Application Note

Porting Z-Wave Appl. SW from 700 to 800 hardware

Document No.:	APL14836
Version:	1
Description:	The purpose of this document is to give guidelines for the Z Wave application developer, when porting software applications based on Z-Wave Framework from 700 to 800 hardware.
Written By:	MNPALANI;JFR;COLSEN;PSH
Date:	2021-12-1008
Reviewed By:	
Restrictions:	Public

Approved by:

Date	CET	Initials	Name	Justification
2021-12-10	09:40:50	JFR	Jorgen Franck	on behalf of NTJ

REVISION RECORD

Doc. Rev	Date	By	Pages affected	Brief description of changes
1	20210608	MNP	ALL	Initial draft; based on APL14440
2	20211012	PSH & COLSEN	ALL	Updated including a detailed description on how to port a non-UC 700 based Switch On/Off App (7.16.x) to a UC 800 based Switch On/Off App (7.17.x) added

Table of Contents

1 ABBREVIATIONS 1

2 INTRODUCTION 1

2.1 Purpose 1
2.2 Audience and prerequisites 1

3 GECKO SDK 4.00 CHANGES 2

3.1 Z-Wave SDK 7.17.x 2
3.1.1 Breaking API changes 3

4 PORTING AREAS 4

4.1 Non-volatile memory 4
4.2 RTOS 4
4.3 Peripherals 4
4.4 Board files 4
4.5 Using existing command classes 4
4.6 Implementing new command classes 5

5 PORTING AN APPLICATION 6

5.1 Porting from 700 series (7.16.x) to 800 series 6
5.1.1 Create an 800 series project in Simplicity Studio 6
5.1.2 Configure the region (optional) 7
5.1.3 New files in a project 9
5.1.4 Building for release/debug 9
5.1.5 Steps to port Switch On/Off 10
5.1.6 Installing a missing command class 10

REFERENCES 11

INDEX 12

1 ABBREVIATIONS

Abbreviation	Explanation
API	Application Programming Interface
OTA	Over The Air (firmware update)
SDK	Software Development Kit
ZAF	Z-Wave Application Framework

2 INTRODUCTION

2.1 Purpose

The purpose of this document is to provide guidelines to Z-Wave application developers for porting applications based on the Z-Wave Application Framework (ZAF) from 700 series SDKs to 800 series SDKs.

2.2 Audience and prerequisites

The audience of this document is Z-Wave partners and Silicon Labs.

3 GECKO SDK 4.00 CHANGES

The Gecko Software Development Kit version 4.00 introduces a new underlying platform architecture based on components. The Z-Wave SDK 7.17.x now uses this component structure and the structure and build method of Z-Wave applications has therefore changed compared to previous releases of Z-Wave SDK. The new component structure offers several new features in the GSDK:

- Search and filter to find and discover software components that work with the target device
- Automatically pull in all component dependencies and initialization code
- Configurable software components including peripheral units and drivers
- All configuration settings in C header files for usage outside of Simplicity Studio
- Configuration validation to alert developers to errors or issues
- GNU makefiles as a build option

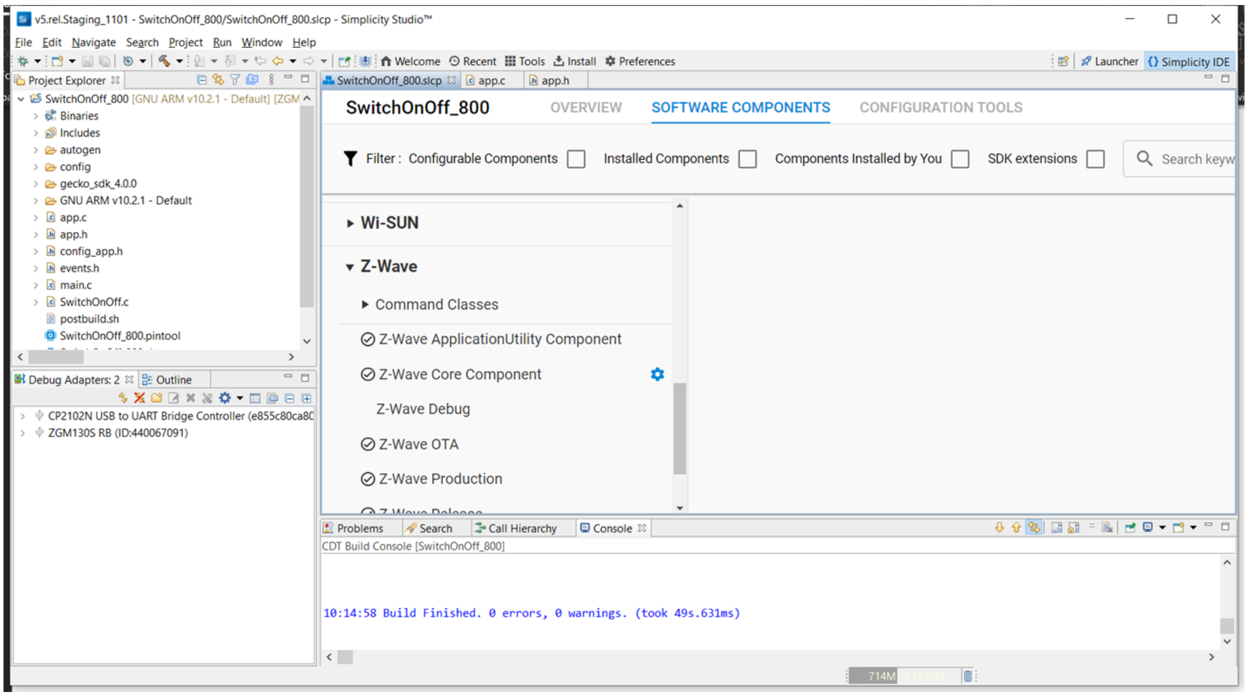
Other changes specific to the Z-Wave Gecko SDK

- main() is now part of the application
- The FreeRTOS configuration is available for application developers
- The region can be configured in Simplicity Studio GUI

3.1 Z-Wave SDK 7.17.x

The Z-Wave SDK 7.17.x is now using the Silicon Labs Configurator (SLC) for project generation and build and the SDK has therefore now been divided into components that can be installed or uninstalled in an existing project in Simplicity Studio. All components now have a description in the z-wave/component directory. The component description file contains a list of all source and header files in the component, all dependencies for the component and all defines used by the component.

The software components can be found in the SOFTWARE COMPONENTS tab in simplicity project view and the Z-Wave software components can be found under the Z-Wave section.



```
10:14:58 Build Finished. 0 errors, 0 warnings. (took 49s.631ms)
```

The components with a checkmark are the components already installed in the current project.

3.1.1 Breaking API changes

All breaking API changes can be found in the release note [2]

4 PORTING AREAS

The following sections describe where to focus the effort when porting from a 700 series application based on ZAF to an 800 series application based on ZAF. In general, all differences described are applied to the certified apps in the 800 series SDK.

It is recommended to take one of the applications from the 800 series SDK and add the business logic of a 700 series application.

4.1 Non-volatile memory

There are no changes in the NVM3 interface between the 700 and the 800. However, the size of the NVM3 instances has increased because of the larger Flash page size in the 800 series.

4.2 RTOS

The Z-Wave 800 series SDK utilizes FreeRTOS like the 700 SDK. There are no changes in the way applications interface with FreeRTOS.

The FreeRTOS is no longer part of the static linked Z-Wave library but is compiled as part of the application build. This means that the FreeRTOS setup is now owned by the application and features can be added by application developers.

4.3 Peripherals

Several peripheral drivers are available (EMDRV and EMLIB) and must be linked to the application before the hardware device in question can be accessed. EMDRV exist on top of the lower level EMLIB.

4.4 Board files

The board.h header file provides a common interface for accessing different boards. This header file provides a common interface for different board specific header files that abstract from the underlying hardware and thereby ease the transition from Silabs development boards to custom made hardware. The board specific header files either begin with “board_” or “extension_board_”.

4.5 Using existing command classes

There are no changes to the existing command classes between the 700 and the 800.

4.6 Implementing new command classes

There is no impact on implementing new command classes between the 700 and the 800.

5 PORTING AN APPLICATION

5.1 Porting from 700 series (7.16.x) to 800 series

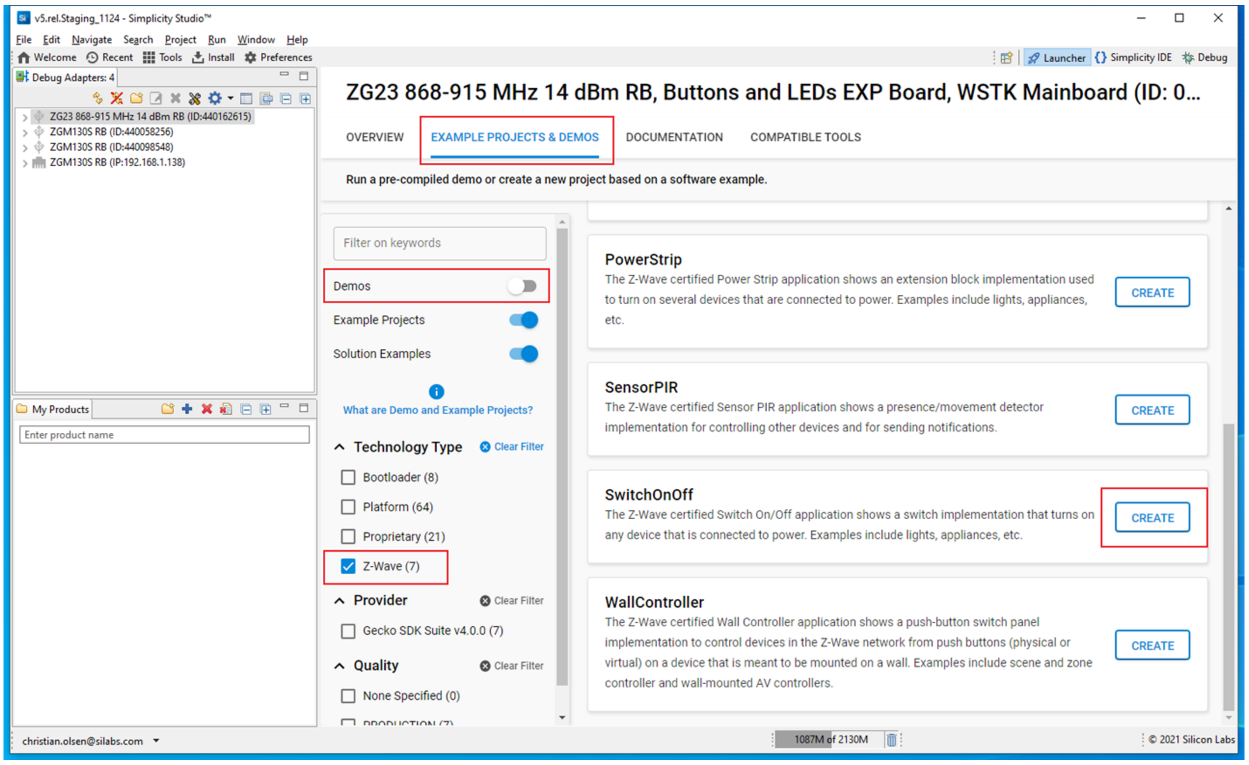
5.1.1 Create an 800 series project in Simplicity Studio

Choose the sample application that you originally used as a starting point for your application and create an 800 series project in Simplicity Studio for that application. If you are using the ZGM230 module then chose the 4205B board and if you are using the ZG23 chip, then use the 4204D board.

Steps to create the application project:

1. Click “EXAMPLE PROJECTS & DEMOS”
2. Click to filter out demos
3. Check Z-Wave to filter out other technologies
4. Click the CREATE button on the desired application,
5. Choose “Link sdk and copy project sources” in the window titled “New Project Wizard”, and
6. Click “FINISH”.

When the project is created it can be built by clicking “Build” (hammer icon).


v5.rel.Staging_1124 - Simplicity Studio™
File Edit Navigate Search Project Run Window Help
Welcome Recent Tools Install Preferences
Launcher Simplicity IDE Debug
Debug Adapters: 4
ZG23 868-915 MHz 14 dBm RB (ID:440162615)
ZGM130S RB (ID:440058256)
ZGM130S RB (ID:440098548)
ZGM130S RB (IP:192.168.1.138)
My Products
Enter product name
ZG23 868-915 MHz 14 dBm RB, Buttons and LEDs EXP Board, WSTK Mainboard (ID: 0...
OVERVIEW
EXAMPLE PROJECTS & DEMOS
DOCUMENTATION
COMPATIBLE TOOLS
Filter on keywords
Demos
Example Projects
Solution Examples
What are Demo and Example Projects?
Technology Type
Clear Filter
Bootloader (8)
Platform (64)
Proprietary (21)
Z-Wave (7)
Provider
Clear Filter
Gecko SDK Suite v4.0.0 (7)
Quality
Clear Filter
None Specified (0)
PRODUCTION (7)
PowerStrip
CREATE
SensorPIR
CREATE
SwitchOnOff
CREATE
WallController
CREATE
christian.olsen@silabs.com
1087M of 2130M
© 2021 Silicon Labs

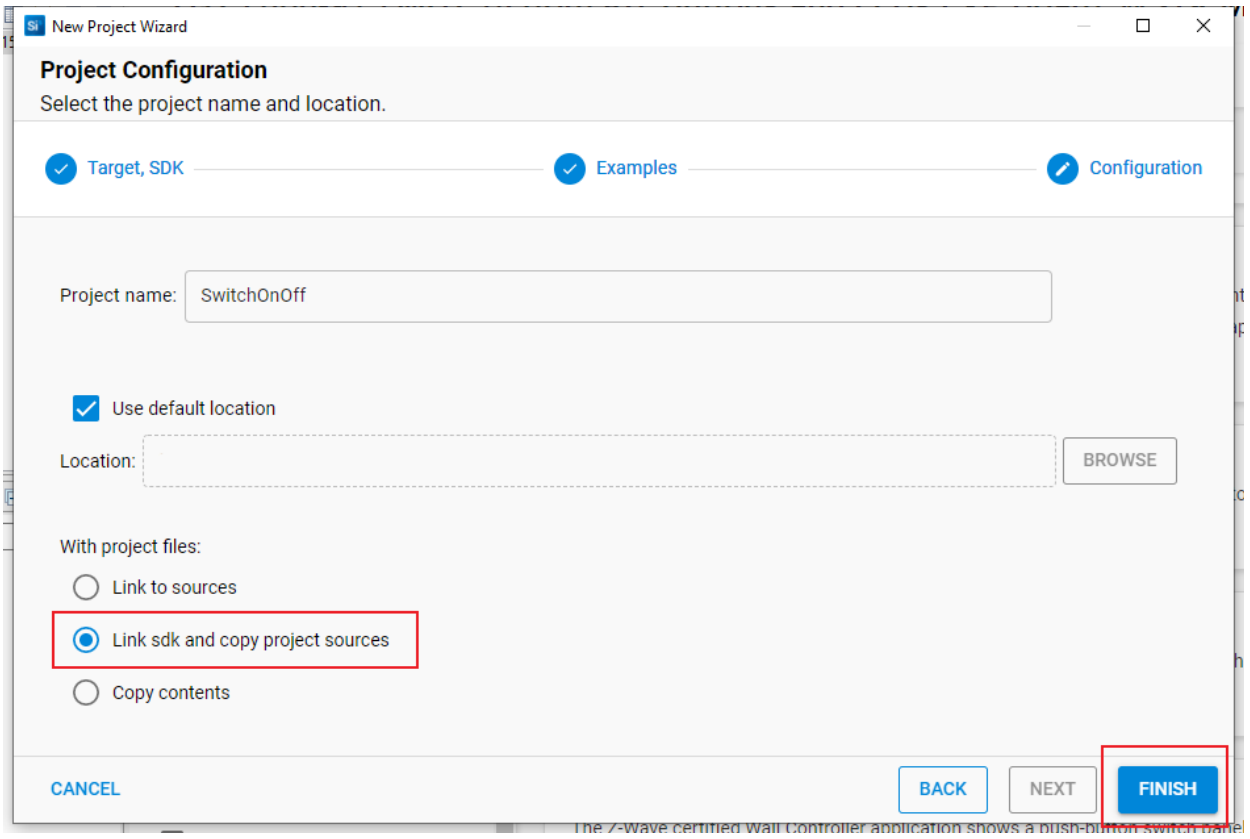
Run a pre-compiled demo or create a new project based on a software example.

PowerStrip: The Z-Wave certified Power Strip application shows an extension block implementation used to turn on several devices that are connected to power. Examples include lights, appliances, etc.

SensorPIR: The Z-Wave certified Sensor PIR application shows a presence/movement detector implementation for controlling other devices and for sending notifications.

SwitchOnOff: 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.

WallController: 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 controller and wall-mounted AV controllers.


New Project Wizard

Project Configuration

Select the project name and location.

Target, SDK — Examples — Configuration

Project name: SwitchOnOff

☑ Use default location

Location:

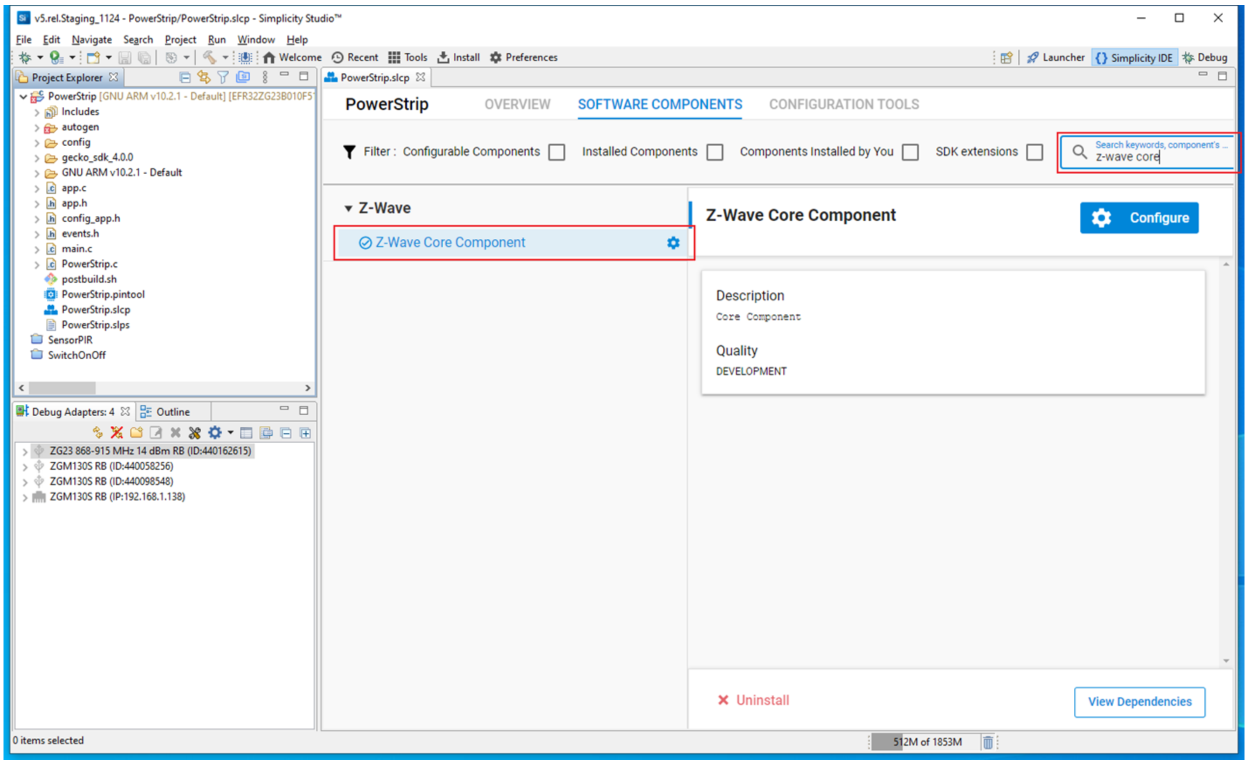
With project files:

- ○ Link to sources
- ◉ Link sdk and copy project sources
- ○ Copy contents

5.1.2 Configure the region (optional)

The project is now created and if another region than the default (EU) is desired, choose it with the following steps:

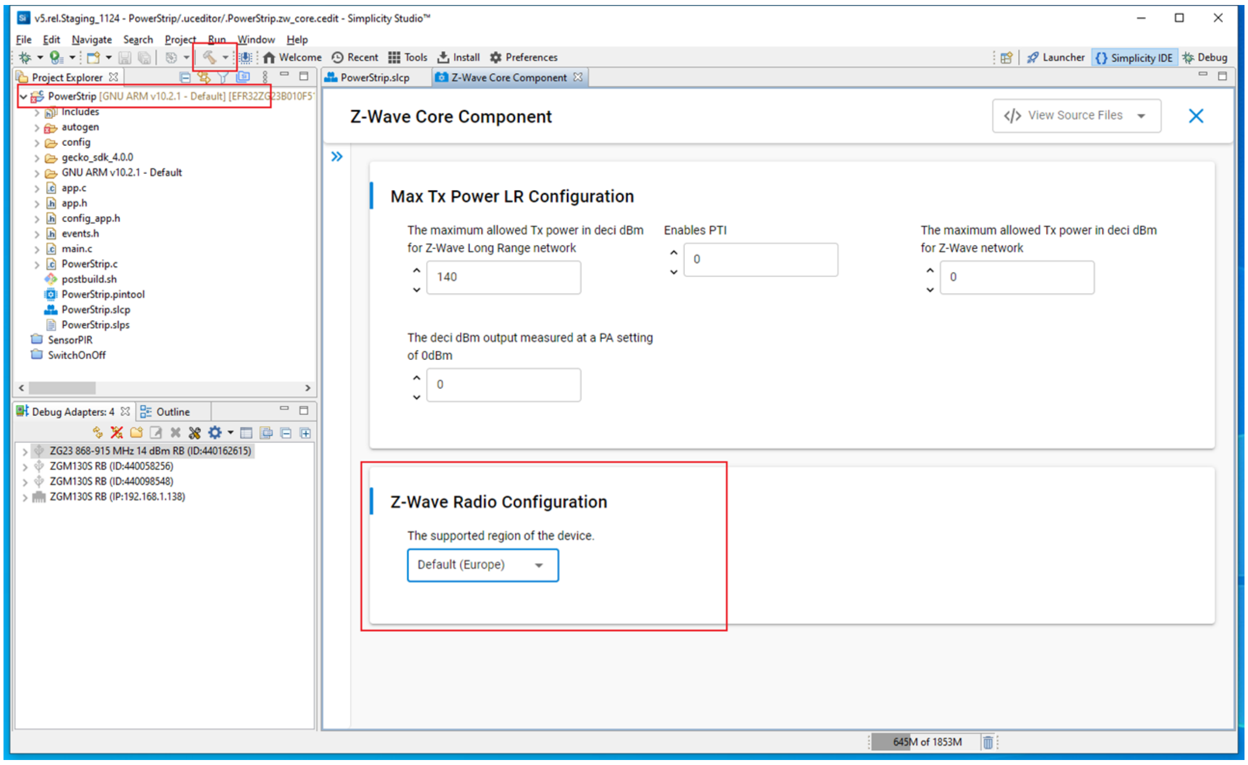
1. Click “SOFTWARE COMPONENTS”,
2. Search for “Z-Wave Core”,
3. Click the cogwheel next to “Z-Wave Core Component”, and
4. Select the desired region in the “Z-Wave Radio Configuration” section.
5. Click the project in “Project Explorer”
6. Click “Build”



Z-Wave Core Component

Description
Core Component

Quality
DEVELOPMENT



Z-Wave Core Component

Max Tx Power LR Configuration

The maximum allowed Tx power in deci dBm for Z-Wave Long Range network: 140

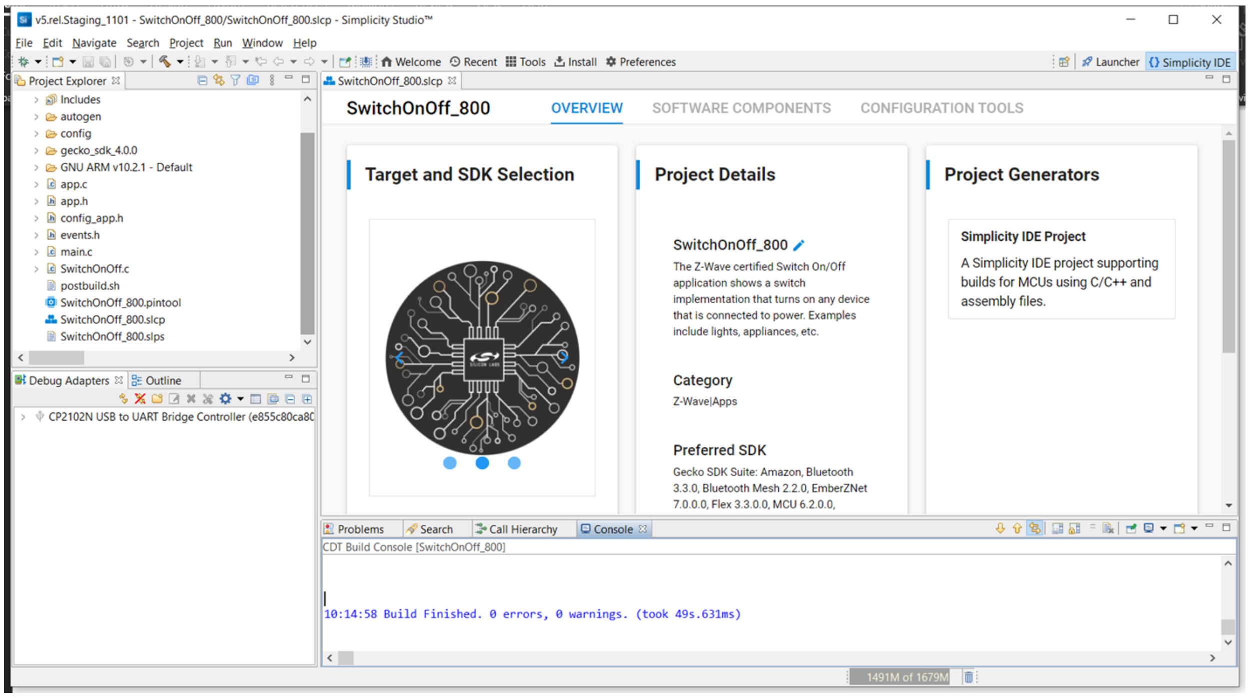
Enables PTI: 0

The maximum allowed Tx power in deci dBm for Z-Wave network: 0

The deci dBm output measured at a PA setting of 0dBm: 0

Z-Wave Radio Configuration

The supported region of the device.: Default (Europe)


v5.rel.Staging_1101 - SwitchOnOff_800/SwitchOnOff_800.slcp - Simplicity Studio™
File Edit Navigate Search Project Run Window Help
Welcome Recent Tools Install Preferences
Launcher Simplicity IDE
Project Explorer
SwitchOnOff_800.slcp
Debug Adapters Outline
Problems Search Call Hierarchy Console
1491M of 1679M

Project Explorer:

- Includes
- autogen
- config
- gecko_sdk_4.0.0
- GNU ARM v10.2.1 - Default
- app.c
- app.h
- config_app.h
- events.h
- main.c
- SwitchOnOff.c
- postbuild.sh
- SwitchOnOff_800.pintool
- SwitchOnOff_800.slcp
- SwitchOnOff_800.slps

Debug Adapters:

- CP2102N USB to UART Bridge Controller (e855c80ca8C

SwitchOnOff_800 — OVERVIEW | SOFTWARE COMPONENTS | CONFIGURATION TOOLS

Target and SDK Selection

Project Details

SwitchOnOff_800

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, Bluetooth 3.3.0, Bluetooth Mesh 2.2.0, EmberZNet 7.0.0.0, Flex 3.3.0.0, MCU 6.2.0.0,

Project Generators

Simplicity IDE Project

A Simplicity IDE project supporting builds for MCUs using C/C++ and assembly files.

CDT Build Console [SwitchOnOff_800]

```
10:14:58 Build Finished. 0 errors, 0 warnings. (took 49s.631ms)
```

5.1.3 New files in a project

main.c

This file contains the main() function for the application. The main() function has been moved out of the Z-Wave library and is now part of the application. The main() function should NOT be modified as the startup of the system depends on the sequence of calls in main().

app.c/app.h

app.c contains the app_init() function for initializing the application. This function should NOT be used as the Z-Wave protocol stack will start the application task.

<app name>.slcp

The extension SLCP is short for Silicon Labs Configurator Project and this file contains the project description and references to the required components.

postbuild.sh

This file is a small script for combining the application and a bootloader into one binary that can be flashed with commander in one programming cycle.

5.1.4 Building for release/debug

Building for release or debug is now a matter of installing the right component. In “SOFTWARE COMPONENTS -> Z-Wave” there exist two components named “Z-Wave Debug” and “Z-Wave Release” respectively. Only one of those components can be installed at a time.

5.1.5 Steps to port Switch On/Off

The following steps describe the required changes to get Switch On/Off building and running after replacing the contents of SwitchOnOff.c from SDK version 7.17.0 with the contents of SwitchOnOff.c from SDK version 7.16.3.

1. Replace inclusion of config_rf.h with zw_config_rf.h
2. Replace APP_FREQ with ZW_REGION and include zw_region_config.h
3. Remove call to CC_AGI_LifeLineGroupSetup() as the function is deprecated. See breaking changes in release note.
4. Include zw_build_no.h as ZAF_BUILD_NO is now defined here.
5. Remove reset reason argument passed to ZAF_setNetworkLearnMode() as it no longer takes a reset reason. See breaking changes in release note.
6. Remove calls to command class handlers delivered by Silicon Labs from Transport_ApplicationCommandHandlerEx(). See breaking changes in release note.
7. Add pPowerDownDebug in ProtocolConfig and set it to either EPOWERDOWNDEBUG_ENABLED if you want the debug interface to be enabled during power down or to EPOWERDOWNDEBUG_DISABLED if it should be disabled.
8. For debug printing there are two steps:
    a. Add the following code block:
    ```
    #ifdef DEBUGPRINT
    #include "sl_iostream.h"
    static void DebugPrinter(const uint8_t * buffer, uint32_t len)
    {
      sl_iostream_write(SL_IOSTREAM_STDOUT, buffer, len);
    }
    #endif // DEBUGPRINT
    ```
    b. Configure the debug printer:
    ```
    #ifdef DEBUGPRINT
      DebugPrintConfig(m_aDebugPrintBuffer,
                       sizeof(m_aDebugPrintBuffer),
                       DebugPrinter);
    #endif // DEBUGPRINT
    ```

5.1.6 Installing a missing command class

In case the newly created project doesn’t include a desired command class, it can be installed from “SOFTWARE COMPONENTS -> Z-Wave -> Command Classes”. If the desired command class is not delivered with the SDK, it must be manually added to the project by clicking “File -> Import”.

Depending on which command class is installed from “SOFTWARE COMPONENTS” it might require certain functions to be invoked from the application, but this is no different than in previous SDK versions.

REFERENCES

[1] Silicon Labs, INS14259, Instruction, Z-Wave Plus V2 Application Framework SDK7
[2] Silicon Labs, SRN14862, Z-Wave and Z-Wave Long Range 700/800

INDEX

No index entries found.

IoT Portfolio
www.silabs.com/products

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

SILICON LABS

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

www.silabs.com