UG103.13: Application Development Fundamentals: RAIL

Silicon Labs RAIL (Radio Abstraction Interface Layer) provides an intuitive, easily-customizable radio interface layer that is designed to support 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 also makes applications portable across Silicon Labs wireless products. RAILtest, included with the RAIL SDK, supports lab evaluation as well as application development.

Silicon Labs' *Application Development 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 Smart, 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 FEATURES

- RAIL overview
- RAIL SDK components, including a core library of features
- Example applications, including RAILtest
- RAIL features

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 provides an intuitive, easily-customizable radio interface layer that is designed to support proprietary or standards-based wireless protocols.

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.

2 RAIL Overview

The Silicon Labs RAIL SDK consists of several components.

- The RAIL library: Provides a programming interface to radio functionality, as shown in Figure 1.
- The Radio Configurator: Part of Simplicity Studio, the interface that allows developers to configure static parameters of the radio physical layer.
- Sample applications: Can be used as is for evaluation and also serve as a starting point for application development.
- Documentation.

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.

RAIL currently supports the following protocols and applications:

- Customer proprietary stack
- 'Stackless' applications (e.g. Range Test, included with the RAIL SDK)
- RAILtest application for lab evaluation (included with the SDK)
- (coming in 2016) 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.

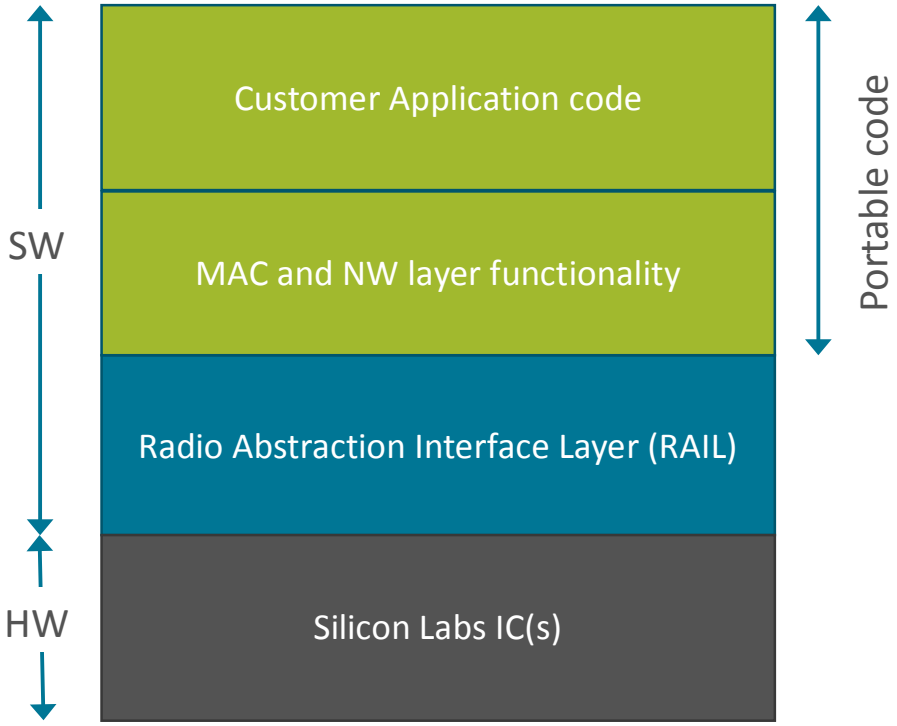


Figure 1. RAIL Stack Structure

3 RAIL SDK Components

The Silicon Labs RAIL SDK includes the RAIL library, the Radio Configurator, sample applications, and documentation. The RAIL SDK is installed alongside and registered inside of the Simplicity Studio development environment.

3.1 RAIL Library

RAIL functionality is delivered as a library that you link to your 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 API is documented in the online API reference as well as other documents installed with the stack or available through Simplicity Studio. The RAIL library supports APIs for:

- 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

3.2 RAIL Sample Applications

The RAIL SDK includes example application code and related documentation to demonstrate the capabilities of the device and the RAIL library. These examples are provided as source code to offer a starting point for application development. The following examples are included in the current release. Additional examples are planned for future releases.

3.2.1 RAILtest

RAILtest is a general test tool for the RAIL library. RAILtest is developed by the core engineering team working on the RAIL library. As each RAIL library feature is implemented, a RAILtest serial command is added 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.

RAILtest includes commands to:

- Transmit and receive packets.
- Schedule transmits at a specific time in the RAIL timebase.
- Configure RAIL address filtering to receive only specific packets.
- Enable CCA mechanisms (CSMA/LBT) to validate that a channel is clear before transmit.

- Set a timer callback in the RAIL timebase to see how the RAIL timer API works.
- Change the transmit channel within the current configuration's band.
- Change the transmit power level.
- Enable RF energy sensing of specified duration across the 2.4 GHz and/or Sub-GHz bands, and sleep to wake on this event.
- Output a continuous unmodulated tone for debugging.
- Output a continuous modulated PN9 stream for debugging.
- Enter into direct mode where data can be sent and received using asynchronous GPIOs as input and output.

3.2.2 Other Example Applications

- Range Test: Enables over-the-air range testing between two devices customized with user-defined parameters. Range Test is designed to be run on the Silicon Labs WSTK hardware without the need for commands from a host computer. This capability allows for mobility during range testing activities.
- Bidirectional link: Enables a simple point-to-point link between two devices. An additional example demonstrates the use of ACKs.
- Using Energy Modes with RAIL: Demonstrates how to use energy modes in your RAIL-based application.

3.3 Radio Configurator

Static parameters such as carrier frequency band, modulation scheme, bit rate, and packet format are set through the Radio Configurator in Simplicity Studio. The Radio Configurator translates requested RF parameters into optimal chip configuration. The RAIL library manages the configuration of the radio using the settings determined by the Radio Configurator. See *EFR32 Radio Configurator Guide* (AN971) for more details.

3.4 RAIL Documentation

RAIL documentation includes:

- Development Environment online documents:
  - API descriptions
  - Readme
  - Release Notes
- *Getting Started with Silicon Labs RAIL on the Wireless Gecko (EFR32) Portfolio* (QSG121)
- *Flex Gecko 2.4 GHz, 20dBm Range Test Demo User's Guide* (UG147)
- *EFR32 Radio Configurator Guide* (AN971)
- *EFR32 RF Evaluation Guide* (AN972)

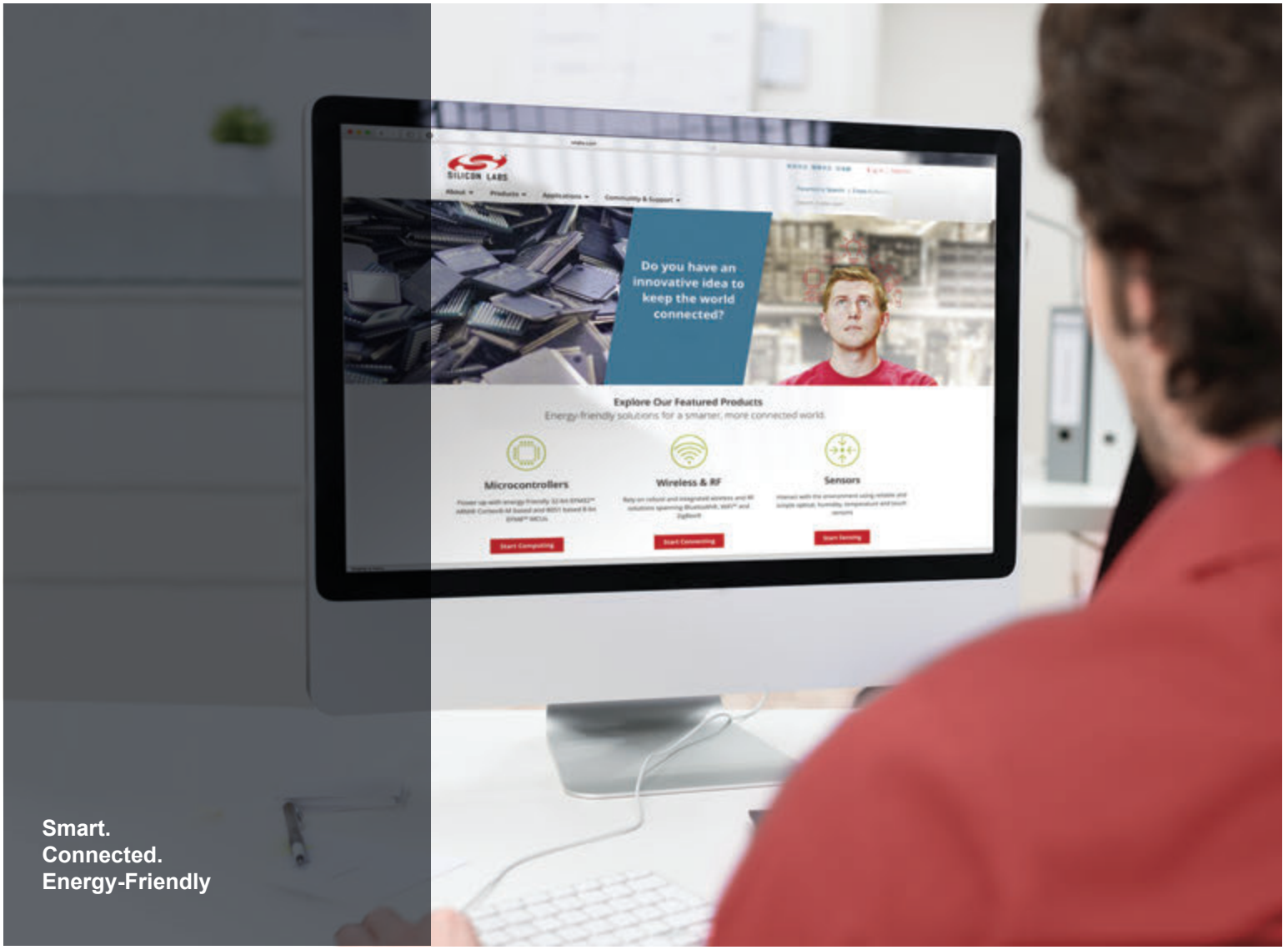
4 RAIL Features

The initial RAIL release (v1.0) includes the following features in software.

RAIL 1.0 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.
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 7.30 or later. GCC (GNU Compiler for ARM)

5 Next Steps

See *Getting Started with Silicon Labs RAIL on the Wireless Gecko (EFR32) Portfolio* (QSG121) for instructions on how to install and get started using the RAIL 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