



# UG103-13: RAIL Fundamentals

---

Silicon Labs RAIL (Radio Abstraction Interface Layer) provides an intuitive, easily-customizable radio interface layer that supports proprietary or standards-based wireless protocols. RAIL is designed to simplify and shorten the development process. Developers no longer have to deal with hundreds of registers across multiple products, but can instead rely on a unified software API. RAIL, delivered through the Silicon Labs Flex SDK (Software Development Kit), also makes applications portable across Silicon Labs wireless products. RAILtest, included with the Flex SDK, supports lab evaluation as well as application development.

Silicon Labs' *Fundamentals* series covers topics that project managers, application designers, and developers should understand before beginning to work on an embedded networking solution using Silicon Labs chips, networking stacks such as EmberZNet PRO or Silicon Labs *Bluetooth*®, and associated development tools. The documents can be used as a starting place for anyone needing an introduction to developing wireless networking applications, or who is new to the Silicon Labs development environment.

## KEY POINTS

- RAIL overview
- RAIL Library description
- RAIL feature summary

## 1. Introduction

Silicon Labs is developing software and hardware products designed to meet the demands of customers as we move to an ever-connected world of devices in the home, what is often referred to as the IoT (Internet of Things). At a high level the goals of IoT for Silicon Labs are to:

- Connect all smart devices with best-in-class networking, whether with Zigbee PRO, Thread, Bluetooth Smart, or other emerging standards.
- Leverage the company's expertise in energy-friendly microcontrollers.
- Enhance established low-power, mixed-signal chips.
- Provide low-cost bridging to existing Ethernet and Wi-Fi devices.
- Enable cloud services and connectivity to smartphones and tablets that promote ease of use and a common user experience for customers.

One challenge in developing these wireless products is understanding and implementing low-level radio capabilities quickly and efficiently, so as to reduce the time to market. Product developers need an easy and future-proof method to configure the radio so they can focus on application development. To meet this need, Silicon Labs has developed RAIL. RAIL is a fundamental building block for all networking stacks developed internally by Silicon Labs, as well as by the company's customers and third-party partners. RAIL supports a diverse set of radio configurations and functionality and is one of the key underlying technologies of Silicon Labs wireless products.

The primary benefits of RAIL are:

- Implements commonly-used radio functionality, so that it does not have to be written in the application or stack.
- Eliminates the need for developers to become expert in RF register details of complex wireless SoCs.
- Simplifies code migration to new wireless ICs and the development of new stacks by providing a common radio interface layer.
- Allows lab evaluation in addition to application code development.

As RAIL has evolved and grown, it became apparent that a significant refactoring of the RAIL code was necessary to achieve the longer term goals of RAIL. As a result, Silicon Labs developed RAIL version 2.0.

The most significant changes from RAIL version 1.6 to RAIL version 2.0 are:

- Updated APIs to allow for Dynamic Multiprotocol functionality (that is, multi-PHY support).
- Unified naming across all APIs to follow a strict VerbNoun naming convention.
- A more powerful channel-based approach to configuring the radio. This adds the ability to automatically change radio configurations and the max output power on a per channel basis.
- A unified callback mechanism for all RAIL events.
- A more advanced packet receive API to allow for greater control over when to process incoming packets.

These are the motivations behind RAIL version 2.0.

### **Enable Dynamic Multiprotocol operation:**

- Support running N different radio configuration simultaneously on one chip using time slicing (where  $N \geq 2$ ).
- Allow multiple instances of RAIL to run simultaneously.
- Enable multiple priority levels to support overriding long running but low-priority tasks with higher priority tasks from other RAIL instances.
- Permit each transmit and receive operation to specify bounds on when it must execute before it is canceled and reported as missed to the calling stack.

**Note:** RAIL version 2.0 includes core features for Dynamic Multiprotocol support. Silicon Labs has initially targeted our Zigbee and Bluetooth low energy (Bluetooth LE) stacks. Support for other protocols and proprietary stacks will be available in future releases.

### **Improve usability and readability of the code:**

- Conform to Silicon Labs coding conventions and APIs.
- Streamline callbacks.
- Reduce the number of APIs.
- Simplify the way TX/RX options are specified.
- Simplify Auto-Ack functionality.

### **Improved functionality and performance:**

- Power Amplifier (PA) configurability through RAIL.
- Packet Trace Interface (PTI) configurability through RAIL.

- Support for up to three simultaneous IEEE 802.15.4 Private Area Network (PAN) identifiers, short addresses, and long addresses in the filtering logic on the EFR32 series of chips.

While the RAIL version 2.0 changes are far-reaching, many of them are largely cosmetic, and none of them should remove any previous functionality. To see this, there is a RAIL 1.x.x compatibility layer which can be combined with a RAIL version 1.x project to get it to compile with a RAIL version 2.x library. The RAIL 1.x.x compatibility layer should only be used as an example or for initial porting attempts because there are many advantages to be gained from porting fully to RAIL version 2.x.

## 2. RAIL Overview

Silicon Labs RAIL is delivered through the Silicon Labs Flex SDK and can be configured in the Simplicity Studio development environment. Developers can develop protocols that work directly with RAIL, or configure applications based on the Silicon Labs Connect stack, also in the Flex SDK. The Silicon Lab Connect stack provides a fully-featured, easily-customizable wireless networking solution optimized for devices that require low power consumption and are used in a simple network topology. For more information, see *UG103.12: Silicon Labs Connect Fundamentals*.

RAIL components in the Flex SDK are.

- The RAIL library: Provides a programming interface to radio functionality, as shown in the following figure.
- The Radio Scheduler: A system used for multiprotocol on-demand scheduling of radio events and transactions as requested by wireless stacks and applications. See *UG305: Dynamic Multiprotocol User's Guide* for more information.
- The Radio Configurator: Part of Simplicity Studio, the interface that allows developers to configure static parameters of the radio physical layer. For details see *AN971: EFR32 Radio Configurator Guide*.
- RAILtest, included with the Flex SDK, includes a serial command for each RAIL library feature, to allow scripted testing and ad hoc experimentation. RAILtest can be built with any PHY, including 802.15.4 and Bluetooth Smart. Many of the RAILtest serial commands can be used for lab evaluation.
- Other example applications: Can be used as is for evaluation and also serve as a starting point for application development.
- Documentation, delivered through Simplicity Studio.

For more information about using the example applications in the Flex SDK to begin development with RAIL, see *QSG138: Getting Started with the Silicon Labs Flex Software Development Kit for the Wireless Gecko (EFR32™) Portfolio*.

RAIL currently supports the following protocols and applications:

- Customer proprietary stack
- 'Stackless' applications (e.g. Range Test, included with the Flex SDK)
- RAILtest application for lab evaluation (included with the Flex SDK)
- Silicon Labs Connect stack

RAIL supports the Wireless Gecko (EFR32™) portfolio, primarily focused on the Flex Gecko family of 2.4 GHz proprietary wireless SoCs. Additional support for Sub-GHz and dual band hardware will be available in phases as the hardware becomes available.

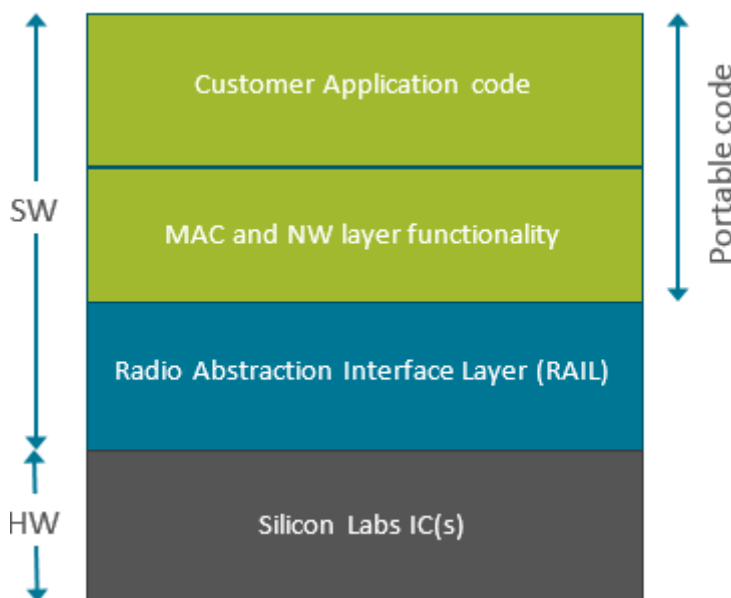


Figure 2.1. RAIL Stack Structure

### 3. RAIL Library

RAIL functionality is delivered as a library that is linked to the developer's application. The RAIL library implements the core features and runtime APIs needed to configure and control the radio. The RAIL library is the heart of the RAIL SDK.

The RAIL library works by taking intuitive and easy-to-use commands from the RAIL API and translating them into register-level code used to control radio and communications functions. The API commands remain constant across ICs. The changes in the underlying code are transparent to the developer or system tester. This also allows developers to create multiple stacks for different products quickly, as they are always presented with a similar software radio interface. RAIL provides the foundation upon which developers can implement their own MAC layer and network layer functionality.

- General Radio Operation
- Channel definition and selection
- Output power configuration
- Transmit
- Clear Channel Assessment before Transmit
- Scheduled Transmit
- Energy Detection
- Receive
- Packet Filtering
- Calibration
- CW (Carrier Wave) Transmission
- Modulated Transmission
- RFSense configuration as wake source

Where possible, all features currently implemented for the EFR32 will be implemented for future ICs, allowing for easy migration of all RAIL-based applications.

Planned RAIL features include:

- Duty Cycled Receive operation
- Frequency Hopping Receive operation
- Acknowledgements
- Antenna Diversity
- 802.15.4 support

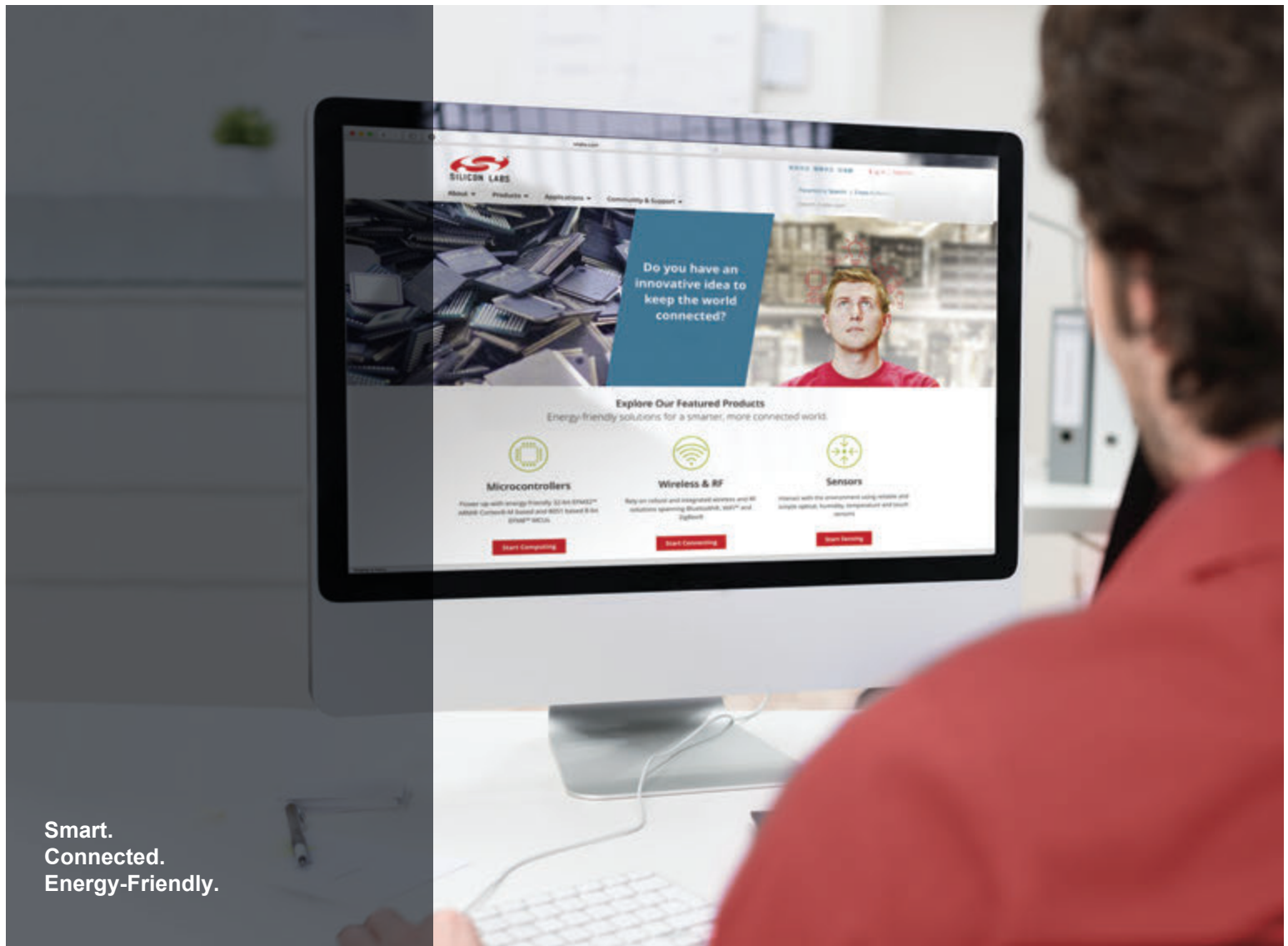
## 4. RAIL Features

RAIL includes the following features in software, as well as many others.

RAIL Highlights	Description
Address Filtering	Each packet contains two fields that will be examined during filtering for addresses. Each field may have up to four addresses. Each address may have a size of 1, 2, 4, or 8 bytes <ul style="list-style-type: none"><li>• Support for broadcast addresses. This is not 15.4 address filtering.</li><li>• Devices can enable or disable address filtering based on a frame type (e.g: ACK).</li></ul>
CCA (Clear Channel Assessment)	Supports two common medium access methodologies, which delay transmission until the channel is clear: <ul style="list-style-type: none"><li>• CSMA-CA (Carrier Sense Multiple Access with Collision Avoidance) -- based on IEEE 802.15.4 specification.</li><li>• LBT (Listen Before Talk) -- based on ETSI EN 300 220-1 specification.</li></ul>
RSSI (Received Signal Strength Indicator) read	Ability to read RSSI manually.
Multiprotocol support	<ul style="list-style-type: none"><li>• N different radio configuration are allowed simultaneously on one chip using time slicing (where <math>N \geq 2</math>).</li><li>• Multiple instances of RAIL can run simultaneously.</li><li>• Multiple priority levels support overriding long running but low-priority tasks with higher priority tasks from other RAIL instances.</li></ul>
Scheduled transmission	Defer transmission to an absolute time after API call.
RFSense support	Works with the EFR32 RFSense feature, the ability to sense the presence of RF energy above -20 dBm within either or both the 2.4 GHz and Sub-GHz bands, and trigger an event if that energy is continuously present for certain durations of time. <ul style="list-style-type: none"><li>• Enable / Disable RFSense.</li><li>• Use RFSense as a wakeup mechanism.</li></ul>
Calibration support	Calibration related to temperature changes during RX and Image Rejection.
Improve memory interface for TX and RX packets	A dynamic and efficient method to handle TX and RX buffers.
Symbol-based timer	Code can start and stop a timer that will fire based on symbol time.
RAIL timer	$\mu$ sec granularity with a RAIL timer (used to timestamp RX, TX etc.).
RAIL API to get entropy from radio	Radio can be used as a source to generate a random number for security.
Compiler support	IAR Embedded Workbench for ARM 7.80 or later. GCC (GNU Compiler for ARM)

## 5. Next Steps

See *QSG138: Getting Started with the Silicon Labs Flex Software Development Kit for the Wireless Gecko (EFR32™) Portfolio* for instructions on how to install Simplicity Studio and get started using RAIL from the Flex SDK.

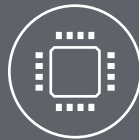


Smart.  
Connected.  
Energy-Friendly.



**Products**

[www.silabs.com/products](http://www.silabs.com/products)



**Quality**

[www.silabs.com/quality](http://www.silabs.com/quality)



**Support and Community**

[community.silabs.com](http://community.silabs.com)

#### 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 and limitation to product information, specifications, and descriptions herein, and does not give warranties as to the accuracy or completeness of the included information. Silicon Labs shall have no liability for the consequences of use of the information supplied herein. This document does not imply or express copyright licenses granted hereunder to design or fabricate any integrated circuits. The products are not designed or authorized to be used within any Life Support System 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 products 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.

#### Trademark Information

Silicon Laboratories Inc.®, Silicon Laboratories®, Silicon Labs®, SiLabs® and the Silicon Labs logo®, Bluegiga®, Bluegiga Logo®, Clockbuilder®, CMEMS®, DSPLL®, EFM®, EFM32®, EFR, Ember®, Energy Micro, Energy Micro logo and combinations thereof, "the world's most energy friendly microcontrollers", Ember®, EZLink®, EZRadio®, EZRadioPRO®, Gecko®, ISOmodem®, Micrium, Precision32®, ProSLIC®, Simplicity Studio®, SiPHY®, Telegesis, the Telegesis Logo®, USBXpress®, Zentri 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. 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

<http://www.silabs.com>