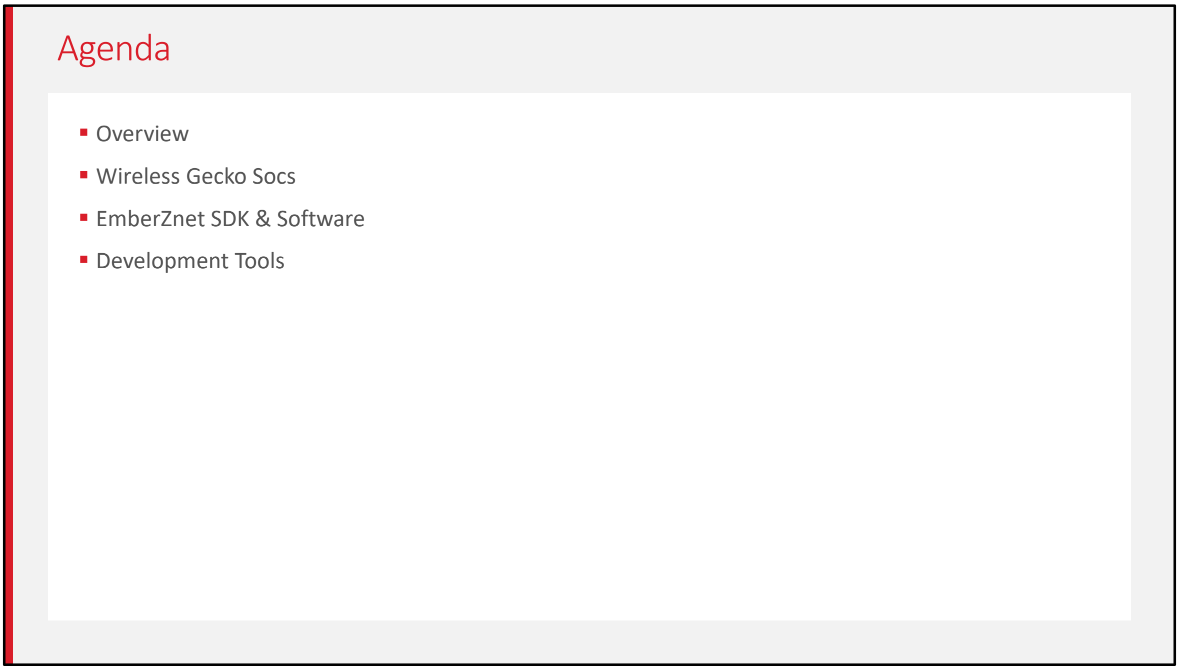
SILICON LABS

EmberZnet and WSTK

2019

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



Agenda

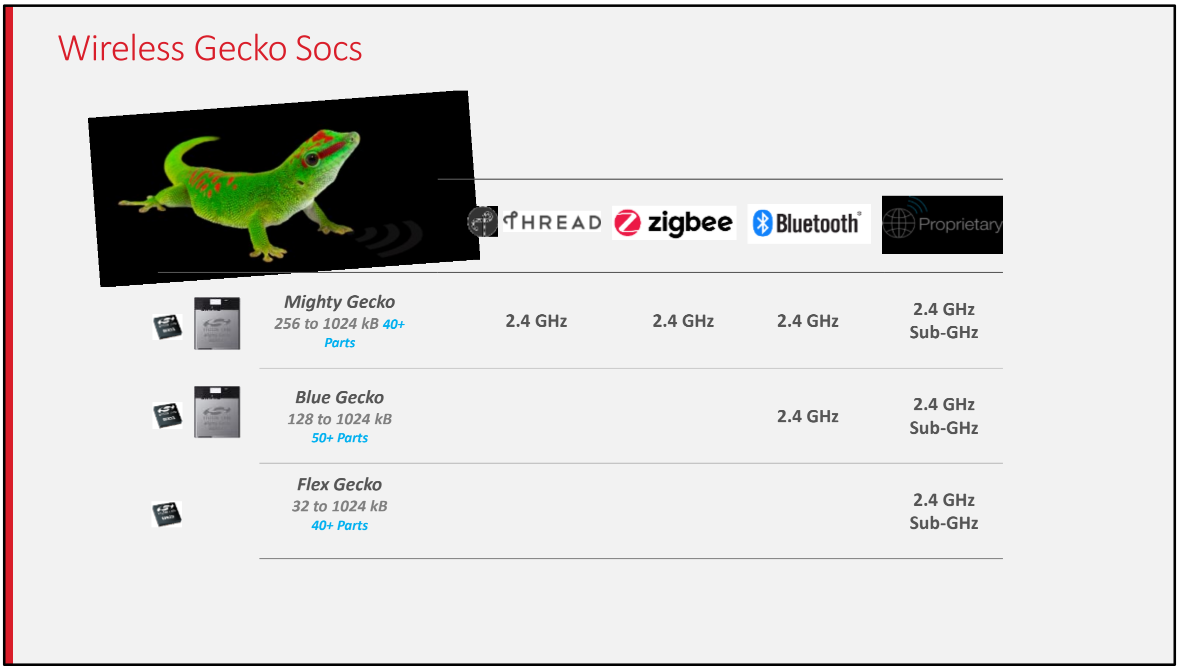
- Overview
- Wireless Gecko Socs
- EmberZnet SDK & Software
- Development Tools

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.
Lets take a look at the Wireless Gecko series Socs first

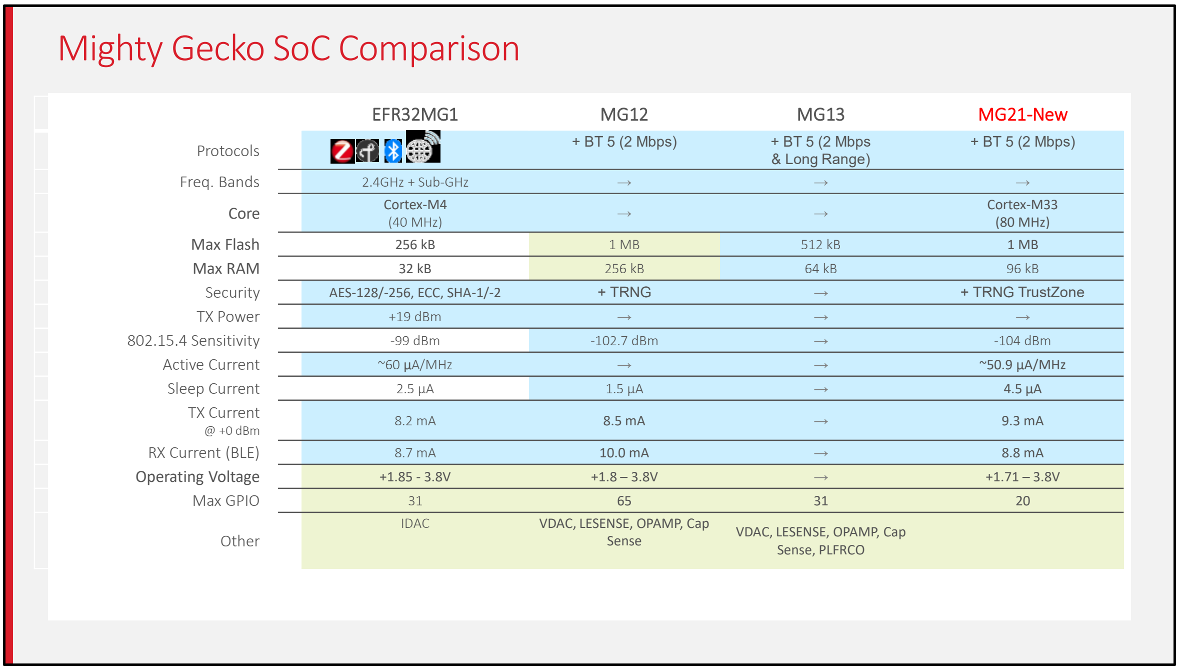


Wireless Gecko Socs

	THREAD	zigbee	Bluetooth	Proprietary
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.



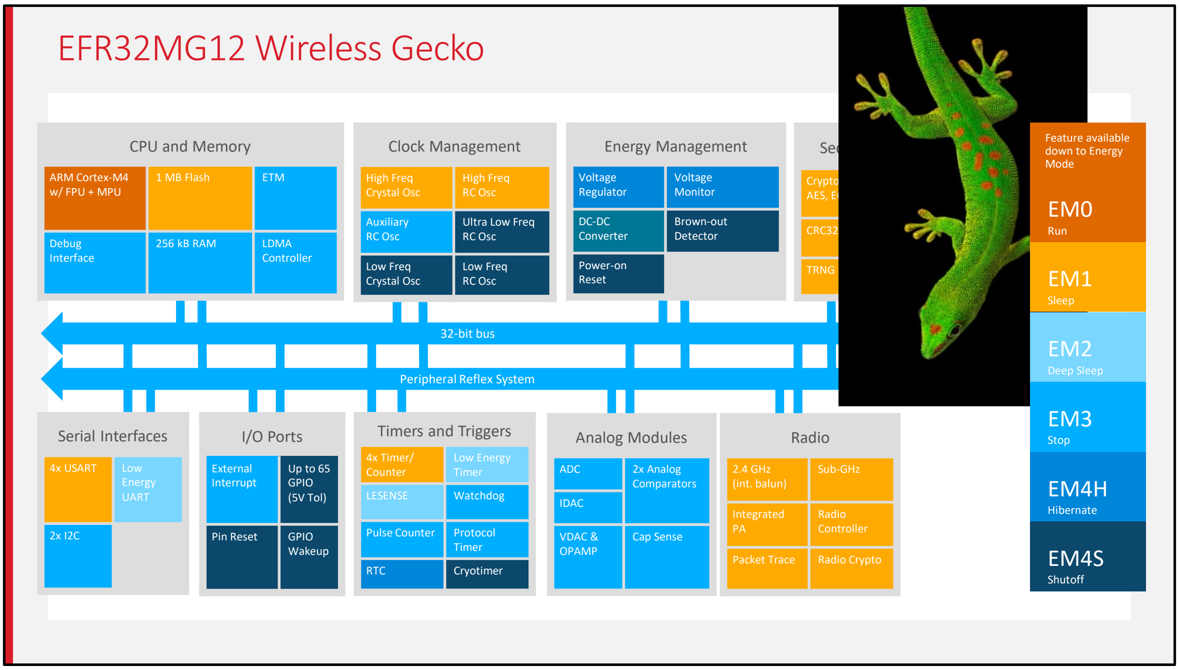
Mighty Gecko SoC Comparison

	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.



EFR32MG12 Wireless Gecko

CPU and Memory

ARM Cortex-M4 w/ FPU + MPU

1 MB Flash

ETM

Debug Interface

256 kB RAM

LDMA Controller

Clock Management

High Freq Crystal Osc

High Freq RC Osc

Auxiliary RC Osc

Ultra Low Freq RC Osc

Low Freq Crystal Osc

Low Freq RC Osc

Energy Management

Voltage Regulator

Voltage Monitor

DC-DC Converter

Brown-out Detector

Power-on Reset

Se

Crypto AES, E

CRC32

TRNG

32-bit bus

Peripheral Reflex System

Serial Interfaces

4x USART

Low Energy UART

2x I2C

I/O Ports

External Interrupt

Up to 65 GPIO (5V Tol)

Pin Reset

GPIO Wakeup

Timers and Triggers

4x Timer/ Counter

Low Energy Timer

LESENSE

Watchdog

Pulse Counter

Protocol Timer

RTC

Cryotimer

Analog Modules

ADC

2x Analog Comparators

IDAC

VDAC & OPAMP

Cap Sense

Radio

2.4 GHz (int. balun)

Sub-GHz

Integrated PA

Radio Controller

Packet Trace

Radio Crypto

Feature available down to Energy Mode

EM0 Run

EM1 Sleep

EM2 Deep Sleep

EM3 Stop

EM4H Hibernate

EM4S Shutoff

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 then 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.

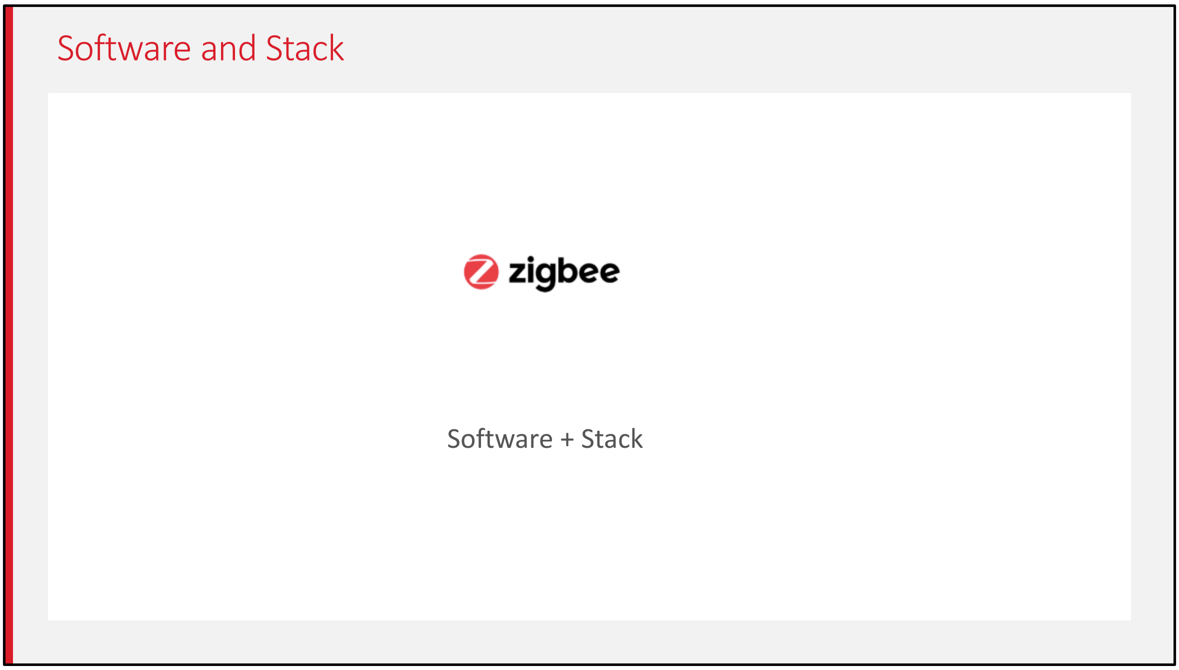
Serial I/O includes up to 4 USARTs, 2 I2C and even a low energy UART that is supported all the way down to deep sleep. The device can support both 16-bit and 32-bit timers with up to 3 PWM per timers.

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.

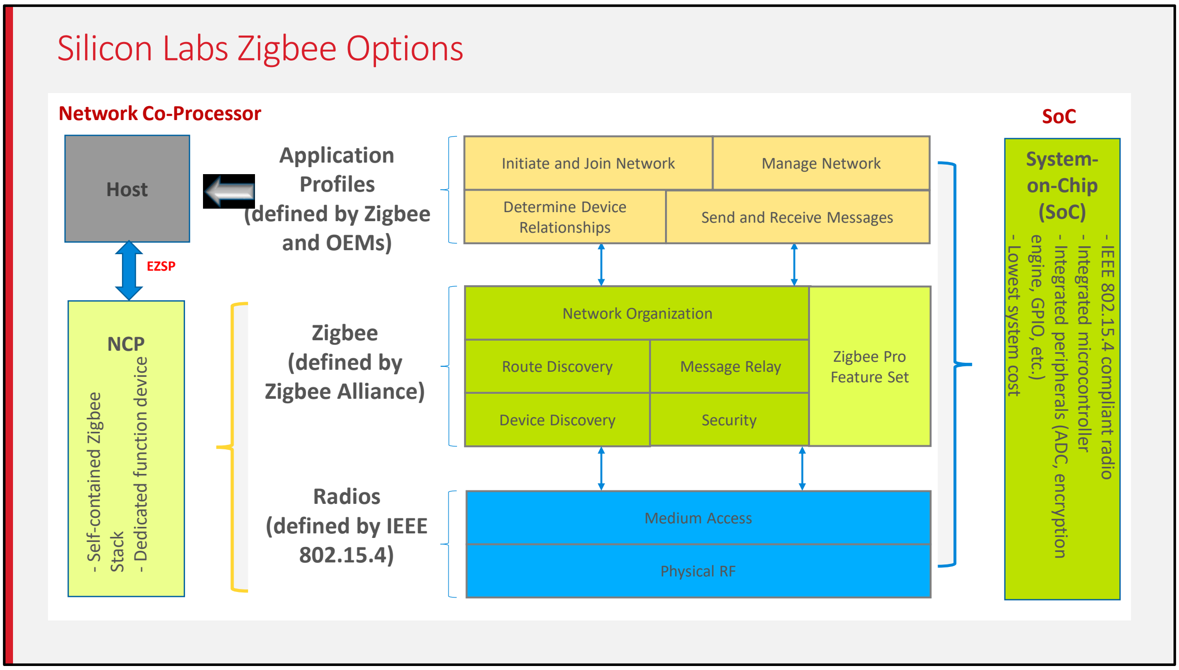


Software and Stack

zigbee

Software + Stack

Now let talk about Zigbee software and stack.



Silicon Labs Zigbee Options

Network Co-Processor

Host

EZSP

NCP

- Self-contained Zigbee Stack
- Dedicated function device

Application Profiles (defined by Zigbee and OEMs)

Zigbee (defined by Zigbee Alliance)

Radios (defined by IEEE 802.15.4)

Initiate and Join Network

Manage Network

Determine Device Relationships

Send and Receive Messages

Network Organization

Route Discovery

Message Relay

Device Discovery

Security

Zigbee Pro Feature Set

Medium Access

Physical RF

SoC

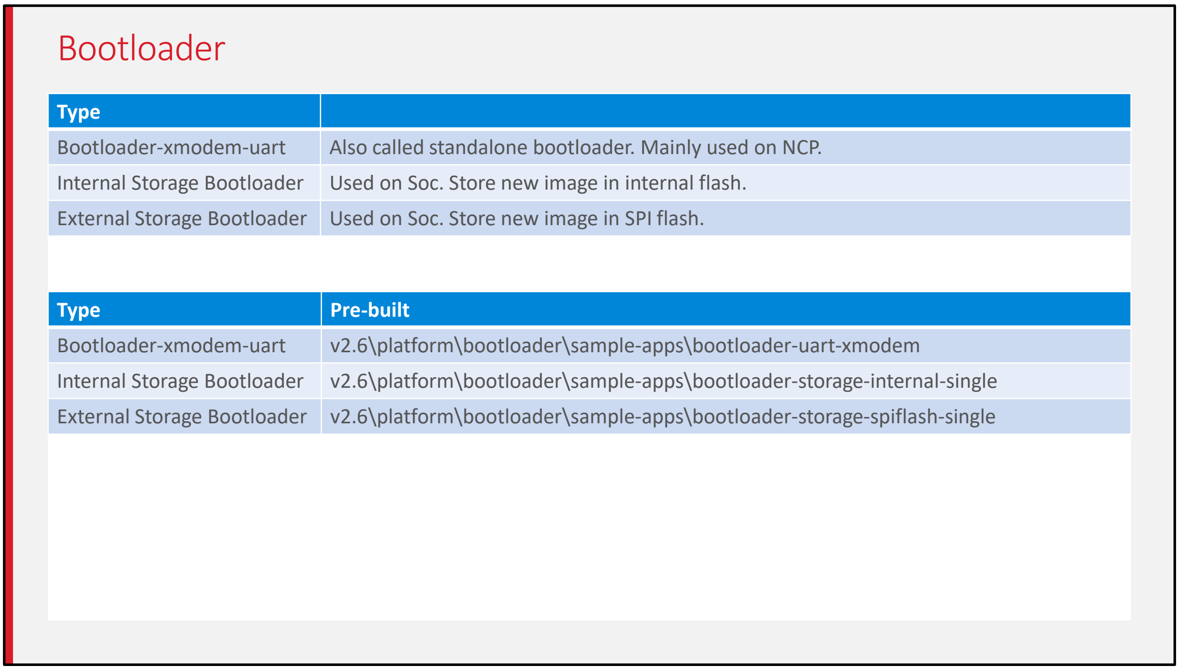
System-on-Chip (SoC)

- IEEE 802.15.4 compliant radio
- Integrated microcontroller
- Integrated peripherals (ADC, encryption engine, GPIO, etc.)
- Lowest system cost

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.
Internal Storage Bootloader	Used on Soc. Store new image in internal flash.
External Storage Bootloader	Used on Soc. Store new image in SPI flash.

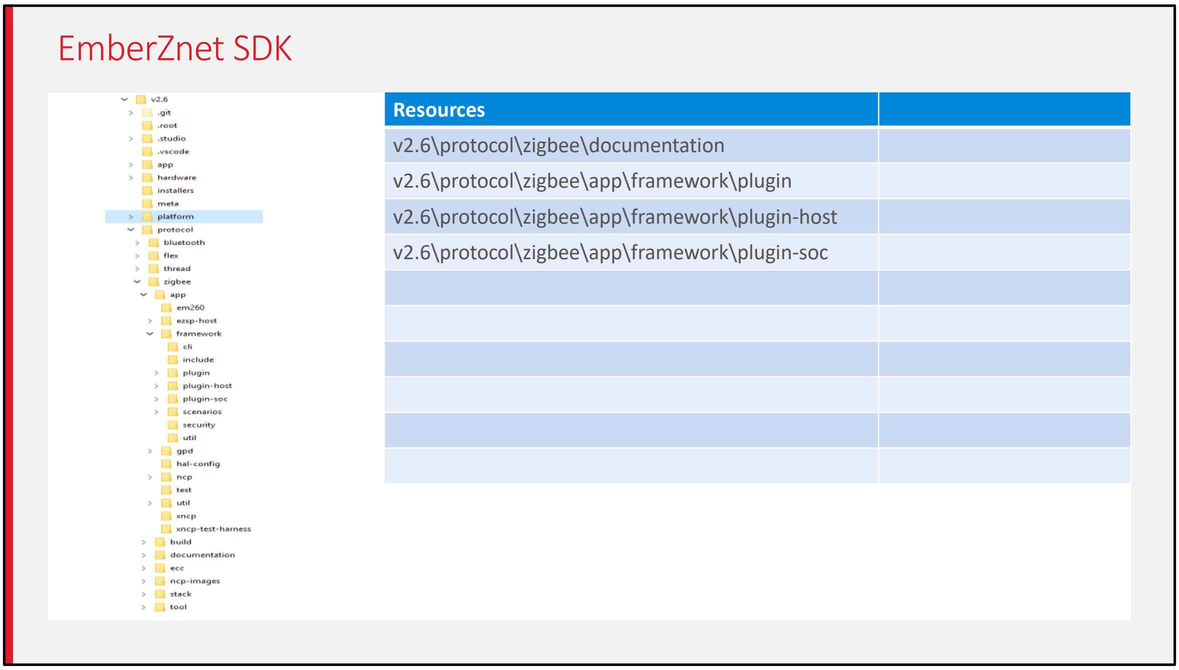
Type	Pre-built
Bootloader-xmodem-uart	v2.6\platform\bootloader\sample-apps\bootloader-uart-xmodem
Internal Storage Bootloader	v2.6\platform\bootloader\sample-apps\bootloader-storage-internal-single
External Storage Bootloader	v2.6\platform\bootloader\sample-apps\bootloader-storage-spiflash-single

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

First, Bootload-xmodem-uart, 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.



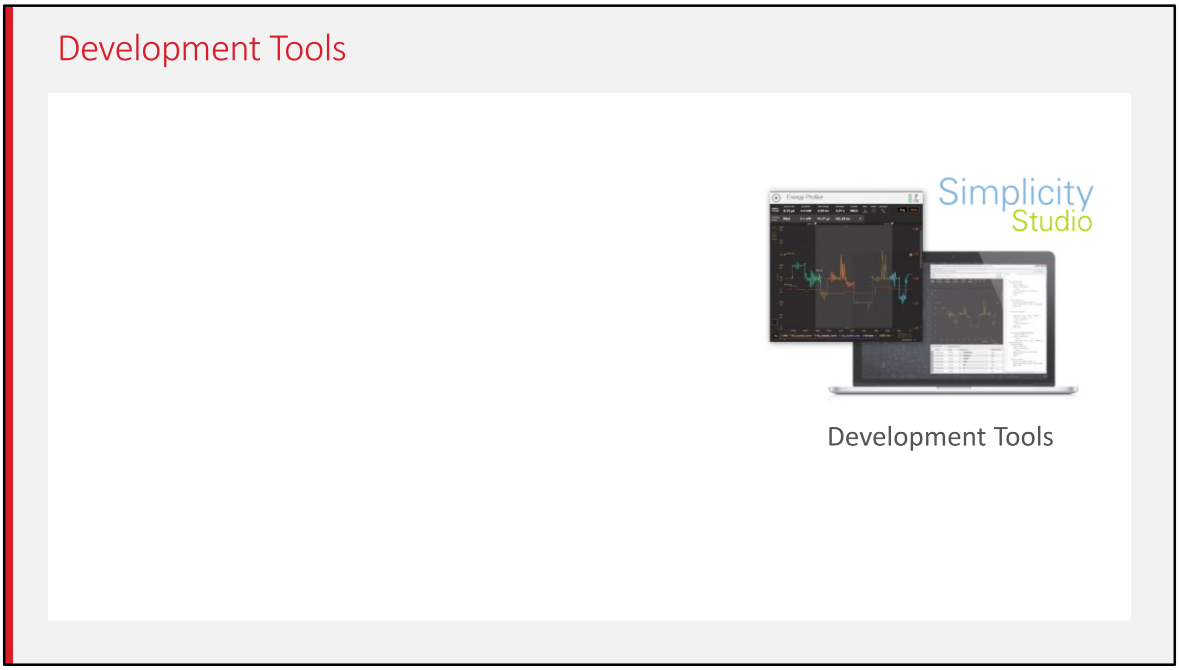
EmberZnet SDK

```
v2.6
  .git
  .root
  .studio
  .vscode
  app
  hardware
  installers
  meta
  platform
  protocol
    bluetooth
    flex
    thread
    zigbee
      app
        em260
        ezsp-host
        framework
          cli
          include
          plugin
          plugin-host
          plugin-soc
          scenarios
          security
          util
        gpd
        hal-config
        ncp
        test
        util
        xncp
        xncp-test-harness
      build
      documentation
      ecc
      ncp-images
      stack
      tool
```

Resources	
v2.6\protocol\zigbee\documentation	
v2.6\protocol\zigbee\app\framework\plugin	
v2.6\protocol\zigbee\app\framework\plugin-host	
v2.6\protocol\zigbee\app\framework\plugin-soc	

This is the directory hierarchy of the SDK.

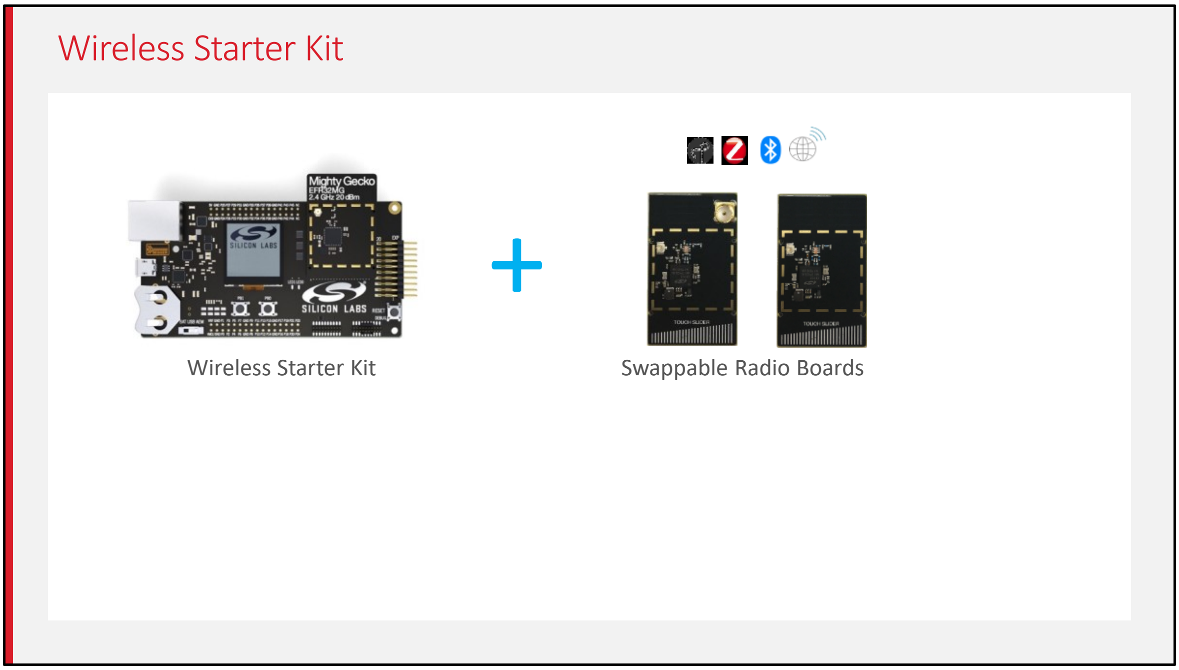
1. In the documentation folder, you can find many documents of our SDK. Those are good learning materials.
2. Most part of the SDK are provided by plugins, and most of the plugins are open-sourced.



Development Tools

Development Tools

Now I will introduce our develop tools.



Wireless Starter Kit

+

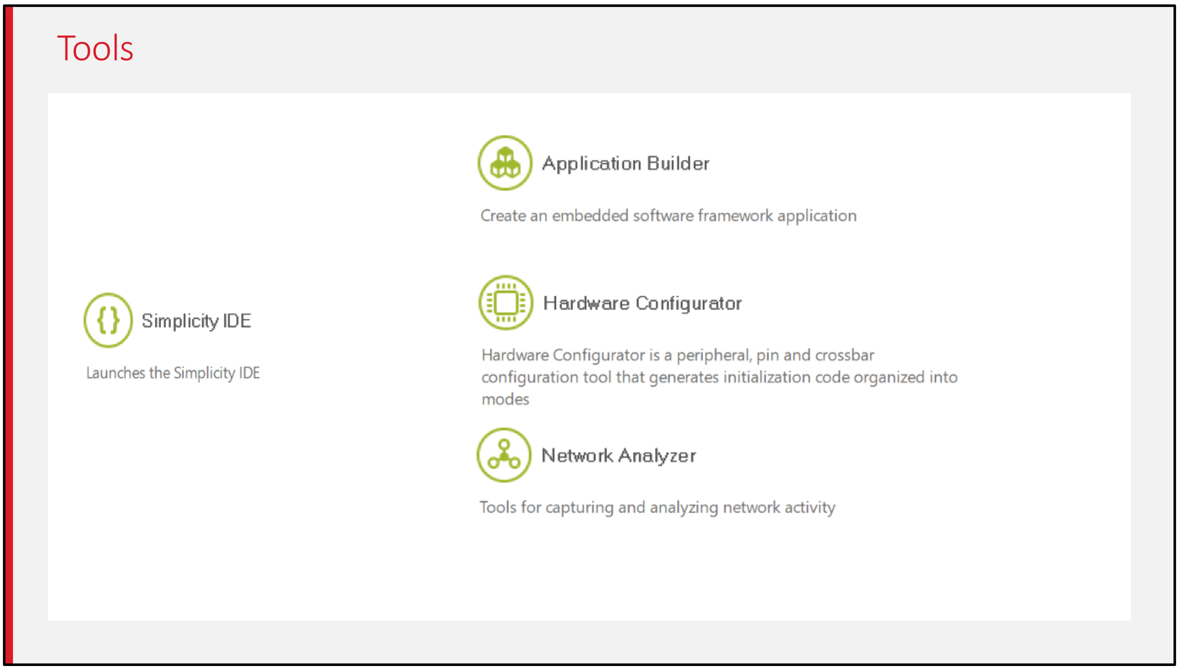
Wireless Starter Kit

Swappable Radio Boards

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

Application Builder

Create an embedded software framework application

Simplicity IDE

Launches the Simplicity IDE

Hardware Configurator

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

Network Analyzer

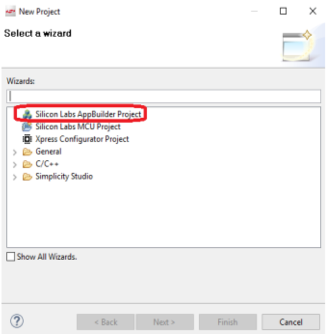
Tools for capturing and analyzing network activity

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.

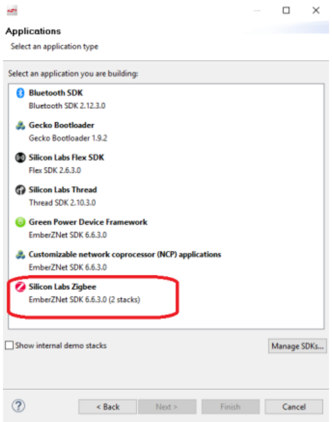
For Zigbee applications, we usually use AppBuilder, Hardware Configurator and Netowork Analyzer.

Create a Zigbee Project – 1/2

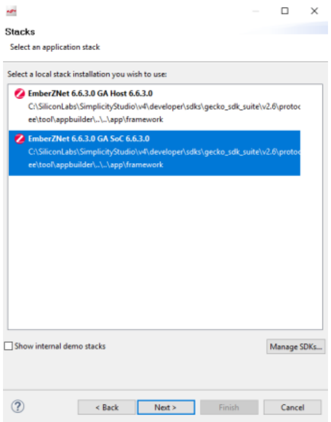
“File”→”New”→”Project”



Choose “Zigbee” Application



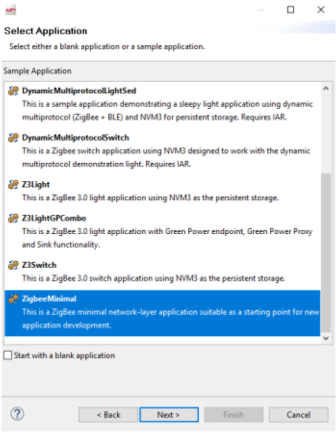
Choose “SoC” Architecture


Stacks
Select an application stack
Select a local stack installation you wish to use:
EmberZNet 6.6.3.0 GA Host 6.6.3.0
C:\SiliconLabs\SimplicityStudio\v4\developer\sdks\gecko_sdk_suite\v2.6\protoc
ee\tool\appbuilder\..\..\app\framework
EmberZNet 6.6.3.0 GA SoC 6.6.3.0
C:\SiliconLabs\SimplicityStudio\v4\developer\sdks\gecko_sdk_suite\v2.6\protoc
ee\tool\appbuilder\..\..\app\framework
Show internal demo stacks
Manage SDKs...
< Back
Next >
Finish
Cancel

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

Create a Zigbee Project – 2/2

Choose example or start from minimal



Select Application

Select either a blank application or a sample application.

Sample Application

DynamicMultiprotocolLightSed
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.

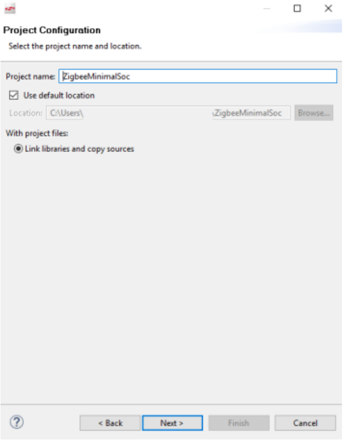
Z3LightGPCombo
This is a ZigBee 3.0 light application with Green Power endpoint, Green Power Proxy and Sink functionality.

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

Name project



Project Configuration

Select the project name and location.

Project name: ZigbeeMinimalSoc

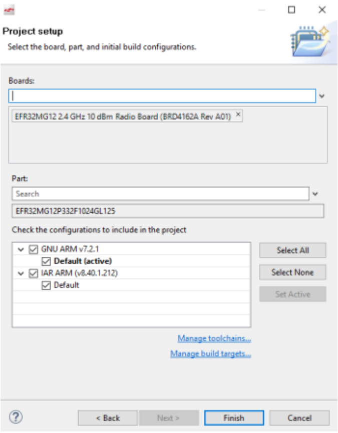
☑ Use default location

Location: C:\Users\ \ZigbeeMinimalSoc

With project files:

● Link libraries and copy sources

Choose “Board”/”Part” and toolchain



Project setup

Select the board, part, and initial build configurations.

Boards:

EFR32MG12 2.4 GHz 10 dBm Radio Board (BRD4162A Rev A01)

Part:

Search

EFR32MG12P332F1024GL125

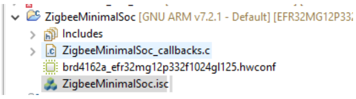
Check the configurations to include in the project

- ☑ GNU ARM v7.2.1
  - ☑ **Default (active)**
- ☑ IAR ARM (v8.40.1.212)
  - ☑ Default

In the “Borads” 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' 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.

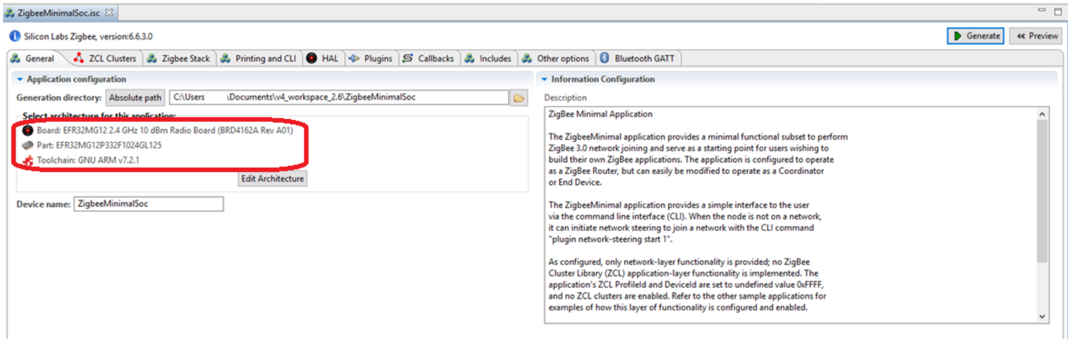
AppBuilder - General

Initial project files



ZigbeeMinimalSoc [GNU ARM v7.2.1 - Default] [EFR32MG12P33:

- Includes
- ZigbeeMinimalSoc_callbacks.c
- brd4162a_efr32mg12p332f1024gl125.hwconf
- ZigbeeMinimalSoc.isc



Silicon Labs Zigbee, version:6.6.3.0

Application configuration

Generation directory: Absolute path C:\Users \Documents\v4_workspace_2.6\ZigbeeMinimalSoc

Select architecture for this application:

- Board: EFR32MG12 2.4 GHz 10 dBm Radio Board (BRD4162A Rev A01)
- Part: EFR32MG12P332F1024GL125
- Toolchain: GNU ARM v7.2.1

Device name: ZigbeeMinimalSoc

Information Configuration

Description

ZigBee Minimal Application

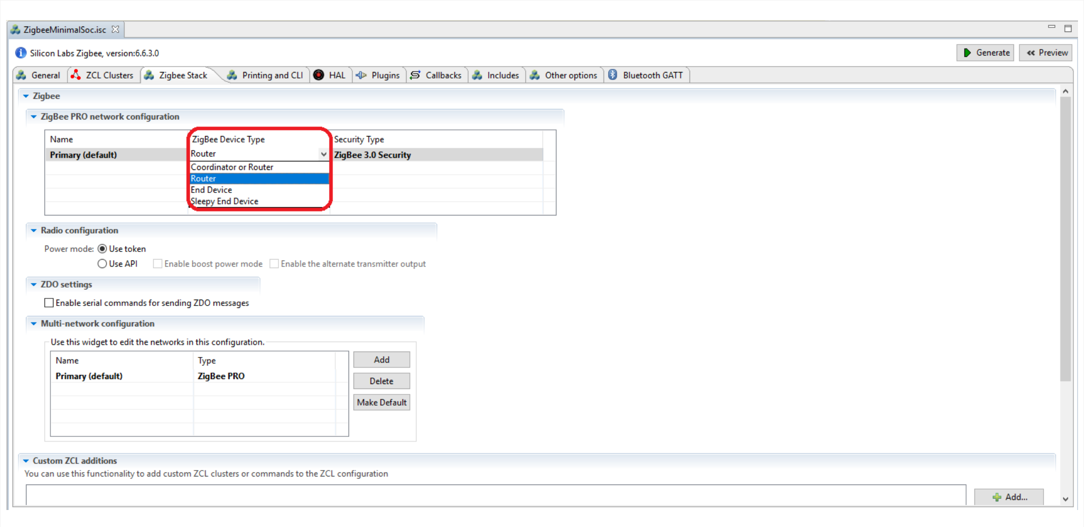
The ZigbeeMinimal application provides a minimal functional subset to perform ZigBee 3.0 network joining and serve as a starting point for users wishing to build their own ZigBee applications. The application is configured to operate as a ZigBee Router, but can easily be modified to operate as a Coordinator or End Device.

The ZigbeeMinimal application provides a simple interface to the user via the command line interface (CLI). When the node is not on a network, it can initiate network steering to join a network with the CLI command "plugin network-steering start 1".

As configured, only network-layer functionality is provided; no ZigBee Cluster Library (ZCL) application-layer functionality is implemented. The application's ZCL ProfileId and DeviceId are set to undefined value 0xFFFF, and no ZCL clusters are enabled. Refer to the other sample applications for examples of how this layer of functionality is configured and enabled.

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



Silicon Labs Zigbee, version:6.6.3.0

Zigbee

ZigBee PRO network configuration

Name	ZigBee Device Type	Security Type
Primary (default)	Router	**ZigBee 3.0 Security**

Coordinator or Router
Router
End Device
Sleepy End Device

Radio configuration

Power mode: ● Use token
○ Use API □ Enable boost power mode □ Enable the alternate transmitter output

ZDO settings

□ Enable serial commands for sending ZDO messages

Multi-network configuration

Use this widget to edit the networks in this configuration.

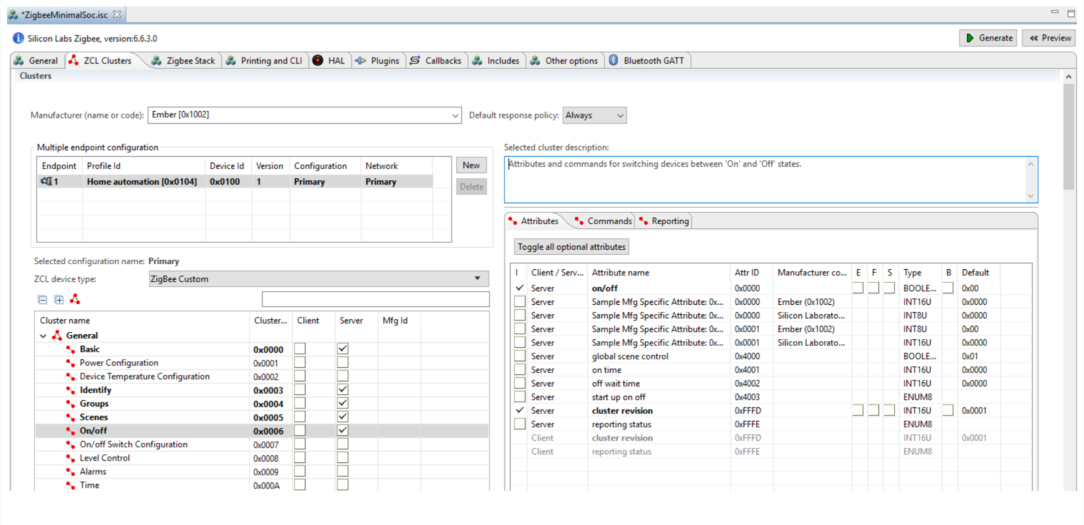
Name	Type
Primary (default)	**ZigBee PRO**

Custom ZCL additions

You can use this functionality to add custom ZCL clusters or commands to the ZCL configuration

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



Silicon Labs Zigbee, version:6.6.3.0

Manufacturer (name or code): Ember [0x1002]

Default response policy: Always

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: ZigBee Custom

Cluster name	Cluster...	Client	Server	Mfg Id
General				
Basic	**0x0000**	□	☑	
Power Configuration	0x0001	□	□	
Device Temperature Configuration	0x0002	□	□	
Identify	**0x0003**	□	☑	
Groups	**0x0004**	□	☑	
Scenes	**0x0005**	□	☑	
On/off	**0x0006**	□	☑	
On/off Switch Configuration	0x0007	□	□	
Level Control	0x0008	□	□	
Alarms	0x0009	□	□	
Time	0x000A	□	□	

Selected cluster description:

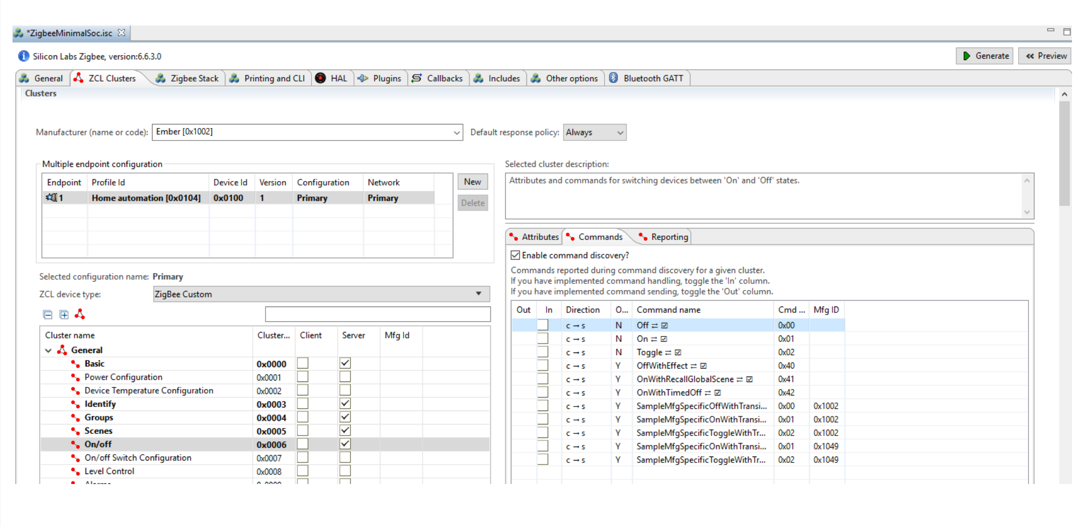
Attributes and commands for switching devices between 'On' and 'Off' states.

I	Client / Serv...	Attribute name	Attr ID	Manufacturer co...	E	F	S	Type	B	Default
✓	Server	**on/off**	0x0000		□	□	□	BOOLE...	□	0x00
□	Server	Sample Mfg Specific Attribute: 0x...	0x0000	Ember (0x1002)				INT16U		0x0000
□	Server	Sample Mfg Specific Attribute: 0x...	0x0000	Silicon Laborato...				INT8U		0x0000
□	Server	Sample Mfg Specific Attribute: 0x...	0x0001	Ember (0x1002)				INT8U		0x00
□	Server	Sample Mfg Specific Attribute: 0x...	0x0001	Silicon Laborato...				INT16U		0x0000
□	Server	global scene control	0x4000					BOOLE...		0x01
□	Server	on time	0x4001					INT16U		0x0000
□	Server	off wait time	0x4002					INT16U		0x0000
□	Server	start up on off	0x4003					ENUM8		
✓	Server	**cluster revision**	0xFFFD		□	□	□	INT16U	□	0x0001
□	Server	reporting status	0xFFFE					ENUM8		
	Client	**cluster revision**	0xFFFD					INT16U		0x0001
	Client	reporting status	0xFFFE					ENUM8		

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



Silicon Labs Zigbee, version:6.6.3.0

Clusters

Manufacturer (name or code): Ember [0x1002]

Default response policy: Always

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: ZigBee Custom

Cluster name	Cluster...	Client	Server	Mfg Id
General				
Basic	**0x0000**	□	☑	
Power Configuration	0x0001	□	□	
Device Temperature Configuration	0x0002	□	□	
Identify	**0x0003**	□	☑	
Groups	**0x0004**	□	☑	
Scenes	**0x0005**	□	☑	
On/off	**0x0006**	□	☑	
On/off Switch Configuration	0x0007	□	□	
Level Control	0x0008	□	□	

Selected cluster description:

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

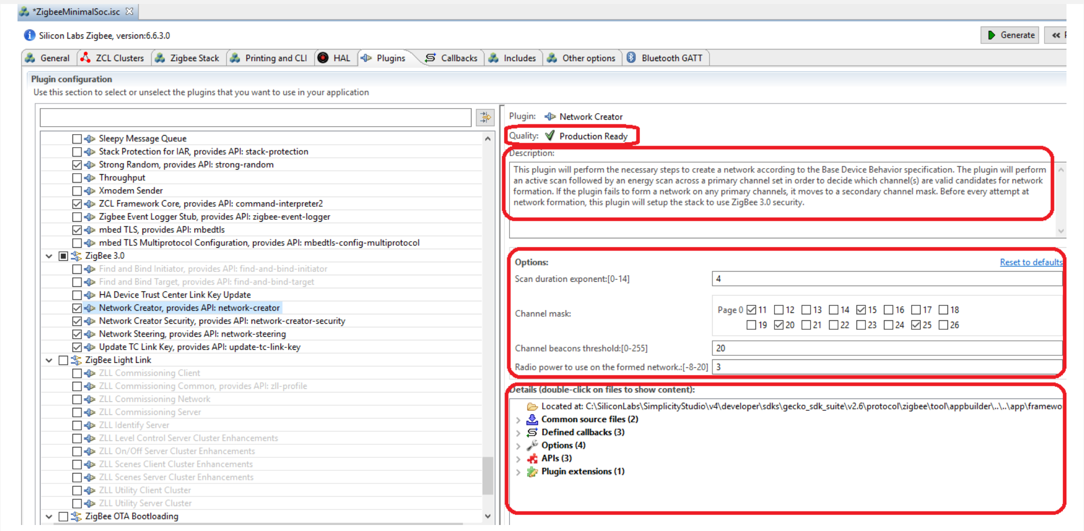
☑ Enable command discovery?

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	O...	Command name	Cmd ...	Mfg ID
	□	c → s	N	Off ⇄ ☑	0x00	
	□	c → s	N	On ⇄ ☑	0x01	
	□	c → s	N	Toggle ⇄ ☑	0x02	
	□	c → s	Y	OffWithEffect ⇄ ☑	0x40	
	□	c → s	Y	OnWithRecallGlobalScene ⇄ ☑	0x41	
	□	c → s	Y	OnWithTimedOff ⇄ ☑	0x42	
	□	c → s	Y	SampleMfgSpecificOffWithTransi...	0x00	0x1002
	□	c → s	Y	SampleMfgSpecificOnWithTransi...	0x01	0x1002
	□	c → s	Y	SampleMfgSpecificToggleWithTr...	0x02	0x1002
	□	c → s	Y	SampleMfgSpecificOnWithTransi...	0x01	0x1049
	□	c → s	Y	SampleMfgSpecificToggleWithTr...	0x02	0x1049

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

AppBuilder - Plugins



Silicon Labs Zigbee, version:6.6.3.0

Plugin configuration

Use this section to select or unselect the plugins that you want to use in your application

- □ Sleepy Message Queue
- □ Stack Protection for IAR, provides API: stack-protection
- ☑ Strong Random, provides API: strong-random
- □ Throughput
- □ Xmodem Sender
- ☑ ZCL Framework Core, provides API: command-interpreter2
- □ Zigbee Event Logger Stub, provides API: zigbee-event-logger
- ☑ mbed TLS, provides API: mbedtls
- □ mbed TLS Multiprotocol Configuration, provides API: mbedtls-config-multiprotocol
- ■ ZigBee 3.0
  - □ Find and Bind Initiator, provides API: find-and-bind-initiator
  - □ Find and Bind Target, provides API: find-and-bind-target
  - □ HA Device Trust Center Link Key Update
  - ☑ Network Creator, provides API: network-creator
  - ☑ Network Creator Security, provides API: network-creator-security
  - ☑ Network Steering, provides API: network-steering
  - ☑ Update TC Link Key, provides API: update-tc-link-key
- □ ZigBee Light Link
  - □ ZLL Commissioning Client
  - □ ZLL Commissioning Common, provides API: zll-profile
  - □ ZLL Commissioning Network
  - □ ZLL Commissioning Server
  - □ ZLL Identify Server
  - □ ZLL Level Control Server Cluster Enhancements
  - □ ZLL On/Off Server Cluster Enhancements
  - □ ZLL Scenes Client Cluster Enhancements
  - □ ZLL Scenes Server Cluster Enhancements
  - □ ZLL Utility Client Cluster
  - □ ZLL Utility Server Cluster
- □ ZigBee OTA Bootloading

Plugin: Network Creator

Quality: ✓ Production Ready

Description:

This plugin will perform the necessary steps to create a network according to the Base Device Behavior specification. The plugin will perform an active scan followed by an energy scan across a primary channel set in order to decide which channel(s) are valid candidates for network formation. If the plugin fails to form a network on any primary channels, it moves to a secondary channel mask. Before every attempt at network formation, this plugin will setup the stack to use ZigBee 3.0 security.

Options: Reset to defaults

Scan duration exponent:[0-14] 4

Channel mask: Page 0 ☑ 11 □ 12 □ 13 □ 14 ☑ 15 □ 16 □ 17 □ 18
□ 19 ☑ 20 □ 21 □ 22 □ 23 □ 24 ☑ 25 □ 26

Channel beacons threshold:[0-255] 20

Radio power to use on the formed network.:[-8-20] 3

Details (double-click on files to show content):

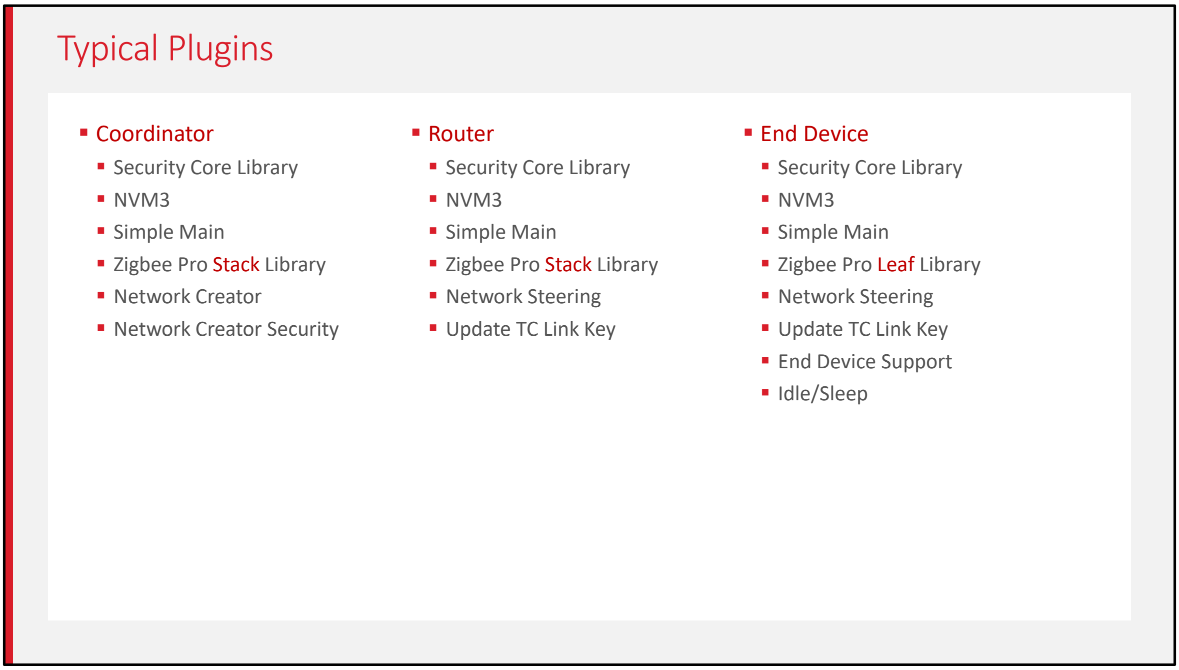
Located at: C:\SiliconLabs\SimplicityStudio\v4\developer\sdks\gecko_sdk_suite\v2.6\protocol\zigbee\tool\appbuilder\..\..\app\framewo

- **Common source files (2)**
- **Defined callbacks (3)**
- **Options (4)**
- **APIs (3)**
- **Plugin extensions (1)**

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.

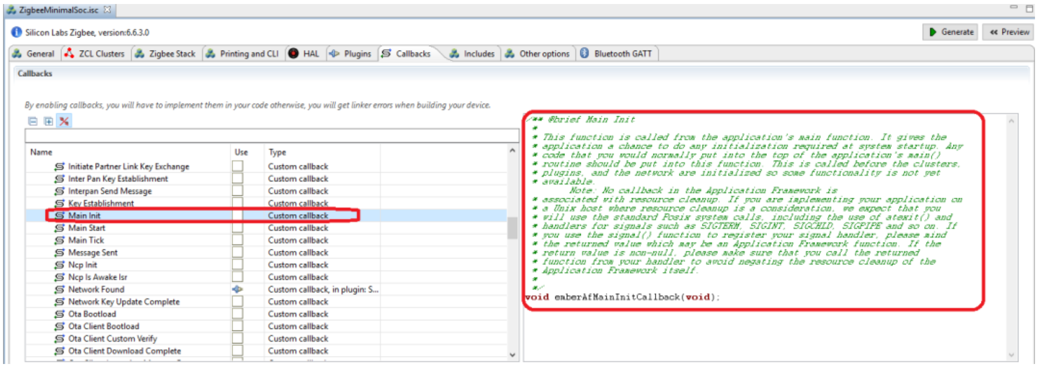


Typical Plugins

- Coordinator
  - Security Core Library
  - NVM3
  - Simple Main
  - Zigbee Pro Stack Library
  - Network Creator
  - Network Creator Security
- Router
  - Security Core Library
  - NVM3
  - Simple Main
  - Zigbee Pro Stack Library
  - Network Steering
  - Update TC Link Key
- End Device
  - Security Core Library
  - NVM3
  - Simple Main
  - Zigbee Pro Leaf Library
  - Network Steering
  - Update TC Link Key
  - End Device Support
  - Idle/Sleep

This is the most commonly used plugins.

AppBuilder - Callbacks



Silicon Labs Zigbee, version:6.6.3.0

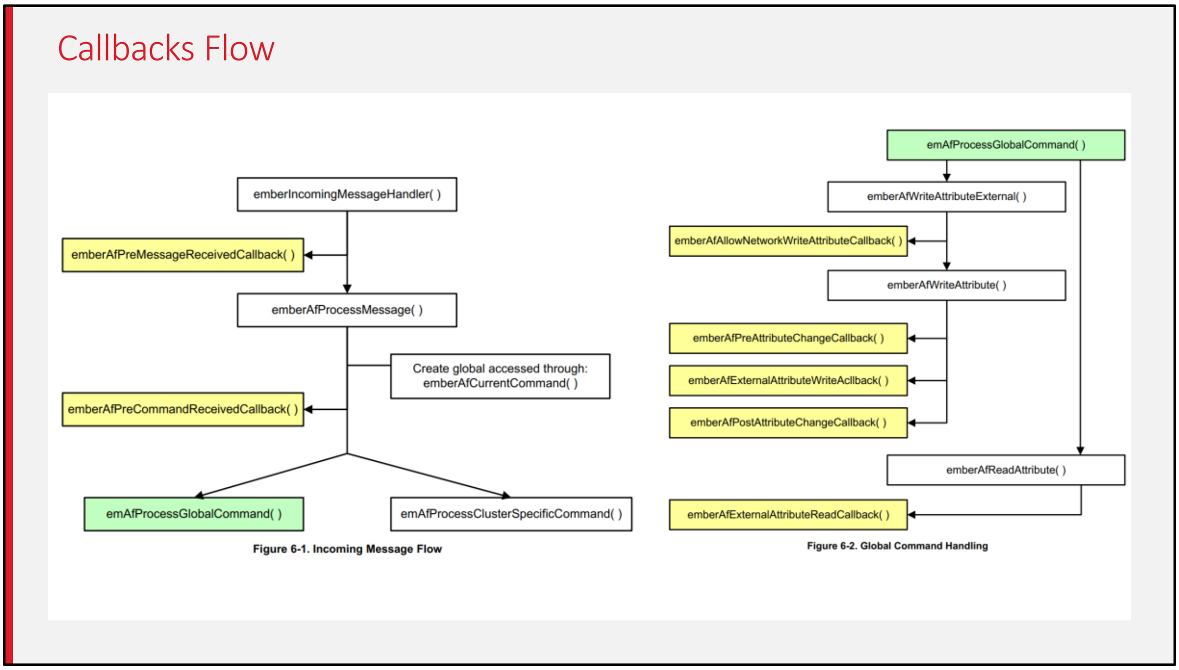
Callbacks

By enabling callbacks, you will have to implement them in your code otherwise, you will get linker errors when building your device.

Name	Use	Type
Initiate Partner Link Key Exchange	□	Custom callback
Inter Pan Key Establishment	□	Custom callback
Interpan Send Message	□	Custom callback
Key Establishment	□	Custom callback
Main Init	□	Custom callback
Main Start	□	Custom callback
Main Tick	□	Custom callback
Message Sent	□	Custom callback
Ncp Init	□	Custom callback
Ncp Is Awake Isr	□	Custom callback
Network Found		Custom callback, in plugin: S...
Network Key Update Complete	□	Custom callback
Ota Bootload	□	Custom callback
Ota Client Bootload	□	Custom callback
Ota Client Custom Verify	□	Custom callback
Ota Client Download Complete	□	Custom callback

```
/** @brief Main Init
 *
 * This function is called from the application's main function. It gives the
 * application a chance to do any initialization required at system startup. Any
 * code that you would normally put into the top of the application's main()
 * routine should be put into this function. This is called before the clusters,
 * plugins, and the network are initialized so some functionality is not yet
 * available.
        Note: No callback in the Application Framework is
 * associated with resource cleanup. If you are implementing your application on
 * a Unix host where resource cleanup is a consideration, we expect that you
 * will use the standard Posix system calls, including the use of atexit() and
 * handlers for signals such as SIGTERM, SIGINT, SIGCHLD, SIGPIPE and so on. If
 * you use the signal() function to register your signal handler, please mind
 * the returned value which may be an Application Framework function. If the
 * return value is non-null, please make sure that you call the returned
 * function from your handler to avoid negating the resource cleanup of the
 * Application Framework itself.
 *
 */
void emberAfMainInitCallback(void);
```

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

emberIncomingMessageHandler( )
emberAfPreMessageReceivedCallback( )
emberAfProcessMessage( )
Create global accessed through:
emberAfCurrentCommand( )
emberAfPreCommandReceivedCallback( )
emAfProcessGlobalCommand( )
emAfProcessClusterSpecificCommand( )

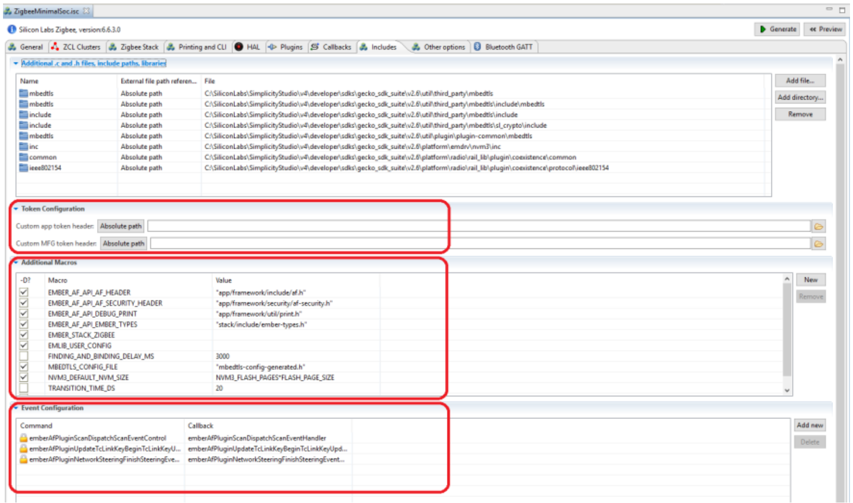
Figure 6-1. Incoming Message Flow

emAfProcessGlobalCommand( )
emberAfWriteAttributeExternal( )
emberAfAllowNetworkWriteAttributeCallback( )
emberAfWriteAttribute( )
emberAfPreAttributeChangeCallback( )
emberAfExternalAttributeWriteAcllback( )
emberAfPostAttributeChangeCallback( )
emberAfReadAttribute( )
emberAfExternalAttributeReadCallback( )

Figure 6-2. Global Command Handling

This is the call flow of the callbacks.

AppBuilder – Custom settings



Silicon Labs Zigbee, version:6.6.3.0

Additional .c and .h files, include paths, libraries

Name	External file path referen...	File
mbedtls	Absolute path	C:\SiliconLabs\SimplicityStudio\v4\developer\sdks\gecko_sdk_suite\v2.6\util\third_party\mbedtls
mbedtls	Absolute path	C:\SiliconLabs\SimplicityStudio\v4\developer\sdks\gecko_sdk_suite\v2.6\util\third_party\mbedtls\include\mbedtls
include	Absolute path	C:\SiliconLabs\SimplicityStudio\v4\developer\sdks\gecko_sdk_suite\v2.6\util\third_party\mbedtls\include
include	Absolute path	C:\SiliconLabs\SimplicityStudio\v4\developer\sdks\gecko_sdk_suite\v2.6\util\third_party\mbedtls\sl_crypto\include
mbedtls	Absolute path	C:\SiliconLabs\SimplicityStudio\v4\developer\sdks\gecko_sdk_suite\v2.6\util\plugin\plugin-common\mbedtls
inc	Absolute path	C:\SiliconLabs\SimplicityStudio\v4\developer\sdks\gecko_sdk_suite\v2.6\platform\emdrv\nvm3\inc
common	Absolute path	C:\SiliconLabs\SimplicityStudio\v4\developer\sdks\gecko_sdk_suite\v2.6\platform\radio\rail_lib\plugin\coexistence\common
ieee802154	Absolute path	C:\SiliconLabs\SimplicityStudio\v4\developer\sdks\gecko_sdk_suite\v2.6\platform\radio\rail_lib\plugin\coexistence\protocol\ieee802154

Token Configuration

Custom app token header: Absolute path

Custom MFG token header: Absolute path

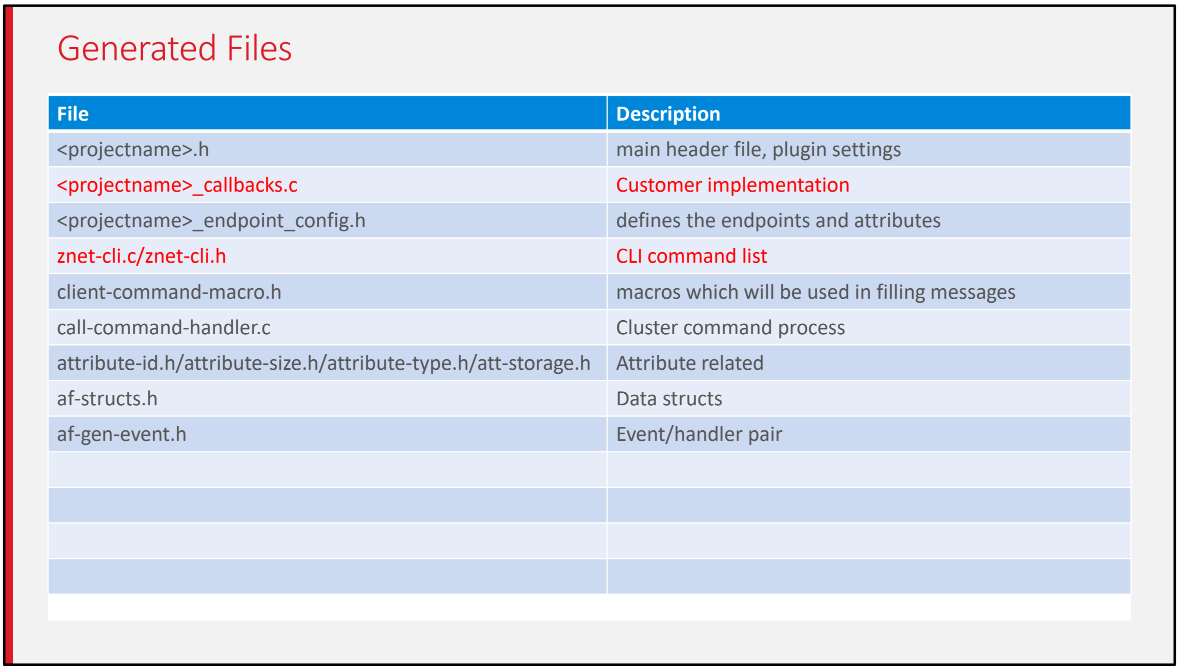
Additional Macros

-D?	Macro	Value
☑	EMBER_AF_API_AF_HEADER	"app/framework/include/af.h"
☑	EMBER_AF_API_AF_SECURITY_HEADER	"app/framework/security/af-security.h"
☑	EMBER_AF_API_DEBUG_PRINT	"app/framework/util/print.h"
☑	EMBER_AF_API_EMBER_TYPES	"stack/include/ember-types.h"
☑	EMBER_STACK_ZIGBEE	
☑	EMLIB_USER_CONFIG	
□	FINDING_AND_BINDING_DELAY_MS	3000
☑	MBEDTLS_CONFIG_FILE	"mbedtls-config-generated.h"
☑	NVM3_DEFAULT_NVM_SIZE	NVM3_FLASH_PAGES*FLASH_PAGE_SIZE
□	TRANSITION_TIME_DS	20

Event Configuration

Command	Callback
emberAfPluginScanDispatchScanEventControl	emberAfPluginScanDispatchScanEventHandler
emberAfPluginUpdateTcLinkKeyBeginTcLinkKeyU...	emberAfPluginUpdateTcLinkKeyBeginTcLinkKeyUpd...
emberAfPluginNetworkSteeringFinishSteeringEve...	emberAfPluginNetworkSteeringFinishSteeringEvent...

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

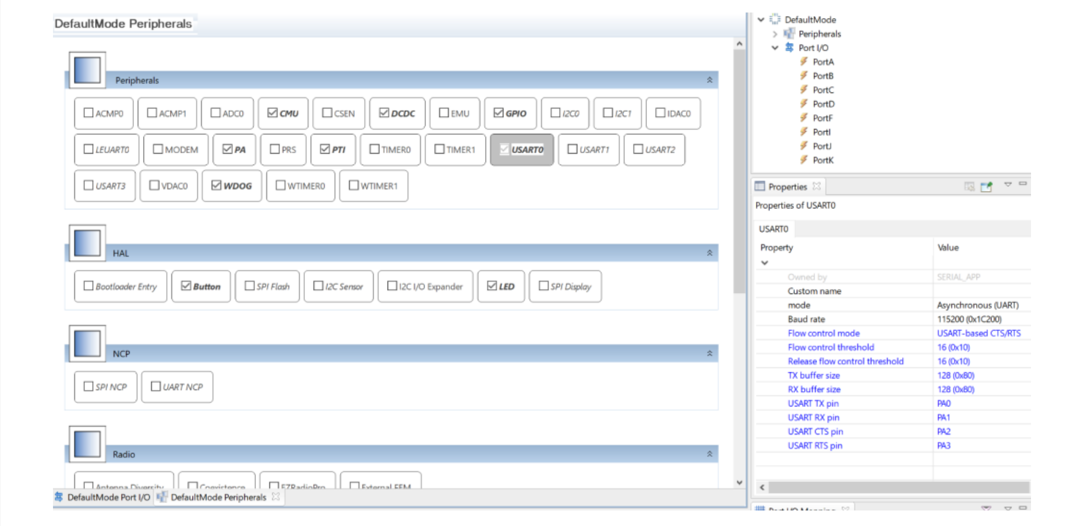


Generated Files

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

Show the generated files.

Hardware Configurator



DefaultMode Peripherals

Peripherals

☐ ACMP0 ☐ ACMP1 ☐ ADC0 ☑ **CMU** ☐ CSEN ☑ **DCDC** ☐ EMU ☑ **GPIO** ☐ *I2C0* ☐ *I2C1* ☐ IDAC0

☐ *LEUART0* ☐ MODEM ☑ **PA** ☐ PRS ☑ **PTI** ☐ TIMER0 ☐ TIMER1 ☑ **USART0** ☐ *USART1* ☐ *USART2*

☐ *USART3* ☐ VDAC0 ☑ **WDOG** ☐ WTIMER0 ☐ WTIMER1

HAL

☐ *Bootloader Entry* ☑ **Button** ☐ *SPI Flash* ☐ *I2C Sensor* ☐ I2C I/O Expander ☑ **LED** ☐ *SPI Display*

NCP

☐ *SPI NCP* ☐ *UART NCP*

Radio

☐ Antenna Diversity ☐ Coexistence ☐ EZRadioPro ☐ External FEM

DefaultMode Port I/O · DefaultMode Peripherals

- DefaultMode
  - Peripherals
  - Port I/O
    - PortA
    - PortB
    - PortC
    - PortD
    - PortF
    - PortI
    - PortJ
    - PortK

Properties

Properties of USART0

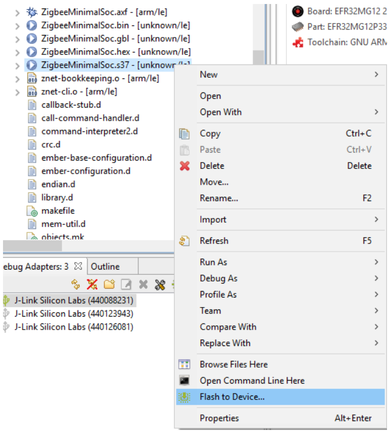
USART0

Property	Value
Owned by	SERIAL_APP
Custom name	
mode	Asynchronous (UART)
Baud rate	115200 (0x1C200)
Flow control mode	USART-based CTS/RTS
Flow control threshold	16 (0x10)
Release flow control threshold	16 (0x10)
TX buffer size	128 (0x80)
RX buffer size	128 (0x80)
USART TX pin	PA0
USART RX pin	PA1
USART CTS pin	PA2
USART RTS pin	PA3

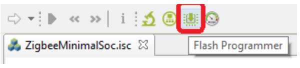
For custom board, you would need to change the hardware configuration according to your schematic.

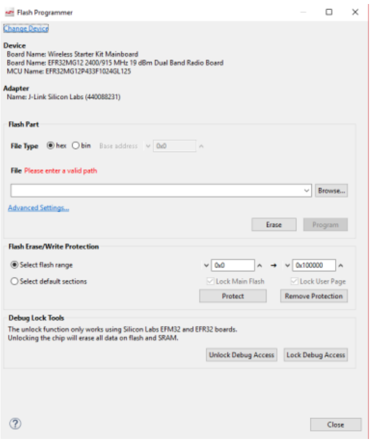
Flash/Program

Method 1:



Method 2:





Device
Board Name: Wireless Starter Kit Mainboard
Board Name: EFR32MG12 2400/915 MHz 19 dBm Dual Band Radio Board
MCU Name: EFR32MG12P433F1024GL125

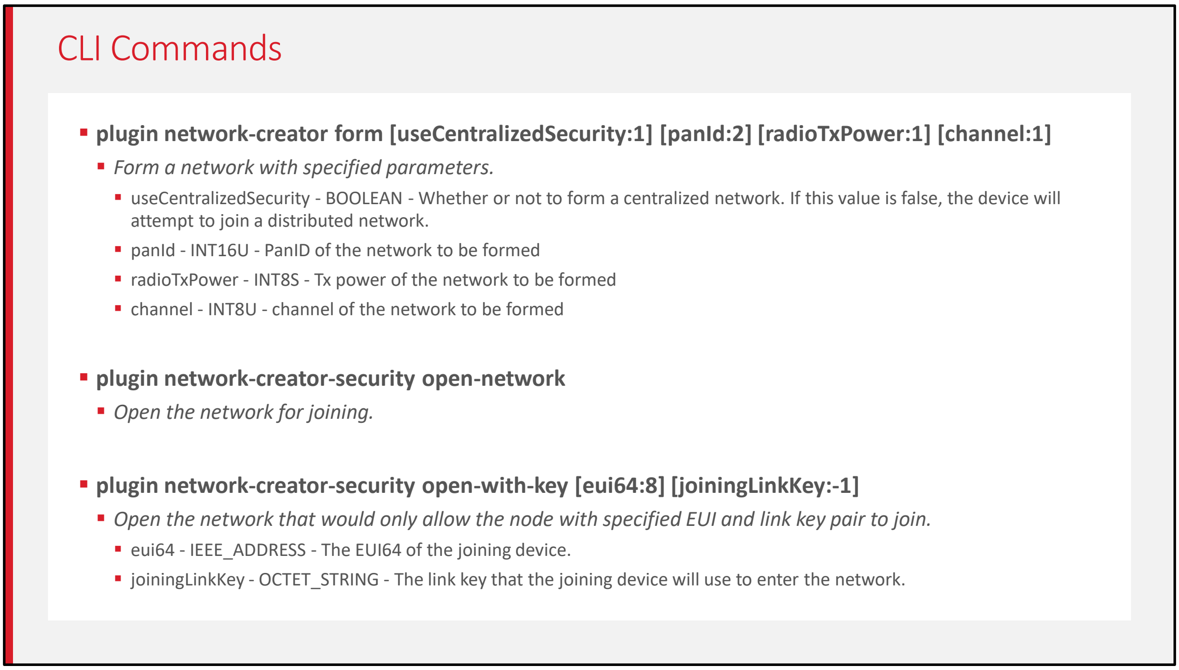
Adapter
Name: J-Link Silicon Labs (440088231)

Flash Part
File Type ● hex ○ bin Base address 0x0
File Please enter a valid path

Flash Erase/Write Protection
● Select flash range 0x0 → 0x100000
○ Select default sections
☑ Lock Main Flash ☑ Lock User Page

Debug Lock Tools
The unlock function only works using Silicon Labs EFM32 and EFR32 boards.
Unlocking the chip will erase all data on flash and SRAM.

Let me show you the approaches of flashing programs.

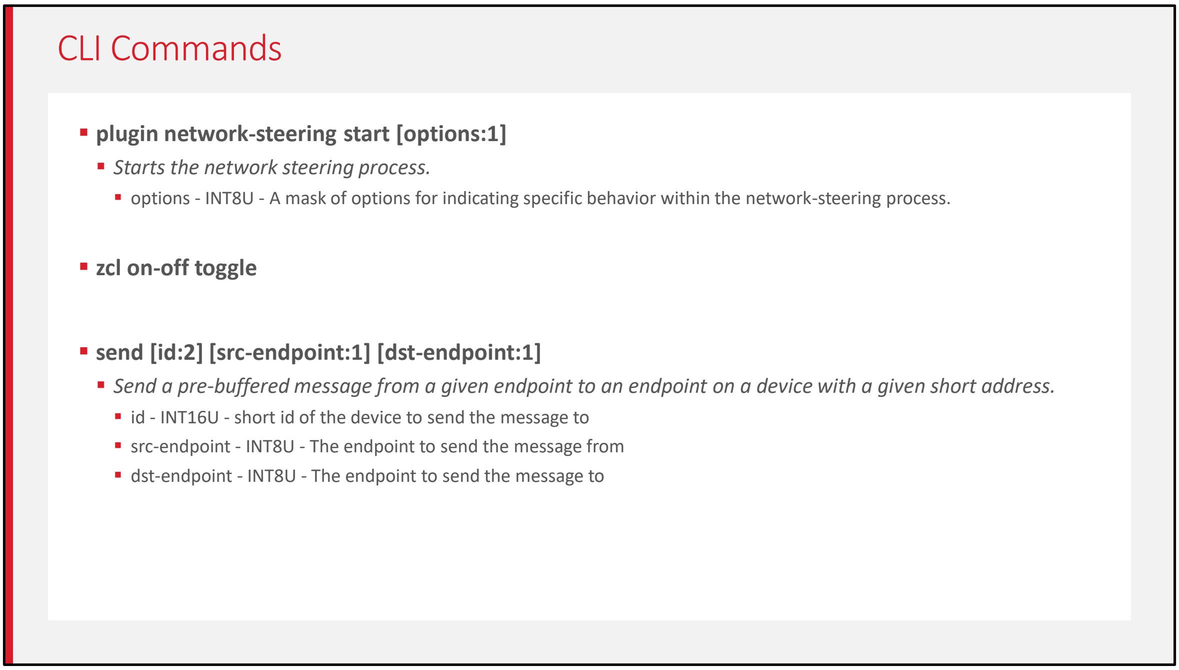


CLI Commands

- **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 used 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.



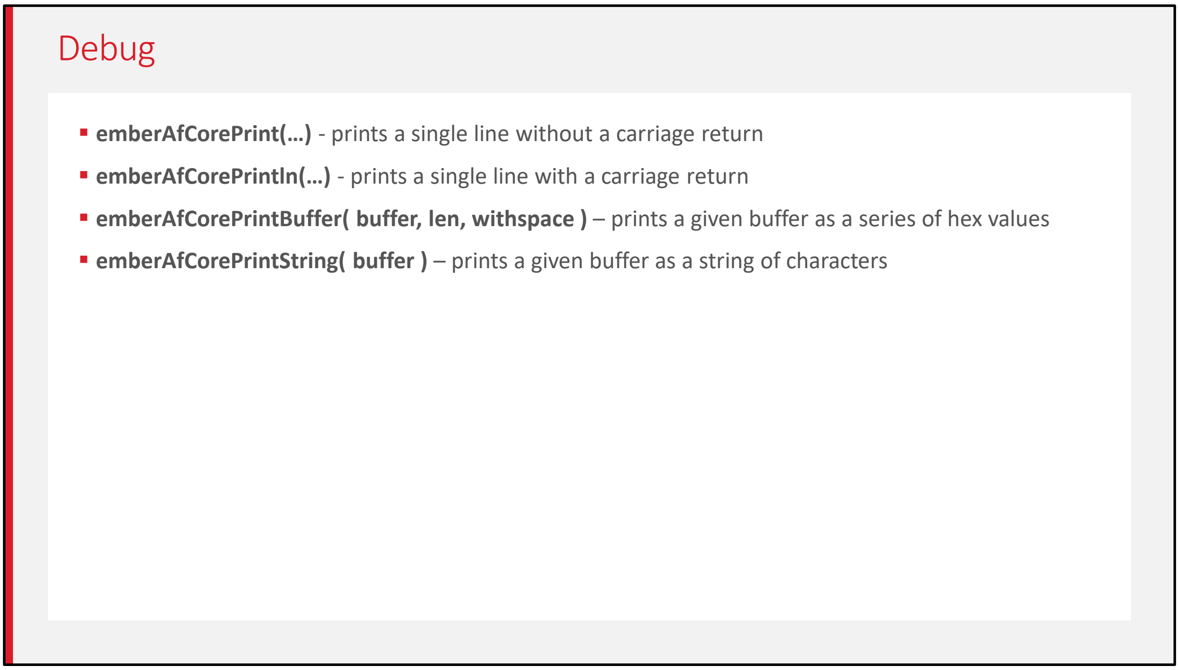
CLI Commands

- **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-steeing 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.



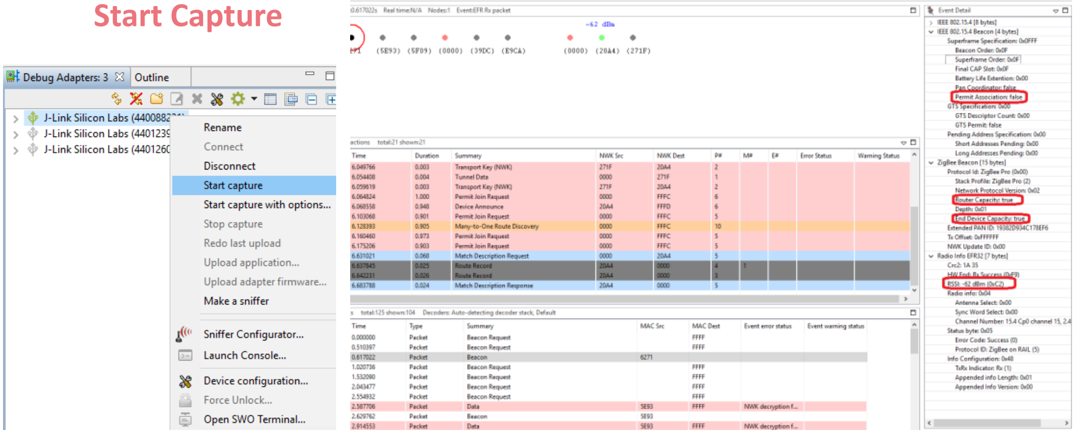
Debug

- **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



0.617022s Real time:N/A Nodes:1 Event:EFR Rx packet

actions total:21 shown:21

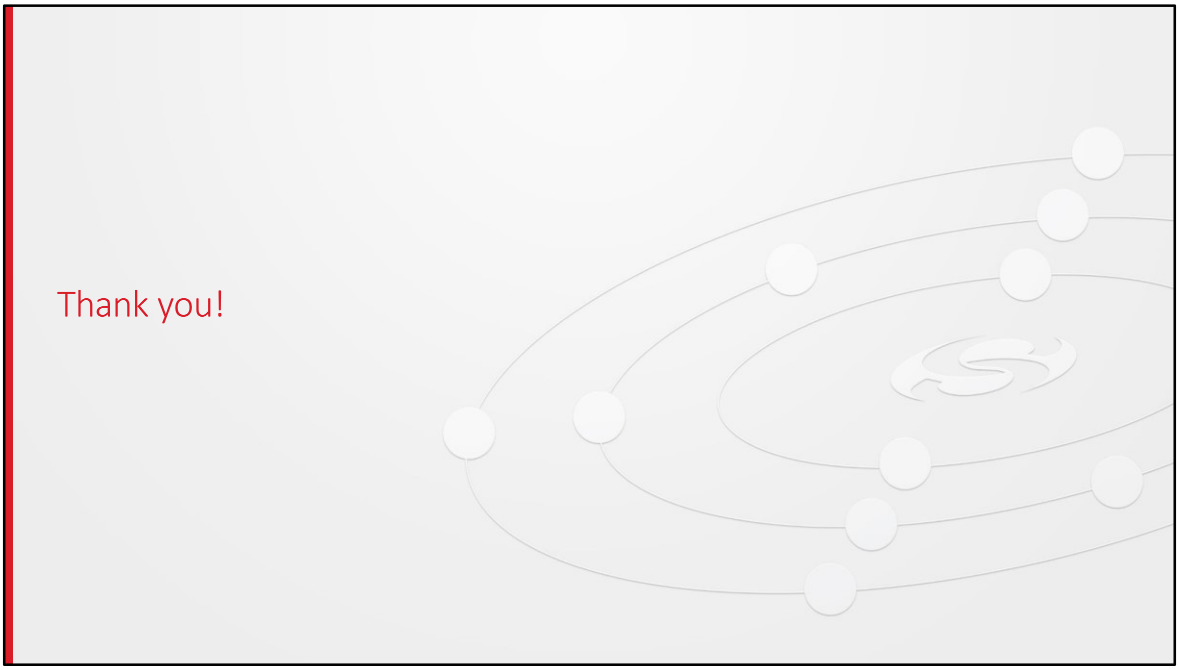
Time	Duration	Summary	NWK Src	NWK Dest	P#	M#	E#	Error Status	Warning Status
6.049766	0.003	Transport Key (NWK)	271F	20A4	2				
6.054408	0.004	Tunnel Data	0000	271F	1				
6.059619	0.003	Transport Key (NWK)	271F	20A4	2				
6.064824	1.000	Permit Join Request	0000	FFFC	6				
6.068558	0.948	Device Announce	20A4	FFFD	6				
6.103068	0.901	Permit Join Request	0000	FFFC	5				
6.128393	0.905	Many-to-One Route Discovery	0000	FFFC	10				
6.160460	0.973	Permit Join Request	0000	FFFC	5				
6.175206	0.903	Permit Join Request	0000	FFFC	5				
6.631021	0.068	Match Description Request	0000	20A4	5				
6.637845	0.025	Route Record	20A4	0000	4	1			
6.642231	0.026	Route Record	20A4	0000	3				
6.683788	0.024	Match Description Response	20A4	0000	5				

s total:125 shown:104 Decoders: Auto-detecting decoder stack, Default

Time	Type	Summary	MAC Src	MAC Dest	Event error status	Event warning status
0.000000	Packet	Beacon Request		FFFF		
0.510397	Packet	Beacon Request		FFFF		
0.617022	Packet	Beacon	6271			
1.020736	Packet	Beacon Request		FFFF		
1.532090	Packet	Beacon Request		FFFF		
2.043477	Packet	Beacon Request		FFFF		
2.554932	Packet	Beacon Request		FFFF		
2.587706	Packet	Data	5E93	FFFF	NWK decryption f...	
2.629762	Packet	Beacon	5E93			
2.914553	Packet	Data	5E93	FFFF	NWK decryption f...	

```
> IEEE 802.15.4 [8 bytes]
v IEEE 802.15.4 Beacon [4 bytes]
      Superframe Specification: 0x0FFF
         Beacon Order: 0x0F
         Superframe Order: 0x0F
         Final CAP Slot: 0x0F
         Battery Life Extention: 0x00
         Pan Coordinator: false
         Permit Association: false
      GTS Specification: 0x00
         GTS Descriptor Count: 0x00
         GTS Permit: false
      Pending Address Specification: 0x00
         Short Addresses Pending: 0x00
         Long Addresses Pending: 0x00
v ZigBee Beacon [15 bytes]
      Protocol Id: ZigBee Pro (0x00)
         Stack Profile: ZigBee Pro (2)
         Network Protocol Version: 0x02
         Router Capacity: true
         Depth: 0x01
         End Device Capacity: true
      Extended PAN ID: 19382D934C178EF6
      Tx Offset: 0xFFFFFF
      NWK Update ID: 0x00
v Radio Info EFR32 [7 bytes]
      Crc2: 1A 35
      HW End: Rx Success (0xF9)
      RSSI: -62 dBm (0xC2)
      Radio info: 0x04
         Antenna Select: 0x00
         Sync Word Select: 0x00
         Channel Number: 15.4 Cp0 channel 15, 2.4
      Status byte: 0x05
         Error Code: Success (0)
         Protocol ID: ZigBee on RAIL (5)
      Info Configuration: 0x48
         TxRx Indicator: Rx (1)
         Appended info Length: 0x01
         Appended Info Version: 0x00
```

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!