UG103.16: Multiprotocol Fundamentals

This document discusses the ways in which more than one protocol can be used on a single chip. It discusses considerations when selecting protocols for multiprotocol implementations. It also describes the basics of the Silicon Labs Radio Scheduler, a required component of a dynamic multiprotocol solution.

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

- Multiprotocol implementation modes
- Core requirements
- Protocol considerations
- Overview of the Radio Scheduler

1. Introduction

Silicon Labs is developing 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 the devices in the home with best-in-class mesh networking, whether with Zigbee PRO or other emerging standards.
- Leverage the company's expertise in low-power, constrained devices.
- 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.

Achieving all of these goals will increase adoption rates and user acceptance for IoT devices in the Connected Home.

In general, implementing more than one protocol on a single device allows for:

- Cost savings: A single device can perform more than one function.
- Space savings: End user product packaging can be smaller and simpler when protocols can share a single radio.
- Energy savings: The number of devices on a network is reduced.

This document describes the four different ways to implement multiprotocol devices and the infrastructure requirements for an effective implementation. It discussed some aspects of protocol operation to be considered when selecting a protocol for a multiprotocol implementation. Finally it summarizes the operation of the Silicon Labs Radio Scheduler, an essential component of a dynamic multiprotocol implementation.

2. Types of Multiprotocol Implementations

Multiple protocols can be implemented on a single device in four different ways:

- Programmable: A device that is compatible with more than one protocol is programmed with one of the protocols in manufacturing.
- Switched: The application switches between protocols using a bootloader.
- Dynamic: The application time-slices between two protocols.
- Concurrent: The application runs both protocols on the same RF channel of the single radio.

The following sections discuss these implementations in more detail.

2.1 Programmable Multiprotocol

A programmable multiprotocol implementation requires a chip with the infrastructure to support more than one protocol. Applications are developed in different protocols, such as Zigbee and Thread. Each chip is programmed with a single application in manufacturing. The customer achieves cost savings by being able to use a single chip for multiple applications and purposes.

2.2 Switched Multiprotocol

In switched multiprotocol, a device is initially programmed with one protocol, and then uses a bootloader to switch to another protocol at some point in the future. Switched multiprotocol has two primary use cases.

Future Proofing: Device manufacturers may need to sell their devices into different protocol environments, or may wish to plan for an environment changing over time.

Commissioning by Smartphone: Device manufacturers who are committed to a single protocol environment, such as Zigbee Home Automation, may wish to make the process of adding a new device more secure and user friendly. In this case the device is initially programmed with a Bluetooth commissioning application. The Home Gateway / Trust Center is also programmed with a Bluetooth application, and the customer uses a commissioning app on their smartphone. Using the smartphone app they can join a new device onto the network, set up a pairing with other appropriate devices in the network, then switch over to the network protocol. Each wireless network protocol has its own mechanisms for joining and pairing devices, but all can be accommodated with this mechanism.

An effective switched multiprotocol implementations requires:

- A multiprotocol-enabled platform with sufficient memory.
- A common code infrastructure, such as low-level MCU and radio drivers.
- A consistent API to use the radio, such as that provided by Silicon Labs RAIL.
- A cross-compatible bootloader.

See *UG267: Switched Multiprotocol User's Guide* for more information about Silicon Labs implementation of Switched Multiprotocol, and *UG103.13: RAIL Fundamentals* for more information about Silicon Labs RAIL.

2.3 Dynamic Multiprotocol

In a dynamic multiprotocol implementation, two protocols run concurrently but the application time-slices the radio and rapidly changes radio configurations, such as channel, to enable different wireless protocols to operate reliably at the same time. In time-slicing, the software schedules tasks based around their priority and minimum duration and will default to a background listen task between scheduled tasks. The scheduler can even interrupt or delay a lower priority task if a higher priority task is scheduled. .

A dynamic multiprotocol implementation allows a device to perform multiple simultaneous functions. For example, an end user can use a smartphone app working with a Bluetooth application on the device to control the device or perform diagnostics, while the device remains connected to its Zigbee network routing packets for other Zigbee devices.

In addition to a multiprotocol-enabled platform with sufficient memory, common code infrastructure, and common radio interface, an effective dynamic multiprotocol implementation requires:

- An RTOS to support task switching and resource sharing.
- A radio scheduler to manage time-slicing. See section 4. Radio Scheduler for background on the Silicon Labs radio scheduler.

See *UG305: Dynamic Multiprotocol User's Guide* for detailed information about Silicon Labs dynamic multiprotocol implementation.

2.4 Concurrent Multiprotocol

In a concurrent multiprotocol implementation, two protocols not only run concurrently but also share the same radio channel. For example, a Zigbee/Thread gateway or controller device could manage the transition of the network from one protocol to another, or could manage both a Zigbee-based and a Thread-based network at the same time. The multiprotocol device always controls what channels the networks use. In the case of two separate networks, they must be able to accommodate reduced bandwidth either by scaling their traffic or limiting the number of devices on the network.

3. Protocol Considerations

It is important to recognize that a single radio cannot receive or transmit two packets for two different protocols simultaneously. To share a single radio, both protocols have to accept they will not have 100% use of the radio. They must therefore be able to survive loss of radio without significantly decreasing performance or losing application messages. “Significantly” in this case means that the application is working outside of its intended operational parameters, for example message latency or message loss is higher than acceptable.

While some protocols and applications will have strict timing requirements for packet transmission, in most cases the critical area of concern with multiprotocol is the receipt of incoming packets from the networks.

In order for a protocol to be a good candidate for a dynamic or concurrent multiprotocol implementation, the following considerations apply.

- Neither protocol can take over the radio for long periods (longer than a few milliseconds at a time) under normal operating conditions (that is except for commissioning, firmware upgrades, and so on).
- One or both protocols must have a robust mechanism for managing loss of incoming packets, such as MAC retries.
- The use case calls for tightly constrained radio use for one or both protocols, for example transmit only, low duty cycle.
- One or both protocols must have very short packets and/or a short time required on the radio;
- One or both protocols must implement strict time slots / connection intervals; and/or both protocols must use the same radio PHY (e.g. IEEE 802.15.4) and channel.

Note: When a radio has to switch between two different PHYs, it 'disappears' from one network or another. For protocols where the device might be the 'parent' of a sleepy end device, if the parent device is not available when the sleepy end device wakes up to send a message, regular dependency on retries will impact the sleepy end device’s battery life.

3.1 Dynamic Multiprotocol Example

Bluetooth LE and Zigbee are suitable protocols for a dynamic multiprotocol implementation. Because of the low duty cycle for Zigbee traffic and the retry mechanisms in the Zigbee networking stack, a Zigbee Router can switch its radio to some other frequency/protocol for short periods without dropping any messages at an application level.

Bluetooth LE radio usage can be predicted and planned in advance. Bluetooth beacons are quite short packets, usually up to 30 bytes. The radio only needs about 1 ms to transmit the beacon and the interval between beacons is typically no shorter than 100 ms, thus providing a duty cycle of just 1%. This means that the radio can devote at least 99% of its time to the main Zigbee network.

3.2 Concurrent Multiprotocol Example

Zigbee and Thread are one example of suitable protocols for a concurrent multiprotocol implementation.

The multiprotocol device can control which IEEE 802.15.4 channel it is operating on, to connect to other Zigbee and Thread devices. It only has a single IEEE 802.15.4 MAC/PHY to listen to/send on both networks concurrently, meaning that switching radio PHYs is not required.

The multiprotocol device can manage incoming packets from either network by filtering both PAN IDs, directing to the appropriate networking stack

Operational constraints include:

The multiprotocol device must control selection of the same IEEE 802.15.4 channel for both Zigbee and Thread, which means that it is most likely a Zigbee Coordinator / Thread Leader, in practice the Gateway/Controller in both networks.

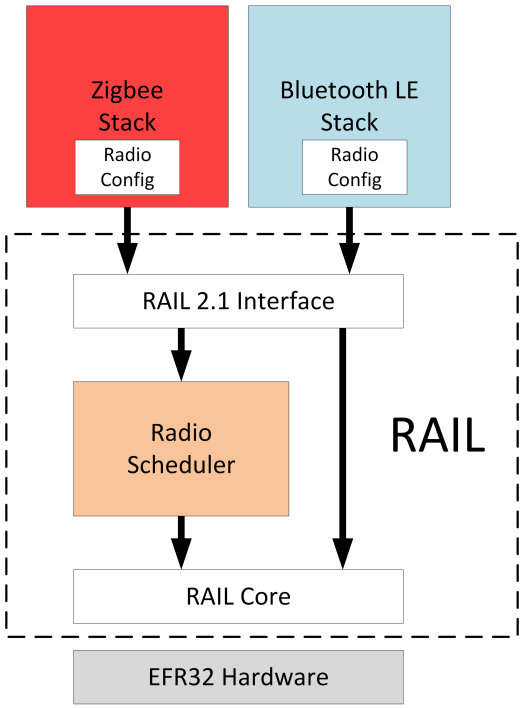
The device cannot be required to receive Zigbee and Thread packets simultaneously. However, because of MAC retries, this should not be a limitation in most use cases.

The combined duty cycle of the Zigbee and Thread traffic to/from the device must not exceed what would normally be tolerated by a single Zigbee or Thread device.

4. Radio Scheduler

A radio scheduler is an essential component of a dynamic multiprotocol implementation. It is a system for on-demand scheduling of radio tasks as requested by wireless stacks and the manufacturer's applications. This section introduces basic radio scheduler concepts. See *UG305: Dynamic Multiprotocol User's Guide* for details, including examples, of the Silicon Labs radio scheduler operation.

The Silicon Labs Radio Scheduler is part of the RAIL library. It operates above the radio hardware and below the RAIL API, as shown in the following figure.



Different radio events in each protocol may be more or less important, or more or less time sensitive, depending on the situation. The Radio Scheduler can take those into account when making decisions about conflicts and how to adjudicate them. The Radio Scheduler uses the following concepts:

Radio Operation: A specific action to be scheduled. A radio operation has both a radio configuration and a priority. Each stack can request that the radio scheduler perform three radio operations:

- Background receive: Continuous receive, intended to be interrupted by other tasks
- Scheduled receive: Receive at a future time with a minimum duration
- Scheduled transmit: Transmit at a future time with a minimum duration

Radio Config: Determines the state of the hardware that must be used to perform a radio operation.

Priority: Each operation from each stack has a default priority. An application can change default priorities.

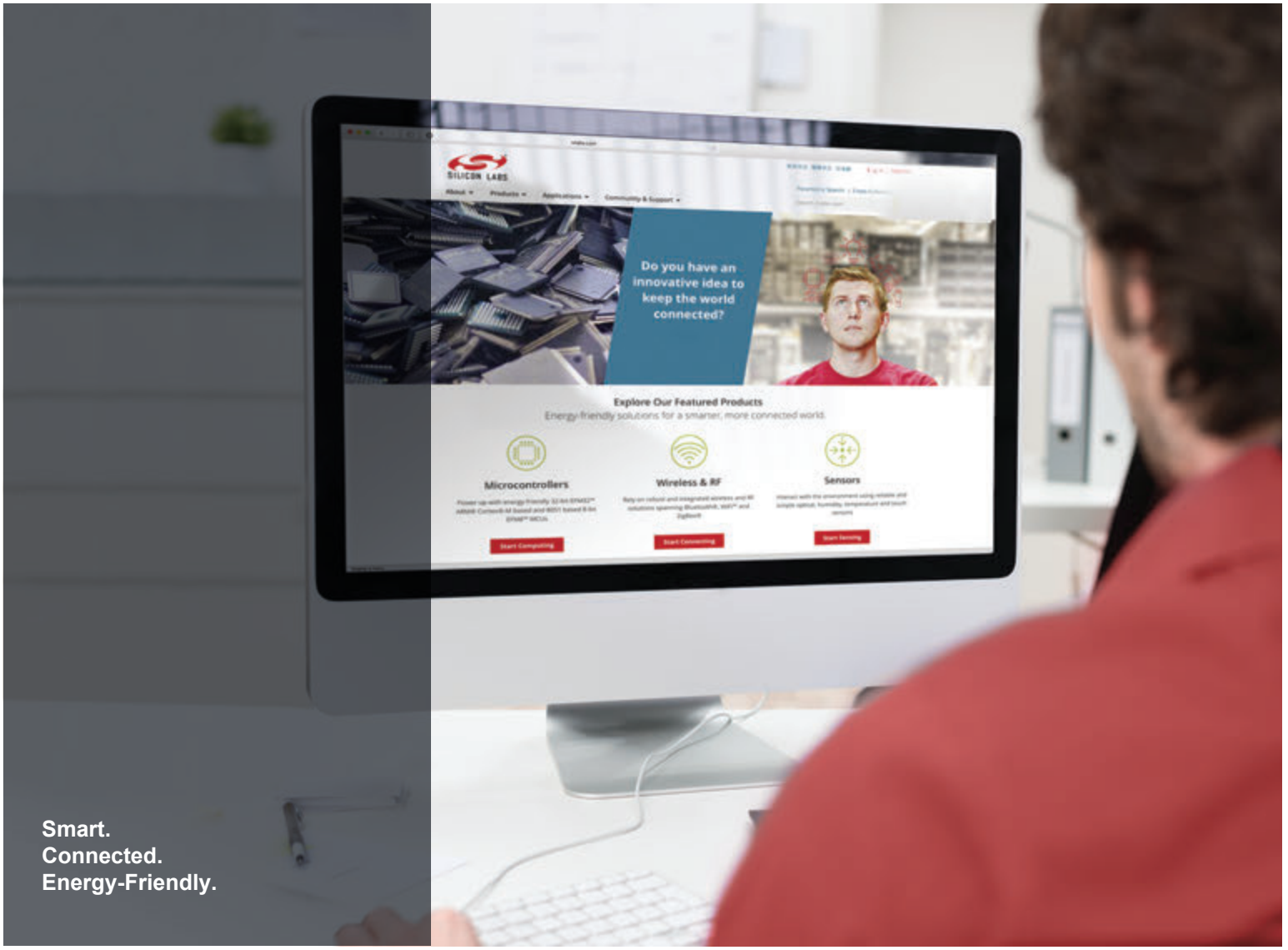
Slip Time: The maximum time in the future when the operation can be started if it cannot begin at the requested start time.

Yield: A stack must voluntarily yield at the end of an operation or sequence of operations, unless it is performing a background receive. Until the stack yields, the scheduler will not scheduler lower priority tasks.

The Radio Scheduler can interrupt a scheduled radio operation if a higher priority task conflicts with it. This could occur in the following two circumstances:

1. A scheduled radio operation takes longer than expected and the corresponding stack does not yield before the higher priority radio operation must start.
2. A higher priority radio operation has just been scheduled to occur in the future and conflicts with a lower priority operation already scheduled.

Certain long-lived radio operations can have an outsized impact on the correct operation of the product. The application may need to coordinate these tasks between the protocols. If the application does not then the radio scheduler priorities will take precedence. This can result in the task being interrupted prematurely.



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