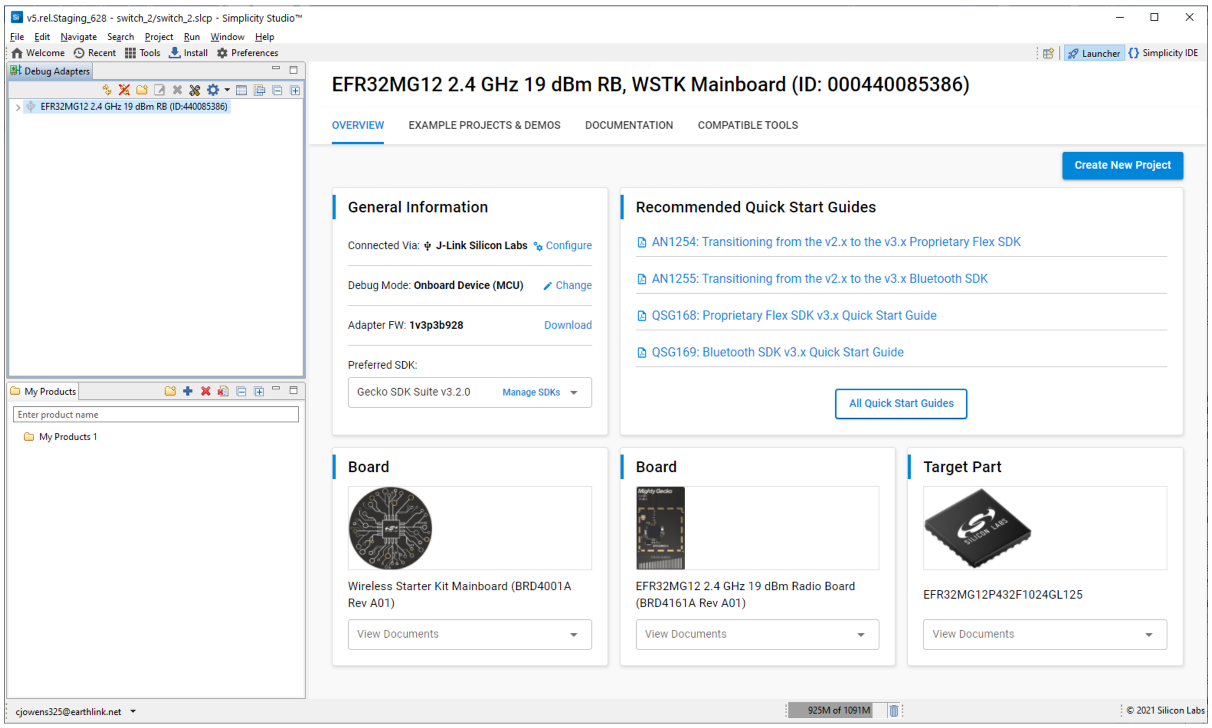
QSG169: *Bluetooth*® Quick-Start Guide for SDK v3.x and Higher

This document describes how to get started with Bluetooth development using the Bluetooth software development kit (SDK) v3.x and Simplicity Studio® 5 with a compatible wireless starter kit (WSTK). If you have purchased an EFR32BG WSTK you can first experiment with precompiled demos and an Android or iOS smartphone app before continuing with your own application development.

If you use Simplicity Studio 4 with Bluetooth SDK v2.x, find corresponding content in *QSG139: Bluetooth® SDK v2.x Quick Start Guide*.

KEY POINTS

- Introducing the Bluetooth development environment.
- Using the WSTK demos and Android or iOS smartphone app to demonstrate Bluetooth features.
- Starting application development for Bluetooth devices with Simplicity Studio.



EFR32MG12 2.4 GHz 19 dBm RB, WSTK Mainboard (ID: 000440085386)

General Information

Connected Via: J-Link Silicon Labs — Configure

Debug Mode: Onboard Device (MCU) — Change

Adapter FW: 1v3p3b928 — Download

Preferred SDK: Gecko SDK Suite v3.2.0 — Manage SDKs

Recommended Quick Start Guides

- AN1254: Transitioning from the v2.x to the v3.x Proprietary Flex SDK
- AN1255: Transitioning from the v2.x to the v3.x Bluetooth SDK
- QSG168: Proprietary Flex SDK v3.x Quick Start Guide
- QSG169: Bluetooth SDK v3.x Quick Start Guide

Board: Wireless Starter Kit Mainboard (BRD4001A Rev A01)

Board: EFR32MG12 2.4 GHz 19 dBm Radio Board (BRD4161A Rev A01)

Target Part: EFR32MG12P432F1024GL125

1 Introduction

This document describes how to get started with Bluetooth development using Silicon Labs products. It introduces the features of the Silicon Labs Bluetooth stack v3.x and the resources available to help with development. Application development is started using the Silicon Labs development environment Simplicity Studio 5 and the Bluetooth Software Development Kit (SDK) v3.x, part of Gecko SDK.

Simplicity Studio 5 includes everything needed for IoT product development with Silicon Labs devices including a resource and project launcher, software configuration tools, full IDE with GNU toolchain, and analysis tools. This document focuses on use and development in the Simplicity Studio 5 environment. Alternatively, Gecko SDK may be installed manually by downloading or cloning the latest from GitHub. See https://github.com/SiliconLabs/gecko_sdk for more information.

The SDK comes with a number of example application that you can then modify to create your own applications. If you are developing with an EFR32BG device and have purchased a EFR32BG Wireless Starter Kit (WSTK), you can use precompiled demos and an Android or iOS smartphone app to demonstrate Bluetooth software features.

This document describes the following:

- Bluetooth Stack features and components (see section 2 About the Bluetooth Stack)
- A description of the precompiled demos and example code available in the SDK (see section 3 About Demos and Examples)
- How to test prebuilt demo software with either an iOS or Android smartphone app (see section 4 Getting Started with Bluetooth Demo Software)
- How to develop your own applications in Simplicity Studio (see section 5 Starting Application Development)
- A description of other tools that are useful in the development process (see section 6 Development Tools)

1.1 Prerequisites

Before beginning application development, you should have:

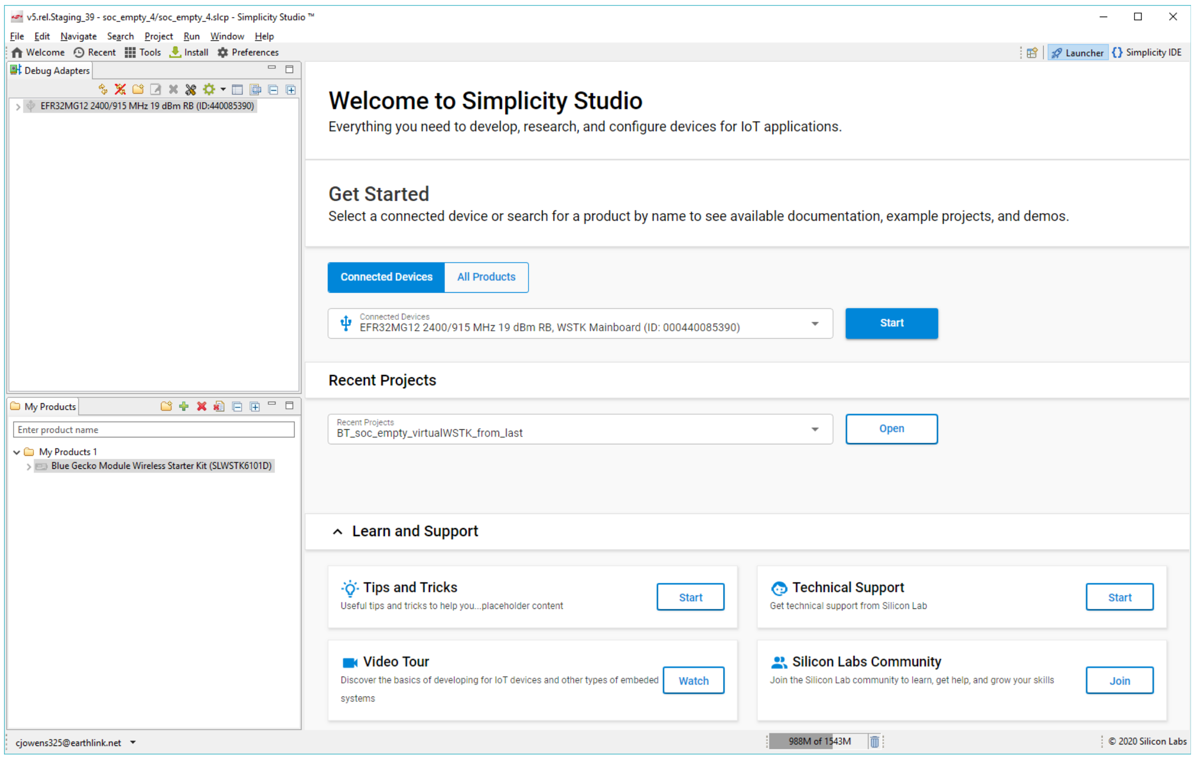
- Acquired a basic understanding of Bluetooth technology and terminology. *UG103.14: Bluetooth LE Fundamentals* provides a good starting point if you have not yet learned about Bluetooth.
- Purchased an EFR32BG Wireless Starter Kit or other compatible target hardware.
- Created an account at Silicon Labs. You can register at https://siliconlabs.force.com/apex/SL_CommunitiesSelfReg?form=short.
- Downloaded Simplicity Studio 5 and the Silicon Labs Gecko SDK containing the Bluetooth SDK and become generally familiar with the SSv5 Launcher perspective. SSv5 installation and getting started instructions along with a set of detailed references can be found in the online *Simplicity Studio 5 User's Guide*, available on https://docs.silabs.com/ and through the SSv5 help menu.
- Obtained a compatible compiler (See the Bluetooth SDK's release notes for the compatible versions):
  - Simplicity Studio comes with a free GCC C-compiler.
  - IAR Embedded Workbench for ARM (IAR-EWARM) can also be used as the compiler for Silicon Labs Bluetooth projects. Once IAR-EWARM is installed, the next time Simplicity Studio starts it will automatically detect and configure the IDE to use IAR-EWARM.

To get a 30-day evaluation license for IAR-EWARM:

- Go to the Silicon Labs support portal at https://www.silabs.com/support.
- Scroll down to the bottom of the page, and click **Contact Support**
- If you are not already signed in, sign in.
- Click the Software Releases tab. In the View list select **Development Tools**. Click **Go**. In the results is a link to the IAR-EWARM version named in the release notes.
- Download the IAR package (takes approximately 1 hour).
- Install IAR.
- In the IAR License Wizard, click **Register with IAR Systems to get an evaluation license**.
- Complete the registration and IAR will provide a 30-day evaluation license.
- Once IAR-EWARM is installed, the next time Simplicity Studio starts it will automatically detect and configure the IDE to use IAR-EWARM.

1.2 Support

You can access the Silicon Labs support portal at https://www.silabs.com/support through Simplicity Studio 5's Welcome view under Learn and Support. Use the support portal to contact Customer Support for any questions you might have during the development process.



Welcome to Simplicity Studio
Everything you need to develop, research, and configure devices for IoT applications.

Get Started
Select a connected device or search for a product by name to see available documentation, example projects, and demos.

Connected Devices: EFR32MG12 2400/915 MHz 19 dBm RB, WSTK Mainboard (ID: 000440085390)

Recent Projects: BT_soc_empty_virtualWSTK_from_last

Tips and Tricks — Useful tips and tricks to help you...placeholder content

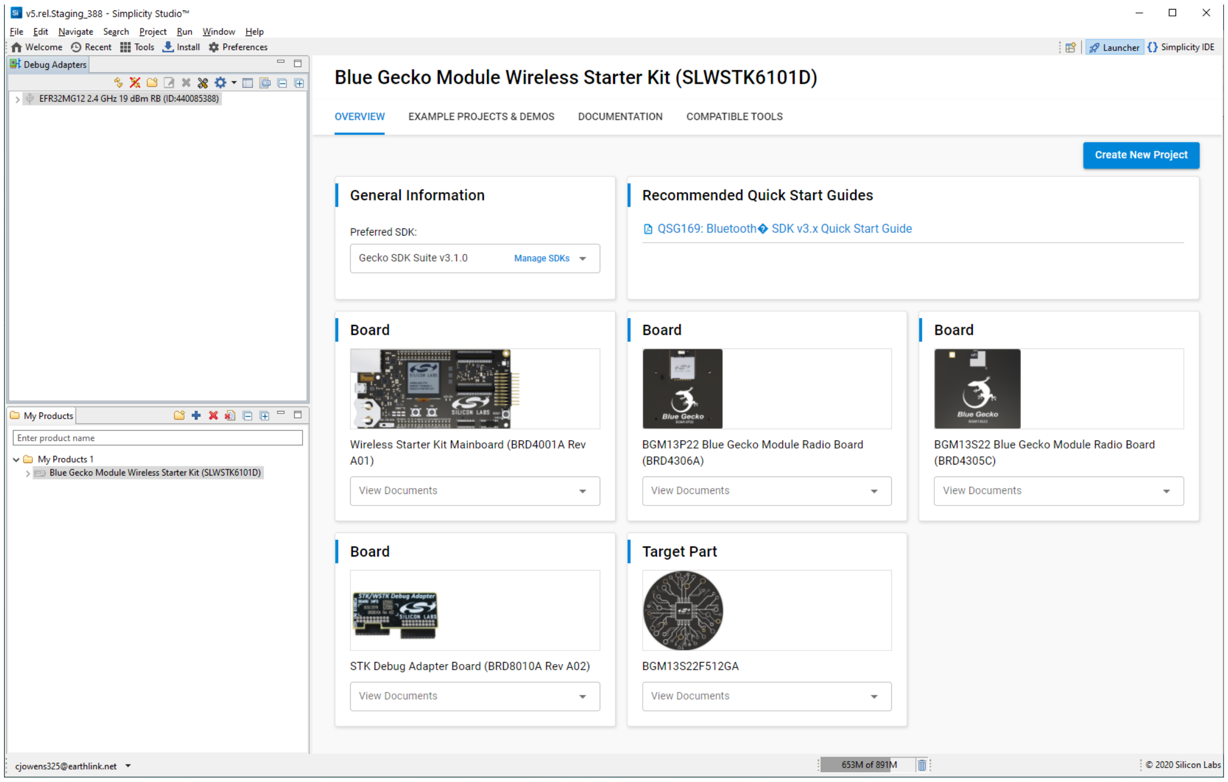
Technical Support — Get technical support from Silicon Lab

Video Tour — Discover the basics of developing for IoT devices and other types of embeded systems

Silicon Labs Community — Join the Silicon Lab community to learn, get help, and grow your skills

1.3 Documentation

Hardware-specific documentation may be accessed through links on the part Overview tab in Simplicity Studio 5.


v5.rel.Staging_388 - Simplicity Studio™
File Edit Navigate Search Project Run Window Help
Welcome Recent Tools Install Preferences
Launcher Simplicity IDE
Debug Adapters
EFR32MG12 2.4 GHz 19 dBm RB (ID:440085388)
My Products
Enter product name
My Products 1
Blue Gecko Module Wireless Starter Kit (SLWSTK6101D)
cjowens325@earthlink.net
653M of 891M
© 2020 Silicon Labs

Blue Gecko Module Wireless Starter Kit (SLWSTK6101D)

OVERVIEW | EXAMPLE PROJECTS & DEMOS | DOCUMENTATION | COMPATIBLE TOOLS

Create New Project

General Information
Preferred SDK:
Gecko SDK Suite v3.1.0 — Manage SDKs

Recommended Quick Start Guides
QSG169: Bluetooth� SDK v3.x Quick Start Guide

Board
Wireless Starter Kit Mainboard (BRD4001A Rev A01)
View Documents

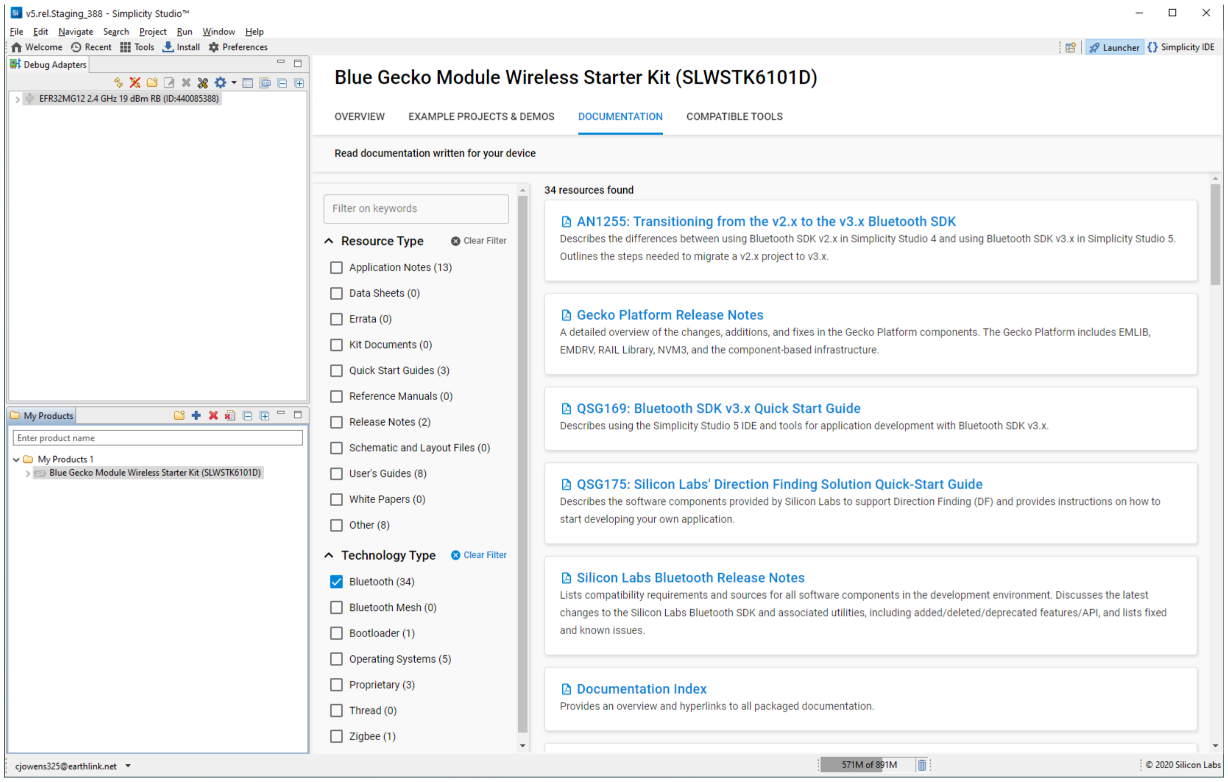
Board
BGM13P22 Blue Gecko Module Radio Board (BRD4306A)
View Documents

Board
BGM13S22 Blue Gecko Module Radio Board (BRD4305C)
View Documents

Board
STK Debug Adapter Board (BRD8010A Rev A02)
View Documents

Target Part
BGM13S22F512GA
View Documents

SDK documentation, User’s Guides, and other references are available through the Documentation tab.


v5.rel.Staging_388 - Simplicity Studio™
File Edit Navigate Search Project Run Window Help
Welcome Recent Tools Install Preferences
Launcher Simplicity IDE
Debug Adapters
EFR32MG12 2.4 GHz 19 dBm RB (ID:440085388)
My Products
Enter product name
My Products 1
Blue Gecko Module Wireless Starter Kit (SLWSTK6101D)
cjowens325@earthlink.net
571M of 891M
© 2020 Silicon Labs

Blue Gecko Module Wireless Starter Kit (SLWSTK6101D)

OVERVIEW | EXAMPLE PROJECTS & DEMOS | DOCUMENTATION | COMPATIBLE TOOLS

Read documentation written for your device

Filter on keywords

Resource Type — Clear Filter
- [ ] Application Notes (13)
- [ ] Data Sheets (0)
- [ ] Errata (0)
- [ ] Kit Documents (0)
- [ ] Quick Start Guides (3)
- [ ] Reference Manuals (0)
- [ ] Release Notes (2)
- [ ] Schematic and Layout Files (0)
- [ ] User's Guides (8)
- [ ] White Papers (0)
- [ ] Other (8)

Technology Type — Clear Filter
- [x] Bluetooth (34)
- [ ] Bluetooth Mesh (0)
- [ ] Bootloader (1)
- [ ] Operating Systems (5)
- [ ] Proprietary (3)
- [ ] Thread (0)
- [ ] Zigbee (1)

34 resources found

AN1255: Transitioning from the v2.x to the v3.x Bluetooth SDK
Describes the differences between using Bluetooth SDK v2.x in Simplicity Studio 4 and using Bluetooth SDK v3.x in Simplicity Studio 5. Outlines the steps needed to migrate a v2.x project to v3.x.

Gecko Platform Release Notes
A detailed overview of the changes, additions, and fixes in the Gecko Platform components. The Gecko Platform includes EMLIB, EMDRV, RAIL Library, NVM3, and the component-based infrastructure.

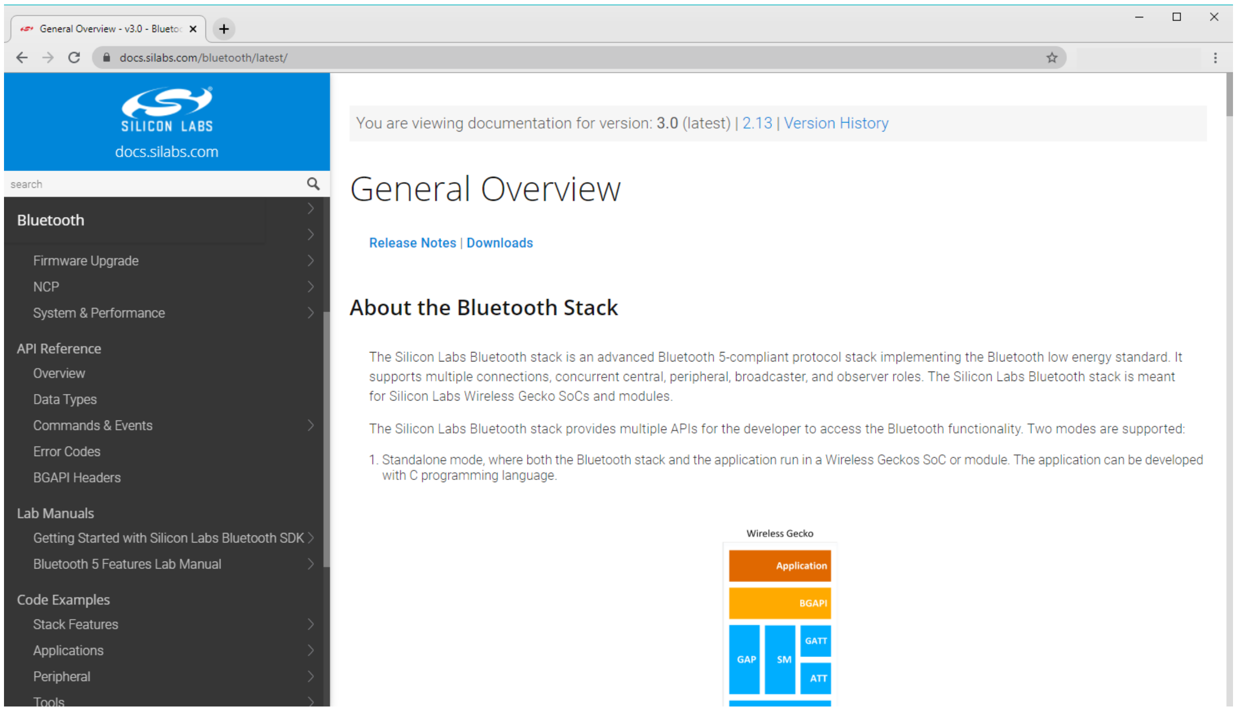
QSG169: Bluetooth SDK v3.x Quick Start Guide
Describes using the Simplicity Studio 5 IDE and tools for application development with Bluetooth SDK v3.x.

QSG175: Silicon Labs' Direction Finding Solution Quick-Start Guide
Describes the software components provided by Silicon Labs to support Direction Finding (DF) and provides instructions on how to start developing your own application.

Silicon Labs Bluetooth Release Notes
Lists compatibility requirements and sources for all software components in the development environment. Discusses the latest changes to the Silicon Labs Bluetooth SDK and associated utilities, including added/deleted/deprecated features/API, and lists fixed and known issues.

Documentation Index
Provides an overview and hyperlinks to all packaged documentation.

The Bluetooth API reference is available online along with further useful documentation and code examples: https://docs.silabs.com/bluetooth/latest/



You are viewing documentation for version: 3.0 (latest) | 2.13 | Version History

General Overview

Release Notes | Downloads

About the Bluetooth Stack

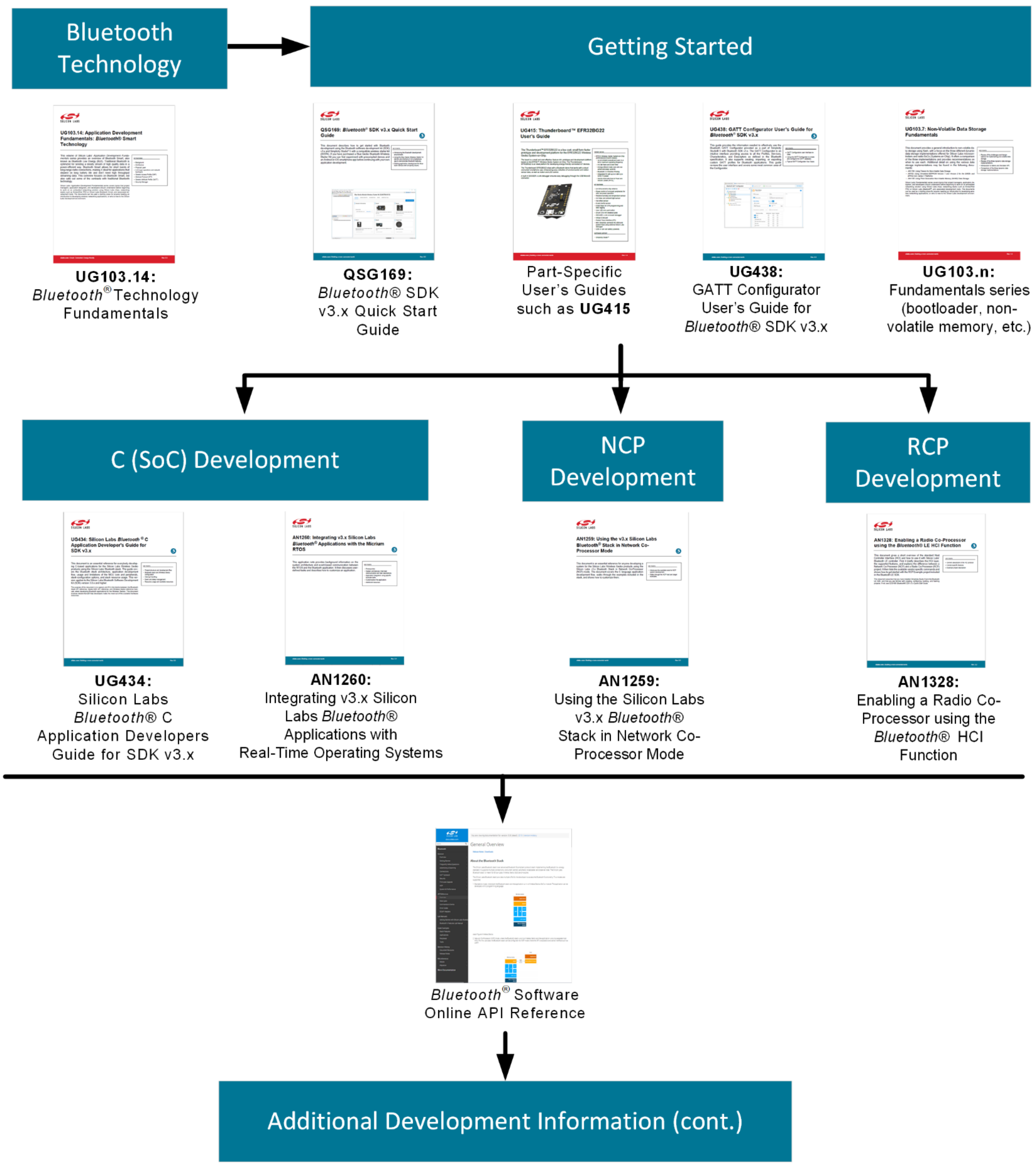
The Silicon Labs Bluetooth stack is an advanced Bluetooth 5-compliant protocol stack implementing the Bluetooth low energy standard. It supports multiple connections, concurrent central, peripheral, broadcaster, and observer roles. The Silicon Labs Bluetooth stack is meant for Silicon Labs Wireless Gecko SoCs and modules.

The Silicon Labs Bluetooth stack provides multiple APIs for the developer to access the Bluetooth functionality. Two modes are supported:

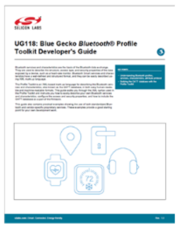
1. Standalone mode, where both the Bluetooth stack and the application run in a Wireless Geckos SoC or module. The application can be developed with C programming language.

Training materials are available on the Silicon Labs website at https://www.silabs.com/support/training/bluetooth.

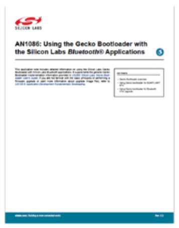
Key documentation for the Bluetooth SDK is summarized in the following figures.

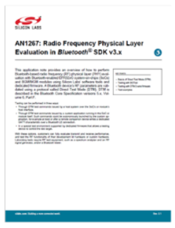

Bluetooth Technology
Getting Started
UG103.14: *Bluetooth*® Technology Fundamentals
QSG169: *Bluetooth*® SDK v3.x Quick Start Guide
Part-Specific User's Guides such as **UG415**
UG438: GATT Configurator User's Guide for *Bluetooth*® SDK v3.x
UG103.n: Fundamentals series (bootloader, non-volatile memory, etc.)
C (SoC) Development
NCP Development
RCP Development
UG434: Silicon Labs *Bluetooth*® C Application Developers Guide for SDK v3.x
AN1260: Integrating v3.x Silicon Labs *Bluetooth*® Applications with Real-Time Operating Systems
AN1259: Using the Silicon Labs v3.x *Bluetooth*® Stack in Network Co-Processor Mode
AN1328: Enabling a Radio Co-Processor using the *Bluetooth*® HCI Function
Bluetooth® Software Online API Reference
Additional Development Information (cont.)

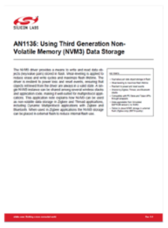
Additional Development Information

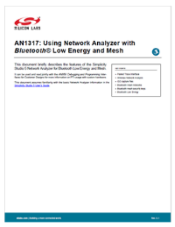

UG118:
Blue Gecko *Bluetooth*® Profile Toolkit Developer's Guide

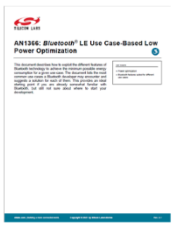
UG489:
Silicon Labs Gecko Bootloader User's Guide for GSDK 4.0 and Higher

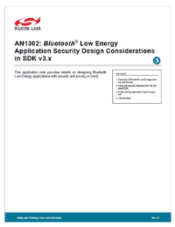

AN1086:
Using the Gecko Bootloader with Silicon Labs *Bluetooth*® Applications


AN1267:
Radio Frequency Physical Layer Evaluation in *Bluetooth*® SDK v3.x


AN1135:
Using Third Generation Non-Volatile Memory (NVM3) Data Storage


AN1317:
Using Network Analyzer with *Bluetooth*® Mesh and Low Energy


AN1366:
Bluetooth® LE Use Case-Based Low Power Optimization


AN1302:
Bluetooth® Low Energy Application Security Design Considerations in SDK v3.x

AN1362:
Amazon FreeRTOS Architecture and Sample Applications

1.4 Gecko Platform

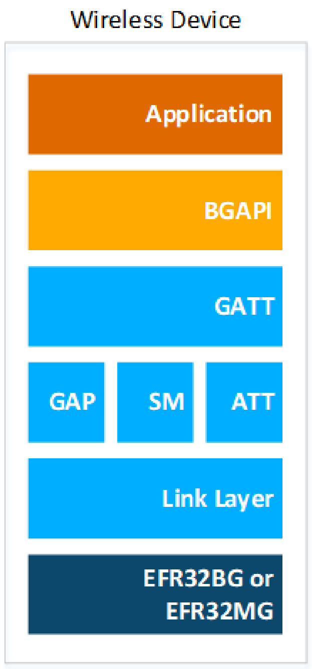
The Gecko Platform is a set of drivers and other lower layer features that interact directly with Silicon Labs chips and modules. Gecko Platform components include EMLIB, EMDRV, RAIL Library, NVM3, and mbed TLS. For more information about Gecko Platform, see release notes that can be found in Simplicity Studio's Documentation tab.

2 About the Bluetooth Stack

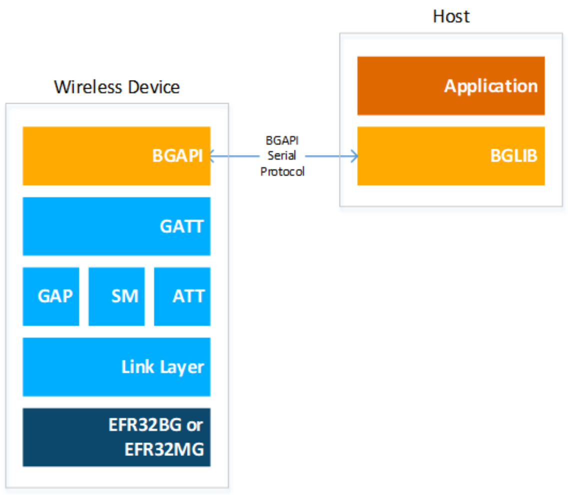
The v3.x Silicon Labs Bluetooth stack is an advanced Bluetooth 5-compliant protocol stack implementing the Bluetooth low energy standard. It supports multiple connections, concurrent central, peripheral, broadcaster, and observer roles. The v3.x Silicon Labs Bluetooth stack is meant for Silicon Labs EFR32 SoCs and modules.

The Silicon Labs Bluetooth stack provides multiple APIs for the developer to access the Bluetooth functionality. Three modes are supported:

1. Standalone mode, where both the Bluetooth stack and the application run in an EFR32SoC or module. The application can be developed with C programming language.

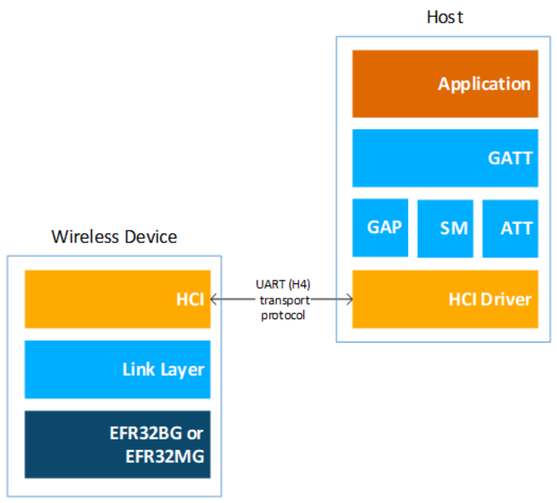


2. Network Co-Processor (NCP) mode, where the Bluetooth stack runs in an EFR32 and the application runs on a separate host MCU. For this use case, the Bluetooth stack can be configured into NCP mode where the API is exposed over a serial inter- face such as UART.



3. Radio Co-Processor (RCP) mode, where only the Link Layer of the Bluetooth stack runs on the EFR32, and the Host Layer of the stack, as well as the application, runs on a separate host MCU or PC. In this use case, the Host Layer is developed by a third party, since Silicon Labs' Bluetooth stack is only built for EFR32 SoCs / modules. The Link Layer and the host layer communicate via HCI

(Host-Controller Interface), which is a standard interface between the two layers. The HCI can be accessed via UART following the Bluetooth SIG's UART (H4) transport protocol or the Silicon Labs' proprietary CPC (Co-Processor Communication) protocol.



2.1 Bluetooth Stack Features

The features of the Silicon Labs Bluetooth stack are listed in the following table.

Feature	EFR32[B|M]G1	EFR32[B|M]G12	EFR32[B|M]G13	EFR32[B|M]G21*	EFR32[B|M]G22	EFR32[B|M]G24
Bluetooth version	Bluetooth 5.3	Bluetooth 5.3	Bluetooth 5.3	Bluetooth 5.3	Bluetooth 5.3	Bluetooth 5.3
Concurrent central, peripheral, broadcaster and observer modes	✓	✓	✓	✓	✓	✓
Simultaneous connections	✓ (up to 8)	✓ (up to 32)	✓ (up to 8)	✓ (up to 32)	✓ (up to 8)	✓ (up to 32)
LE secure connections	✓	✓	✓	✓	✓	✓
LE Privacy 1.2 (peripheral)	✓	✓	✓	✓	✓	✓
LE packet length extensions	✓	✓	✓	✓	✓	✓
LE dual topology	✓	✓	✓	✓	✓	✓
Link Layer Device Filtering (central side only)	✓	✓	✓	✓	✓	✓
LE Power Control	✓	✓	✓	✓	✓	✓
Bluetooth 5 GATT caching	✓	✓	✓	✓	✓	✓
Bluetooth 5 2M PHY	x	✓	✓	✓	✓	✓
Bluetooth 5 LE Long Range	x	x	✓	✓	✓	✓
Bluetooth 5 advertisement sets and scan event reporting	✓	✓	✓	✓	✓	✓

Feature	EFR32[B|M]G1	EFR32[B|M]G12	EFR32[B|M]G13	EFR32[B|M]G21*	EFR32[B|M]G22	EFR32[B|M]G24
Bluetooth 5 extended advertisements.	x	✓ (up to 1650B)	✓ (up to 1650B)	✓ (up to 1650B)	✓ (up to 1650B)	✓ (up to 1650B)
Bluetooth 5 periodic advertisements	x	✓ (up to 1650B)	✓ (up to 1650B)	✓ (up to 1650B)	✓ (up to 1650B)	✓ (up to 1650B)
Bluetooth 5 periodic advertising synchronization	x	✓	✓	✓	✓	✓
Directed advertising	✓	✓	✓	✓	✓	✓
Adaptive Frequency Hopping	✓	✓	✓	✓	✓	✓
L2CAP Connection Oriented Channels	✓	✓	✓	✓	✓	✓
CTE transmitter	x	x	x	x	✓	✓
CTE receiver	x	x	x	x	select part numbers	select part numbers
Maximum throughput	700 kbps over 1M PHY	700 kbps over 1M PHY, 1300 kbps over 2M PHY	700 kbps over 1M PHY, 1300 kbps over 2M PHY	700 kbps over 1M PHY, 1300 kbps over 2M PHY	700 kbps over 1M PHY, 1300 kbps over 2M PHY	700 kbps over 1M PHY, 1300 kbps over 2M PHY
Encryption	AES-128	AES-128	AES-128	AES-128	AES-128	AES-128
Pairing modes	Just works, numeric comparison, passkey entry, Out-Of-Band	Just works, numeric comparison, passkey entry, Out-Of-Band	Just works, numeric comparison, passkey entry, Out-Of-Band	Just works, numeric comparison, passkey entry, Out-Of-Band	Just works, numeric comparison, passkey entry, Out-Of-Band	Just works, numeric comparison, passkey entry, Out-Of-Band
Number of simultaneous bondings	Up to 13 with PS Store, up to 32 with NVM3	Up to 13 with PS Store, up to 32 with NVM3	Up to 13 with PS Store, up to 32 with NVM3	Up to 32	Up to 32	Up to 32
Link Layer packet size	Up to 251 B	Up to 251 B	Up to 251 B	Up to 251 B	Up to 251 B	Up to 251 B
ATT protocol packet size	Up to 250 B	Up to 250 B	Up to 250 B	Up to 250 B	Up to 250 B	Up to 250 B
Supported Bluetooth profiles and services	All GATT based profiles and services are supported	All GATT based profiles and services are supported	All GATT based profiles and services are supported	All GATT based profiles and services are supported	All GATT based profiles and services are supported	All GATT based profiles and services are supported
Apple HomeKit	x	Apple HomeKit R15-compliant implementation	Apple HomeKit R15-compliant implementation	Apple HomeKit R15-compliant implementation	Apple HomeKit R15-compliant implementation	Apple HomeKit R15-compliant implementation

Feature	EFR32[B|M]G1	EFR32[B|M]G12	EFR32[B|M]G13	EFR32[B|M]G21*	EFR32[B|M]G22	EFR32[B|M]G24
Host (NCP/RCP) interfaces	4-wire UART with RTS/CTS control or 2-wire UART without RTS/CTS, GPIOs for sleep and wake-up management	4-wire UART with RTS/CTS control or 2-wire UART without RTS/CTS, GPIOs for sleep and wake-up management	4-wire UART with RTS/CTS control or 2-wire UART without RTS/CTS, GPIOs for sleep and wake-up management	4-wire UART with RTS/CTS control or 2-wire UART without RTS/CTS, GPIOs for sleep and wake-up management	4-wire UART with RTS/CTS control or 2-wire UART without RTS/CTS, GPIOs for sleep and wake-up management	4-wire UART with RTS/CTS control or 2-wire UART without RTS/CTS, GPIOs for sleep and wake-up management
Wi-Fi Coexistence	Using Packet Trace Arbitration (PTA)	Using Packet Trace Arbitration (PTA)	Using Packet Trace Arbitration (PTA)	Using Packet Trace Arbitration (PTA)	Using Packet Trace Arbitration (PTA)	Using Packet Trace Arbitration (PTA)
Non-volatile memory	NVM3 or Persistent Store (PS)**	NVM3 or Persistent Store (PS)**	NVM3 or Persistent Store (PS)**	NVM3	NVM3	NVM3

* EFR32MR21 has the same feature set as xG21, but works only in the RCP mode. The SoC and NCP modes are not supported.

** Example applications in the SDK that are generated for these platforms will use PS by default.

2.2 Bluetooth Qualification

All products using Bluetooth technology must go through the Bluetooth SIG's Qualification Process, even if the product does not have the Bluetooth logo or Bluetooth is not mentioned in the packaging and the documentation. In practice this means that, before you can sell a Bluetooth-enabled product to the market, the product must be qualified as an End Product through the Bluetooth SIG. The qualification listing has a Declaration Fee. There are online resources to learn more about the Bluetooth Qualification Process as well as tutorials on the Launch Studio, which is the online tool used to complete the Bluetooth Qualification Process. If you need assistance to qualify your device consider reaching out to your nearest Bluetooth Qualification Consultant.

When qualifying your end-product based on the Silicon Labs Bluetooth stack, you will integrate the pre-qualified components listed in the table below, depending on which SDK version was used to build your application.

Bluetooth SDK version	Component	QDID
v2.13.x up to v3.1.x	Link Layer (Bluetooth 5.2)	Launch Studio Listing Details: 147971
“	Host stack (Bluetooth 5.2)	Launch Studio Listing Details: 146950
V3.2.x and above	Link Layer (Bluetooth 5.3)	Launch Studio Listing Details: 178212
“	Host stack (Bluetooth 5.3)	Launch Studio Listing Details: 175341

Note: According to Bluetooth SIG Qualification Program Reference Document (PRD), the Assessment Date of the tested Component must be less than three years old at the time it is being imported into a Launch Studio project for a new End Product Listing (EPLs). After the expiration of a Component QDID (Qualified Design ID), a newer SDK version than the one used for the outdated QDID should be used in order to qualify your product. There can be also newer QDIDs than the ones listed in the table above if there are newer Component versions. You can browse our valid Qualified Components and their Assessment Date by inserting Silicon Laboratories in the search bar of Launch Studio. Contact the technical support in case there is a need to use an older SDK version.

The above software-based pre-qualified components are two out of the three components to integrate when proceeding with the "Qualification Process with Required Testing". Despite the “Required Testing", customers do not need to do any additional testing, given that the test reports are embedded in the pre-qualified components for the SIG to review.

In addition to these two software components, you must also have integrated a qualified RF-PHY component in your end-product listing. If you are designing with one of the Silicon Labs Bluetooth modules, then refer to the module datasheet for the appropriate component QDID to use. If you are designing with an SoC then you may need to obtain your own RF-PHY qualification with the Bluetooth SIG, depending on your hardware design. In the latter case, consult your nearest Bluetooth Qualification Consultant, or Silicon Labs through the support portal, to understand if an existing Silicon Labs RF-PHY pre-qualification could be used.

Silicon Labs does not provide prequalified profiles. Customers must provide these with their end applications that implement the functionality as per the SIG profile specification.

2.3 The Bluetooth Stack APIs

This section briefly describes the different software APIs available for the developer when developing a Bluetooth application either in SoC or NCP mode. In RCP mode the standard HCI is used, which is defined in the Bluetooth Core Specification and therefore is not discussed here.

2.3.1 The BGAPI Bluetooth API

The BGAPI is the Bluetooth API provided by the Silicon Labs Bluetooth stack. It provides access to all the Bluetooth functionality implemented by the Bluetooth stack, such as: the Generic Access Profile (GAP), connection manager, the security manager (SM), and GATT client and server.

In addition to the Bluetooth APIs, the BGAPI also provides access to a few other functions like the Direct Test Mode (DTM) API for RF testing purposes, the NVM (Non-Volatile Memory) API for reading and writing settings to and from the devices flash memory, the DFU (Device Firmware Update) API for field firmware updates, and the System API for various system level functions.

2.3.2 CMSIS and emlib

The Cortex Microcontroller Software Interface Standard (CMSIS) is a common coding standard for all ARM Cortex devices. The CMSIS library provided by Silicon Labs contains header files, defines (for peripherals, registers and bitfields), and startup files for all devices. In addition, CMSIS includes functions that are common to all Cortex devices, like interrupt handling, intrinsic functions, etc. Although it is possible to write to registers using hard-coded address and data values, it is recommended to use the defines to ensure portability and readability of the code.

To simplify programming Wireless Geckos, Silicon Labs developed and maintains a complete C function library called emlib that provides efficient, clear, and robust access to and control of all peripherals and core functions in the device. This library resides within the em_xxx.c (for example, em_dac.c) and em_xxx.h files in the SDK.

The emlib documentation is available on https://docs.silabs.com.

2.3.3 The BGAPI Serial Protocol and BGLIB Host API

When configured in NCP (network co-processor) mode, the Bluetooth stack also implements the BGAPI serial protocol. This allows the Bluetooth stack to be controlled over a serial interface such as UART from a separate host like an EFM32 microcontroller. The BGAPI serial protocol provides exactly the same Bluetooth APIs over UART as the BGAPI API when used in a standalone mode. Additionally, an extra command and an event are reserved for user messaging in case the interface should be extended with custom commands.

The BGAPI serial protocol is a lightweight, binary protocol that carries the BGAPI commands from the host to the Bluetooth stack and responses and events from the Bluetooth stack back to the host.

The Bluetooth SDK delivers a ready-made BGAPI serial protocol parser implementation, called BGLIB. It implements the serial protocol parser and C language function and events for all the APIs provided by the Bluetooth stack. The host code developed on top of BGLIB can be written to be identical to the code for the Wireless Gecko, which allows easy porting of the application code from the Wireless Gecko to a separate host or vice versa.

A Python based BGAPI serial protocol parser is also available here: https://pypi.org/project/pybgapi/

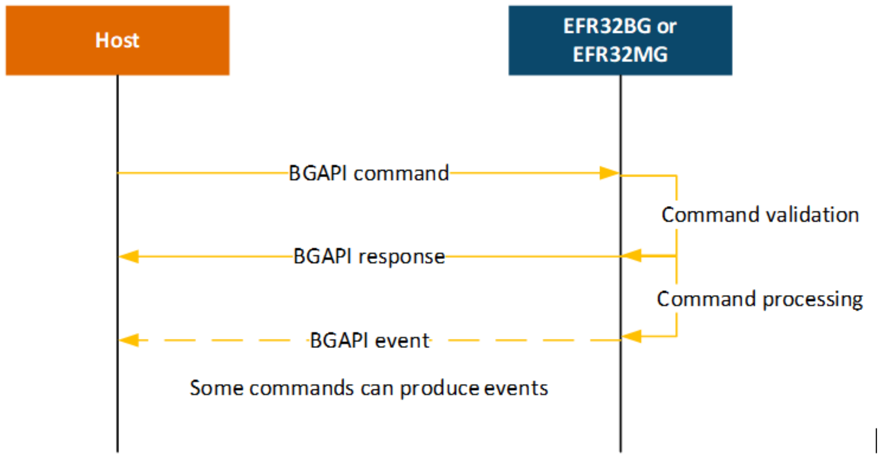


Figure 2.1. BGAPI Serial Protocol Message Exchange

The BGAPI serial protocol packet structure is described in the following table.

Table 2-1. BGAPI Packet Structure

Byte	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4-255
Explanation	Message type	Minimum payload length	Message class	Message ID	Payload
Values	0x20: command	0x00 - 0xFF	0x00 - 0xFF	0x00 - 0xFF	Specific to command, response, or event
"	0x20: response	0x00 - 0xFF	0x00 - 0xFF	0x00 - 0xFF	Specific to command, response, or event
"	0xA0: event	0x00 - 0xFF	0x00 - 0xFF	0x00 - 0xFF	Specific to command, response, or event

2.3.4 GATT Configuration

Bluetooth applications usually need a GATT database. The structure of the GATT database can be defined in the Bluetooth application. The Silicon Labs' Bluetooth SDK provides two ways to define the GATT database:

- A static GATT database can be defined in compile time with the appropriate tools provided by the Bluetooth SDK. In this case the database structure is stored in the ROM, which means faster start-up time and lower memory usage.
- A dynamic GATT database can be defined in runtime with the appropriate BGAPI commands. In this case the database structure is stored in the RAM, which makes it more flexible. This is recommended in the NCP use case to avoid re-building the target code that runs on the Wireless Gecko.

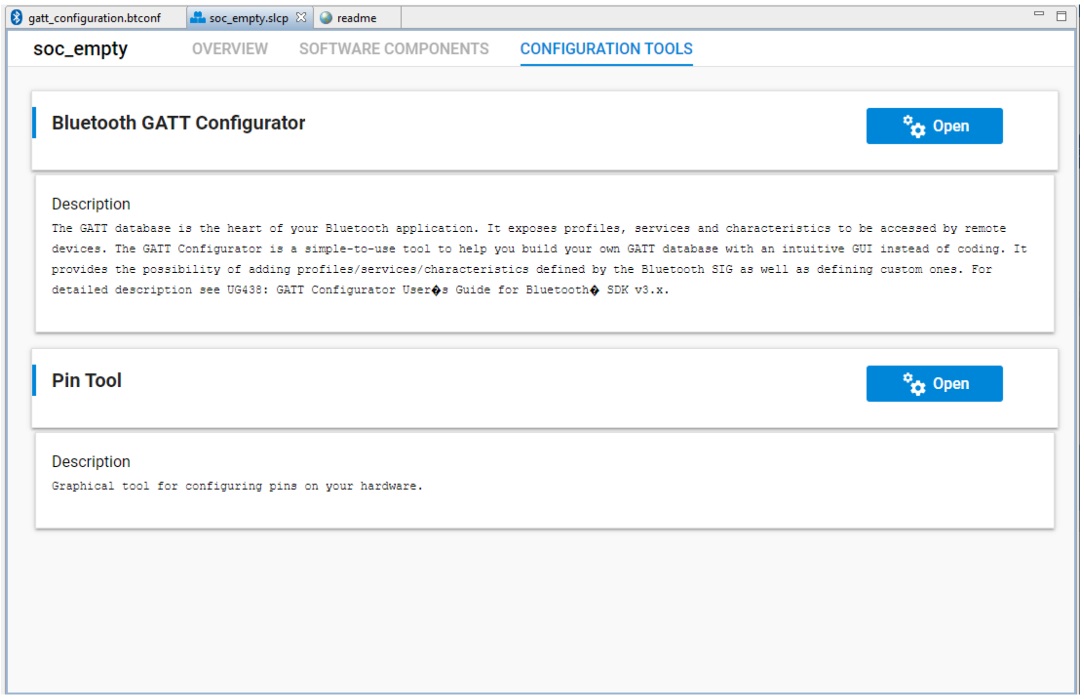
The Bluetooth Profile Toolkit GATT Builder

The Bluetooth Profile Toolkit is a simple XML-based API and description language used to describe (static) GATT-based services and characteristics easily without the need to write them in code. The XML files can be easily written by hand based on the information contained in *UG118: Blue Gecko Bluetooth® Profile Toolkit Developer Guide*. Use the Profile Toolkit GATT Builder if you are developing outside of Simplicity Studio, and follow the instructions of *UG118: Blue Gecko Bluetooth® Profile Toolkit Developer Guide* to convert your GATT database into C code.

The GATT Configurator

Simplicity Studio includes the GATT Configurator, a tool that allows building the (static) GATT database in a visual way, without hand editing the XML file. It also automatically converts the database structure into C code upon saving. See section 6.1, The GATT Configurator for summary information, and *UG438: GATT Configurator User's Guide for Bluetooth SDK v3.x* for details. Open the GATT

Configurator in Simplicity Studio through the Project Configurator, Configuration Tools tab. Click **Open** and the GATT Configurator tool opens the file *gatt_configuration.btconf* in a new tab.



Description
The GATT database is the heart of your Bluetooth application. It exposes profiles, services and characteristics to be accessed by remote devices. The GATT Configurator is a simple-to-use tool to help you build your own GATT database with an intuitive GUI instead of coding. It provides the possibility of adding profiles/services/characteristics defined by the Bluetooth SIG as well as defining custom ones. For detailed description see UG438: GATT Configurator User�s Guide for Bluetooth� SDK v3.x.

Description
Graphical tool for configuring pins on your hardware.

Figure 2.2. Opening the Bluetooth GATT Configurator

gatt_configuration.btconf gives the trunk of the GATT database. It is located inside the *config* > *btconf* directory of your project. This file can be edited using the GATT Configurator.

The contents of the additional xml files in the *config* > *btconf* folder will appear as *Contributed Items* in the GATT Configurator UI. See for example the in_place_ota_dfu.xml file provided by the OTA DFU software component. If these files are edited through GATT Configurator, they become a part of Custom BLE GATT and will be removed from the Contributed Items. Also, the contributed items (services and its characteristics) will move from the xml file to gatt_configuration.btconf file.

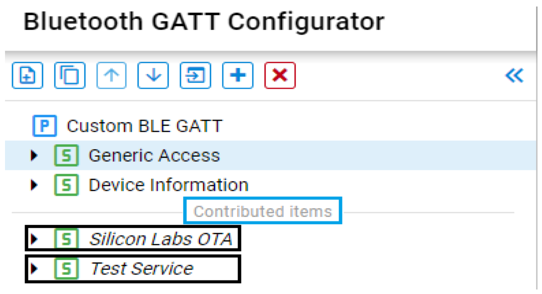
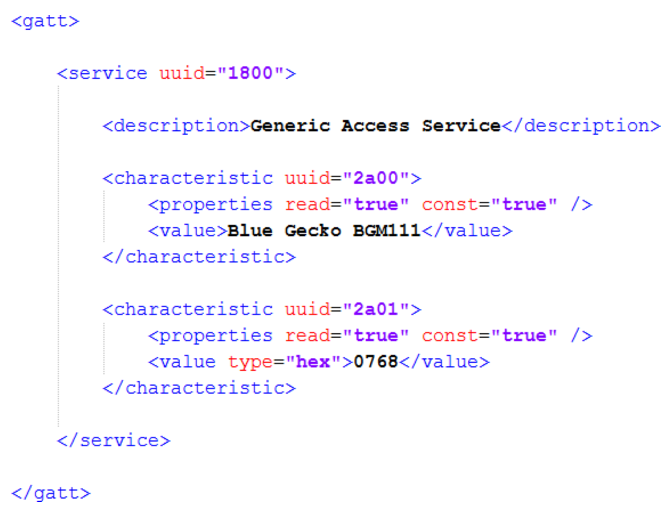


Figure 2.3. Contributed Items in the GATT Configurator UI

Upon saving *gatt_configuration.btconf,* the GATT database developed with the GATT Configurator is converted to a .c file and an .h file and included in the application project as a pre-build step when the firmware is compiled. Then the GATT can be accessed with the Bluetooth stack GATT APIs or by a remote Bluetooth device.



```
<gatt>

    <service uuid="1800">

        <description>Generic Access Service</description>

        <characteristic uuid="2a00">
            <properties read="true" const="true" />
            <value>Blue Gecko BGM111</value>
        </characteristic>

        <characteristic uuid="2a01">
            <properties read="true" const="true" />
            <value type="hex">0768</value>
        </characteristic>

    </service>

</gatt>
```

Figure 2.4. Example of a Generic Access Service

Building a Dynamic GATT database

The GATT database can also be built dynamically from the application using the GATT database API class of the Bluetooth API if the Dynamic GATT Database software component is installed in your project, or if this API class is explicitly initialized. For more information see the Bluetooth API reference manual at https://docs.silabs.com/bluetooth/latest/ and the corresponding section of *UG438: GATT Configurator User's Guide for Bluetooth SDK v3.x.* In NCP mode it is also possible to take a static GATT database code on the host side and turn it into dynamic API calls to transmit the database structure over UART. For more information see *AN1259: Using the v3.x Silicon Labs Bluetooth® Stack in Network Co-Processor Mode.*

2.4 About the Bluetooth SDK

The Bluetooth SDK is a full software development kit that enables you to develop applications on top of the Bluetooth stack using C programming language. The SDK also supports making standalone applications, where the Bluetooth stack and the application both run in the Wireless Gecko, or the network co-processor (NCP) architecture, where the application runs on an external host and the Bluetooth stack runs in the Wireless Gecko. SDK contents and folder structure are described in the following sections.

2.4.1 Libraries

The following libraries are delivered with the Bluetooth SDK and must be included in C application projects.

Library	Explanation	Mandatory
libbluetooth.a	Bluetooth stack library	Yes
librail_efr32xg1_gcc_release.a	RAIL library for GCC	Yes for GCC projects on EFR32xG1 platform
librail_efr32xg12_gcc_release.a	RAIL library for GCC	Yes for GCC projects on EFR32xG12 platform
librail_efr32xg13_gcc_release.a	RAIL library for GCC	Yes for GCC projects on EFR32xG13 platform
librail_efr32xg14_gcc_release.a	RAIL library for GCC	Yes for GCC projects on EFR32xG14 platform
librail_efr32xg21_gcc_release.a	RAIL library for GCC	Yes for GCC projects on EFR32xG21 platform
librail_efr32xg22_gcc_release.a	RAIL library for GCC	Yes for GCC projects on EFR32xG22 platform
librail_efr32xg24_gcc_release.a	RAIL library for GCC	Yes for GCC projects on EFR32xG24 platform
librail_efr32xg1_iar_release.a	RAIL library for IAR	Yes for IAR projects on EFR32xG1 platform

Library	Explanation	Mandatory
librail_efr32xg12_iar_release.a	RAIL library for IAR	Yes for IAR projects on EFR32xG12 platform
librail_efr32xg13_iar_release.a	RAIL library for IAR	Yes for IAR projects on EFR32xG13 platform
librail_efr32xg14_iar_release.a	RAIL library for IAR	Yes for IAR projects on EFR32xG14 platform
librail_efr32xg21_iar_release.a	RAIL library for IAR	Yes for IAR projects on EFR32xG21 platform
librail_efr32xg22_iar_release.a	RAIL library for IAR	Yes for IAR projects on EFR32xG22 platform
librail_efr32xg24_iar_release.a	RAIL library for IAR	Yes for IAR projects on EFR32xG24 platform
libpsstore.a	PSStore library	Yes, on series 1
binapploader.o	Apploader for OTA updates	No
libcoex.a	Wi-Fi and Bluetooth coexistence	No
libnvm3_CM33_gcc.a / libnvm3_CM33_iar.a	-	Yes, on series 2

2.4.2 Include Files

The following files are delivered with the Bluetooth SDK and must be included in C application projects.

Library	Explanation	When needed
bg_gattdb_def.h	Bluetooth GATT database structure definition.	Included automatically.
sl_bt_version.h	Bluetooth stack version in plain text. The boot event reports the same version values as this file has.	For convenient access to Bluetooth SDK version Information. Not mandatory for application development.
sl_bt_ll_config.h	Bluetooth Link Layer configuration data type definitions. Included by sl_bt_stack_config.h.	Included automatically.
sl_bt_stack_config.h	Bluetooth stack configuration data type definitions. Included by sl_bluetooth_config.h.	Included automatically.
sl_bluetooth_config.h	Bluetooth configuration.	Included automatically if application is generated with the Project Configurator.
sl_bt_types.h	Bluetooth API data type definitions.	Included automatically.
sl_bt_stack_init.h	Bluetooth feature and API initialization functions on SoC.	Included automatically if application is generated with the Project Configurator.
sl_bt_api.h	Bluetooth API declarations with comprehensive documentation. This is the single file for Bluetooth API in SoC or NCP mode.	Included automatically if application is generated with the Project Configurator.
sli_bt_api.h	Bluetooth API library in plain source code for NCP host applications.	Included automatically if application is generated with the Project Configurator.
sl_bt_ncp_host_api.c	Bluetooth API library in plain source code for NCP host applications.	Included automatically if application is generated with the Project Configurator.
sl_bt_ncp_host.h	An adaptation layer between host application and Bluetooth API serial protocol.	Included automatically if application is generated with the Project Configurator.
sl_bt_ncp_host.c	An adaptation layer between host application and Bluetooth API serial protocol.	Included automatically if application is generated with the Project Configurator.
sl_bt_rtos_adaptation.h	An adaptation layer for running Bluetooth in Micrium OS on SoC.	Included automatically if application is generated with the Project Configurator.
sl_bt_rtos_adaptation.c	An adaptation layer for running Bluetooth in Micrium OS on SoC.	Included automatically if application is generated with the Project Configurator.

2.4.3 Platform Components

The following components are delivered with the Bluetooth SDK. The platform components are under the platform folder.

Folder	Explanation
bootloader	Gecko Bootloader source code and project files.
CMSIS	Silicon Laboratories CMSIS-CORE device headers. (1)
common	Silicon Labs status codes
Device	EFR32BG and EFR32MG device files. (2)
emdrv	A set of function-specific high-performance drivers for EFR32 on-chip peripherals. Drivers are typically DMA based and are using all available low-energy features. For most drivers, the API offers both synchronous and asynchronous functions. (3)
emlib	A low-level peripheral support library that provides a unified API for all EFM32, EZR32 and EFR32 MCUs and SoCs from Silicon Laboratories. (4)
Halconfig	Peripheral configuration
Hwconf_data	Gather chip-specific hardware configuration
micrium_os	Micrium OS
middleware	Display driver for WSTK development kits (5)
radio	Silicon Labs RAIL (Radio Abstraction Interface Layer) library
service	Sleeptimer driver and configuration file. Used by the Bluetooth LE stack.

(1) https://docs.silabs.com/mcu/latest/bgm21/group-Parts

(2) https://docs.silabs.com/mcu/latest/bgm21/group-Parts

(3) https://docs.silabs.com/mcu/latest/bgm21/group-emdrv

(4) https://docs.silabs.com/mcu/latest/bgm21/group-emlib

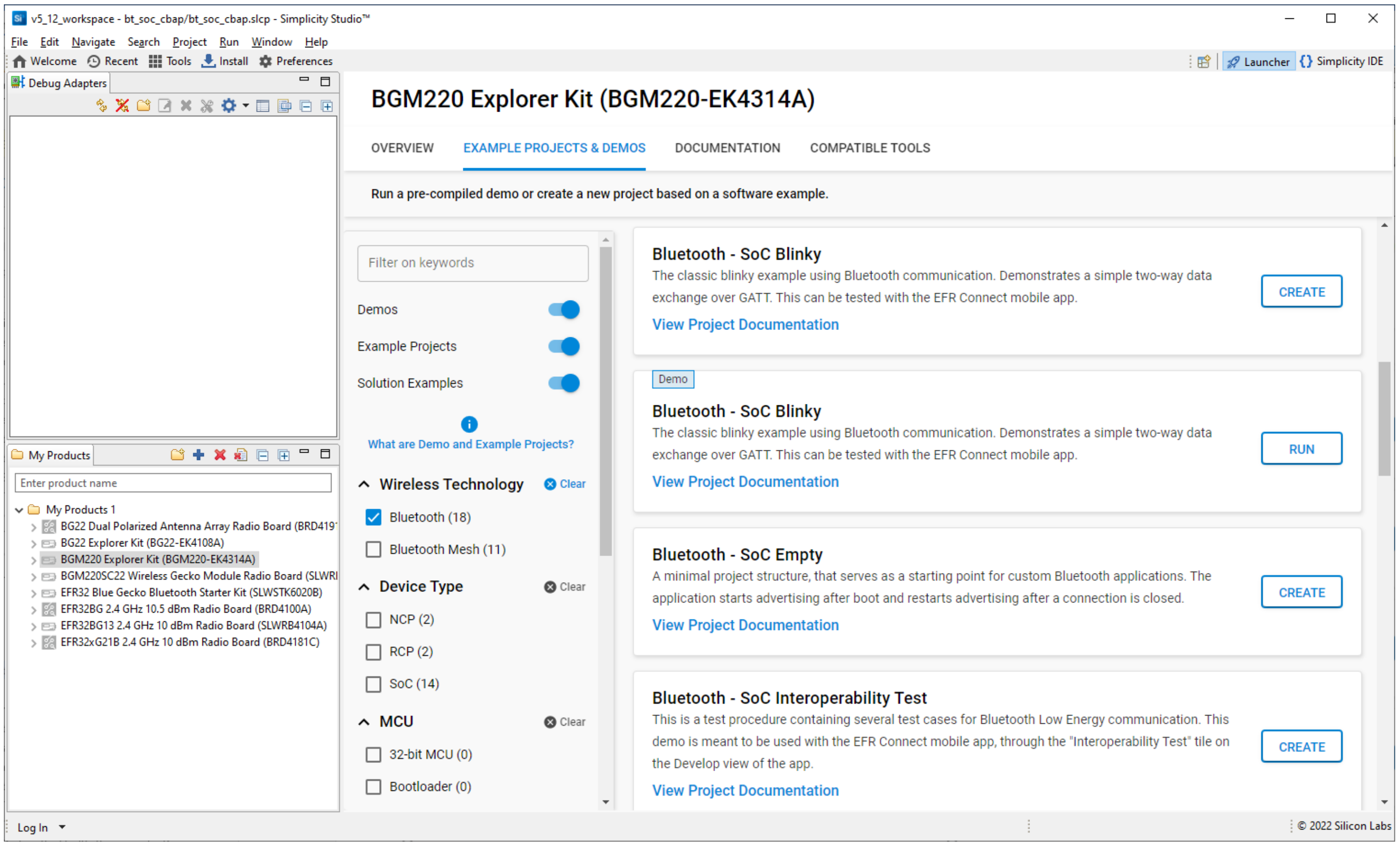
(5) https://docs.silabs.com/mcu/latest/bgm21/group-emlib

3 About Demos and Examples

3.1 Bluetooth SDK Examples

Because starting application development from scratch is difficult, the Bluetooth SDK comes with a number of built-in demos and examples covering the most frequent use cases, as shown in the following figure. Demos are pre-built application images that you can run immediately. Software examples (example projects) can be modified before building the application image. Demos with the same name as Software examples are built from their respective example. Click **View Project Documentation** to see additional information about some examples. This is also displayed on a **readme** tab when you create a project based on the example.

Use the **Demos** and **Example Projects** switches to filter on only examples or only demos. Demos are also noted by the blue Demo tag in the upper left of the card. The Solution Examples switch is provided for future use.



v5_12_workspace - bt_soc_cbap/bt_soc_cbap.slcp - Simplicity Studio™

File Edit Navigate Search Project Run Window Help

Welcome Recent Tools Install Preferences Launcher Simplicity IDE

Debug Adapters

My Products

Enter product name

My Products 1
- BG22 Dual Polarized Antenna Array Radio Board (BRD419
- BG22 Explorer Kit (BG22-EK4108A)
- BGM220 Explorer Kit (BGM220-EK4314A)
- BGM220SC22 Wireless Gecko Module Radio Board (SLWRI
- EFR32 Blue Gecko Bluetooth Starter Kit (SLWSTK6020B)
- EFR32BG 2.4 GHz 10.5 dBm Radio Board (BRD4100A)
- EFR32BG13 2.4 GHz 10 dBm Radio Board (SLWRB4104A)
- EFR32xG21B 2.4 GHz 10 dBm Radio Board (BRD4181C)

BGM220 Explorer Kit (BGM220-EK4314A)

OVERVIEW EXAMPLE PROJECTS & DEMOS DOCUMENTATION COMPATIBLE TOOLS

Run a pre-compiled demo or create a new project based on a software example.

Filter on keywords

Demos
Example Projects
Solution Examples

What are Demo and Example Projects?

Wireless Technology Clear
- [✓] Bluetooth (18)
- [ ] Bluetooth Mesh (11)

Device Type Clear
- [ ] NCP (2)
- [ ] RCP (2)
- [ ] SoC (14)

MCU Clear
- [ ] 32-bit MCU (0)
- [ ] Bootloader (0)

Bluetooth - SoC Blinky
The classic blinky example using Bluetooth communication. Demonstrates a simple two-way data exchange over GATT. This can be tested with the EFR Connect mobile app.
View Project Documentation
CREATE

Demo
Bluetooth - SoC Blinky
The classic blinky example using Bluetooth communication. Demonstrates a simple two-way data exchange over GATT. This can be tested with the EFR Connect mobile app.
View Project Documentation
RUN

Bluetooth - SoC Empty
A minimal project structure, that serves as a starting point for custom Bluetooth applications. The application starts advertising after boot and restarts advertising after a connection is closed.
View Project Documentation
CREATE

Bluetooth - SoC Interoperability Test
This is a test procedure containing several test cases for Bluetooth Low Energy communication. This demo is meant to be used with the EFR Connect mobile app, through the "Interoperability Test" tile on the Develop view of the app.
View Project Documentation
CREATE

Log In

© 2022 Silicon Labs

Note: The demos and examples you see are determined by the part selected. If you are using a custom solution with more than one part, click on the part you are working with to see only the items applicable to that part.

To download and run a demo on your device, click **RUN** on the desired demo card. The demo automatically downloads to the selected device. See section 4 Getting Started with Bluetooth Demo Software for more information about testing the demos.

Software examples include a configurable GATT database and configurable components. See section 5 Starting Application Development.

3.1.1 Demo/Example Descriptions

The following examples are provided as part of the Bluetooth SDK. Examples with (*) in their names have a matching pre-built demo.

- **Silicon Labs Gecko Bootloader examples** (see *UG266: Silicon Labs Gecko Bootloader User's Guide for GSDK 3.2 and Lower* or *UG489: Silicon Labs Gecko Bootloader User's Guide for GSDK 4.0 and Higher*, and AN1086: Using the Gecko Bootloader with Silicon Labs Bluetooth Applications)
- **Bluetooth Examples**
  - **Bluetooth – RCP(*):** Radio Co-Processor (RCP) target application. Runs the Bluetooth Controller (i.e. the Link Layer only) and provides access to it using the standard HCI (Host-Controller Interface) over a UART connection.
  - **Bluetooth – RCP CPC(*):** Radio Co-Processor (RCP) target application. Runs the Bluetooth Controller (i.e. the Link Layer only) and provides access to it using the standard HCI (Host-Controller Interface) over CPC (Co-Processor Communication) protocol through a UART connection.
  - **Bluetooth – NCP(*)**: Network Co-Processor (NCP) target application. Runs the Bluetooth stack dynamically and provides access to it via the Bluetooth API (BGAPI) using a UART connection. NCP mode makes it possible to run your application on a host controller or PC.
  - **Bluetooth – NCP Host**: Reference implementation of an NCP (Network Co-Processor) host, which typically runs on a central MCU without radio. It can connect to an NCP target via UART to access the Bluetooth stack of the target and to control it using BGAPI.
  - **Bluetooth AoA – NCP Locator(*)**: Network Co-Processor (NCP) target application extended with CTE Receiver support. It enables Angle of Arrival (AoA) calculation. Use this application with Direction Finding host examples.
  - **Bluetooth AoA –Asset Tag(*)**: Demonstrates a CTE (Constant Tone Extension) transmitter that can be used as an asset tag in a Direction Finding setup estimating Angle of Arrival (AoA).
  - **Bluetooth – SoC Application OTA DFU**: A minimal project structure that serves as a starting point for custom Bluetooth applications providing Over-the-Air device firmware update in the user application runtime.
  - **Bluetooth – SoC Application OTA DFU FreeRTOS:** Demonstrates the integration of FreeRTOS into Bluetooth applications. RTOS is added to the *Bluetooth - SoC Application OTA DFU* sample application that realizes over-the-air device firmware updates in user application scope.
  - **Bluetooth – SoC Application OTA DFU MicriumOS:** Demonstrates the integration of MicriumOS into Bluetooth applications. RTOS is added to the *Bluetooth - SoC Application OTA DFU* sample application that realizes over-the-air device firmware updates in user application scope.
  - **Bluetooth – SoC Blinky(*)**: The classic blinky example using Bluetooth communication. Demonstrates a simple two-way data exchange over GATT. This can be tested with the EFR Connect mobile app.
  - **Bluetooth – SoC Certificate Based Authentication and Pairing**: Demonstrates Certificate-Based Authentication and Pairing over Bluetooth LE.
  - **Bluetooth – SoC CSR Generator**: Certificate-generating firmware example. Software generates the device EC key pair, the signing request for the device certificate, and other related data. The generated data can be read out by the Central Authority. See *Bluetooth – SoC Certificate Based Authentication and Pairing.*
  - **Bluetooth – SoC DTM**: This example implements the direct test mode (DTM) application for radio testing. DTM commands can be called via UART. See AN1267: Radio Frequency Physical Layer Evaluation in Bluetooth® SDK v3.x for more information.
  - **Bluetooth – SoC Empty:** A minimal project structure that serves as a starting point for custom Bluetooth applications. The application starts advertising after boot and restarts advertising after a connection is closed.
  - **Bluetooth – SoC Interoperability Test(*)**: A test procedure containing several test cases for Bluetooth Low Energy communication. This sample app (also provided as a demo) is meant to be used with the EFR Connect mobile app, through the "Interoperability Test" tile on the Develop view of the app.
  - **Bluetooth – SoC Thermometer(*)**: Implements a GATT Server with the Health Thermometer Profile, which enables a Client device to connect and get temperature data. Temperature is read from the Si7021 digital relative humidity and temperature sensor of the WSTK or of the Thunderboard.
  - **Bluetooth – SoC Thermometer Client**: Implements a GATT Client that discovers and connects with up to four Bluetooth LE devices advertising themselves as Thermometer Servers. It displays the discovery process and the temperature values received via UART.

    Note: Some radio boards will exhibit random pixels in the display when this example is running because they have a shared pin for sensor- and display-enabled signals.
  - **Bluetooth – SoC Thermometer FreeRTOS**: Demonstrates the integration of FreeRTOS into Bluetooth applications. RTOS is added to the Bluetooth - SoC Thermometer sample app.
  - **Bluetooth – SoC Thermometer Micrium OS**: Demonstrates the integration of Micrium RTOS into Bluetooth applications. RTOS is added to the Bluetooth - SoC Thermometer sample app.
  - **Bluetooth – SoC Throughput(*)**: Tests the throughput capabilities of the device and can be used to measure throughput between two EFR32 devices, as well as between a device and a smartphone using the EFR Connect mobile app, through the Throughput demo tile.

    - **Bluetooth – SoC Voice(*):** Voice over Bluetooth Low Energy sample application. It is supported by Thunderboard Sense 2 and Thunderboard EFR32BG22 boards and demonstrates how to send voice data over GATT, which is acquired from the on-board microphones.
    - **Bluetooth – SoC iBeacon(*)**: Sends non-connectable advertisements in iBeacon format. The iBeacon Service gives Bluetooth accessories a simple and convenient way to send iBeacons to smartphones. This example can be tested together with the EFR Connect mobile app.
    - **Bluetooth – SoC Thunderboard Sense 2(*),** and **Thunderboard EFR32BG22(*)**: Demonstrate the features of the Thunderboard Kit. These can be tested with the EFR Connect mobile app.
- **Dynamic Multiprotocol Examples** (see AN1134: Dynamic Multiprotocol Development with Bluetooth and Proprietary Protocols on RAIL for more information)
    - **Bluetooth RAIL DMP – SoC Empty FreeRTOS**: A minimal project structure, used as a starting point for custom Bluetooth + Proprietary DMP (Dynamic Multiprotocol) applications. It runs on top of FreeRTOS and multiprotocol RAIL.
    - **Bluetooth RAIL DMP – SoC Empty Micrium OS**: A minimal project structure, used as a starting point for custom Bluetooth + Proprietary DMP (Dynamic Multiprotocol) applications. It runs on top of Micrium OS and multiprotocol RAIL.
    - **Bluetooth RAIL DMP – SoC Empty Standard FreeRTOS**: A minimal project structure, used as a starting point for custom Bluetooth + Standard DMP (Dynamic Multiprotocol) applications. It runs on top of FreeRTOS and multiprotocol RAIL utilizing IEE802.15.4 standard protocol.
    - **Bluetooth RAIL DMP – SoC Empty Standard Micrium OS**: A minimal project structure, used as a starting point for custom Bluetooth + Standard DMP (Dynamic Multiprotocol) applications. It runs on top of Micrium OS and multiprotocol RAIL, utilizing IEE802.15.4 standard protocol.
    - **Bluetooth RAIL DMP – SoC Light RAIL FreeRTOS(*)**: A Dynamic Multiprotocol reference application demonstrating a light bulb that can be switched both via Bluetooth and via a Proprietary protocol. Can be tested with the EFR Connect mobile app and the **RAIL – SoC Switch** sample app.
    - **Bluetooth RAIL DMP – SoC Light RAIL Micrium OS**: A Dynamic Multiprotocol reference application demonstrating a light bulb that can be switched both via Bluetooth and via a Proprietary protocol. Can be tested with the EFR Connect mobile app and the **RAIL – SoC Switch** sample app.
    - **Bluetooth RAIL DMP – SoC Light Standard FreeRTOS(*)**: A Dynamic Multiprotocol reference application demonstrating a light bulb that can be switched both via Bluetooth and via a standard protocol. Can be tested with the EFR Connect mobile app and the **RAIL – SoC Switch** Standards sample app.
    - **Bluetooth RAIL DMP – SoC Light Standard Micrium OS(*)**: A Dynamic Multiprotocol reference application demonstrating a light bulb that can be switched both via Bluetooth and via a standard protocol. Can be tested with the EFR Connect mobile app and the **RAIL – SoC Switch Standards** sample app.
- **NCP Host Examples (**located in <GSDK install location>\app\bluetooth\example_host)
    - **bt_host_empty:** Minimal host-side project structure, used as a starting point for NCP host applications. Use it with the **Bluetooth – NCP** target application flashed to the radio board.
    - **bt_host_ota_dfu:** Demonstrates how to perform an OTA DFU on a Silicon Labs Bluetooth Device. It requires a WSTK with a radio board flashed with NCP firmware to be used as the GATT client that performs the OTA.
    - **bt_host_uart_dfu:** Demonstrates how to perform a UART DFU on a Silicon Labs Bluetooth Device running NCP firmware.
    - **bt_host_voice:** On a WSTK programmed with NCP firmware, it to connects to the **Bluetooth – SoC Voice** example, sets the correct configuration on it, receives audio via Bluetooth, and stores audio data into a file.
    - **bt_aoa_host_locator:** A locator host sample app that works together with a **Bluetooth AoA – NCP Locator** target app. It receives IQ samples from the target and estimates the Angle of Arrival (AoA). For more information see *QSG175: Application Development with Silicon Labs' RTL Library.*
    - **bt_host_positioning:** Connects to multiple **bt_aoa_host_locator** sample apps (via MQTT) and estimates a position from Angles of Arrival (AoA). For more information, see *QS175: Application Development with Silicon Labs' RTL Library.*
    - **bt_host_positioning_gui:** Connects to the **bt_host_positioning** sample app (via MQTT), reads out the position estimations and displays the tags and locators on a 3D GUI. This sample app is python based. For more information, see *QSG175: Application Development with Silicon Labs' RTL Library.*
    - **bt_host_throughput:** Tests the throughput capabilities of the device in NCP mode and can be used to measure throughput between two devices as well as between a device and a smartphone.
    - **bt_host_cpc_hci_bridge:** A background application to be run when HCI interface is exposed via CPC. This application retrieves the HCI commands/events from the CPC messages and forwards them toward the Bluetooth host running on the PC. Similarly, it forwards the HCI commands from the host toward the target over CPC.

3.2 Other Bluetooth Functionality Examples

The Bluetooth SDK provides examples that demonstrate basic Bluetooth features (such as a simple Bluetooth server, Bluetooth client, OTA DFU) and serve as a starting point for your (RCP, NCP, SoC, RTOS or DMP based) development. Often, however, developers need guidance on how to use additional Bluetooth features (such as extended advertisements, and so on). Also, it may be very useful to see full solution examples that cover a common use case and demonstrate how to integrate peripheral programming with Bluetooth.

To satisfy these needs, additional examples are provided in repositories on GitHub. The repositories have detailed readme files that instruct the developer on how to set up the project. To make the setup even easier, some repositories also provide .slcp files that make it possible to automatically generate the project with Simplicity Studio. Currently, the following Bluetooth-related repositories are available:

- **Bluetooth Stack Features**

  This repository demonstrates the different features of the Bluetooth stack. Each example focuses on one feature, and only a minimal application logic is added to demonstrate it. This repository provides .slcp files. The repository is available here: https://github.com/SiliconLabs/bluetooth_stack_features.

- **Bluetooth Applications**

  This repository provides full application examples, which demonstrate how to add Bluetooth communication to an application that provides solution for a real problem. It is worth checking this repository both to learn how a full Bluetooth application should look like, and also because you might find an example that is already very similar to the one you want to implement. The repository is available here: https://github.com/SiliconLabs/bluetooth_applications

- **Python-Based NCP Host Examples**

  Python-based NCP host examples can be accesssed at https://github.com/SiliconLabs/pybgapi-examples. These examples are meant to be used with PyBGAPI (https://pypi.org/project/pybgapi/), which enables implementing NCP host codes using Python.

Note: The examples in these repos are not maintained to the same degree as are the examples provided with the SDK. If you find an issue, report it on https://community.silabs.com/.

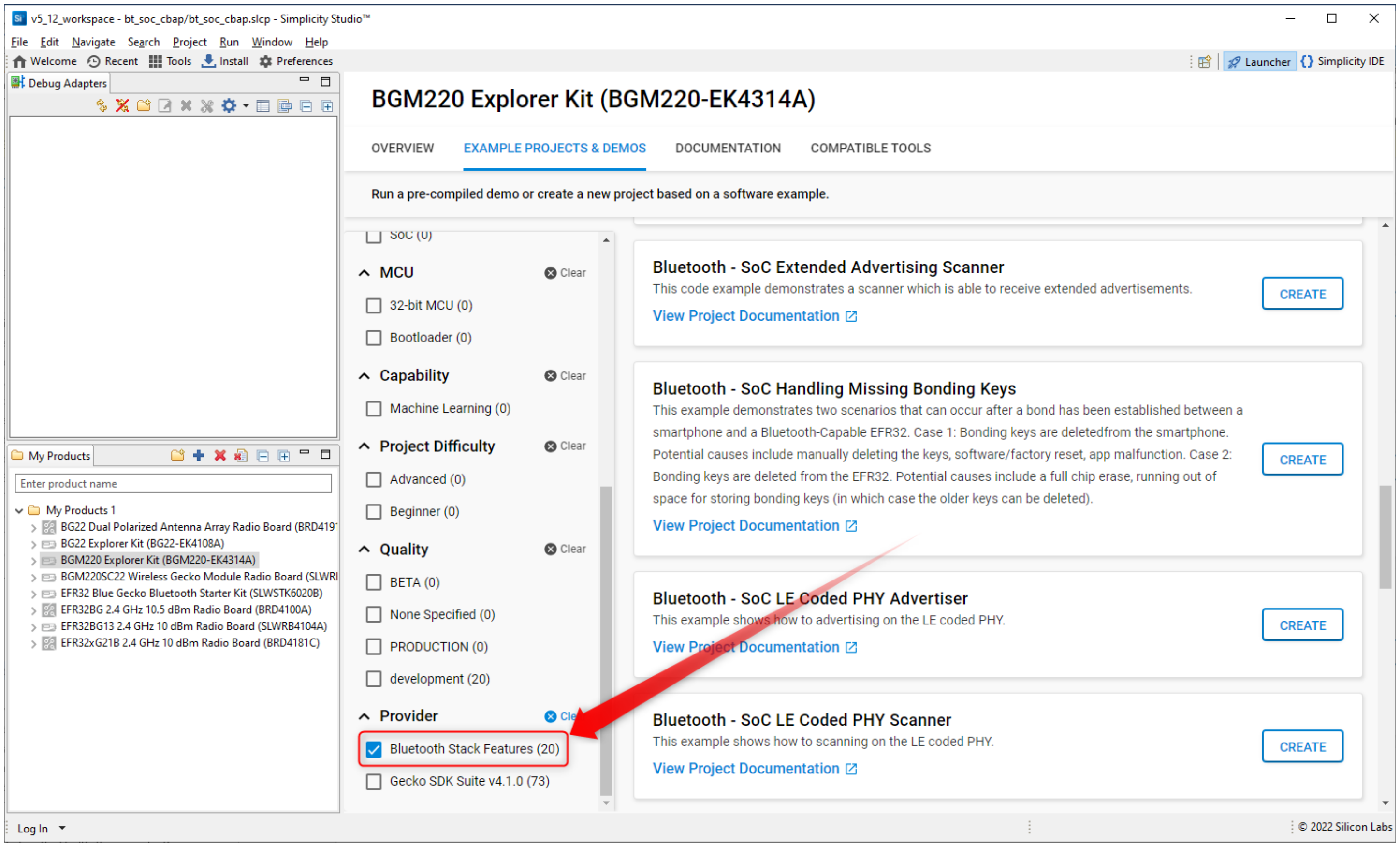
To generate GitHub example projects with Simplicity Studio (only applicable to repositories with slcp files):

1. Open Simplicity Studio and navigate to Window > Preferences > Simplicity Studio > External Repos.
2. Click **Add**.
3. In the URI field enter the repository's HTTPS clone address, for example https://github.com/SiliconLabs/bluetooth_stack_features.git

   Alternatively, clone the repository onto your hard drive, and provide the path to the .git folder in your local repository, for example C:\MyRepositories\bluetooth_stack_features\.git.
4. Enter an arbitrary name (such as 'Bluetooth Stack Features') and description for the repository, which will be displayed later.
5. Click **Next**. If you have entered the repository's clone address, Simplicity Studio will clone the repository for you.
6. Click **Finish,** then click **Apply and Close.**
7. If you are not on the Launcher perspective, open it from the Perspectives toolbar in the upper-right corner.
8. Select your board in the Debug Adapters view or in the My Products view.
9. On the General card, verify the Gecko SDK version and change if necessary.

   Note: Bluetooth stack feature examples are currently compatible with Gecko SDK v4.1 only!
10. Go to the Example Projects & Demos tab.
11. Now you should see your repository listed under the Provider filtering class. Select this filter.

    Note: The repository only shows up if it contains at least one example that is compatible with your device.
12. All the examples contained in the repository that are compatible with your device are displayed. Click **Create** on any of them to create a new example project. The example project installs all the software components necessary to demonstrate the given feature, and all the needed code is automatically copied into your project. Additional configuration might be needed, so read the readme file of the example carefully.


v5_12_workspace - bt_soc_cbap/bt_soc_cbap.slcp - Simplicity Studio™
File Edit Navigate Search Project Run Window Help
Welcome Recent Tools Install Preferences
Launcher Simplicity IDE
Debug Adapters
My Products
Enter product name
My Products 1
BG22 Dual Polarized Antenna Array Radio Board (BRD419
BG22 Explorer Kit (BG22-EK4108A)
BGM220 Explorer Kit (BGM220-EK4314A)
BGM220SC22 Wireless Gecko Module Radio Board (SLWRI
EFR32 Blue Gecko Bluetooth Starter Kit (SLWSTK6020B)
EFR32BG 2.4 GHz 10.5 dBm Radio Board (BRD4100A)
EFR32BG13 2.4 GHz 10 dBm Radio Board (SLWRB4104A)
EFR32xG21B 2.4 GHz 10 dBm Radio Board (BRD4181C)
BGM220 Explorer Kit (BGM220-EK4314A)
OVERVIEW
EXAMPLE PROJECTS & DEMOS
DOCUMENTATION
COMPATIBLE TOOLS
SOC (0)
MCU
Clear
32-bit MCU (0)
Bootloader (0)
Capability
Machine Learning (0)
Project Difficulty
Advanced (0)
Beginner (0)
Quality
BETA (0)
None Specified (0)
PRODUCTION (0)
development (20)
Provider
Bluetooth Stack Features (20)
Gecko SDK Suite v4.1.0 (73)
CREATE
View Project Documentation
Log In
© 2022 Silicon Labs

Run a pre-compiled demo or create a new project based on a software example.

Bluetooth - SoC Extended Advertising Scanner
This code example demonstrates a scanner which is able to receive extended advertisements.

Bluetooth - SoC Handling Missing Bonding Keys
This example demonstrates two scenarios that can occur after a bond has been established between a smartphone and a Bluetooth-Capable EFR32. Case 1: Bonding keys are deletedfrom the smartphone. Potential causes include manually deleting the keys, software/factory reset, app malfunction. Case 2: Bonding keys are deleted from the EFR32. Potential causes include a full chip erase, running out of space for storing bonding keys (in which case the older keys can be deleted).

Bluetooth - SoC LE Coded PHY Advertiser
This example shows how to advertising on the LE coded PHY.

Bluetooth - SoC LE Coded PHY Scanner
This example shows how to scanning on the LE coded PHY.

4 Getting Started with Bluetooth Demo Software

The Blue Gecko Bluetooth Wireless Starter Kit is meant to help you evaluate Silicon Labs' Bluetooth modules and get you started with your own software development. The kits come in different versions with different module radio boards. See https://www.silabs.com/products/development-tools/wireless/bluetooth/bluegecko-bluetooth-low-energy-module-wireless-starter-kit for details on the current configurations.

To get started with Bluetooth demo software, you should have downloaded Simplicity Studio 5 (SSv5) and the Bluetooth SDK v3.x as described in the Simplicity Studio 5 User's Guide, available online and through the SSv5 help menu. The Bluetooth SDK comes with some prebuilt demos that can be flashed to your EFR32 device and tested using a Smartphone. This section describes how to test three prebuilt demos on both Android and iOS devices:

- NCP Empty demo
- iBeacon demo
- Health Thermometer demo

4.1 Prepare the Mainboard

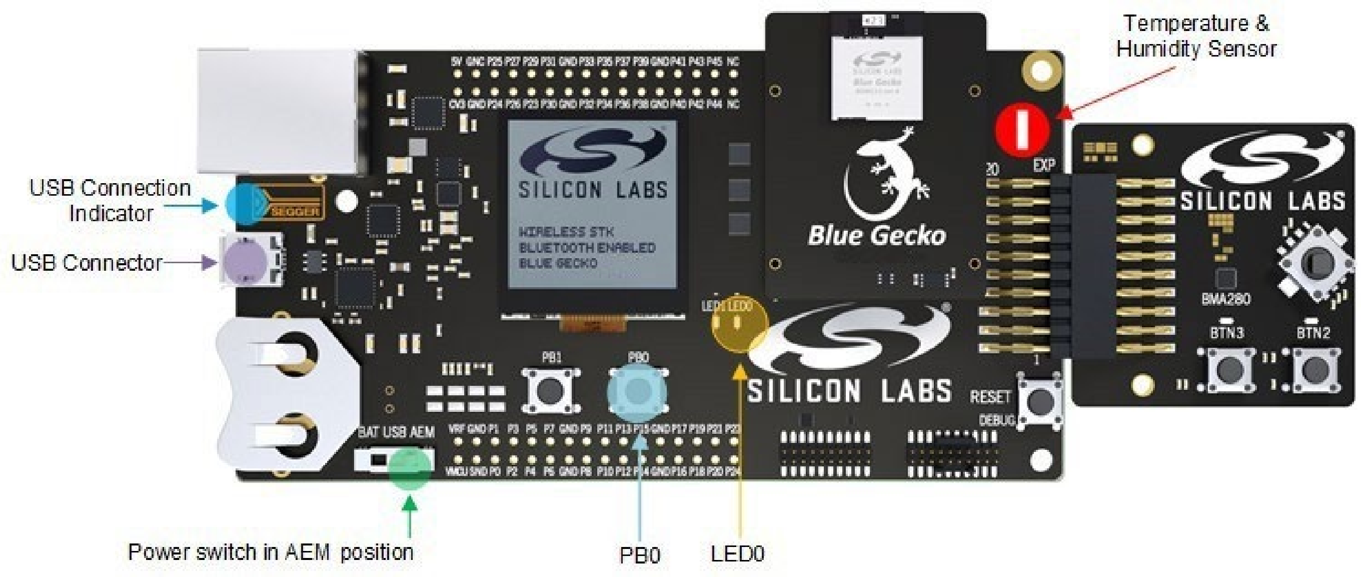
1. Connect a Bluetooth Module Radio Board to the mainboard as shown in the following figure.
2. Connect the mainboard to a PC using the **Mainboard USB** connector.
3. Turn the **Power switch** to **"AEM"** position.

Note: At this stage you might be prompted to install the drivers for the mainboard, but you can skip this for now.

4. Check that the blue **USB Connection Indicator** LED turns on or starts blinking.
5. Check that the mainboard LCD display turns on and displays a Silicon Labs logo.

Before starting to test the demo application note the following parts on the mainboard

- Temperature & Humidity Sensor
- PB0 button
- LED0



4.2 Flash the Demo

- With your device connected as described above, open SSv5.
- Select your device in the Debug Adapters view.
- On the Example Projects & Demos tab, click **RUN** on the demo of choice.

4.3 Test the Bluetooth Demos Using an Android or iOS Smartphone

4.3.1 Testing the NCP Demo

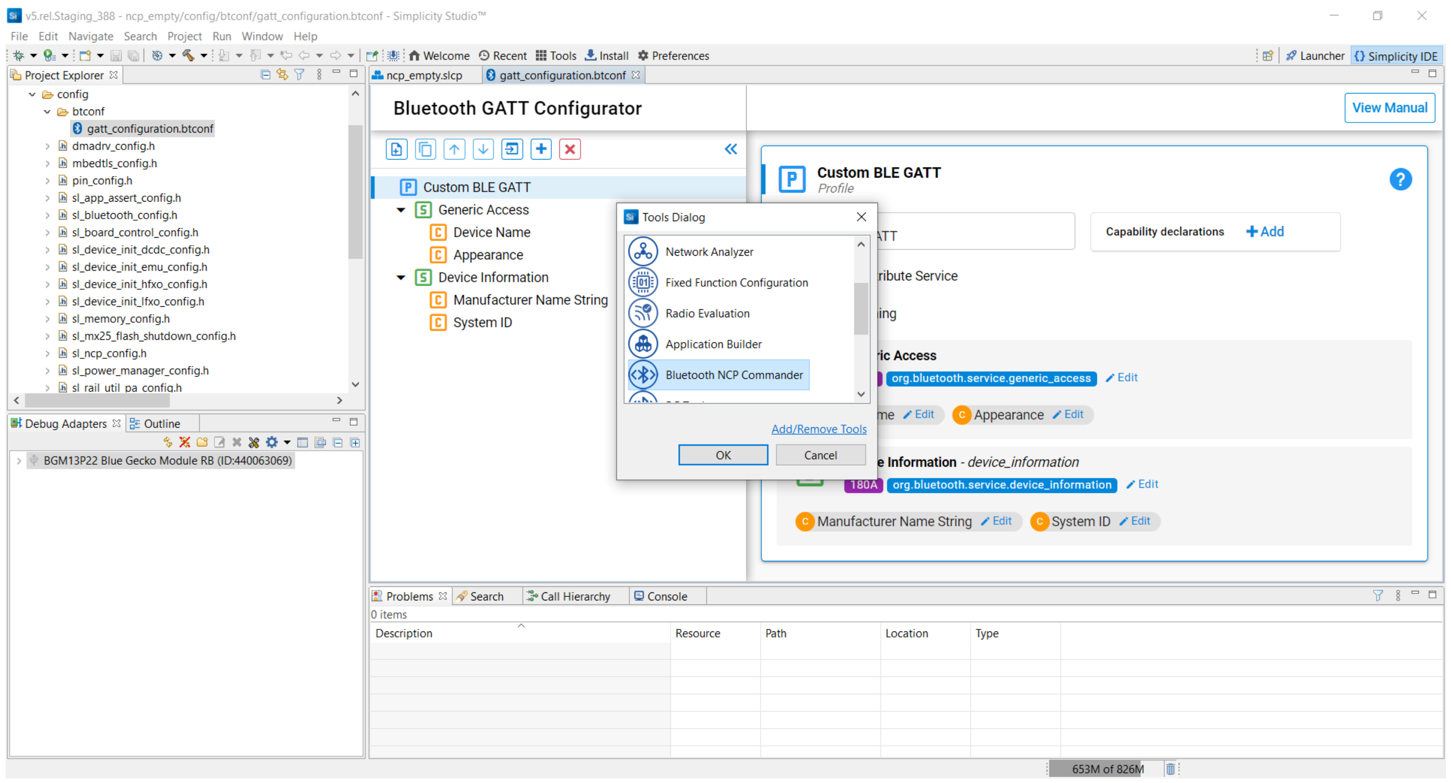
Load the NCP demo on the target:

1. Open Simplicity Studio with a mainboard and radio board connected and select the corresponding debug adapter.
2. On the OVERVIEW tab, under “General Information”, select the Gecko SDK Suite if it is not selected. On the Example Projects & Demos tab, find the **Bluetooth - NCP** demo and click RUN. This flashes the demo to your device, but it does not start advertising automatically.

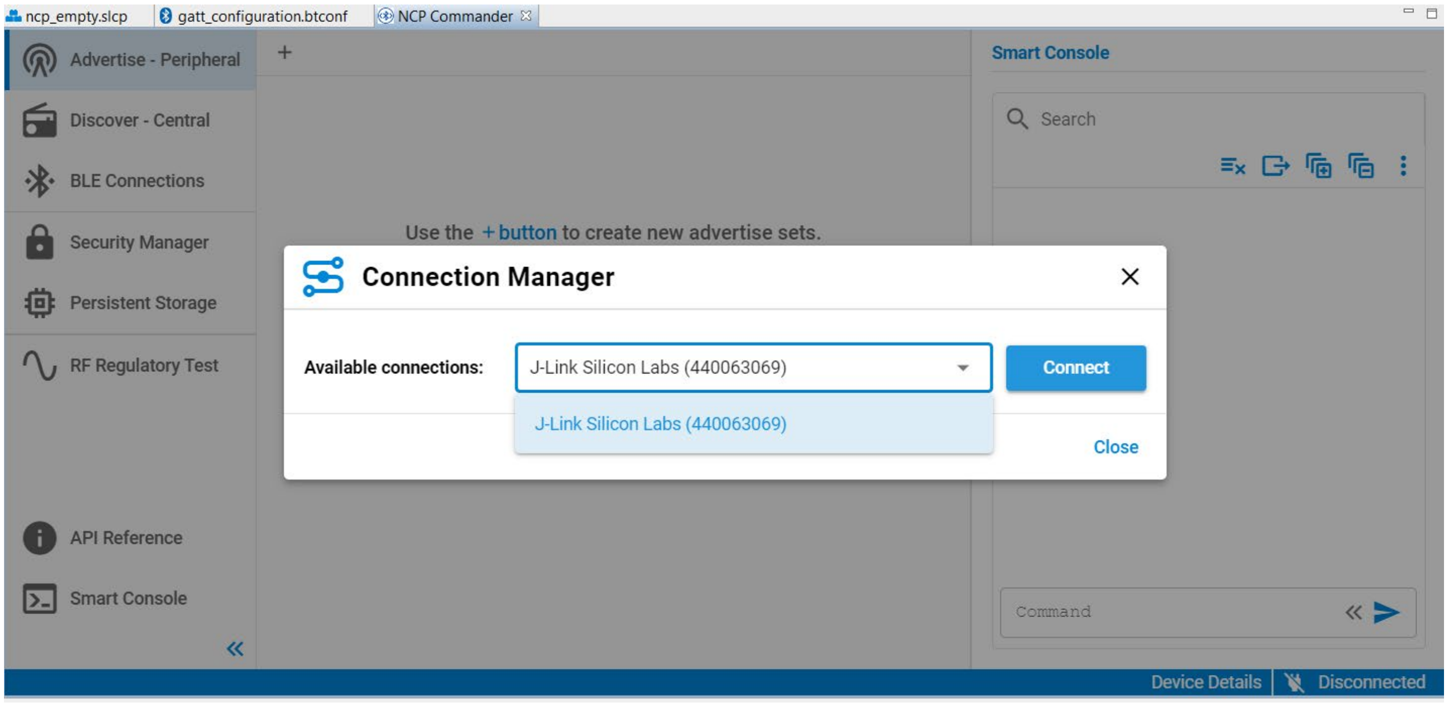
At this point, BGAPI 3.x commands can be sent to the kit. Starting with Bluetooth SDK version 3.1.0, Silicon Labs introduced a new tool, the Bluetooth NCP Commander, that can be used to send BGAPI 3.x commands to the kit, using UART. Connections, advertising and other standard Bluetooth Low Energy operation can be controlled via this tool.

Bluetooth NCP Commander

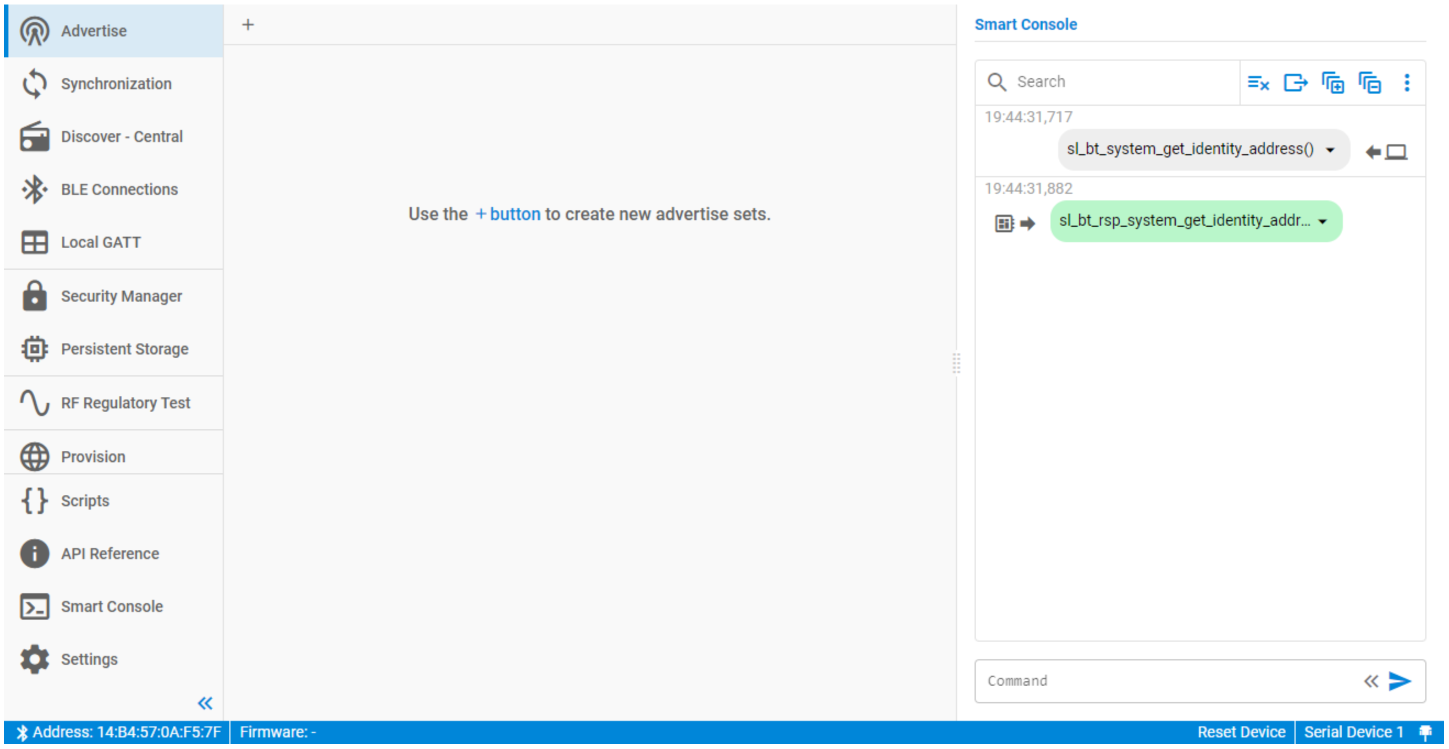
Bluetooth NCP Commander can be opened through the Project Configurator’s Configuration Tools tab, or from the Simplicity Studio Tools menu.



1. Launch Bluetooth NCP Commander and then establish the virtual UART connection to the kit using the JLink adapter:

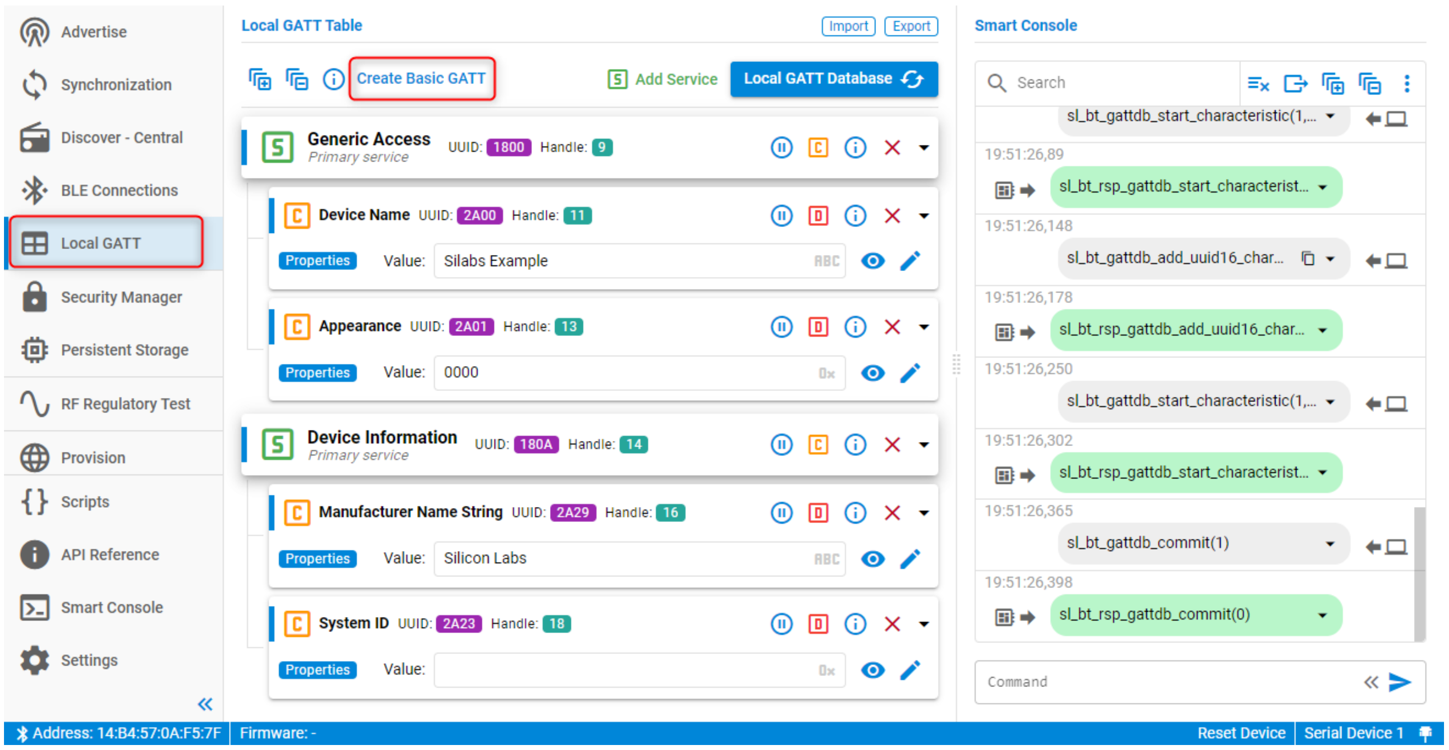


2. Click **Connect**. If everything works correctly you should see the result of the "sl_bt_system_get_identity_address" command displayed in green:



3. Unlike SoC examples, the NCP demo does not have a built-in GATT database and it expects the host to build the GATT database using the dynamic GATT database BGAPI commands. To create a basic GATT database, select the Local GATT menu, and click

Create Basic GATT. This triggers a series of BGAPI commands that will build a basic database. You can modify this GATT database as you want. You can also change the device name here by changing the value of the Device Name characteristic.



Generic Access — Primary service — UUID: 1800 Handle: 9

Device Name UUID: 2A00 Handle: 11 — Properties — Value: Silabs Example

Appearance UUID: 2A01 Handle: 13 — Properties — Value: 0000

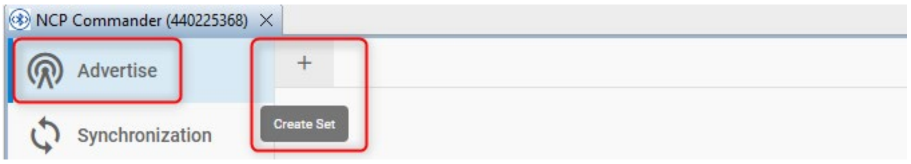
Device Information — Primary service — UUID: 180A Handle: 14

Manufacturer Name String UUID: 2A29 Handle: 16 — Properties — Value: Silicon Labs

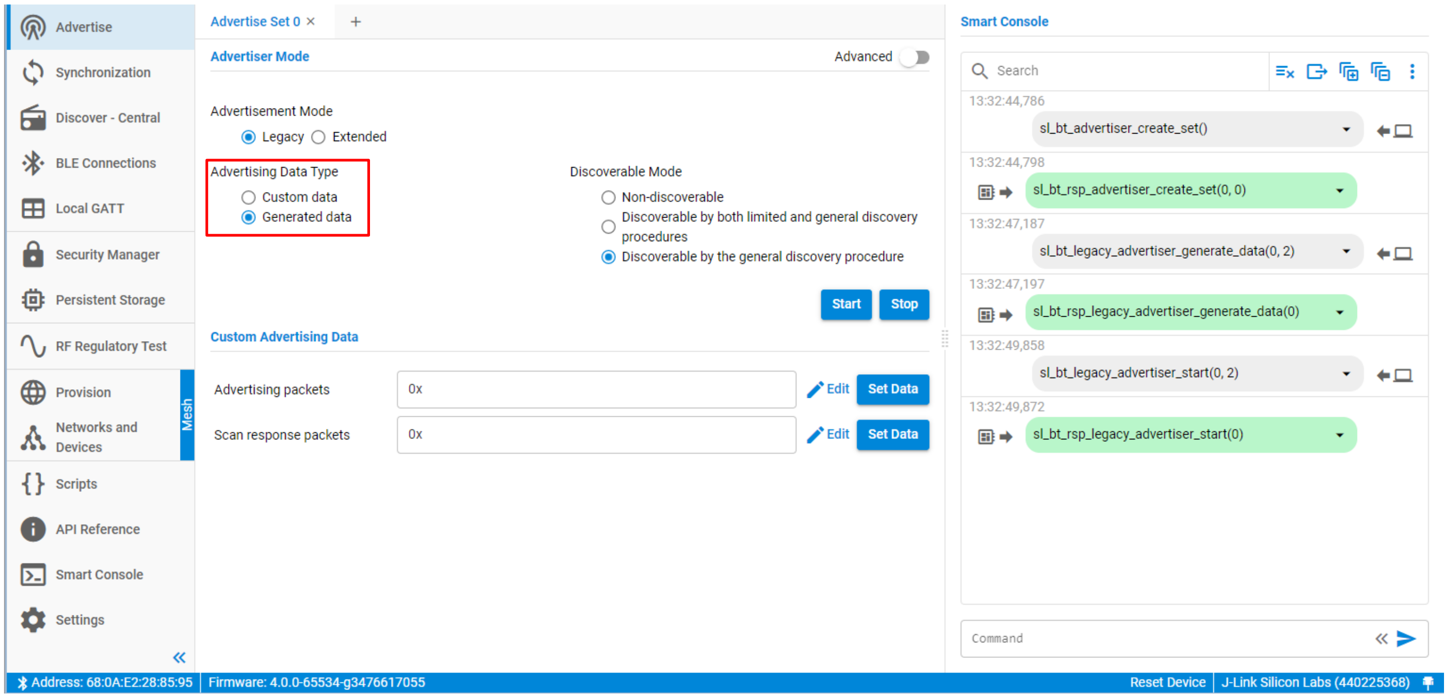
System ID UUID: 2A23 Handle: 18 — Properties — Value:

```
sl_bt_gattdb_start_characteristic(1,...
19:51:26,89
sl_bt_rsp_gattdb_start_characterist...
19:51:26,148
sl_bt_gattdb_add_uuid16_char...
19:51:26,178
sl_bt_rsp_gattdb_add_uuid16_char...
19:51:26,250
sl_bt_gattdb_start_characteristic(1,...
19:51:26,302
sl_bt_rsp_gattdb_start_characterist...
19:51:26,365
sl_bt_gattdb_commit(1)
19:51:26,398
sl_bt_rsp_gattdb_commit(0)
```

4. To start advertising your device so that other devices can discover it and connect to it, select the 'Advertise' menu, click the '+' button (**Create Set)** to create an advertiser set.



5. To populate the advertisement payload with the device name set the Advertising Data Type to **Generated data**.



Advertisement Mode: [✓ Legacy] Extended

Advertising Data Type: Custom data [✓ Generated data]

Discoverable Mode: Non-discoverable / Discoverable by both limited and general discovery procedures / [✓ Discoverable by the general discovery procedure]

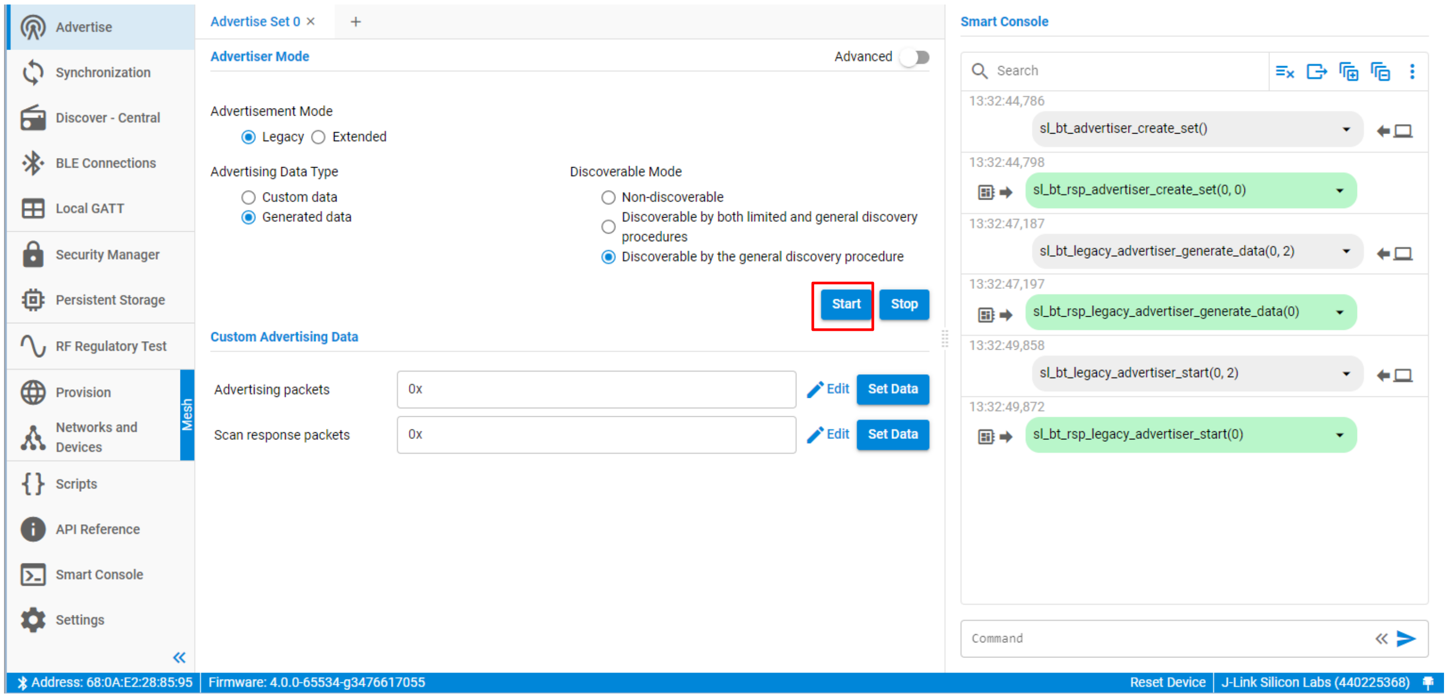
Advertising packets: 0x

Scan response packets: 0x

```
13:32:44,786
sl_bt_advertiser_create_set()
13:32:44,798
sl_bt_rsp_advertiser_create_set(0, 0)
13:32:47,187
sl_bt_legacy_advertiser_generate_data(0, 2)
13:32:47,197
sl_bt_rsp_legacy_advertiser_generate_data(0)
13:32:49,858
sl_bt_legacy_advertiser_start(0, 2)
13:32:49,872
sl_bt_rsp_legacy_advertiser_start(0)
```

Address: 68:0A:E2:28:85:95 | Firmware: 4.0.0-65534-g3476617055

6. Click **Start** to start advertising.



Advertisement Mode: [✓ Legacy] Extended

Advertising Data Type: Custom data [✓ Generated data]

Discoverable Mode: Non-discoverable / Discoverable by both limited and general discovery procedures / [✓ Discoverable by the general discovery procedure]

Advertising packets: 0x

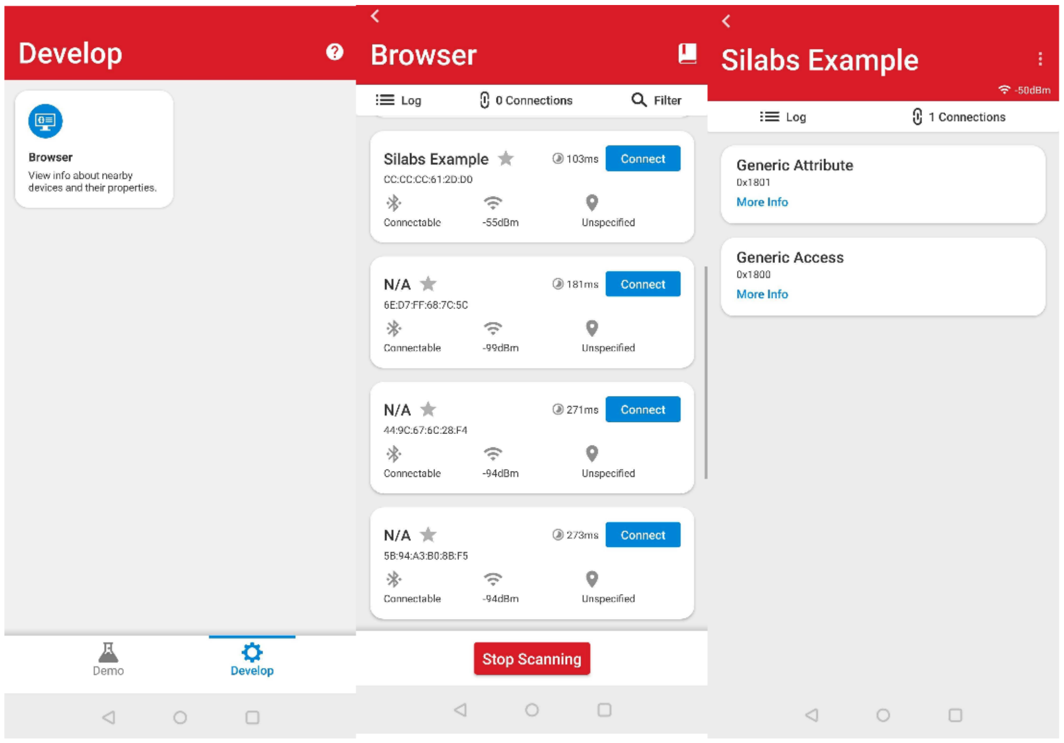
Scan response packets: 0x

```
13:32:44,786
sl_bt_advertiser_create_set()
13:32:44,798
sl_bt_rsp_advertiser_create_set(0, 0)
13:32:47,187
sl_bt_legacy_advertiser_generate_data(0, 2)
13:32:47,197
sl_bt_rsp_legacy_advertiser_generate_data(0)
13:32:49,858
sl_bt_legacy_advertiser_start(0, 2)
13:32:49,872
sl_bt_rsp_legacy_advertiser_start(0)
```

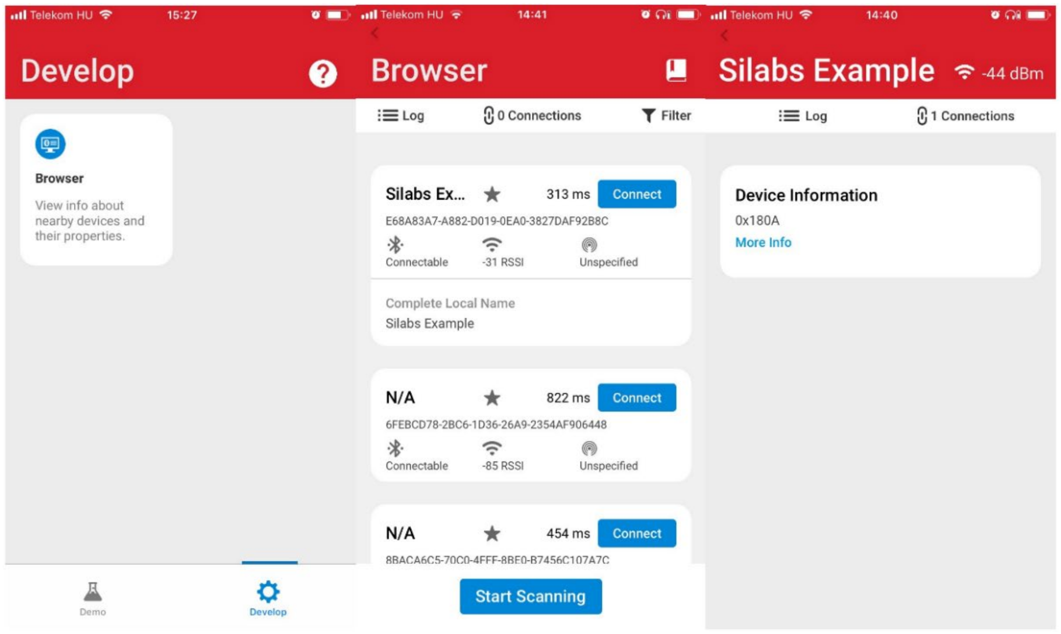
Address: 68:0A:E2:28:85:95 | Firmware: 4.0.0-65534-g3476617055

Testing with the Smartphone App

1. On the master side (smartphone), install the **EFR Connect** app from the Google Play Store or the App Store, and open it. To find your advertising device, tap the Develop tab, and tap **Bluetooth Browser**. This shows all advertising devices nearby. Connect to your device by tapping **Connect** next to "Silabs Example". Its GATT database is automatically discovered and displayed. Tap any service to list its characteristics and tap any characteristic to read its value.



4.1 Testing on Android Smartphone

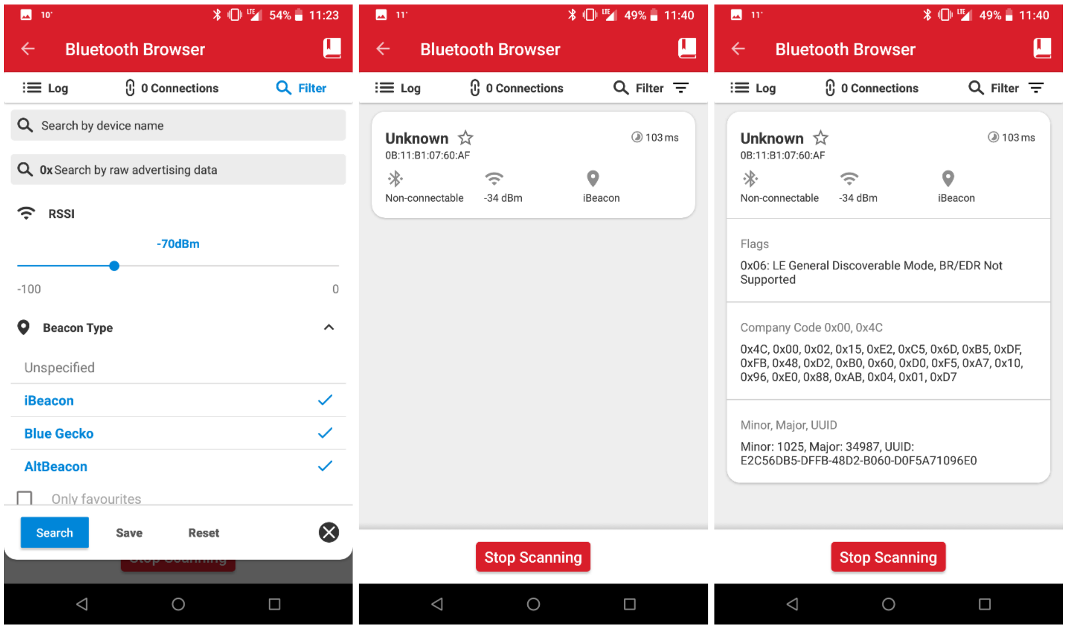


4.2 Testing on iOS Smartphone

4.3.2 Testing the iBeacon Demo

Bluetooth beacons are unconnectable advertisements that help you locate a device, determine your own position, or get minimal information about an asset the beaconing device is attached to.

After flashing the **iBeacon** demo to your device, you can find the beacon signal with the **Bluetooth Browser** in the **EFR Connect** app. Start **EFR Connect**, tap the Develop tab, and tap **Bluetooth Browser**. To filter beacons, tap **Filter**, and select the beacon types you want to be displayed. The app provides you with basic information about the beacon, like RSSI - which can help determine the distance of the beacon. Tap the beacon to get more information about the data it provides.



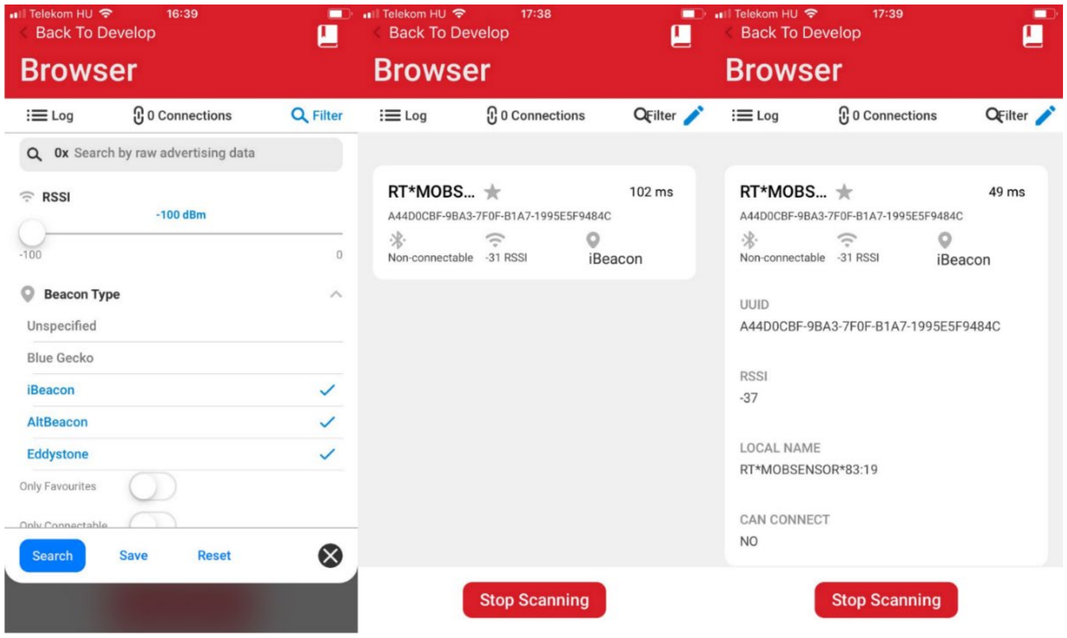
0B:11:B1:07:60:AF

Flags
0x06: LE General Discoverable Mode, BR/EDR Not Supported

Company Code 0x00, 0x4C
0x4C, 0x00, 0x02, 0x15, 0xE2, 0xC5, 0x6D, 0xB5, 0xDF, 0xFB, 0x48, 0xD2, 0xB0, 0x60, 0xD0, 0xF5, 0xA7, 0x10, 0x96, 0xE0, 0x88, 0xAB, 0x04, 0x01, 0xD7

Minor, Major, UUID
Minor: 1025, Major: 34987, UUID: E2C56DB5-DFFB-48D2-B060-D0F5A71096E0

4.3 Testing on Android Smartphone



A44D0CBF-9BA3-7F0F-B1A7-1995E5F9484C

UUID
A44D0CBF-9BA3-7F0F-B1A7-1995E5F9484C

RSSI
-37

LOCAL NAME
RT*MOBSENSOR*83:19

CAN CONNECT
NO

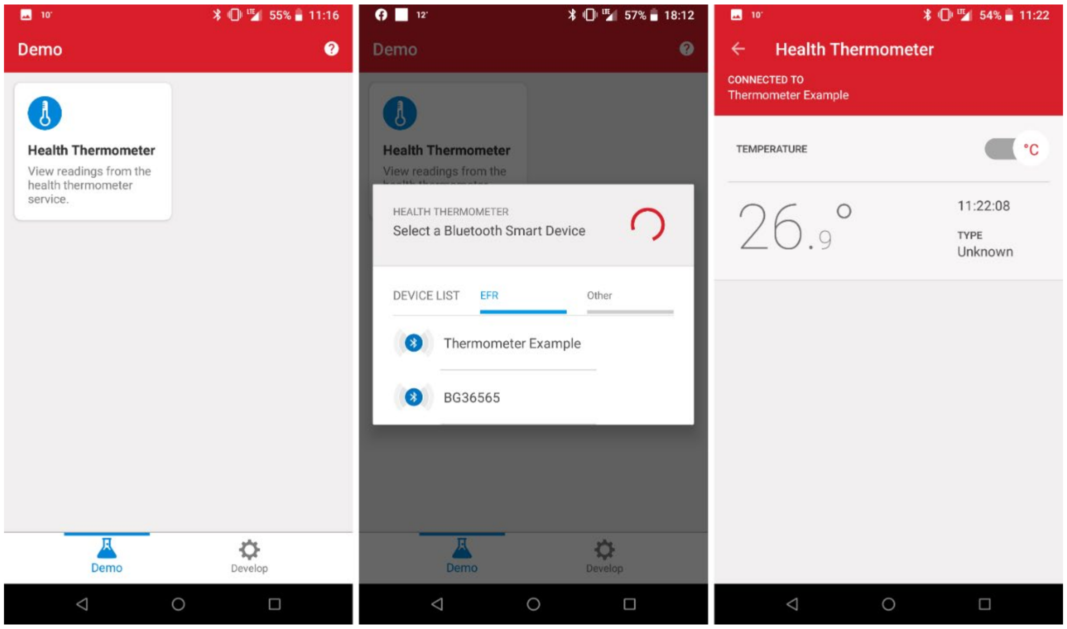
4.4 Testing on iOS Smartphone

4.3.3 Testing the Health Thermometer Demo

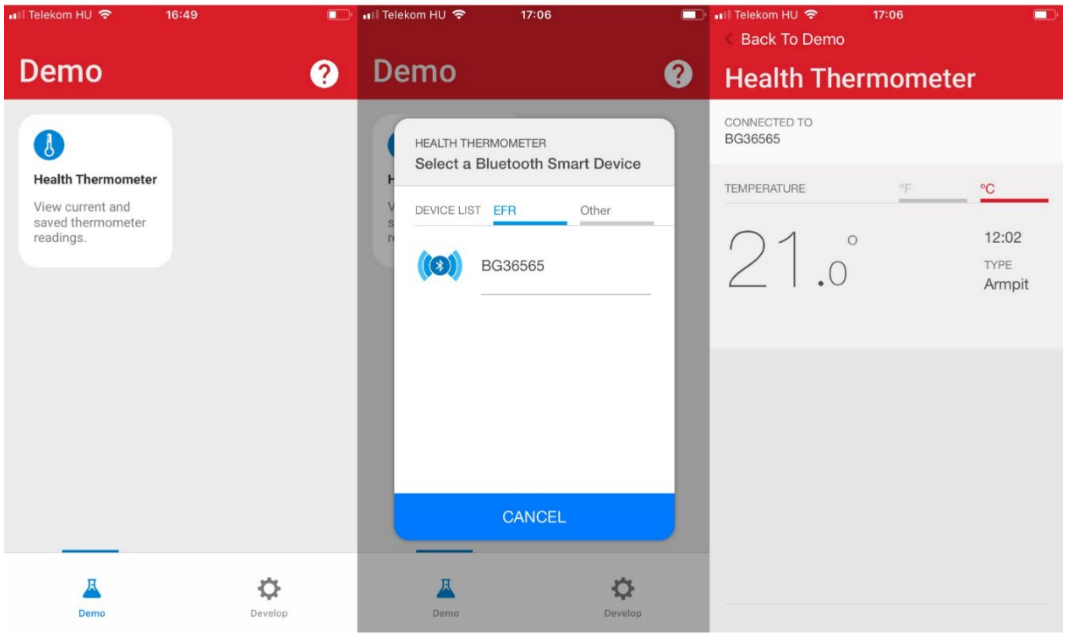
While the NCP Empty demo implements a minimal GATT database with basic static information like device name, the Health Thermometer demo extends this database with live temperature measurements.

After flashing the **Health Thermometer** demo to your device, start **EFR Connect**, tap the Demo tab, and tap **Health Thermometer**. Find your device advertising as Thermometer Example in the device list and tap it to connect. The smartphone app automatically finds the Temperature measurement characteristic of the device, reads its value periodically, and displays the value on the screen of the phone.

Try touching the temperature sensor located on the mainboard (see section 4.1 Prepare the). You should be able to see the temperature changing.



4.5 Testing on Android Smartphone



4.6 Testing on iOS Smartphone

5 Starting Application Development

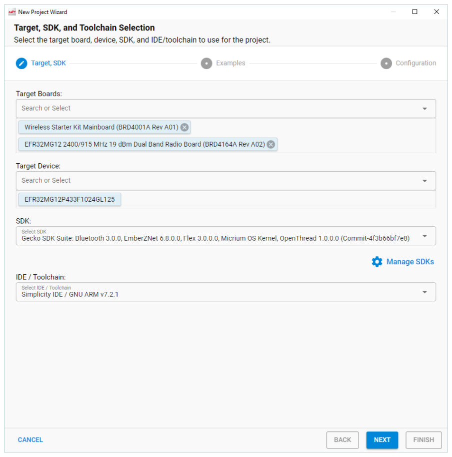
Developing a Bluetooth application consists of two main steps: defining the GATT database structure and defining the event handlers for events such as `connection_opened`, `connection_closed`, and so on.

The most common starting point for application development is the **SoC Empty** example. This project contains a simple GATT database (including the Generic Access service, Device Information service, and OTA service) and a while loop that handles some events raised by the stack. You can extend both the GATT database and the event handlers of this example according to your needs.

Note: Beginning with Bluetooth SDK version 2.7.0.0, all devices must be loaded with the Gecko Bootloader as well as the application. While you are getting started, the easiest way to do this is to load any of the precompiled demo images that come with the bootloader configured as part of the image. When you flash your application it overwrites the demo application, but the bootloader remains. Subsequently you may wish to build your own bootloader, as described in *UG266: Silicon Labs Gecko Bootloader User's Guide for GSDK 3.2 and Lower* or *UG489: Silicon Labs Gecko Bootloader User's Guide for GSDK 4.0 and Higher*.

New Project creation is done through three dialogs:

- Target, SDK, and Toolchain
- Examples
- Configuration



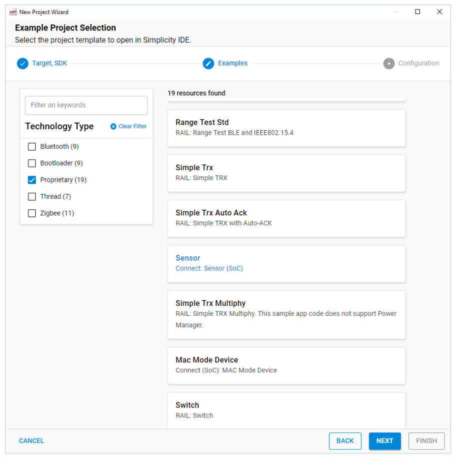
Target, SDK, and Toolchain Selection
Select the target board, device, SDK, and IDE/toolchain to use for the project.

Target Boards: Search or Select
Wireless Starter Kit Mainboard (BRD4001A Rev A01)
EFR32MG12 2400/915 MHz 19 dBm Dual Band Radio Board (BRD4164A Rev A02)

Target Device: Search or Select
EFR32MG12P433F1024GL125

SDK: Select SDK
Gecko SDK Suite: Bluetooth 3.0.0, EmberZNet 6.8.0.0, Flex 3.0.0.0, Micrium OS Kernel, OpenThread 1.0.0.0 (Commit-4f3b66bf7e8)

IDE / Toolchain: Select IDE / Toolchain
Simplicity IDE / GNU ARM v7.2.1



Example Project Selection
Select the project template to open in Simplicity IDE.

Technology Type
- [ ] Bluetooth (9)
- [ ] Bootloader (9)
- [x] Proprietary (19)
- [ ] Thread (7)
- [ ] Zigbee (11)

19 resources found

Range Test Std
RAIL: Range Test BLE and IEEE802.15.4

Simple Trx
RAIL: Simple TRX

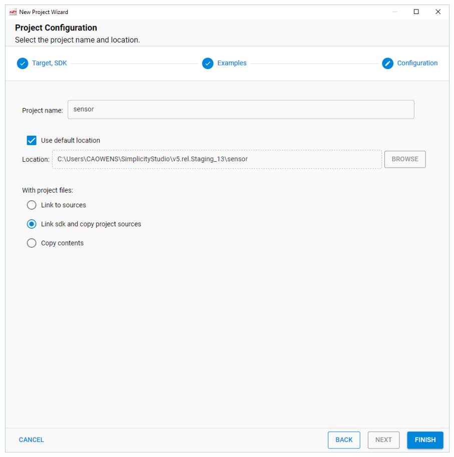
Simple Trx Auto Ack
RAIL: Simple TRX with Auto-ACK

Sensor
Connect: Sensor (SoC)

Simple Trx Multiphy
RAIL: Simple TRX Multiphy. This sample app code does not support Power Manager.

Mac Mode Device
Connect (SoC): MAC Mode Device

Switch
RAIL: Switch



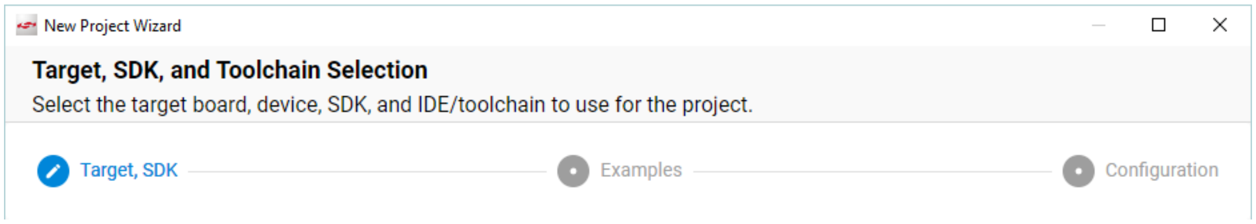
Project Configuration
Select the project name and location.

Project name: sensor

[✓] Use default location
Location: C:\Users\CAOWENS\SimplicityStudio\v5.rel.Staging_13\sensor

With project files:
Link to sources [✓ Link sdk and copy project sources] Copy contents

An indicator at the top of the dialog shows you where you are.



Target, SDK, and Toolchain Selection
Select the target board, device, SDK, and IDE/toolchain to use for the project.

You can start a project from different locations in the Launcher Perspective, as described in the [Simplicity Studio 5 User's Guide](). While you are getting started, we suggest starting from the File menu, as that takes you through all three of the above dialogs.

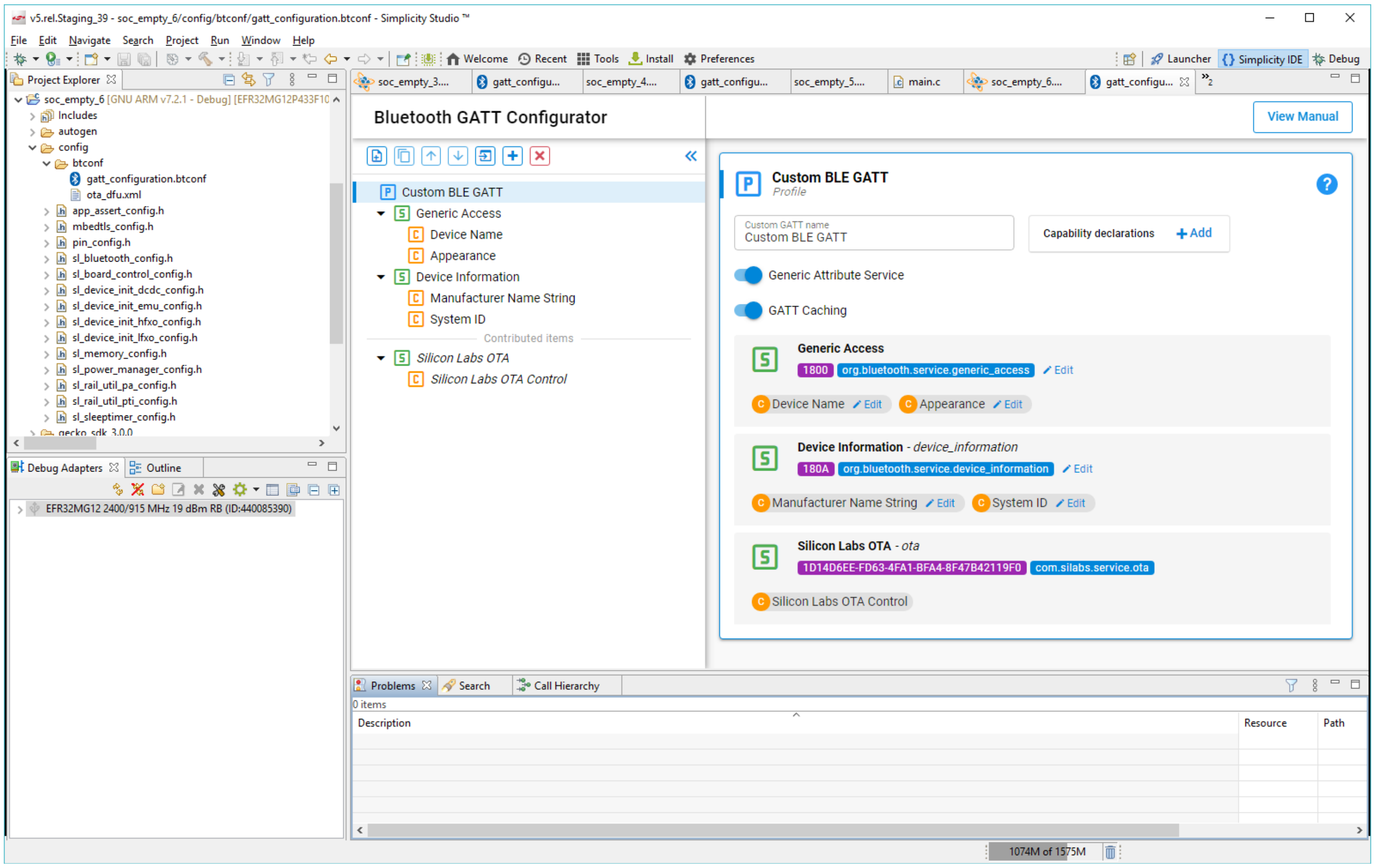
1. Select **New >> Silicon Labs Project Wizard**.
2. Review your SDK and toolchain. If you wish to use IAR instead of GCC, be sure to change it here. Once you have created a project it is difficult to change toolchains. Click **NEXT**.
3. On the Example Project Selection dialog, filter on Bluetooth and select **Bluetooth - SoC Empty**. Click **NEXT**.

On the Project Configuration dialog, rename your project if you wish. Note that if you change any linked resource, it is changed for any other project that references it. While you are getting started the default choice to include project files but link to the SDK is best. Click **FINISH.** If the example has documentation, the project opens on a readme tab. Note that a Simplicity IDE perspective control is now included in the upper right of the screen.

5.1 GATT Database

A visual GATT Configurator is available on the gatt_configuration.btconf tab when you create the project, to help you create your own GATT database with a few clicks.

You can create your own database at this point, or return to it later either by double-clicking the gatt_configuration.btconf file under your project in Project Explorer, or through the Project Configurator's Advanced > GATT Configurator component. For more information, see section 6.1 The GATT Configurator.



A reference for each characteristic is generated and defined in gatt_db.h. You can use this reference in your code to read / write the values of the characteristics in the local GATT database with `sl_bt_gatt_server_read_attribute_value()` / `sl_bt_gatt_server_write_attribute_value()` commands.

You can also build the GATT database from the application using the GATT database API. In this case you need to install the Dynamic GATT Database software component on the Project Configurator Software Components tab.

You will find the event handlers in the main loop in app.c. You can extend this list with further event handlers. The full list of events – and stack commands – can be found in the Bluetooth Software API Reference Manual.

5.2 Component Configuration

Bluetooth SDK v3.x projects are based on a Gecko Platform component-based architecture. Software features and functions can be installed and configured through Simplicity Studio's Component Editor. When you install a component, the installation process will:

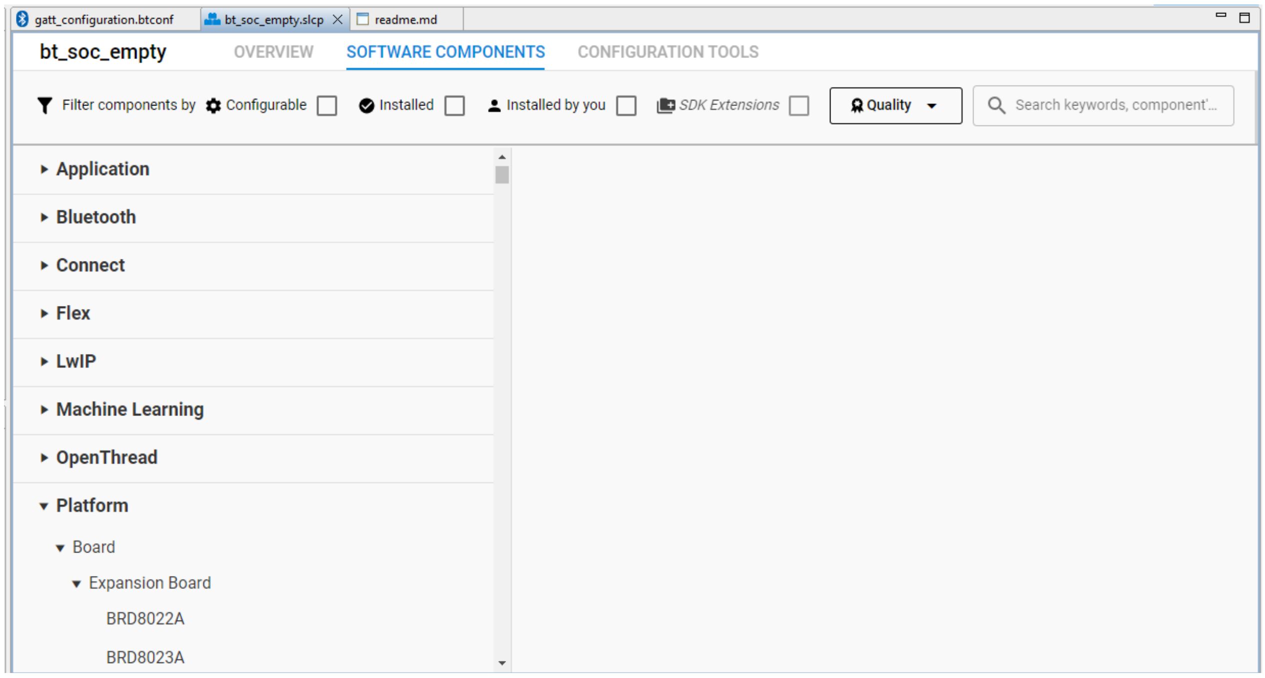
1. Copy the corresponding SDK files from the SDK folder into the project folder.
2. Copy all the dependencies of the given component into the project folder.
3. Add new include directories to the project settings.
4. Copy the configurations files into the /config folder.
5. Modify the corresponding auto-generated files to integrate the component into the application.

Additionally, “init” type software components will implement the initialization code for a given component, utilizing their corresponding configuration file as input.

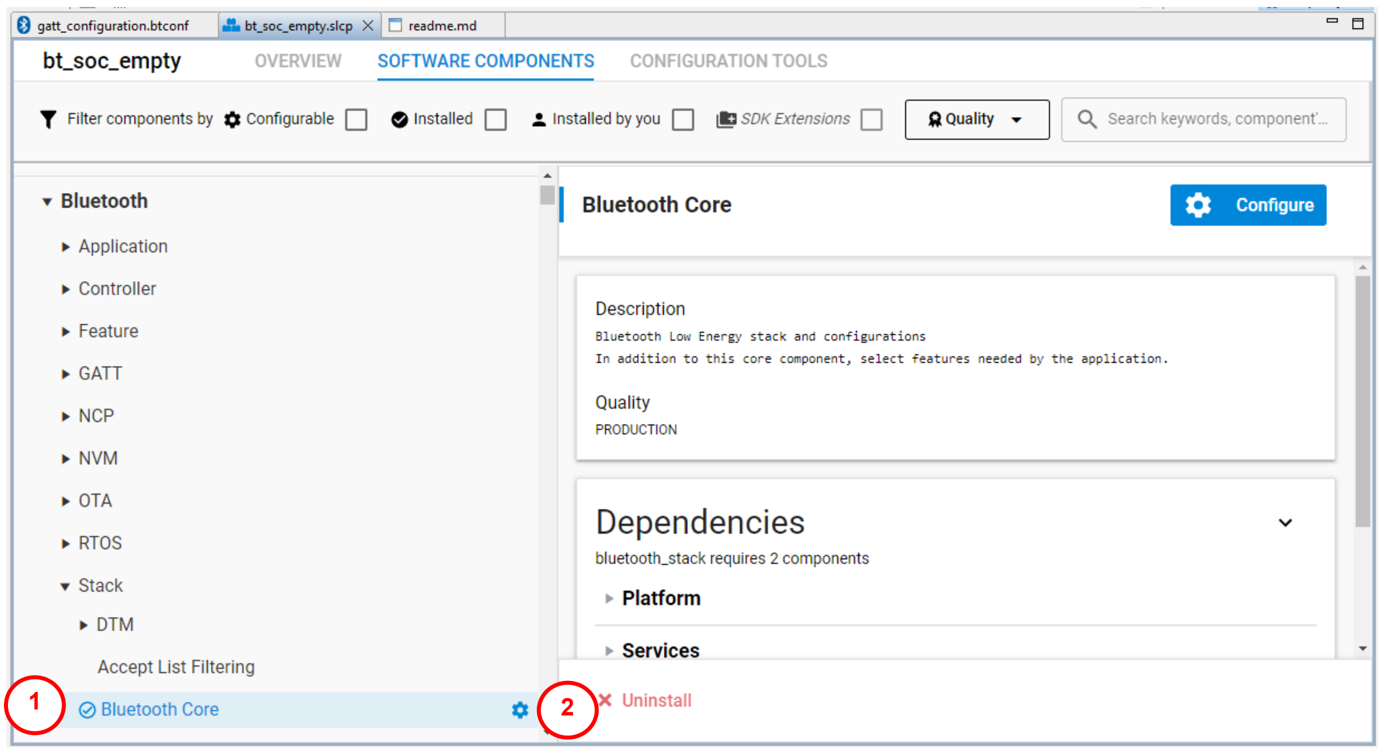
Some software components (like OTA DFU) will fully integrate into the application to perform a specific task without the need of any additional code, while other components provide an API to be used in the application.

Note: All EFR32 parts have a unique RSSI offset. In addition, board, antenna and enclosure design can also impact RSSI. When creating a new project, install the **RAIL Utility, RSSI** component. This feature includes the default RSSI Offset Silicon Labs has measured for each part. This offset can be modified if necessary, after RF testing of your complete product.

To see the component library, click the <project-name>.slcp tab of your project, and click **Software Components**. A number of filters as well as a keyword search are available to help you explore the various component categories. Note that components for all SDKs are presented.



Components installed in the project are checked (1) and can be uninstalled. Configurable components are indicated by a gear symbol (2).



Bluetooth Core

Description

```
Bluetooth Low Energy stack and configurations
In addition to this core component, select features needed by the application.
```

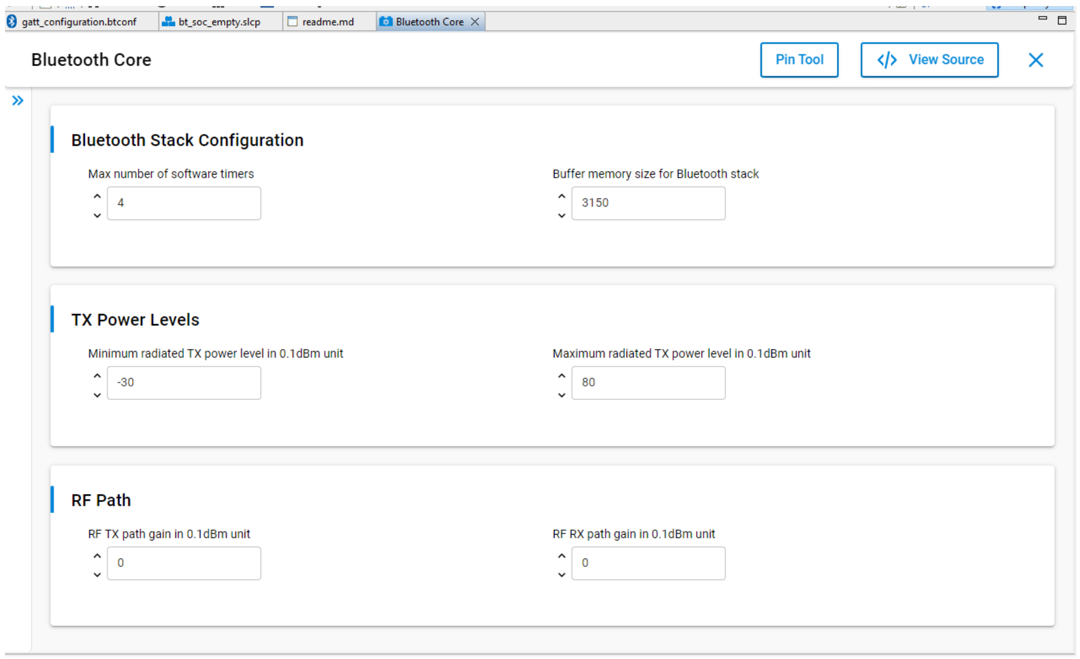
Quality

PRODUCTION

Dependencies

bluetooth_stack requires 2 components

Click **Configure** to open the Component Editor and see a configurable component's parameters.



Bluetooth Core

Bluetooth Stack Configuration

Max number of software timers: 4

Buffer memory size for Bluetooth stack: 3150

TX Power Levels

Minimum radiated TX power level in 0.1dBm unit: -30

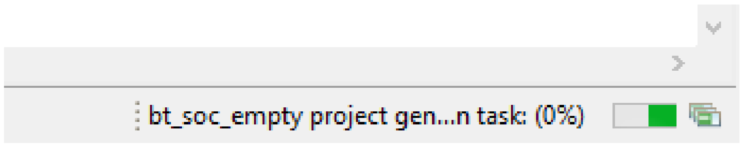
Maximum radiated TX power level in 0.1dBm unit: 80

RF Path

RF TX path gain in 0.1dBm unit: 0

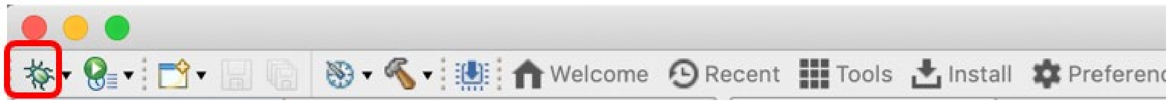
RF RX path gain in 0.1dBm unit: 0

As you change component configurations, your changes are automatically saved and project files are automatically generated. You can see generation progress in the lower right corner of the Simplicity IDE. Wait until generation is complete before building the application image.



5.3 Building and Flashing

To build and debug your project click **Debug** (bug icon) on the Simplicity IDE It will build and download your project and open up the Debug perspective. Click **Play** (next to Debug) to start running you project on the device.



5.4 Enabling Field Updates

Deploying new firmware for devices in the field can be done by UART DFU (Device Firmware Update) or, for SoC applications, OTA DFU. For more information on each of these methods refer to AN1086: Using the Gecko Bootloader with the Silicon Labs Bluetooth Applications.

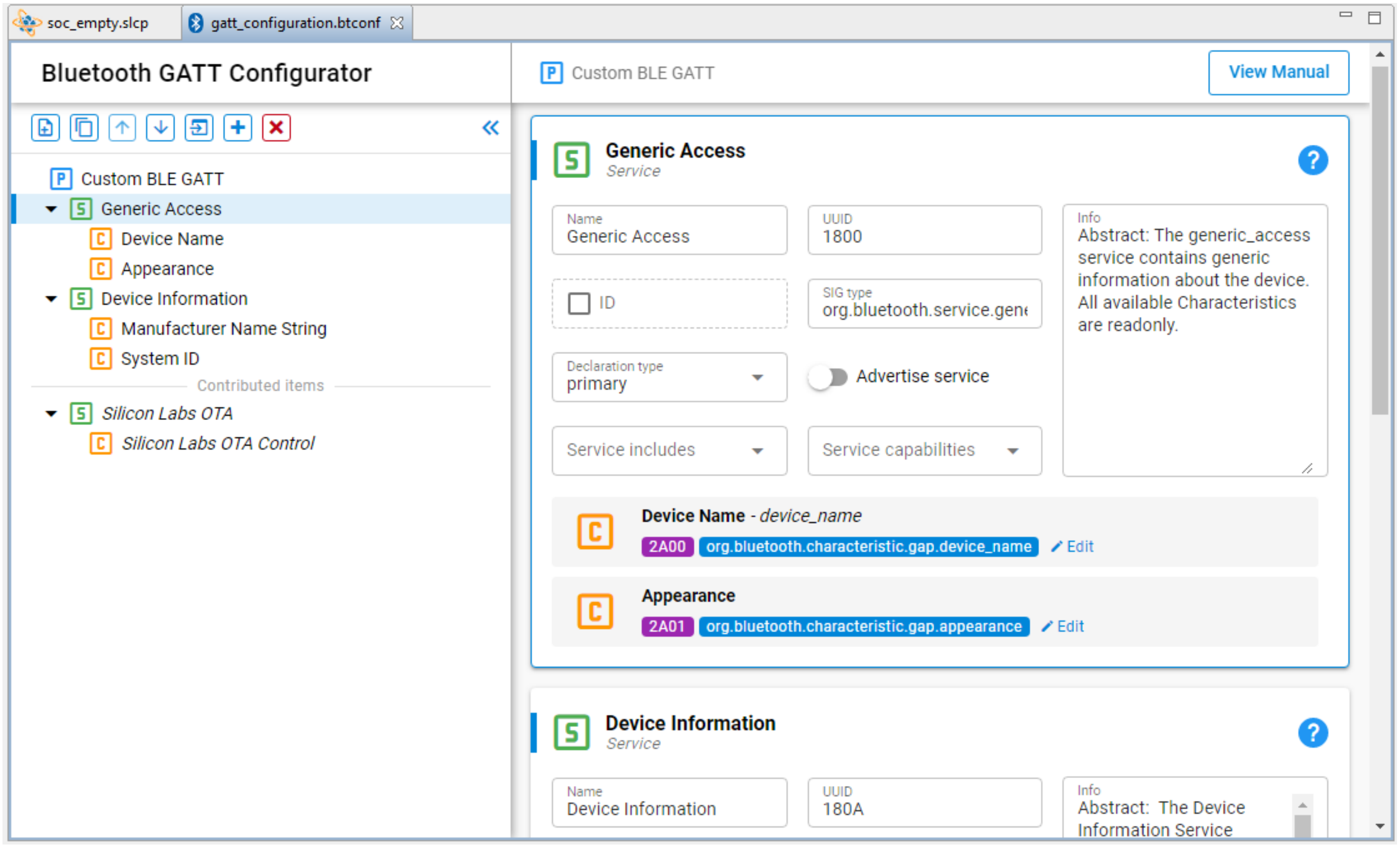
6 Development Tools

6.1 The GATT Configurator

Every Bluetooth connection has a GATT client and a GATT server. The server holds a GATT database: a collection of *Characteristic*s that can be read and written by the client. The Characteristics are grouped into *Services*, and the group of Services determines a *Bluetooth Profile*.

If you are implementing a GATT server (typically on the peripheral device), you have to define a GATT database structure. Clients (typically the central device) can also have a GATT database, even if no device will query it, so you can keep the default database structure in your code. This structure can either be designed in runtime using the dynamic GATT API or in compile-time using the GATT Configurator. For an SoC application, implementing a static GATT database with the GATT Configurator is recommended, because it ensures faster startup and lower memory consumption than you can achieve with dynamic GATT configuration.

The GATT Configurator is a simple-to-use tool to help you build your own GATT database. A list of project Profiles/Services/Characteristics/Descriptors is shown on the left and details about the selected item is shown on the right. An options menu is provided above the Profiles list.



Generic Access
Service

Name: Generic Access
UUID: 1800
Info: Abstract: The generic_access service contains generic information about the device. All available Characteristics are readonly.
ID
SIG type: org.bluetooth.service.gene
Declaration type: primary
Advertise service
Service includes
Service capabilities

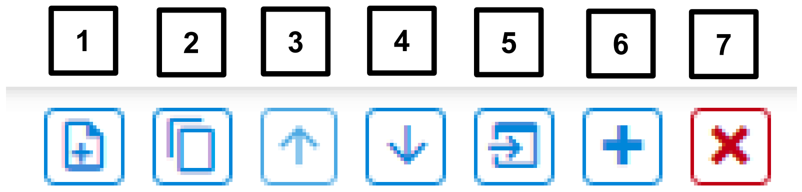
Device Name - *device_name*
2A00 org.bluetooth.characteristic.gap.device_name Edit

Appearance
2A01 org.bluetooth.characteristic.gap.appearance Edit

Device Information
Service

Name: Device Information
UUID: 180A
Info: Abstract: The Device Information Service

The GATT Configurator menu is:



1) Add an item.
2) Duplicate the selected item.
3) Move the selected item up.
4) Move the selected item down.
5) Import a GATT database.
6) Add Predefined.
7) Delete the selected item.

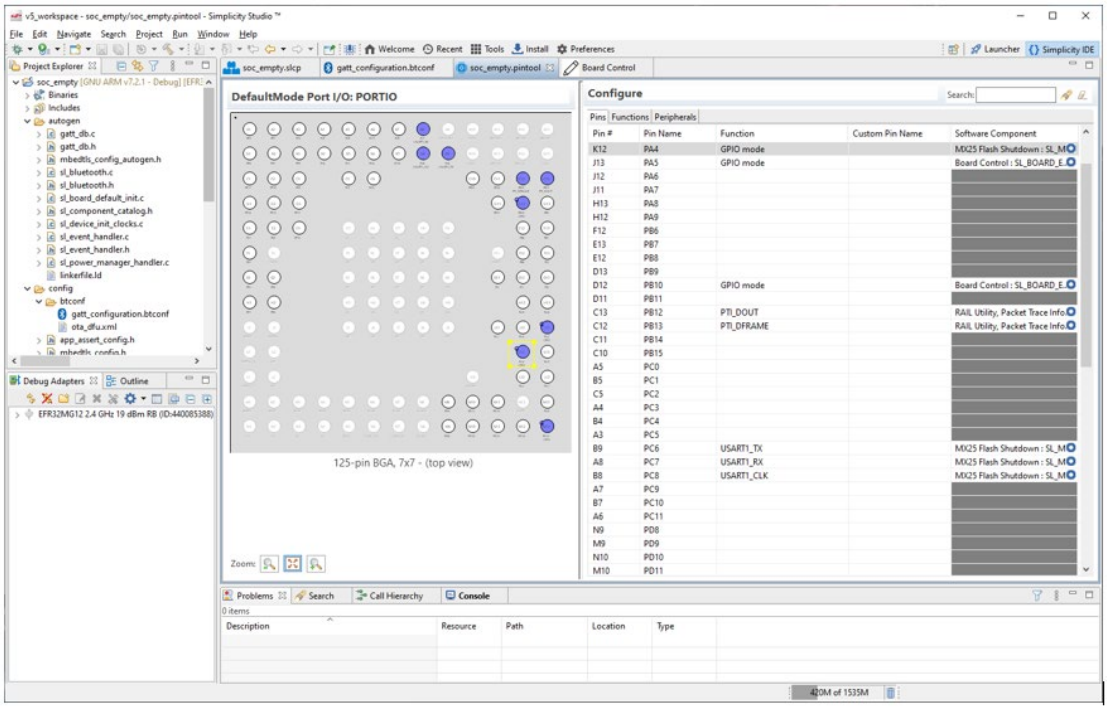
To add a custom service, click the **Profile (Custom BLE GATT)**, and then click **Add** (1). To add a custom characteristic, select a service and then click **Add** (1). To add a predefined service/characteristic click **Add Predefined** (6). To learn more about the configurator see UG438: GATT Configurator User's Guide for Bluetooth SDK v3.x.

You can find a detailed description of any Profile/Service/Characteristic/Descriptor on https://www.bluetooth.com/specifications/gatt.

Characteristics are generally complex structures of fields. If you want to know what fields a characteristic has, visit https://www.bluetooth.com/specifications/gatt/characteristics.

6.2 The Pin Tool

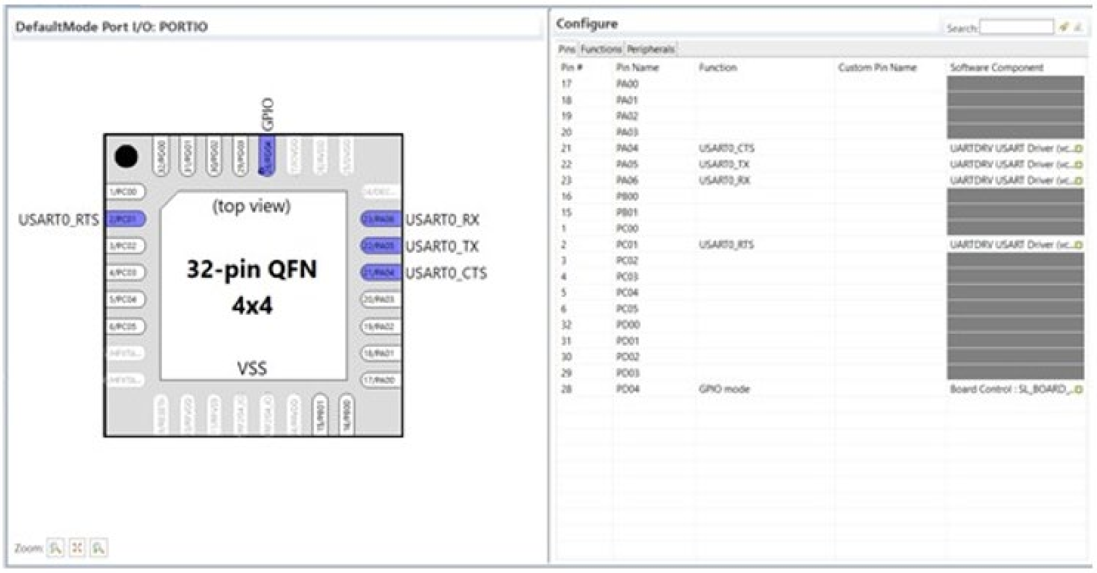
Simplicity Studio 5 offers a Pin Tool that allows you to easily configure new peripherals or change the properties of existing ones. In the Project Configurator SOFTWARE COMPONENTS tab, expand the Advanced Configurators group and open the Pin Tool. The graphical view differs based on the chip.


v5_workspace - soc_empty/soc_empty.pintool - Simplicity Studio ™
File Edit Navigate Search Project Run Window Help
Welcome Recent Tools Install Preferences
Launcher Simplicity IDE
Project Explorer
soc_empty.slcp gatt_configuration.btconf soc_empty.pintool Board Control
DefaultMode Port I/O: PORTIO
125-pin BGA, 7x7 - (top view)
Zoom:
Debug Adapters Outline
EFR32MG12 2.4 GHz 19 dBm RB (ID:440085388)
Problems Search Call Hierarchy Console
0 items
Description Resource Path Location Type
420M of 1535M

Configure

Pins | Functions | Peripherals

Pin #	Pin Name	Function	Custom Pin Name	Software Component
K12	PA4	GPIO mode		MX25 Flash Shutdown : SL_M
J13	PA5	GPIO mode		Board Control : SL_BOARD_E
J12	PA6			
J11	PA7			
H13	PA8			
H12	PA9			
F12	PB6			
E13	PB7			
E12	PB8			
D13	PB9			
D12	PB10	GPIO mode		Board Control : SL_BOARD_E
D11	PB11			
C13	PB12	PTI_DOUT		RAIL Utility, Packet Trace Info
C12	PB13	PTI_DFRAME		RAIL Utility, Packet Trace Info
C11	PB14			
C10	PB15			
A5	PC0			
B5	PC1			
C5	PC2			
A4	PC3			
B4	PC4			
A3	PC5			
B9	PC6	USART1_TX		MX25 Flash Shutdown : SL_M
A8	PC7	USART1_RX		MX25 Flash Shutdown : SL_M
B8	PC8	USART1_CLK		MX25 Flash Shutdown : SL_M
A7	PC9			
B7	PC10			
A6	PC11			
N9	PD8			
M9	PD9			
N10	PD10			
M10	PD11			

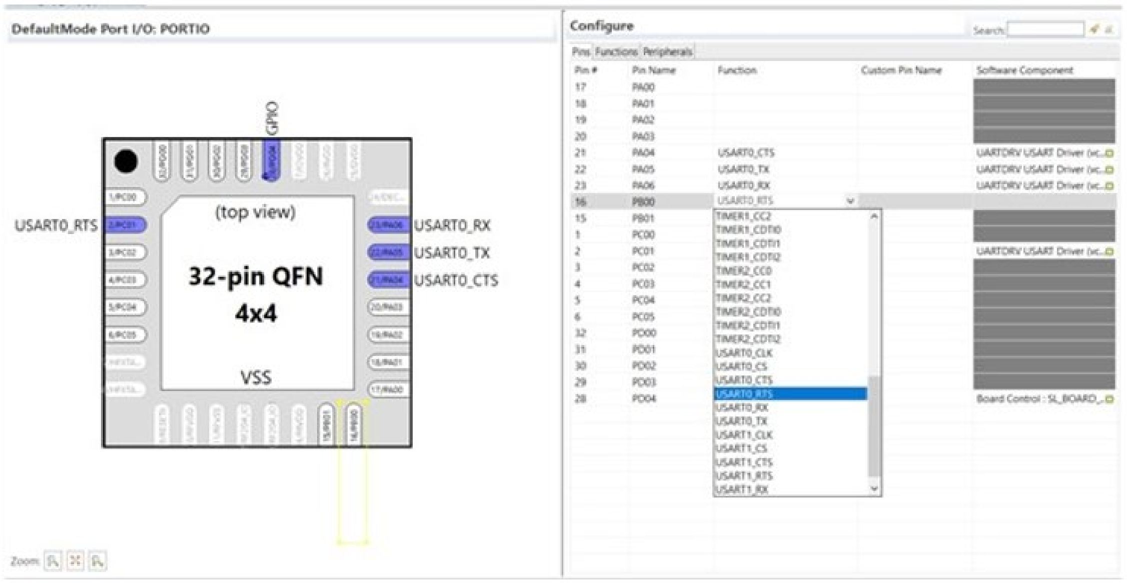


Configure

Pins | Functions | Peripherals

Pin #	Pin Name	Function	Custom Pin Name	Software Component
17	PA00			
18	PA01			
19	PA02			
20	PA03			
21	PA04	USART0_CTS		UARTDRV USART Driver (vc...
22	PA05	USART0_TX		UARTDRV USART Driver (vc...
23	PA06	USART0_RX		UARTDRV USART Driver (vc...
16	PB00			
15	PB01			
1	PC00			
2	PC01	USART0_RTS		UARTDRV USART Driver (vc...
3	PC02			
4	PC03			
5	PC04			
6	PC05			
32	PD00			
31	PD01			
30	PD02			
29	PD03			
28	PD04	GPIO mode		Board Control : SL_BOARD_...

For example, you can reassign the pins used for USART communication to the appropriate layout for a custom board design by selecting the desired pin in the list and then selecting its functionality from the drop-down list.



Pin #	Pin Name	Function	Custom Pin Name	Software Component
17	PA00			
18	PA01			
19	PA02			
20	PA03			
21	PA04	USART0_CTS		UARTDRV USART Driver (vc...
22	PA05	USART0_TX		UARTDRV USART Driver (vc...
23	PA06	USART0_RX		UARTDRV USART Driver (vc...
16	PB00	USART0_RTS		
15	PB01			
1	PC00			
2	PC01			UARTDRV USART Driver (vc...
3	PC02			
4	PC03			
5	PC04			
6	PC05			
32	PD00			
31	PD01			
30	PD02			
29	PD03			
28	PD04			Board Control : SL_BOARD_...

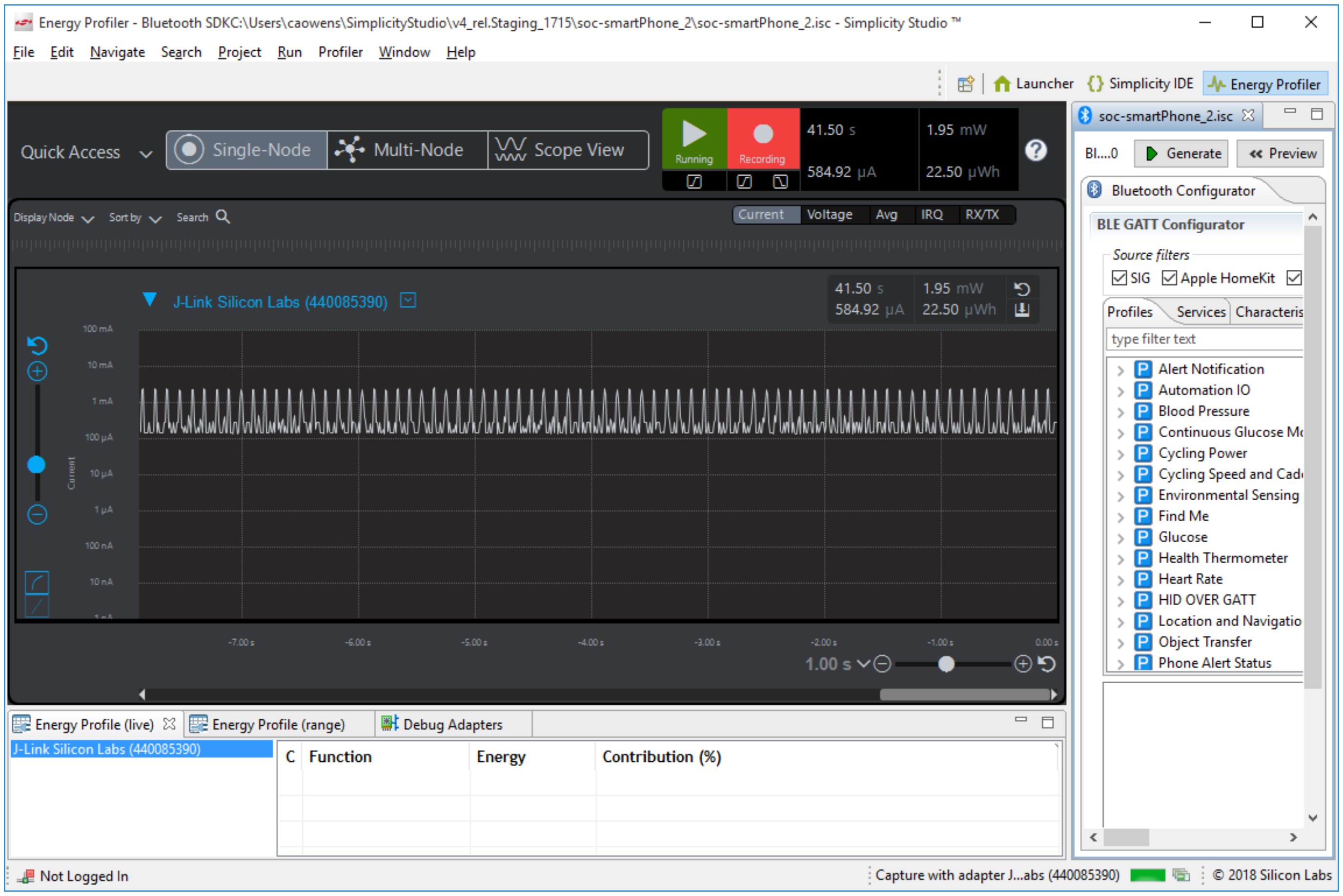
After clicking the selected item, the layout will be updated. After saving the file, the configuration source codes will be automatically generated. For more information see the Simplicity Studio 5 User's Guide.

6.3 Multi-Node Energy Profiler

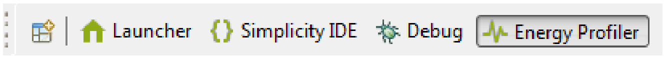
Multi-Node Energy profiler is an add-on tool, with which you can easily measure the energy consumption of your device in runtime. You can easily find peak and average consumption, and check for sleep mode current.

Note: The SDK sample apps for EFR32BG22 enable EM2 debug (see init_mcu.c), which adds current consumption overhead compared to the datasheet values.

To profile the current project, click Tools in the menu bar and select Energy Profiler or right-click on the <project>.slcp file in the Project Explorer view and select **Profile as / Simplicity Energy Profiler target**. This automatically builds your project, uploads it to the device, and starts Energy Profiler. A new Energy Profiler perspective appears, shown in the following figure.

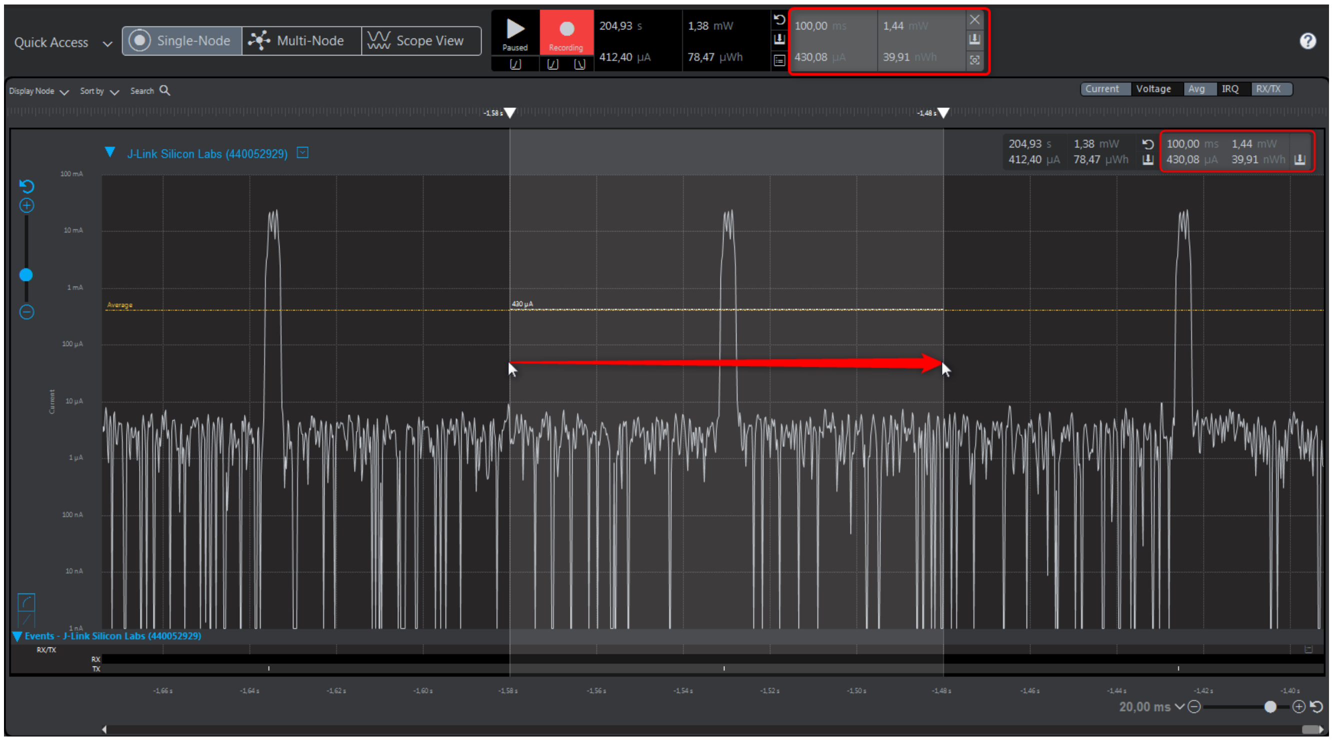


See *UG343: Multi-Node Energy Profiler User's Guide* for details on how to use this tool. You can switch easily between Simplicity IDE and Energy Profiler perspectives using the Perspective buttons in the upper right corner of your current perspective.

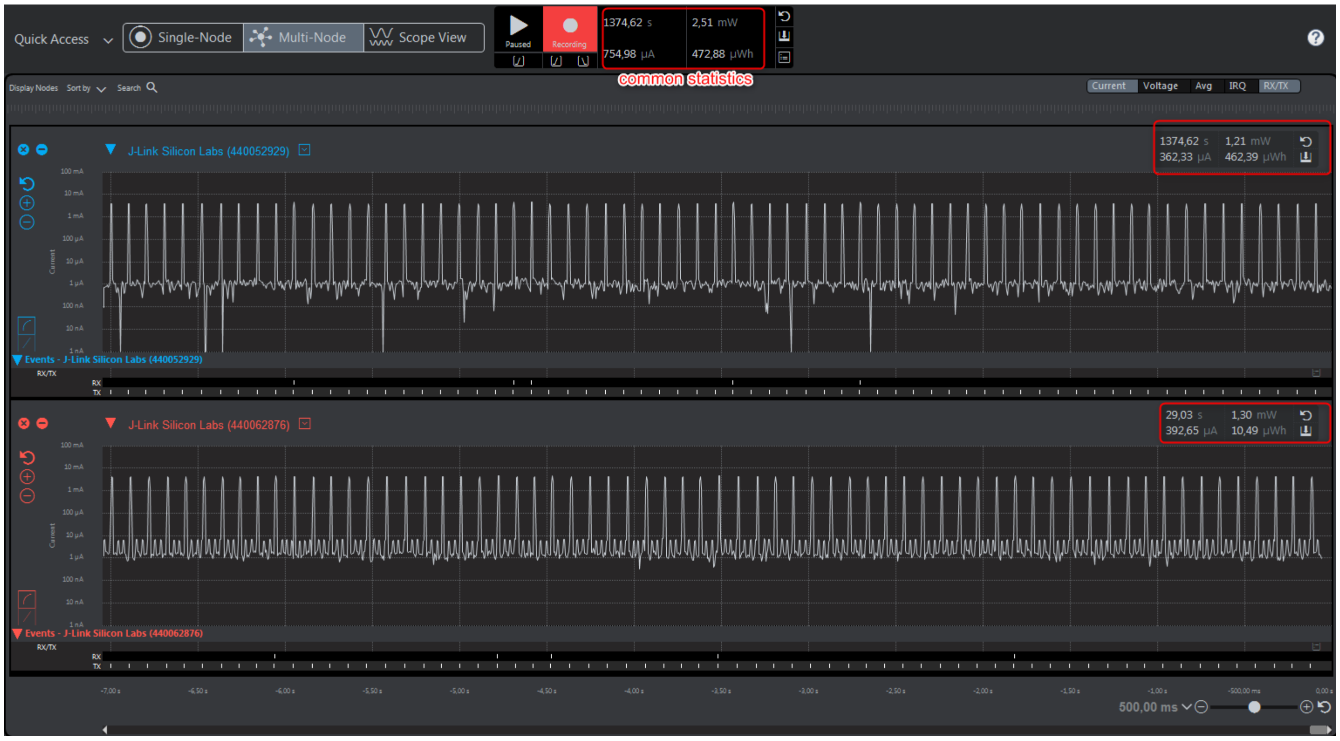


You can see peaks in the energy consumption diagram. Pause profiling by clicking Play, click one of the peaks, and zoom in with time axis (y-axis) zoom until you see three distinguishable peaks. These represent the three advertisement packets sent on the three advertisement channels. You can also see the three corresponding Tx events in the Rx/Tx bar below, provided that you enabled Rx/Tx view in the upper right corner. Note that the maximum consumption may now be greater than it appeared on the diagram before you zoomed in. This is because in zoomed-out mode, the displayed values are averaged. If you need exact values, always zoom in.

To measure average consumption, simply click and drag your mouse over a time interval. A new window appears in the upper right corner showing consumption information for the given interval. Bluetooth communication typically has a periodicity: the advertisement or the connection interval. It is recommended to measure average over an advertisement or connection interval to obtain a proper average consumption. Overall average is measured as well, but this is influenced by transient events.



Multi-node Energy Profiler is also able to simultaneously measure the consumption of multiple devices. To start measuring a new device click the Quick Access menu (upper left corner) and select **Start Energy Capture**. To stop measuring, click the Quick Access menu, and select **End/Save session**.

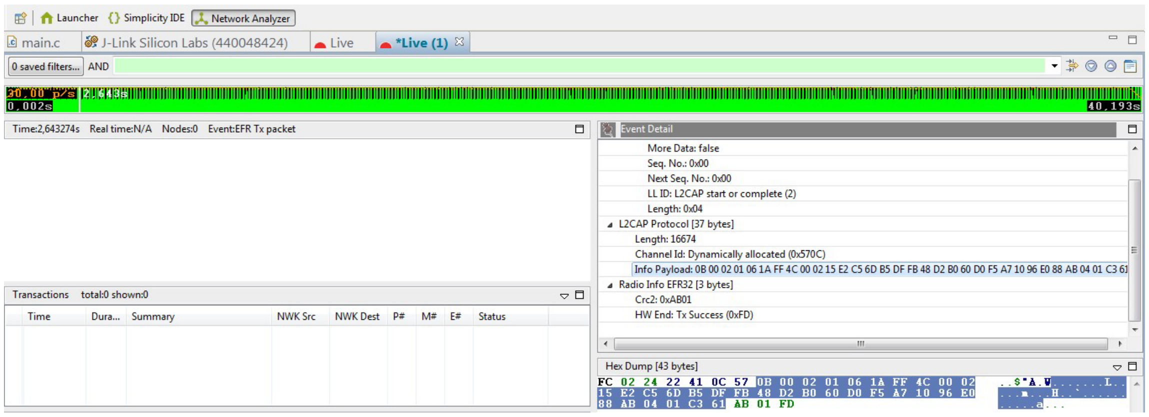


To learn more about how to use this tool, see the Simplicity Studio 5 User's Guide.

6.4 Network Analyzer

Silicon Labs Network Analyzer is a free-of-charge packet capture and debugging tool that can be used to debug Bluetooth connectivity between Wireless Geckos and other Bluetooth devices. It significantly accelerates the network and application development process with graphical views of network traffic, activity, and duration.

The Packet Trace application captures the packets directly from the Packet Trace Interface (PTI) available on the Wireless Gecko SoCs and modules. It therefore provides a more accurate capture of the packets compared to air-based capture.



Time:2,643274s Real time:N/A Nodes:0 Event:EFR Tx packet

Transactions total:0 shown:0

```
More Data: false
Seq. No.: 0x00
Next Seq. No.: 0x00
LL ID: L2CAP start or complete (2)
Length: 0x04
L2CAP Protocol [37 bytes]
  Length: 16674
  Channel Id: Dynamically allocated (0x570C)
  Info Payload: 0B 00 02 01 06 1A FF 4C 00 02 15 E2 C5 6D B5 DF FB 48 D2 B0 60 D0 F5 A7 10 96 E0 88 AB 04 01 C3 61
Radio Info EFR32 [3 bytes]
  Crc2: 0xAB01
  HW End: Tx Success (0xFD)
```

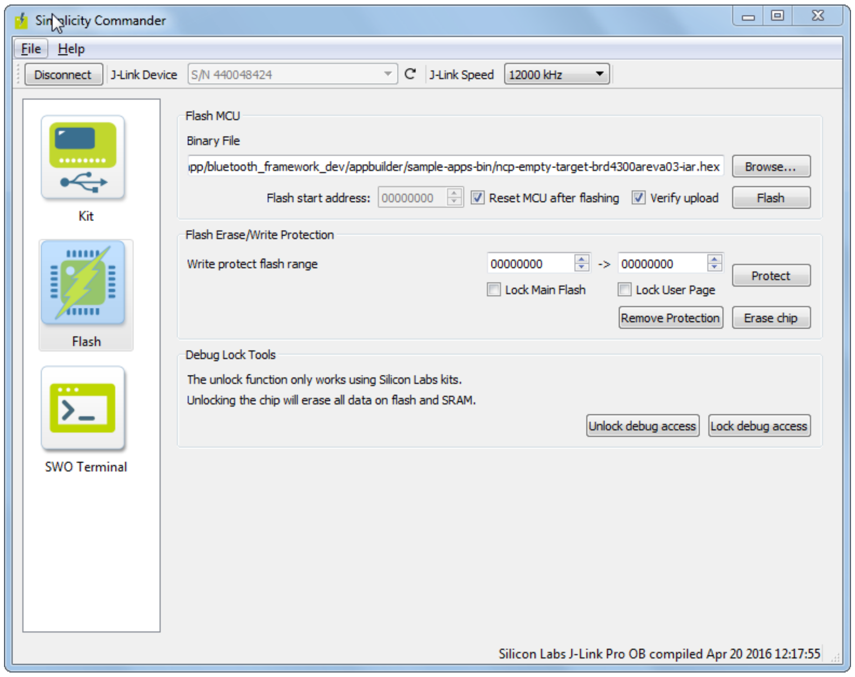
```
FC 02 24 22 41 0C 57 0B 00 02 01 06 1A FF 4C 00 02   ..$"A.W.......L..
15 E2 C5 6D B5 DF FB 48 D2 B0 60 D0 F5 A7 10 96 E0   ...m...H..`......
88 AB 04 01 C3 61 AB 01 FD                           .....a...
```

Figure 6.1. Bluetooth Traffic Capture with Packet Trace

See *AN1317: Using Network Analyzer with Bluetooth® Mesh and Low Energy* for more information.

6.5 Simplicity Commander

Simplicity Commander is a simple flashing tool, which can be used to flash firmware images, erase flash, lock and unlock debug access, and write-protect flash pages via the J-Link interface. Both GUI and CLI (Command Line Interface) are available. See *UG162: Simplicity Commander Reference Guide* for more information.



J-Link Device: S/N 440048424
J-Link Speed: 12000 kHz

Flash MCU
Binary File: ipp/bluetooth_framework_dev/appbuilder/sample-apps-bin/ncp-empty-target-brd4300areva03-iar.hex
Flash start address: 00000000 ☑ Reset MCU after flashing ☑ Verify upload

Flash Erase/Write Protection
Write protect flash range: 00000000 -> 00000000
☐ Lock Main Flash ☐ Lock User Page

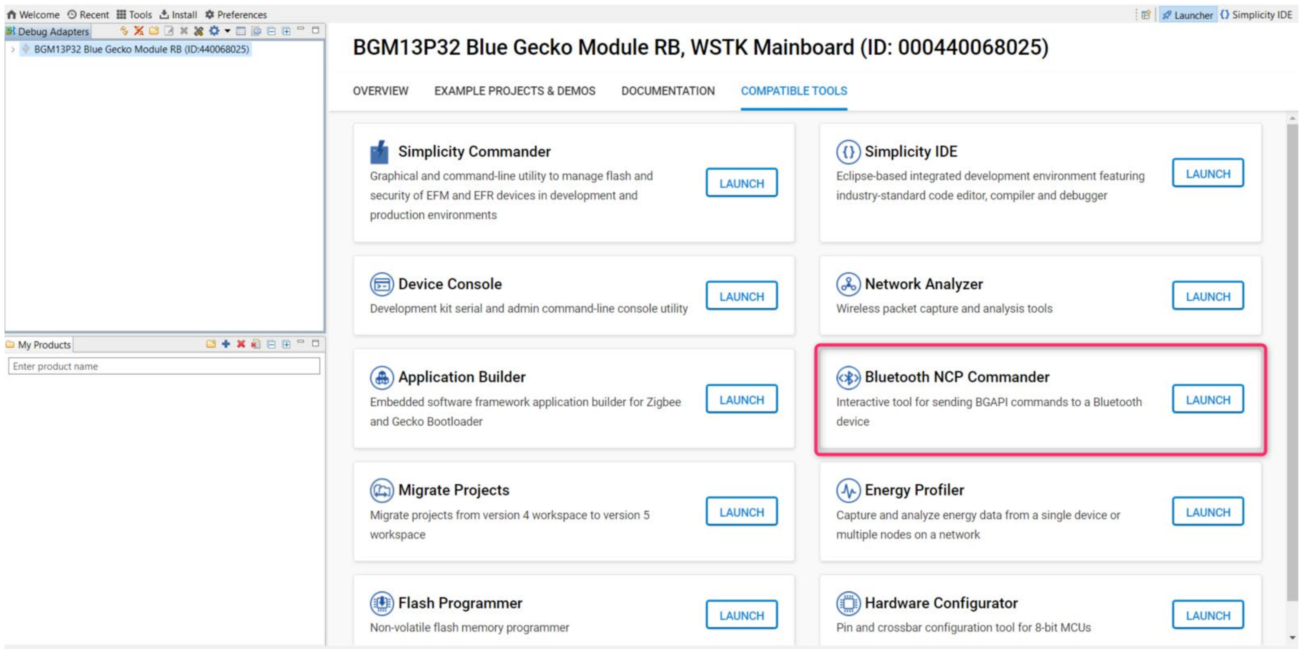
Debug Lock Tools
The unlock function only works using Silicon Labs kits.
Unlocking the chip will erase all data on flash and SRAM.

Silicon Labs J-Link Pro OB compiled Apr 20 2016 12:17:55

Figure 6.2. Simplicity Commander

6.6 Bluetooth NCP Commander

The Bluetooth NCP Commander application can be used to test and evaluate Bluetooth SoCs and modules, and it can be used to control the Bluetooth hardware using the BGAPI Serial Protocol (NCP) over a Serial/UART interface. It also supports building a GATT database dynamically on the target device. See *AN1259: Using the Silicon Labs v3.x Bluetooth® Stack in Network Co-Processor Mode* for more information.



BGM13P32 Blue Gecko Module RB, WSTK Mainboard (ID: 000440068025)

OVERVIEW | EXAMPLE PROJECTS & DEMOS | DOCUMENTATION | COMPATIBLE TOOLS

Simplicity Commander — Graphical and command-line utility to manage flash and security of EFM and EFR devices in development and production environments — LAUNCH

Simplicity IDE — Eclipse-based integrated development environment featuring industry-standard code editor, compiler and debugger — LAUNCH

Device Console — Development kit serial and admin command-line console utility — LAUNCH

Network Analyzer — Wireless packet capture and analysis tools — LAUNCH

Application Builder — Embedded software framework application builder for Zigbee and Gecko Bootloader — LAUNCH

Bluetooth NCP Commander — Interactive tool for sending BGAPI commands to a Bluetooth device — LAUNCH

Migrate Projects — Migrate projects from version 4 workspace to version 5 workspace — LAUNCH

Energy Profiler — Capture and analyze energy data from a single device or multiple nodes on a network — LAUNCH

Flash Programmer — Non-volatile flash memory programmer — LAUNCH

Hardware Configurator — Pin and crossbar configuration tool for 8-bit MCUs — LAUNCH

Figure 6.3. NCP Commander Application

Bluetooth NCP Commander is also available as a standalone application, located in:

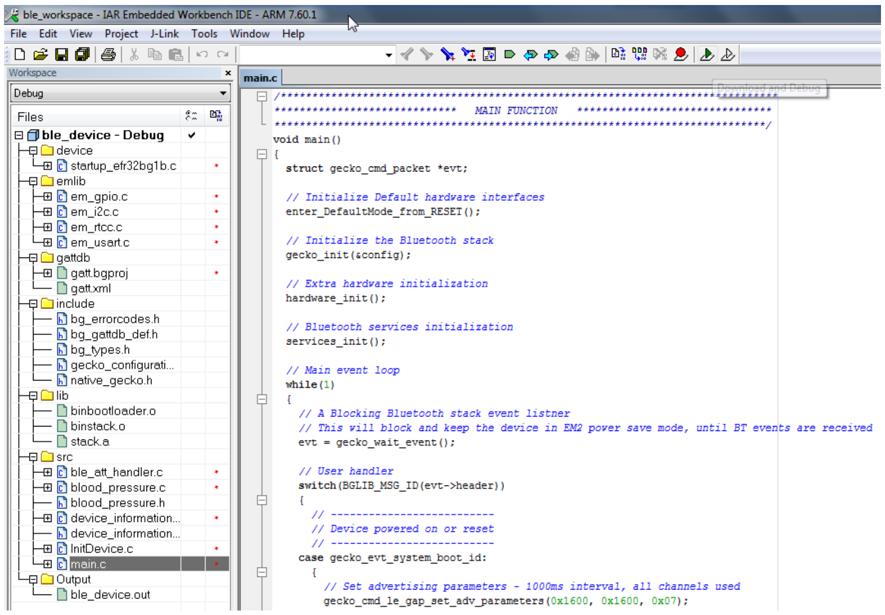
```
C:\SiliconLabs\SimplicityStudio\v5\developer\adapter_packs\ncp_commander
```

To find the standalone version on MAC, find Simplicity Studio, right-click and select Show Package Content. The NCP commander can be found under:

```
developer\adapter_packs\ncp_commander
```

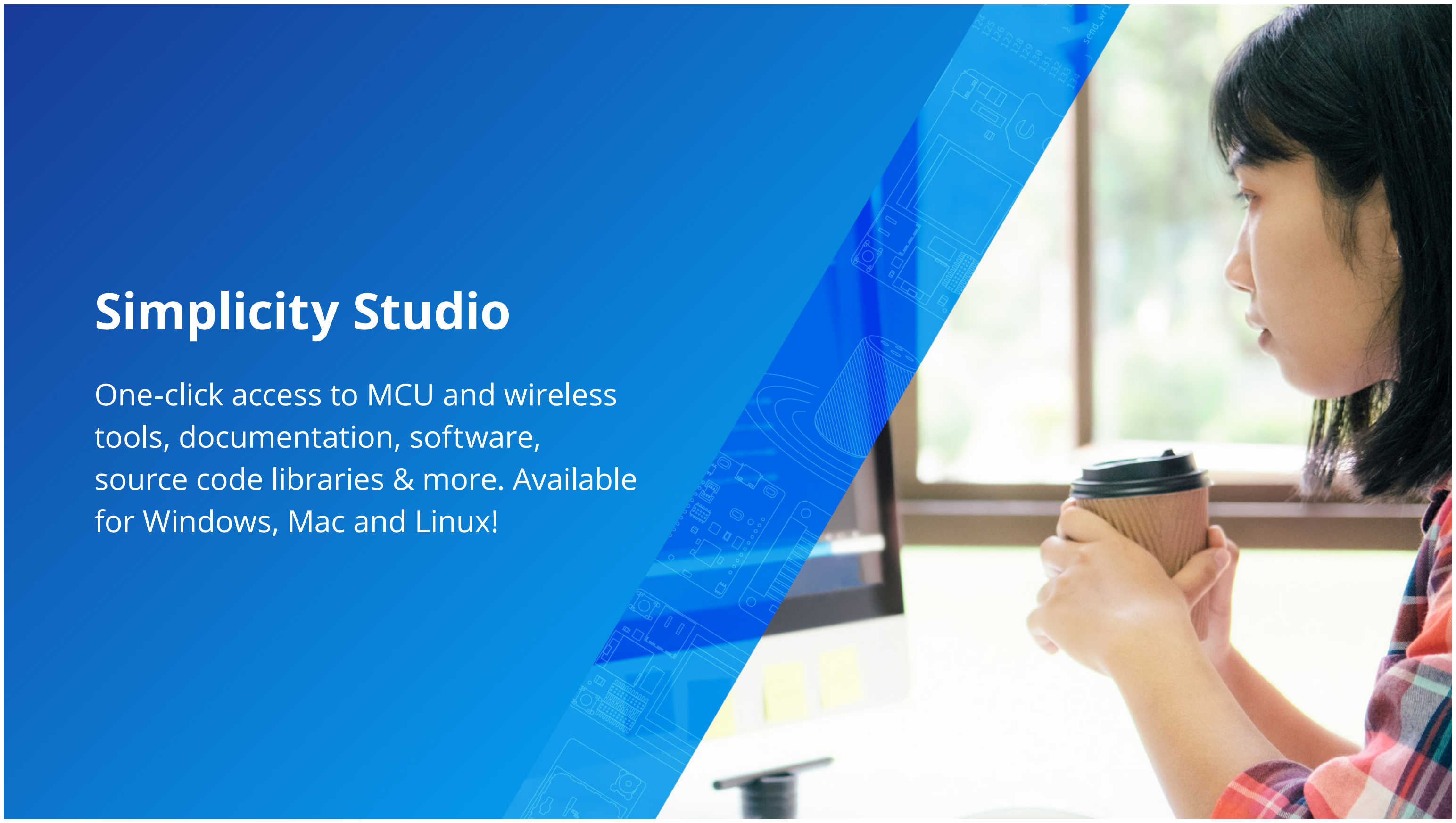
6.7 IAR Embedded Workbench

IAR’s Embedded Workbench can also be used as an IDE for developing and debugging Bluetooth applications. You must use the version of IAR that is compatible with the SDK version. See the SDK's release notes for compatible version information.



```
/*****************************************************************************
****************************   MAIN FUNCTION   ******************************
*****************************************************************************/
void main()
{
  struct gecko_cmd_packet *evt;

  // Initialize Default hardware interfaces
  enter_DefaultMode_from_RESET();

  // Initialize the Bluetooth stack
  gecko_init(&config);

  // Extra hardware initialization
  hardware_init();

  // Bluetooth services initialization
  services_init();

  // Main event loop
  while(1)
  {
    // A Blocking Bluetooth stack event listner
    // This will block and keep the device in EM2 power save mode, until BT events are received
    evt = gecko_wait_event();

    // User handler
    switch(BGLIB_MSG_ID(evt->header))
    {
      // -------------------------
      // Device powered on or reset
      // -------------------------
    case gecko_evt_system_boot_id:
      {
        // Set advertising parameters - 1000ms interval, all channels used
        gecko_cmd_le_gap_set_adv_parameters(0x1600, 0x1600, 0x07);
```

Figure 6.4. IAR Embedded Workbench


Simplicity Studio
One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

www.silabs.com