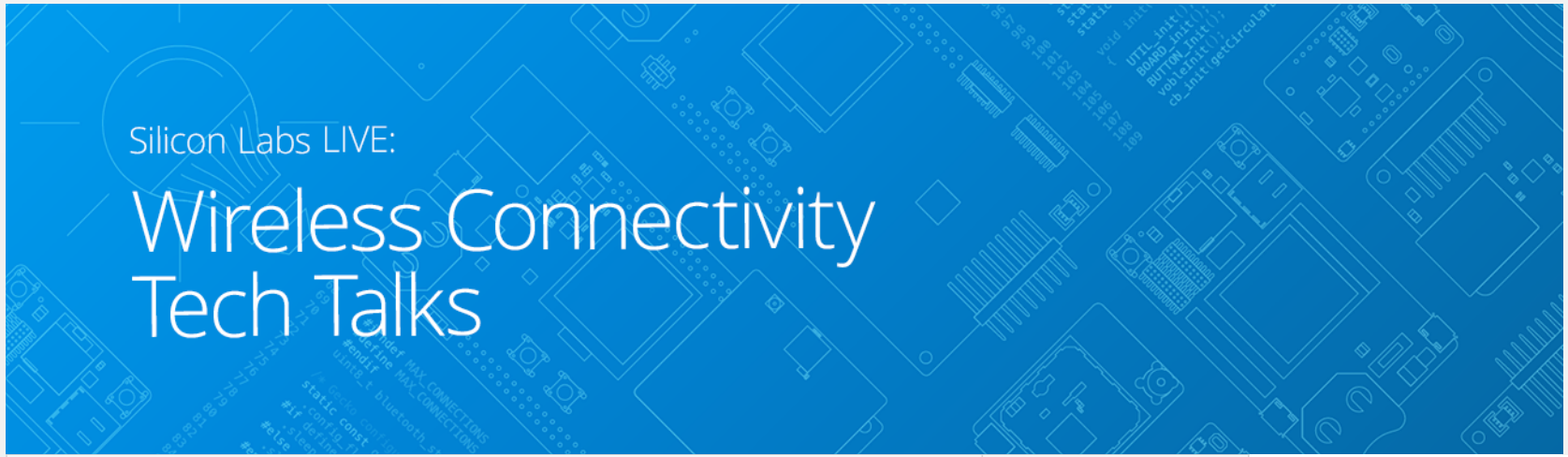
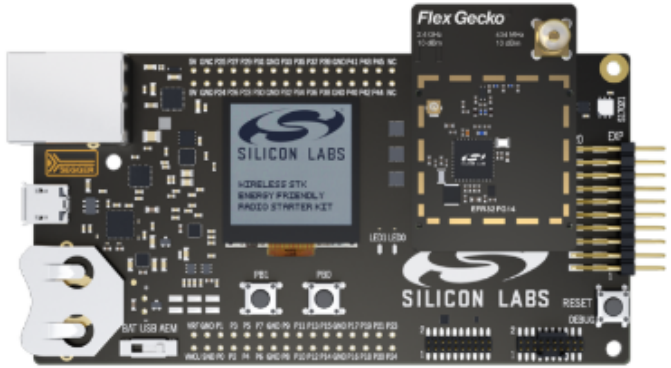
Tech Talks LIVE Schedule – Presentation will begin shortly



How to Measure and Debug Network Performance - Using Silicon Labs Network Analyzer	Thursday, May 7
RF Regulatory and Qualification Testing for Bluetooth, Zigbee & Z-Wave	Tuesday, May 12
Simplicity Studio Tips & Tricks: Our FAEs Know All The Tricks - Improve Your Life in Simplicity Studio	Thursday, May 14
Wireless Module vs Wireless SoC Tradeoffs and Decision Making Criteria	Tuesday, May 19
Thunderboard BG22 Unboxing. You Have Our Kit… What Can You Do With It?	Thursday, May 21
Designing in Bluetooth using Bluetooth Xpress Modules with Minimal Code Writing	Tuesday, May 26
Overview of Silicon Labs Wi-Fi Solutions (Including Redpine Signals Wi-Fi Solutions)	Thursday, May 28

Please take the 3 question poll while waiting and be entered to receive a Flex Gecko Starter Kit.



Find Past Recorded Sessions at:
https://www.silabs.com/support/training

WELCOME

Silicon Labs LIVE:

Wireless Connectivity Tech Talks

SubGHz, Proprietary and Connect Software Stack

MAY 2020 | CHRIS LEAGUE

Proprietary Wireless Solutions

- Smart Meters

- Home Automation and Security

- Garage Door Openers

- Public Infrastructure

- Agriculture

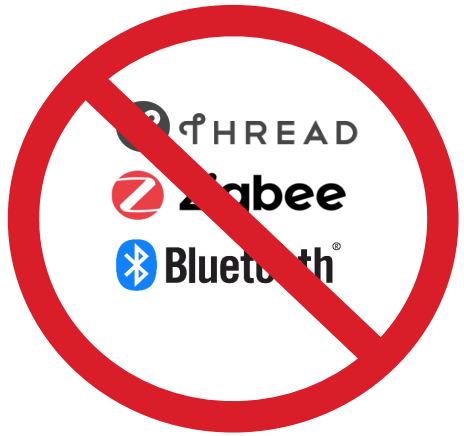
- Asset Tracking & ESL

Why Consider Proprietary Solution?

Proprietary

USE CASE MISMATCH

- If a Standard Protocol does not fit your Use Case. Bluetooth Mesh does not have enough throughput for sensors; THREAD take too long to rebuild its mesh after a power outage; Zigbee can not support enough end nodes.
- Your design has special requirements, like time flooding to synchronize end node measurements.



INNATE SECURITY

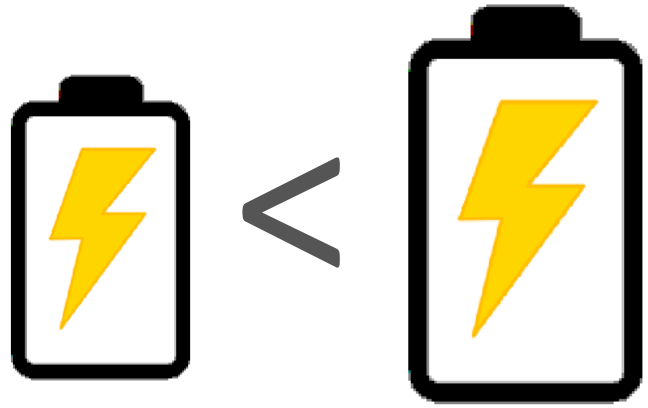
- Standard Protocols have known data rates, modulation, and package structure, making them more vulnerable.
- Proprietary network innately are more because of obfuscation of the above network parameters.

Why Consider Proprietary Solution

Proprietary

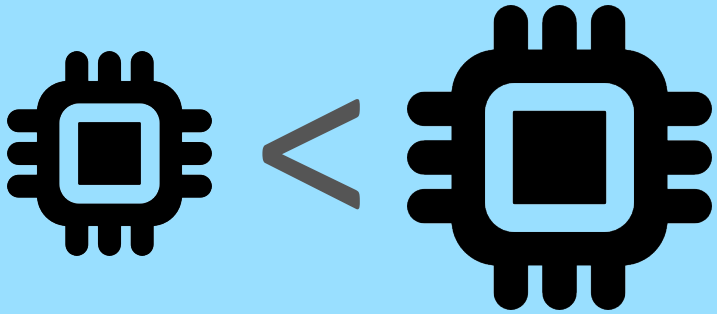
LOWER POWER

- Standard Protocols have requirements for battery-powered end node involvement.
- Being totally in control of your battery-powered end nodes can lead to much lower power.



LOWER COST

- To Serve more market, Standard protocol continue to grow in memory size.
- Specifically satisfying your own Use Case can reduce memory and processing requirements.

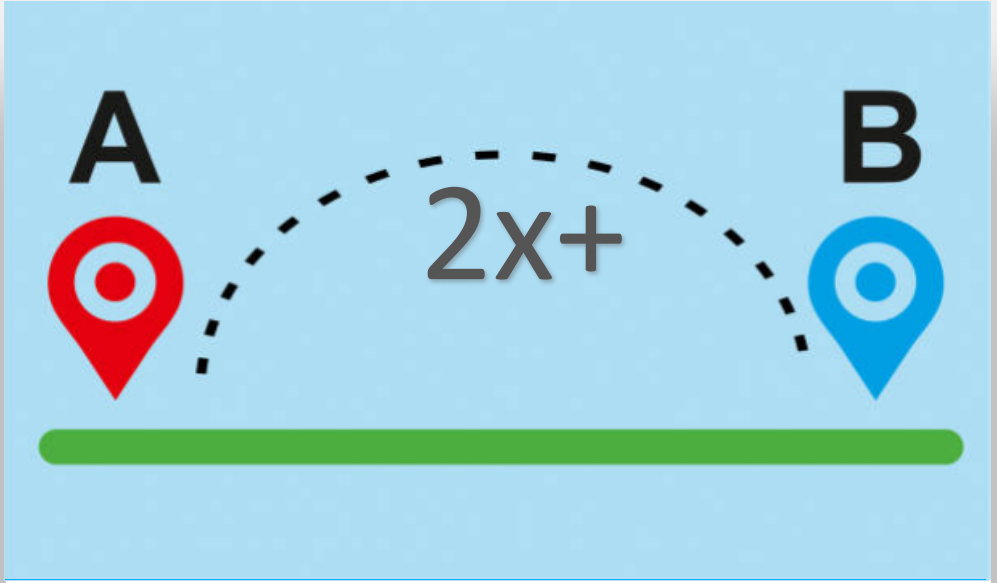


Why Consider SubGHz Solution?

Proprietary

LONGER RANGE

- Through Objects - As radio waves pass through walls and other obstacles, the signal weakens. Attenuation rates increase at higher frequencies, therefore the 2.4GHz signal weakens faster than a sub-GHz signal.
- Around Objects- Even though radio waves travel in a straight line, they do bend when they hit a solid edge (like the corner of a building). As frequencies decrease the angle of diffraction increases, allowing sub-GHz signals to bend farther around an obstacle, reducing the blocking effect.
- LOS - Even in line of sight, Sub GHz signals have superior propagation due to Path Loss:
  - 2.4GHz vs 915MHz => 8.5dB or 2.8x range
  - 2.4GHz vs 433MHz => 14.8dB or 5x range



$$Path\ Loss = 20 * log_{10}\left(\frac{4\pi d}{\lambda}\right)$$

Where *d* is distance and

λ is wave length.

Rule of Thumb: Range doubles for every 6dB of link budget increase.

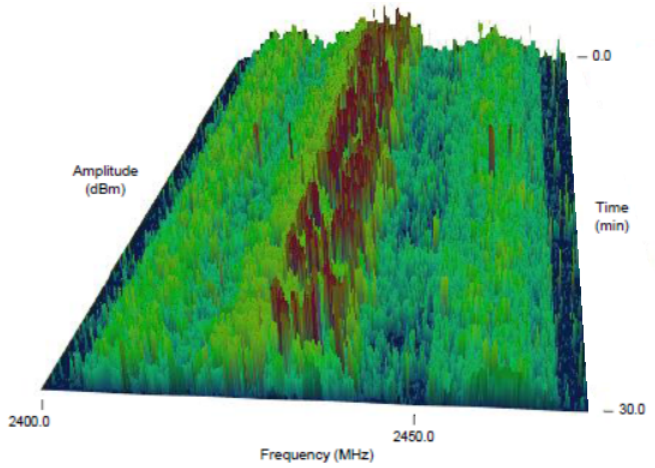
Why Consider SubGHz Solution?

Proprietary

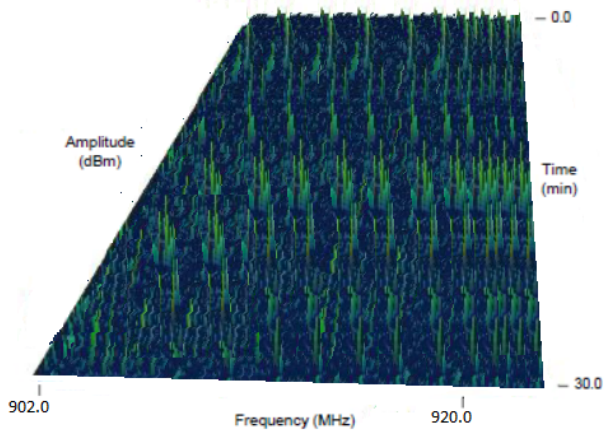
LESS INTERFERENCE

- The 2.4GHz airways are crowded with colliding 2.4GHz signals from various sources, such as Wi-Fi, Bluetooth-enabled phones and peripherals, ZigBee, and even microwave ovens and arc welders.
- Sub-GHz ISM bands are mostly used for proprietary low-duty-cycle links and are not as likely to interfere with each other. The quieter spectrum means easier transmissions and fewer retries, which is more efficient and saves battery power.

2.4 GHz Band



915MHz Band

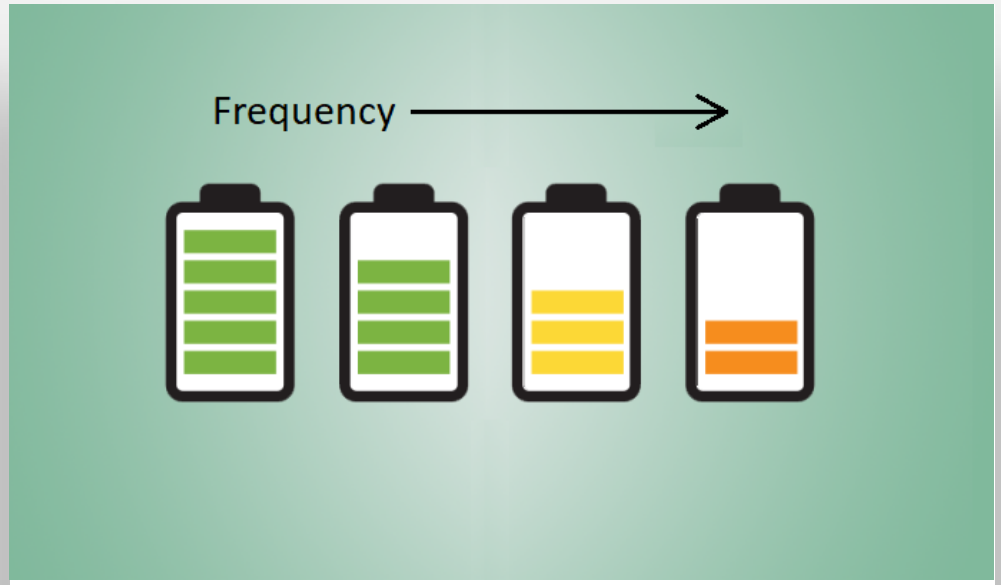


Why Consider SubGHz Solution?

Proprietary

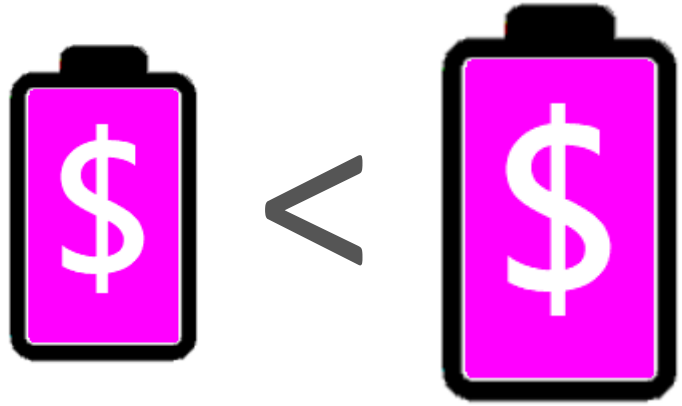
LOWER POWER

- SubGHz designs can transmit at lower power levels and achieve the same link budgets, saving battery life.
- Less interference means less retries, saving battery life.



LOWER COST

- When designing 20 years life devices, the long life chemistry battery (Lithium/Thionyl Chloride) will be the highest cost component of the design. Being able to take advantage of the energy savings of Proprietary SubGHz can be the difference between a D cell or C cell.



Why Silicon Labs for Proprietary SubGHz?

Proprietary

- Leadership
  - Leader in proprietary wireless
    - 12+ years of experience in proprietary wireless market
  - Trusted partnership with market leaders in metering, security, lighting, home and industrial automation
- Extensive portfolio comprising RF transceivers and Wireless SoC platform solutions
  - Excellent link budget up to 148 dBm for long range connectivity
  - Excellent performance in the presence of blockers
  - Industry leading integrated +20dBm PA
  - Full-featured radio configuration software and networking stacks
- Cutting-edge software and development tools
  - Comprehensive, easy-to-use tools and development environment
  - Radio Configurator, Packet Trace, Network Analyzer, AppBuilder, Energy Profiler

600+ Million proprietary wireless ICs shipped

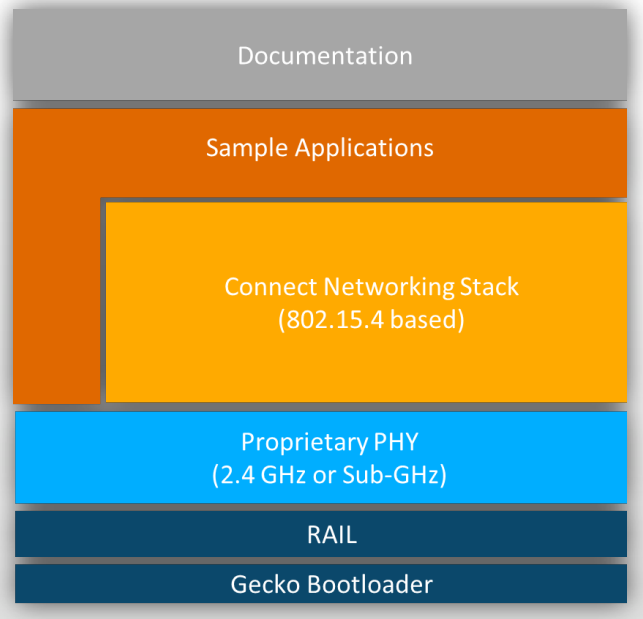
A Comprehensive Proprietary Portfolio

Proprietary

WIRELESS SOCS AND TRANSCEIVERS

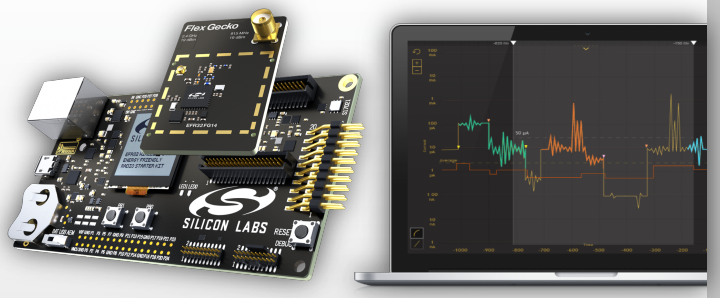
Ultra Low Power
2.4 GHz and Sub GHz
Industry Leading Multi-Protocol SoCs

FLEX SDK – RAIL API & CONNECT STACK



Predefined PHYs
Complete Development Suite
Sample Application With Source Code

TOOLS AND EASE-OF-USE

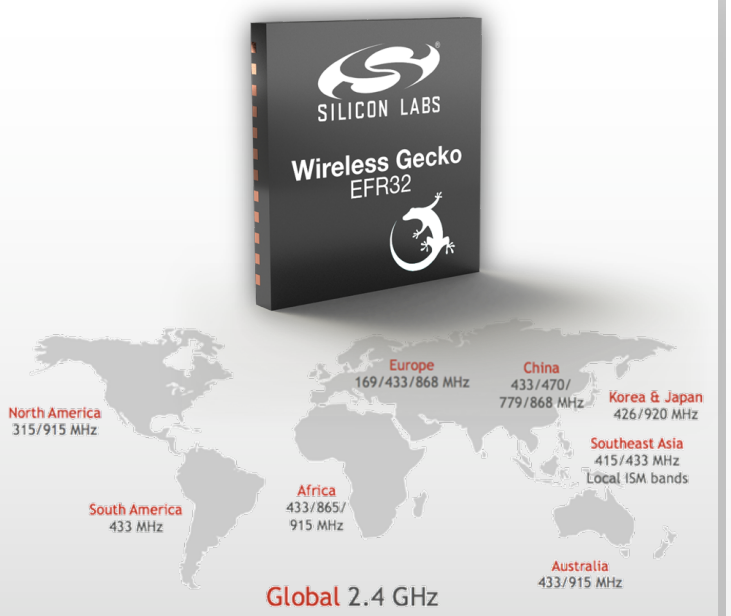


Radio boards
Simplicity Studio
Wireless Starter Kits

The Parts

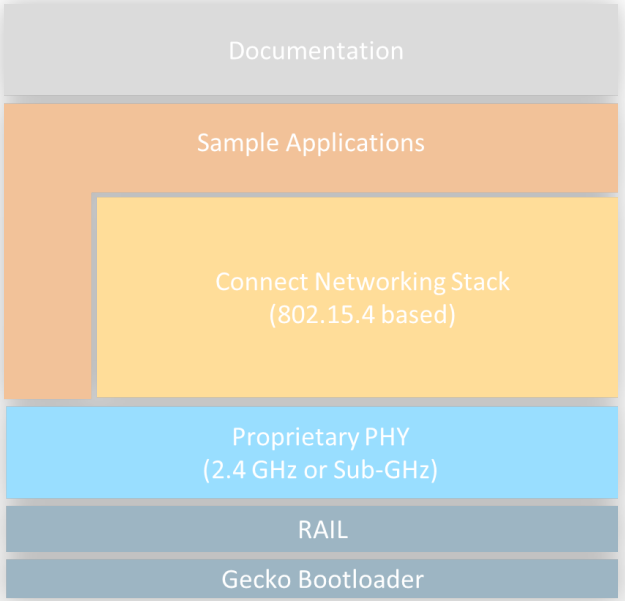
Proprietary

WIRELESS SOCS AND TRANSCEIVERS



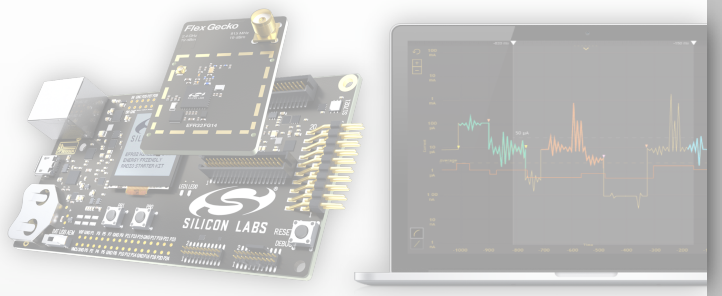
Ultra Low Power
2.4 GHz and Sub GHz
Industry Leading Multi-Protocol SoCs

FLEX SDK – RAIL API & CONNECT STACK



Predefined PHYs
Complete Development Suite
Sample Application With Source Code

TOOLS AND EASE-OF-USE



Radio boards
Simplicity Studio
Wireless Starter Kits

Introducing Wireless Gecko…

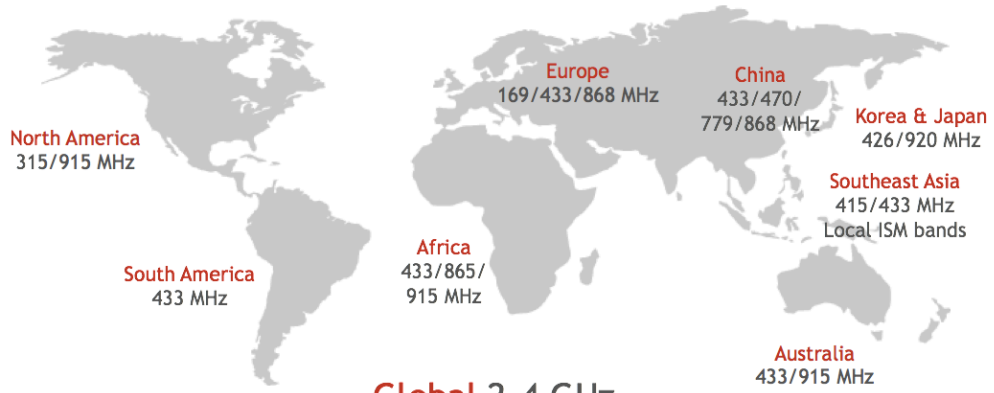
Proprietary

Multiband, Multiprotocol Wireless SoC for the IoT

Proprietary

Wi SUN

THREAD
zigbee
Bluetooth®



Global 2.4 GHz

- ARM® Cortex®-M4
- 2.4 GHz and sub-GHz
- Scalable Memory
  - Up to 1 MB Flash / 256 kB RAM
- Ultra low power
  - Down to 63 µA/MHz (active)
  - Down to 1.3 µA (sleep)
  - Integrated DC/DC converter
  - Autonomous peripherals
- High performance and integration
  - Up to +20 dBm PA
  - Down to -126dBm RX Sensitivity
  - 2.4 GHz balun
  - Advanced crypto block

Flex Gecko SoC Comparison

Proprietary

	EFR32FG1	FG12	FG13	FG14
Freq. Bands	2.4 GHz + Sub-GHz	→	→	→
Modulations	2/4(G)FSK, OOK, ASK, OQPSK, DBPSK, DSSS, FEC	→	→	→
Core	Cortex-M4 (40 MHz)	→	→	→
Max Flash	256 kB	1 MB	512 kB	256 kB
Max RAM	32 kB	256 kB	64 kB	32 kB
Security	AES-128/256, ECC, SHA-1, SHA-2	AES-128/256, ECC, SHA-1, SHA-2, TRNG, Radio CRYPTO	AES-128/256, ECC, SHA-1, SHA-2, TRNG, Radio CRYPTO	AES-128/256, ECC, SHA-1, SHA-2, TRNG
Max TX Power	+20 dBm	→	→	→
Max RX Sensitivity	-126 dBm	→	→	→
Active Current	63 µA/MHz	70 µA/MHz	69 µA/MHz	67 µA/MHz
Sleep Current	2.5 µA	1.5 µA	1.3 µA	→
2.4 GHz @ +0 dBm TX Current	8.2* mA	9.5 mA	8.5 mA	→
2.4 GHz RX Current	9.8* mA	10.8 mA	10.2 mA	→
868 MHz @ +14 dBm TX Current	34.5 mA	35.3 mA	→	→
868 MHz @ +20 dBm TX Current	84.9 mA	79.7 mA	→	→
868 MHz RX Current	8.1 mA	8.6 mA	9.6 mA	8.6 mA
Max GPIO	31	65	31	31
Other Features	IDAC	VDAC, LESENSE, OPAMP, Cap Sense	VDAC, LESENSE, OPAMP, Cap Sense	VDAC, LESENSE, OPAMP
Operating Voltage	+1.85 - 3.8 V	+1.8 – 3.8 V	→	→

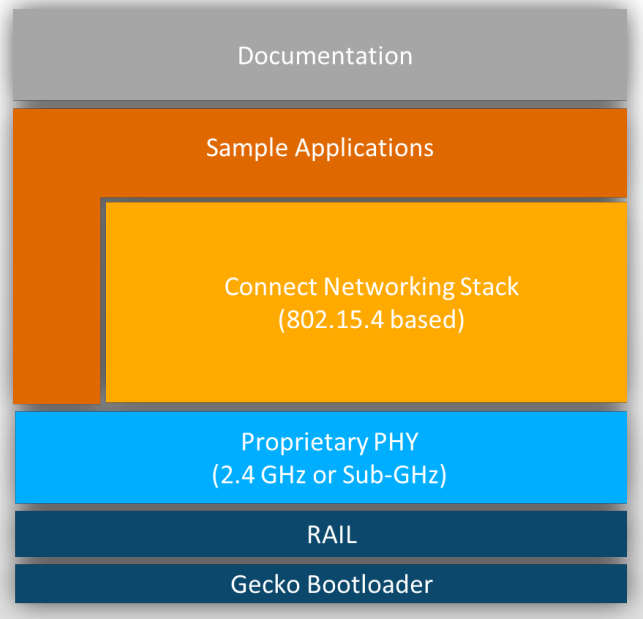
The Software

Proprietary

WIRELESS SOCS AND TRANSCEIVERS

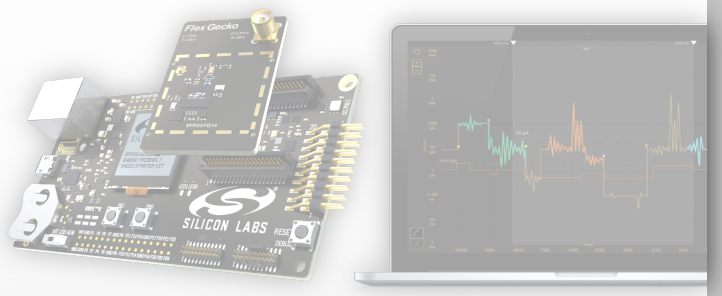
Ultra Low Power
2.4 GHz and Sub GHz
Industry Leading Multi-Protocol SoCs

FLEX SDK – RAIL API & CONNECT STACK



Predefined PHYs
Complete Development Suite
Sample Application With Source Code

TOOLS AND EASE-OF-USE

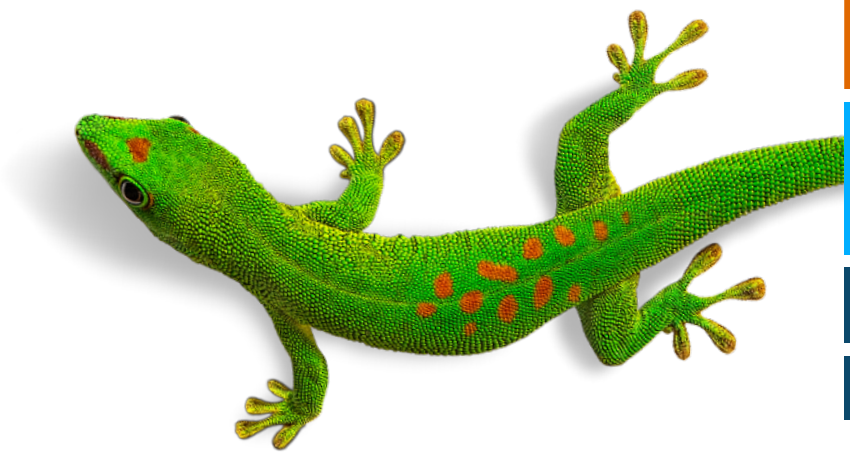


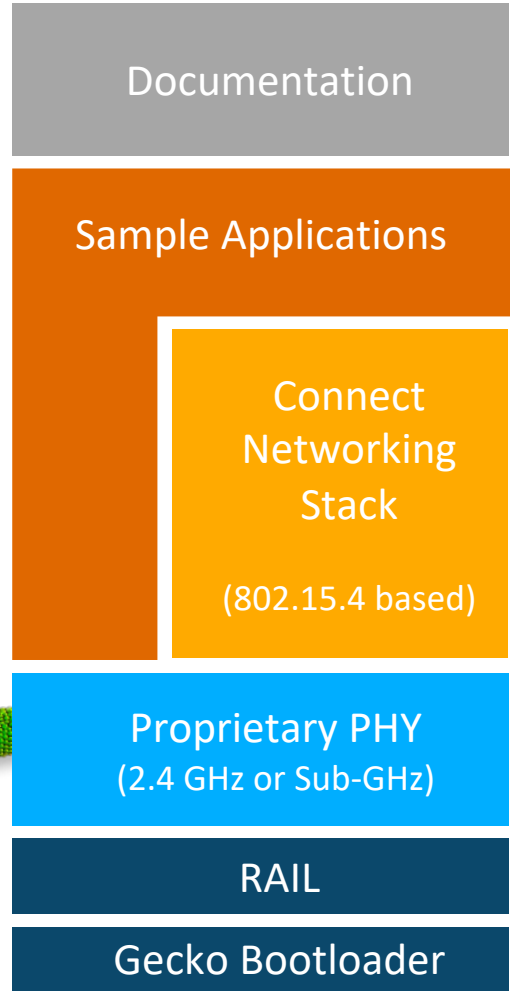
Radio boards
Simplicity Studio
Wireless Starter Kits

Flex SDK

Proprietary

- Complete software suite for proprietary wireless applications
  - Easy to use Radio Abstraction Interface Layer (**RAIL**) API
  - Feature Rich and Scalable **Connect** Networking Stack
  - Sample applications and extensive documentation
- Simplified Developer Experience through Simplicity Studio
  - Radio configurator
  - Application builder
  - Network analyzer
  - Energy profiler

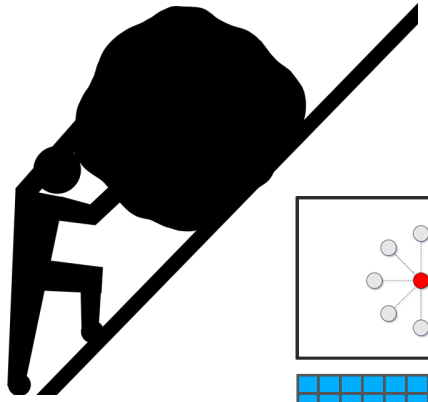


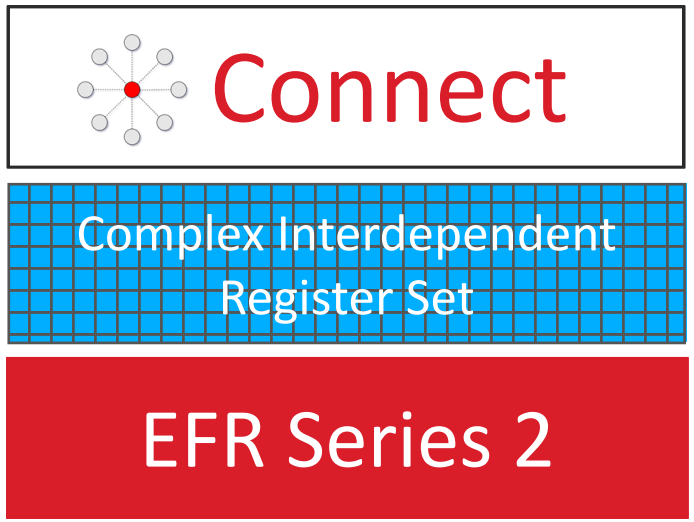


RAIL – Radio Abstraction Interface Layer

Proprietary

- Creating a Proprietary Protocol from scratch or porting a legacy protocol is no small task and can be a time-intensive and complicated process.
- A Proprietary Protocol designer has to understand every aspect of the Radio:
  - PHY layer to Regulatory Requirements
  - Packet Structure and Parsing to Network timing
- Traditionally, this feat was made more complex by thousands of hardware registers that controlled complex interdependent radio characteristics.
- RAIL was created to abstracting away those complexities in to easier to use API.
- Silicon Labs uses this same consistence RAIL Abstraction Layer to implement all of our Standard Protocols, making Multiprotocol possible.
- All Silicon Labs Energy Friendly Radios have the same consistence RAIL API, making migration between different EFR devices and EFR Series seamless.





RAIL API: https://docs.silabs.com/rail/latest/

RAIL API

Proprietary

Data path

TX
- Configuration
- Data Load
- Start TX
- Set TX Power
- Get TX Status
- TX events

Pre-TX
- Schedule TX
- CCA-CSMA
- CCA-LBT
- Set CCA threshold

RX
- Configuration
- Start/Schedule RX
- Get RSSI
- Start/Poll/Get Average RSSI
- RX events

Management path

Radio Configuration
- Radio Init
- Set Protocol
- Set Automatic TX/RX Transitions
- Packet/Frame configuration
- Channel configuration
- Bit Rate configuration
- Symbol Rate configuration
- PA tune configuration
- RF Sense configuration
- Load custom configuration
- RF Idle/Extended Idle
- Get Radio Entropy
- Get Version

Radio Calibration
- Calibration Init
- Calibration Start
- Calibration Status
- Calibration events

Auto ACK
- Configuration
- Enable/Disable
- Pause/Resume/Cancel
- Load/Use ACK Buffer
- Status

Address Filtering
- Configuration
- Enable/Disable/Reset Filter
- Enable/Disable Address
- Set Filter Address
- IsEnabled

Radio Diagnostic
- BER Start/Stop/GetStatus
- Direct Mode Configuration
- Set/Get Tune
- Start/Stop Stream
- Start/Stop TX Tone

BLE Specific
- Init/De-init BLE
- Set PHY Modes (1Mbps, 2Mbps, w/ Viterbi)
- Set channel RF parameters

802.15.4 Specific
- Init/De-init 802.15.4
- Set Short/Long Address
- Set PanId/Coordinator
- Set Promiscuous Mode
- Data request callback

Radio Timings
- Set/Get time base
- Start/Stop timer

Memory Manager Abstraction
- Allocate/Free Memory CBs
- Begin/End Write Memory events

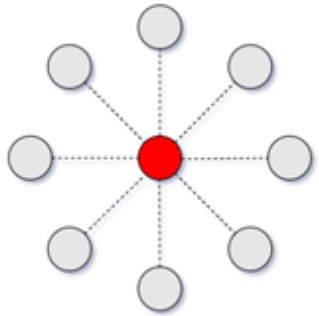
FIFO Management
- Data Configuration
- Reset FIFO
- Enable/Disable RX FIFO
- Set/Get RX/TX FIFO Thresholds
- Write TX FIFO
- Read RX FIFO
- FIFO events

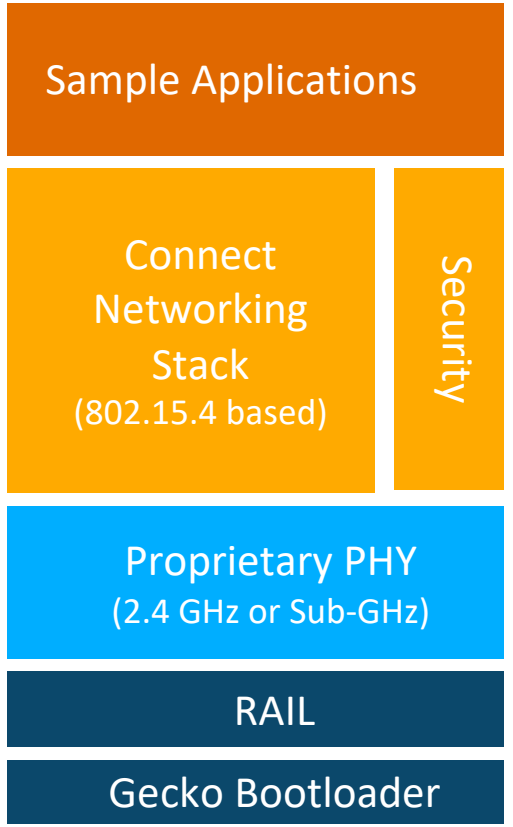
EFR32 HAL (interrupt handlers)

Connect Networking Stack

Proprietary

- Feature Rich Proprietary Wireless Networking Stack
  - Highly Scalable
    - Up to 2K nodes in extended STAR mode, Up to 65K nodes in Direct Mode
  - Small memory footprint (~75 KB Flash/~10 KB RAM)
  - Low Power modes
  - Dynamic Multi Protocol (DMP) with BLE
  - Built In Security
- Faster Time To Market
  - Ready to use and customizable PHYs
  - Application builder, Network analyzer and Energy profiler
  - GCC and IAR compiler support
  - Sample Applications
- Field Upgradable via OTA
  - Over-the-Air Firmware Updates (unicast & multicast)



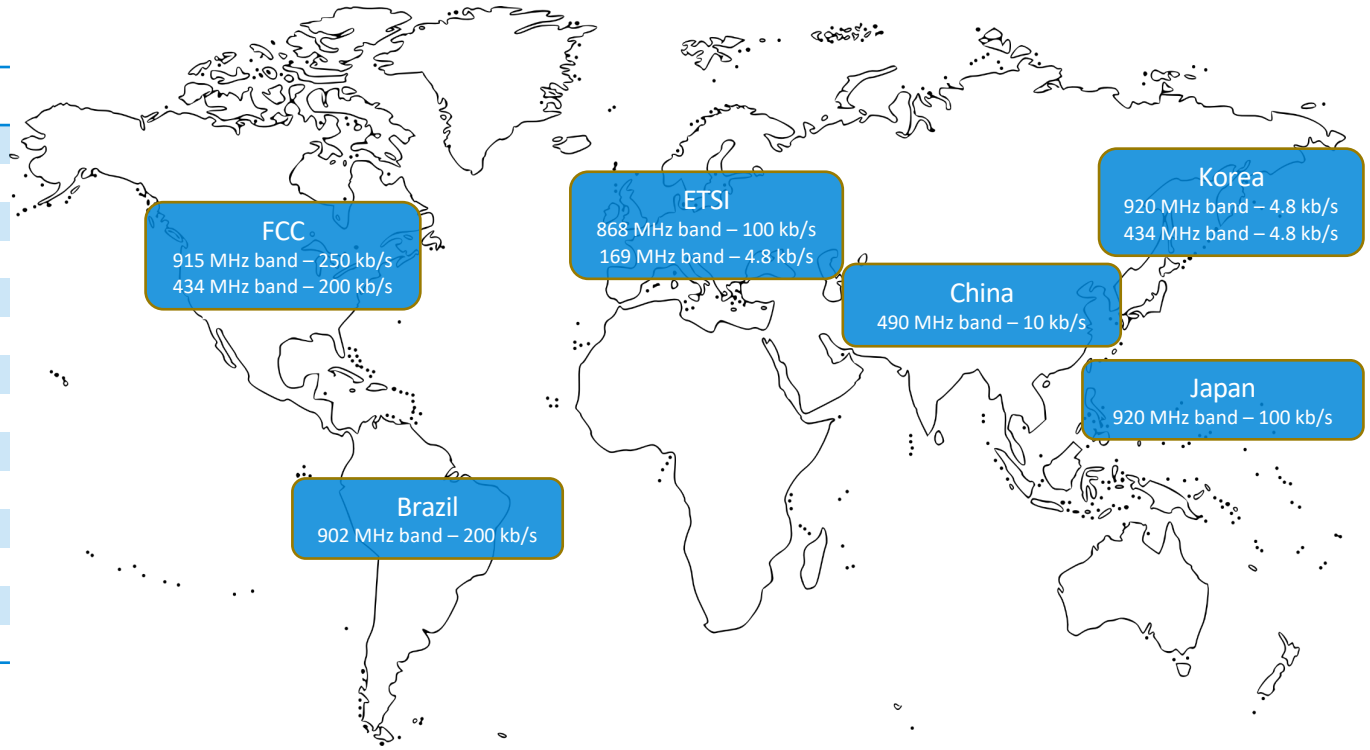


Connect Networking Stack : Worldwide PHYs

Proprietary

- Regional standards compliant

PHY profile	Frequency	Modulation	Bitrate
2.4GHz 802.15.4	2.4 GHz	OQPSK	250 kbps
DSSS 915-100	915 MHz	OQPSK	100 kbps (0.8Mcps)
DSSS 915-250	915 MHz	OQPSK	250 kbps (2Mcps)
915-500	915 MHz	2GFSK	500 kbps
DSSS 915-500	915 MHz	OQPSK	500 kbps
China 490	490 MHz	2GFSK	10 kbps
Europe 169	169 MHz	2GFSK	4.8 kbps
Europe 868	863 MHz	2GFSK	100 kbps
Japan 915	920 MHz	2GFSK	100 kbps
Korea 424	424 MHz	2GFSK	4.8 kbps
Korea 447	447 MHz	2GFSK	4.8 kbps
Korea 915	917 MHz	2GFSK	4.8 kbps
US FCC 434	434 MHz	2GFSK	200 kbps
US FCC 902, Brazil 902	902 MHz	2GFSK	200 kbps

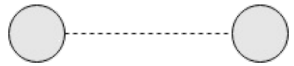


- Each RF configuration is tuned and tested by Silicon Labs for maximal performance
- Custom PHYs can be configured via Radio Configurator in Simplicity Studio

Connect Networking Stack - Topologies

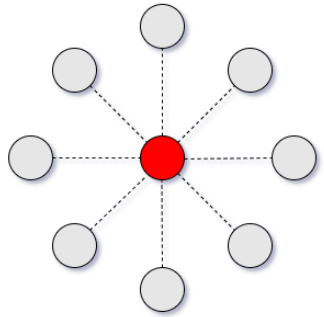
Proprietary

- Point to point
  - Simple communication between two devices



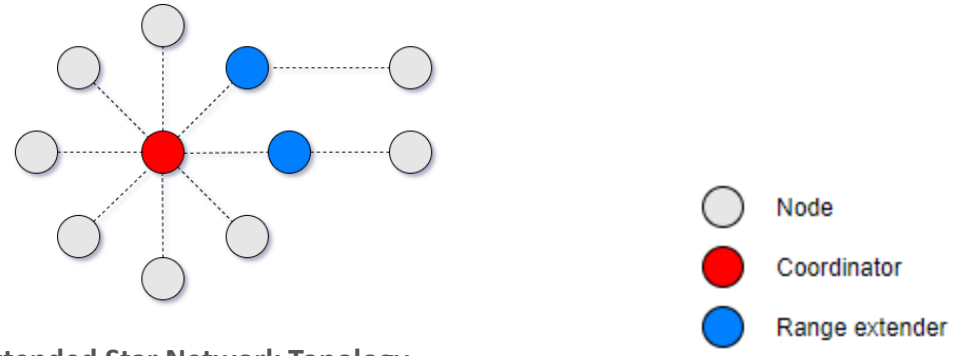
Point to Point

- Star
  - Single coordinator with multiple end devices
  - All communication passes through the coordinator
  - Up to 64 devices within radio range



Star Network Topology

- Extended star
  - Range extenders to extend the range of end-nodes
  - Up to 2048 devices

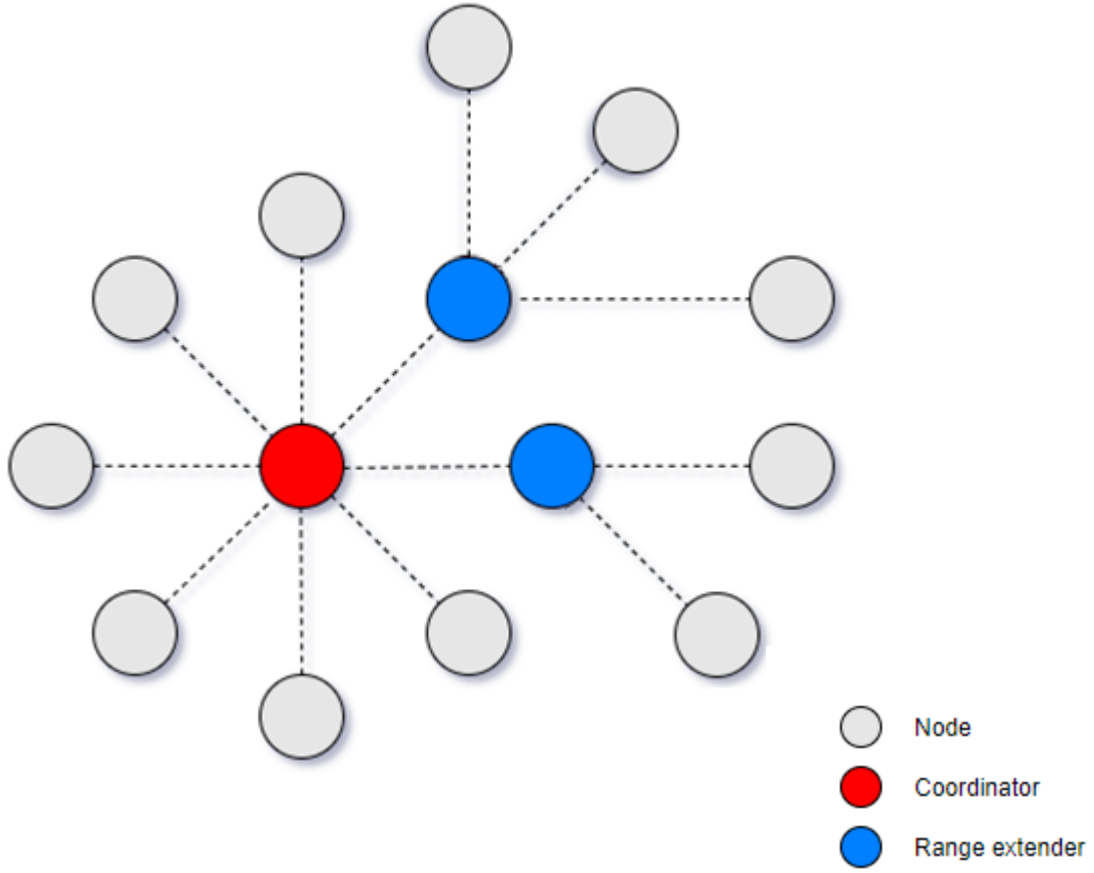


Extended Star Network Topology

Connect Networking Stack – Devices Types

Proprietary

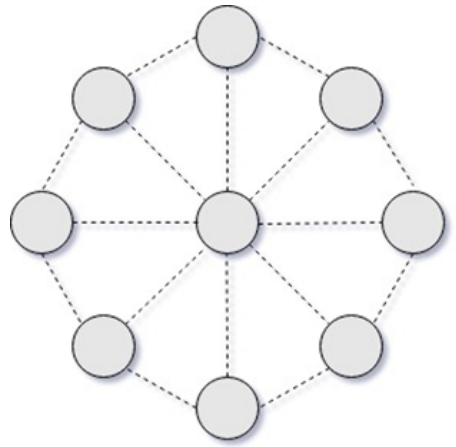
- Coordinator
  - Forms and manages the network - One coordinator per network
  - Typically Line Powered
  - Up to 64 nodes per coordinator
- Range extender
  - Extends the range of the nodes
  - Joins the coordinator
  - Typically Line Powered
  - Up to 32 nodes per range extender
- End Nodes
  - Joins the coordinator or a range extender
  - Typically Battery Powered
    - Can be configured as regular or sleepy nodes



Additional Connect Modes

Proprietary

- Direct Mode
  - No coordinator – All nodes are equal
  - Nodes talk to each other directly
  - No default routing capabilities
    - Customers can implement their own routing at the App layer
  - Highly scalable
    - Up to 65K nodes (limited by 16 bit address space)
  - Supports OTA
  - No Sleepy Nodes
- MAC Mode
  - Fully 802.15.4 compliant
  - PHY and MAC layer only, no upper layer functionality
  - Customer can reuse/implement their own network and upper layer functions

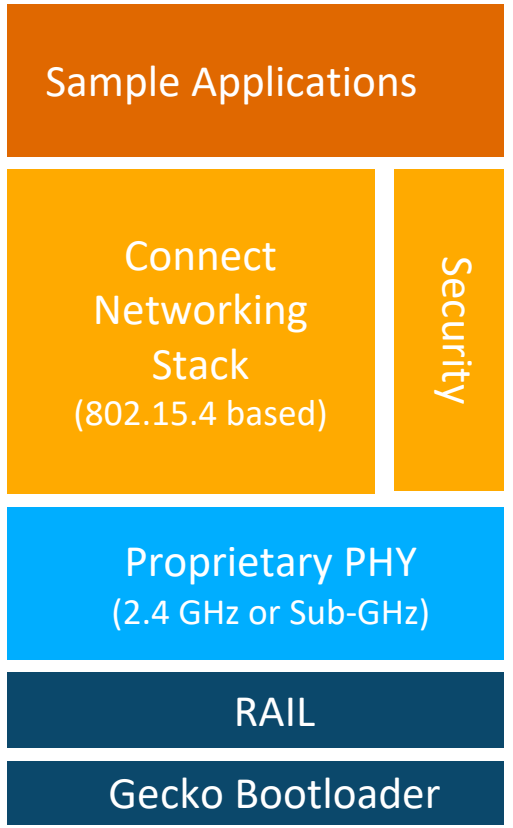


Direct Mode

Connect Networking Stack – Building Blocks

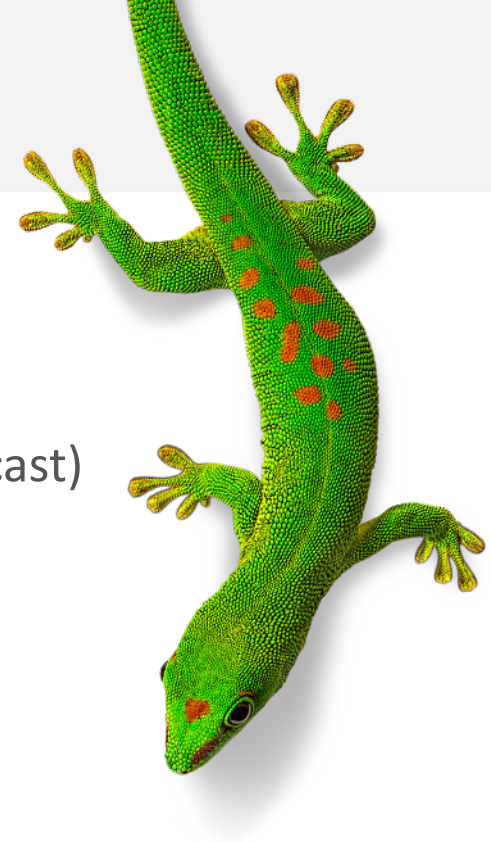
Proprietary

- Hardware Abstraction Layer (HAL) through EFR32 peripherals drivers
  - Common drivers for devices and their peripherals across the EFR32 platform
- PHY
  - PHY provided via RAIL with 802.15.4 acceleration
  - Fully tested and ready to use PHYs, Customizable via Radio Configurator
- Message builder/parser
  - 802.15.4 like PHY/MAC packet format builder/parser (max 127 bytes PHY frame)
  - Proprietary network layer format builder/parser
- Simulated EEPROM
  - Wear-leveled persistent storage of network and application data
- Event system
  - Allows the stack and the application to schedule code to run after certain events

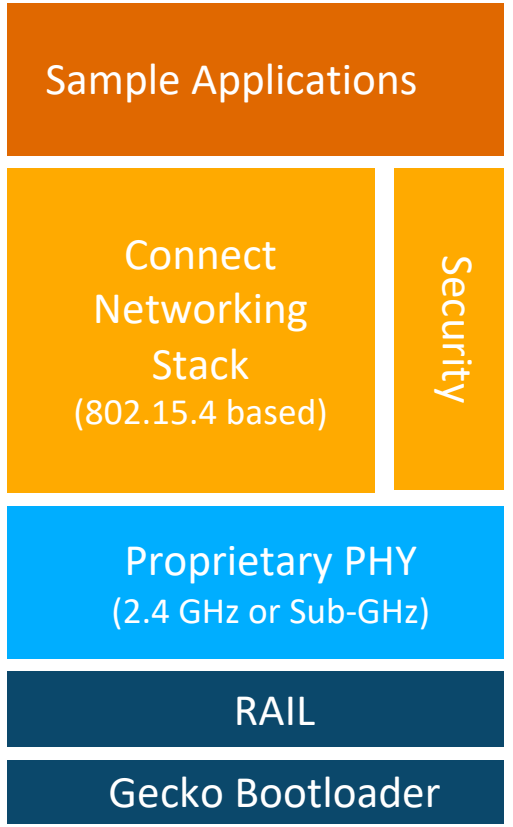


Connect Stack - Additional Features

Proprietary



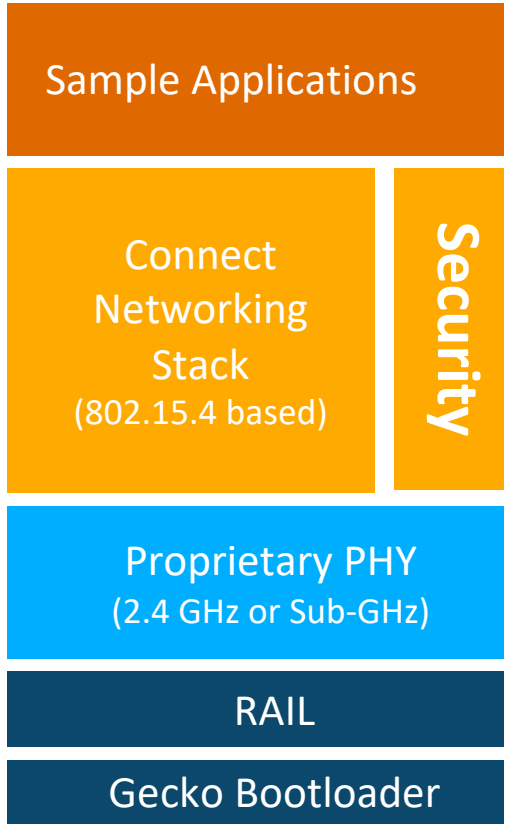
- Frequency hopping for regulatory compliance
- Serial and Over-the-air bootloaders (Broadcast and Unicast)
- Network Co-Processor (NCP) mode
- Memory manager with garbage collector
- CSMA-CA based MAC layer
- MAC security
- MAC queues for TX and RX
- Mailbox that allows server to buffer messages for client nodes typically sleepy nodes (default 25, max 254)
- Packet priorities



Connect Networking Stack - Security

Proprietary

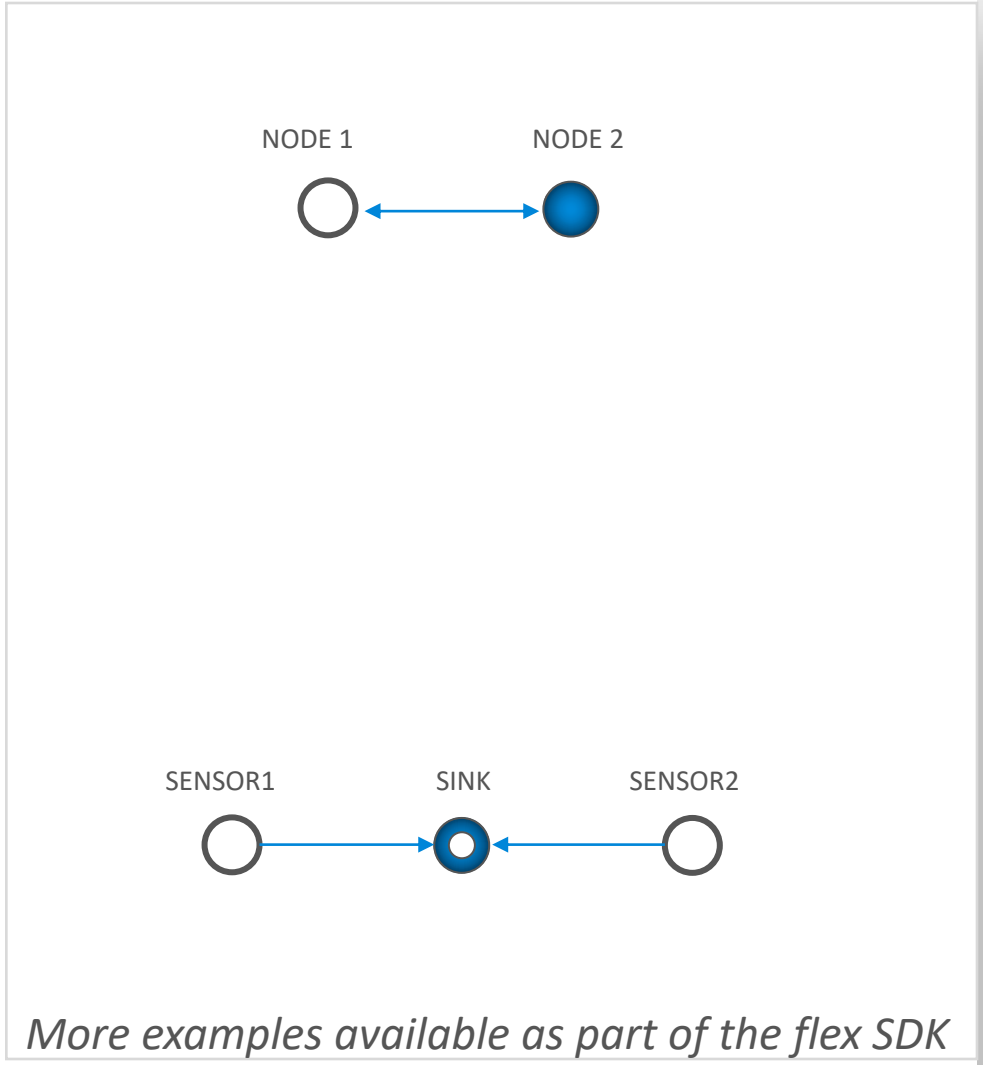
- 802.15.4 Mode 5 MAC Layer Security
- AES
  - All messages will be AES-128 encrypted and signed with a network wide pre-shared key provided to the device at commissioning
  - Enables nodes to exchange secured messages based on standard encryption/authentication scheme
  - Takes advantage of the AES hardware acceleration block available on EFR32
- Mbed TLS
  - Application Security



Example Projects

Proprietary

- Wire replacement
  - Bidirectional point-to-point network topology with two nodes
    - Master and Slave
  - Configurable Slave node
    - Regular slave (radio always on)
    - Sleepy slave (radio off most of the time)
- Commissioned Device
  - Single Hop Direct Network Topology
  - Any node can communicate to any other node directly
  - Network parameters are commissioned by the application
    - Node ID, PAN ID etc.
- Sensor/Sink
  - Star Network Topology
  - Sink node acts as network coordinator
  - Sensor nodes are joined to Sink and periodically report back
  - Sensor nodes can be regular or sleepy slave nodes

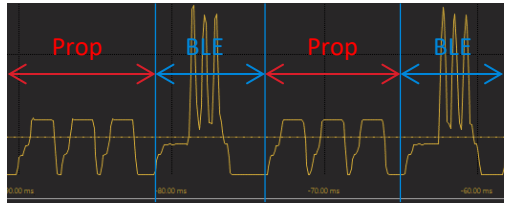


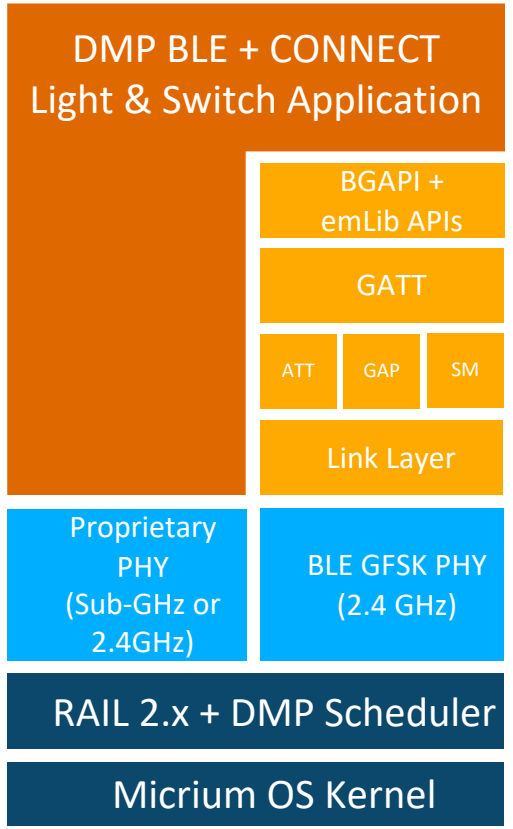
More examples available as part of the flex SDK

Dynamic Multiprotocol BLE + CONNECT

Proprietary

- Dynamic Multiprotocol BLE + CONNECT
  - Customer application can directly call CONNECT and BLE APIs
  - CONNECT and BLE APIs are thread safe
  - API to configure the priority of BLE and CONNECT TX / RX operations
    - By default CONNECT has lower priority
- Available today in Simplicity Studio
  - Dynamic Multiprotocol BLE + CONNECT Light and Switch example in Flex SDK
  - Wireless Gecko Mobile Application available for Android and iOS
  - Quick Start Guide (QSG) and User Guide (UG)
- For more information on DMP BLE + Connect:
  - Sub-GHz and Bluetooth Low Energy Multiprotocol Learning Center: https://www.silabs.com/products/wireless/learning-center/multiprotocol/sub-ghz-ble
  - Sub-GHz and Bluetooth Low Energy Multiprotocol Learning Center: https://www.silabs.com/products/wireless/learning-center/multiprotocol/sub-ghz-ble





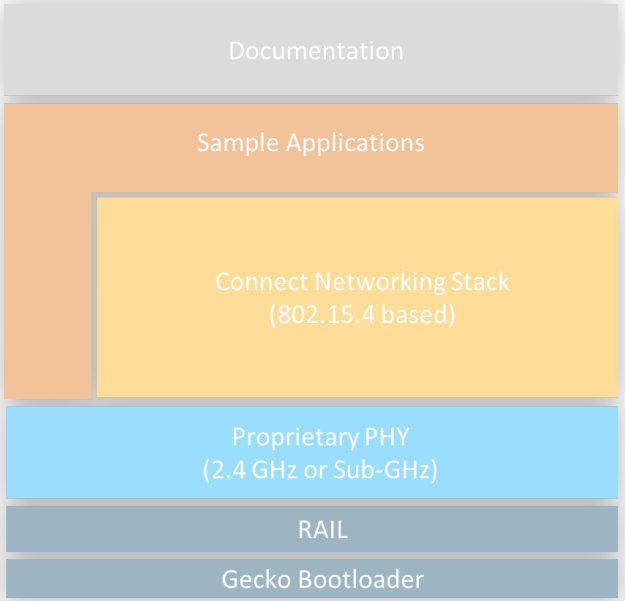
The Tools

Proprietary

WIRELESS SOCS AND TRANSCEIVERS

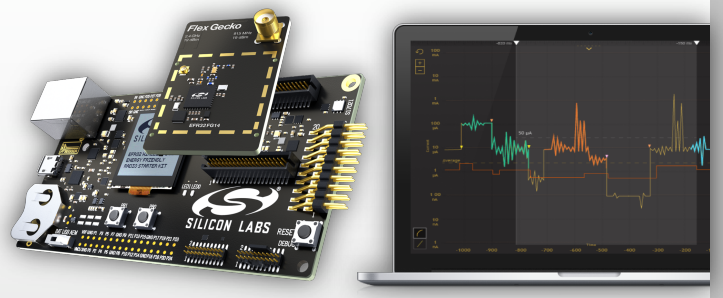
Ultra Low Power
2.4 GHz and Sub GHz
Industry Leading Multi-Protocol SoCs

FLEX SDK – RAIL API & CONNECT STACK



Predefined PHYs
Complete Development Suite
Sample Application With Source Code

TOOLS AND EASE-OF-USE



Radio boards
Simplicity Studio
Wireless Starter Kits

The Right Tool Set – Simplicity Studio™

Proprietary

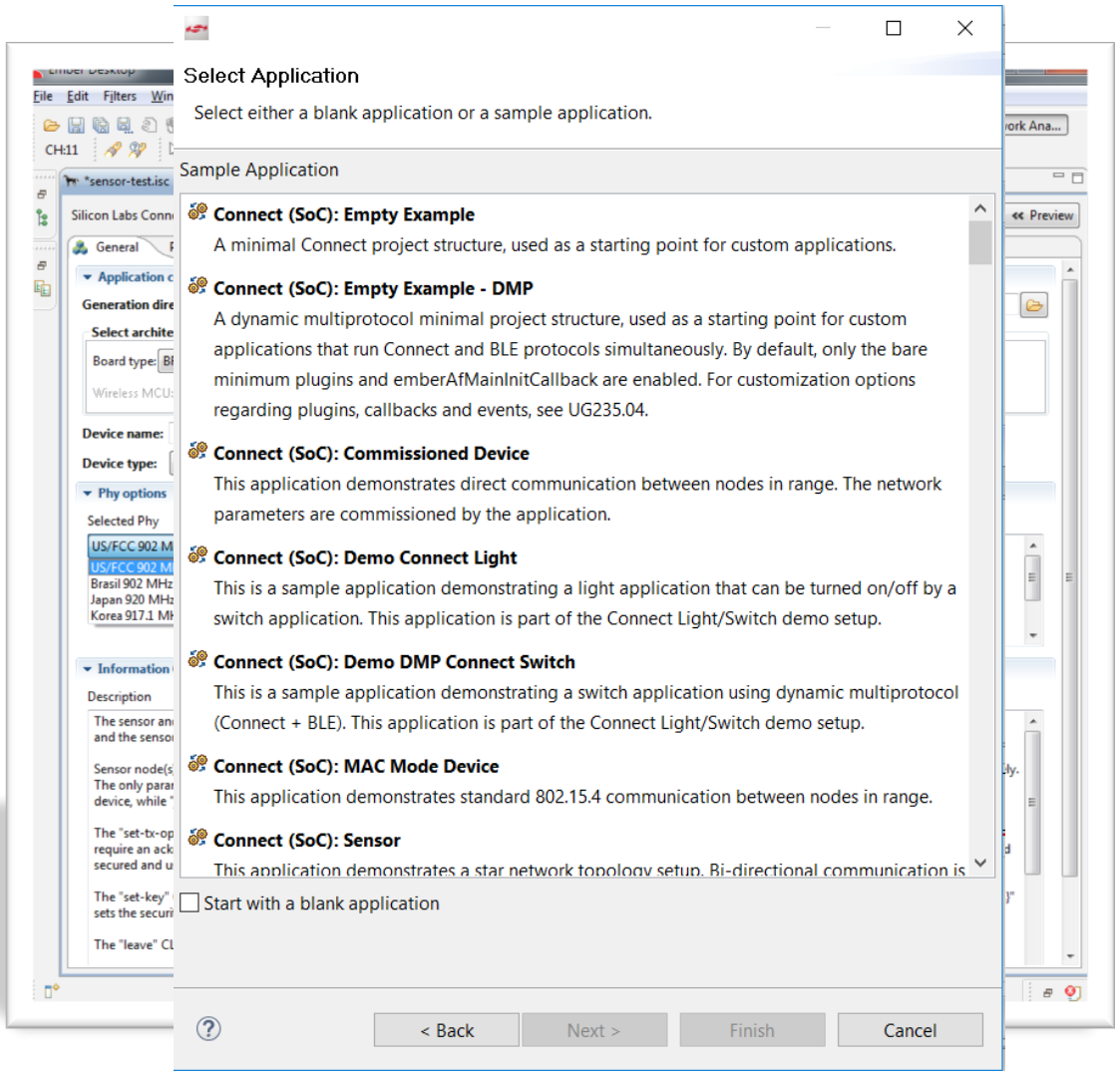
Wireless and MCU design made simple

- Common development environment for MCU and Wireless products
- Eclipse-based IDE with wireless stack support
- Complete Documentation
- Demos / Software Examples
- Advanced Tools
  - AppBuilder
  - Radio Configurator
  - Energy Profiler
  - Network Analyzer

Single Tool for Development

Application Builder – RAIL and Connect Stack

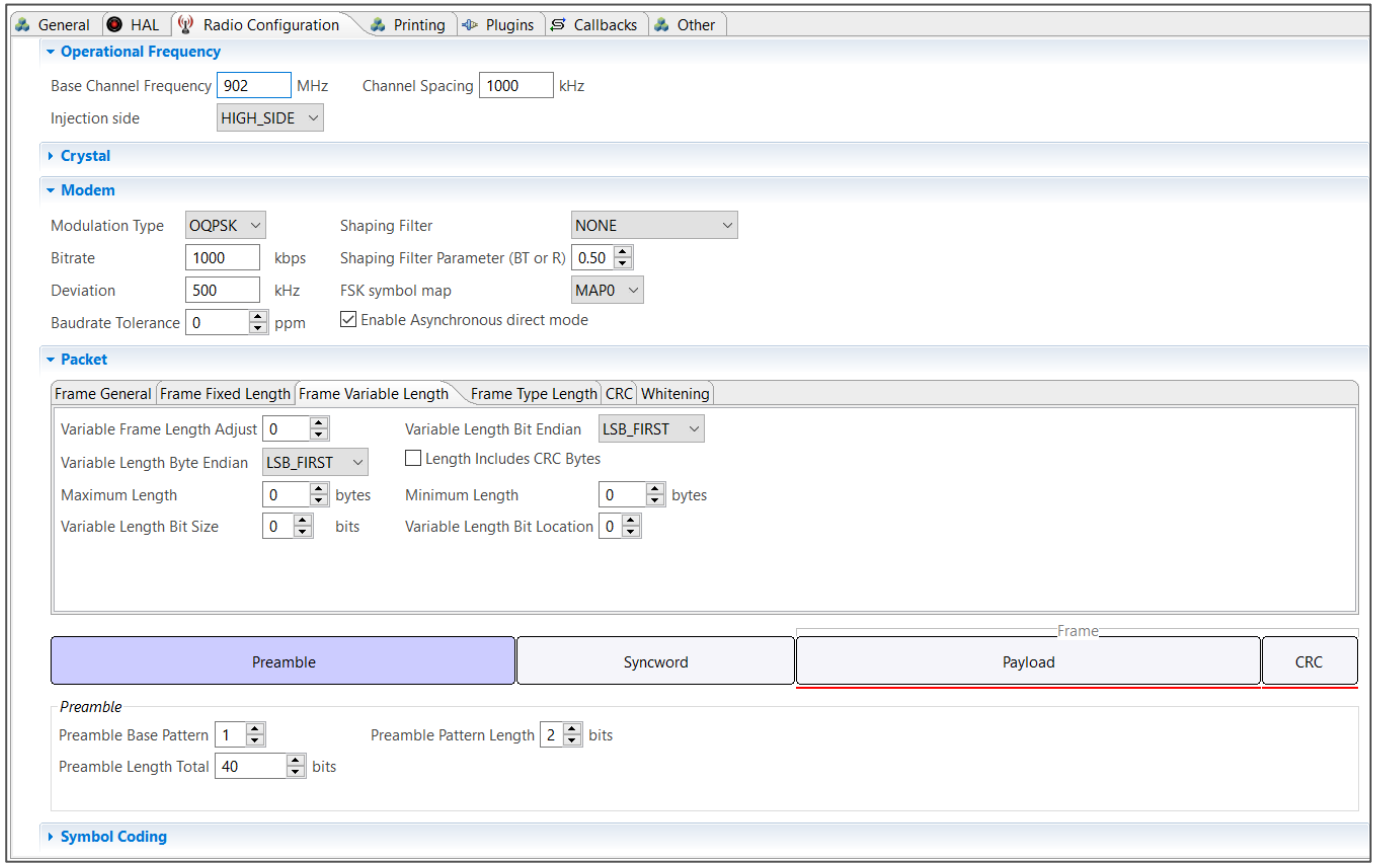
Proprietary



- Application Builder is a complete GUI solution for configuration of a stack
  - Fully integrated into Simplicity Studio
- Enables setup and configuration of entire Connect stack for desired application
- GUI-based configuration allows trade-off in code size / features
  - PHY modes (for worldwide regions)
  - Frequency hopping: On/off
  - Node type: Coordinator, Range Extender, End-node, Sleepy end-node
  - Association: Dynamic or fixed
  - OTA Bootloader: Enabled or disabled
  - Security: On/off
  - GPIO configuration
  - Debug mode: On/off
- Using AppBuilder for Proprietary Applications: https://www.silabs.com/documents/public/training/wireless/using-app-builder-proprietary-applications.pdf

Radio Configurator

Proprietary

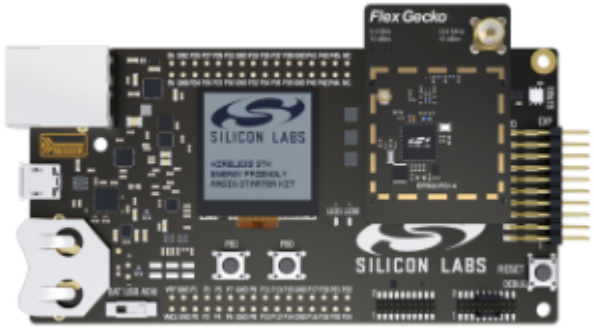


- Rapid Radio configuration and prototyping
  - Predefined PHY settings for most common world regions
  - Ability to create custom PHY settings for proprietary wireless applications
- Intuitive GUI to configure PHY parameters
  - Frequency bands, channel spacing, modulation
  - Bit rate, symbol maps, symbol coding, filtering
  - Timing detection, AFC, AGC and many other
- Quick learning curve for new radio engineers
  - Human readable configurations
  - No need to learn specific radio registers and other IC internal information

https://www.silabs.com/documents/public/application-notes/an971-efr32-radio-configurator-guide.pdf

Getting Started

1. Decided your target frequency and order a Flex Gecko Wireless Starter Kit



SLWSTK606xB

915MHz: 0
868MHz: 1
490MHz: 2
433MHz: 3
169MHz: 5

2. Install Simplicity Studio

https://www.silabs.com/products/development-tools/software/simplicity-studio

3. Follow the Quick Start Guide (QSG138):

"Getting Started with the Silicon Labs Flex SDK"

https://www.silabs.com/documents/public/quick-start-guides/qsg138-flex-efr32.pdf

4. Explore our online resources:

- Getting Started with RAIL: https://www.silabs.com/products/development-tools/software/radio-abstraction-interface-layer-sdk
- Getting Started with Connect: https://www.silabs.com/products/development-tools/software/connect-networking-stack

Useful Links on FLEX SDK

Proprietary

- RAIL API: https://docs.silabs.com/rail/latest/
- Connect API: https://docs.silabs.com/connect-stack/latest/
- Sub-GHz and Bluetooth Low Energy Multiprotocol Learning Center: https://www.silabs.com/products/wireless/learning-center/multiprotocol/sub-ghz-ble
- Getting Started with Sub-GHz and Bluetooth® Multiprotocol Development: https://www.silabs.com/support/getting-started/multiprotocol/sub-ghz-ble
- Getting Started with RAIL: https://www.silabs.com/products/development-tools/software/radio-abstraction-interface-layer-sdk
- Getting Started with Connect: https://www.silabs.com/products/development-tools/software/connect-networking-stack
- Using AppBuilder for Proprietary Applications: https://www.silabs.com/documents/public/training/wireless/using-app-builder-proprietary-applications.pdf
- QSG138 Quick start guide for the Flex SDK: https://www.silabs.com/documents/public/quick-start-guides/qsg138-flex-efr32.pdf
- AN971: EFR32 Radio Configurator Guide https://www.silabs.com/documents/public/application-notes/an971-efr32-radio-configuratorguide.pdf

BG22 Virtual Workshop

Learn how to develop and deploy more powerful, efficient, and secure IoT products with your own BG22 Thunderboard – free for all registrants!

Sessions had book up!

No worries, we are in the process of adding more.

So, check the link below for newly opening sessions and their dates:

Register today! https://www.silabs.com/about-us/events/virtual-bluetooth-workshop

Thank You!

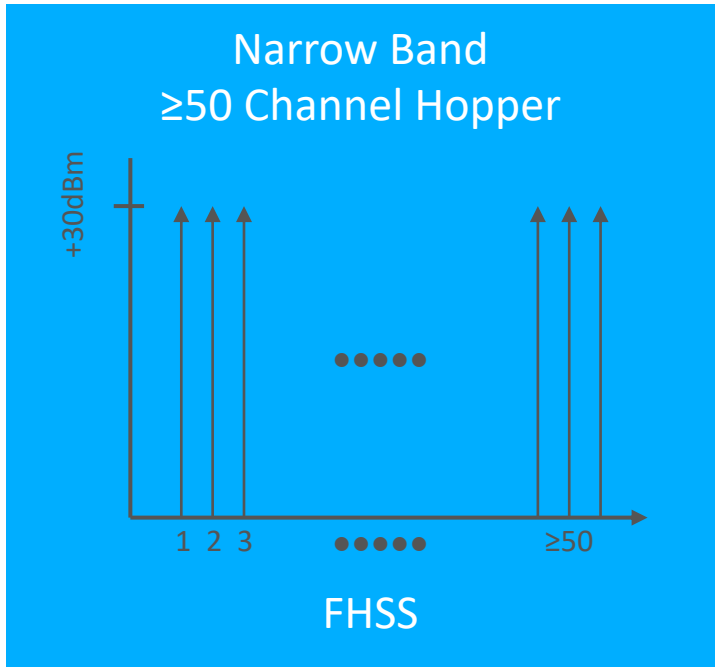
Q & A

Ways to Communicate 900MHz Band

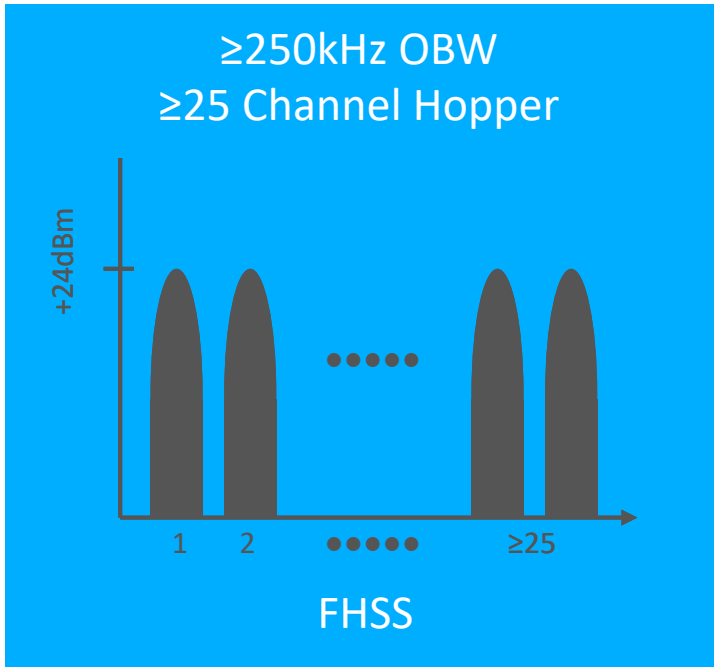
Proprietary

- Communication in the 902-926MHz Band is governed by FCC Part15.247 in the US. Canada, Brazil, and Australia have similar rules for this exact band.

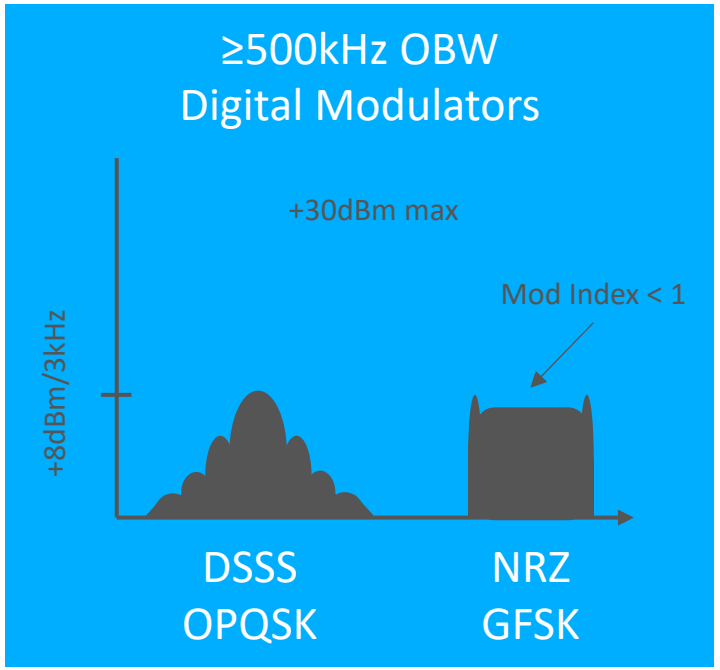
  https://www.govinfo.gov/content/pkg/CFR-2013-title47-vol1/pdf/CFR-2013-title47-vol1-sec15-247.pdf



Max Link Budget 156dBm



Max Link Budget 130dBm



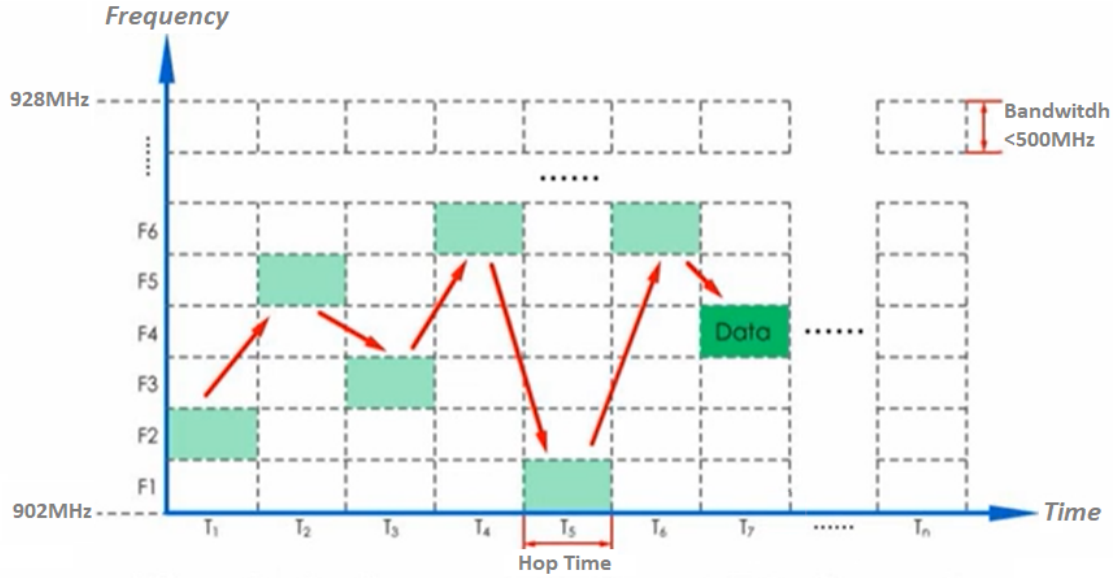
Max Link Budget 132dBm for DSSS
Max Link Budget 127dBm for GFSK

FHSS vs DSSS

Proprietary

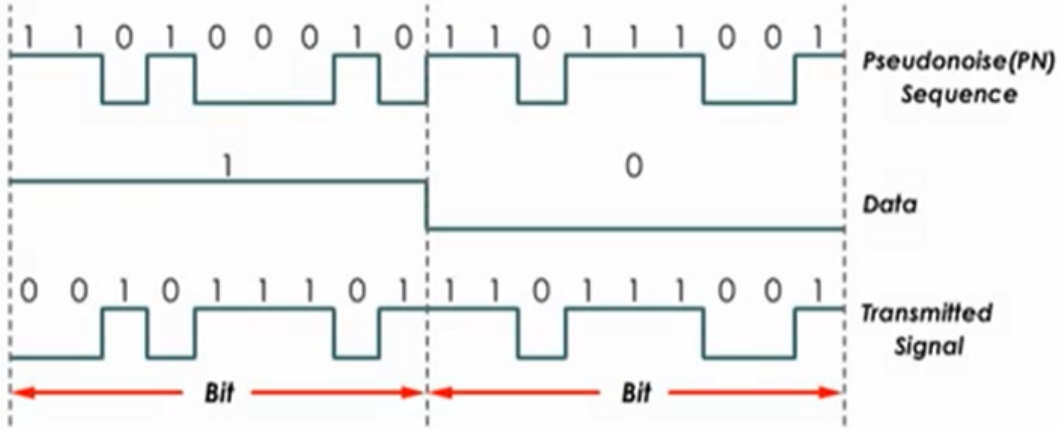
- Both provide protection from interferers while spreading the spectrum to avoid inferring with others.

Frequency-Hopping-Spread-Spectrum



- FHSS hops the transmitting Data from channel to channel in a pseudorandom way that “wears” all of the channels equally over time.

Direct-Sequence-Spread-Spectrum



- DSSS XORs a Psuedonoise Sequence (chips) with the Data to provide error correction or processing gain in the resulting transmitted signal.