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.

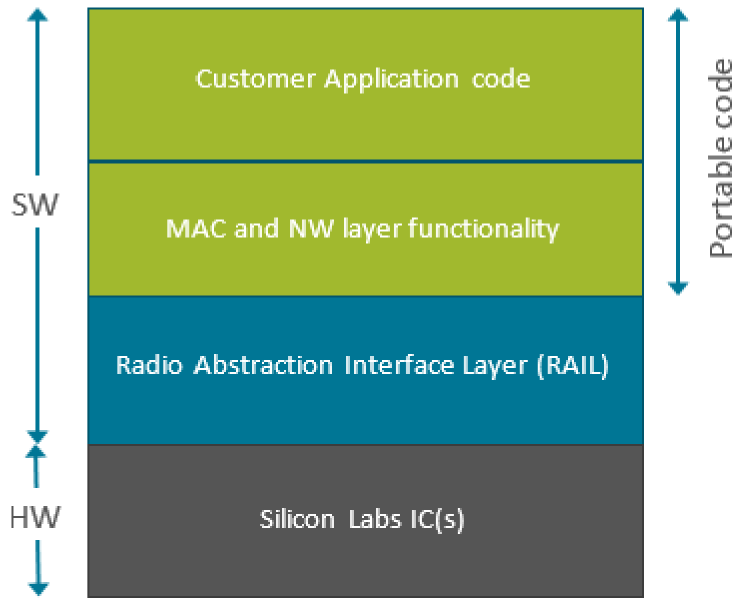


Figure 2.1. RAIL Stack Structure

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 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 configuration
- Channel definition and selection
- Output power configuration
- Transmit
- Scheduled Transmit
- Clear channel assessment before transmit
- Energy Detection
- Receive
- Scheduled Receive
- Packet address filtering
- Chip-specific radio calibration
- Automatic state transitions
- Support for acknowledgement (ACK) packets
- Receive antenna diversity on supported platforms
- IEEE 802.15.4-specific helper APIs
- Bluetooth LE-specific helper APIs
- Entropy collection from the radio
- A multiprotocol scheduler on supported platforms
- CW (Carrier Wave) transmission
- Modulated PN9 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.

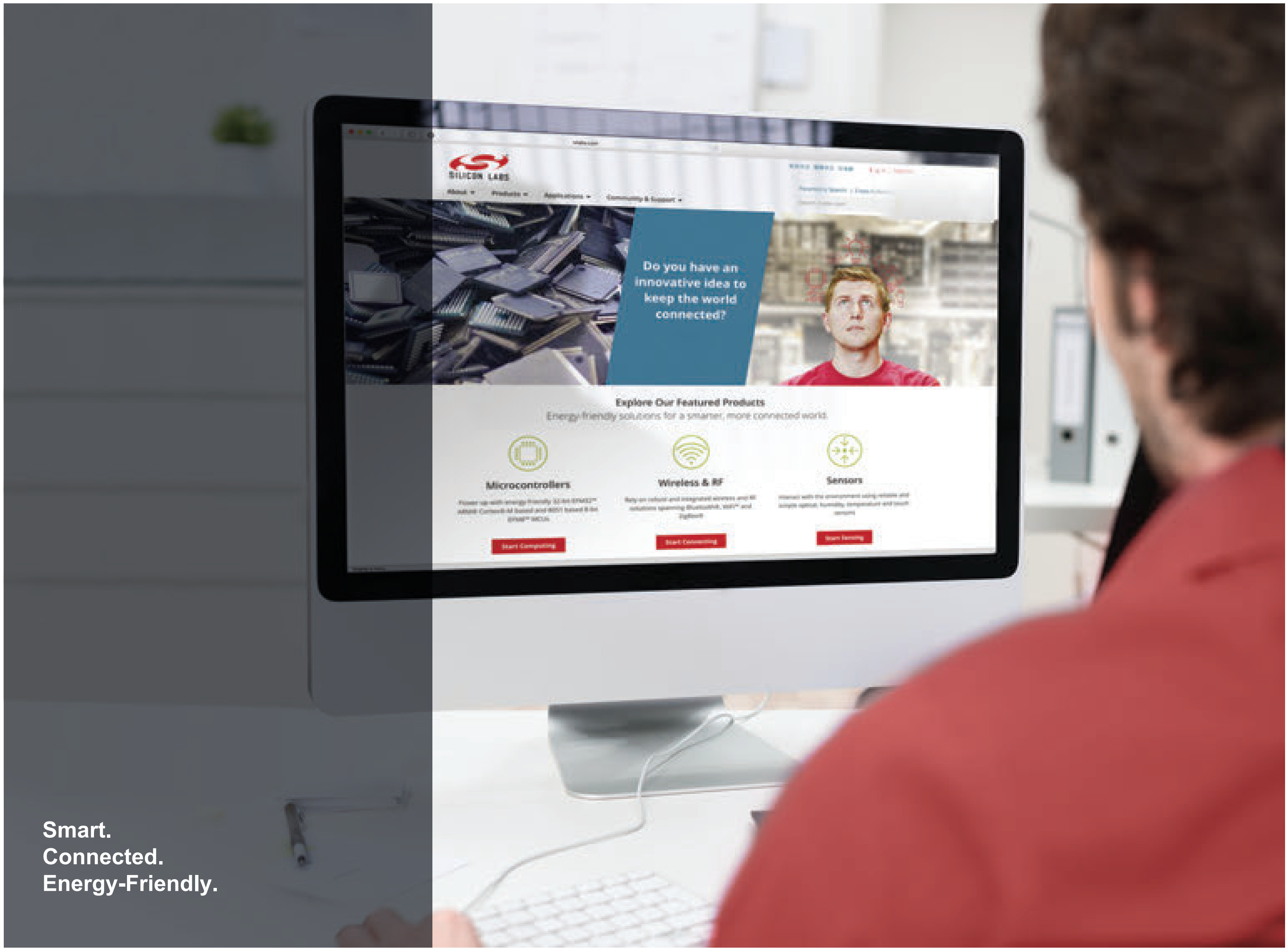
4. RAIL Features

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 with the same size where that size is up to 8 bytes. • Addresses can be enabled and configured at runtime. • 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 instantaneous and average RSSI values from hardware.
Multiprotocol support	Separate library with a radio scheduler that can manage requests from different radio configurations on the same chip. • Supports time-slicing N different radio configurations where N >= 2. • Allows for different priority tasks from the different configurations and manages them such that the highest priority is always completed. • Allows for tasks to be moved around if the timing isn't critical for most robust network.
Scheduled transmit	Transmit at an absolute or relative time in the future.
Scheduled receive	Start receiving at an absolute or relative time in the future with an optional timeout after that start time.
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. This will 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	Chip-specific calibrations. Commonly these include recalibrating for large temperature changes while in receive and image rejection calibration for increased sensitivity.
Auto ACK	Fixed transmission of a response packet after some delay to every successful receive. This can be a constant payload or loaded by the user any time before the transmit is supposed to begin. For IEEE 802.15.4 this is handled automatically to improve performance.
Receive antenna diversity	Allows toggling between multiple antennas in receive mode and selecting the one with the strongest signal to improve sensitivity. Note that this is only supported on certain chips and PHY configurations.
Automatic state transitions	Configurable transitions to go from one state to the next based on successful or unsuccessful transmit and receive operations. These support an optional delay as well to allow for consistent timings from one state to the next.
IEEE 802.15.4 and Bluetooth LE helper APIs	Helpers for implementing an IEEE 802.15.4 or Bluetooth LE networking stack. These provide special modes and optimizations in hardware for these specific modes.
RAIL timer	Timer with μsec granularity for triggering RX/TX, timestamping packets, or general RAIL timing.
RAIL API to get entropy from radio	Radio can be used to collect entropy from receive chain noise.
Compiler support	IAR Embedded Workbench for ARM and 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

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, 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. All other products or brand names mentioned herein are trademarks of their respective holders.

SILICON LABS

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

http://www.silabs.com