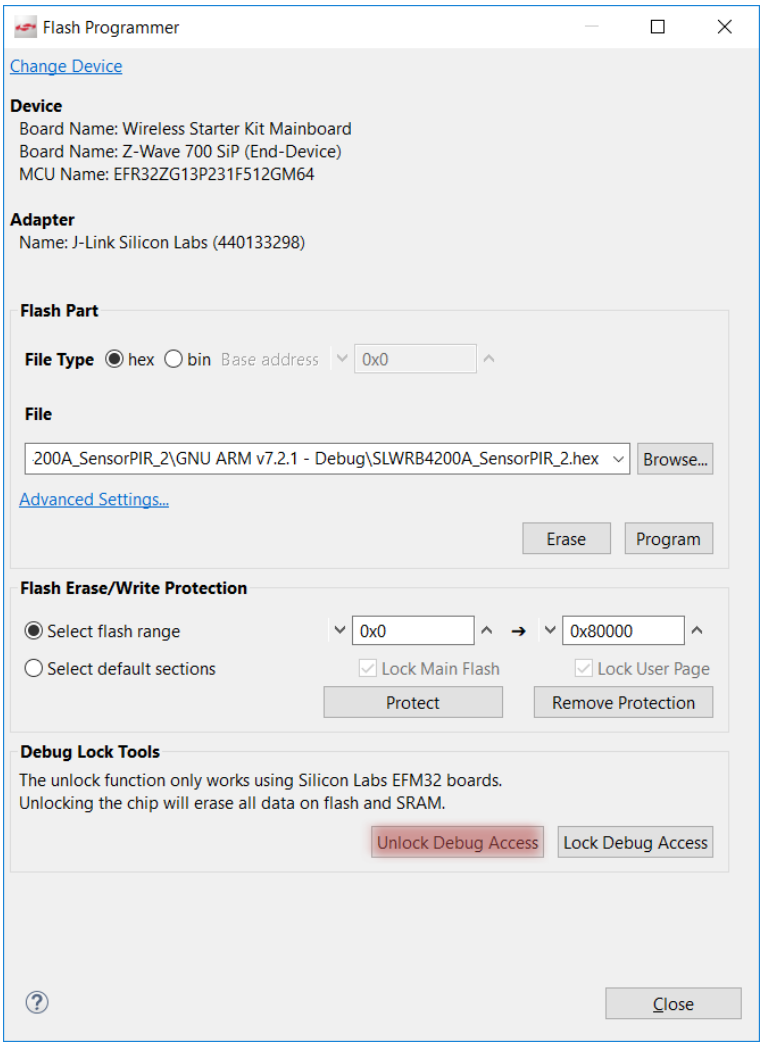


Figure 13: Recover a Sleeping Device by 'Unlock Debug Access'

7 Debug a Z-Wave Application

7.1 Debugger

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

Make sure your application can build without errors. From the tools menu, click on the '*Debug*' symbol  . 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 14, the sample application 'Switch On / Off' is flashed to the device in debug mode. A breakpoint is then set near the ToggleLed-method. When the user presses on the button *S1*, the code will halt execution at the breakpoint and show the relevant information.

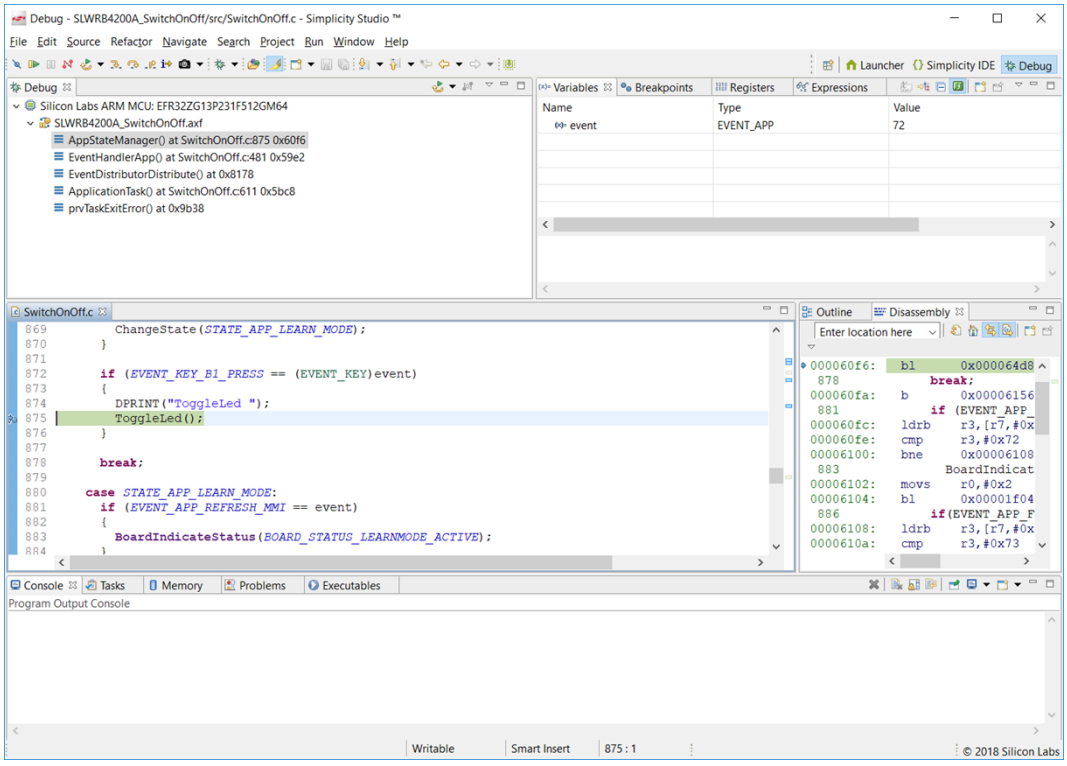


Figure 14: Debugging Switch On / Off

By using the 'Step-Over' and 'Step-Into' functionalities of the debugger, it is possible to follow the flow in the application.

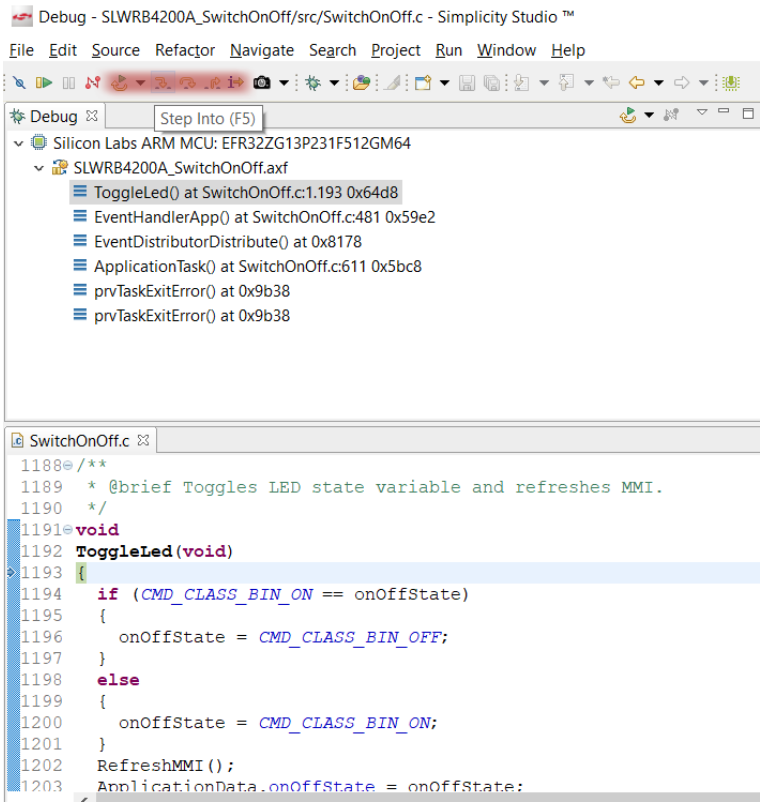


Figure 15: Using Step-Into to Debug the Functionalities of ToggleLed

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

7.2 Serial Debug

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

In order to enable the serial debugger, uncomment the *define* line in the Include-section, as show in Figure 16.

Then build the project again and flash the device.

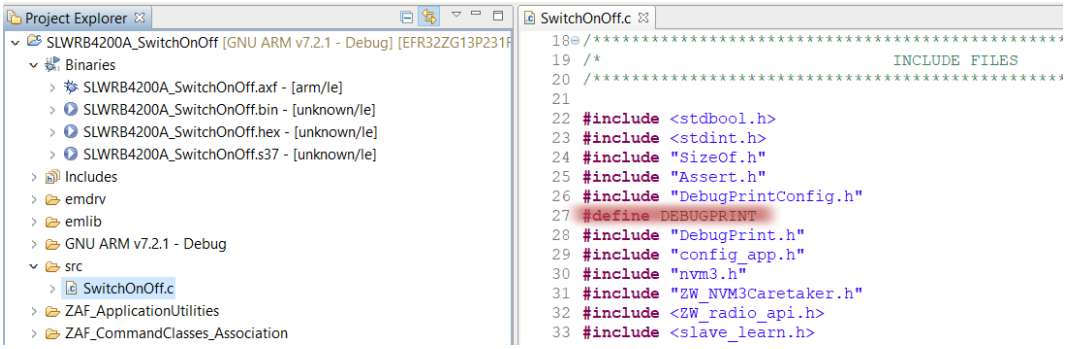


Figure 16: Enable Serial Connection for Debugging

When the device has been flashed, right click on the adapter, click *Connect,* and then *Launch Console*. In the Console view, select *Serial 1*, 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 for debug input. Refer to Figure 17.

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.

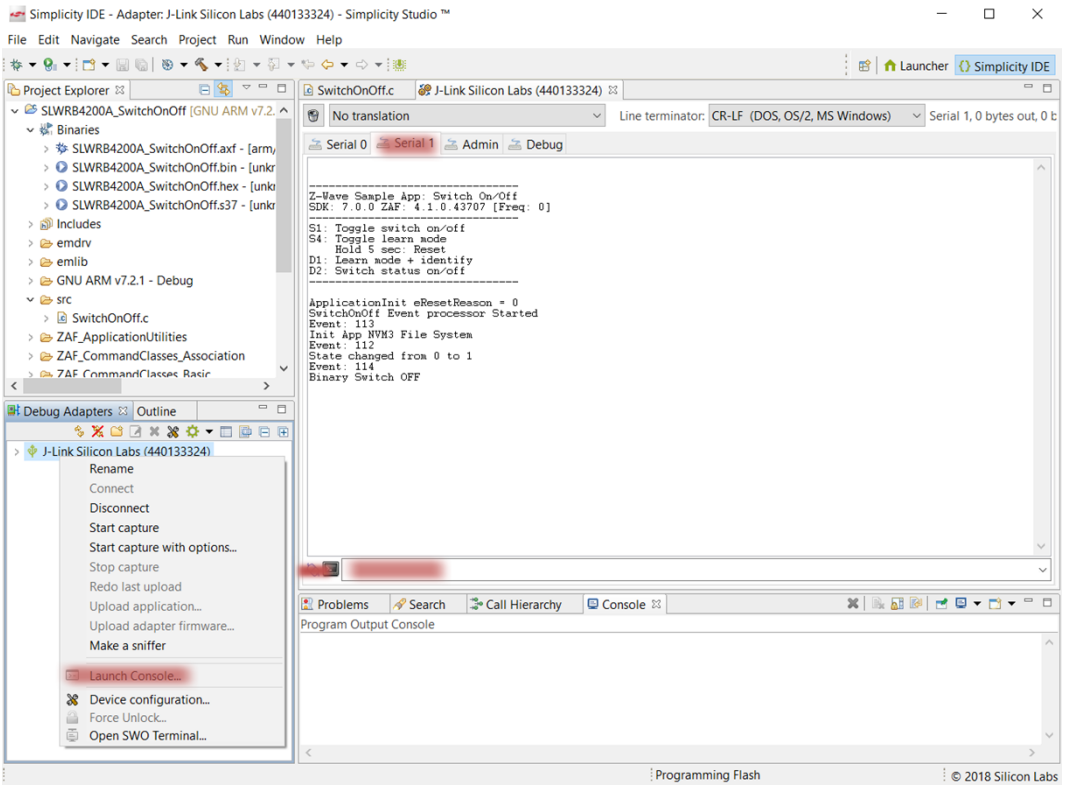


Figure 17: Debug Using Serial Connection

8 Measure the Power Consumption Using the Energy Profiler

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 in order to analyze and improve the power performance of these systems. Real-time information on current consumption is correlated with the program counter providing advanced energy software monitoring capabilities.

Energy profiler is an add-on tool, with which you can easily measure the energy consumption of your device in runtime. You can easily find peak and average consumption, and check for sleep mode current.

Click the '*Open Perspective*' button. Refer to Figure 6, step 2. In this window, click on '  Energy Profiler'. This opens a new perspective called the Energy Profiler.

To start a measurement, click on '*Quick Access*' and select '*Start Energy Capture*'. Then select your connected hardware and the measurement will begin.

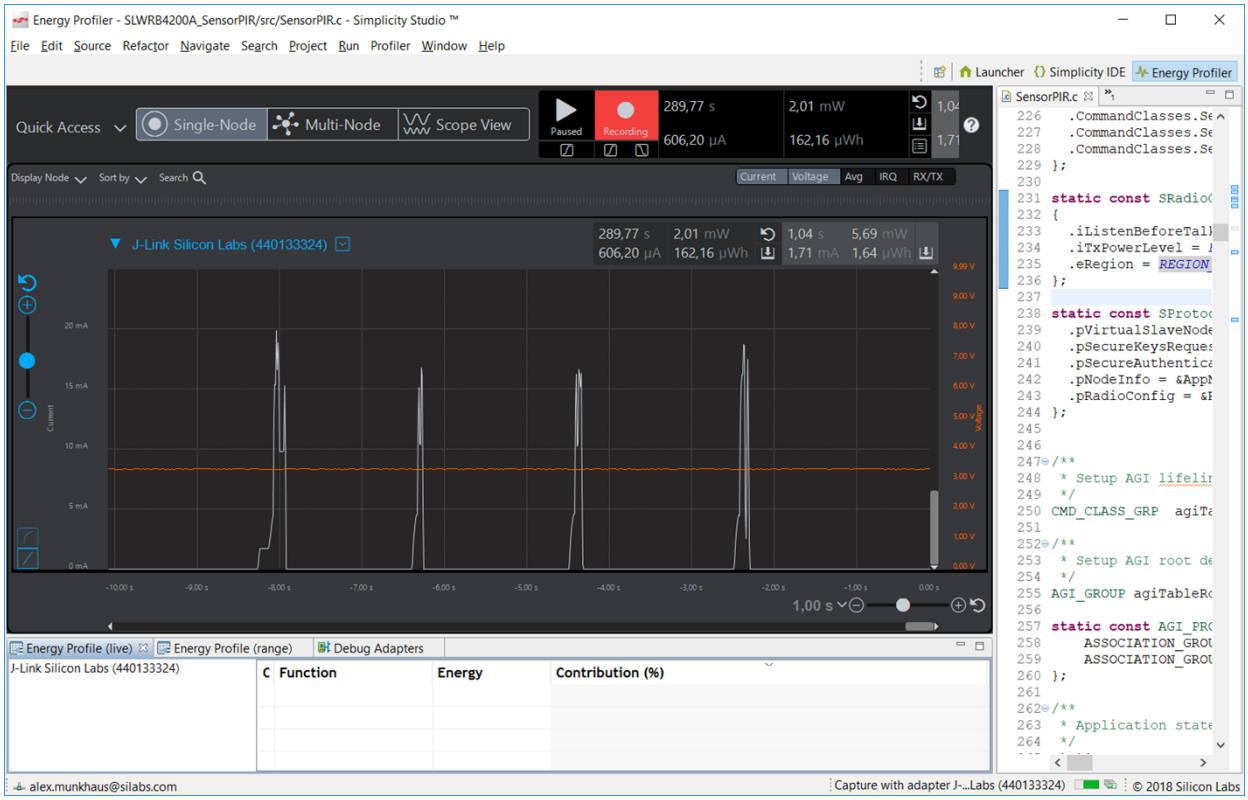


Figure 18: Energy Profiler Showing the Consumption of SensorPIR Sample Application

To measure something meaningful, make sure you have flashed your hardware with one of the sample applications. In Figure 18 you can see the SensorPIR sample application getting activated 4 times, each time waking up from sleep, sending a notification, and then falling asleep again. The first activation is a 'Wake-Up Notification', whereas the 3 following is a 'Motion Detected Notification'.

To measure average consumption, simply click and drag your mouse over a time interval. A new window appears in the upper right corner showing consumption information for the given interval. It is recommended that you measure an average over an advertisement or connection interval to obtain a proper average consumption. Overall average is measured as well, but this is influenced by transient events.

Figure 19: Measure Average Consumption

Multi-Node view is used to display multiple device waveforms and events. It allows you to investigate system behavior and interactions between devices. Two devices are visible in the main portion of the UI, while others are viewed by scrolling. The Multi-Node view groups each individual device's waveform and events together.

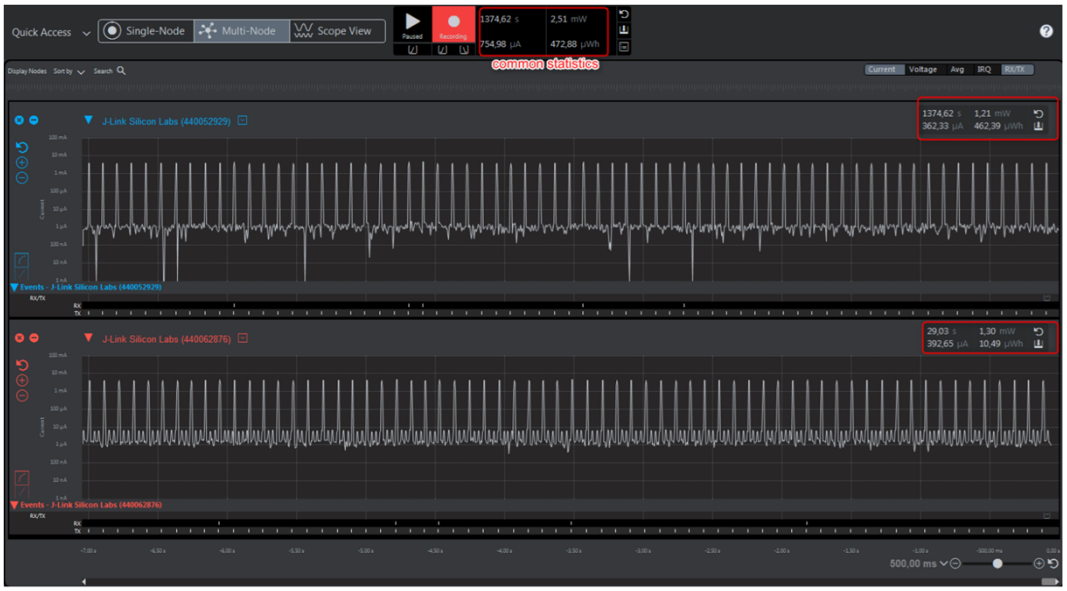


Figure 20: Simultaneously Measuring the Consumption of Multiple Devices

Scope view is used to display multiple device waveforms and events, allowing you to investigate system behavior and interactions between devices. In Scope view, all waveforms are displayed in one waveform area, so that you can see all waveforms without needing to scroll the UI. It also enables the user overlay and align traces like an oscilloscope. All event traces are grouped below the waveform view.

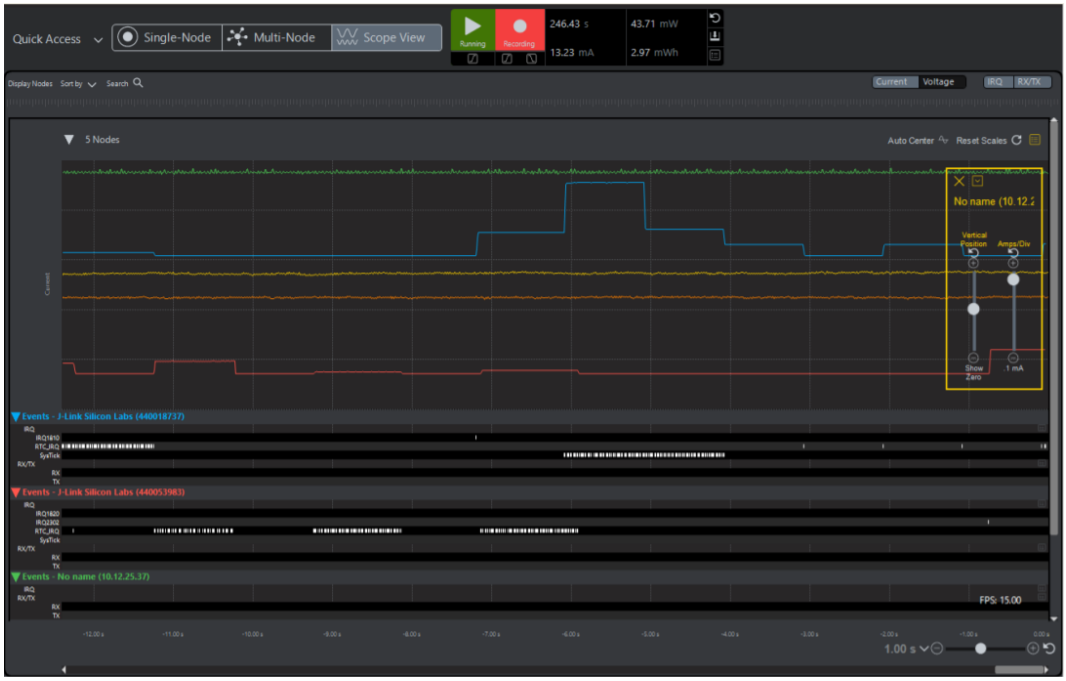


Figure 21: Scope View

8.1 Accuracy and Performance

Advanced Energy Monitor is capable of measuring currents in the range of 0.1 µA to 95 mA. For currents above 250 µA, the AEM is accurate within 0.1 mA. When measuring currents below 250 µA, the accuracy increases to 1 µA. Even though the absolute accuracy is 1 µA in the sub 250 µA range, the AEM is able to detect changes in the current consumption as small as 100 nA. The AEM current sampling rate is 10 kHz.

Note: The AEM circuitry only works when the kit is powered, and the power switch is in the AEM position.

8.2 Code Correlation

Code correlation is one of the most powerful features in Energy Profiler. Energy Profiler captures program execution in conjunction with the energy data. This allows Energy Profiler to calculate the power consumed by each function executed in the application. This data may then be sorted to highlight the portions of an application that consume the most power. This enables the application developer to know where they should concentrate their software development efforts to reduce power consumption.

To enable Code Correlation, simply call the method:

```
BSP_TraceSwoSetup();
```

When this method has been added to the source code, start the Energy Profiler as shown in Figure 22. This compiles the code and flash it to the device automatically.

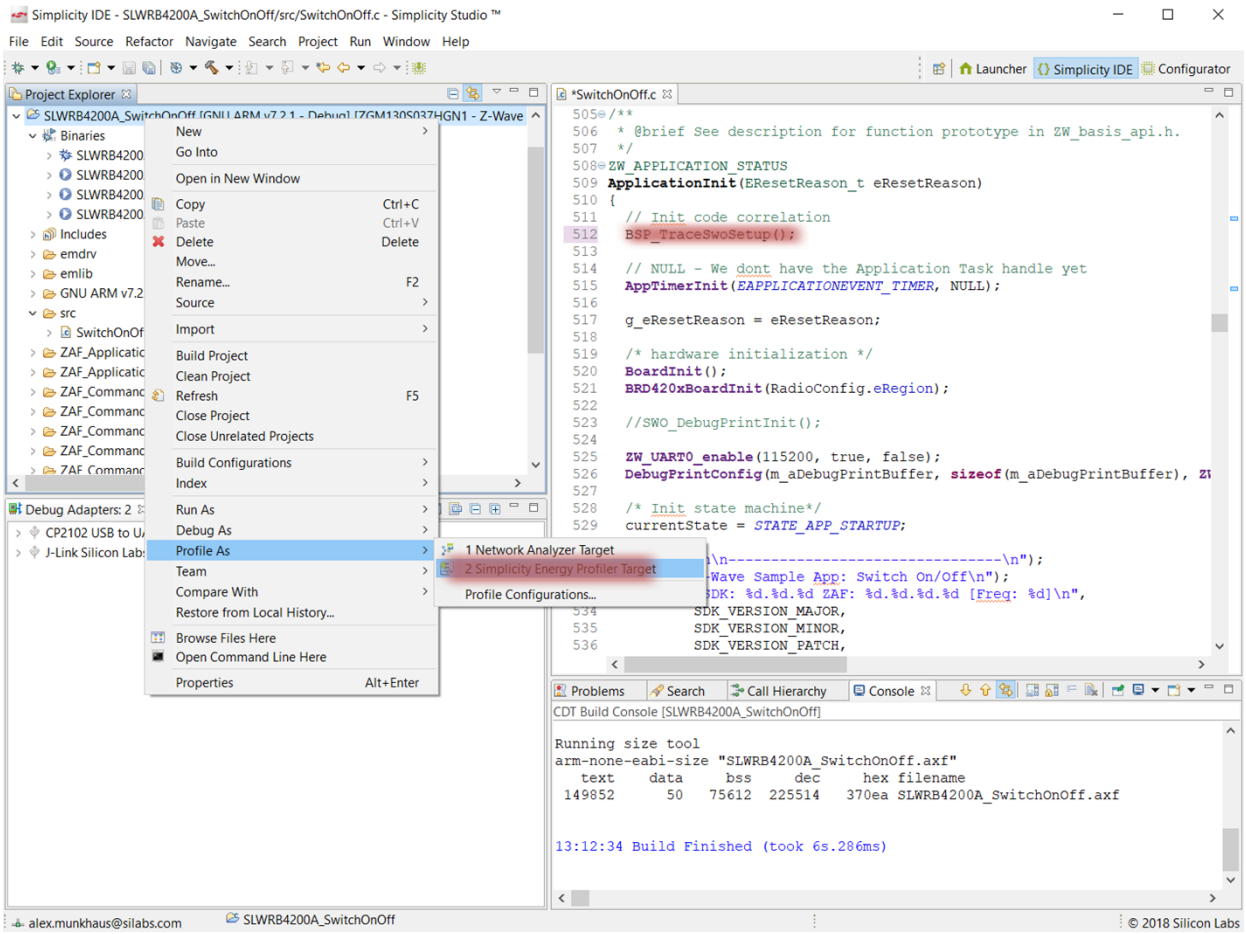


Figure 22: Start Energy Profiler with Code Correlation

The Energy Profiler opens automatically when the code has been flashed. In this case, we have a high energy consumption, and with the Code Correlation feature we can see this is mainly caused by the “ToggleLed” method.

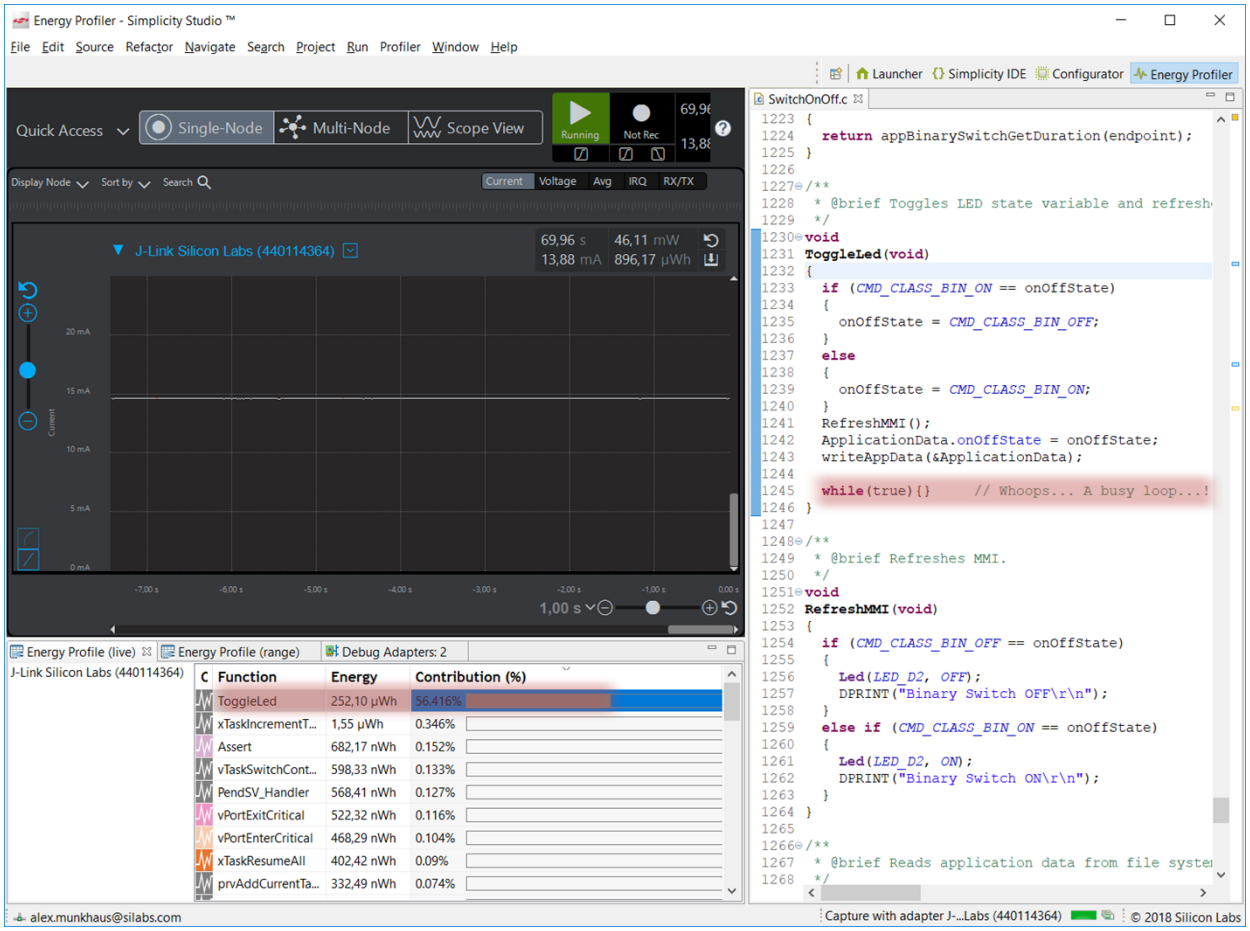


Figure 23: Code Correlation Associates High Energy Consumption with Source Code

9 Next Step 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.

We recommend you investigate the following documents, as a natural next step in your learning curve.

9.1 Next Step for Software Developers

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

The purpose of this document is to describe how to use the sample applications which come as part of the Z-Wave SDK 7.00.

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

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 Step for Hardware Developers

INS14487 - 700 Integration Guide [5]

Hardware Integration Guide for Z-Wave 700.

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

This document describes how to use Silicon Labs development tools to bringup and validate hardware based on the Z-Wave 700 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 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].

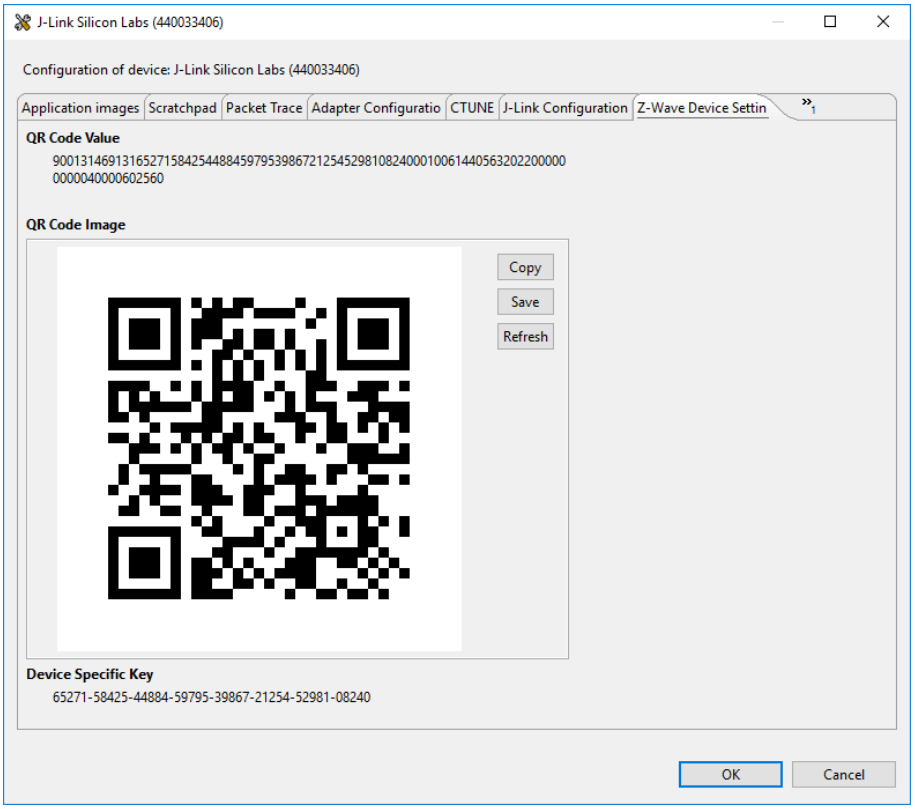
INS14284 - Prepare for Z-Wave Certification Tests [11]

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 by 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' section. See Figure 6, section 5. Right-click and select 'Device Configuration'. From this menu, select '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.

References

[1] Silabs, INS14278, Instruction, How to Use Certified Apps in Z-Wave 700.
[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 SDK7.
[5] Silabs, INS14487, Instruction, 700 Integration Guide.
[6] Silabs, INS14283, Instruction, Bring-up/test HW development.
[7] Silabs, INS14285, Instruction, Manufacture Z-Wave 700 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.