Developing a Custom Application on Digi Wi-SUN Modules

(Japan HAN Wi-SUN Profile)

This guide outlines the recommended development materials and steps for customers who want to develop a custom application on Digi Wi-SUN modules based on the **Japan HAN Wi-SUN profile**, using Silicon Labs development tools and APIs.

---

1) Development Materials Needed

Digi Hardware

- **Digi XBee Wi-SUN FG25 module**
    - Silicon Labs radio: **EFR32FG25B221F1920IM56-B**

Required Software Tools

- **Silicon Labs Simplicity Studio 6**
- **GCC 12.2.1** (recommended by Silicon Labs)
  OR
- **IAR Embedded Workbench v9.50.1** (used by Digi)
- **Silicon Labs Simplicity SDK 6.0 (v2025.12.1)**

J-Link Programming Hardware

A **SEGGER J-Link programmer** is required to program firmware onto the Digi module.

Supported options include:

- **SEGGER J-Link PLUS Classic (8.08.28)**
  https://shop-us.segger.com/product/j-link-plus-classic-8-08-28/
- **Silicon Labs Wi-SUN-PK6016A Pro Kit**

The **Silicon Labs BRD4002A Wireless Pro Kit Mainboard** (included with the PK6016A kit) contains an onboard SEGGER J-Link debugger that can also be used to program the Digi module.

In order to use this method a separate 20-pin ribbon cable will be needed to connect between the Silabs Main Board and the T-connector provided by Digi, such as the Samtec FFSD-10-D-06.00-01-N. Silabs may also have a source for this cable.

https://www.samtec.com/products/ffsd-10-d-06.00-01-n#featureinformation

Programming can be performed using **Simplicity Commander** in combination with a J-Link device and the appropriate JTAG adapter and ribbon cable.

---

2) Optional Development Hardware (For Comparison & Validation)

- **Silicon Labs Wi-SUN-PK6016A Pro Kit**
  https://www.silabs.com/development-tools/wireless/proprietary/wi-sun-pk6016a-efr32fg25-pro-kit

This kit may be used for:

- Comparing radio behavior against Digi modules
- Reviewing Silicon Labs reference examples
- Validating PHY/channel configurations during development

Development can be completed entirely on Digi modules without this kit.

---

3) Recommended Development Approach

Step 1 — Load the Digi-Provided RAIL Test Application

Digi will provide:

- A pre-built **RAIL Test application binary** for the Digi Wi-SUN module (.s37 file)
- The associated **Simplicity Studio project files**

Customers may:

- Program the provided binary directly onto the Digi module for immediate radio evaluation
- Import the project into Simplicity Studio to:
  - Rebuild the application
  - Modify configurations

- Extend functionality as required

Programming is performed using **Simplicity Commander** and a supported **J-Link device**.

This provides a validated starting point tailored specifically to the Digi module hardware.

Step 2 — Understand and Use RAIL (Radio Abstraction Interface Layer)

The primary interface to the EFR32 radio transceiver is the:

Radio Abstraction Interface Layer (RAIL)

RAIL is a C-based API that serves as the low-level radio driver for the EFR32 platform.

It provides:

- Radio configuration
- Packet transmission and reception
- PHY configuration
- Radio event callbacks
- Timing and state management

Custom proprietary or third-party stacks should be implemented on top of RAIL.

RAIL API documentation: https://docs.silabs.com/rail/2.4/api-index/

Step 3 — Configure the Radio Using the Radio Configurator

Simplicity Studio includes the **Radio Configurator**, which enables generation of:

- PHY parameters
- Channel plans
- Modulation settings
- RAIL configuration files

This tool should be used to generate the radio configuration required for the Japan HAN Wi-SUN profile or any proprietary PHY implementation.

Step 4 — Use the Silicon Labs HAL for On-Chip Peripherals

For device-level hardware access, Silicon Labs recommends using the **Silicon Labs HAL framework** to interface with on-chip peripherals such as:

- GPIO
- Timers
- UART
- Interrupts
- Other peripheral modules

Using the HAL ensures portability and compatibility with the Simplicity SDK.

Step 5 — Implement a Translation Layer Between RAIL and Your Stack

To maintain a clean and portable architecture, it is recommended that customers implement a translation layer between:

- The RAIL configuration and radio event callbacks
- The custom-developed protocol stack or application layer

This abstraction layer isolates the application from low-level RAIL details and simplifies long-term maintenance.

Step 6 — Configure I/O Using the Pin Configurator

Digi modules use different pin mappings than Silicon Labs development boards.

Customers should use the **Pin Configurator tool within Simplicity Studio** to:

- Assign peripheral functions to the appropriate Digi module pins
- Configure GPIO modes
- Route UART, SPI, or other peripheral interfaces as required

This tool simplifies mapping Digi module pins to the intended application functionality.

---

4) Porting Considerations

When adapting examples or code originally developed for Silicon Labs evaluation boards:

Crystal Frequency Differences

Silicon Labs development boards typically use a **39 MHz crystal (HFXO)**, while Digi modules use a **40 MHz external crystal**.

When porting code, ensure that:

- The HFXO frequency configuration is updated appropriately
- The clock tree configuration reflects the 40 MHz source
- Any radio timing or PHY configuration assumptions tied to HFXO frequency are validated

Failure to update these settings can result in incorrect radio timing, channel spacing errors, or degraded RF performance.

Why the HFXO Crystal Frequency Matters (39 MHz vs. 40 MHz)

The HFXO crystal frequency is the primary high-frequency reference for the EFR32 clock tree and a key input to the radio subsystem. RAIL and the radio configuration rely on this reference to correctly calculate and maintain:

- RF synthesizer tuning
- Channel frequency and spacing
- Symbol timing
- Packet timing and radio state transitions
- Calibration and radio timing parameters

If an application or radio configuration is built assuming a **39 MHz HFXO** (typical Silicon Labs reference hardware) and is then moved to a Digi module using a **40 MHz HFXO**, subtle but significant issues may occur, including:

- Frequency offset from the intended channel center
- Incorrect channel spacing
- Timing-related receive sensitivity degradation
- Reduced link margin or unreliable network join behavior

For this reason, any radio configuration or example project migrated from Silicon Labs evaluation hardware should be reviewed to ensure that HFXO and clock settings are correct for Digi module hardware.

---

5) Additional Notes and Resources

Silicon Labs provides many examples and community references for integrating third-party libraries and custom protocol stacks.

The following documents are especially relevant:

Application Notes

- **AN0004.2 — EFR32 Series 2 Wireless MCU Clock Management Unit (CMU)**
  https://www.silabs.com/documents/public/application-notes/an0004.2-efr32-series2-cmu.pdf
- **AN0918.2 — Gecko Series 1 to Series 2 Compatibility and Migration Guide**
  https://www.silabs.com/documents/public/application-notes/an0918.2-efm32-to-efr32xg2x-migration-guide.pdf
- **AN0012 — General Purpose Input Output (GPIO)**
  https://www.silabs.com/documents/public/application-notes/an0012-efm32-gpio.pdf
- **AN1244 — EFR32 Migration Guide for Proprietary Applications**
  https://www.silabs.com/documents/public/application-notes/an1244-migration-efr32-families.pdf
- **AN961 — Bringing Up Custom Devices for EFR32FG**
  https://www.silabs.com/documents/public/application-notes/an961-custom-nodes-efr32.pdf

RAIL API Documentation

- **Getting Started with RAIL**
  https://docs.silabs.com/rail/latest/rail-getting-started-overview/
- **RAIL API Reference**
  https://docs.silabs.com/rail/2.4/api-index/

Quick Start Guides

- **QSG138 — Proprietary Flex SDK v2.x Quick Start Guide**
  https://www.silabs.com/documents/public/quick-start-guides/qsg138-flex-efr32.pdf

Technical Presentation

- **Silicon Labs Tech Talk: Proprietary Solutions with Simplicity Studio**
  https://www.silabs.com/documents/public/presentations/tech-talks-uncover-sub-ghz-and-proprietary-solutions-with-simplicity-studio-v5-india.pdf