UG103.8

APPLICATION DEVELOPMENT FUNDAMENTALS: TOOLS

This document provides an overview of the toolchain used to develop, build, and deploy EmberZNet and Silicon Labs Thread applications, and discusses some additional tools and utilities. It also provides references to more detailed information about the item in the toolchain. If you are developing using Silicon Labs Simplicity Studio, refer to its online and referenced documentation for information about tools.

New in This Revision

Updated for Thread release; adds Manufacturing Library section.

Contents

1 Introduction

As with most embedded development technologies, Silicon Labs provides a set of tools to allow you (the developer) to create a product using Silicon Labs wireless networking products. Each Silicon Labs chip family has a toolchain associated with it that addresses its unique development requirements. Wherever possible, we have selected the best development tools available, or we have developed the tool on our own.

Silicon Labs is introducing a new development environment, Simplicity Studio. In that environment, some of these tools are no longer required. This document revision is specific to the Ember Desktop development environment, used with the EmberZNet and Silicon Labs Thread stacks. If you are developing using Silicon Labs Simplicity Studio, for example with Silicon Labs Connect, refer to the online help and referenced documentation for information about tools.

This document provides an overview of the toolchain that you will use to develop, build and deploy your applications. This document does not provide a step-by-step guide to developing, building, and deploying your application. 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. Table 1 summarizes the major tools for each device.

Table 1. Toolchain Summary

Stack Software	Compiler	Application Development and Debugging
EM35x SoC		
Stack Libraries, HAL source, API Documentation, Sample Applications, Development Kit	IAR EWARM: IDE: Compiler, Online Help; Debugger (device level); Document Library	Ember Desktop + Application Builder + Online Help
EM35x NCP		
Stack Libraries, HAL source, API Documentation, Sample Applications, Utilities, Optional Add-on Development Kit	3rd Party Compiler Toolchain (depends upon Host Processor Selection)	Ember Desktop + Application Builder + Online Help

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.

This document applies to the EmberZNet and Silicon Labs Thread network stacks. EmberZNet 5.x is the Silicon Labs implementation of the ZigBee 2007 stack supporting the ZigBee PRO and ZigBee RF4CE feature set. Silicon Labs Thread is the implementation of the Thread protocol, introduced in the summer of 2015. Both Thread and ZigBee PRO support mesh networks, with their increased flexibility and reliability. All EmberZNet PRO and Silicon Labs Thread applications must be linked with the stack library. Figure 1 illustrates how customer and Silicon Labs stack software interact.

In addition to the *Fundamentals* document family, other resources are available for learning more about your stack software. See the documentation index installed with your stack for more information about available documentation.

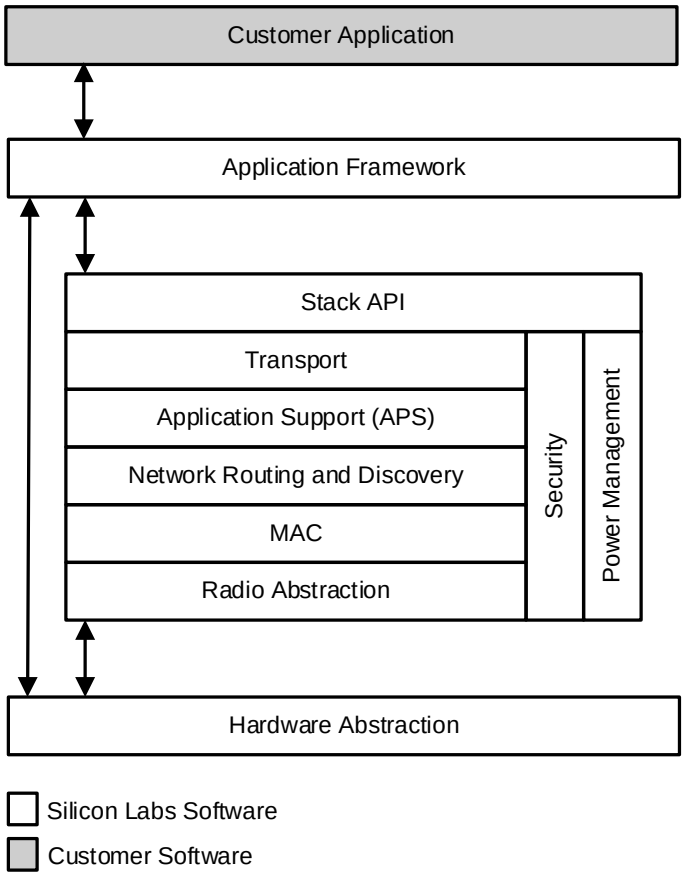

Figure 1. Silicon Labs Stack and Customer Software Interaction

3 Compiler Toolchain

As mentioned above, the compiler toolchain is different based upon which platform and processor model you are using. The EM3x devices, when being used as Systems-on-Chip, have customer-accessible processors. The EM3x uses an ARM® Cortex™-M3 processor and an IAR toolchain that include a compiler, linker, debugger, sample applications, and user documentation.

Note: The IAR toolchain is licensed through IAR. Specific information is supplied with your development kit and stack release notes.

The EM3x, when acting as a Network Co-Processor, is designed to be a coprocessor to whatever host you might be using. The network coprocessor software is shipped as a binary image, so there is no need to recompile it. One of the most commonly selected hosts is an ARM processor, though many others can be used. You can get additional information at the Silicon Labs Support Portal (http://www.silabs.com/support/Pages/default.aspx).

4 Development Environment

Along with the stack, Silicon Labs provides a development environment (Ember Desktop or Simplicity Studio) and one or more application frameworks. The application frameworks serve as starting places for your application development, and can be customized to your needs using the Application Builder tool in the development environment. The Network Analyzer tool, also part of the development environment, manages the development kit hardware and displays network and node activity in real time. Refer to the Software Quick Start Guide installed with your development kit software for more information about your development environment.

At the time of this writing, Ember Desktop is being integrated into Simplicity Studio so that customers will have a single tool chain in the future.

Refer to document *UG102—Application Framework Developer Guide,* and to the *Application Framework API* for more information about the application frameworks applicable to your protocol and platform.

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 for your platform.

6 Bootloaders

The 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. If you are developing with EmberZNet PRO, see the document *UG103.6—Application Development Fundamentals: ZigBee PRO Bootloaders* 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.

8 Node Test

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 the *AN710—Bringing up Custom Devices*.

Most customers have standard product manufacturing test flows, but some do not incorporate RF testing. To address this issue, please see document *AN700—Manufacturing Test Guidelines*. This document describes the different options available for integrating RF testing and characterization into your standard test flows. This application note 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 Hex File Utilities

A set of tools for manipulating hex files is also available. All of these utilities are command line (DOS console) applications. The tools are listed in Table 2.

Table 2. Hex File Tools

Tool	Description
`em3xx_load`	These utilities can be used to program the flash memory space of the EM3x through the SWJ interface.
`em3xx_load`	These utilities can be used to examine (dump) or generate a .hex file from the flash of the EM3x through SWJ.
`em3xx_convert`	This utility is intended for use in converting IAR .s37 application files into Ember .ebl bootload format. In addition to the representation of the application, you can include a representation of the application bootloader of the customer manufacturing tokens.
`em3xx_buildimage`	This utility is intended for use in manipulating EM3x file images, which includes generating Intel Hex format (.hex) files from a variety of sources, such as .s37 and .ebl files. In addition to the representation of the application, you can include a representation of the application bootloader of the customer manufacturing tokens.
`em3xx_load`	This utility is used to program selected portions of the flash memory space of the EM3x chips through the SWJ interface.

See document UG107, *EM3x Utilities Guide: For the EM3x Platform*, for detailed information on the EM3x utilities.

10 Debug Adapter (ISA3)

The Debug Adapter (ISA3) provides the programming, debugging, and data emulation capability for an EM3x-based application. The Ember EM3x chip families integrate the ARM® Cortex™-M3, 32-bit microcontroller core. The Debug Adapter (ISA3) converts between the JTAG and Serial Wire (SW) commands, Packet Trace Interface, TCP/IP, and UDP for an easy-to-deploy system over 10/100 Ethernet.

As part of the EM3x development kits, the Debug Adapter (ISA3) connects to the EM3x module through two interfaces: the 10-pin Packet Trace Port and the 12-pin data emulation interface (DEI). These two interfaces provide access to most EM3x GPIO as well as the EM3x programming and debug I/O.

See document *TS7—Ember Debug Adapter (ISA3) Technical Specification*, for more information about the adapter. See the User Guide for your development kit for information about configuring the Debug Adapter (ISA3). See document AN717, *Programming Options for the EM35x* for information about using the Debug Adapter (ISA3) in a production environment.

11 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*.

CONTACT INFORMATION

Silicon Laboratories Inc.

400 West Cesar Chavez
Austin, TX 78701
Tel: 1+(512) 416-8500
Fax: 1+(512) 416-9669
Toll Free: 1+(877) 444-3032

For additional information please visit the Silicon Labs Technical Support page:
http://www.silabs.com/support/Pages/default.aspx