

UG495: Silicon Labs Wi-SUN Developer's Guide

This document is a reference for those developing applications using the Silicon Labs Wi-SUN (Wireless Smart Ubiquitous Network, Field Area Network) SDK (Software Development Kit). The guide covers the (Wi-SUN) stack architecture, application development flow, steps to configure the application Wi-SUN radio settings and advanced debug features. This version applies to the Silicon Labs Wi-SUN SDK version 1.x.x and higher.

The purpose of this document is to fill in the gaps between the Silicon Labs Wi-SUN Field Area Network (FAN) API reference, Gecko Platform references, and documentation for the target EFR32xG part. This document provides details that will help developers optimize their application for their target environment.

KEY POINTS

- Wi-SUN SDK stack, including firmware and application project structure, and software components
- Application development guidelines
- Wi-SUN radio configuration
- Advanced tools to test and debug a Wi-SUN application

Table of Contents

1	Intro	oduction	1
2	Wi-	SUN FAN Stack	2
	2.1	Firmware Structure	2
	2.2	Application Project Structure	3
	2.2.	.1 Wi-SUN Files Library Files	3
	2.2.	.2 RAIL	3
	2.2.	.3 EMLIB and EMDRV	3
	2.2.	.4 Mbed TLS	3
	2.3	Optional Software Components	4
3	Wi-	SUN Application Development	5
	3.1	Responding to Wi-SUN Events	5
	3.2	Implementing Application Logic	5
	3.3	Changing Operating System	5
	3.4	Using a Different Development Environment	6
	3.5	Wi-SUN Stack Heap Requirement	7
4	Wi-	SUN Configurator	8
	4.1	Application Panel	8
	4.2	Security Panel	9
	4.3	Radio Panel	10
	4.4	Changing the Default Wi-SUN Radio Configuration	11
5	Tes	sting and Debugging	13
	5.1	Access Debug Traces from the Wi-SUN Stack	13
	5.2	Export Wi-SUN Traces to Wireshark	13
	5.3	Connect the Wi-SUN Network to Another IP Network	16

1 Introduction

This document contains information for anyone developing applications in the Silicon Labs Wi-SUN SDK. It assumes that the current version of the Silicon Labs Wi-SUN SDK has been installed and that the developer is familiar with creating and flashing applications, and with the functionality available as a starting point in the example files contained in the SDK. If you are not familiar with these items, and are just getting started, see the <u>Simplicity Studio 5 User's Guide</u> and <u>QSG181: Silicon Labs Wi-SUN Quick-Start Guide</u>. For more information about configuring a Wi-SUN network, see <u>AN1332: Silicon Labs Wi-SUN Network Setup and Configuration</u>.

The Silicon Labs Wi-SUN API reference that matches the installed SDK is available on the Simplicity Studio DOCUMENTS tab. All versions are available at https://docs.silabs.com/wisun/latest/wisun-stack-api/.

2 Wi-SUN FAN Stack

2.1 Firmware Structure

The following figure describes the high-level firmware structure. The developer creates an application on top of the stack, which Silicon Labs provides as a precompiled object-file, enabling the Wi-SUN connectivity for the end-device.



Figure 2.1. Wi-SUN Stack Architecture Block Diagram

The Wi-SUN stack contains following blocks.

- Wi-SUN stack Wi-SUN functionality consisting of an IP stack, MAC layer, the routing protocol (RPL), and security manager.
- Wi-SUN RF test plugin Optional software component to add an API to perform RF tests (for example, create an RF tone).
- Wi-SUN Util Functions Optional software component to add helper functions to inform the application about the Wi-SUN PHY configured in the RAIL configuration file.

2.2 Application Project Structure

This section explains the application project structure and the mandatory and optional resources that must be included in the project.

2.2.1 Wi-SUN Files Library Files

The Wi-SUN stack libraries are summarized in the following table.

	Table	2.1.	Wi-SU	N Stac	k L	ibraries
--	-------	------	-------	--------	-----	----------

Wi-SUN stack library name	To use with
libwisun_router(_core)_efr32xg X x_micrium_gcc_debug.a	Wi-SUN router with Micrium OS, GCC, with debug traces
libwisun_router(_core)_efr32xg X x_micrium_gcc_release.a	Wi-SUN router with Micrium OS, GCC, no debug trace
libwisun_router(_core)_efr32xg X x_micrium_iar_debug.a	Wi-SUN router with Micrium OS, IAR, with debug traces
libwisun_router(_core)_efr32xg X x_micrium_iar_release.a	Wi-SUN router with Micrium OS, IAR, no debug trace
libwisun_router(_core)_efr32xg X x_freertos_gcc_debug.a	Wi-SUN router with FreeRTOS, GCC, with debug traces
libwisun_router(_core)_efr32xg X x_freertos_gcc_release.a	Wi-SUN router with FreeRTOS, GCC, no debug trace
libwisun_router(_core)_efr32xg X x_freertos_iar_debug.a	Wi-SUN router with FreeRTOS, IAR, with debug traces
libwisun_router(_core)_efr32xg X x_freertos_iar_release.a	Wi-SUN router with FreeRTOS, IAR, no debug trace
libwisun_rcp_efr32xg X x.a	Radio coprocessor (Linux border router)
libwisun_mac(_core)_efr32xg1 X x.a	MAC layer for the Radio coprocessor (Linux border router)

The Wi-SUN stack library file names containing "_core_" do not support the LFN feature. The **X** stands for the supported MCU generation. The file names containing "_efr32xg1x_" are built to run on Series 1 MCUs and those containing "_efr32xg2x_" are built to run on Series 2 MCUs.

2.2.2 RAIL

The Wi-SUN stack uses RAIL to access the radio and RAIL libraries needs to be linked with Wi-SUN stack. RAIL has separate libraries for each device family and for single- and multi-protocol environments. RAIL libraries are provided in the Gecko SDK Suite. For more information refer to *UG103.13: RAIL Fundamentals* and other RAIL documentation.

2.2.3 EMLIB and EMDRV

The Wi-SUN stack uses EMLIB and EMDRV libraries to access EFR32 hardware. EMLIB and EMDRV peripheral libraries are provided in source code, and they must be included in the project. EMLIB and EMDRV are part of the Gecko SDK Suite. For more details on EMLIB and EMDRV, see platform EMDRV documentation and EMLIB documentation on https://docs.silabs.com.

2.2.4 Mbed TLS

The Wi-SUN stack uses the Mbed TLS library for cryptographic operations. The Mbed TLS library is provided in source code and must be included in the project. Mbed TLS is part of the Gecko SDK Suite. For more details, refer to the Mbed TLS documentation.

2.3 Optional Software Components

In addition to the Wi-SUN stack core functionality, the Wi-SUN SDK contains optional software components that you can leverage to customize the application. Add those components in the SOFTWARE COMPONENTS tab of a project, as shown in the following figure:

5 v5.master.Staging_2269 - wisun_soc_coap_meter/wisun_soc	_coap_meter.slcp - Simplicity Studio™						-	□ ×
<u>File Edit Navigate Search Project Run Window H</u> e	lp							
<u>** • 9: • : ☆ • ::: </u>	🏷 -?* 🗘 ▾ -▷ ▾ 📷 🖬 🖬 🖬 Welcome 💿 R	ecent 🚺 Tools 🛃	🛃 Install 🗱 Prefere	nces			🔡 👔 Simplicity	IDE 🚀 Launcher
Project Explorer 🗙 📄 😫 🍟 🗖	😅 Wi-SUN Configurator 🛛 🚢 wisun_soc_coap_me	ter.slcp 🗙 🗖 rea	dme.md					- 0
✓ Sincludes > Sincludes	wisun_soc_coap_meter	OVERVIEW	SOFTWARE C	OMPONENTS	CONFIGU	JRATION TO	OOLS	
> >> autogen > >> config > gecko_sdk_4.1.0 > & app_custom_callback.c	▼ Filter components by ✿ Configurable	📄 🛛 Installe	d 🗌 💄 Insta	lled by you 🗌	🖿 SDK exter	nsions	Q Search keywords, component's name	
> h app_custom_callback.h > & app_init.c	▼ Wi-SUN		*					
> h app_init.h > k app.c	 Application 							
> In app.n > Io main.c	⊘ Wi-SUN Configuration		•					
 readme_img0.png readme_img1.ang 	▼ Wi-SUN Plugin							
readme_mgi.png	RF Test Plugin							
wisun_soc_coap_meter.slop	⊘ Stack Debug & Traces Plugin							
	▼ Wi-SUN Services							
	⊘ Application Core		•					
<>	⊘ CoAP		•					
Debug Adapters × 📴 Outline	Ø POSIX-compliant Socket		•					
◇ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓	⊘ Ping		•					
	⊘ Setting							
	⊘ Utilities		_					
	Ø Wi-SUN Crypto							
	Wi-SUN FAN Certification		•					
	Ø Wi-SUN Stack							
	Problems X 🔗 Search 🔅 Call Hierarchy	Console	•					78 - 0
	0 items							
	Description	Resource	Path	Location	Туре			
pJ	L							

Figure 2.2. Wi-SUN Software Components

There are four important components:

- Application core: Provides application basic functionalities like event handling, callback management, and the Wi-SUN network connection.
- CoAP: Provides a Constrained Application Protocol (CoAP) implementation running on top of the Wi-SUN stack. The CoAP component should be used as an example of implementation of other software libraries on top of the Silicon Labs Wi-SUN stack.
- iPerf: Provides a widely supported tool to evaluate throughput on IP interfaces. The implementation can interoperate with other iPerf2 implementations. It currently only supports UDP server and client modes.
- POSIX-Compliant Socket: Provides a POSIX-like socket API on top of the standard Wi-SUN stack socket API. In addition to the API abstraction, this component makes the socket accesses thread-safe.

For the complete software documentation, visit https://docs.silabs.com/wisun/latest/wisun-stack-api/sl-wisun-services .

3 Wi-SUN Application Development

To get started with Wi-SUN application development, Silicon Labs recommends that you become familiar with different Wi-SUN sample applications. Then, you can use the Wi-SUN SoC Empty sample application as a template and a starting point for a new application.

The development of a Wi-SUN application consists of two main steps:

- 1. Responding to the events raised by the Wi-SUN stack.
- 2. Implementing additional application logic.

Optionally, you can change several Wi-SUN application settings with a few clicks:

- 1. Operating system used by the application.
- 2. IDE (Integrated Development Environment) used during the development.

3.1 Responding to Wi-SUN Events

A Wi-SUN application is event-driven. The Wi-SUN stack generates events when a connection is successful, data has been sent, or an IP packet is received. The application has to handle these events in the $sl_wisun_on_event()$ function. The prototype of this function is implemented in app.c. To handle more events, the switch-case statement of this function can be created and extended. For the list of Wi-SUN events, visit <u>https://docs.silabs.com</u>.

3.2 Implementing Application Logic

Additional application logic can be implemented in the $app_task()$ function, defined in app.c. The $app_task()$ function is called once after the device is booted and the Wi-SUN stack is initialized. Most Wi-SUN applications' first step is to call $sl_wisun_join()$ to connect the Wi-SUN device to a Wi-SUN border router. The remaining implementation is up to the developer. Visit <u>https://docs.silabs.com</u> to check the list of Wi-SUN APIs available to the application.

3.3 Changing Operating System

Simplicity Studio 5 provides the ability to easily replace software components. This feature is leveraged to change the Real-Time Operating System (RTOS) used by the application and the Wi-SUN stack. To change the RTOS, complete these steps:

- 1. Go to the project SOFTWARE COMPONENTS tab.
- 2. Uninstall the Micrium OS Kernel component (default RTOS).



3. Install the FreeRTOS component.

🕈 Wi-SUN Configurator 🔒 wisun_soc_coap_meter.slcp 🗙 🗖 readme.md	- 1
	Validation Errors
wisun_soc_coap_meter OVERVIEW SOFTWA	RE COMPONENTS CONFIGURATION TOOLS
▼ Filter components by ✿ Configurable □ ● Installed □ L	installed by you 🔲 📾 SDK extensions 🔲 🔍 Search keywords, component's name
► Platform	FreeRTOS
▼ RTOS	
▼ FreeRTOS	Bundering
FreeRTOS 🌣	Description FreeRTOS kernel
FreeRTOS Heap 1	Quality
FreeRTOS Heap 2	PRODUCTION
FreeRTOS Heap 3	
FreeRTOS Heap 4	Dependencies ~
FreeRTOS Heap 5	freertos requires 2 components
▼ Micrium OS	▶ Platform
► Common	▶ RTOS
▼ Kernel	Dependents
Micrium OS Kernel	2 components require freertos
► Radio	▶ RTOS
* Services	

3.4 Using a Different Development Environment

In addition to Simplicity Studio 5, you can use alternative Integrated Development Environment (IDEs). To generate a GCC makefile or an IAR Embedded Workbench project:

- 1. Go to the OVERVIEW tab.
- 2. Scroll down to the end of the Target and Tool Settings card and click Change Target/SDK/Generators.
- 3. In the CHANGE PROJECT GENERATORS list, select the type of projects to be generated.
- 4. Click Save and wait for Simplicity Studio to generate the project.

 	 Ielp								
株 ▼ 😪 ▼ 🗂 ▼ 🔛 🔞 🕲 ▼ 🗞 ▼ 🖄 ▼ 🖓 ▼	🕫 🏷 🗇 🗸 🗢 👻 📷 🐘 🚹 Welcome 🕑 I	Recent 🔢 Tools	🛃 Install 🔹 Pref	erences				: 🗈 {} Simj	plicity IDE 🚀 Launcher
🍋 Project Explorer 🗙 📄 😫 🦳 🗖	🔿 Wi-SUN Configurator 🛛 🚢 wisun_soc_coap_m	eter.slcp 🗙 🗖 r	eadme.md						- 0
Signature Signature	wisun_soc_coap_meter	OVERVIEW	SOFTWARE	COMPONENT	IS CONF	GURATION 1	TOOLS		
 > config > config > getko_sdk_4.1.0 > config > app_custom_callback.c > map_custom_callback.h > maping 	FER32MG12P433E1024GI 125		Gecko SD Bluetooth 3.4.0.0, H Kernel, Op	K Suite: Amazon, E Mesh 3.0.0, Embe omeKit 1.2.0.0, MC benThread 2.1.0.0 (Bluetooth 3.4.0, IrZNet 7.1.0.0, Fle CU 6.3.0.0, Micriu (GitHub-9a2d84a	x m OS 4b),	ò		
 by pp_int.h c) app.c b) app.h c) main.c iii main.c 	Wireless Starter Kit Mainboard (BRD4001A R EFR32MG12 2400/915 MHz 19 dBm Dual Ba Board (BRD4164A Rev A02)	ev A01) nd Radio	Wave SDF	(7.18.0.0 Mode	u, Wi-SUN 1.3.0.0	2-	٢	Pin Tool	
	Selected SDK Gecko SDK Suite: Amazon, Bluetooth 3.4.0, E Mesh 3.0.0, EmberZNet 7.1.00, Flex 3.4.0.0, 1.2.0.0, MCU 6.3.0.0, Micrium OS Kernel, Ope	luetooth HomeKit nThread	Link s	Link sdk and copy project sources 👻				Memory Editor	
wisun_soc_coap_meter.project.mak wisun_soc_coap_meter.project.mak wisun_soc_coap_meter.groject.mak wisun_	2.1.0.0 (6itHub-9a2d84a4b), Platform 4.1.0. 1.0.0.0, WI-SUN 1.3.0.0, Z-Wave SDK 7.18.0.0 Project Generators Simplicity IDE Project), USB					Z	docs.silabs.com	- 1
> EFR32MG12.2400/915 MHz 19 dBm RB (ID:440085386)	Change Target/SDK/Genera	tors		Force Gen	eration				
	Problems X 🔗 Search 🎲 Call Hierarchy	📮 Console							78-0
	0 items								
	Description	Resource	Path	Location	Туре				

3.5 Wi-SUN Stack Heap Requirement

The Wi-SUN stack relies on dynamic memory allocation to function. It stores received and outgoing packets, security tokens, routing information, and more. The peak memory requirement is reached during the connection to a Wi-SUN network, especially during the authentication step. The heap size is defined under the *config/sl_memory_config.h* file in the Wi-SUN sample applications. By default, the heap size is configured through the SL_HEAP_SIZE define and is equal to 0x10000.

```
// <o SL_HEAP_SIZE> Minimum heap size for the application.
// <i> Default: 2048
// <i> Note that this value will configure the c heap which is normally used by
// <i> <u>malloc()</u> and free() from the c library. The value defines a minimum heap
// <i> size that is guaranteed to be available. The available heap may be larger
// <i> to make use of any memory that would otherwise remain unused.
#ifndef SL_HEAP_SIZE
#define SL_HEAP_SIZE 0x10000
```

#endif

This heap size is largely inflated to accommodate potential application level requirements. The bare minimum heap size recommended to run the Wi-SUN stack is 0xC000.

In addition to the standard heap size requirement, the Wi-SUN stack relies on an RTOS: Micrium OS or FreeRTOS. The stack requires a number of tasks, queues, mutexes to be created. The size of this memory pool is defined either by:

- LIB_MEM_CFG_HEAP_SIZE when using Micrium OS
- configTOTAL_HEAP_SIZE when using FreeRTOS when using the heap_4 implementation

4 Wi-SUN Configurator

When creating a new Wi-SUN sample application, a Wi-SUN Configurator is added to the project by default. It provides a configuration to the main settings of the Wi-SUN application through three panels: Application, Security, and Radio. The Wi-SUN Configurator tab is available when a project is created or can be displayed by opening */config/wisun/wisun_settings.wisunconf*.

4.1 Application Panel

The Application panel exposes multiple Wi-SUN stack settings associated with the application. It allows editing the following, among other things:

- The network name the device will try to connect to
- The network size setting
- The device's TX output power
- The unicast dwell interval

≡	Wi-SUN Configu	urator								View Manual
G	Application	Network Information								
.	Security Radio	Network Name Wi-SUN Network	0	Network Size Small 👻	0					
		Device Information MAC Address EFR32 unique MAC addr	ess by default	0	Unicast Dwell Interval	0	TX Output Power	dBm	0	
		MAC Allow/Deny List	,							
		MAC Address			Add ≡ ₊	0				
		List Type O Allo MAC Address List	w 💿 Deny		Clear List					

4.2 Security Panel

The Security panel displays the private key and certificates used by the device to authenticate itself when connecting to a Wi-SUN network. By default, it uses the Silicon Labs demonstration samples. They can be modified to use a distinct certificate infrastructure aligned with the border router certificate.

≡	Wi-SUN Config	urator	View Manual
L\$	Application	Device Private Key	
Ô	Security	BEGIN PRIVATE KEY MIGHAgEAMBMGByqGSM49AgEGCCqGSM49AwEHBG0wawIBAQQgN0ZJ70BWw1o/ZQWK fzslLkJWw4eUEMWZK5VII0aA0PahRANCAARqgQLAeOxmxkKWWZt4gzWI3jfUtgBIz	0
Ē	Radio	BVBss6GL6FYP11DYR+WeOngFRSLfP/rX3IPVIWtsY9ClovQq90NBJj+q END PRIVATE KEY	v
		Browse	
		Device Certificate	
		BEGIN CERTIFICATE MIIBzzCCAXWgAwiBAgiUV9/WCXwDqRGrLVU/JRipf/q5ARUwCgYIKoZIzJ0EAwiw HijEcMBoGA1UEAwwTV2ktU1V0IERIbW8gUm9vdCBDQTAghv0yMTAzMDEwNzQyMTha GA850Tk5MTizMTIzNTI10TK10VowJDEIMCAGA1UEAwwZV2ktU1V0IERIbW8gQm9y2GVy IFJvdRRl6jZMBM6ByqGSM49AgECCqGSM49AwHALA0JABGQBASB47GbE2D503IDN axxN952AEJMFUUgyzQYvoVg/XUNhH5Z46e0vFII8/+tfel9Uhazyl0KKI9Cr300Em P6qjgYgwgYUwDgYDVR0PAQH/BAQDAgOIMCEGA1Ud.QEB/wQXMBUGCSsGAQQBguQI AQYIKWYBBQUHAwEvLwYDVR0RAQH/BCUWI6AhBggrBgEFBQcIBKAVMBMGCSsGAQQB grdBAQQGMTIzNDU2MBBGA1UdIwQYMBaAEJNk1A73aKLuXfCs+nK8xR9NTzokMAoG CCqGSM49BAMCAGgAMCUC+IRXr50NLx2NurS2CoursPvoGbKbpLyitai3PwayX AIEAqxtUaEjjpWJUby/RsX/yXLgD9/aATj9YFTR+ZdZ1VLo= END CERTIFICATE	0
		Browse	
		CA Certificate	
		BEGIN CERTIFICATE MIBOJCCAUmgAwiBAgUBOJG108JDWdAJuqvH3REMyjjFswCgYIKoZI2j0EAwiw HjccMBoGA ULEAwwTV2ktU1V0IERIbW8gUm9vdCBDQTAgFw0yMTAyMjIw0TU5NDFa GA850Tk5MTi2MTI2NTk10VowHjECMB0GA1UEAwwTV2ktU1V0IERIbW8gUm9vdCBD QTBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABG1Mn4dd9+IVJZSEcjpFKehvRyQ t9QcIBCN2ysf+BJUFI/UBTvC3w2waFLuC+JHM+1TEEm1GLNDF7piCgqItVIJY2Bh MBIGA1UdewEB/wQIMAYBAf8CAQiwCwYDVR0PBAQDAgEGMB0GA1Uddg0WBBSTZNQ0 92Li7J3wrPpyW0fTU8GDAFBgVHSMEGDAVg0BSTZNQ092i7J3wrPpyW0fTU86 1DAVEBcsbidDD0DAf8gVHSMEGDAVgBSTZNQ092i7J3wrPpyW0fTU86	0

4.3 Radio Panel

The Radio panel is an interface to configure the radio profiles included in a Wi-SUN application. It provides a user interface to access any specified Wi-SUN FAN 1.0 or FAN 1.1 PHY. A radio button allows the user to choose between the FAN 1.0 or FAN 1.1 context.

The complete list can be filtered to help you find the right PHY configuration. An application can embed several PHYs from different regions and different specification versions.

=	Wi-SUN Config	urator											Vie	w Manual
L\$	Application	Reference	Reference PHYs											
ê	Security	₹_	▼_				Add All PH	IYs		PHY	Version	FEC		
â	Radio	Regula	egulatory Domain						~	CN - 1 - 2a	FAN 1.0		1	
		Channe	el Plan ID								FAN 1.1		/	Î
		PHY O	PHY Operating Mode						~	NA - 1 - 19	FAN 1.1		1	Î
		ID Regulatory Domain Channel Plan ID PHY Operating Mode ID				Information	*	Application's	s Default PHY	NA - 1 -	19 FAN	1.1	•	
		~	≡+	NA	1	2	2-FSK, 902.2 MHz, 50 kbps				<u> </u>]
		~	=+	NA	1	18	2-FSK, 902.2 MHz, 50 kbps, With FEC					A	pply Conf	iguration
		~	=+	NA	1	3	2-FSK, 902.2 MHz, 100 kbps		Other Custo	m Profiles (1)	0			
		~	=+	NA	1	19	2-FSK, 902.2 MHz, 100 kbps, With FEC							
		~	=+	NA	1	80	OFDM, 902.2 MHz, kbps							
		~	=+	NA	2	5	2-FSK, 902.4 MHz, 150 kbps	•						

The "Application's Default PHY" input defines the PHY that the application starts with. The default value depends on the EFR32 radio board used to create the project. For example, a BRD4163A radio board supporting the 868 MHz band defaults to the mandatory Wi-SUN PHY for the European region (that is, Wi-SUN FAN EU - 1 - 1a). On the other hand, a BRD4164A radio board supporting the 915 MHz band defaults to the North America mandatory Wi-SUN PHY (that is, Wi-SUN FAN NA - 1 - 1b). The user can always open the dropdown list and select another default PHY.

Every selected Wi-SUN PHY can be edited in the Radio Configurator by clicking the pen icon. This action opens the Radio Configurator user interface on the selected Wi-SUN PHY. Moreover, non-Wi-SUN FAN PHYs are listed under "Other Custom Profile" for information.

Radio Configurator v2										Search	Q	View Manual
	*			n in								
P Protocol Configuration		General Settings	×	Channels Overview					×			
C Channel Group 1					Chart		Ch	-hannel				
Ceneral Settings		Protocol name Protocol Configuration		Name	No.	Frequency	No.	Frequency				
Channels Overview	•	C variable name Protocol_Configuration		Channel Group 1	0	902.20 Mhz	128	927.80 Mhz				
		Select radio profile Wi-SUN FAN 1.0 Profile	*									
		Select regulatory domain NA	•									
		Select operating class 1	*									
		Select operating mode Mode1b	•									
		Customized										

A PHY can be edited in the Radio Configurator to step out of the Wi-SUN FAN configuration (change the number of channels or frequencies).

4.4 Changing the Default Wi-SUN Radio Configuration

If a Wi-SUN application needs to use a different Wi-SUN PHY, use the Radio Configurator to select another one. In the **General Settings** card, open the **Select radio PHY** list. Select a new Wi-SUN PHY in the list. Keep in mind the Wi-SUN PHY selected should match the radio board capabilities. Silicon Labs does not recommend using a Chinese Wi-SUN PHY (470 MHz) on a radio board supporting the 868 MHz band.

The Wi-SUN CLI project, the network performance application, the RCP sample application, and the border router demo embed all the Wi-SUN PHYs listed in the Radio panel list. This supports changing the PHY dynamically using the CLI interface. The other sample applications only embed a single PHY by default. In this case, the PHY change must be done before project compilation through the Wi-SUN Configurator.

5 Mode Switch

The EFR32FG25 device supports the Wi-SUN FAN stack with the Mode Switch feature for OFDM and FSK PHYs specified in the Wi-SUN PHY Specification Revision 1VA8. To use Mode Switch for the OFDM or FSK modulations with the stack, use the **Wi-SUN - SoC Border Router** project for the border router alongside the **Wi-SUN - SoC CLI** project for the node devices.

The mode switching feature can be used from the wisun_cli and wisun_brcli applications with the command:

wisun mode_switch [Mode Switch mode] [PhyModeID] [MAC Address]

The command parameters are:

- Mode switch mode:
 - 0 = Mode Switch disabled for the selected neighbor(s)
 - The [PhyModelD] parameter is ignored in this case.
 - 1 = Mode Switch enabled, with a specific PhyModeID for the selected neighbor(s)
 - When [MAC Address] is ff:ff:ff:ff:ff:ff:ff;ff, the global PhyModelD is set to [PhyModID] for all neighbors.
 - 2 = Mode Switch enabled, with the global PhyModeID for the selected (unique) neighbor
 - The [PhyModeID] parameter is ignored in this case.
 - ff:ff:ff:ff:ff:ff:ff:ff cannot be used.
- PhyModeID: PhyModeID to switch to (when applicable)
- MAC Address: MAC address of the neighbor(s). Either a single neighbor, or ff:ff:ff:ff:ff:ff:ff:ff:ff for all neighbors (when applicable)

Note: Mode Switch only applies to unicast data frames.

5.1 Fallback Mechanism

When Mode Switching fails too often with a neighbor (4 times more retries than successes), Mode Switching is disabled for this neighbor.

The following examples show Mode Switching in action:

Assume 3 neighbors:

- 1. 01:02:03:04:05:06:07:08 2. 11:12:13:14:15:16:17:18
- 3. 21:22:23:24:25:26:27:28

1- To use PhyModeld 34 (OFDM option 1, MCS2) globally (for all existing neighbors), the command is:

wisun mode switch 1 34 ff:ff:ff:ff:ff:ff:ff:ff

2- To use PhyModeID 52 (OFDM option 2, MCS4) for the first and second neighbors only, two commands are necessary:

wisun mode_switch 1 52 01:02:03:04:05:06:07:08 wisun mode_switch 1 52 11:12:13:14:15:16:17:18

3- To use the global PhyModeID for the second neighbor, the command is:

wisun mode switch 2 xx 11:12:13:14:15:16:17:18

4- To disable mode switch for the first neighbor, the command is:

wisun mode switch 0 xx 01:02:03:04:05:06:07:08

5- To disable mode switch for all neighbors using the global PhyModeID, the command is:

wisun mode switch 0 xx ff:ff:ff:ff:ff:ff:ff:ff

6 Testing and Debugging

6.1 Access Debug Traces from the Wi-SUN Stack

The Wi-SUN stack provides a logging mechanism based on the Segger RTT feature to allow a finer tracing capability. To access the Wi-SUN stack RTT traces:

- 1. Install the <u>J-Link RTT Viewer</u>.
- 2. Open the J-Link RTT Viewer.
- 3. In the Configuration panel, Connection to J-Link section, select USB.
- In the Specify Target Device list, select the connected part (for example EFR32MG12PXXXF1024). EFR32FG25 is not yet known to the RTT Viewer, so use any EFR32MG12 instead.
- 5. In the Target Interface & Speed panel, select SWD and 4000 kHz.
- 6. In the RTT Control Block panel, select Auto Detection.
- 7. Click **OK**.
- 8. If you have several boards connected, a list appears. When you need to monitor several devices, open an instance of RTT viewer per device.
- 9. Select a WSTK board running the Wi-SUN stack (border router or node).
- 10. Click **OK**.

A terminal opens and the Wi-SUN stack traces are output as shown below.

```
[DBG ] [wisun]: net_init_core: 0
[DBG ] [wisun]: sli_wisun_task_event_handler_id: 2
[DBG ] [SLRF]: sli_wisun_driver_register()
[DBG ] [SLRF]: sli_wisun_driver_register() - driver_id: 0
[DBG ] [SLRF]: rf_address_write: PHY_MAC_64BIT: 00:0d:6f:ff:fe:20:bd:95
[DBG ] [mlme]: SW-MAC driver support rf extension 50000 symbol/seconds 20 us symbol time length
[DBG ] [swm ]: Set MAC mode to IEEE 802.15.4-2011, MTU size: 127
[DBG ] [SLRF]: rf_address_write: PHY_MAC_64BIT: 00:0d:6f:ff:fe:20:bd:95
[DBG ] [wisun]: arm nwk interface lowpan init: 1
```

The application must install the 'Third Party/Segger/RTT/SEGGER RTT' component to generate RTT traces from the Wi-SUN stack. The component uses a configurable RTT trace 'channel' (0 by default).

The Trace and Debug component provides APIs to set the Wi-SUN stack traces level and filters. Visit <u>Wi-SUN Stack traces and debug</u> <u>API</u> for more information.

These logs can be used to report an issue to Silicon Labs support.

NB: RTT Viewer needs to be disconnected from a device to allow flashing again. The most convenient way to quickly disconnect is to press 'F3', flash, then press 'F2' once ready to reconnect. RTT viewer will use previous settings by default.

6.2 Export Wi-SUN Air Capture Traces to Wireshark

The Wi-SUN traces export feature requires Simplicity Studio 5.1.0 or higher. It relies on the 'Platform/Radio/RAIL Utility, PTI' component.

Simplicity Studio's Network Analyzer enables debugging of complex wireless systems on a number of Silicon Labs part families. Network Analyzer includes a partial Wi-SUN protocol analyzer (that is, the Wi-SUN payload cannot be decrypted). However, it can be used to export traces to another analyzer like Wireshark.

Beginning with GSDK 4.2.1, the PTI baud Rate has been changed from the default 1600000 to 3200000. Since all out-of-box WSTKs are set by default with 1600000, you need to change this once per WSTK in the 'Admin' tab, using 'pti config 0 efruart 3200000'. Check the results using 'pti config'. The hardware will select the closest possible baud rate and show it in the (current) section.

```
WSTK> WSTK> pti config 0 efruart 3200000
Configuration successful!
WSTK> WSTK> pti config
PTI enabled
                   Yes
PTI configured
                   Yes
    -- PTI config (current)-----
Interface
                    0
                  EFR UART
Line protocol
                  : 3230769
Bitrate
    - PTI config (stored)---
Interface
                    0
                    EFR UART
Line protocol
Bitrate
                    3200000
```

To export Wi-SUN traces with the Network Analyzer to Wireshark, install Wireshark and follow this procedure in Simplicity Studio 5:

- 1. In the Simplicity IDE perspective, Debug Adapter view, right-click an EFR32xG12 running the Wi-SUN stack.
- 2. Select Start Capture.
- 3. A Live tab opens in the Editor area. It traces packets sent and received by the Wi-SUN device.
- 4. When you have traced the communication, click Export.

S v5_workspace - Live capture stream - Simplicity Studio™						
<u>File Edit Navigate Search Project Run Window H</u> elp <u>Network</u>	c Analyzer					
┣	Ġ 🖗 🗉 🖻 🔁 🐅 🕼 🖉 📶 🖧 🥔 🕄 🗳 🔍 📄 🖅 🖧 🧩 🏌					
😰 🖉 Launcher - {} Simplicity IDE 🙏 Network Analyzer	Export network analyzer data into various file formats.					
📲 Debug Adapters 🛛 📃 🗖	▲ *Live 🛛					
🎭 💥 🗳 🗶 💥 ▼ 🔲 🖨 🕀	2 saved filters AND					
> 🎇 EFR32MG12 2400/915 MHz 19 dBm Dual Band Radio Board (BRD416	34.00 p/S					
> 🌠 Wireless Starter Kit Mainboard (BRD4001A Rev A01)	<u>0.000s</u>					
	Time:not set Real time:N/A Nodes:0 Event:none					

- 5. Under Select export format, select PCAP NG exporter,
- 6. Enter a path and a file name in which to store the trace.
- 7. In Export mode, select Wi-SUN (auto-detect protocol).
- 8. Click OK.

Select export format:	PCAP NG exporter	
PCAP NG exporter	Exports the data into the PCAPNG format.	
ISD exporter CPU usage comma-separated file DCF exit orter (format 4) Energy Profile CSV file Events comma-separated file Simple text event log Transactions comma-separated file Wireshark live process YAML exporter Text2Pcap format (obsolete, use pcapng exporter instead)	Common options for all exporters: Output file: /home/dockerUser/SharedSpace/wisun_trace.pcapng Export only the events that match the filters set in network analyzer. Exporter specific options, if any: Export mode: Wi SUN (auto-detect protocol) Defaults	6
	Cancel	K

Open the new file in Wireshark. Wireshark should automatically analyze the file as a Wi-SUN exchange. The communication is initially encrypted thanks to the Wi-SUN encryption protocol. To decrypt the communications, the GAK key and key index set information are required. They can be retrieved on the border router CLI by issuing the following command:

```
> wisun get wisun
wisun.state = started (2)
wisun.network name = "Wi-SUN Network"
wisun.phy_config_type = FAN 1.0 (0)
wisun.regulatory_domain = NA (1)
wisun.operating_class = 1
wisun.operating mode = 0x1b
wisun.fec = 0
wisun.chan plan id = 1 (unused)
wisun.phy mode id = 2 (unused)
wisun.ch0 frequency = 863100 (unused)
wisun.number of channels = 69 (unused)
wisun.channel spacing = 100 \text{ kHz} (0) (unused)
wisun.network size = small (1)
wisun.tx power = 20
wisun.unicast dwell interval = 255
wisun.broadcast interval = 1020
wisun.broadcast dwell interval = 255
wisun.channel mask =
wisun.allowed channels = "0-255"
wisun.gak1 = 76:ab:9f:ce:8a:08:b8:88:6d:fc:56:fa:0e:6f:1e:ce
wisun.lfn gak1 = 7b:92:f2:9e:68:a5:41:37:b3:5e:e7:14:04:be:37:04
wisun.mac address = b4:e3:f9:ff:fe:c5:84:93
wisun.ip addresses = [ll: fe80::b6e3:f9ff:fec5:8493 gua: :: dodagid: ::]
wisun.trace filter.000-031 = 0xfffffff
wisun.trace filter.032-063 = 0xfffffff
wisun.regulation = none (0)
wisun.rx_phy_mode_ids = 0x02,0x05,0x06,0x08,0x22,0x23,0x24,0x25,0x26
```

On Wi-SUN Border Router Linux, the GAK Key can be retrieved using wsbrd_cli or in /var/lib/wstk/network-keys:

\$ wsbrd cli status network_name: Wi-SUN Network domain: EU mode: 1a class: 1 panid: 0x3287 size: SMALL GAK[0]: eb:7b:11:dd:f0:23:c3:7e:36:64:c5:9b:5b:ed:c6:c7 GAK[1]: fc:d3:1f:a7:4c:c8:c4:0f:96:7e:d0:5d:0e:a1:01:fc GAK[2]: fc:d3:1f:a7:4c:c8:c4:0f:96:7e:d0:5d:0e:a1:01:fc GAK[3]: fc:d3:1f:a7:4c:c8:c4:0f:96:7e:d0:5d:0e:a1:01:fc GTK[0]: a8:d6:dc:7c:66:14:2f:b8:db:dd:af:fb:c2:27:0c:18 84:fd:27:ff:fe:fe:55:bd

Silicon Labs Wi-SUN devices in the network use wisun.gak1 as the GAK key. In the first capture the GAK key is 76:ab:9f:ce:8a:08:b8:88:6d:fc:56:fa:0e:6f:1e:ce and in the second capture the GAK key is eb:7b:11:dd:f0:23:c3:7e:36:64:c5:9b:5b:ed:c6:c7. The key index to use in Wireshark is 1 in both cases, for the first GAK in the list.

In Wireshark:

- 1. Click Edit.
- 2. Click Preferences...
- 3. Expand the Protocols list and select IEEE 802.15.4.
- 4. Next to Decryption Keys, click Edit.

- 5. In the Keys window, click + (plus).
- 6. Under **Decryption key** enter the GAK key, and under **Decryption key index** enter the key index (starting at 1 for GAKs, starting at 5 for LFN-GAKs).
- 7. Click **OK**.

Wireshark is now able to decrypt the traces and the higher-level protocols (ICMP, TCP, UDP...). The following example of traces shows a router pinging its border router.

348	-140463039.786897	00:0d:6f:ff:fe:20:b6:f9		0.025546000 Wi-SUN	105
349	-140463039.761292	00:0d:6f:ff:fe:20:b6:f9		0.025605000 Wi-SUN	105
350	-140463039.735743	00:0d:6f:ff:fe:20:b6:f9		0.025549000 Wi-SUN	105
351	-140463065.782571	fd00:7283:7e00:0:20d:6fff:fe	fd00:6172:6d00:0:20d:6fff:fe20:bd45	25.953172000 ICMPv6	151
352	-140463065.753483	00:0d:6f:ff:fe:20:bd:45	00:0d:6f:ff:fe:20:b6:f9	0.029088000 Wi-SUN	44
353	-140463065.332494	fd00:6172:6d00:0:20d:6fff:fe	fd00:7283:7e00:0:20d:6fff:fe20:b6f9	0.420989000 ICMPv6	151
354	-140463065.306434	00:0d:6f:ff:fe:20:b6:f9	00:0d:6f:ff:fe:20:bd:45	0.026060000 Wi-SUN	44
355	-140463065.104863	fd00:6172:6d00:0:20d:6fff:fe	fd00:7283:7e00:0:20d:6fff:fe20:b6f9	0.201571000 ICMPv6	151
356	-140463065.078796	00:0d:6f:ff:fe:20:b6:f9	00:0d:6f:ff:fe:20:bd:45	0.026067000 Wi-SUN	44
357	-140463067.250051	00:0d:6f:ff:fe:20:bd:45		1.828745000 Wi-SUN	105
358	-140463069.782009	00:0d:6f:ff:fe:20:bd:45		1.468042000 Wi-SUN	105
359	-140463073.927166	fd00:7283:7e00:0:20d:6fff:fe	fd00:6172:6d00:0:20d:6fff:fe20:bd45	3.854843000 ICMPv6	151
360	-140463073.898023	00:0d:6f:ff:fe:20:bd:45	00:0d:6f:ff:fe:20:b6:f9	0.029143000 Wi-SUN	44
361	-140463073.880264	fd00:6172:6d00:0:20d:6fff:fe	fd00:7283:7e00:0:20d:6fff:fe20:b6f9	0.017759000 ICMPv6	151
362	-140463073.854198	00:0d:6f:ff:fe:20:b6:f9	00:0d:6f:ff:fe:20:bd:45	0.026066000 Wi-SUN	44
363	-140463076.612640	fd00:7283:7e00:0:20d:6fff:fe	fd00:6172:6d00:0:20d:6fff:fe20:bd45	3.241558000 ICMPv6	151
364	-140463076.557104	fd00:7283:7e00:0:20d:6fff:fe	fd00:6172:6d00:0:20d:6fff:fe20:bd45	0.055536000 ICMPv6	151
365	-140463076.142182	fd00:6172:6d00:0:20d:6fff:fe	fd00:7283:7e00:0:20d:6fff:fe20:b6f9	0.414922000 ICMPv6	151
366	-140463076.116118	00:0d:6f:ff:fe:20:b6:f9	00:0d:6f:ff:fe:20:bd:45	0.026064000 Wi-SUN	44
367	-140463077.809511	fd00:7283:7e00:0:20d:6fff:fe	fd00:6172:6d00:0:20d:6fff:fe20:bd45	0.306607000 ICMPv6	151
368	-140463077.740374	00:0d:6f:ff:fe:20:bd:45	00:0d:6f:ff:fe:20:b6:f9	0.069137000 Wi-SUN	44

The PTI output is limited in number of bytes per messages. Packets above 1022 bytes are truncated using the WSTK Firmware 1v4p0 or later (200 bytes with earlier versions). This cannot be increased with WSTKs due to hardware limitations on BRD4001. It will be improved for Wireless Pro Kits (WPKs) once the proper firmware is available.

6.3 Connect the Wi-SUN Network to Another IP Network

Refer to the steps in AN1332: Silicon Labs Wi-SUN Network Setup and Configuration to open a backhaul connection from the Wi-SUN border router.

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