UG103.12

EMBER® APPLICATION DEVELOPMENT FUNDAMENTALS: SILICON LABS CONNECT

This document describes the features and functions of the Silicon Labs Connect stack, including its device types, network topologies, and its “building block” development methodology using plugins.

New in This Revision

Initial release.

Contents

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 Ember 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.

One such challenge is managing devices requiring low power consumption, such as battery-powered devices where long battery life is essential. To meet this challenge Silicon Labs has developed the Silicon Labs Connect stack. Connect provides a fully-featured, easily-customizable wireless networking solution optimized for devices that require low power consumption and are used in a simple network topology. Connect is configurable to be compliant with regional communications standards worldwide. Each RF configuration is designed for maximum performance under each regional standard.

The Silicon Labs Connect stack supports many combinations of radio modulation, frequency and data rates. The stack provides support for end nodes, coordinators, and range extenders. It includes all wireless MAC (Medium Access Control) layer functions such as scanning and joining, setting up a point-to-point or star network, and managing device types such as sleepy end devices, routers, and coordinators. With all this functionality already implemented in the stack, users can focus on their end application development and not worry about the lower-level radio and network details.

The Connect stack should be used in applications with simple network topologies, such as a set of data readers feeding information directly to a single central collection point. It does not provide a full mesh networking solution such as that provided by the ZigBee PRO or Thread stack.

The Connect stack supports efficient application development through its "building block" plug-in design. When used with Silicon Labs Application Builder, developers can easily select which functions should be included in the application. The resulting applications are completely portable, in that they can be recompiled for different regions, different MCUs, and different radios.

2 Devices

The Connect stack supports the following device types:

Coordinator: The coordinator forms and manages the network. The coordinator also communicates with other range extenders and end nodes. Each Connect network has a single coordinator.

Range extender: A device between the coordinator and one or more end nodes that can be used to extend the range of the end nodes. Each range extender can serve up to 32 end nodes.

End node: Joins to a coordinator or a range extender.

An example of a Connect network is a network of temperature and humidity sensor end nodes installed throughout a home. Each end node periodically takes a reading and transmits that data either directly to a coordinator (sink) or, for those sensors placed farther away from the coordinator, to a range extender. The range extenders take data

from the sensors and forward them to the coordinator. The coordinator not only forms and manages the network, but also sends the compiled data to an environmental management system that is part of another network.

Another example is a topology of two minimally-featured nodes that exchange data in both directions. This topology can be used as a generic wire replacement.

3 Network Topology

The Connect stack supports three topologies, shown in Figure 1:

- Point to Point
- Star
- Extended Star

A Point to Point network provides simple communications between two devices: a coordinator (●) and an end node (○).

A Star network has a single coordinator hub communicating with multiple end nodes. All communication is through the coordinator.

An Extended Star network includes a range extender (●) between the coordinator and end node(s) in one or more arms of the star. Communications between the coordinator and the far end node(s) pass through the range extender.

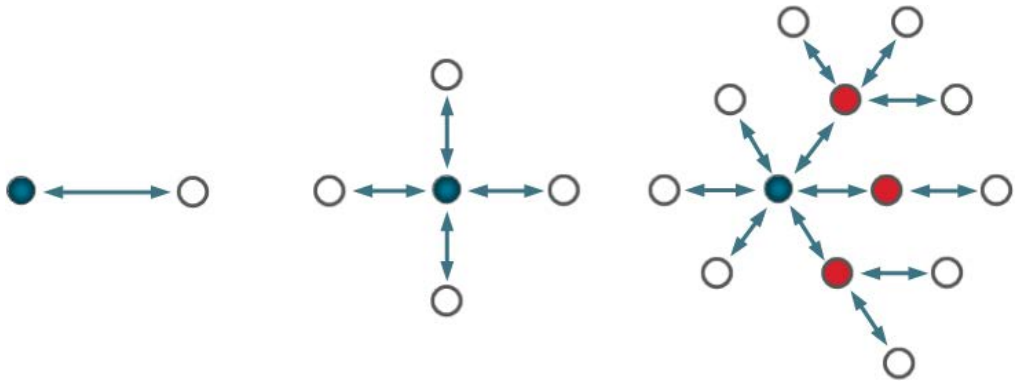

Figure 1.Point to Point, Star, and Extended Star Network Topologies

4 Stack Structure

The Connect stack provides code organized into three functional layers, as shown in Figure 2:

- PHY (physical)
- MAC
- Network

The PHY and MAC layers are based on the IEEE 802.15.4-2006[1] standard. The Network layer is based on a proprietary protocol.

[1] IEEE 802.15.1-2006 Specification, http://standards.ieee.org/findstds/standard/802.15.4-2006.html

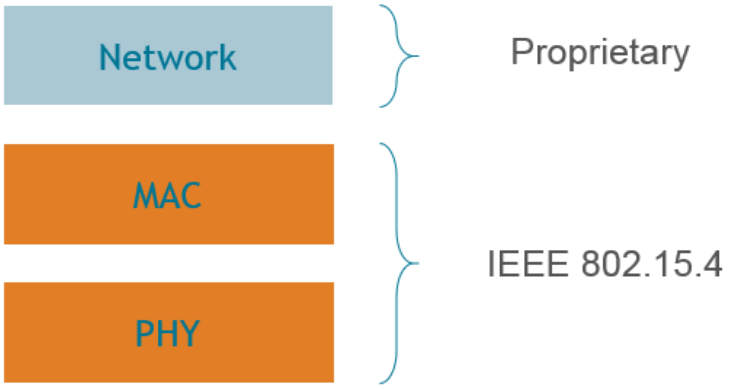

Figure 2. Connect Stack Layers

Finally, the Connect Application Framework provides a complete tool and API infrastructure over the underlying stack layers. Functionality within the Application Framework and the Connect stack layers is provided in the form of individual building blocks called plugins, summarized in section 5, Connect Plugins. Details of the plugins for each layer are provided in the *Silicon Labs Connect Application Framework API Reference* included in the stack documentation.

4.1 PHY and MAC Layers

The IEEE 802.15.4-2006 specification is a standard for wireless communication that defines the MAC and PHY operating at subGHz frequencies as well as at 250kbps in the 2.4GHz band. 802.15.4 was designed with low power in mind.

The 802.15.4 MAC layer is used for basic message handling and congestion control. The network layer builds on these underlying mechanisms to provide reliable end-to-end communications in the network. The MAC layer includes a CSMA (Carrier Sense Multiple Access) mechanism for devices to listen for a clear channel, as well as a link layer to handle retries and acknowledgement of messages for reliable communications between adjacent devices. The MAC layer also provides security functionality (authentication, encryption, and replay attack protection). The MAC auxiliary header indicates which security scheme is used for that packet. Optional security schemes that can be implemented through plugins include XXTEA (Corrected Block Tiny Encryption Algorithm) and AES (Advanced Encryption Standard). The destination node looks at the auxiliary header and uses the correct security scheme (if it supports it) to decrypt and authenticate the incoming packet.

One of the characteristics derived from the need for low power and limiting the BER (Bit Error Rate) is enforcing smaller sized packets to be sent over the air. These can be up to a maximum of 127 bytes at the PHY layer. The MAC layer payload can vary depending on the security options and addressing type as illustrated in Figure 3.

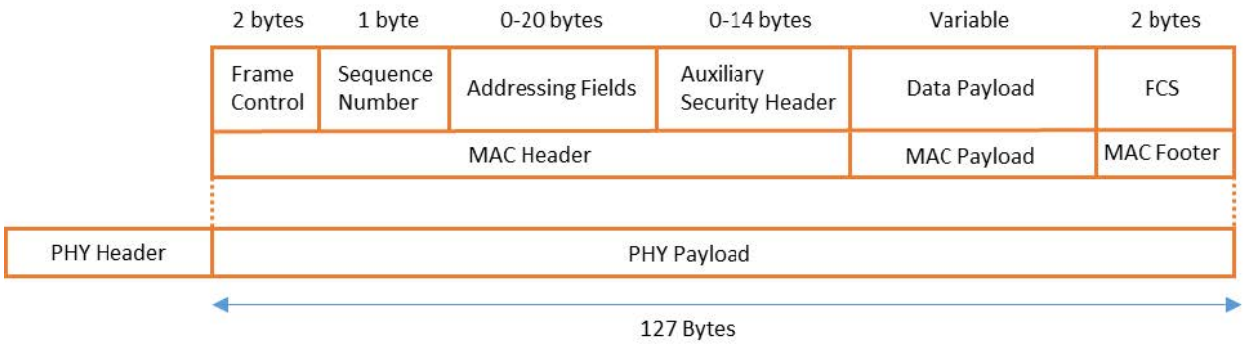

Figure 3. 802.15.4 MAC Payload

Nodes are provided with both short (2-byte) and long (8-byte) identifiers. A network is identified by a 2-byte PAN ID.

4.2 Network Layer

The proprietary network layer provides network formation and full routing support, meaning every node in the network can communicate to any other node in the network in both directions. Routing is transparent to the application layer.

The network formation functionality offers an association mechanism that, while it is similar to that in the 802.15.4 protocol, has been improved and made more secure by providing a special encrypted association request command not present in the 802.15.4 protocol. Network formation also includes centralized address allocation at the coordinator.

4.3 Application Framework

The Connect Application Framework leverages the Ember Application Framework v6 and its bookkeeping functionality implemented in callbacks such as `init()`, `tick()`, and `stackStatus()`. Application Framework plugins can provide callbacks and can implement the callbacks of other plugins.

4.4 Functionality Blocks

Every Connect application includes the following functionality blocks. The HAL (hardware abstraction layer) and Simulated EEPROM functionality blocks reside below the Connect stack. The PHY, event system, and message builder/parser functionality blocks are part of the stack itself.

- **HAL:** Drivers for devices and their peripherals such as SPI (Serial Peripheral Interface), UART (Universal Asynchronous Receiver/Transmitter), timers, and so on.
- **Simulated EEPROM:** Wear-leveled persistent storage of network and application data.
- **Event system:** System that allows the stack and the application to schedule code to run after some specified time interval. Events are also useful when an ISR (Interrupt Service Routine) needs to initiate an action that should run outside the ISR context.
- **PHY:** Software module that interfaces with the transceiver over the SPI bus and provides basic radio TX, RX, and radio sleep functionality.
- **Message builder/parser:** Provides a 15.4-like PHY/MAC packet format builder/parser and a proprietary network layer format builder/parser.

5 Connect Plugins

The Connect stack code has been structured to support optional functionality blocks called plugins. Each plugin is provided as a stand-alone library or set of source code. The corresponding stub library is also provided, and is compiled in place of the full library if the plugin has not been selected during development. Underlying code checks at run time whether libraries are present or not. Common stack code performs some actions conditionally if a library is present or not. Plugins may depend on other plugins. For example, the MAC queue plugin depends on the dynamic memory allocation plugin. Plugin dependencies are managed by Simplicity Studio's Application Builder.

5.1 MAC and Network Layer Optional Plugins

MAC and Network layer optional plugins include the following:

- 802.15.4 Software MAC
- Dynamic memory allocation
- MAC queue
- Child table
- Indirect queue
- Routing table
- Scan
- Network formation
- Security options
    - XXTEA
    - AES

802.15.4 Software MAC: Some transceiver parts support 802.15.4 natively. Other, usually less expensive, devices require the 802.15.4 Software MAC to accomplish similar tasks.

Dynamic memory allocation: A generic lightweight library that provides dynamic memory allocation and garbage collection.

MAC queue: (requires dynamic memory allocation) Some applications need to submit multiple packets to the Connect stack. If this plugin is not selected, the stack can only handle one packet at a time.

Child table: (part of the "parent support" feature-set bundle) Allows a coordinator or a range extender to support multiple children. Children are aged and eventually removed. The child information table is stored in NVM (non-volatile memory).

Indirect queue (part of the "parent support" feature-set bundle; requires child table and dynamic memory allocation): Buffers packets destined to sleepy children. If this plugin is not selected, coordinators and range extenders cannot send packets to sleepy children, although they can still receive packets from them.

Routing table: Needed for coordinator applications if the network includes range extenders. It stores the information collected from range extenders. If this plugin is not selected, only a simple star topology can be supported, in which the coordinator performs routing using only the child table.

Scan: Enables nodes to perform active scans, send 802.15.4 beacon request commands, and collect beacons.

Network formation: (requires scan functionality) Performs over-the-air network form and join operations. If this plugin is not selected, network parameters will be pre-commissioned by the application.

XXTEA-based security: Enables nodes to exchange secured messages using an XXTEA software-implemented encryption algorithm. This plugin enables security for 8-bit parts that have no hardware AES acceleration block. It should be included in applications for 32-bit parts that need to interoperate with 8-bit parts. The Connect over-the-air packet format makes it possible for nodes to support multiple security schemes and distinguish between them at run time.

AES-based security: Enables nodes to exchange secured messages based on the CCM* (Counter with CBC-MAC variant for 802.15.4) standard encryption/authentication scheme. This takes advantage of the AES hardware acceleration block available on 32-bit parts.

5.2 Application Framework Plugins

The Connect Application Framework plugins are used to manage application layer functionality, as follows:

- Main
- Diagnostic
- Utility
  - Idle / Sleep
  - Polling
- I/O
  - Serial
  - Command Interpreter
  - Debug Print
  - Heartbeat
  - WSTK (Wireless Starter Kit) sensors

Main: Defines the `main()` function. It calls all the required the initialization functions. It also implements the stack handlers and dispatches them to every plugin that subscribes to them by calling into the bookkeeping auto-generated callbacks. If the diagnostic plugin was selected, prints out reset information upon reset.

Diagnostic: Provides program counter diagnostic functions and, if the software crashes and the system resets, prints out on the serial port reset information including the call stack.

Utility - Idle/sleep: Manages idle and sleep mode.

- Idle mode: The main application loop is halted while the radio stays on. An incoming packet causes the node to get out of idle mode. Other interrupts are also served.
- Sleep mode: The MCU processing is halted and the radio is disabled.

The plugin includes the logic for determining if and when the device can idle or sleep and initiates idle/sleep whenever it is possible. It attempts to sleep first, but if that is not possible, then it attempts to idle. It queries the stack, Application Framework plugins, and the application.

Utility - Polling: Manages periodic polling for end devices. Regular end devices need to exchange some sort of traffic with the parent as a “keep-alive” mechanism, also referred to as a “long poll interval.” Sleepy end devices need to poll the parent for incoming packets, also referred to as a “short poll interval.” End devices are in long poll mode by default. The application can switch to short poll mode when appropriate using the plugin. For instance, a sleepy end device sends out a packet that expects a response. The application then switches to short poll mode until the expected response is received.

I/O - Serial: Provides high-level read/write serial communication functionality, including `readByte()`, `readData()`, `readLine()`, `writeByte()`, `writeHex()`, `writeString()`, `writeData()`, `writeBuffer()`, `printf()`, `guaranteedPrintf()`, `printfLine()`, `printCarriageReturn()`, and `printVarArg()`. Relies on the HAL low-level UART APIs.

I/O - Command interpreter: Provides a common framework for defining CLI commands and for parsing serial input. Each CLI command is defined by the command string, the set of parameters and the corresponding function to be called when a command and its parameters are successfully parsed. Application developers can easily define a custom set of CLI commands.

I/O - Debug print: Manages each plugin’s `printf()` debug APIs. These APIs are auto-generated by the Application Framework. The application can also define its own debug printf types. This makes it possible to easily

configure which plugins have debug print routines included or excluded and turned on or off at runtime. If this plugin is not selected, the API calls have no effect.

I/O - Heartbeat: For use with the WSTK. Periodically toggles an LED on the WSTK board. The application can set which LED to toggle and the toggling period. Uses the HAL APIs to toggle board LEDs.

I/O - WSTK sensors: The WSTK board is equipped with a temperature and humidity sensor. This plugin provides APIs to read humidity and temperature values. It initializes the sensor and reads values using the low level HAL APIs.

6 Next Steps

See *AN889: Silicon Labs Connect Quick Start Guide* for instructions on using the Application Builder tool and the WSTK to develop a Connect application. *AN902: Building Low Power Networks with the Silicon Labs Connect Stack* provides instructions specific to low-power implementations. Refer to *AN903: Porting Silicon Labs Connect Applications to Customer Hardware* if you plan to implement a Silicon Labs Connect application on devices other than those provided by Silicon Labs.

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