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 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 RAIL (Radio Abstraction Interface Layer) 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.

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 2.x.

The most significant changes from RAIL 1.6 to RAIL 2.x are:

- Updated APIs to allow for Dynamic Multiprotocol functionality.
- 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 2.x.

Enable Dynamic Multiprotocol operation:

- Support multiple radio configurations simultaneously on one radio using time slicing.
- Allow multiple instances of RAIL.
- Enable multiple priority levels to support overriding long-running but low-priority radio operations with higher priority radio operations from other RAIL instances.
- Permit each transmit and receive radio operation to specify bounds on when it must execute. If it cannot be scheduled within these bounds, it is canceled and reported as dropped to the calling stack.

Note: RAIL 2.x 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 may be added 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 2.x changes are far-reaching, many of them are largely cosmetic, and none of them should remove any previous functionality. See *AN1113: Porting RAIL Applications to RAIL Version 2.x* for more information.

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.

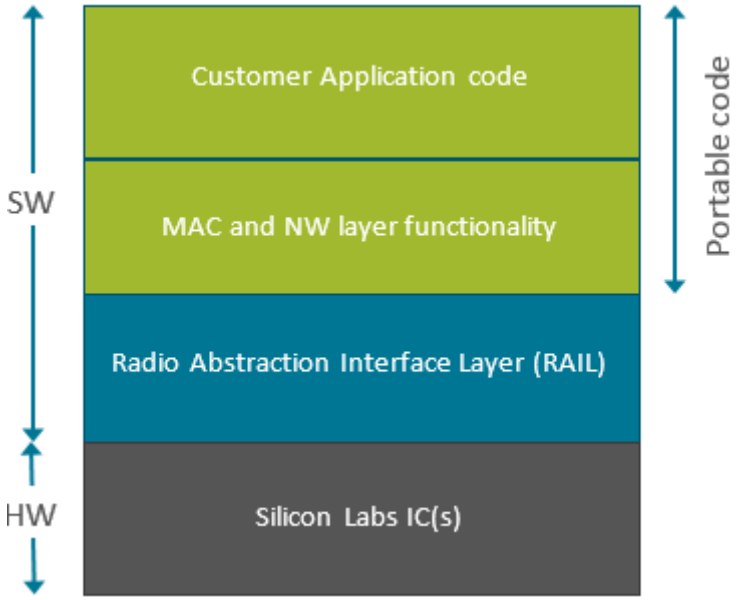


Figure 2.1. RAIL Stack Structure

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

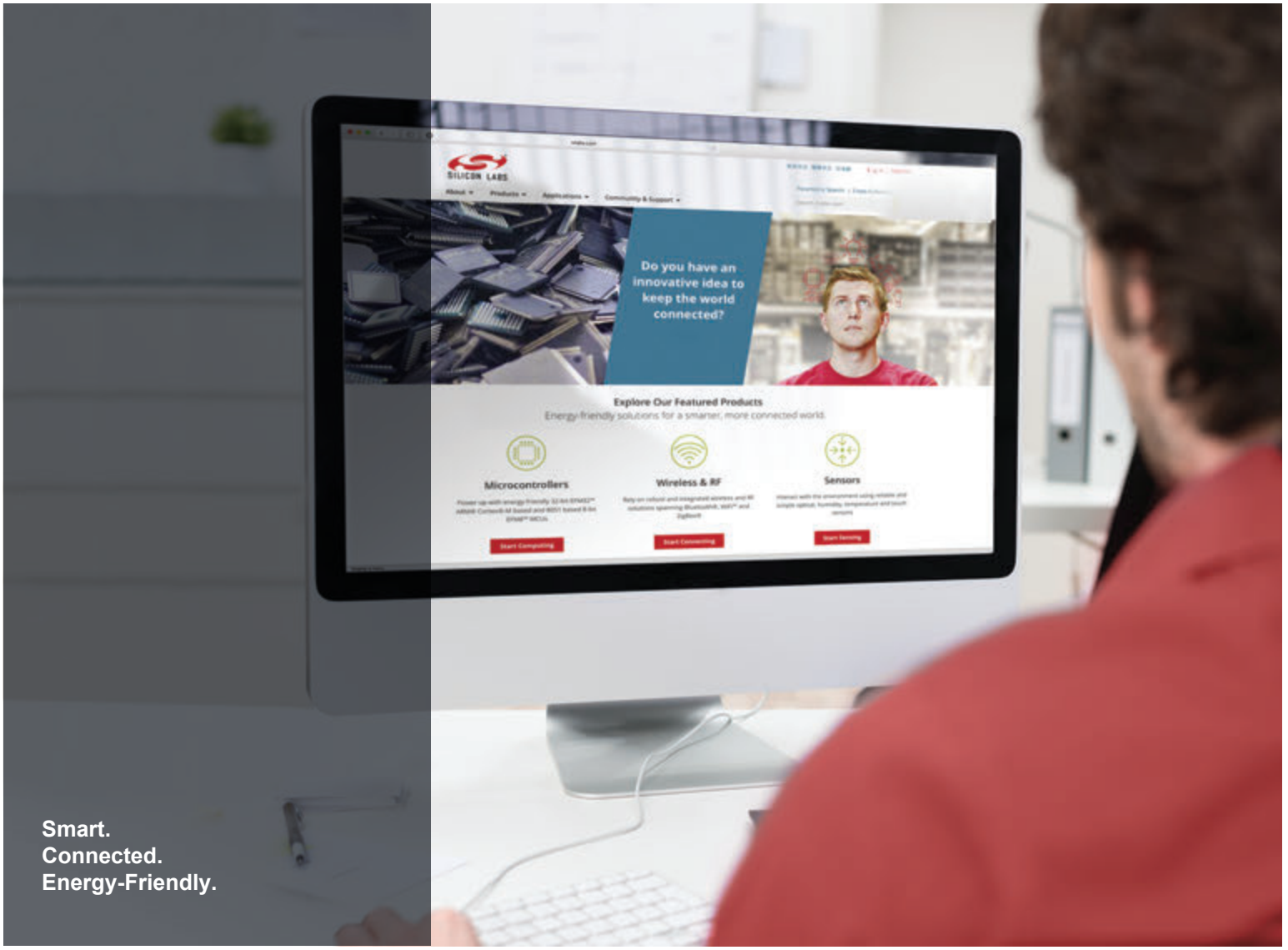
4. RAIL Features

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 • Support for broadcast addresses. This is not 15.4 address filtering. • Devices can enable or disable address filtering based on a frame type (e.g: ACK).
CCA (Clear Channel Assessment)	Supports two common medium access methodologies, which delay transmission until the channel is clear: • CSMA-CA (Carrier Sense Multiple Access with Collision Avoidance) -- based on IEEE 802.15.4 specification. • LBT (Listen Before Talk) -- based on ETSI EN 300 220-1 specification.
RSSI (Received Signal Strength Indicator) read	Ability to read RSSI manually.
Multiprotocol support	• N different radio configuration are allowed simultaneously on one chip using time slicing (where N >= 2). • Multiple instances of RAIL can run simultaneously. • Multiple priority levels support overriding long running but low-priority tasks with higher priority tasks from other RAIL instances.
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. • Enable / Disable RFSense. • Use RFSense as a wakeup mechanism.
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	µsec 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.



Products
www.silabs.com/products

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

SILICON LABS®

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

http://www.silabs.com