

AN1133: Dynamic Multiprotocol Development with *Bluetooth[®]* and Zigbee EmberZNet SDK 6.x and Lower

This application note provides details on developing Dynamic Multiprotocol applications using Bluetooth and Zigbee. It describes how to configure applications in Simplicity Studio using the Zigbee EmberZNet SDK. It then provides a detailed walkthrough on how the underlying code functions. For details on Dynamic Multiprotocol Application development that apply to all protocol combinations see *UG305: Dynamic Multiprotocol User's Guide*.

This document applies to Zigbee EmberZNet SDK 6.10.x and lower. Zigbee EmberZNet SDK 7 contains significant changes compared to earlier SDKs. Many of these changes are due to an underlying framework redesign that results in an improved developer experience within Simplicity Studio 5. Projects are now built on a component architecture instead of AppBuilder. Simplicity Studio 5 includes project configuration tools that provide an enhanced level of software component discoverability, configurability, and dependency management. See AN1322: Dynamic Multiprotocol Development with Bluetooth and Zigbee EmberZNet 7.0 and Higher if you are using or planning to use that version.

KEY POINTS

- Generating and loading dynamic multiprotocol example applications.
- Adding dynamic multiprotocol functionality to an existing project.
- Details on the application User Interface.
- How the Zigbee example applications function.
- How the Bluetooth application functions.

1 Introduction

The example applications referenced here can be controlled either from a protocol-specific switch application or from a Bluetooth-enabled smartphone app. This application note provides details on how these examples are designed and implemented. It also describes how to generate, compile, and load example application code, and how to add dynamic multiprotocol functionality to an existing Zigbee project. The application note is intended to be used when developing your own Zigbee/Bluetooth dynamic multiprotocol implementations.

Note: The Zigbee dynamic multiprotocol solution is currently only supported for SoC architectures. Support for NCP architectures is not yet available. Please contact Silicon Labs Sales for more information on our multiprotocol software roadmap.

1.1 Resources

- UG305: Dynamic Multiprotocol User's Guide provides details on:
 - Dynamic Multiprotocol Architecture
 - Radio Scheduler operation (with examples)
 - Task Priority management
- AN1135: Using Third Generation Non-Volatile Memory (NVM3) Data Storage explains how NVM3 can be used as non-volatile data storage in Dynamic Multiprotocol applications with Zigbee and Bluetooth.
- **Note:** EmberZNet SDK 6.8.0.0 was released as part of Gecko SDK Suite 3.0.0.0 (GSDK v3.x) and is used with Bluetooth SDK v3.x and Simplicity Studio 5. EmberZNet SDK 6.7.x continues to be used with Bluetooth 2.13.x and Simplicity Studio 4. Because of changes to the Bluetooth SDK v3.x, a few instructions and examples in this document vary based on version. Both variants are included and are clearly noted in the text.

1.2 Development Environment Requirements

EmberZNet 6.7.x

- Simplicity Studio 4
- EmberZNet SDK version 6.4.0 or higher
- Bluetooth SDK version 2.10.0 or higher
- Micrium OS-5 kernel version 5.3. or higher (installed automatically with EmberZNet SDK in Simplicity Studio 4)
- An EFR32 chip with at least 512 kB of flash (required to run all the necessary software components)
- IAR Embedded Workbench for ARM (IAR-EWARM) version compatible with your SDK (see the release notes for version details).

EmberZNet 6.8.x and higher

- Simplicity Studio 5
- EmberZNet SDK version 6.8.0 or higher
- Bluetooth SDK version 3.0.0 or higher
- Micrium OS kernel version 6.0.0 or higher (installed automatically with EmberZNet SDK in Simplicity Studio)
- An EFR32 chip with at least 512 kB of flash (required to run all the necessary software components)
- IAR Embedded Workbench for ARM (IAR-EWARM) version compatible with your SDK (see the release notes for version details).

2 Working with the Zigbee/Bluetooth Examples

This section describes

- How to build and flash the dynamic multiprotocol applications supplied with the EmberZNet SDK.
- How to configure a Zigbee project into a dynamic multiprotocol project.

2.1 Application Generation

To work with Zigbee/Bluetooth dynamic multiprotocol applications you must install both the EmberZNet SDK and the Bluetooth SDK. The Micrium kernel is installed along with the EmberZNet SDK. IAR Embedded Workbench for ARM (IAR-EWARM) 8.30 must be installed and used as your compiler. See QSG106: Getting Started with EmberZNet PRO for information on installing the SDKs and IAR-EWARM.

Dynamic multiprotocol applications are generated, built, and uploaded in the same way as other applications. If you are not familiar with these procedures, see *QSG106: Getting Started with EmberZNet PRO* for details. The dynamic multiprotocol applications included with the EmberZNet SDK are:

- DynamicMultiprotocolLight is an application designed to demonstrate a DMP device with Zigbee 3.0 coordinator capabilities.
- DynamicMultiprotocolLightSed is an application designed to demonstrate a DMP device with SED capabilities.
- DynamicMultiprotocolSwitch is a Zigbee-only application designed to work with the two Zigbee/Bluetooth applications.

The following summary procedure uses the DynamicMultiprotocolLight example application.

- 1. In Simplicity Studio, start a new project selecting the DynamicMultiprotocolLight example.
- 2. If your project General tab shows GNU-ARM as a compiler, change to IAR EWARM.

🕸 • 🚱 • [월 • 🔛 🕼] 🕲 • 🐔 • [월 • 월 • 1	≥ ⇔ • ⊕ • [<u>∎</u>	8	A Launcher () Simplicity IDI	E is Editin
🖕 Project Explorer 🕮 📄 🤹 🗢 🗖	🚜 *DynamicMultiprotocolLightSoc.isc 🕄		- (5
 DynamicMultiprotocolLightSoc [IAR ARM - Default] [E S Includes 	Silicon Labs Zigbee, version:6.6.0.0		Generate ec Preview	8
DynamicMultiprotocolLightSoc_callbacks.c brd4162a_efr32mo12p332f1024pl125_hwconf	🚜 General 🔥 ZCL Clusters 🗸 Zigbee Stack 🚜 Printing and CLI 🔮 HAL 🚸 Plugins	Select architecture	-	o x
2 DynamicMultiprotocolLightSoc.isc	Application configuration	Boards:		
	Generation directory: Relative to ISC file C:\Users\CAOWENS\SimplicityStudio\v4_rel.Staging_2954	Search		~
	Select architecture for this application: Select architecture for this application: Search: EFR32M0122343F4: 10 dBm (JRD4162A Rev A01) Part: EFR32M012P332F42A04125 Coolchain: IAR ARM v8.30.1.114	EFR32MG12 2.4GHz 10 dBm (BRD4162A Rev A	.01) ×	
	Edit Architecture	Part		
	Device name: DynamicMultiprotocolLightSoc	EFR32MG12P332F1024GL125		
< >>		Testerio		
🚯 Debug Adapters: 2 🗱 🔡 Outline 👘 🗇	▼ Information Configuration	IAR ARM (v8.30.1.114)		
S X 22 2 X X 2 - □ 2 2 0	Description Sample application demonstration a light application using dynamic	(No toolchain) (v0.0.0) GNU ARM v7.2.1		105
>	multiprotocol (ZigBee + BLE) and NVM3 for persistent storage	4AR ARM (v8.30.1.114)		
	The Dynamic Multiprotocol Demo Light application demonstrate the dynamic		OK	Cancel
	multiprotocol feature from Silabs. In particular, these applications demonstrate the ZigBee and BLE stacks running concurrently as MicriumOS tasks.			-
	Dephlane 22 Catter Secondary		~	

- 3. Click Generate to generate project files.
- 4. Click **Build** (hammer icon) to build the application image.

5. Note the board and part number for your device and the directory for generated files.

Simplicity IDE - DynamicMultiprotocolLightSoc/	DynamicMultiprotocolLightSoc.is	c - Simplicity Studio ™			-	
<u>File Edit Navigate Search Project Run Wi</u>	ndow <u>H</u> elp 🚓 App Builder					
* • 8 • 1 🖸 • 🗟 🐚 🛞 • 🗞 • 1 🖢	• 🖗 • % 🗘 • 🖓 • 📳			E	🖇 🕇 Launcher 🛛 🖁 Si	mplicity IDE
Project Explorer 🛛 📄 🔄 🔽 🗖	🚴 DynamicMultiprotocolLigh	Soc.isc 🛛				
> 🌮 DynamicMultiprotocolLightSoc [IAR ARM - [Silicon Labs Zigbee, version 	on:6.6.0.0			Generate	« Preview
	🙈 General 🔥 ZCL Clu	sters 👌 🚓 Zigbee Stack	👶 Printing and CLI 🔘 H	AL 🔷 Plugins 🛱 Callbacks	🖂 Includes 🎇	
	 Application configuration 	n				
	ieneration directory: Rel	ative to ISC file C:\Use	rs\CAOWENS\SimplicityStudio\	v4_rel.Staging_2968\DynamicMult	tiprotocolLightSoc	
	Select architecture for t	his application:				
	Board: EFR32MG12 2.4	GHz 19 dBm (BRD4161A F	Rev A01)			
	Part: EFR32MG12P432	1024GL125				
	📩 Toolchain: IAR ARM 🕫	3.30.1.114				
			Edit Architecture			
	Device name: DynamicM	ultiprotocolLightSoc				
🗱 Debug Adapters 🖾 🔚 Outline	 Information Configuration 	on				
	Description					
 ✓	Sample application demo multiprotocol (ZigBee + E	nstrating a light applicati LE) and NVM3 for persist	on using dynamic ent storage			^
EFR32MG12P432F1024GL125	The Dynamic Multiprotoc	ol Demo Light applicatio	n demonstrate the dynamic			
S E Wireless Starter Kit Wainboard (BKD4001.	multiprotocol feature from demonstrate the ZigBee a	n Silabs. In particular, the nd BLE stacks running co	se applications neurrently as MicriumOS tasks.			
						· · · ·
	😰 Problems 🛛 🔗 Search	Call Hierarchy	📮 Console 🛛	ት 🗘 🛃 🖬 🗗	= 🖳 🛃 📮 🕇	3 - 🗆 🗖
	CDT Build Console [DynamicM	ultiprotocolLightSoc]				
						^
	11:41:01 Build Finish	ned (took 1m:29s.	564ms)			
						~
	<					>
👍 cjowens325@earthlink.net					© 201	18 Silicon Labs

- 6. Right-click the target J-Link under Devices, and select Upload Application.
- 7. Browse to <folder on General tab>\IAR ARM <qualifier>\<project name> and select the .gbl file.
- 8. Silicon Labs strongly recommends that, if you have not already loaded a bootloader onto your device, you do so now. Check **Erase chip before uploading image**. Check **Bootloader image**, then browse to the following folder:

C:\SiliconLabs\SimplicityStudio_v5\developer\sdks\gecko_sdk_suite\<version>\platform\boot-loader\sample-apps\bootloader-storage-spiflash-single\

Open the folder that corresponds to your board and part number and select the .s37 file, for example:

\efr32mg12p432f1024g1125-brd4161a\bootloader-storage-spiflash-single-combined.s37

9. When both images are selected, the dialog should resemble the following figure. Click OK.

🗢 Binary image upload	ł								×
Application ima	age upload								
Select an image from th Please make sure the se	e list or browse for a different o lected image matches the hards	ne. vare.						Show all i	mages
Name		Chip	Board	Image Type	Location	Description			^
RFEval application		EM250	Wirele	Custom a	<internal></internal>	RFEval application firmware image			
HA Light		EM250	Wirele	Custom a	<internal></internal>	HA Light application			
HA Light		EM351	Wirele	Custom a	<internal></internal>	HA Light application			
HA Light		EM357	Wirele	Custom a	<internal></internal>	HA Light application			
HA Light		EM3581	Wirele	Custom a	<internal></internal>	HA Light application			
HA Light		EM3582	Wirele	Custom a	<internal></internal>	HA Light application			
HA Light		EM3585	Wirele	Custom a	<internal></internal>	HA Light application			
HA Light		EM3586	Wirele	Custom a	<internal></internal>	HA Light application			
HA Light		EM3587	Wirele	Custom a	<internal></internal>	HA Light application			
HA Light		EM3588	Wirele	Custom a	<internal></internal>	HA Light application			
HA Switch		EM250	Wirele	Custom a	<internal></internal>	HA Switch application			
HA Switch		EM351	Wirele	Custom a	<internal></internal>	HA Switch application			
HA Switch		EM357	Wirele	Custom a	<internal></internal>	HA Switch application			~
Application image path Upload options	C:\Users\CAOWENS\Simplic	ityStudio\v4_r	el.Staging_2	968\DynamicM	ultiprotocolLightSo	c\IAR ARM - Default\DynamicMultiprotocolLig	jhtSoc.gbl	ad 27	
Bootloader image:	strorm\bootloader\sample-a	pps\bootioadi	er-storage-s	pinasn-single\e	fr52mg12p452f1024	gi125-brd4161a\bootloader-storage-spinasn-s	ingle-combin	led.so/ V	
Erase chip before up	loading image								
After uploading:	◉ Run ◯ Halt								
Flash:	● Internal ○ External SPI								
							OK	Canc	el

 Application load success indicators are code-dependent. With the DynamicMultiprotocolLight example, the LCD should display the following before changing over to the light bulb display:



Whether the application is a full function or a sleepy end device is determined by the Device Type on the ZNet tab.

2.2 Converting a Zigbee Application to a Zigbee/Bluetooth LE Dynamic Multiprotocol Application

This section describes the configuration changes required to convert a working Zigbee application into a Zigbee/Bluetooth LE Dynamic Multiprotocol application. The instructions assume you have started with a non-DMP Zigbee sample application or your own Zigbee project, and that the application is working correctly.

Requirements:

- Zigbee application set up to build with IAR ARM v8.30.1 (for these instructions we use Z3 Light)
- EFR32MG12 or other EFR32 with sufficient memory (for these instructions we assume BRD4161 (EFR32MG12P432F1024GL125))
- **Note:** The Dynamic Multiprotocol sample applications supplied with Simplicity Studio are already correctly defined and do not require modification before project generation unless performing an OTA update. There is a potential conflict with the DMP sample app LCD screen and the external flash. If you need to perform OTA updates, check the **Dynamic Multiprotocol UI Demo Code Stub** plugin as described in the following section.

2.2.1 Generate and Build the Zigbee Application

The purpose of this procedure is to verify that the base application had loaded and is working correctly, and that output is printing to the console. This example uses the Z3Light sample application in the EmberZNet SDK It begins with the default settings, so that the configuration changes are clear. Remember to select IAR as the toolchain.

Generate and build the project, load it to the board and check the Serial 1 output to make sure it's up and running.

Z3LightSoc2DMP_6610_brd4161.isc	🦃 No name (10.4.180.249) 🙁
No translation	 Line terminator:
🔁 Serial 0 🚔 Serial 1 🚊 Admin 🛔	C Debug
Join network start: 0x00	
Starting scan on channel	20
Starting scan on channel	24
Starting scan on channel	25
Starting scan on channel	11
Starting scan on channel	14
Starting scan on channel	15
NWK Steering State: Scan	Secondary Channels
Starting scan on channel	16
Starting scan on channel	17
Starting scan on channel	18
Starting scan on channel	19
Starting scan on channel	20
Starting scan on channel	21
Starting scan on channel	22
Starting scan on channel	23
Starting scan on channel	24
Starting scan on channel	25
Starting scan on channel	26
Starting scan on channel	11
Starting scan on channel	12
Starting scan on channel	13
Starting scan on channel	14
Starting scan on channel	15
NWK Steering State: Scan	Primary Channels an
Starting scan on channel	19
Starting scan on channel	20

2.2.2 Reconfigure the Project

The search bar at the top of the Plugins and other tabs is helpful when modifying the configuration. For plugins, the description explains its utility for DMP. The following figure illustrates finding the RTOS plugin and its description.

	» ₂
igin configuration e this section to select or unselect the plugins that you want to use	in your application
Q RTOS	Plugin: ↔ RTOS Common Quality: ✓ Production Ready Description: This plugin provides OS support. If this plugin is enabled, the stack will be running within an OS as a an OS task. The application can also define up to 3 custom OS tasks.
	Options: Reset to defaults ✓ CPU usage tracking ✓ Poll CLI □ Enable EM2

Note: Beginning with SDK 6.9.0, the RTOS configurations options moved from the Micrium plugin to the RTOS common plugin and the Zigbee Task call stack size is treated as words instead of bytes.

- 1. On the Plugins tab, check the following:
- RTOS

In SDKs 6.8.x or lower, check Micrium RTOS.

Plugin configuration Use this section to select or unselect the plugins that you want to use in your application		Í	
Micr Utility Micrium RTOS, provides API: main, micrium-rtos	Plugin:		

In SDKs 6.9.0 and higher, you have the option of using Micrium RTOS or Free RTOS. For Micrium RTOS, check Micrium RTOS and RTOS Common.

Plugin configuration Use this section to select or unselect the plugins that you want to use in your application		
Q RTOS 2	-+1 -+1	Plugin: 🕪 Micrium RTOS
▼ ● \$ HAL ↓ ◆ HAL RTOS Library ▼ ● \$ Utility ↓ ◆ FreeRTOS, provides API: freertos, rtos-instance		Quality: 划 Production Ready Description: This plugin provides Micrium OS support. If this plugin is enabled, the stack will be running within Micrium OS as a Micrium OS task. The application can also define up to 3 custom Micrium OS tasks.
 ✓		

From SDK 6.9.x you can use FreeRTOS by selecting the FreeRTOS plugin and unchecking the Micrium RTOS plugin FreeRTOS is not supported yet. Theoriticially it should work, but we do not plan on testing it before Q4.

Plugin configuration			
Use this section to select or unselect the plugins that you want to use in your application			
Q RTOS C	-+	Plugin: 🚸 FreeRTOS	
		Quality: 🖋 Production Ready	
		Description:	
✓ → Soundy ✓ IPFeeRTOS, provides API: freertos, rtos-instance		This plugin provides FreeRTOS support. If this plugin is enabled, the stack will be running within FreeRTOS as a FreeRTOS task	
Micrium RTOS, provides API: micrium-rtos, rtos-instance			
🗹 🕸 RTOS Common, provides API: main, rtos-common			

BLE

•

P	lugin configuration Jse this section to select or unselect the plugins that you want to use in your application			^
	BLE	Plugin: 📣 BLE	^	
	🗸 🔳 🏂 Stack Libraries	Quality: 🖋 Production Ready		
	🖂 🍁 BLE, provides API: ble	Description:		
	Binding Table Library, provides API: binding Sulfity Address Table, provides API: address-table Ovice Table, provides API: device-table	If this plugin is enabled, the BLE stack will be run in parallel with the ZigBee stack. Both stacks run as Micrium OS tasks.		*

Idle/Sleep

Plugin configuration Use this section to select or unselect the plugins that you want to use in your application			^
Idle The second	Plugin:	^	
✓ ■ Subtility ✓ Image: Second sec	Quality: V Production Ready Description:		
	Ember implementation of idling and sleeping. This plugin can be used on devices that should deep sleep as well as on devices that need to stay awake. For devices with an RX-on-when-idle network (such as a router), the plugin will attempt to idle the processor when it has no other tasks to perform. Idling helps save power by halting the main loop of the application, but does not interfere with the timely handling of interrupts. For example, when idling, the radio can still receive		~

mbed TLS Multiprotocol Configuration

lugin configuration		
Use this section to select or unselect the plugins that you want to use in your application		
mBed 🔧	Plugin: 📣 mbed TLS Multiprotocol Configuration	^
🗸 🔳 🏂 Utility	Quality: 🖋 Production Ready	
✓ ↓ mbed TLS, provides API: mbedtls	Description:	
✓ ⊲> mbed TLS Multiprotocol Configuration, provides API: mbedtls-config-	This plugin provides the mbed TLS configuration file that enables the cryptographic algorithms that meet Silicon Labs multiprotocol requirements. The mbed TLS configuration file can be found at config-multiprotocol.h.	^

(optional) Dynamic Multiprotocol UI Demo Code Stub

The USART used to communicate with the external flash on Silicon Labs radio boards is the same USART that communicates with the LCD display on the WSTK. In order to perform an OTA update on a sample app that uses the LCD display, check **Dynamic Multiprotocol UI Demo Code Stub**. This disables the LCD display but allows the OTA to take place.

llugin configuration Use this section to select or unselect the plugins that you want to use in your application		
dyn the	Plugin: I Dynamic Multiprotocol UI Demo Code Stub	^
Duity Utility why and the second secon	Description:	
	This plugin provides stubs for the DynamicMultiprotocolDemoLight app. This plugin is to be used to satisfy compilation when the display driver is not used (primarily, when the Dynamic Multiprotocol UI Demo Code plugin is not enabled).	

2. On the Plugins tab, uncheck (disable) the following:

• Simple Main

•

Plugin configuration Use this section to select or unselect the plugins that you want to use in your application			^
simple	Plugin: 🚸 Simple Main	^	
✓ □ Smart Energy	Quality: 🖋 Production Ready (Certifiable)		
Simple Metering Client	Description:		
	This plugin provides a working main entry point into an EmberZNet application. It supplies a main () function that provides necessary initialization and main loop code for the program. Since this feature is a plugin, users can easily choose whether to use this plugin for their main() function, or		
□ - 🕼 Simple Main, provides API: main	to supply their own implementation. If the user does decide to write their own main() function, then this plugin can serve as a good example for main loop design.		~

- 3. On the Plugins tab, change settings for the following:
- HAL
 - Uncheck Simulated EEPROM version 1 Library.
 - Check Simulated EEPROM Version 2 to NVM3 Upgrade Library (this will also enable the NVM plugin).

Plugin configuration Use this section to select or unselect the plugins that you want to use in your application	1	
simu Image: Simulated EEPROM version 1 Library, provides API: sim-eeprom, toker Image: Simulated EEPROM version 1 to version 2 Upgrade Library, provides API: Image: Simulated EEPROM version 1 to version 2 Upgrade Stub, provides API: Image: Simulated EEPROM version 1 to version 2 Upgrade Stub, provides API: Image: Simulated EEPROM version 2 Library, provides API: sim-eeprom, toker Image: Simulated EEPROM version 2 to NVM3 Upgrade Library, provides API: Image: Simulated EEPROM version 2 to NVM3 Upgrade Stub, provides API: Image: Simulated EEPROM version 2 to NVM3 Upgrade Stub, provides API: Image: Simulated EEPROM version 2 to NVM3 Upgrade Stub, provides API:	Plugin: Quality: Description:	*

- RAIL
 - Uncheck RAIL Library.
 - Check RAIL Library Multiprotocol.

1 Silicon Labs Zigbee, version:6.6.1.0	« Pr	review
🝰 General 🖧 ZCL Clusters 🐊 Zigbee Stack 🐊 Printing and CLI 🔘 HAL 🗇 Plugins 🖉 Callbacks 🚴 Includes 🗼 Other options 🔞 Bluetooth GATT		
Plugin configuration		^
Use this section to select or unselect the plugins that you want to use in your application		
RAIL Plugin:	^	
Quality:		
□ 🗣 RAIL Library, provides API: rail-library Description:		
RAIL Library Multiprotocol, provides API: rail-library, radio-multiproto	^	
		×

- UTILITY
 - In the CCM* Encryption plugin, uncheck **AES-CCM Software Implementation** and check **mbedTLS Implementation**.

Plugin configuration

	· login comparation
l	Use this section to select or unselect the plugins that you want to use in your application
L	

CCM	Plugin:		^
Vility	Quality: 🧭 working		
✓ ⊲p> CCM* Encryption, provides API: ccm-star	Description:		
	This plugin provides the AES-CCM* api for encryption, decryption, and Message Integrity Authentication. Implementation can be provided by the mbedTLS framework, or through software	< >	
	Options: Reset to defaults AES-CCM Software Implementation Implementation	>	*

4. Add Bluetooth GATT elements:

On the **Bluetooth GATT** tab, **Services** tab, drag and drop the following into the Custom BLE GATT window:

- Device Information
- Generic Access

s s comparator Source filters @81 @ Silicon Labs	Custom BLE GATT	13
vofiles Services Characteristics Descriptors		18
devi	£	0
Device Information Generic Access Generic Parme Human Interface Device	Concert Information	4 23
envice Jame Device Information yes: org bluetooth service.device_information JUID: 180A ource BT Jutract:	Select a GATT item to configure	
The Device Information Service exposes manufacturer and/or vendor information about a device. ummaty: his service exposes manufacturer information about a device. The Device Information Service is rotantiated as a Primary Service. Only one instance of the Device Information Service is exposed in a device.		
Characteristics of the service: Manufacturer Name String Characteristic		

5. Add an identifiable name for your device to advertise by modifying the **Generic Access** > **Device Name Value**. Be sure to increase the **Length** 1 byte per character.

4	 C PPP ID Generic Access Device Name Appearance Peripheral Privacy Flag Reconnection Address Peripheral Preferred Connection Parar 	neters	
Genera	al settings		
Name	Device Name		
	User description]
Charao	cteristic settings		
1	ID device_name	UUID	2A00
SIG typ	e org.bluetooth.characteristic.gap.device		
Value	settings		
Value	DMP4EVA	Value type	utf-8 🔻
Length	7 🖶 byte	Variable	e length
Droport	ties		

6. Add BLE callback code:

In the Callbacks tab, enable Plugin-specific callbacks: Event.

*	General 💰 ZCL Clusters 🎄 Zigbee Stack 🍰	Printing ar	nd CLI 🔘 HAL 🗇 Plugins	; s	Callbacks 🛛 🍰 Iı
	E E X				/** @brief *
	Name	Use	Туре	-	* This fund the BLE stad
	a 🖧 Non-cluster related				application
	😅 Clear Report Table	-4⊳	Custom callback, in plugin:		* stack eve
	Start Search For Joinable Network		Custom callback		*/
	a 🖧 Plugin-specific callbacks				voia omboràfPlua:
	S Event	\checkmark	Plugin specific: BLE		uct gecko cr
	😅 Get Config		Plugin specific: BLE	-	uee geene_er
	A Handlers defined by stack				
	A Callbacks defined by APIs				

7. Generate the project.

8. Add emberAfPluginBleEventCallback code to the [project_name]_callbacks.c file, based on the SDK version you are using..

For SDK 6.7.x:

```
/** @brief
```

```
* This function is called from the BLE stack to notify the application of a
* stack event.
* In this case it will provide simple advertising for the BLE component of the Z3Light DMP app
*/
```

void emberAfPluginBleEventCallback(struct gecko_cmd_packet* evt){

```
switch (BGLIB MSG ID(evt->header)) {
```

case gecko_evt_system_boot_id:

```
gecko_cmd_le_gap_start_advertising(0, le_gap_general_discoverable, le_gap_connectable_scanna-
```

```
ble);
```

```
emberAfCorePrintln("BLE Advertising started");
    break;
case gecko_evt_le_connection_opened_id:
    emberAfCorePrintln("BLE connection opened"); //Will cause advertising to stop
    break;
```

```
case gecko_evt_le_connection_closed_id:
        emberAfCorePrintln("BLE connection closed");
        gecko_cmd_le_gap_start_advertising(0, le_gap_general_discoverable, le_gap_connectable_scanna-
ble); // restarting advertising
        break;
        default :
        emberAfCorePrintln("unhandled BLE event\r\n");
        break; }
```

}

For SDK 6.8.x:

```
#include "sl_bt_rtos_adaptation.h"
static uint8 t advertising set handle = 0xff;
/** @brief
 ^{\star} This function is called from the BLE stack to notify the application of a
 * stack event.
 * In this case it will provide simple advertising for the BLE component of the Z3Light DMP app
 */
    void emberAfPluginBleEventCallback(sl_bt_msg_t* evt){
        switch (SL_BT_MSG_ID(evt->header)) {
        case sl_bt_evt_system_boot_id:
               sl_bt_advertiser_create_set(&advertising_set_handle);
               sl_bt_advertiser_start(advertising_set_handle,
                                                                         // advertising set handle
                                      advertiser_general_discoverable, // discoverable mode
                                      advertiser_connectable_scannable); // connectable mode
               emberAfCorePrintln("BLE Advertising started");
               break;
        case sl_bt_evt_connection_opened_id:
               emberAfCorePrintln("BLE connection opened"); //Will cause advertising to stop
               break;
        case sl_bt_evt_connection_closed_id:
               emberAfCorePrintln("BLE connection closed");
                                                                         // advertising set handle
               sl_bt_advertiser_start(advertising_set_handle,
                                      advertiser_general_discoverable, // discoverable mode
                                      advertiser_connectable_scannable); // connectable mode
               break;
        default:
               emberAfCorePrintln("unhandled BLE event\r\n");
               break;
       }
```

```
}
```

9. For SDK 6.8.x and up, the following defines need to be added to the file mbedtls-config-generated.h:

```
#define MBEDTLS_CTR_DRBG_C
#define MBEDTLS_SHA256_C
#define MBEDTLS_ENTROPY C
```

10. Build and flash your project and look for your device in the Bluetooth Browser screen of the EFR Connect cell phone app.

:		
Browser		
i 三 Log	0 Connection	s Q Filter
N/A ★		④ 103ms
35:68:78:40:84:9A		
·*·	(;	9
Non-Connectable	-38dBm	Unspecified
DMP4EVA ★		③ 103ms Connect
00:0B:57:64:8D:EB		
·*	((:	•
Connectable	-49dBm	Unspecified
N/A ★ 4F:07:18:B5:61:73		Oms Connect
·*·	()	0
Connectable	-96dBm	Unspecified
N/A +		@ 100ms
15:5A:87:F3:53:78		e round
.¥.		0
Non-Connectable	-94dBm	Unspecified
N/A 🛒 7A:4A:DC:E8:34:DB		@ 2/4ms Connect
·*	(;	9
Connectable	-66dBm	Unspecified
		_
	Stop Scanni	ng

You can also see comments for the BLE activity in the Serial 1 window among the Zigbee prints

😤 Serial 0 🚔 Serial 1 🚔 Admin 🚔 Debug
Z3LightSoc2DMP_6610_brd4161>success
Z3LightSoc2DMP_6610_brd4161>Reset info: 0x06 (SW)
Extended Reset info: 0x0600 (UNK)
ZllCommInit - device is not joined to a network
Setting rx on period to 300000
Setting default channel to 11
BLE Advertising started
NWK Steering State: Scan Primary Channels and use Install Code
Error: NWK Steering could not setup security: 0xB7
NWK Steering State: Scan Secondary Channels and use Install Code
Error: NWK Steering could not setup security: 0xB7
NWK Steering State: Scan Primary Channels and Use Centralized Key
Starting scan on channel 24
NWK Steering: Start: 0x00
Join network start: 0x00
Starting scan on channel 25
Starting scan on channel 11
lles is The second s

This is very basic Bluetooth functionality. To learn more about programming BLE functionality, see QSG139: Bluetooth® SDK v2.x Quick Start Guide/QSG169: Bluetooth® SDK v3.x Quick Start Guide, included with the corresponding Bluetooth SDK.

3 About the Zigbee/Bluetooth LE Examples

The Zigbee/Bluetooth LE Dynamic Multiprotocol examples demonstrate a light that can be controlled from both Bluetooth and a Zigbee network. Software is included both as compiled demonstrations and as example code in the EmberZNet SDK. The purpose of the examples is to show the way of implementing a dynamic multiprotocol application using the Silicon Labs EmberZNet stack.

The Dynamic Multiprotocol Demo application has three main components.

- 1. User Interface (LCD and Buttons)
- 2. Zigbee application (FFD and/ or SED)
- 3. Bluetooth application

3.1 User Interface

The user interface is developed specifically for the dynamic multiprotocol demonstration, and APIs to update the text and graphic on the LCD are called directly from Zigbee and Bluetooth event handlers. The implementation to manipulate the LCD is contained in the following files,

bitmaps.h //Contains the arrays containing the bitmap of the graphics drawn on the LCD

 ${\tt dmp_ui.c}$ //Contains the functions to change the state of the display based on the state of the application

dmp ui.h //Header file exporting functions implemented in the dmp ui.c

The above uses the display driver library supplied by Silicon Labs to update the content on the LCD display mounted on the WSTK.

3.2 Zigbee Application

The example **DynamicMultiprotocolLight** is set up to be a light and a coordinator on the Zigbee network.

The following cluster set is supported by both the **DynamicMultiprotocolLight** and **DynamicMultiprotocolLightSed** applications.

Supported Clusters		
Basic		
Identify		
Scenes		
Groups		
On/Off		
ZLL Commissioning		

The **DynamicMultiprotocolLight** example also supports Green Power Proxy Basic behavior. Please note that the examples were developed with a focus on demonstrating dynamic multiprotocol features and may not be Zigbee-certifiable.

The On/Off cluster controls the LEDs and the bulb icon on the WSTK board to represent the state of the light.

The dynamic multiprotocol applications make use of Micrium OS and the Zigbee applications are run as a task of Micrium OS.

The hardware and peripherals of the chip are initialized before any tasks are created. A Zigbee task is created after initialization, which then creates the application tasks and Bluetooth task.

The Micrium plugin also includes the source file micrium-rtos-sleep.c, which enables the sleepy DMP application to manage the sleep functionality.

Note that in Bluetooth SDK v3.x commands were renamed and restructured. The code example in the last box illustrates both.

AN1133: Dynamic Multiprotocol Development with Bluetooth and Zigbee EmberZNet SDK 6.x and Lower About the Zigbee/Bluetooth LE Examples



AN1133: Dynamic Multiprotocol Development with Bluetooth and Zigbee EmberZNet SDK 6.x and Lower About the Zigbee/Bluetooth LE Examples

From: af-main-soc.c



On either DMP light application, once the Zigbee stack is set up to run, subsequent interactions with the stack occurs via event handlers, as shown in the following figures. The following figure shows the event handlers in the full function light application.



Figure 3-1. DMP Full Function Light Event Handler Definition

Note: Whenever the light starts pjoin, it starts identifying **and** also puts all the connected lights in identify mode. This helps the joining switch to identify all the lights present in the network.

The following figure shows the application interaction with the stack with the event handlers used for the sleepy light application.



Figure 3-2 DMP Sleepy Light Event Handler Definition

Note: To avoid the risk of shared resources, if you want to send Zigbee messages from a task other than the Zigbee Stack Task, we advise you to schedule a custom event from within the non-Zigbee Stack task. In the corresponding event handler function for the custom event the Zigbee stack APIs can be used, as the event handler will be called from the Zigbee Stack Task context.

3.3 Bluetooth Application

The Bluetooth application supports following services and characteristics. These are pre-selected in the GATT editor during project generation.

Service	Characteristic
Device Information	Manufacturer Name String Model Number String Serial Number String Firmware Revision String
Generic Access	Device Name Appearance
Silabs DMP Light	Light Trigger Source

3.3.1 Silabs DMP Light Service

In the above table the Silabs DMP Light is a custom service with a UUID of bae55b96-7d19-458d-970c-50613d801bc9. This custom UUID is used to uniquely identify the Light by the Wireless Gecko application.

The Service has two characteristics,

Characteristic	Data Type	Description
Light	8bit Boolean	Used to get and set the light state 1 = Light On 0 = Light Off
Trigger Source	8bit <u>enum</u>	Indicates the source of the Light state change command. 0 = Bluetooth 1 = Zigbee 2 = Button Press

3.3.2 Beacons

The application implements both an iBeacon as well as an Eddystone beacon. The default behavior is to transmit each beacon at 100 mS intervals.



3.3.3 Bluetooth Event Handling

The Bluetooth stack is initialized as part of the Zigbee Task, as shown in the Zigbee implementation section. The Bluetooth task handles the Bluetooth LE link layer messaging and management. The Bluetooth stack's interaction with the user application is through a framework plugin. A number of events that are called in the context of the Zigbee task allow the user application to interact with the Bluetooth stack. The following diagram describes the Bluetooth-related events. In Bluetooth v3.x commands and events were renamed, substituting **sl_bt_** for **gecko_cmd_** and **gecko_**, respectively. Both variants are shown.

Note: Bluetooth event handling is same for both DMP demos.



Figure 3-3 DMP Bluetooth Event Handler Definition

3.3.4 Bluetooth and Zigbee Interaction

The primary purpose of the example applications is to show Zigbee and Bluetooth working together on a device. For this purpose, when the Light receives a command to change its state through one protocol, it executes the command and sends out a notification to the other devices using the other protocol to keep everything in sync. Their interaction is the same in both examples.

Two basic operations are described below, first a write to Light characteristics from a Bluetooth connected device (shown in the following figure) and then a change in the Light state from a Zigbee device.

Write from the Bluetooth Connected Device



The application's services and characteristics are pre-selected in the GATT editor in Simplicity Studio. Upon generation the characteristics are #define in the gatt_db.h. Using the #define reference, the characteristics can then be coupled to read and write Bluetooth requests. For example the Light characteristic is reference from GATT as gatt_light_state which is then tied to an application specific write API of writeLightState in the AppCfgGattServerUserWriteRequest as shown below.

```
static const AppCfgGattServerUserWriteRequest_t appCfgGattServerUserWriteRequest[] =
{
    { gattdb_light_state, writeLightState },
    { 0, NULL }
};
```

The application implements the Zigbee attribute write and a Bluetooth write response in the writeLightState function as follows:

The emberAfWriteAttribute() is used to write the attribute table of the Zigbee application with the value supplied by the Bluetooth connected device above. Since the on-off attribute of the on-off server cluster is a reportable attribute it is reported to all devices setup in the binding table of the Light.

The <code>emberAfOnOffClusterServerAttributeChangedCallback()</code> is then used to change the state of the LEDs and the LCD to indicate the state of the light on the WSTK main board.

Write from the Zigbee Connected Device

The flow in the other direction, that is a change in the Light state from Zigbee connected device, is shown in the following figure.



Any on-off client on the same network as the Light can send an on-off cluster's On, Off or Toggle command to the Light to change its state. Once such a command is received over the Zigbee interface the Silicon Labs Zigbee framework interprets it and calls an appropriate handler to change the value of on-off attribute of the on-off server cluster. In the example **DynamicMultiprotocolSwitch** application the on-off client sends a Toggle command to the Light, which toggles the value of the on-off attribute and triggers the <code>emberA-fOnOffClusterServerAttributeChangedCallback()</code>. The callback is then used to change the state of the light as well as send notifications for both Trigger Source and Light characteristics to the connected Bluetooth devices and to update the LEDs and the LCD to indicate the change in the Light state.

```
void emberAfOnOffClusterServerAttributeChangedCallback(int8u endpoint,
                                                        EmberAfAttributeId attributeId)
{
  EmberStatus status;
  int8u data;
  if (attributeId == ZCL ON OFF ATTRIBUTE ID) {
    status = emberAfReadAttribute(endpoint,
                                    ZCL ON OFF CLUSTER ID,
                                    ZCL ON OFF ATTRIBUTE ID,
                                    CLUSTER MASK SERVER,
                                    (int8u*)&data,
                                    sizeof(data),
                                    NULL);
    if (status == EMBER ZCL STATUS SUCCESS) {
      if (data == 0x00) {
        halClearLed(BOARDLED0);
       halClearLed(BOARDLED1);
        dmpUiLightOff();
       notifyLight(currentConnection, 0);
      } else {
       halSetLed(BOARDLED0);
        halSetLed(BOARDLED1);
        notifyLight(currentConnection, 1);
        dmpUiLightOn();
      if ( (lightDirection == DMP UI DIRECTION BLUETOOTH)
           || (lightDirection == DMP UI DIRECTION SWITCH) ) {
        dmpUiUpdateDirection(lightDirection);
      } else {
        lightDirection = DMP UI DIRECTION ZIGBEE;
        dmpUiUpdateDirection(lightDirection);
      ble_lastEvent = lightDirection;
      lightDirection = DMP_UI_DIRECTION_INVALID;
      if (ble_lastEvent != DMP_UI_DIRECTION_INVALID) {
if ( (ble_lightState_config !=
GAT_RECEIVE_INDICATION)) {
                                                    GAT RECEIVE INDICATION)
                                                                                66
                                                                                       (ble lastEvent config
         notifyTriggerSource(currentConnection, ble lastEvent);
        }
      }
    }
  }
   else {
  }
} }
```

4 Document Revision History

Revision 0.9

• Documents the new freeRTOS support

Revision 0.8

- Removed the "Define the mbedTLS path" step in the procedure in section 2.2.
- Modified the 6.8.x code that illustrates adding emberAfPluginBleEventCallback code to the [project_name]_callbacks.c file,

Revision 0.7

• Update to reflect changes for EmberZNet 6.8.0/Bluetooth SDK 3.0.0.

Revision 0.6

- Section 2.2 re-inserted with functional instructions.
- Fixed duplicated bookmarks in PDF.

Revision 0.5

• Updated section 2.1. Temporarily removed section 2.2 on project configuration.

Revision 0.4

• Added note about conflict between LCD and external flash for OTA.

Revision 0.3

• Added note about threat-safe implementation to section 3.2.

Revision 0.2

• Modifications for supporting sleepy light device.

Revision 0.1

Initial release

Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!



www.silabs.com/IoT



www.silabs.com/simplicity



www.silabs.com/quality



Support & Community www.silabs.com/community

Disclaimer

Silicon Labs intends to provide customers with the latest, accurate, and in-depth documentation of all peripherals and modules available for system and software implementers using or intending to use the Silicon Labs products. Characterization data, available modules and peripherals, memory sizes and memory addresses refer to each specific device, and "Typical" parameters provided can and do vary in different applications. Application examples described herein are for illustrative purposes only. Silicon Labs reserves the right to make changes without further notice to the product information, specifications, and descriptions herein, and does not give warranties as to the accuracy or completeness of the included information. Without prior notification, Silicon Labs may update product firmware during the manufacturing process for security or reliability reasons. Such changes will not alter the specifications or the performance of the product. Silicon Labs shall have no liability for the consequences of use of the information supplied in this document. This document does not imply or expressly grant any license to design or fabricate any integrated circuits. The products are not designed or authorized to be used within any FDA Class III devices, applications for which FDA premarket approval is required or Life Support Systems without the specific written consent of Silicon Labs. A "Life Support System" is any product or system intended to support or sustain life and/or health, which, if it fails, can be reasonably expected to result in significant personal injury or death. Silicon Labs products are not designed or authorized for military applications. Silicon Labs product shall under no circumstances be used in weapons of mass destruction including (but not limited to) nuclear, biological or chemical weapons, or missiles capable of delivering such weapons. Silicon Labs disclaims all express and implied warranties and shall not be responsible or liable for any injuries or damages related to use of a Silicon Lab

Trademark Information

Silicon Laboratories Inc.[®], Silicon Laboratories[®], Silicon Labs[®], SiLabs[®] and the Silicon Labs logo[®], Bluegiga[®], Bluegiga Logo[®], EFM[®], EFM32[®], EFR, Ember[®], Energy Micro, Energy Micro logo and combinations thereof, "the world's most energy friendly microcontrollers", Redpine Signals[®], WiSeConnect, n-Link, ThreadArch[®], EZLink[®], EZRadio[®], EZRadio[®], Gecko[®], Gecko OS, Gecko OS Studio, Precision32[®], Simplicity Studio[®], Telegesis, the Telegesis Logo[®], USBXpress[®], Zentri, the Zentri logo and Zentri DMS, Z-Wave[®], and others are trademarks or registered trademarks of Silicon Labs. ARM, CORTEX, Cortex-M3 and THUMB are trademarks or registered trademarks of ARM Holdings. Keil is a registered trademark of ARM Limited. Wi-Fi is a registered trademark of the Wi-Fi Alliance. All other products or brand names mentioned herein are trademarks of their respective holders.



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

www.silabs.com