



EmberZnet and WSTK

2019



Welcome to the training. In this training, we will introduce the SDK and starter kits of Silicon Labs's Zigbee solution.

Agenda

- Overview
- Wireless Gecko Socs
- Software
- Development Tools

Here is the agenda of this training. We will start with three aspects, the Socs, software and SDK and development tools.

Zigbee Portfolio



SoCs + Modules

Silicon Labs provides IoT solutions that include hardware, software and tools.
Let's take a look at the Wireless Gecko series SoCs first




 	Mighty Gecko 256 to 1024 kB 40+ Parts	2.4 GHz	2.4 GHz	2.4 GHz	2.4 GHz Sub-GHz
 	Blue Gecko 128 to 1024 kB 50+ Parts			2.4 GHz	2.4 GHz Sub-GHz
	Flex Gecko 32 to 1024 kB 40+ Parts				2.4 GHz Sub-GHz

As you can see from this slide there are 3 different families in the Wireless Gecko portfolio that support different wireless standards.

Mighty Gecko is the superset part. While it is focused on 2.4 GHz mesh, including zigbee and Thread, it supports Bluetooth Low Energy and can support both 2.4 GHz and sub-GHz for proprietary applications. This makes it a great option for customers that want a single design to support multiple options or want the capability for a single product in the field to support multiple protocols. With 256K to 1meg of Flash and over 40 parts, including SoCs and modules, it is the ideal multi-protocol solution.

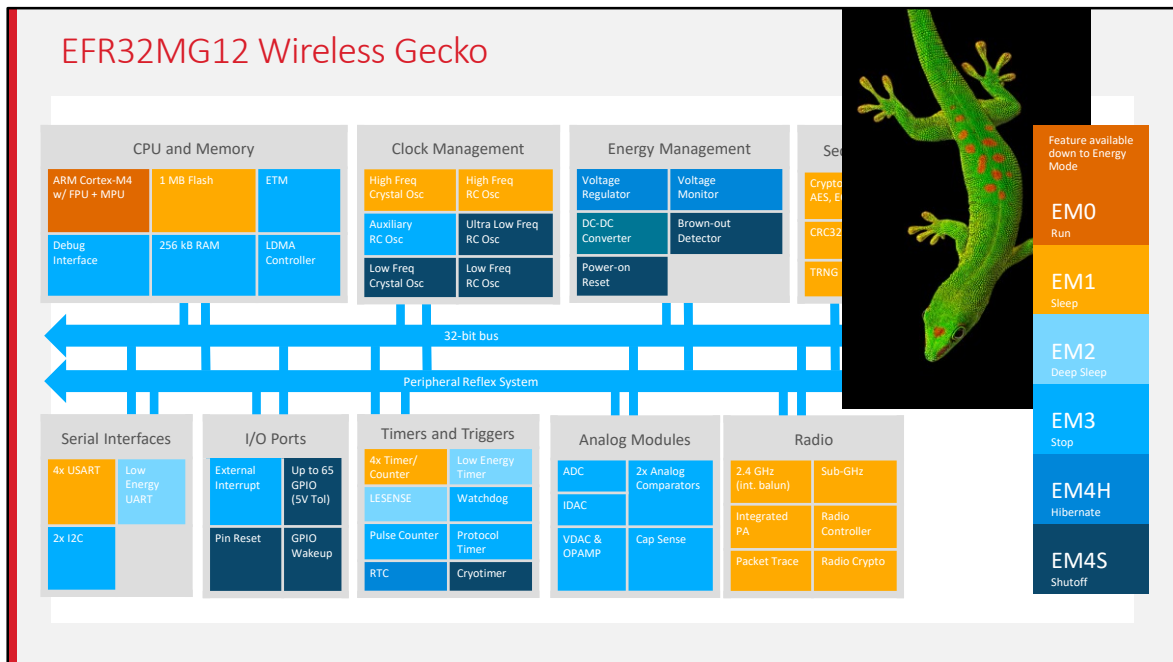
Mighty Gecko SoC Comparison

	EFR32MG1	MG12	MG13	MG21-New
Protocols		+ BT 5 (2 Mbps)	+ BT 5 (2 Mbps & Long Range)	+ BT 5 (2 Mbps)
Freq. Bands	2.4GHz + Sub-GHz	→	→	→
Core	Cortex-M4 (40 MHz)	→	→	Cortex-M33 (80 MHz)
Max Flash	256 kB	1 MB	512 kB	1 MB
Max RAM	32 kB	256 kB	64 kB	96 kB
Security	AES-128/-256, ECC, SHA-1/-2	+ TRNG	→	+ TRNG TrustZone
TX Power	+19 dBm	→	→	→
802.15.4 Sensitivity	-99 dBm	-102.7 dBm	→	-104 dBm
Active Current	~60 µA/MHz	→	→	~50.9 µA/MHz
Sleep Current	2.5 µA	1.5 µA	→	4.5 µA
TX Current @ +0 dBm	8.2 mA	8.5 mA	→	9.3 mA
RX Current (BLE)	8.7 mA	10.0 mA	→	8.8 mA
Operating Voltage	+1.85 - 3.8V	+1.8 - 3.8V	→	+1.71 - 3.8V
Max GPIO	31	65	31	20
Other	IDAC	VDAC, LESENSE, OPAMP, Cap Sense	VDAC, LESENSE, OPAMP, Cap Sense, PLFRCO	

Now let's take a look at the Mighty Gecko series products.

We have four device families in Mighty Gecko series, MG1, MG12, MG13 and MG21. The transmitting power of each can be up to 19dbm.

For MG21 family, we have different flash size from 512KB, 768KB and 1MB.



As you can see from the block diagram, Mighty Gecko is a full featured MCU and wireless SoC.

The part shown is the EFR32MG12 that was launched in March of 2017, but there are other products in the portfolio that provide developers with the exact features set they need.

The MCU is based on Gecko technology so not only does it provide a rich set of peripherals, but also offers exceptional low power capabilities. Low power is more than just low active and sleep states. It is about how the entire system operates in these modes and how quickly it can switch between them. The shaded blocks show what operating modes the feature can operate in. You will notice that a significant amount of blocks can work in the deeper sleep modes, which dramatically reduces power consumption of the system. Features like LESENSE and PRS allow for autonomous operating of peripherals, lowering system power consumption.

With up to 1024k of Flash and 256k of RAM developers have the memory to support not only multi-protocols, but also complex applications. Advanced energy management provides for voltage support for from 1.8 to 3.8 volts and the DCDC ensure low active and sleep currents.

Serial I/O includes up to 4 USARTs, 2 I2C and even a low energy UART that is supported all the way down to deep sleep. The device can support both 16-bit and 32-bit timers with up to 3 PWM per timers.

Features like LESENSE and Pulse counters are ideal for flow meters while the integrated cap sense provides a great option for removing mechanical switches from applications.

The radio has options for both 2.4 GHz and sub-GHz with output power up to +20 dBm and excellent sensitivity eliminates the need for costly front end modules.

And of course, no system is complete without the necessary security blocks to offload the cryptos required for IoT devices.

Wireless Gecko brings together all the critical components for the ultimate IoT hardware platform.

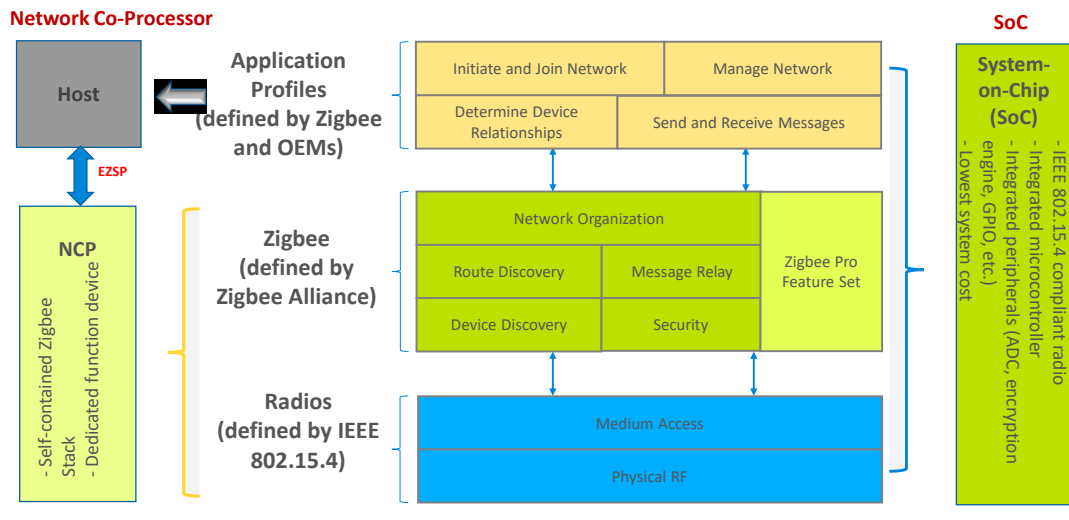
Software



Software

Now let talk about Zigbee software and stack.

Silicon Labs Zigbee Options



Silicon Labs provides two models for Zigbee design: the SoC model, and the NCP model.

In the SoC model, all stack layers as well as the application are implemented on a single chip, with lower level stack functions implemented in hardware as peripherals of the microcontroller.

Access to the stack functionality here is generally provided as library API calls.

In NCP model, the stack and low-level radio functionality all reside on one chip for best integration and efficiency where the stack features are concerned.

However, the application interface to the stack is through a serial interface such as SPI or UART, rather than a library of function calls.

The host and NCP uses a proprietary serial protocol to exchange data. The protocol is named EZSP, which is short for EmberZnet serial protocol.

This model allows for great flexibility on the application design and the host processor architecture. It allows the application designer to ignore many implementation details about the stack itself. You may

Bootloader

Type	
Bootloader-Xmodem-UART	Also called standalone bootloader. Mainly used on NCP.
EZSP-SPI-Bootloader	Similar to the above one, use SPI instead of UART. Mainly used on NCP.
Internal Storage Bootloader	Used on Soc. Store new image in internal flash.
External Storage Bootloader	Used on Soc. Store new image in SPI flash.

Type	Pre-built
Bootloader-Xmodem-UART	platform\bootloader\sample-apps\bootloader-uart-xmodem
EZSP-SPI-Bootloader	platform\bootloader\sample-apps\bootloader-spi-ezsp
Internal Storage Bootloader	platform\bootloader\sample-apps\bootloader-storage-internal-single
External Storage Bootloader	platform\bootloader\sample-apps\bootloader-storage-spiflash-single

For Zigbee projects, we have four types of bootloader. The purpose of using a bootloader is to support upgrading.

First, Bootload-xmodem-uart and EZSP-SPI-Bootloader, also called as standalone bootloader. Normally used in NCP. When ncp needs to upgrade, it will reset and stay at bootloader stage. Then the host will transfer the new ncp image through xmodem and overwrite the current ncp image.

Then we have internal storage bootloader and external storage bootloader. These two are mainly used for Soc applications. The difference is where the new image is saved. With internal storage bootloader, new image will be saved in internal flash. With external storage bootloader, new image will be save in external flash.

We have already provided some pre-built bootloader for some of our starter kits.

EmberZnet SDK

	Resources
v2.6 <ul style="list-style-type: none">.git.root.studio.vscodeapphardwareinstallersmetaplatformprotocol<ul style="list-style-type: none">bluetoothflecthreadzigbee<ul style="list-style-type: none">app<ul style="list-style-type: none">emv260xncp-hostframework<ul style="list-style-type: none">clicincludepluginplugin-hostplugin-socsenariossecurityutilgpdhal-confignrcptestutilxmcpxmcp-test-harnessbuilddocumentationeccnrcp-imagesstackkool	protocol\zigbee\dokumentation protocol\zigbee\app\framework\plugin protocol\zigbee\app\framework\plugin-host protocol\zigbee\app\framework\plugin-soc

This is the directory hierarchy of the SDK.

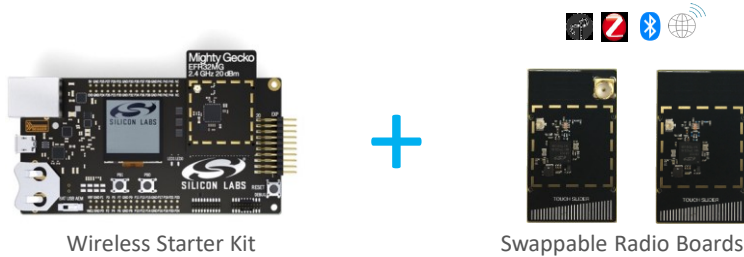
1. In the documentation folder, you can find many documents of our SDK. Those are good learning materials.
2. Most part of the SDK are provided by plugins, and most of the plugins are open-sourced.

Development Tools



Now I will introduce our develop tools.

Wireless Starter Kit



Silicon Labs has a great development platform based on the Wireless STK and radio boards.

The Wireless STK provides the starting point for development providing the hardware and access to mesh software.

Different radio boards for both SoCs and modules plug into the WSTK main boards, providing a unified development platform from both the hardware and software perspective.

The mother board of the starter kit can be used as a debugger. It can be used to debug custom board as well as our develop kits.

Tools



Simplicity IDE

Launches the Simplicity IDE



Application Builder

Create an embedded software framework application



Hardware Configurator

Hardware Configurator is a peripheral, pin and crossbar configuration tool that generates initialization code organized into modes



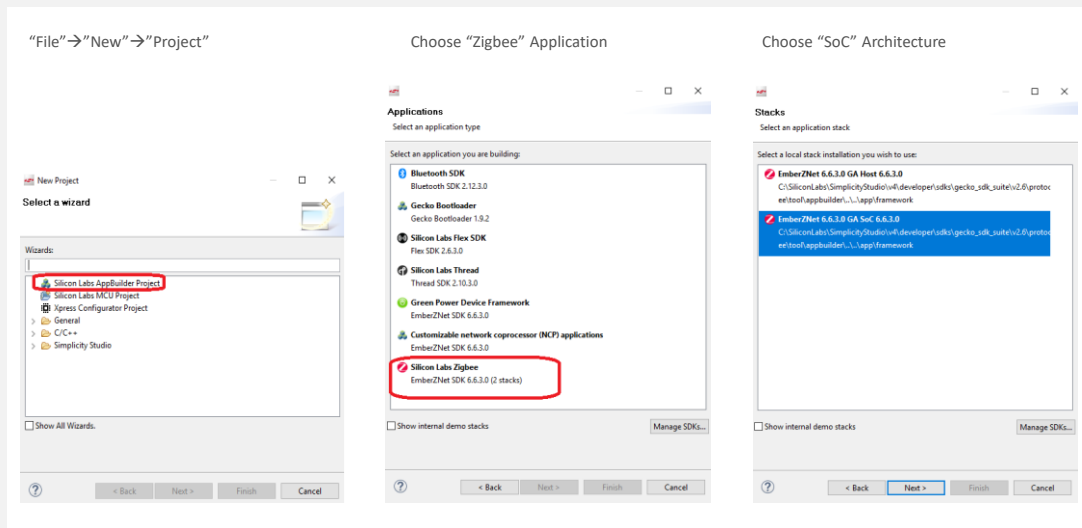
Network Analyzer

Tools for capturing and analyzing network activity

Then we will talk about our tools. Our IoT projects are developed through Simplicity Studio, which is a super IDE, and many useful tools are integrated in it.

For Zigbee applications, we usually use AppBuilder, Hardware Configurator and Network Analyzer.

Create a Zigbee Project – 1/2



First I'll show you how to create a Zigbee project.

Create a Zigbee Project – 2/2

The image displays three sequential screenshots of the Zigbee IDE project creation wizard, illustrating the steps from selecting a sample application to configuring the project name and board.

Choose example or start from minimal

Select either a blank application or a sample application.

Sample Application

- DynamicMultiprotocolLightSeed**
This is a sample application demonstrating a sleepy light application using dynamic multiprotocol (Zigbee + BLE) and NVM3 for persistent storage. Requires IAR.
- DynamicMultiprotocolSwitch**
This is a Zigbee switch application using NVM3 designed to work with the dynamic multiprotocol demonstration light. Requires IAR.
- Z3Light**
This is a Zigbee 3.0 light application using NVM3 as the persistent storage.
- Z3LightGFCCombo**
This is a Zigbee 3.0 light application with Green Power endpoint, Green Power Proxy and Sink functionality.
- Z3Switch**
This is a Zigbee 3.0 switch application using NVM3 as the persistent storage.
- ZigbeeMinimal**
This is a Zigbee minimal network-layer application suitable as a starting point for new application development.

☐ Start with a blank application

< Back Next > Finish Cancel

Name project

Select the project name and location.

Project name:

☒ Use default location
Location:

With project files:
☒ Link libraries and copy sources

< Back Next > Finish Cancel

Choose "Board"/"Part" and toolchain

Select the board, part, and initial build configurations.

Boards:

Part:

Check the configurations to include in the project

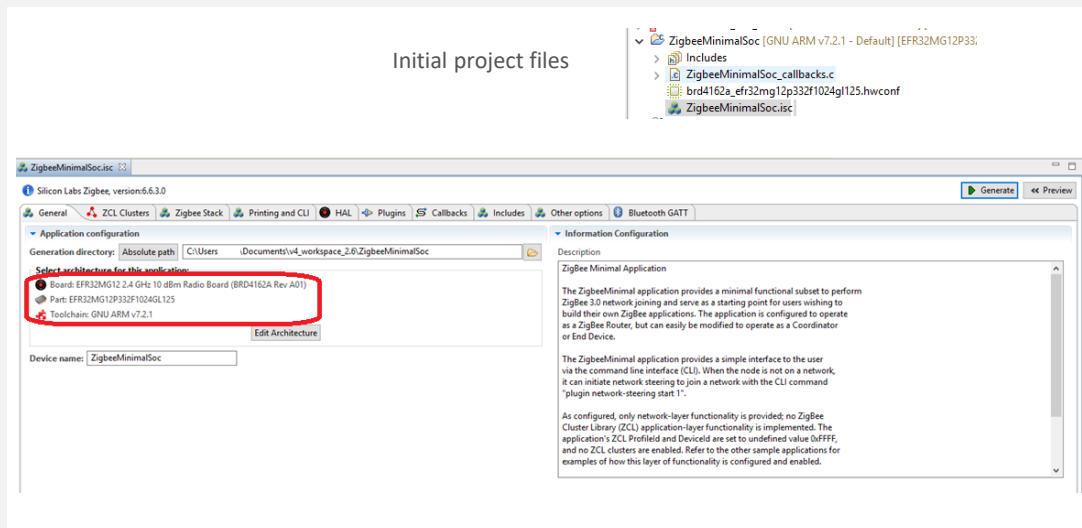
<input checked="" type="checkbox"/> GNU ARM v7.2.1	<input type="button" value="Select All"/>
<input checked="" type="checkbox"/> Default (active)	<input type="button" value="Select None"/>
<input checked="" type="checkbox"/> IAR ARM v6-45.1.212	<input type="button" value="Set Active"/>
<input checked="" type="checkbox"/> Default	

[Manage toolchains...](#)
[Manage build targets...](#)

< Back Next > Finish Cancel

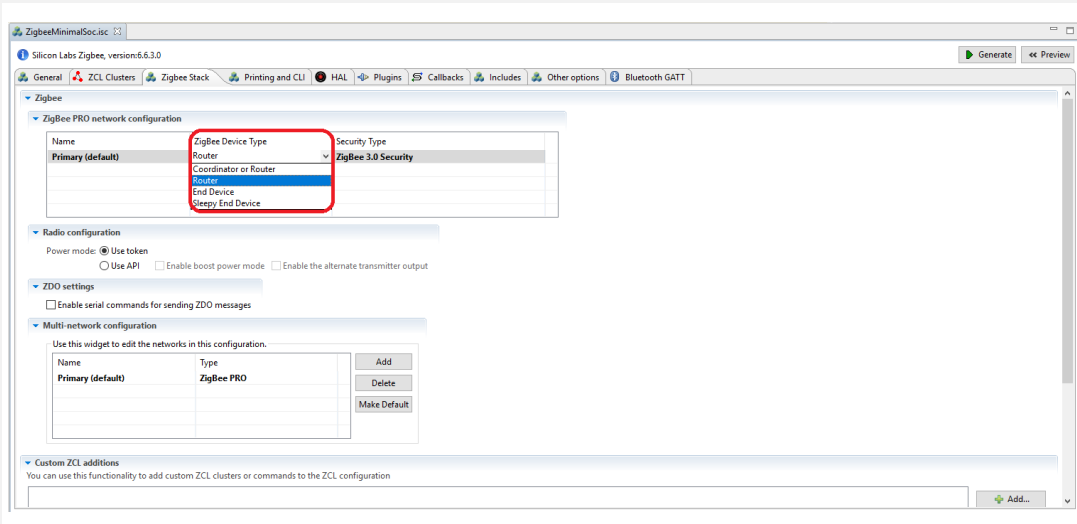
In the "Boards" field, if you are using a starter kit, you can just select one from the list. If you are using a custom board and you are familiar with Silicon Labs's solution, you can leave this field empty and just select the part. Otherwise, you should select a starter kit which has the closest part with yours.

AppBuilder - General



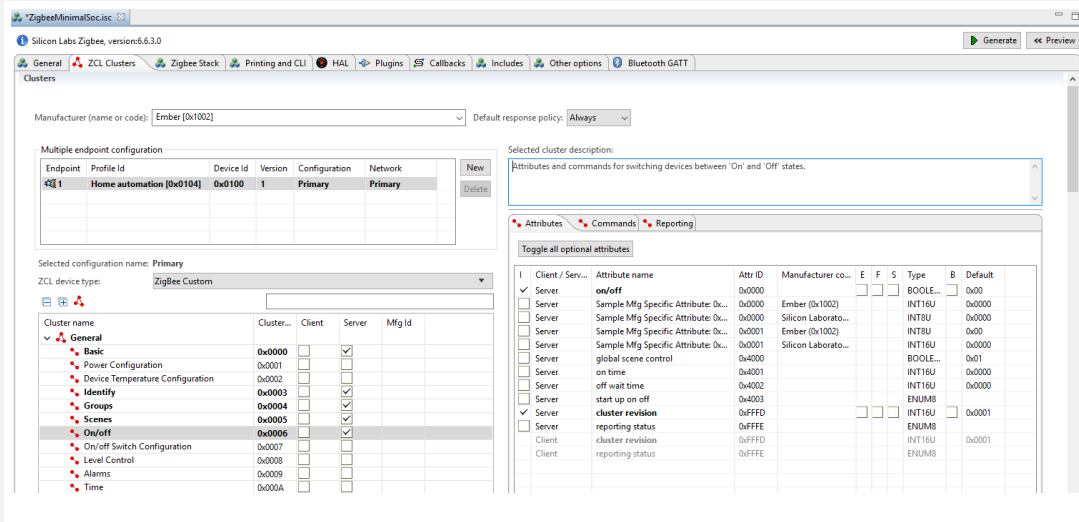
In "General" tab, you can see the board/part and toolchain. You can also change them here. Just remember that you need to save and generate the project after you changed.

AppBuilder - Stack



In the “stack” tab, you can choose the device type. Here you can choose it as a coordinator, router, end device or sleepy end device.

AppBuilder – Cluster Attributes



In “ZCL Clusters” tab, you can configure the endpoint and clusters in the endpoint. Each cluster has a client side and a server side. You need to select the right side according to your design.

You would also need to select the corresponding attributes.

AppBuilder – Cluster Commands

The screenshot shows the ZigBeeMinimaSoc AppBuilder interface. The top bar includes tabs for General, ZCL Clusters, ZigBee Stack, Printing and CLI, HAL, Plugins, Callbacks, Includes, Other options, and Bluetooth GATT. The 'ZCL Clusters' tab is active.

Clusters

Manufacturer (name or code): Default response policy:

Multiple endpoint configuration

Endpoint	Profile Id	Device Id	Version	Configuration	Network
1	Home automation [0x0104]	0x0100	1	Primary	Primary

Selected configuration name: Primary

ZCL device type:

Cluster list:

Cluster name	Cluster...	Client	Server	Mfg Id
General				
Basic	0x0000	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
Power Configuration	0x0001	<input type="checkbox"/>	<input type="checkbox"/>	
Device Temperature Configuration	0x0002	<input type="checkbox"/>	<input type="checkbox"/>	
Identify	0x0003	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
Groups	0x0004	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
Scenes	0x0005	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
On/Off	0x0006	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
On/Off Switch Configuration	0x0007	<input type="checkbox"/>	<input type="checkbox"/>	
Level Control	0x0008	<input type="checkbox"/>	<input type="checkbox"/>	

Selected cluster description:

Attributes and commands for switching devices between 'On' and 'Off' states.

Attributes **Commands** **Reporting**

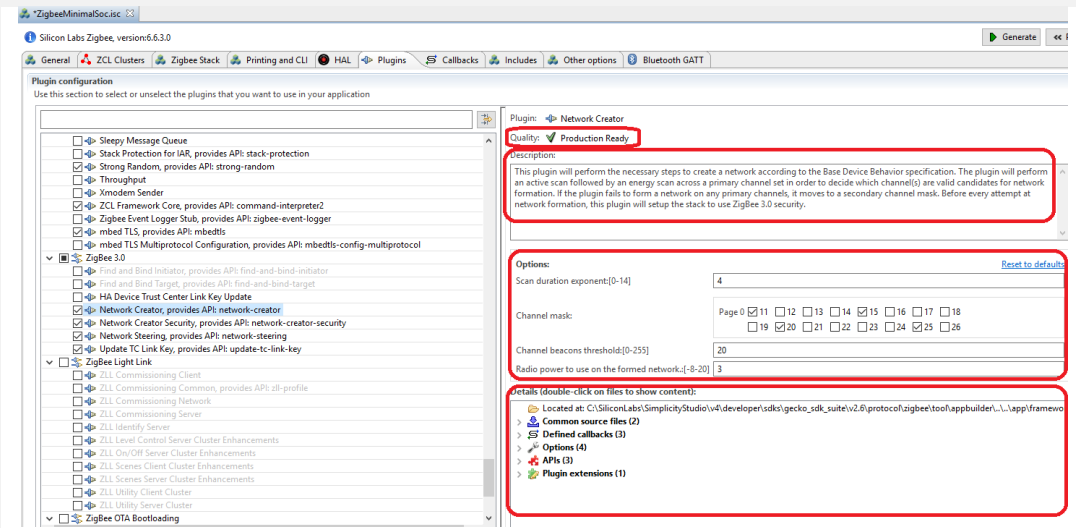
☒ **Enable command discovery?**

Commands reported during command discovery for a given cluster. If you have implemented command handling, toggle the 'In' column. If you have implemented command sending, toggle the 'Out' column.

Out	In	Direction	Command name	Cmd ...	Mfg ID
<input type="checkbox"/>	<input type="checkbox"/>	c → s	N Off = 0	0x00	
<input type="checkbox"/>	<input type="checkbox"/>	c → s	N On = 0	0x01	
<input type="checkbox"/>	<input type="checkbox"/>	c → s	N Toggle = 0	0x02	
<input type="checkbox"/>	<input type="checkbox"/>	c → s	Y OffWithEffect = 0	0x40	
<input type="checkbox"/>	<input type="checkbox"/>	c → s	Y OnWithRecallGlobalScene = 0	0x41	
<input type="checkbox"/>	<input type="checkbox"/>	c → s	Y OnWithTimedOff = 0	0x42	
<input type="checkbox"/>	<input type="checkbox"/>	c → s	Y SampleMfgSpecificOffWithTransi...	0x00	0x1002
<input type="checkbox"/>	<input type="checkbox"/>	c → s	Y SampleMfgSpecificOnWithTransi...	0x01	0x1002
<input type="checkbox"/>	<input type="checkbox"/>	c → s	Y SampleMfgSpecificToggleWithTr...	0x02	0x1002
<input type="checkbox"/>	<input type="checkbox"/>	c → s	Y SampleMfgSpecificOnWithTransi...	0x01	0x1049
<input type="checkbox"/>	<input type="checkbox"/>	c → s	Y SampleMfgSpecificToggleWithTr...	0x02	0x1049

You would also need to select the corresponding commands of the selected clusters.

AppBuilder - Plugins



As we talked before, the SDK is provided by many plugins. You can select the plugins you need in the “plugins” tab.

In the right side of the plugin, you can see the status of this plugin. Also the description of this plugin.

There might be some options of the plugin.

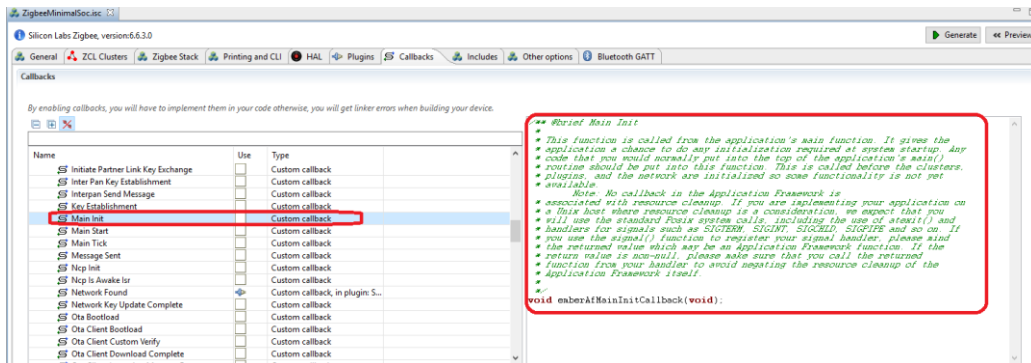
In the right bottom. You can find there is some properties of the plugin, like the source or library path of the plugin, the callbacks implemented and defined. You can also get the dependency of the plugin by the APIs field.

Frequently Used Plugins

Category	Plugin	Comments
Core Stack	Zigbee PRO Stack Library	Core stack with routing support, used by routes and coordinator
	Zigbee PRO Leaf Library	Core stack without routing support, used by end devices
	End Device Support	A plugin to support end devices
	Network Creator	Create network, used by coordinator
	Network Creator Security	Security settings for coordinator, such as configuring link key for new device
	Network Steering	Scanning joinable networks and join
Sleeping	Idle/Sleep	Used by sleepy end device. Device will enter EM2 when idle.
	EM4	A plugin helps sleepy end device enter EM4
Main Entry	Simple Main	Main entry of the project
Non-volatile Data	Simulate EEPROM Version 1 Library	Library of Simulate EEPROM Version 1, used to store non-volatile data
	Simulate EEPROM Version 2 Library	Library of Simulate EEPROM Version 2, used to store non-volatile data
	NVM3 Library	Library of NVM3, used to store non-volatile data

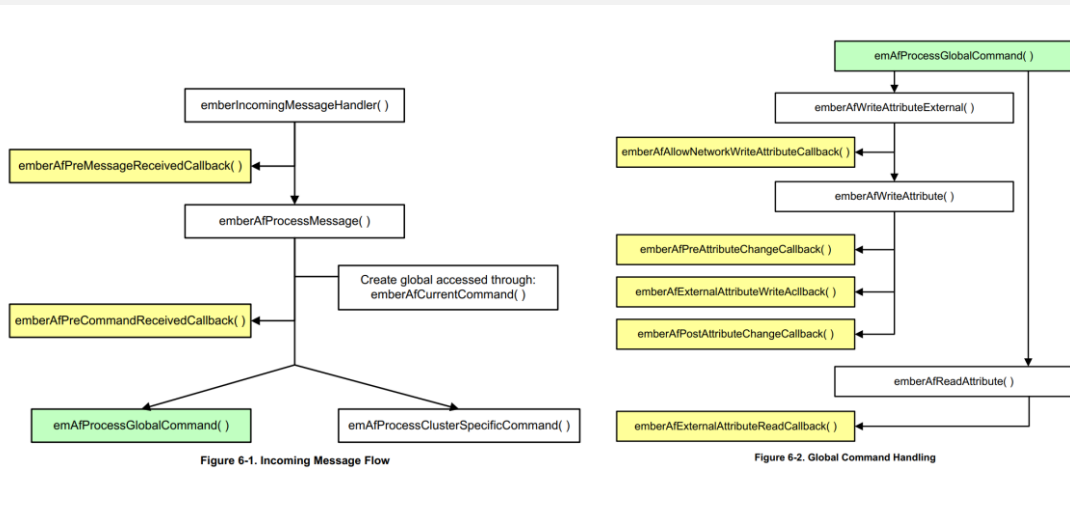
This is the most commonly used plugins.

AppBuilder - Callbacks



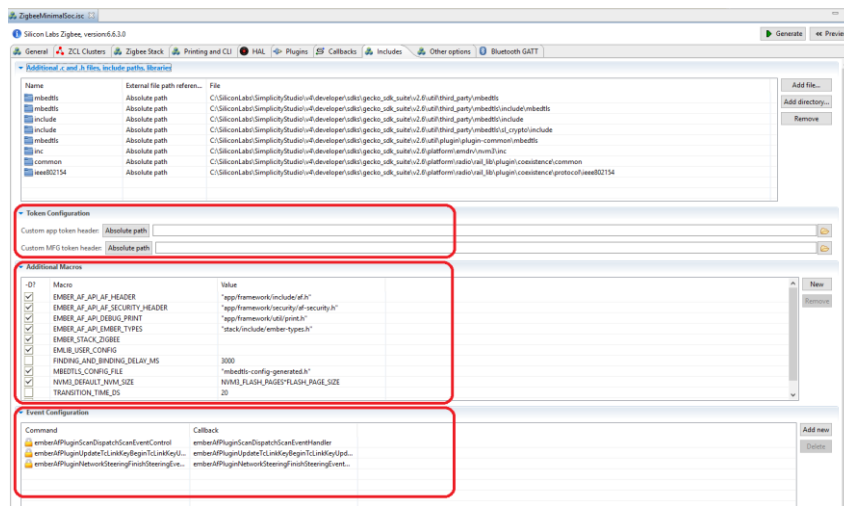
Silicon Labs recommends you to use callbacks to implement your application. Actually most of the source code has been finished by the framework. You can add your custom source code in the callbacks if you need.

Callbacks Flow



This is the call flow of the callbacks.

AppBuilder – Custom settings



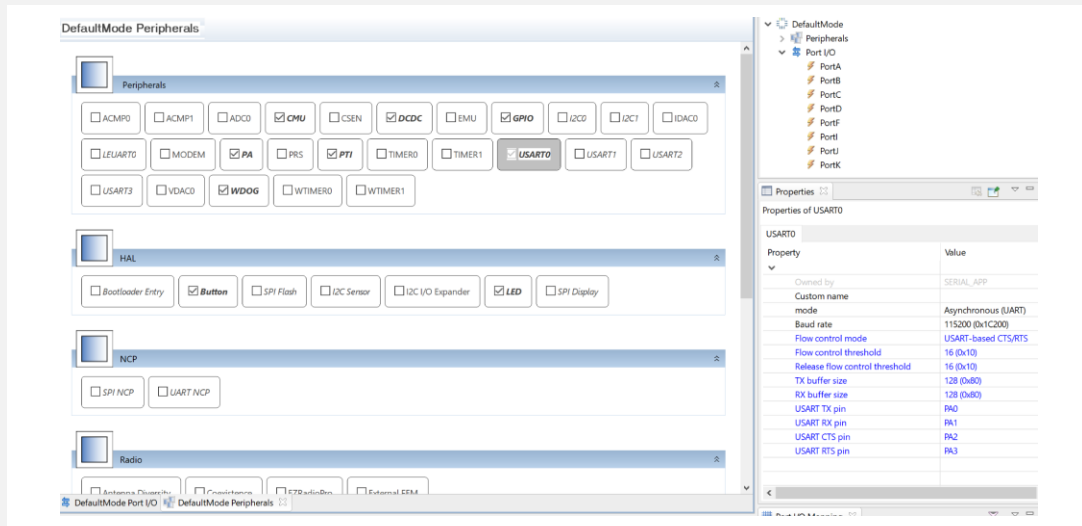
In “Includes” tab, you can add your custom macros, event and custom tokens. In the hands-on part this afternoon, you will use them.

Generated Files

File	Description
<projectname>.h	main header file, plugin settings
<projectname>_callbacks.c	Customer implementation
<projectname>_endpoint_config.h	defines the endpoints and attributes
znet-cli.c/znet-cli.h	CLI command list
client-command-macro.h	macros which will be used in filling messages
call-command-handler.c	Cluster command process
attribute-id.h/attribute-size.h/attribute-type.h/att-storage.h	Attribute related
af-structs.h	Data structs
af-gen-event.h	Event/handler pair
hal-config.h	All hardware settings are generated in this file

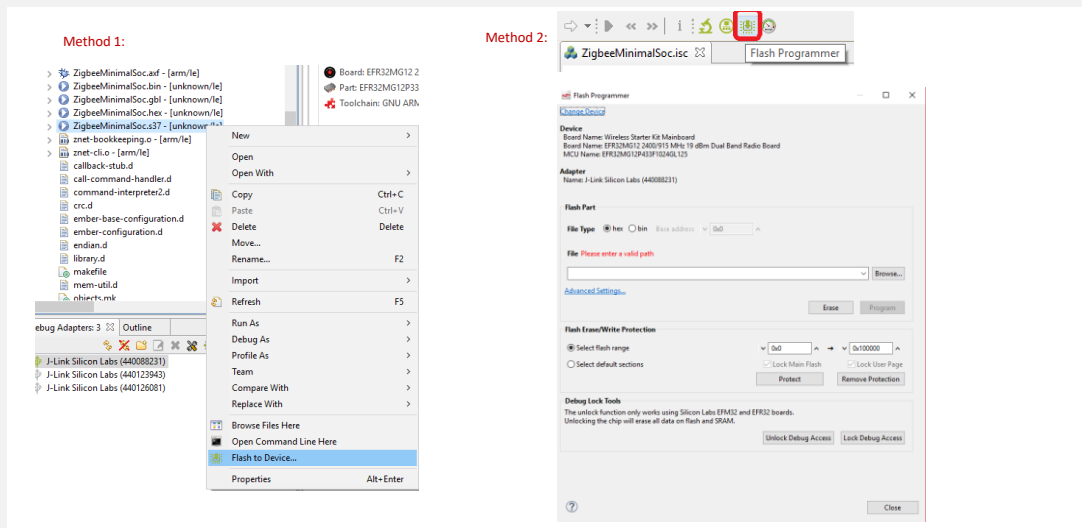
Show the generated files.

Hardware Configurator



For custom board, you would need to change the hardware configuration according to your schematic.

Flash/Program



Let me show you the approaches of flashing programs.

CLI Commands

- **plugin network-creator form [useCentralizedSecurity:1] [panId:2] [radioTxPower:1] [channel:1]**
 - *Form a network with specified parameters.*
 - useCentralizedSecurity - BOOLEAN - Whether or not to form a centralized network. If this value is false, the device will attempt to join a distributed network.
 - panId - INT16U - PanID of the network to be formed
 - radioTxPower - INT8S - Tx power of the network to be formed
 - channel - INT8U - channel of the network to be formed
- **plugin network-creator-security open-network**
 - *Open the network for joining.*
- **plugin network-creator-security open-with-key [eui64:8] [joiningLinkKey:-1]**
 - *Open the network that would only allow the node with specified EUI and link key pair to join.*
 - eui64 - IEEE_ADDRESS - The EUI64 of the joining device.
 - joiningLinkKey - OCTET_STRING - The link key that the joining device will use to enter the network.

Here are some useful commands you would use during debugging and testing. First, the command to create a network. You would need to specify four parameters, like the security model, PAN ID, power and channel.

Then you will need to open the network for new device to join.

With the first command, you tell the coordinator to use the well-known link key for new devices.

With the second command, you tell the coordinator to use the specified link key for the new device.

CLI Commands

- **plugin network-steering start [options:1]**
 - *Starts the network steering process.*
 - options - INT8U - A mask of options for indicating specific behavior within the network-steering process.
- **zcl on-off toggle**
- **send [id:2] [src-endpoint:1] [dst-endpoint:1]**
 - *Send a pre-buffered message from a given endpoint to an endpoint on a device with a given short address.*
 - id - INT16U - short id of the device to send the message to
 - src-endpoint - INT8U - The endpoint to send the message from
 - dst-endpoint - INT8U - The endpoint to send the message to

With the network-steering command, you can start joining.

With the zcl command, you fill a message buffer with the command.

With the send command, you send the filled message buffer out.

Debug

- `emberAfCorePrint(...)` - prints a single line without a carriage return
- `emberAfCorePrintln(...)` - prints a single line with a carriage return
- `emberAfCorePrintBuffer(buffer, len, withspace)` – prints a given buffer as a series of hex values
- `emberAfCorePrintString(buffer)` – prints a given buffer as a string of characters

To debug, you can use these print functions to print debug info in your application.

Network Analyzer

Start Capture

The screenshot displays the Network Analyzer software interface. On the left, a sidebar shows 'Debug Adapters: 3' with a tree view containing 'J-Link Silicon Labs (4400882...)', 'J-Link Silicon Labs (4401235...)', and 'J-Link Silicon Labs (4401260...)'. A context menu is open over the first adapter, with 'Start capture' highlighted. The main window shows a 'Real time/PA Nodes: 1' status bar and a table of captured packets. The table has columns for Time, Duration, Summary, MAC Src, MAC Dest, PA, LMP, EP, Error Status, and Warning Status. The packets are listed in a table with alternating red and blue rows. The right sidebar shows a detailed view of the selected packet, including fields like 'IEEE 802.15.4 (8 bytes)', 'IEEE 802.15.4 Beacon (4 bytes)', 'Superframe Specification (0xFFFF)', 'Beacon Order (0xFF)', 'Superframe Order (0x01)', 'Frame CAP (0x00)', 'Battery Life Extension (0x00)', 'Piggybacked MAC', 'Frame M Association Page', 'GTS Specification (0x00)', 'GTS Descriptor Count (0x00)', 'GTS Permit Page', 'Pending Address Specification (0x00)', 'Short Address Pending (0x00)', 'Long Addresses Pending (0x00)', 'ZigBee Beacon (15 bytes)', 'Permitted to ZigBee (0x00)', 'Stack Profile ZigBee Pro (02)', 'Network Protocol Version (0x02)', 'Sequence Number (0x00)', 'Length (0x01)', 'Frame Control Page (0x00)', 'Extended PAN ID (0x1010101010101010)', 'To Offset (0x0000)', 'Next Update (0x0000)', 'Radio Info (0x0017 (17 bytes))', 'Cntrl (0x00)', 'Mac Data Success (0x00)', 'Error (0x00000000)', 'Radio Info', 'Antenna Select (0x00)', 'Sync Word Select (0x00)', 'Channel Number (15.4 Gp channel 15, 2.4)', 'Status (0x00)', 'Error Code Success (0x00)', 'Protocol ID ZigBee on RAIL (0)', 'Init Configuration (0x00)', 'Tally Indicator Rx (1)', 'Appended Info Length (0x00)', 'Appended Info Version (0x00)'. The bottom of the interface shows a 'Sniffer Configurator...' button and a 'Launch Console...' button.

Time	Duration	Summary	MAC Src	MAC Dest	PA	LMP	EP	Error Status	Warning Status
0.000000	0.000	Transmit Key (0x00)	271F	2044	2				
0.000000	0.004	Turned Data	0000	271F	1				
0.000000	0.003	Transmit Key (0x00)	271F	2044	2				
0.000000	1.000	Permit Join Request	0000	FFFF	6				
0.000000	0.000	Device Announcement	2044	FFFF	6				
0.000000	0.001	Permit Join Request	0000	FFFF	5				
0.000000	0.005	Many-to-One Route Discovery	0000	FFFF	10				
0.000000	0.003	Permit Join Request	0000	FFFF	5				
0.000000	0.003	Permit Join Request	0000	FFFF	5				
0.000000	0.000	Match Description Request	0000	2044	5				
0.000000	0.000	Match Request	2044	0000	6				
0.000000	0.000	Match Request	2044	0000	5				
0.000000	0.004	Match Description Response	2044	0000	5				

Finally I will show you how to use Network Analyzer to start capture. It's the most useful approach to debug a network issue.

Thank you!

