UG103.8: Silicon Labs Tools Fundamentals

This document provides an overview of the toolchain used to develop, build, and deploy EmberZNet applications, and discusses some additional tools and utilities. It also provides references to more detailed information about each item, where applicable.

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 as 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

- Describes Silicon Labs stack software and development environment
- Reviews additional tools for use during different development phases.

1. Introduction

As with most embedded development technologies, Silicon Labs provides a set of tools to allow developers to create a product using Silicon Labs wireless networking products. This document provides an overview of the toolchain that used to develop, build and deploy EmberZNet applications. The document does not provide a step-by-step guide. If you are just getting started with Silicon Labs development kits, see the Quick Start Guide in your kit as a starting point.

The tools in the toolchain fall into one of three categories:

- Stack Software
- Compiler Toolchain
- Application Development and Debugging Toolchain

The actual toolchain that you will use is device- and processor model-dependent. For this discussion, the processor model is either System-on-Chip (SoC) or Network Coprocessor (NCP). The SoC model requires that the customer application to be co-resident with the stack. The NCP model requires that the customer application be on a separate host processor and the stack run on the NCP. The following table summarizes the major tools for each device.

Table 1.1. Toolchain Summary

Stack Software	Compiler	Application Development and Debugging
SoC		
Stack Libraries, HAL source, API Documentation, Sample Applications, Development Kit	IAR EWARM (required for parts with less than 512 kB, such as the EFR32xG1, and any of the Dynamic Multiprotocol examples) or GCC	Simplicity Studio
NCP		
Stack Libraries, HAL source, API Documentation, Sample Applications, Utilities	3rd Party Compiler Toolchain (depends on Host Processor Selection)	Simplicity Studio

In addition to the major tools above, Silicon Labs also supplies a number of single function tools and utilities such as

- Bootloaders
- Programming Support Tools

The following sections provide more detail about the most important elements of the toolchain.

2. Stack Software

The network stack software is a collection of libraries, source code, tools, sample applications, and product documentation. The network stack API is documented in an online API reference as well as other documents installed with the stack installer, or available through the development environment. The network stack is delivered as a collection of libraries that you can link to your applications. A description of each library is provided in the development environment.

In addition to the *Fundamentals* document family, other resources are available for learning more about your stack software. See the documentation list on Simplicity Studio's Launcher perspective for more information about available documentation.

This document applies to the EmberZNet stack. EmberZNet is the Silicon Labs implementation of the Zigbee protocol supporting the Zigbee PRO feature set. All EmberZNet PRO applications must be linked with the stack library. The following figure illustrates how customer and Silicon Labs stack software interact.

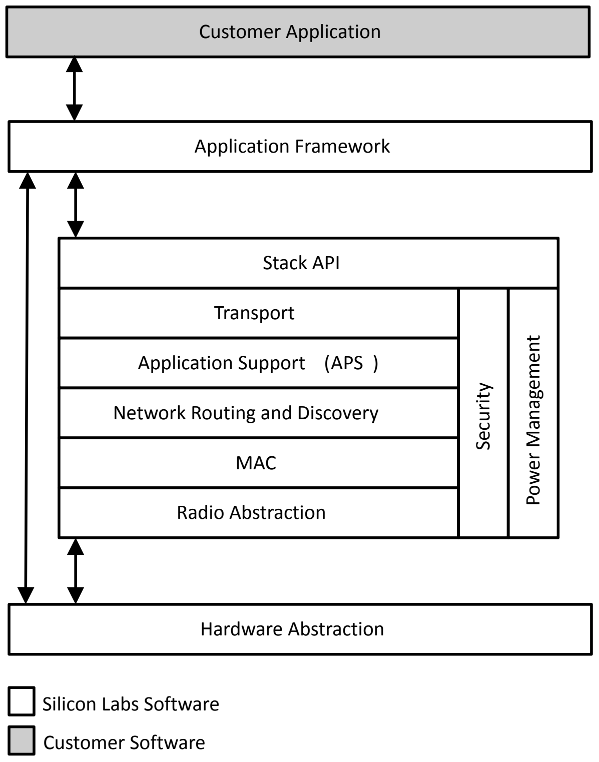


Figure 2.1. Silicon Labs Stack and Customer Software Interaction

3. Compiler Toolchain

The compiler toolchain is different based upon which platform and processor model you are using. EFR32 devices can use the IAR toolchain and must do so when compiling dynamic multiprotocol applications and when building applications for devices with less than 512 kb of flash. Otherwise developers have the option of using GCC (The GNU Compiler Collection), provided with Simplicity Studio.

Note: Application images created with GCC are larger than those created with IAR. If GCC is used to compile the example applications in EmberZNet SDK, the target must be a part with at least 512 kB of flash.

Note: The IAR toolchain is licensed through IAR, and the version of the compiler must be compatible with the stack version. Specific information is supplied with the development kit and stack release notes.

The EFR32, when acting as a Network Co-Processor, is designed to be a coprocessor to whatever host developers are using. The network coprocessor software for a number of parts is shipped as a binary image, so there is no need to recompile it. These can be found in the *SimplicityStudio\v5\developer\sdks\gecko_sdk_suite\v3.0\protocol\zigbee\ncp-images* folder. One of the most commonly selected hosts is an ARM processor, though many others can be used. Additional information is available at the Silicon Labs Support Portal (http://www.silabs.com/support/Pages/default.aspx).

4. Development Environment

Along with the stack, Silicon Labs provides the Simplicity Studio development environment. Simplicity Studio 5 (SSv5) (for Zigbee stack version 6.8.x) and Simplicity Studio 4 (SSv4) (for Zigbee stack version 6.7.x) provide access to target device-specific web and SDK resources; software and hardware configuration tools; an integrated development environment (IDE); and advanced, value-add tools for network analysis and code-correlated energy profiling. For information on using Simplicity Studio 5 see the online Simplicity Studio 5 User's Guide, available on https://docs.silabs.com/ and through the SSv5 help menu. For an overview of Simplicity Studio 4 see *AN0822: Simplicity Studio User's Guide*.

The following are some of the more important development tools also provided with Simplicity Studio:

The **Hardware Configurator** tool (for EFR32 platforms only) allows you to easily configure new peripherals or change the properties of existing ones. Hardware Configurator options are available on many of the HAL and I/O plugins. See *AN1115: Configuring Peripherals for 32-Bit Devices in Simplicity Studio* for more information about the Hardware Configurator both within the Simplicity IDE and as a separate tool.

Simplicity Commander is a single, all-purpose tool to be used in a production environment. It is invoked using a simple Command Line Interface (CLI) that is also scriptable. Simplicity Commander enables customers to complete these essential tasks:

- Flash their own applications.
- Configure their own applications.
- Create binaries for production.

See *UG162: Simplicity Commander Reference Guide* for more information.

The **Network Analyzer** tool helps debug network connectivity by displaying radio packets and certain debug interface events in a format that is easy to visualize and analyze.

The **Energy Profiler** provides energy debugging capability by displaying graphical real-time energy consumption information. This can be particularly useful for developing a low-power application. See *UG343: Multi-Node Energy Profiler User's Guide* for more information.

5. Peripheral Drivers

Embedded source C code is provided for drivers of peripherals such as the serial controller and analog-to-digital converter (ADC). These drivers let you incorporate standard functionality into custom applications. For more information on these drivers, see the stack API reference and, if applicable, Platform API reference for your platform.

6. Bootloaders

A bootloader is a program stored in reserved flash memory that allows a node to update its image on demand, either by serial communication or over the air. Production-level programming is typically done during the product manufacturing process, yet it is desirable to be able to reprogram the system after production is complete. More importantly, it is valuable to be able to update the device's firmware with new features and bug fixes after deployment. The bootloading capability makes that possible. See *UG103.6: Bootloader Fundamentals* for more details.

7. Manufacturing Library

Silicon Labs supplies a library called the manufacturing library, which is meant to be linked with your production application and provide a means of doing end-of-line manufacturing testing with a light weight library that provides similar functionality to the Node Test application (described in section 8). It can be used to characterize radio performance and validate proper functionality of your device. You can use a token to indicate which mode the application should boot into, either production or test mode. Once done testing, the token will indicate that the application should boot into production mode. See *AN1162: Using the Manufacturing Test Library* for more information.

8. NodeTest

The NodeTest applications provide low-level control of the radio and can be used to perform these tasks:

- Characterize radio performance.
- Set manufacturing and stack parameters (tokens).
- Verify proper functionality after manufacturing.
- Control the radio properly for the certification process required by many countries.

For more information about nodetest, see *AN1019: Using the NodeTest Application*.

Most customers have standard product manufacturing test flows, but some do not incorporate RF testing. To address this issue, see *AN700.1: Manufacturing Test Guidelines for the EFR32*. This document describes the different options available for integrating RF testing and characterization into your standard test flows. It is intended for test engineers who are moving from the early prototype development stage to the manufacturing production environment and need assistance with manufacturing test process development.

9. Development Hardware

Wireless Starter Kit Mainboard for the EFR32

The Wireless Starter Kit (WSTK) includes software stacks, sample code, radio boards, and mainboards. The Wireless Starter Kit Mainboard contains an on-board J-Link USB connector that provides access to most of the kit's development features when the kit is connected to a host computer. Features include:

- Debugging and programming the target device using the on-board J-Link debugger. The debugger supports three different debug interfaces: Serial Wire Debug, JTAG, and C2 Debug.
- Communication with the target device over the virtual COM port using USB-CDC.
- Accurate current profiling using the Advanced Energy Monitor.

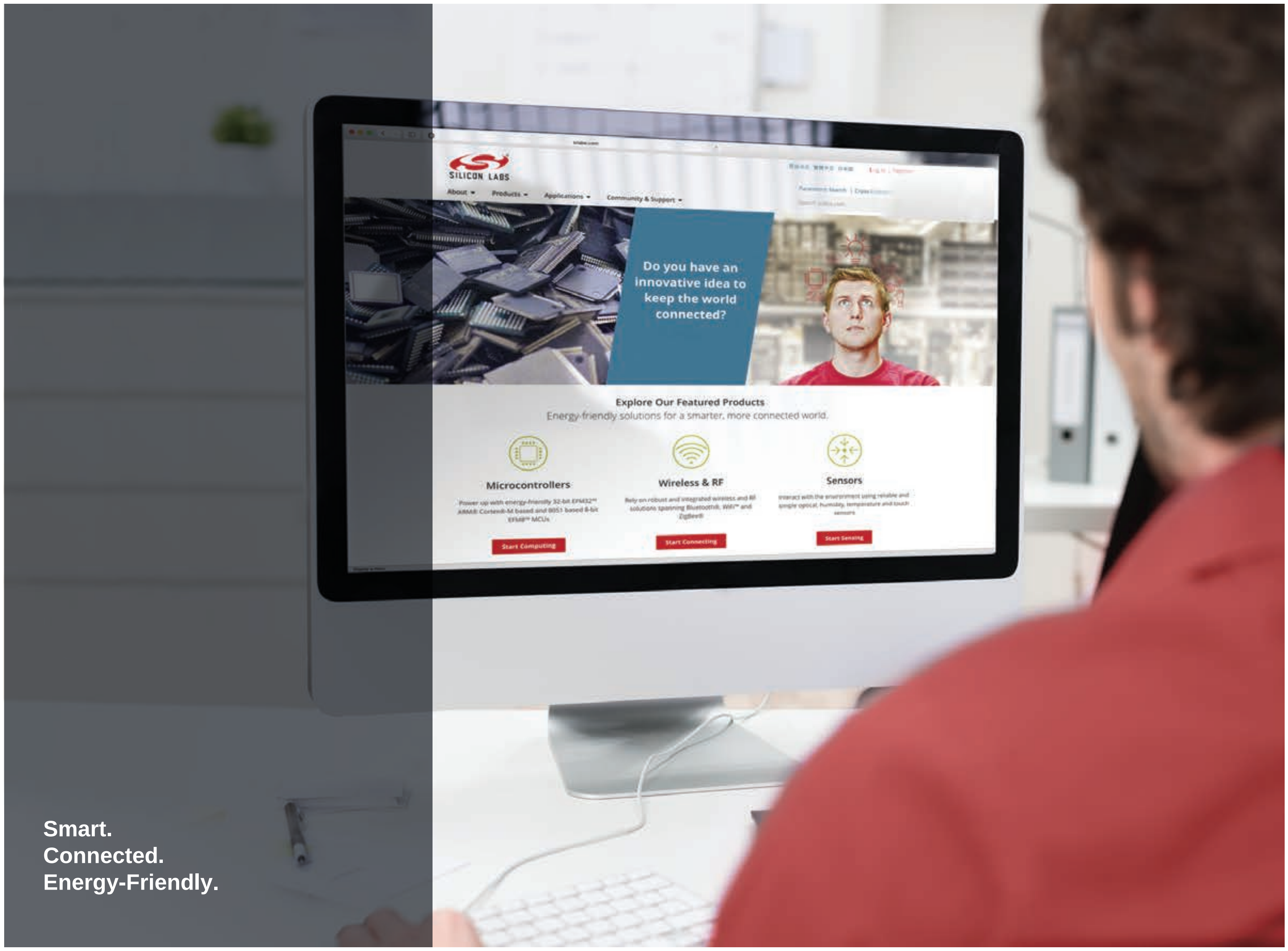
In addition to providing access to development features of the kit, the USB connector is also the main power source for the kit. USB 5V from the connector powers the board controller and the Advanced Energy Monitor.

The WSTK Mainboard also contains sensors and peripherals for easy demonstration of some of the EFR32's many capabilities.

See the documentation set for the specific WSTK for more information about the mainboard.

10. Zigbee Over-the-air Bootload Image Generation (image-builder)

The Zigbee Over-the-air bootload specification describes a universal container format for transporting bootload images. Silicon Labs provides a tool known as image-builder that can generate, parse, and manipulate those images. Detailed information on this tool can be found in *AN716: Instructions for using Image Builder*.


Smart.
Connected.
Energy-Friendly.
Do you have an innovative idea to keep the world connected?
Explore Our Featured Products
Microcontrollers
Wireless & RF
Sensors

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