AN1244: EFR32 Migration Guide for Proprietary Applications

This document presents a high-level collection of elements that differentiate Wireless MCU EFR32 generations from each other, as well as more detailed descriptions of features exclusive to the newest EFR32 generations. This information is designed to support those considering migrating proprietary applications from one EFR32 generation to another.

Device Support

Proprietary is supported on all EFR32FG devices. For others, check the device's data sheet under Ordering Information > Protocol Stack to see if Proprietary is supported. In Proprietary SDK version 2.7.n, Connect is not supported on EFR32xG22.

KEY POINTS

- Reviews the differences across generations that can impact proprietary applications
- Describes an enhancement to RF Sense now available on the EFR32xG22
- Introduces EM1P, a new radio-friendly low-power mode on the EFR32xG22

1. Introduction

This application note uses the following terms to classify devices in accordance with the EFR32 device datasheets:

- Device Configuration: EFR32xG1, EFR32xGx1, EFR32xGx2, EFR32xGx3, EFR32xGx4, and EFR32xGx5 devices
- Series: EFR32xG1x (including EFR32xG1 devices) and EFR32xG2x devices
- Device Family: EFR32MGxx, EFR32BGxx, EFR32FGxx, and EFR32ZGxx devices

In this document, device generations are defined as the combinations of the device configuration and series. The complete list of EFR32 device generations is: EFR32xG1, EFR32xG12, EFR32xG13, EFR32xG14, EFR32xG21, EFR32xG22, EFR32xG23, EFR32xG24, and EFR32xG25.

For more information on the existing combinations of EFR32 generations and families, visit silabs.com.

This application note discusses differences among the Series 1 and Series 2 EFR32 Wireless MCU generations that are relevant to proprietary wireless applications. “Proprietary” refers to applications based on RAIL (Radio Abstraction Interface Layer) or Connect, and typically implementing a **custom** radio configuration not defined by any wireless protocol standard. The target audience for this perspective consists of developers who are migrating existing EFR32 proprietary wireless applications to alternative EFR32 devices. This insight is also of interest to developers of new EFR32 proprietary solutions. However, this document is not intended to guide the migration of existing proprietary applications from non-EFR32 Silicon Labs wireless devices (EZRadioPRO, Si10xx, EZR32xG, and so on).

For developers migrating from EFR32 Series 1 to Series 2, AN0918.2: Series 1 to Wireless Gecko Series 2 Compatibility and Migration Guide offers essential and extensive guidance for this endeavor from a platform perspective. Proprietary wireless applications (on any EFR32 device) additionally rely on RAIL or Connect as well as the Radio Configurator (a tool within Simplicity Studio) to manage and initially configure the integrated radio transceiver.

At a high level, RAIL and the Radio Configurator abstract away most differences that exist between EFR32 devices. The Connect stack and application framework leverage RAIL implicitly, and as such also benefit from (and further extend) abstraction to the application from most variation across potential target EFR32 devices. However, some device capabilities vary enough - or some features are unique enough - that they cannot simply be normalized "under the hood" of RAIL and/or the Radio Configurator. In these cases, customers must consider the impact of these generation-specific differences when planning a migration path for EFR32 proprietary wireless applications. Often, these distinguishing characteristics may in fact be enhancements that invite the porting of existing proprietary applications to new, more capable, EFR32 devices. More information on these differences relevant to proprietary wireless applications is presented in the following sections:

- Section 2. Minor Differences between EFR32 Generations
- Section 3. RF Sense on EFR32xG22
- Section 4. EM1P

Note: This document focuses on device-specific variations and where to find information on how RAIL accommodates them. As stated above, Connect abstracts most of these differences away. In the few cases where it is appropriate, they will be highlighted in this document.

2. Minor Differences between EFR32 Generations

This section presents a brief collection of generation-specific EFR32 differences that can impact proprietary wireless applications. Specific references for further reading are provided in each section. For more detailed information, see the online RAIL API Reference (https://docs.silabs.com/rail/latest/) as well as the reference manual and datasheet for your device.

2.1 RAIL (Radio) Configuration Compatibility

RAIL configures the radio on EFR32 devices with a desired PHY using one of two techniques:

- Load a static radio configuration generated for your project by the Simplicity Studio wireless application workflow.
- Load a RAIL embedded PHY using one of the protocol-specific APIs.

Static RAIL configurations are compatible across EFR32 families. All device families in a single generation share a common radio, but the static configurations are not compatible across EFR32 generations. In other words, for example, EFR32MG12 and EFR32FG12 share the same radio and therefore the same RAIL radio configuration, but a radio configuration generated for EFR32xG12 is not compatible with an EFR32xG14 device. When using the embedded PHYs, RAIL abstracts away most underlying differences, but in some cases, generation-specific limitations remain, as described in the following sections.

Connect may use the same static configuration files, hence it has the same compatibility limitations. The embedded RAIL configurations are available from Gecko SDK Suite version 3.1 onwards when using Connect.

Note: Static radio configurations are not currently supported on the EFR32xG21 platform. Hence, migrating to EFR32xG21 is only currently viable for applications using the embedded PHYs.

For more information on static radio configurations, see https://docs.silabs.com/rail/latest/group-radio-configuration, and on embedded configuration routines, see https://docs.silabs.com/rail/latest/group-protocol-specific in RAIL.

2.2 Power Amplifier (PA)

Some PAs might not be available on certain parts. For example, devices with limited TX power might not have the high-power PA, or “sub-GHz only” parts won't have 2.4 GHz PAs. Beyond these OPN-specific limitations, PA options differ by EFR32 generation as follows:

- Sub-GHz PAs are only available on EFR32xG1, xG12, xG13, xG14, and xG23 generations.
- The EFR32xG1, xG12, xG13, xG14, and xG22 generations have both low-power and high-power 2.4 GHz PAs (the number of power levels per PA varies in xG22 vs xG1x).
- EFR32xG21 devices have low-power, mid-power, and high-power 2.4GHz PAs.
- EFR32xG23 devices have either a mid-power (14 dBm) or a high-power (20 dBm) sub-GHz PA, based on the device’s OPN.
- EFR32xG24 devices have either a mid-power (10 dBm) or a high-power (19.5 dBm) 2.4 GHz PAs, based on the device’s OPN.
- EFR32xG25 devices have a high-linearity PA dedicated for OFDM modulation, and a high-efficiency PA, both operating only on sub-GHz with a maximum power of 16 dBm.

For more info on EFR32 Power Amplifier (PA) Initialization, see https://docs.silabs.com/rail/latest/efr32-main, on Power Amplifier treatment in RAIL, see https://docs.silabs.com/rail/latest/group-p-a, and on the available PAs, see the device's datasheet.

2.3 Antenna Switch

- EFR32xG21 has an integrated antenna switch supporting two RF paths.
- All other EFR32 devices have a single RF path, though external RF switches can be used.

For more information on antenna control in RAIL, see https://docs.silabs.com/rail/latest/group-antenna-control.

2.4 RAIL Time Base

- The EFR32xG1 time base tick is 2 μs.
- For all other EFR32 generations, the time base tick is 0.5 μs.

For more information about the EFR32 RAIL timebase, see https://docs.silabs.com/rail/latest/efr32-main, and about system timing in RAIL, see https://docs.silabs.com/rail/latest/group-system-timing.

2.5 Low-Frequency Timer Dependencies

RAIL does not use any low-frequency clocks by default, but LF clocks and a PRS channel are required when timer sync is enabled through `RAIL_ConfigSleep()`.

- EFR32xG1, EFR32xG12: An RTCC compare channel is required for RAIL when timer sync is used. This is channel 0 by default, but it can be customized in the application.
- All other EFR32 generations: The radio has a dedicated RTC timer for timer sync.

For more information on RAIL usage of EFR32 low-frequency clocks, see https://docs.silabs.com/rail/latest/efr32-main, and on the PRS and RTCC channel customization, see https://docs.silabs.com/rail/latest/group-sleep.

2.6 Buffer Handling

- Beginning with EFR32xG22, the RX and TX buffers must be word-aligned.

For more information on EFR32 Receive and Transmit FIFO Buffers in RAIL, see https://docs.silabs.com/rail/latest/efr32-main.

2.7 IEEE 802.15.4 Options

- EFR32xG1 has limited 15.4g/e support compared to newer parts:
  - 4B CRC is not supported.
  - Only whitened packets are supported in 15.4g mode.
  - Enhanced ACKs are not supported.
  - Cannot receive MultiPurpose frame types.

For more information on IEEE 802.15.4 support in RAIL, see https://docs.silabs.com/rail/latest/group-i-e-e-e802-15-4.

2.8 Calibrations

- EFR32xG1 performs temperature (VCO) calibration when crossing 0^{o}C (with 5^{o}C hysteresis).

For more information on EFR32 radio calibration support, see https://docs.silabs.com/rail/latest/efr32-main, and on radio calibrations in RAIL, see https://docs.silabs.com/rail/latest/group-calibration.

2.9 Listen Before Talk (LBT)

- EFR32xG1 always averages RSSI during Clear Channel Assessment (CCA).
- All other parts can select between RSSI averaging and peak measurement.

For more information on the relevant RAIL option, see https://docs.silabs.com/rail/latest/group-transmit#gaa887d34998698109746e6763fbf17cda in the online RAIL documentation.

2.10 DMP Transition Time

In a Dynamic Multiprotocol (DMP) application, the time required for the radio to switch protocols varies as follows:

- Series 1 devices use 430 μs transition time by default.
- Series 2 devices use 510 μs transition time by default except EFR32xG21, which uses 500 μs.

For more information on understanding the protocol switch time, see https://docs.silabs.com/rail/latest/rail-multiprotocol#rail-radio-scheduler-switch-time.

2.11 Direct Mode

TX and RX Direct Modes are available for 2-level modulations on most EFR32 Wireless MCU devices, except on EFR32xG21 and EFR32xG22.

- EFR32xG23 introduces direct to buffer mode.
- EFR32 Series 1 devices have better sensitivity results in packet mode compared to the direct mode when OOK modulation is used.

2.12 Wi-SUN Capabilities

EFR32xG25 devices are the first EFR32 Wireless MCU devices that support OFDM PHYs defined by the Wi-SUN FAN 1.1 standard, and SUN-OQPSK (also known as MR-OQPSK) defined by IEEE 802.15.4. The transceiver has a software-defined modem running on RISC-V architecture supporting OFDM modulation.

The Wi-SUN FAN 1.0 is supported by EFR32xG25, EFR32xG12, and EFR32xG23 device generations. However, only EFR32FG25 and EFR32MG12 support the Wi-SUN stack provided by Silicon Laboratories.

The radio has a dedicated Power Amplifier module optimized for Wi-SUN OFDM.

Note: Concurrent listening on EFR32xG25 devices for FSK and OFDM signals is supported, but once a signal is detected, only one can be demodulated.

2.13 RFFPLL on EFR32xG25

EFR32xG25 device generation introduces a new clock source module called Radio Frequency Friendly Phase Locked Loop (RFFPLL). RFFPLL generates a roughly 100 MHz clock from the XO frequency which is then used for driving the radio to avoid clock spurs in the RF chain. The actual configuration depends on the radio frequency band and the XO frequency.

RFFPLL settings are controlled via the Device Init: RFFPLL component, and can be set via the `CMU_RFFPLLInit()` API. RFFPLL must be configured during the initialization and cannot be changed at runtime. When creating a radio PHY, the RFFPLL band selection (**RF Frequency Planning** field) and the HFXO frequency must match the configuration applied via the platform code. If a PHY is assuming a certain RFFPLL band but the RFFPLL is configured differently, a RAIL assert error will be generated.

Note: RFFPLL can be optionally used as the system clock.

3. RF Sense on EFR32xG22

RF Sense is a low-power feature available on EFR32 Wireless MCU devices, except on EFR32xG23. With it, the radio can sense the presence of RF energy and "wake up" an MCU from EM2 (or any other) power mode. In practice, RF Sense provides an ultra-low power interrupt source that runs on the ULFRCO clock.

Since the RFSENSE block implements a wide band circuit, it can detect energy in a broad frequency range between 100MHz and 5GHz - filtered only by the matching network of the RF front end. On one hand this is an advantage, as there is no need for additional PCB components to support the feature. However, there is a drawback: the wake-on-RF capability is responsive to any unfiltered interferer signals.

EFR32xG22 devices include an enhanced RF Sense module, which improves performance as compared to the EFR32 Series 1 implementation in multiple ways:

- RF Sense works below 0^{o}C.
- RF Sense works even when voltage is scaled down.
- In addition to legacy behavior, the EFR32xG22 introduces "Selective Mode" RF Sense.

3.1 Legacy Mode

In Legacy (Energy Detection) mode, EFR32xG22 RF Sense is fully compatible with the feature in Series 1 devices. This means that if the RF Sense module detects energy for a configured duration, it generates an interrupt.

3.2 Selective Mode

Selective mode mitigates the unfiltered nature of RF Sense. Instead of simply detecting energy for a given time period, it detects "a pattern of energy" - which is essentially an On-Off Keying (OOK) packet. The packet is Manchester-coded and uses a fixed 1 kbps bitrate, 1 B preamble, and 1-4 B sync word (no payload is added, see additional details below). This packet can be transmitted by any 2.4 GHz OOK-capable device, including most EFR32 wireless MCUs.

3.2.1 Setting Up Selective RF Sense

Selective RF Sense can be started with the API `RAIL_StartSelectiveOokRfSense()` instead of the legacy `RAIL_StartRfSense()`. The API takes a `config` parameter of type `RAIL_RfSenseSelectiveOokConfig_t` that sets up the desired RFSENSE Selective mode sync word configuration. After this function is called, the interrupt is enabled and the device can enter sleep mode.

On the transmit side, calling `RAIL_ConfigRfSenseSelectiveOokWakeupPhy()` switches the radio config to the special wakeup PHY. Next, configure the wakeup packet using `RAIL_SetRfSenseSelectiveOokWakeupPayload()` so that it matches the configuration on the RX side. Finally, use `RAIL_StartTx()` (or any other TX API) to send the wakeup packet. See 3.2.3 Selective Mode Transmit Example Without Using API for an illustration.

Selective RF Sense (RX and TX) can be performed in RAILtest as well, see UG409: RAILtest User’s Guide for details.

Note: On EFR32xG21 devices, you cannot transmit the Wakeup Packet due to the lack of OOK support.

3.2.2 The Wakeup Packet

The wakeup packet is a fixed-configuration OOK packet with the following settings:

- Starts with a 1 B preamble (always 0x55).
- Followed by a 1-4 B sync word.
- No payload required.
- Both the preamble and sync word are transmitted LSB first.
- 1 kcps chiprate (Manchester coded)/500 bps bitrate (useful data rate).
- Recommended carrier is 2.45 GHz (EFR32 radios use this frequency).

3.2.3 Selective Mode Transmit Example Without Using API

Assume you select `0xb16e` as your sync word, and you want to transmit it with only a signal generator (or a simple radio with MSB-first byte handling and no Manchester coder).

First, flip the endianness of both preamble and sync word: `0x55` becomes `0xaa` and `0xb16e` becomes `0x768d`.

The full packet is then `0xaa768d`, which after Manchester coding becomes `0x99996a6995a6`.

Configuring this encoded packet and transmitting on 2.45 GHz with high enough TX power should wake up a device configured for selective RF Sense with the `0xb16e` sync word.

4. EM1P

Beyond the energy modes available on prior devices, EFR32xG22 introduces **EM1P**, a new method to reduce energy consumption while using the radio. In this (intermediate) mode, the HF crystal oscillator (HFXO) is kept running (and the radio remains active) if the following conditions are met:

- The radio state is anything other than idle (for example RX or TX), and
- Power Manager or the application software requests to enter EM2.

This sequence puts the Cortex-M33 into sleep mode, and clocks to the core, and all high-speed peripherals are disabled (peripherals and oscillators capable of EM2, EM3, or EM4 operation remain available).

On earlier EFR32 devices, entering EM2 mode would unconditionally shut down the radio - regardless of the operation in progress. Doing so during certain radio activities would potentially incur undesirable side effects (for example, FIFO corruption).

4.1 Comparison of EM1P and Other Energy Modes

EM1P is not technically a standalone energy mode (like EM1, EM2, and so on). Rather, it is an operating condition where most of the EFR32 enters EM2 Deep Sleep, but the radio (and its requisite HFXO clock source) are retained. As such, EM1P can be (perhaps confusingly) viewed as "EM2 with Radio".

This combination results in lower current consumption than EM1 mode, and higher consumption than full EM2. The following table depicts some observations of these influences on energy usage taken on a single device. Current consumption was measured initially at EM0, when the radio was idle and in RX. After transitioning into EM1/EM2, current was again measured. The cell in the bottom-right corner represents the EM1P measurement result (radio was in RX, device then transitioned to EM2).

Radio State	Initial (EM0) Current	Final Energy Mode	Resulting Current
Idle	2.36 mA	EM1	1.60 mA
"	"	EM2	1.56 µA
RX	5.92 mA	EM1	5.15 mA
"	"	EM2	4.85 mA

Note: The values above are **not** guaranteed, and will vary across devices and scenarios due to numerous factors. They are presented here as a single illustrative example of the general performance trends among these different modes of operation. Consult your device datasheet for all definitive guidance regarding specifications and performance.

4.2 Scheduled Radio Operations

EM1P mode does not impact scheduled RX and scheduled TX operations, as in those cases the radio is in the idle state when the firmware enters EM2.

IoT Portfolio
www.silabs.com/products

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

SILICON LABS

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

www.silabs.com