

<docnum>: 3A Compile Switch On/Off and Enable Debug

This hands-on exercise will demonstrate how to use the Z-Wave Embedded SDK to enable serial debugging and compile a sample application using Simplicity Studio.

This exercise is part of the series 'Z-Wave 1-Day Course'.

- 1 Include using SmartStart
- 2 Decrypt Z-Wave RF Frames using the Zniffer
- 3A Compile Switch On/Off and Enable Debug
- 3B Modify Switch On/Off
- 4 Understand FLiRS devices

KEY FEATURES

- Compile a Z-Wave Sample Application
- Enable serial debug
- Running and using a Z-Wave Sample Application
- Introduction to Z-Wave Command Classes.

1 Introduction

This is the first exercise in the 'Z-Wave 1-Day Course' series, where you will use Simplicity Studio to compile a Z-Wave sample application.

This exercise will guide you in how to set the appropriate frequency for your region and enable serial debug to facility serial output messages useful when testing a sample application.

1.1 Hardware Requirements

- 1 WSTK Main Development Board
- 1 Z-Wave Radio Development Board: ZGM130S SiP Module
- 1 UZB Controller
- 1 USB Zniffer

1.2 Software Requirements

- Simplicity Studio v4
- Z-Wave 7 SDK
- Z-Wave PC Controller
- Z-Wave Zniffer



Figure 1: Main development board with Z-Wave SiP module

1.3 Pre-requisites

Previous Hands-On exercises has covered how to use the PC Controller and Zniffer application to build a Z-Wave network and capturing the RF communication for development purpose.

This exercise assumes you are familiar with these tools.

2 Compile your first Z-Wave Sample Application

2.1 Open sample project

- Connect your Z-Wave hardware to the USB port of the computer and it should show up in the 'Debug Adapters' section in Simplicity Studio.
- Click once on the "J-Link Silicon Labs" which instructs to studio the show relevant information about Z-Wave 700.
- Under "Software Example" click on the Switch On/Off sample application.



Figure 2: Open a Z-Wave sample application: Switch On/Off

2.2 Set the frequency

The sample app will not compile just yet. You need to set the frequency that matches the region you intend to use the Z-Wave Product in.

• In the main source file "SwitchOnOff.c", locate the variable APP FREQ:

```
static const SRadioConfig_t RadioConfig =
{
    .iListenBeforeTalkThreshold = ELISTENBEFORETALKTRESHOLD_DEFAULT,
    .iTxPowerLevelMax = APP_MAX_TX_POWER,
    .iTxPowerLevelAdjust = APP_MEASURED_0DBM_TX_POWER,
    .eRegion = APP_FREQ
};
```



Refer to Table 1 for a complete list of supported frequencies by the SDK.

Hint Navigate to Silicon Labs website, to see which countries has been approved for the Z-Wave RF.

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

Table 1: Overview of a possible frequencies

In this guide we will be using the European frequency, thus we enter 'REGION_EU'.

```
static const SRadioConfig_t RadioConfig =
{
    .iListenBeforeTalkThreshold = ELISTENBEFORETALKTRESHOLD_DEFAULT,
    .iTxPowerLevelMax = APP_MAX_TX_POWER,
    .iTxPowerLevelAdjust = APP_MEASURED_0DBM_TX_POWER,
    .eRegion = REGION_EU
};
```

```
Figure 4: APP_FREQ is set to REGION_EU
```

2.3 Enable Serial Debug

The serial debug interface is useful for printing various information without affecting the flow and timing in the application, such as using a debugger.

- In the main source file "SwitchOnOff.c", locate the variable DEBUGPRINT in the include section.
- Uncomment the define to enable serial debug.

2.4 Compile your first Z-Wave Application!

You have now configured the Z-Wave sample application, and you are ready to compile.

- Click on the 'Build' button to start building the project.
- When the build finishes after a short while, a new folder named '*Binaries*' are showed in the Project Explore. Expand the folder and right click on the *.hex file to select '*Flash to Device..*'.
- Select the connected hardware in the pop-up window. The 'Flash Programmer' is now prefilled with all needed data, and you are ready to click on 'Program'. See Figure 6.
- Click 'Program'.

			_	~
Hash Programmer				X
Change Device				
Device Board Name: Wireless Starter Kit Mainb Board Name: ZGM130S Radio Board Board Name: Buttons and LEDs EXP Boo MCU Name: ZGM130S037HGN1	poard ard			
Name: J-Link Silicon Labs (440114364)				
,,				
Flash Part				
File Type her hin Race addres				
	0.0			
File				
.00A_SwitchOnOff\GNU ARM v7.2.1 - I	Debug\SLWRB4200A_SwitchO	nOff.hex ~	Browse	e
Advanced Settings				
rterenced bettingsm		Erase	Program	n
Flash Erase/Write Protection				
Select flash range	∽ 0x0 ^ →	✓ 0x8000	0	^
O Select default sections	Lock Main Flash	🗹 Loc	k User Pa	ge
	Protect	Remove P	rotection	n
Debug Lock Tools				
The unlock function only works using Si Unlocking the chip will erase all data or	licon Labs EFM32 and EFR32 b n flash and SRAM.	ooards.		
	Unlock Debug Access	S Lock De	bug Acce	ess
(?)			Close	
			_	

Figure 6: 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.

3 The Serial Debugger

You have now flashed the Z-Wave Sample Application on to your device. In this section we will be including it into a Z-Wave Network and learn how to use the Serial Debugger.

3.1 Include into network

In previous exercises we have already included the device into a secure Z-Wave network using SmartStart. Refer to exercise "Include using SmartStart" for instructions.

Hint The internal file system 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 e.g. the frequency at which the module operates or the DSK, you need to "Erase" the chip before the new frequency will be written to the internal NVM.

As such, your device is already included in the network.

3.2 Turn the switch On and Off

Test the functionality by verifying you can turn ON and OFF the LED0.

- Test the functionality using the "Basic Set ON" and "Basic Set OFF" in the PC Controller. LED0 should be turning ON and OFF.
- LED0 can also be turned ON and OFF using BTN0 on the hardware.

We have now verified that the device is correct included in the network and that the functionality is still working as expected.

3.3 Using the serial debugger

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

- When the device has been flashed, right click on the adapter, click Launch Console.
- In the Console view, select Serial 1
- Click in the input field and press on enter.
- The small connection icon to the left of the input field should change from 🧏 to 🗐 to show you are connected.

The connection is now open, so let's try to see some data.

- Press the RESET button on the board to see the welcome message.
- Then press on BTN0 to change the state of the LED0
- Use the PC Controller to send a Basic Set ON

Can you explain the difference in the serial debug log depending on when pressing BTN0 and when clicking on Basic Set ON?

🖛 Simplicity IDE - Adapter: J-Link Silicon Labs (440114364) - Simplicity Studio ™ Х File Edit Navigate Search Project Run Window Help 😰 🏫 Launcher 🚯 Simplicity IDE 🔸 Energy Profiler 🖃 🔄 🤝 🗖 🔲 🖻 *SwitchOnOff.c - -Project Explorer 🛛 🞯 J-Link Silicon Labs (440114364) 🛛 > 🖉 SLWRB4200A_DoorLockKeyPad [GNU ARM v7.2.1 - Debug] [Z 🔺 No translation Line terminator: CR-LF (DOS, OS/2, MS Windows) V Serial 1, 0 byte > 🖉 SLWRB4200A_PowerStrip [GNU ARM v7.2.1 - Debug] [ZGM13(差 Serial 0 🚄 Serial 1 🚔 Admin 🚔 Debug SLWRB4200A_SwitchOnOff [GNU ARM v7.2.1 - Debug] [ZGM1] V 🕊 Binaries > 🕸 SLWRB4200A_SwitchOnOff.axf - [arm/le] Z-Wave Sample App: Switch On/Off SDK: 7.0.1 ZAF: 10.0.0.43707 [Freq: 0] > SLWRB4200A SwitchOnOff.bin - [unknown/le] > SLWRB4200A_SwitchOnOff.hex - [unknown/le] Toggle switch on/off Toggle learn mode Hold 5 sec: Reset Learn mode + identify Switch status on/off BTN0: BTN1: > SLWRB4200A_SwitchOnOff.s37 - [unknown/le] > 🗊 Includes LED1: LED0: > 🗁 emdrv > 🗁 emlib ApplicationInit eResetReason = 0 SwitchOnOff Event processor Started > 🗁 GNU ARM v7.2.1 - Debug SwitchOmOff Event processor Sta Event: 113 AppStateManager St: 0, Ev: 113 Init App NVM3 File System Event: 112 AppStateManager St: 0, Ev: 112 State changed from 0 to 1 Event: 114 Binary Switch ON Incoming Status msg Incoming Rx msg 🗸 🗁 src > 🖻 SwitchOnOff.c > > > ZAF ApplicationUtilities 7AF ApplicationIItilities TrueStatusEngine < > 📲 Debug Adapters: 2 🛛 🔚 Outline TAppH 32 appBinarySwitchSet 255 Binary Switch ON Incoming Rx msg 🎭 💥 😂 🖉 🗙 💥 🗢 🗖 🖨 🕀 CP2102N USB to UART Bridge Controller (ID:0) (92f6fb1aa44fe8) Connect TAppH 32 appBinarySwitchSet 0 Binary Switch OFF Incoming Rx msg > 🔣 Buttons an Disconnect > Wireless St Start capture > 8 ZGM130S Start capture with options... 2 Stop capture Redo last upload -\$ -} 🔁 📑 🔐 = 🖳 🚽 = 🔿 🕶 🖹 Problems 🛛 🔗 Search Call Hierarchy 🗳 Console 🖾 Upload application... CDT Build Console [SLWRB4200A_SwitchOnOff] nex IIIename Upload adapter firmware... ^ 3cfb2 SLWRB4200A SwitchOnOff.axf 173388 76340 249778 50 Rename Make a sniffer 08:42:21 Build Finished (took 11s.633ms) I Sniffer Configurator... Launch Console... ~ < Device configuration... © 2018 Silicon Labs 📥 alex.munkhaus@ 🔒 Force Unlock..



3.4 Using Command Class Binary Switch

In previous exercises we covered how command class Basic is mapped to command class Binary Switch, which is being specified by the Z-Wave Device Type specification. Refer to exercise "2 Decrypt Z-Wave RF Frames using the Zniffer" for more details, or the Z-Wave specification <u>SDS14224 Z-Wave Plus v2 Device Type Specification</u>.

4.5.9.3 Basic Command Class Requirements

The Basic Command Class MUST be mapped according to Table 17.

Table 17,	Binary	Switch	Device	Туре	Basic	mapping
-----------	--------	--------	--------	------	-------	---------

Basic Command	Mapped Command	
Basic Set (Value)	Binary Switch Set (Value)	
Basic Report (Current Value, Duration)	Binary Switch Report (Value, Duration).	

Figure 8: Basic mapping for device type Binary Switch

In this section, we will try to turn the LED ON and OFF using the Binary Switch command class, and compare the Serial output log when sending Basic command class commands.

- In the PC Controller, double click on "25 SWITCH_BINARY" under secure Command Classes in the lower left corner.
- This opens the "Command Classes" view in the PC Controller and select the Switch Binary Command class.
- Set the Command to "0x01 SWITCH_BINARY_SET"
- Set the "Target Value" to either "00-OFF_DISABLE" or "FF-ON_ENABLE"
- Click "Send".



Figure 9: Command Class View in PC Controller

Can you explain the difference in the serial debug log depending on when sending a Basic Set and a Binary Switch Set?

	*SwitchOnOff.c	🔗 J-Link Silicon Labs (440114	364)	23	
9	No translation		\sim	Lin	
e	🚔 Serial 0 🚔 Seria	l 1 🚔 Admin 🚔 Debug			
-	Incoming Rx msg				
]	TAppH 32 appBinarySwitchSe Binary Switch ON Incoming Rx msg	t 255			
i I I	TAppH 32 appBinarySwitchSe Binary Switch OFF Incoming Rx msg	t 0			
]	TAppH 37 appBinarySwitchSe Binary Switch ON Incoming Rx msg	t 255			
, []	TAppH 37 appBinarySwitchSe Binary Switch OFF	t 0			
Figure 10: Serial Debug Log with both Basic Set and Binary Switch Set.					

This concludes the tutorial in how to compile and debug a Z-Wave Sample Application.