# UG350: Silicon Labs Coexistence Development Kit (SLWSTK-COEXBP)

The Coexistence Development Kit (SLWSTK-COEXBP) is a development kit designed by Silicon Labs to demonstrate coexistence between different radios transmitting in the 2.4 GHz ISM band. This document version pertains to revision 3.0 of the SLWSTK-COEXBP kit. SLWSTK-COEXBP also requires the EFR32MG Wireless Starter Kit (SLWSTK6000B), which contains the following devices:

- 3 x Wireless starter kit (WSTK) mainboard (BRD4001A)
- 3 x EFR32MG12 2.4 GHz 19 dBm radio board (BRD4161A)
- 3 x EFR32MG12 2.4 GHz 10 dBm radio board (BRD4162A)

A Logic Analyzer, to view the logic signals, is optional but recommended.

A Silicon Labs protocol stack is required to enable and run coexistence features. Any of the following stack versions can be used:

- Silicon Labs EmberZNet PRO (Zigbee) stack (v. 6.9.0 and up - note that configuration tools change in v. 7.0)
- Silicon Labs Bluetooth SDK (v. 3.1.0 and up)
- Silicon Labs OpenThread (OpenThread) SDK (v. 1.1.0.0 and up)
- Software required to run and view logic analyzer waveforms

The software images required to create a Zigbee, Bluetooth, or OpenThread network with coexistence features enabled must be built through sample applications in the respective stack, with added modifications discussed in this document.

The PTA Controller software required to emulate the Wi-Fi device behavior and exercise the coexistence features is available as a custom application. Instructions on how to download the application from GitHub are provided.

The following additional documents from Silicon Labs are useful resources for understanding protocol networks and coexistence, and will also be referred to in this user's guide:

- *QSG106: Zigbee EmberZNet PRO Quick-Start Guide for SDK 6.x and Lower*
- *QSG180: Zigbee EmberZNet PRO Quick-Start Guide for SDK 7.0 and Higher*
- *QSG169: Bluetooth® SDK v3.x Quick-Start Guide*
- *QSG170: Silicon Labs OpenThread Quick-Start Guide*
- *AN1017: Zigbee® and OpenThread Coexistence with Wi-Fi®*
- *AN1128: Bluetooth® Coexistence with Wi-Fi®*
- *UG103.17 Wi-Fi® Coexistence Fundamentals*

<div style="border:1px solid">

**KEY POINTS**

- Describes coexistence hardware and related software requirements.
- Provides step-by-step instructions for the installation and configuration of coexistence features in software.
- Details hardware included in the kit and how to use it.
- Explains additional software components/plugins and packages used for advanced testing.

</div>

# 1.  Introduction

*SLWSTK-COEXBP: Silicon Labs Coexistence Development Kit* is designed to demonstrate multiple radio coexistence features available as part of the SDK. These features enable radios in a single product (that is, Gateway, Hub, and so on) running on different protocols to arbitrate network traffic to avoid interference.

A Coexistence test configuration requires the following components:

- Three WSTK boards with either 19 or 10 dBm radio boards mounted, and loaded with the following software images:
  - PTA (Packet Traffic Arbitration) Controller: Acts as a Wi-Fi emulator and responds to activity on the device running the coexistence features (coexistence device).
  - Coexistence device application: Sample protocol application with coexistence features enabled.
  - End-node device application: Sample protocol application to help create a network and generate wireless traffic.
- Coexistence Backplane Board

A logic analyzer is optional but highly recommended to enable signal analysis and debugging. In our setup, we use a Saleae logic analyzer.

To set up the recommended coexistence test configuration, the PTA Controller and coexistence device both are plugged into headers on the Coexistence Backplane Board. The Coexistence Backplane Board has routes between those headers that enable connections between the two boards. The Coexistence Backplane Board also has routes from the headers to external test points. This allows probing the signals for debugging and evaluation purposes. The end-node device is not plugged into the Coexistence Backplane Board. The recommended configuration for using the Silicon Labs Coexistence Development Kit is shown in the following figure.
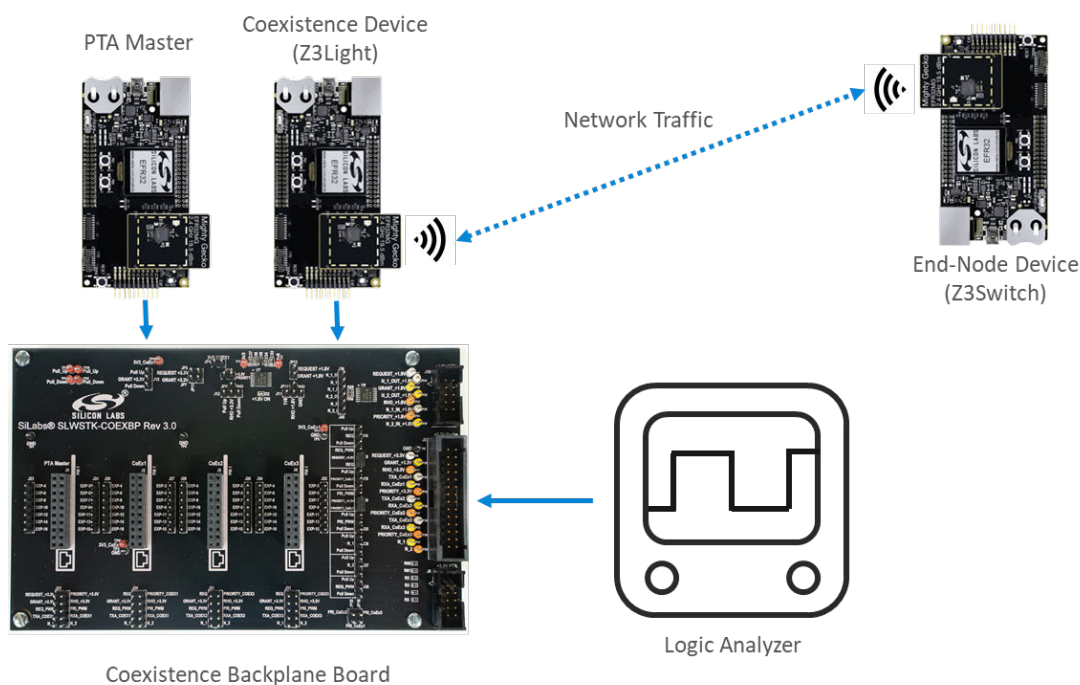


**Figure 1.1.  Typical Coexistence Test Configuration Using a Zigbee Example**
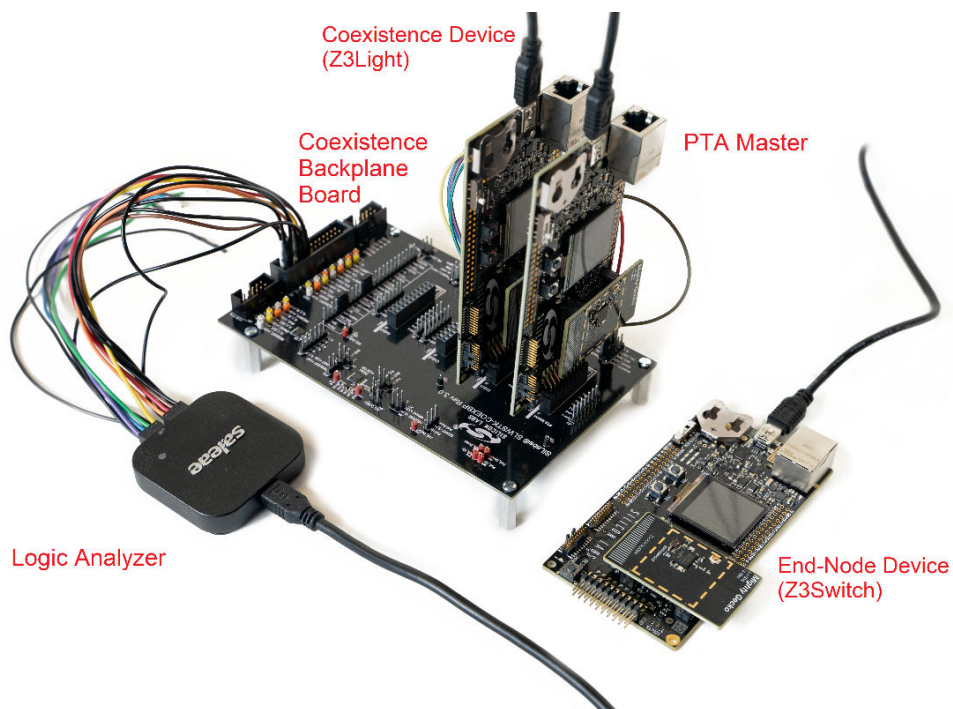
This configuration is shown in the following figures.

**Figure 1.2. Coexistence Test Setup Using a Zigbee Example (1)**
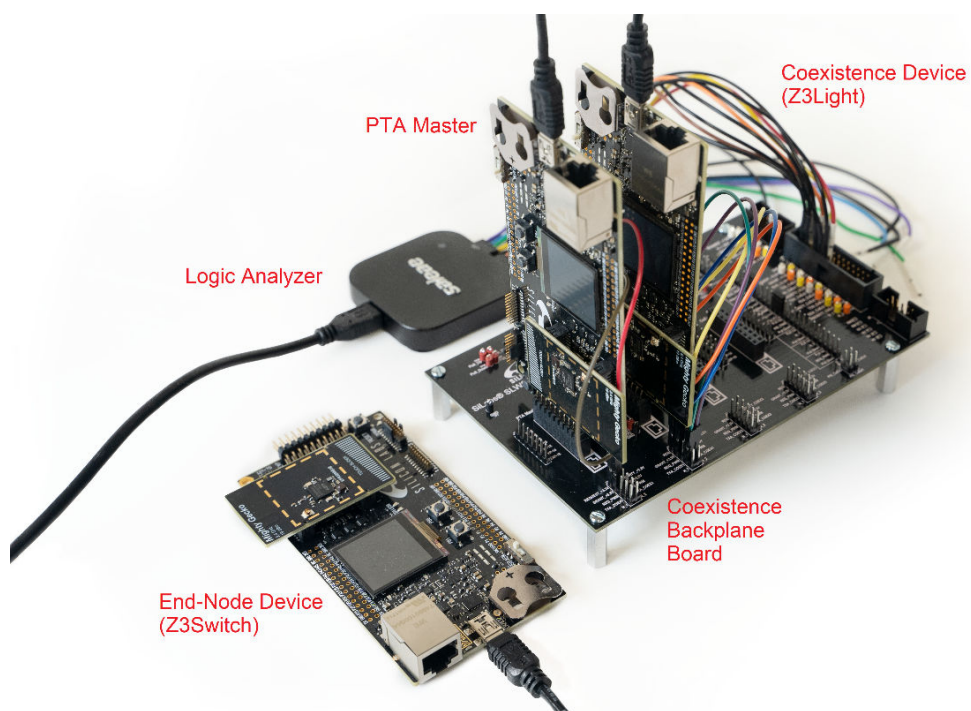


**Figure 1.3. Coexistence Test Setup Using a Zigbee Example (2)**

## 2. Theory of Operation

This section describes the steps required to bring up a network with coexistence features enabled. The section is divided into multiple subsections detailing each step required. The subsections are:

- 2.1 Bringing Up a Standalone Zigbee Network. Introduces building and flashing the Z3Light and Z3Switch applications, and sending commands over the network between them.
- 2.2 Adding the Throughput Library to Zigbee Devices. The applications are rebuilt with the throughput library, flashed, and network traffic created between them.
- 2.3 Adding the Coexistence Feature to Zigbee Devices. Coexistence features are added and the coexistence backplane board is introduced. Network traffic is created using the throughput library added in the previous step.
- 2.4 Using PTA Controller in the Test Setup. The PTA Controller application is built and flashed to a device, which then is introduced to the test setup as a coexistence controller device.
- 2.5 Using Coexistence Features in Bluetooth Devices. Coexistence features are set up on a Bluetooth network, in a procedure that builds on the instructions for setting up a Zigbee network.
- 2.6 Configuring Devices in Multi-Radio Mode. The coexistence board and applications are configured for multi-radio mode.
- 2.7 Replacing PTA Controller with an External Wi-Fi Device. The PTA Controller device is replaced with an external Wi-Fi chipset.

**Note:** This section walks through the steps required to set up coexistence on EFR32MG12-based devices. Refer to section 3.1 Jumper Wire Configurations for Different Boards/Devices regarding connection and wiring information for other compatible devices.

**Note:** Zigbee EmberZNet SDK introduced a new component-based architecture, along with a Project Configurator and other tools to replace AppBuilder and plugin configuration. In general, the new software components are comparable to the plugins. When applicable, instructions for both version 7.0 and higher and 6.10.x and lower are provided. For more information, see *AN1301: Transitioning from Zigbee EmberZNet SDK 6.x to SDK 7.x.*

### 2.1 Bringing Up a Standalone Zigbee Network

The first step in setting up for coexistence testing is understanding how to set up a Zigbee network between two devices. For this purpose, we recommend following the steps in the applicable quick-start guide for your version to build, flash, and verify network commands between the Z3Light and Z3Switch example applications. These applications will be modified and used in subsequent steps. This allows you to become familiar with application development using the Zigbee stack. It is also important to be familiar with the steps in the quick-start guide as the document will be referred to extensively in the following steps. Quick-start guides can be found through the list of documents on the Simplicity Studio Launcher perspective, and on the Silicon Labs website under Zigbee documentation. Direct links to the documents follow:

https://www.silabs.com/documents/public/quick-start-guides/qsg106-efr32-zigbee-pro.pdf

https://www.silabs.com/documents/public/quick-start-guides/qsg180-zigbee-emberznet-7x-quick-start-guide.pdf

### 2.2 Adding the Throughput Library to Zigbee Devices

This step explains how to modify the example applications to add the throughput library. The throughput library is an additional component/plugin available in the EmberZNet PRO stack that allows transmitting data between devices at controllable speeds. It is useful for the evaluation of network quality and connection stability. More information regarding the throughput library can be found in section 4. Throughput Library.

## 2.2.1 Rebuilding and Flashing the Images

If you have not already done so in the previous section, follow the steps in the applicable quick-start guide to start the Z3Light and Z3Switch example applications. Otherwise return to the Simplicity IDE perspective for one of the examples. Customize the examples as described below, then follow the steps in the applicable quick-start guide to build and flash them to your devices.

**In SDK 7.0 and higher:**

If the *<project>.slcp* file is not open, double-click it in the Project Explorer view to open it. Go to the Software Components tab and enter the keyword 'throughput' to search for the throughput component. Install the component. Repeat for the other example application.
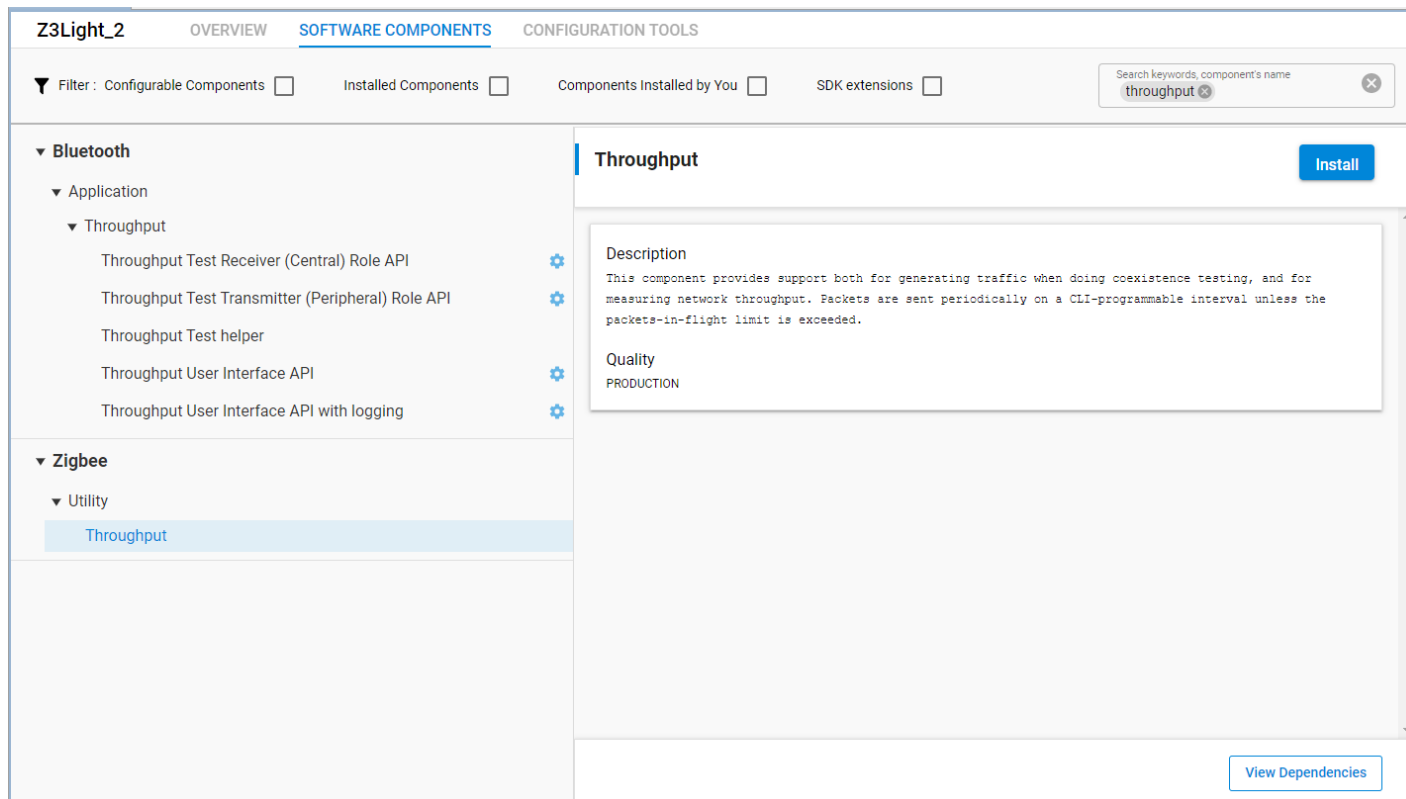


**In SDK 6.10.x and lower:**

If the *<project>..isc* file is not open, double-click it in the Project Explorer view to open it. Do not generate the application yet. Go to the Plugins tab and enter the keyword "throughput" to search for the throughput library plugin. Check the checkbox next to the plugin name to enable the plugin. Repeat for the other example application.
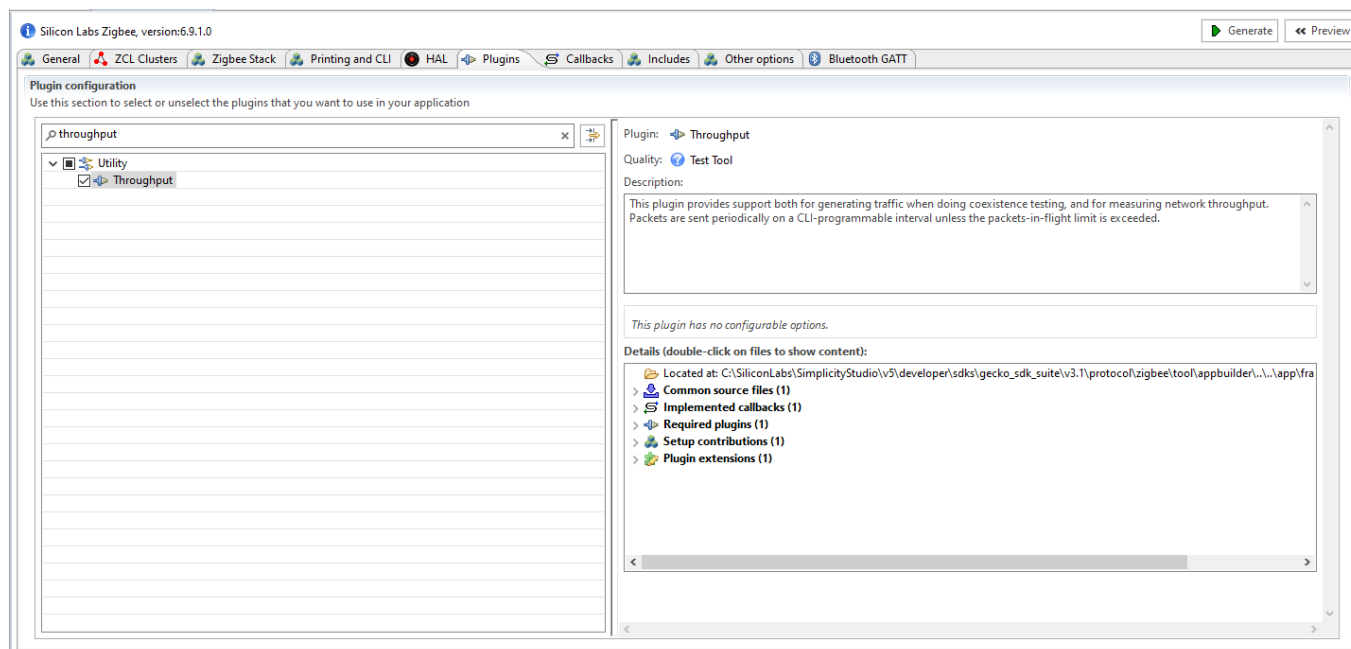
### 2.2.2 Using the Throughput Library to Send Zigbee Traffic

1. From Simplicity Studio, access the CLI consoles of the Z3Light and Z3Switch WSTKs. Refer to the applicable quick-start guide for instructions on opening CLI consoles.
2. In the console of the Z3Switch example, send the command `info` to display network information regarding the device. Note the Node ID in the command's response.
3. In the console of the Z3Light, send the following command

```
plugin throughput set-all <0xNodeID> 10 10 127 1 0 100000
```

where NodeID is the node ID of the Z3Switch noted in step 2 above, typed as a hex number. This configures the throughput settings.

4. Still in the console of the Z3Light, send the following command:

```
plugin throughput start
```

A series of 10 packets is then sent from the Z3Light to the Z3Switch, generating Zigbee traffic. The Z3Switch console also indicates packets being received.

More information on the commands used in step 3 and 4 as well as other features of the throughput library can be found in section 4. Throughput Library .

### 2.3 Adding the Coexistence Feature to Zigbee Devices

This step describes the procedure to enable and configure coexistence features in the example applications. This section uses the Z3Light application as the example, however the same procedure can be used to enable coexistence on any application. The coexistence features are contained in an additional component/plugin available in the EmberZNet PRO stack. More information regarding the coexistence component/plugin can be found in *AN1017: Zigbee® and OpenThread Coexistence with Wi-Fi*.

Enabling the FEM driver component/plugin is also recommended. This provides visibility into the state of the radio for debugging purposes and monitoring radio activity.

**Note:** This procedure assumes you have created, built, flashed, and verified example applications with the Throughput library as described in section 2.2 Adding the Throughput Library to Zigbee Devices.

### 2.3.1 Rebuilding and Flashing the Images

Customize the examples as described below, then follow the steps in the applicable quick-start guide to build and flash them to your devices.
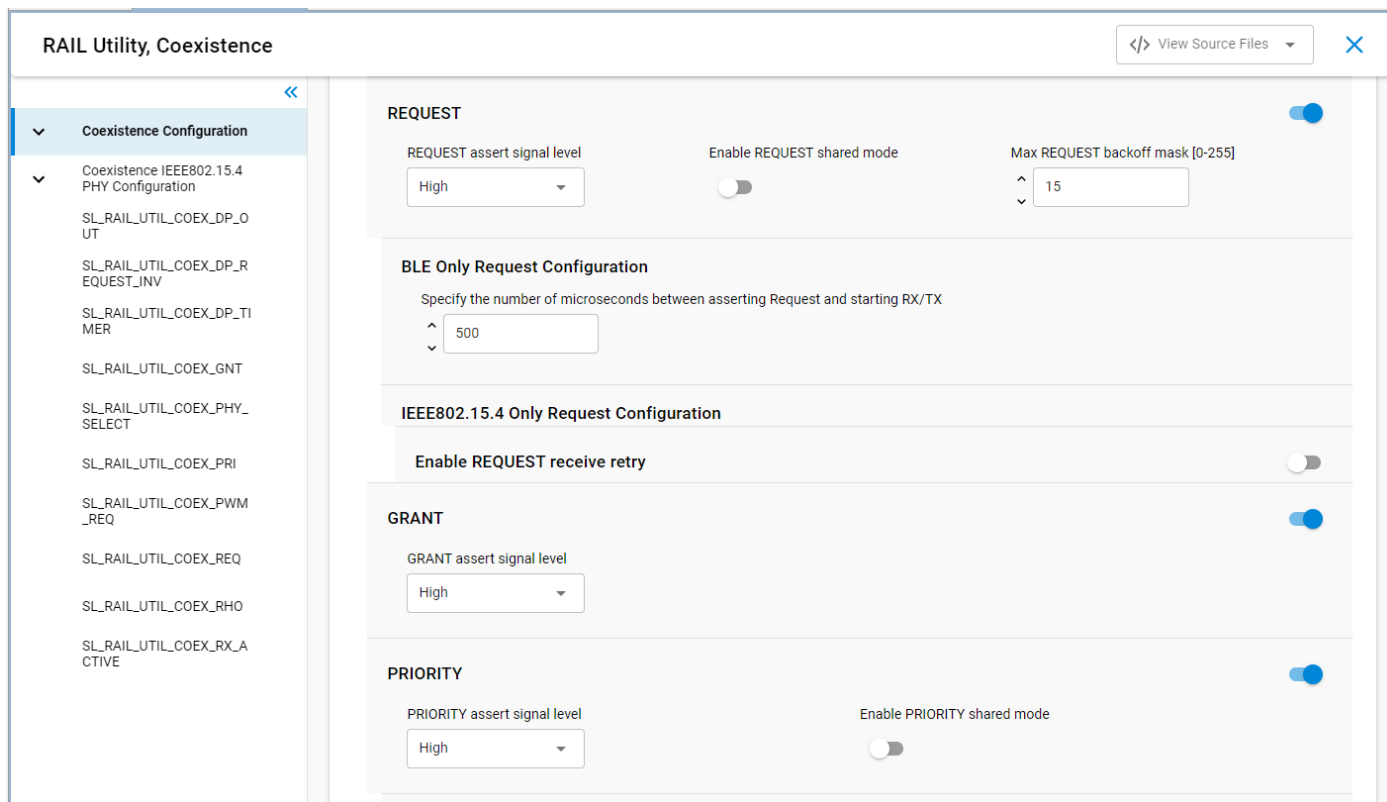
**In SDK 7.0 and higher:**

1. Open the Simplicity IDE perspective to access the project files.
2. In the Project Explorer view, open the *<project_name>.slcp* file. If you have just created the project, the file may already be open.
3. Click the Software Components tab, and type 'coexistence' in the search field. Select the **RAIL Utility, Coexistence** component.
4. Click [**Install**] to add the component to the project. A [**Configure**] control is now available.



5. Click [**Configure**] to open the Component Editor.

6. By default, most coexistence options on the Coexistence Configuration card are disabled. Disable anything that is enabled, and then enable the following:
   - **Request Signal**, with all options enabled except Shared mode.
   - **Grant Signal**
   - **Priority Signal**, with all options enabled except Shared mode.



Changes are automatically saved, as shown at the top of the Component Editor.

7. A series of cards for different coexistence features are provided after the configuration card. Configure GPIO routes on the following cards:
   - **SL_RAIL_UTIL_COEX_GNT** (GRANT) = PC9
   - **SL_RAIL_UTIL_COEX_PRI** (PRIORITY) = PD12
   - **SL_RAIL_UTIL_COEX_REQ** (REQUEST) = PC10
   - **SL_RAIL_UTIL_COEX_RHO** (Radio Hold-off) = None

8. Close the Component Editor.

9. Enabling the FEM driver component is also recommended, to provide visibility into the state of the radio. On the Software Components tab, search for 'FEM'. Select the **Radio Utility, FEM** component.

10. Click [**Install**], then click [**Configure**] to open the Component Editor.

11. Configure the component as follows:
- In the **FEM Configuration** card, enable **RX Mode** and **TX Mode**.



- In the **SL_FEM_UTIL_RX** card, set Selected Module to PRS Channel CH6 and pin PD11.



- In the **SL_FEM_UTIL_TX** card, set Selected Module to PRS Channel CH5 and pin PD10.
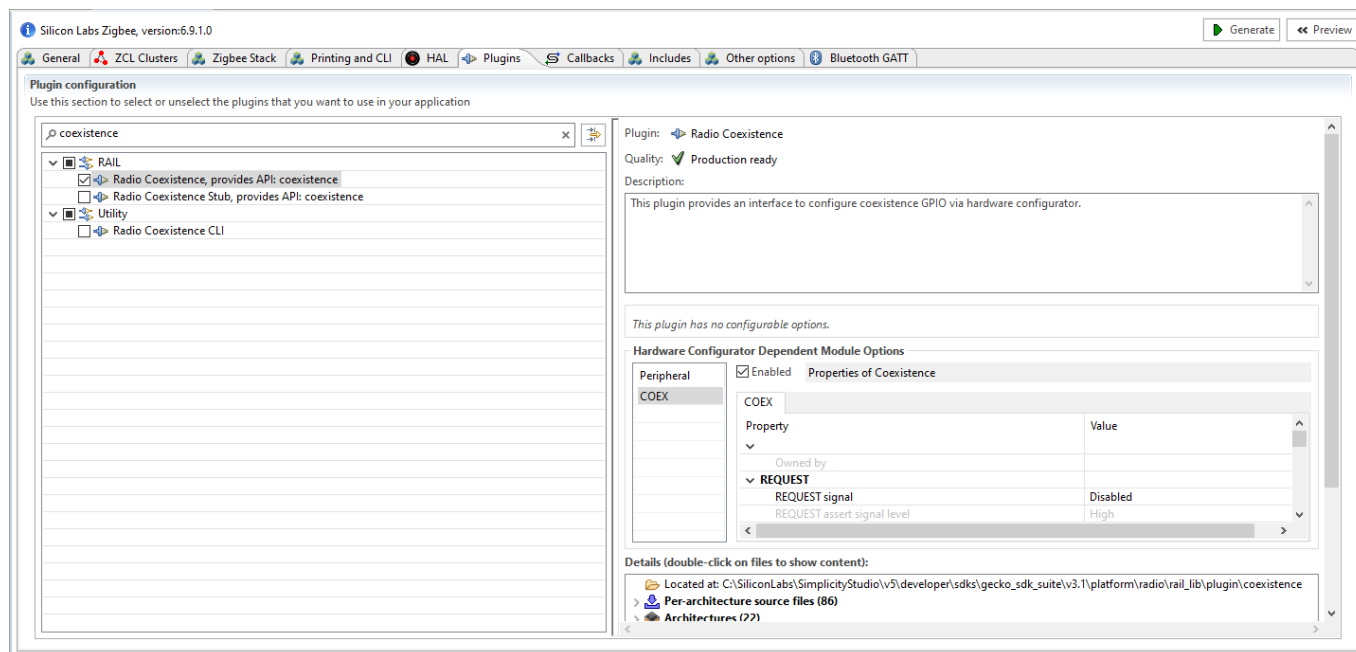


**In SDK 6.10.x and lower:**
1. Return to the Simplicity IDE perspective for the Z3Light sample application .isc file.
2. Go to the Plugins tab and search for the keyword 'coexistence' to find the coexistence configuration plugin.

3. Check the checkbox next to the plugin name to enable the plugin.



4. Uncheck the checkbox next to the Stub plugin to disable Stub functions.

5. In the menu on the right, make sure the Enabled box next to **Properties of Coexistence** is checked.

6. Modify the fields in the menu on the right as follows:

- REQUEST Signal: PC10
- GRANT Signal: PC9
- PRIORITY Signal: PD12
- RHO Signal: Disabled

   Leave all other settings with their default values.

7. Still in the Plugins tab, search for keyword "fem" to bring the FEM driver plugin.

8. Check the checkbox next to the plugin name to enable the plugin.



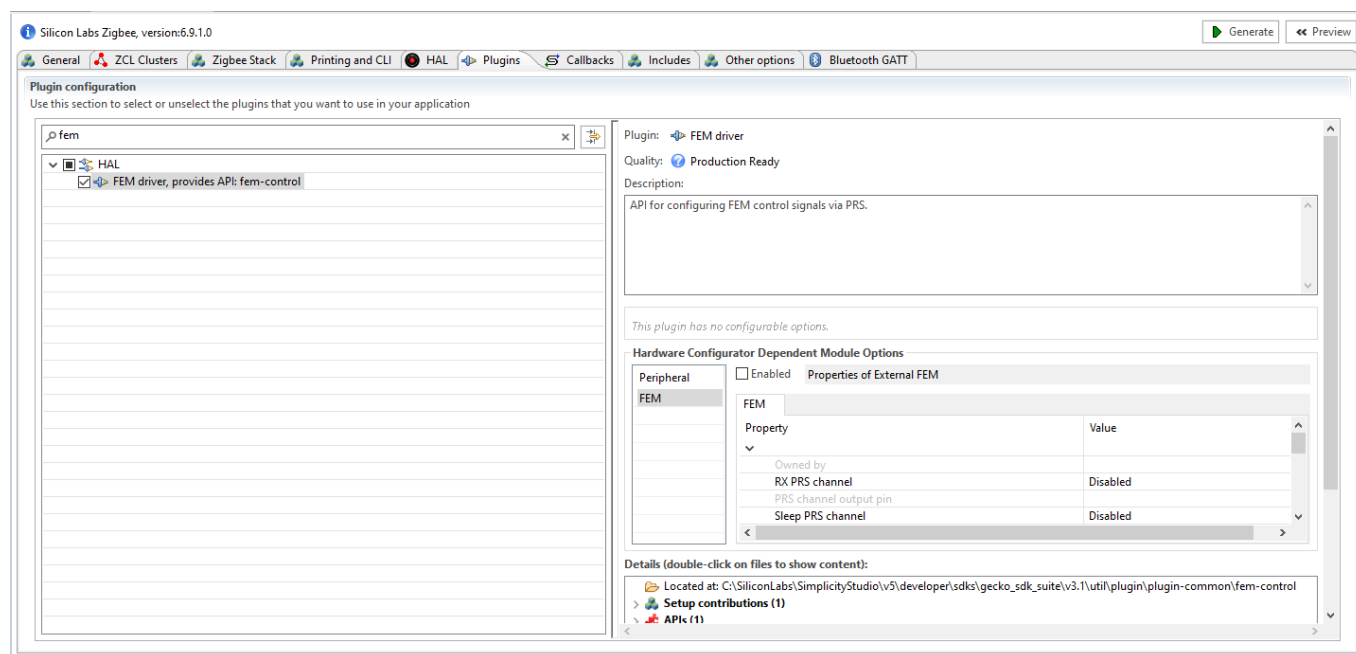9. In the menu on the right, make sure the Enabled box next to **Properties of External FEM** is checked.

10. Modify the fields in the menu on the right as follows:
  - RX PRS channel: CH6
  - PRS channel 6 output pin: PD11
  - TX PRS channel: CH5
  - PRS channel 5 output pin: PD10
  - Enable RX mode: True
  - Enable TX mode: True

Leave all other settings with their default values.

### 2.3.2 Setting up the Coexistence Backplane Board

1. Add the appropriate jumper wires and jumpers as shown in the following figure. This sets the GRANT line to always asserted, and allows the Z3Light to exercise the coexistence signals.
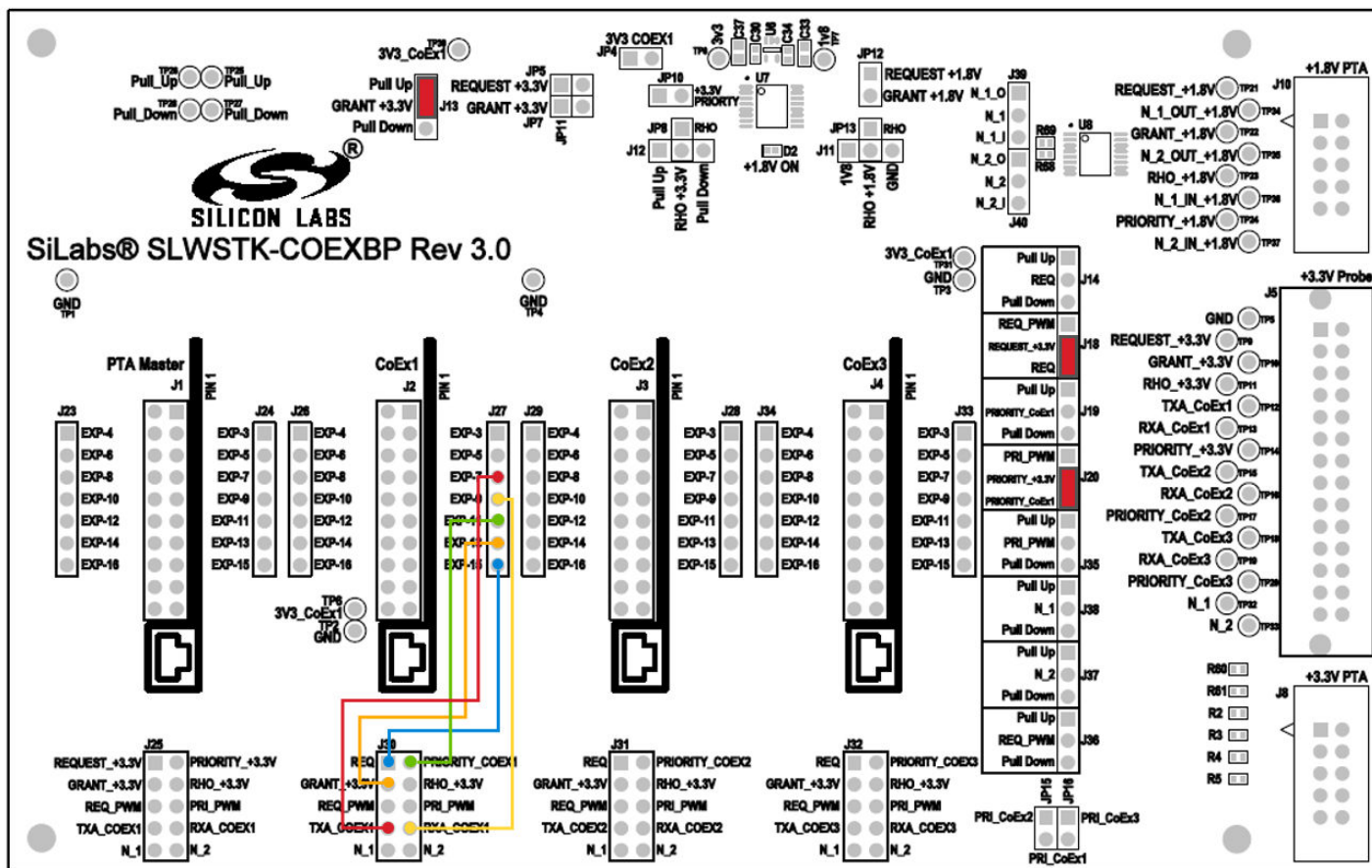


**Figure 2.1. Jumper Configurations for Coexistence Testing**

2. Plug the Z3Light board into the CoEx1 position (J2) in the Coexistence Backplane board. Follow the silk label for proper orientation.

3. If using a logic analyzer, probe the test points or pins in header J5 (+3.3V probe header) on the right side to access the following signals:
  - REQUEST_+3.3V – TP9 or J5 pin 2
  - GRANT_+3.3V – TP10 or J5 pin 4
  - PRIORITY_+3.3V – TP14 or J5 pin 12
  - TXA_CoEx1 – TP12 or J5 pin 8
  - RXA_CoEx1 – TP13 or J5 pin 10

### 2.3.3 Using the Throughput Library to Send Zigbee Traffic

1. From Simplicity Studio, access the CLI consoles of the Z3Light and Z3Switch WSTKs.
2. In the console of the Z3Switch example, send the command `info` to display network information regarding the device. Note the Node ID in the command's response.
3. In the console of the Z3Light, send the following command

```
plugin throughput set-all <0xNodeID> 10 10 127 1 0 100000
```

where NodeID is the node ID of the Z3Switch noted in step 2 above, typed as a hex number. This configures the throughput settings.
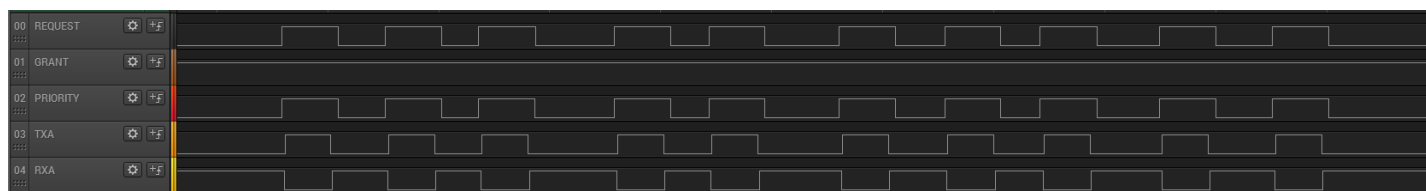
4. Still in the console of the Z3Light, send the following command:

```
plugin throughput start
```

A series of 10 packets is then sent from the Z3Light to the Z3Switch, generating Zigbee traffic. The Z3Switch console also indicates packets being received.

More information on the commands used in step 3 and 4 as well as other features of the throughput library can be found in section 4. Throughput Library.

5. Activity corresponding to sending 10 packets can also be observed on the logic analyzer. It will look similar to the following capture.



**Figure 2.2. Logic Analyzer Result for Coexistence Testing**

REQUEST and PRIORITY signals are asserted 10 times during the throughput run, indicating 10 attempts to send the messages. In addition, TXA is also asserted within these REQUEST cycles, indicating the radio was transmitting. GRANT signal is always asserted due to the configuration of the board. The RXA signal is de-asserted when the radio is transmitting, otherwise it remains asserted, indicating the radio is in receive mode.

### 2.4 Using PTA Controller in the Test Setup

The PTA Controller application is a custom application. It is designed to assist in evaluating the coexistence features available in the stack by acting as a Wi-Fi controller device emulator. The PTA Controller only emulates the GPIO behaviors and cannot generate actual Wi-Fi traffic. More information regarding the features of the PTA Controller can be found in section 5. PTA Controller Application .

**Note:** This procedure uses the Z3Light and Z3Switch devices created and installed with the Coexistence Backplane board in the previous procedure.

### 2.4.1 Obtaining the PTA Controller Application

The PTA Controller application is available for download on GitHub. Use the following link to obtain the binary files.
https://github.com/SiliconLabs/pta_controller

Follow the steps in the applicable quick-start guide to flash the PTA Controller application onto one of the boards.

#### 2.4.2 Setting up the PTA Controller Device with the Coexistence Backplane Board

1. Remove the jumper from J13 (added to assert GRANT line in Section 2.3.2 Setting up the Coexistence Backplane Board) and add additional jumper wires required to communicate with the PTA Controller, as shown in the following diagram:
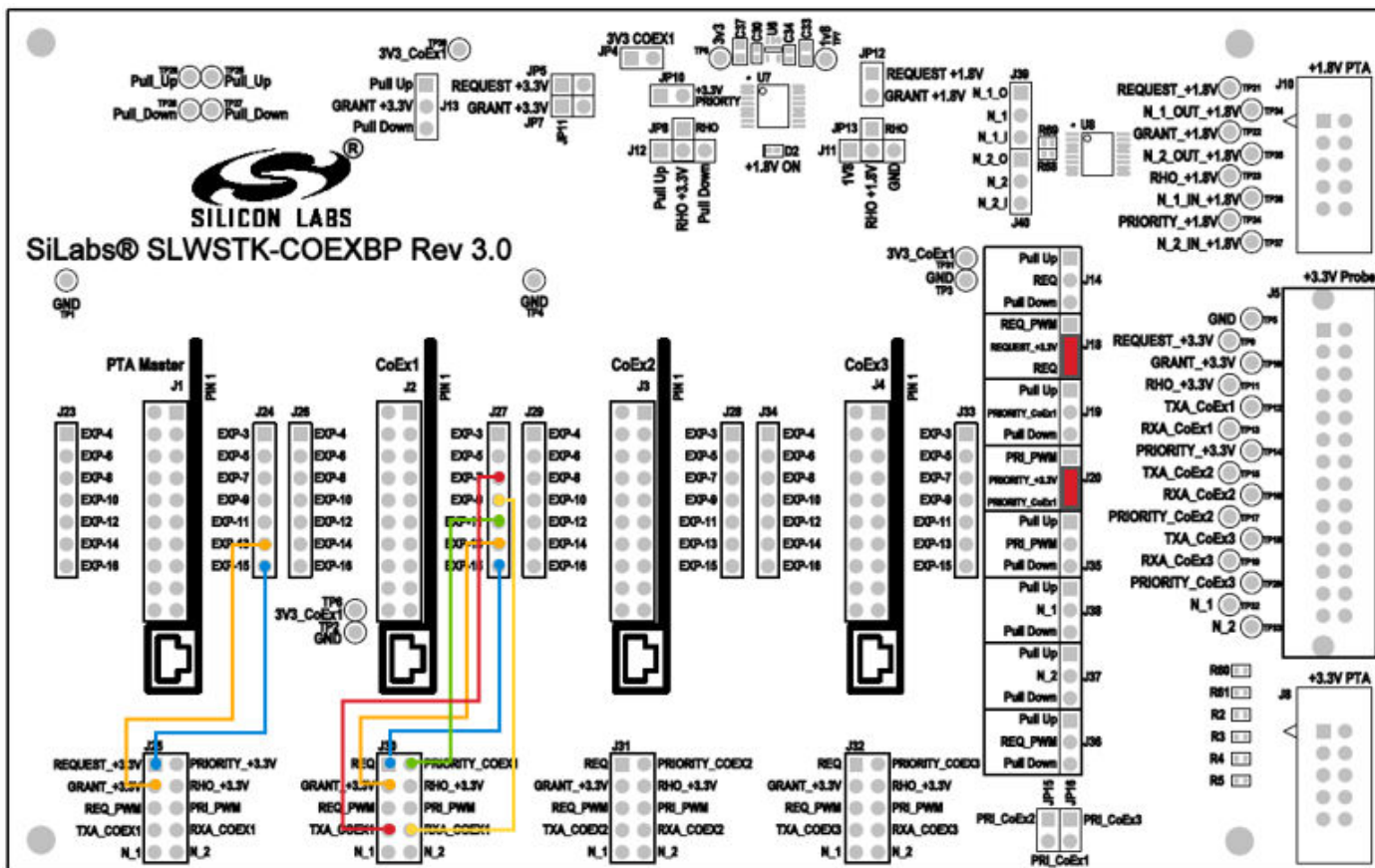


**Figure 2.3. Jumper Configuration for Coexistence Testing with PTA Controller Device**

2. Plug PTA Controller device into dedicated header J1, on the left side of the board. Make sure the board is oriented according to the silk near the header.

#### 2.4.3 Configuring the PTA Controller Application to Work with Z3Light

The PTA Controller application features run-time configurable options to configure the behavior. By default, all additional features are turned off, and all signals are assumed active High.

Therefore, the PTA Controller will work with the Z3Light device as is, without needing additional configurations. Details regarding additional configurations can be found in section 5. PTA Controller Application.

#### 2.4.4  Using the Throughput Library to Send Zigbee Traffic

1. From Simplicity Studio, access the CLI consoles of the Z3Light and Z3Switch WSTKs.
2. In the console of the Z3Switch example, send the command `info` to display network information regarding the device. Note the Node ID in the command's response.
3. In the console of the Z3Light, send the following command:

```
plugin throughput set-all <0xNodeID> 10 10 127 1 0 100000
```

where NodeID is the node ID of the Z3Switch noted in step 2 above, typed as a hex number. This configures the throughput settings.
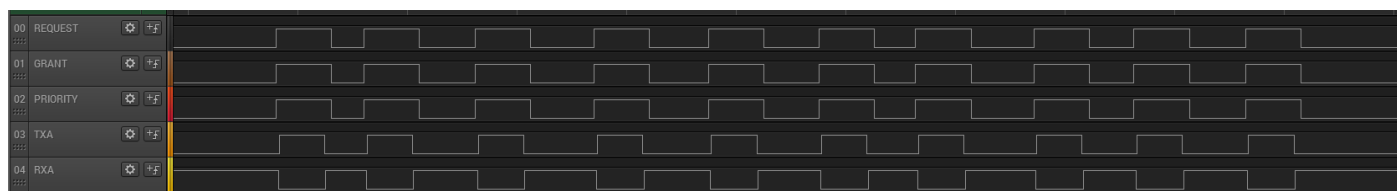
4. Still in the console of the Z3Light, send the following command:

```
plugin throughput start
```

A series of 10 packets is then sent from the Z3Light to the Z3Switch, generating Zigbee traffic. The Z3Switch console also indicates packets being received.

More information on the commands used in step 3 and 4 as well as other features of the throughput library can be found in section 4. Throughput Library.

5. Activity corresponding to sending 10 packets can also be observed on the logic analyzer. It will look similar to the following capture.



**Figure 2.4.  Logic Capture for Coexistence Testing with the PTA Controller Device**

REQUEST and PRIORITY signals are asserted 10 times during the throughput run, indicating 10 attempts to send the messages. Note here that the GRANT signal controlled by PTA Controller is also asserted and de-asserted with REQUEST. In default settings, the PTA Controller is configured to always GRANT when the peripheral device REQUESTs. In addition, TXA is also asserted within these REQUEST cycles, indicating the radio was transmitting. The RXA signal is de-asserted when the radio is transmitting, otherwise it remains asserted, indicating the radio is in receive mode.

### 2.5  Using Coexistence Features in Bluetooth Devices

Coexistence features are also available in the Bluetooth SDK, and can be evaluated using the same hardware setup explained previously for Zigbee. For the following instructions, the coexistence backplane board must be configured as explained in section 2.4.3 Configuring the PTA Controller Application to Work with Z3Light, with the PTA Controller device and coexistence device plugged in and connected to the Coexistence backplane board.
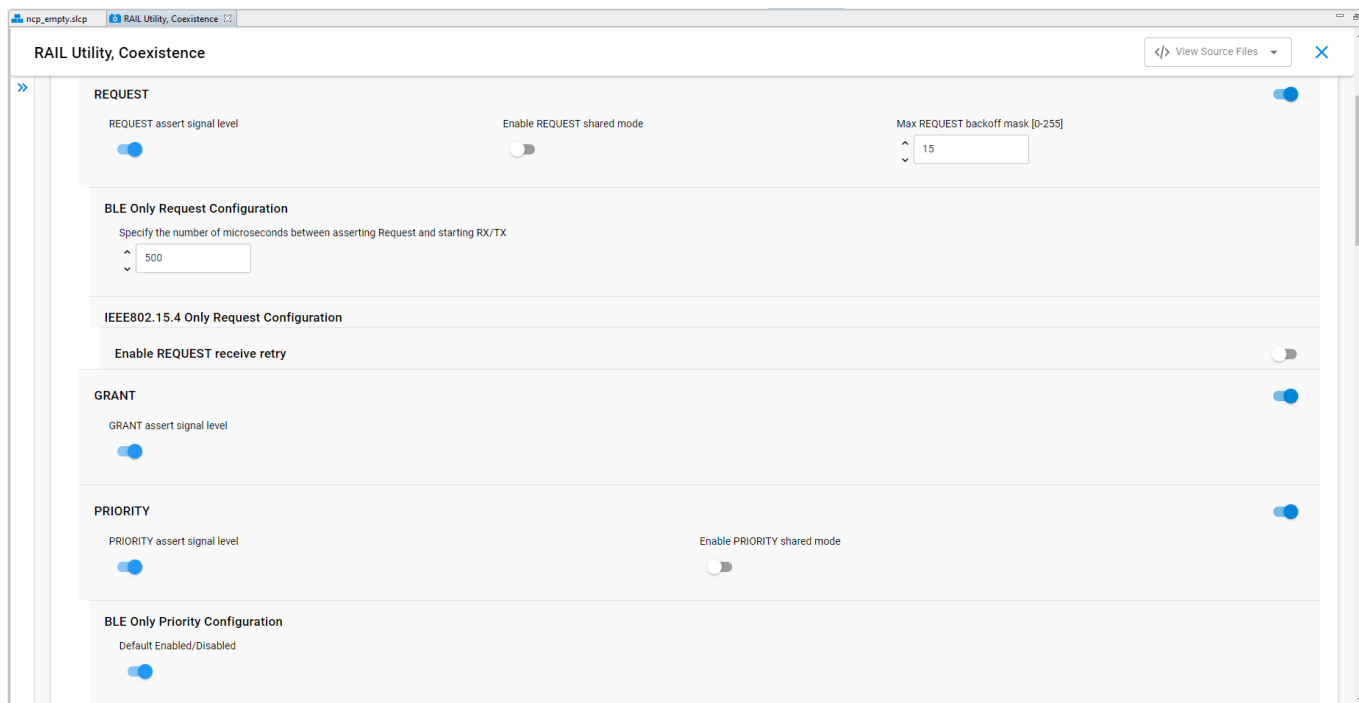
#### 2.5.1  Setting up a Bluetooth Sample Application

The Silicon Labs Bluetooth SDK contains many useful examples to test and run Bluetooth. For the purposes of demonstrating coexistence features in Bluetooth, the "NCP Empty" example will be used. Refer to *QSG169: Bluetooth® SDK v3.x Quick-Start Guide* for instructions on how to create a sample ncp-empty application and use it to advertise. For the purpose of generating Bluetooth PTA traffic in this test, only one Bluetooth device will be required.

### 2.5.2  Adding Coexistence Features to a Bluetooth Sample Application

Once a sample Bluetooth application is set up, coexistence features can be added to the application using the following steps:

1. Open the Simplicity IDE perspective to access the project files.
2. In the Project Explorer view, open the *<project_name>.slcp* file (for example, *ncp_empty.slcp*). If you have just created the project, the file may already be open.
3. Click the Software Components tab, and type 'coexistence' in the search field. Select the **RAIL Utility, Coexistence** component.
4. Click [**Install**] to add the component to the project. A [**Configure**] control is now available.
5. Click [**Configure**] to open the Component Editor.
6. By default, most coexistence options on the Coexistence Configuration card are disabled. Disable anything that is enabled, and then enable the following:
   - **Request Signal**, with all options enabled except Shared mode.
   - **Grant Signal**
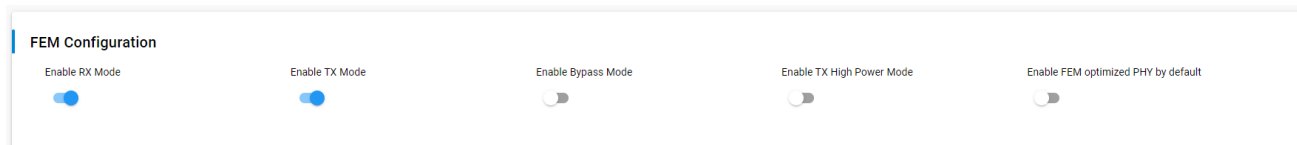   - **Priority Signal**, with all options enabled except Shared mode.



Changes are automatically saved, as shown at the top of the Component Editor.

7. A series of cards for different coexistence features are provided after the configuration card. Configure GPIO routes on the following cards:
   - **SL_RAIL_UTIL_COEX_GNT** (GRANT) = PC9
   - **SL_RAIL_UTIL_COEX_PRI** (PRIORITY) = PD12
   - **SL_RAIL_UTIL_COEX_REQ** (REQUEST) = PC10
8. Close the Component Editor.
9. Enabling the FEM driver component is also recommended, to provide visibility into the state of the radio. On the Software Components tab, search for 'FEM'. Select the **Radio Utility, FEM** component.
10. Click [**Install**], then click [**Configure**] to open the Component Editor.
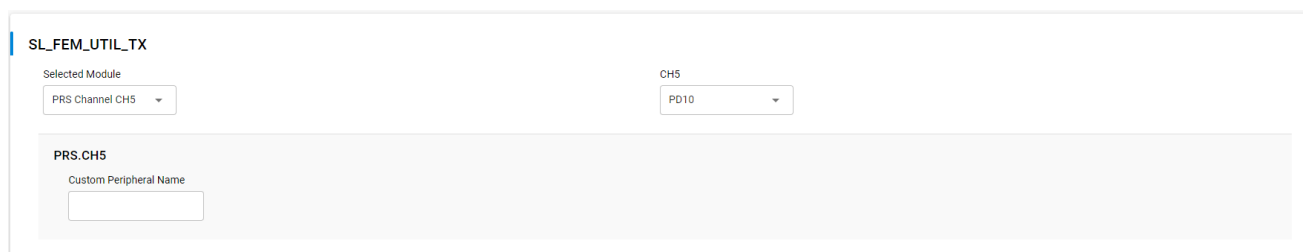
11. Configure the component as follows:

- In the **FEM Configuration** card, enable **RX Mode** and **TX Mode**.

> **FEM Configuration**
>
> Enable RX Mode      Enable TX Mode      Enable Bypass Mode      Enable TX High Power Mode      Enable FEM optimized PHY by default

- In the **SL_FEM_UTIL_RX** card, set Selected Module to PRS Channel CH6 and pin PD11.

> **SL_FEM_UTIL_RX**
>
> Selected Module      CH6
> PRS Channel CH6      PD11
>
> **PRS.CH6**
> Custom Peripheral Name

- In the **SL_FEM_UTIL_TX** card, set Selected Module to PRS Channel 5 and pin PD10.

> **SL_FEM_UTIL_TX**
>
> Selected Module      CH5
> PRS Channel CH5      PD10
>
> **PRS.CH5**
> Custom Peripheral Name

12. Rebuild and flash the new image to the coexistence device, and enable advertising as described in *QSG169: Bluetooth® SDK v3.x Quick-Start Guide*.

The following logic capture shows an example of the capture during Bluetooth advertising events:

## 2.6  Configuring Devices in Multi-Radio Mode

Depending on the system requirements, multiple coexistence-enabled devices may need to coordinate network activity with Wi-Fi device using PTA. This can be achieved in coexistence features by enabling Shared mode. Shared mode allows multiple devices to operate on a single coexistence line by configuring the GPIO to open-drain or open-source, with external pull resistors. The coexistence backplane board already contains pull resistors that can be enabled through jumpers. Note that, for this section, an additional WSTK board with BRD4161 or BRD4162 radio board is required.

The following steps explain how to set up the coexistence board and sample applications for multi-radio configuration:

1. In a Zigbee sample application, after setting up all coexistence options as explained in section 2.3.1 Rebuilding and Flashing the Images, also enable Shared mode under Request and Priority options.
2. Rebuild and flash the new image to a WSTK device with BRD4161 radio board.
3. In a Bluetooth sample application, after setting up all coexistence options as explained in section 2.5.2 Adding Coexistence Features to a Bluetooth Sample Application, also enable Shared mode under Request and Priority options.
4. Rebuild and flash the new image to a different WSTK device with BRD4161 radio board.
5. Configure the Coexistence backplane board as shown in the following figure. Note that there are three WSTK boards now plugged into the Coexistence backplane board.
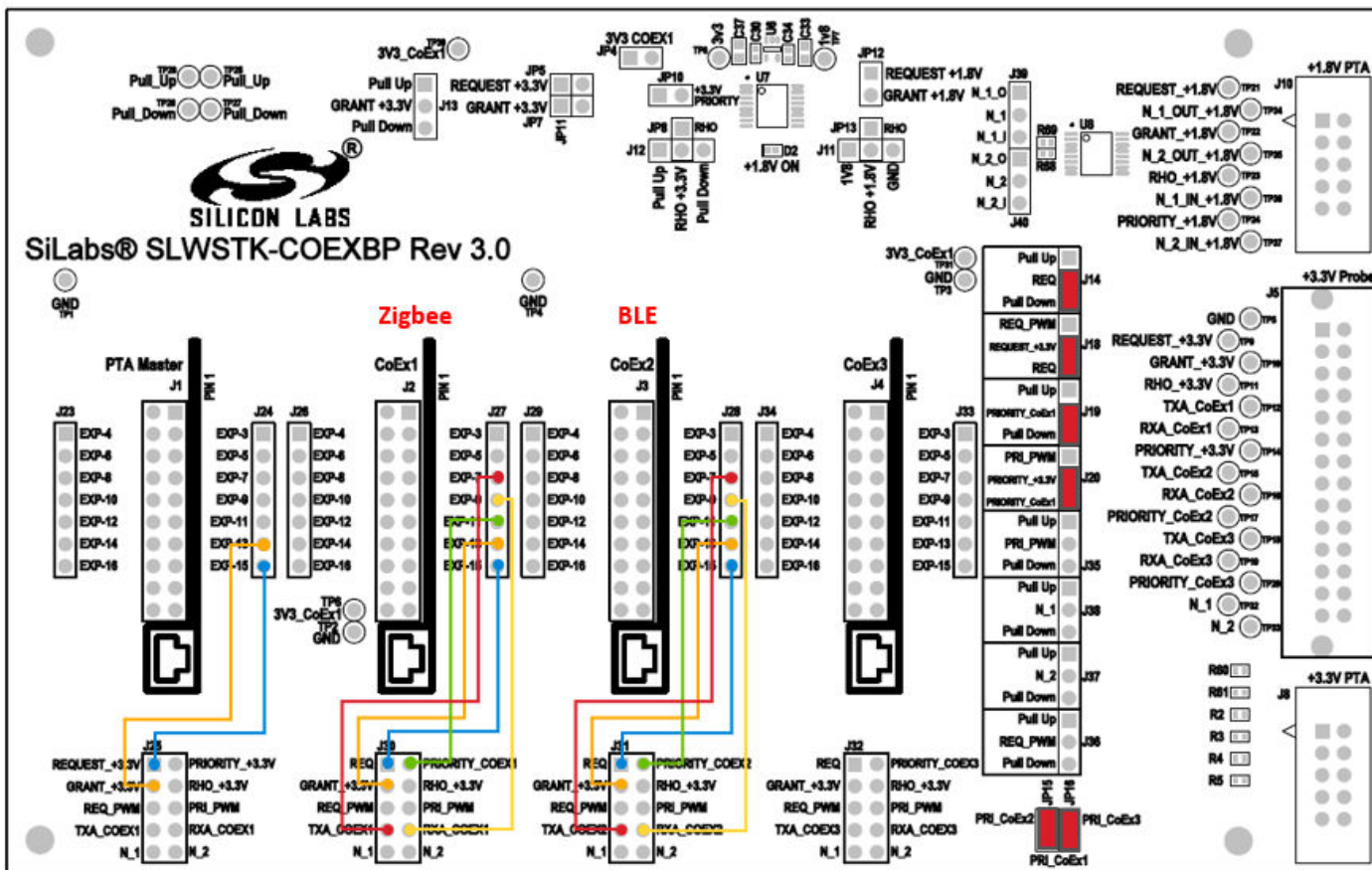


6. Enable advertising on the Bluetooth coexistence device, and generate throughput activity on the Zigbee coexistence device.

The following image shows an example of the logic capture when both Zigbee and Bluetooth are communicating on the same coexistence lines. Note that only one or the other radio is using the coexistence lines at a time. This is developed into the Shared mode options, as a radio must check the coexistence lines to make sure they are available, before attempting to use them.
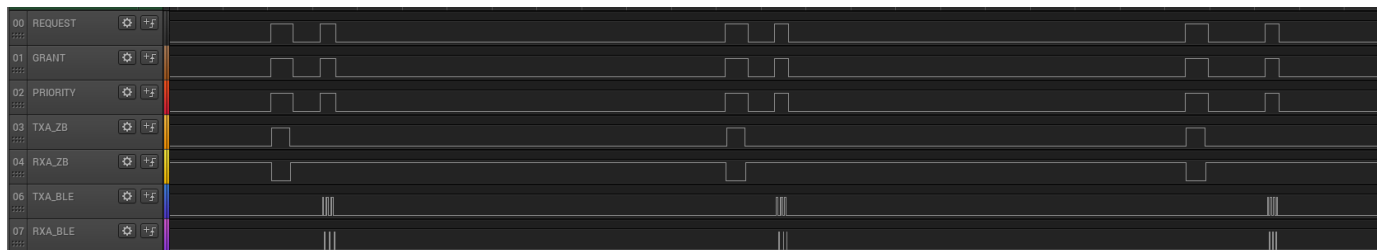
## 2.7  Replacing PTA Controller with an External Wi-Fi Device

Once evaluation has been completed using the PTA Controller application, the next step is to use an actual external Wi-Fi chipset to interface with the coexistence features in the Z3Light device. The following steps detail how this can be achieved.

1. Remove the PTA Controller device.
2. Connect the external Wi-Fi device coexistence signals to the appropriate header.
   - If using a +3.3V signals Wi-Fi device, use header J8.
   - If using a +1.8V signals Wi-Fi device, use header J10.

   Refer to section 3. Coexistence Backplane Board for more detail.
3. The PTA signals are routed from the CoEx component/plugin headers to the probe headers J8 and J10 when configured to be used, so the Wi-Fi device should be able to receive and respond to the signal states.

# 3. Coexistence Backplane Board

This section gives a detailed explanation of the Coexistence Backplane Board. The board is compatible with all EFR32MG1x series devices. Table 3.1 Jumper Wire Recommendations for PTA on Different Devices on page 20 shows recommended PTA pin selections for different supported boards.

**Note:** Due to GPIO limitations on some boards, GPIOs routed to peripherals on the WSTK may need to be used for PTA functions. This requires disabling the default peripheral configurations in the Hardware Configurator. Instructions on how to achieve this are in section 3.1.1 Disabling Peripheral Functionality.

For BRD4169: Button and LED peripherals must be disabled.

For BRD4180: Button, LED, and UART RTS/CTS must be disabled.

Other radio boards do not require any peripherals to be disabled.

## 3.1 Jumper Wire Configurations for Different Boards/Devices

**Table 3.1.  Jumper Wire Recommendations for PTA on Different Devices**

| Board/Device | REQUEST (EXP/GPIO) | GRANT (EXP/GPIO) | PRIORITY (EXP/GPIO) | RHO (EXP/GPIO) |
|---|---|---|---|---|
| BRD4151/EFR32MG1P | EXP15/PC10 | EXP13/PF3 | EXP11/PD12 | EXP16/PC11 |
| BRD4161/EFR32MG12P | EXP15/PC10 | EXP13/PC9 | EXP11/PD12 | EXP16/PC11 |
| BRD4168/EFR32MG13P | EXP15/PC10 | EXP13/PF3 | EXP11/PD12 | EXP16/PC11 |
| BRD4169/EFR32MG14P | EXP15/PC10 | EXP13/PF3 | EXP10/PC9 | EXP16/PC11 |
| BRD4180/EFRMG21 | EXP11/PB00 | EXP6/PC01 | EXP13/PB01 | Not supported |

It is also helpful to observe the FEM control signals, and the recommended pins for different boards are shown in the following table.

**Table 3.2.  Jumper Wire Recommendations for Radio State on Different Devices**

| Board/Device | TX PRS Channel | TX (EXP/GPIO) | RX PRS Channel | RX (EXP/GPIO) |
|---|---|---|---|---|
| BRD4151/EFR32MG1P | CH5 | EXP7/PD10 | CH6 | EXP9/PD11 |
| BRD4161/EFR32MG12P | CH5 | EXP7/PD10 | CH6 | EXP9/PD11 |
| BRD4168/EFR32MG13P | CH5 | EXP7/PD10 | CH6 | EXP9/PD11 |
| BRD4169/EFR32MG14P | CH2 | EXP7/PF6 | CH3 | EXP9/PF7 |
| BRD4180/EFRMG21 | CH8 | EXP7/PD02 | CH9 | EXP9/PD03 |

For jumper positions for different coexistence configurations, refer to section 3.2 Configuring for Different PTA Scenarios.

### 3.1.1 Disabling Peripheral Functionality

The following instructions explain how to disable default peripheral configurations on the WSTK, to allow GPIOs routed to these peripherals to be used for PTA functions. Customize the examples as described below, then follow the steps in the applicable quick-start guide to build and flash thenm to your devices.

**In SDK 7.0 and higher:**

1. In the Project Configurator, click the SOFTWARE COMPONENTS tab and type 'button' in the search field to locate the **btn1** component.
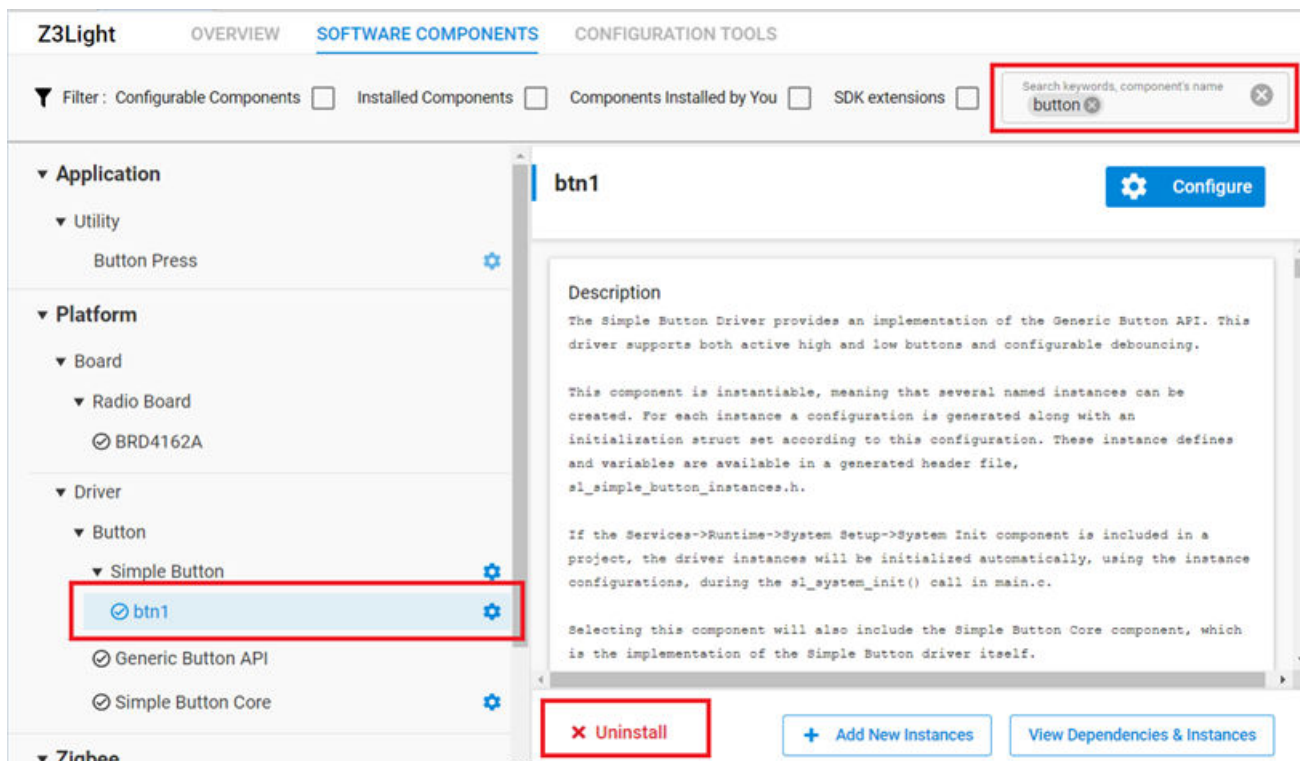


2. To disable Button peripherals, select the **btn1** component and click [**Uninstall**] to remove it.
3. Repeat the process to disable LED peripherals.
4. Type 'USART' in the search field to locate the **vcom** component.
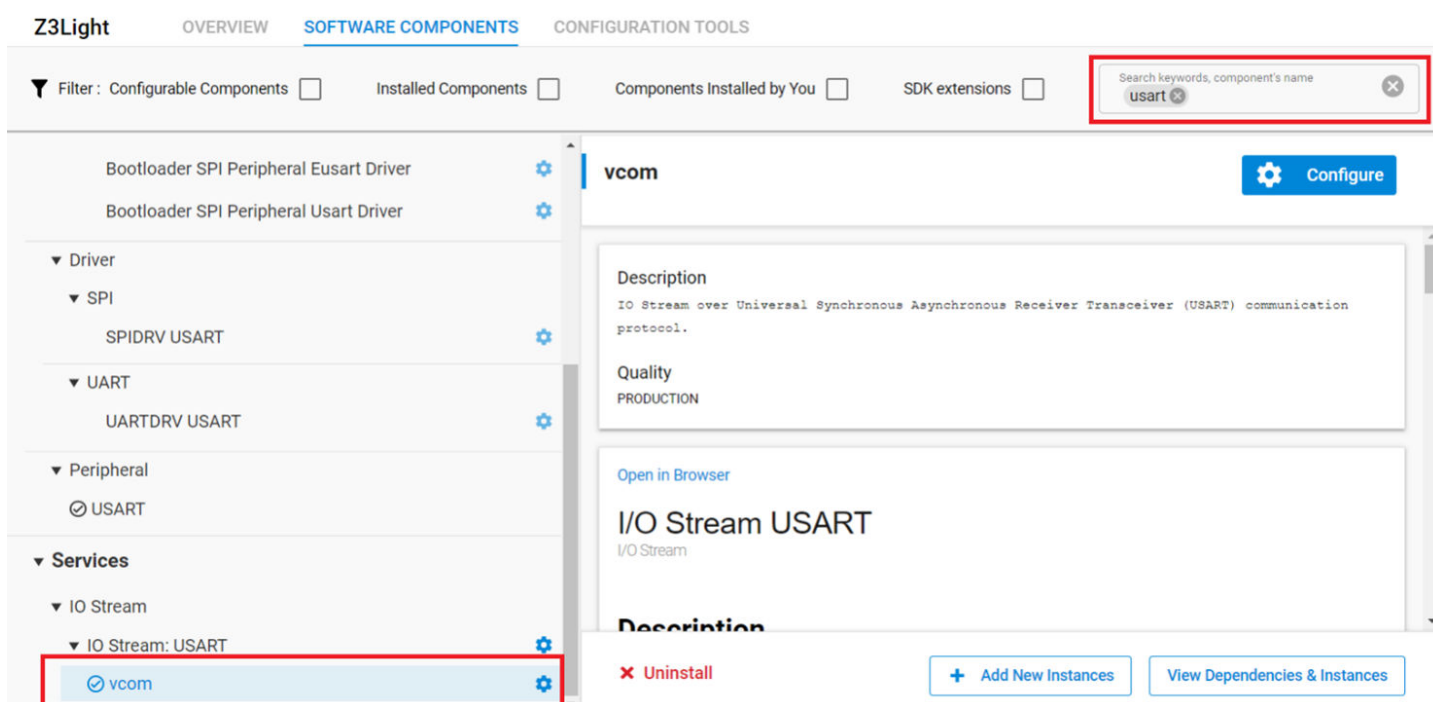
5. Click [**Configure**] to open the Component Editor. To disable the USART CTS/RTS support, change the following settings to None:
  • FlowControl
  • CTS
  • RTS

**In SDK 6.10.x and lower:**

1. In the project .isc file, select the HAL configuration tab and click [**Open Hardware Configurator**].
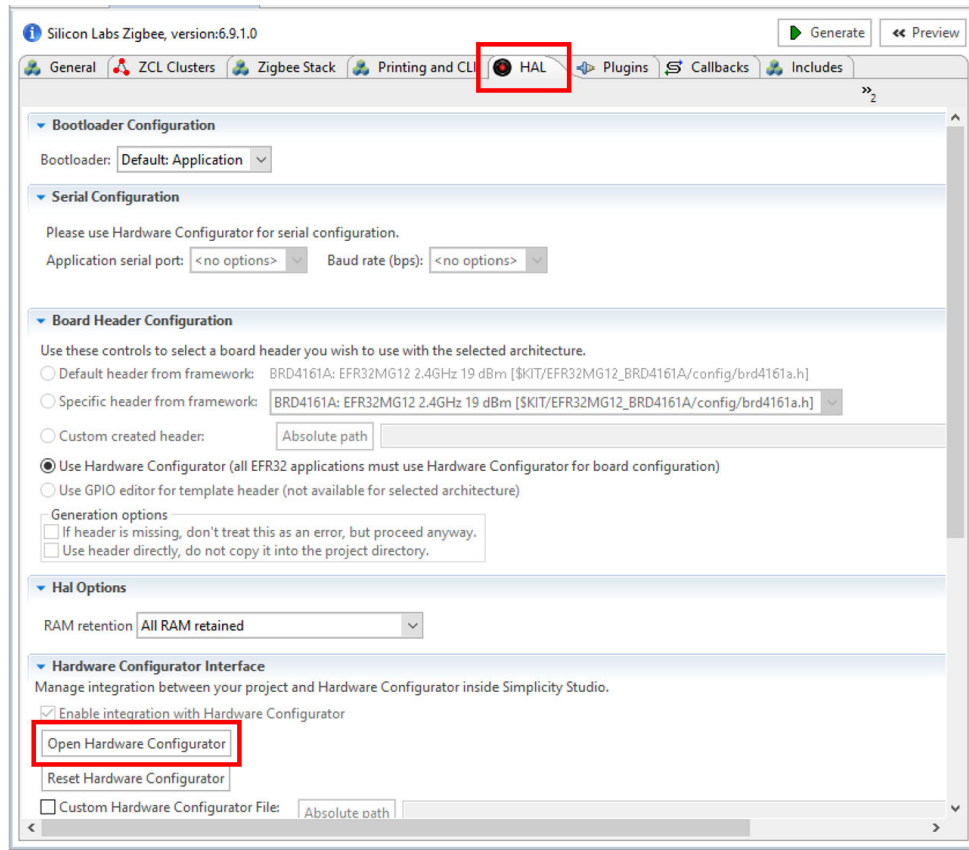


This opens a new tab with the Configurator perspective active.

2. Click the DefaultMode Peripherals tab to open a list of enabled configurators.

3. To disable LED and Button peripherals, uncheck the checkboxes next to **LED** and/or **Button**, and click [**Save**] (or press Ctrl+S). The hal-config.h file will be automatically updated with the latest changes.
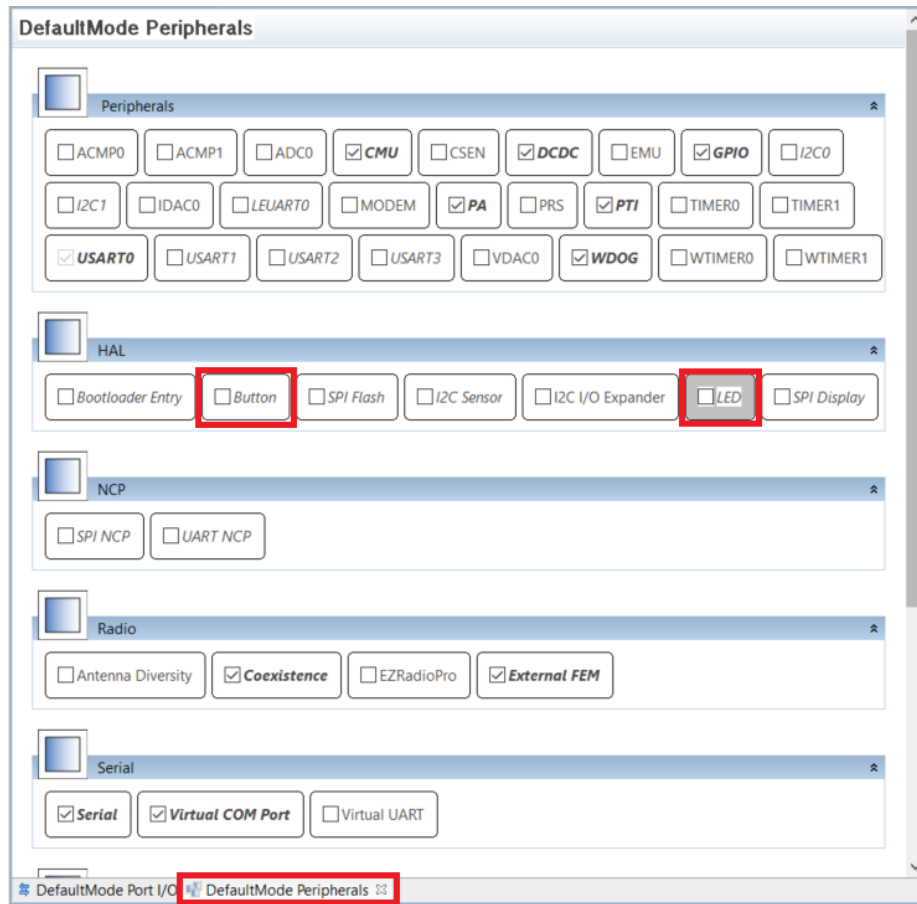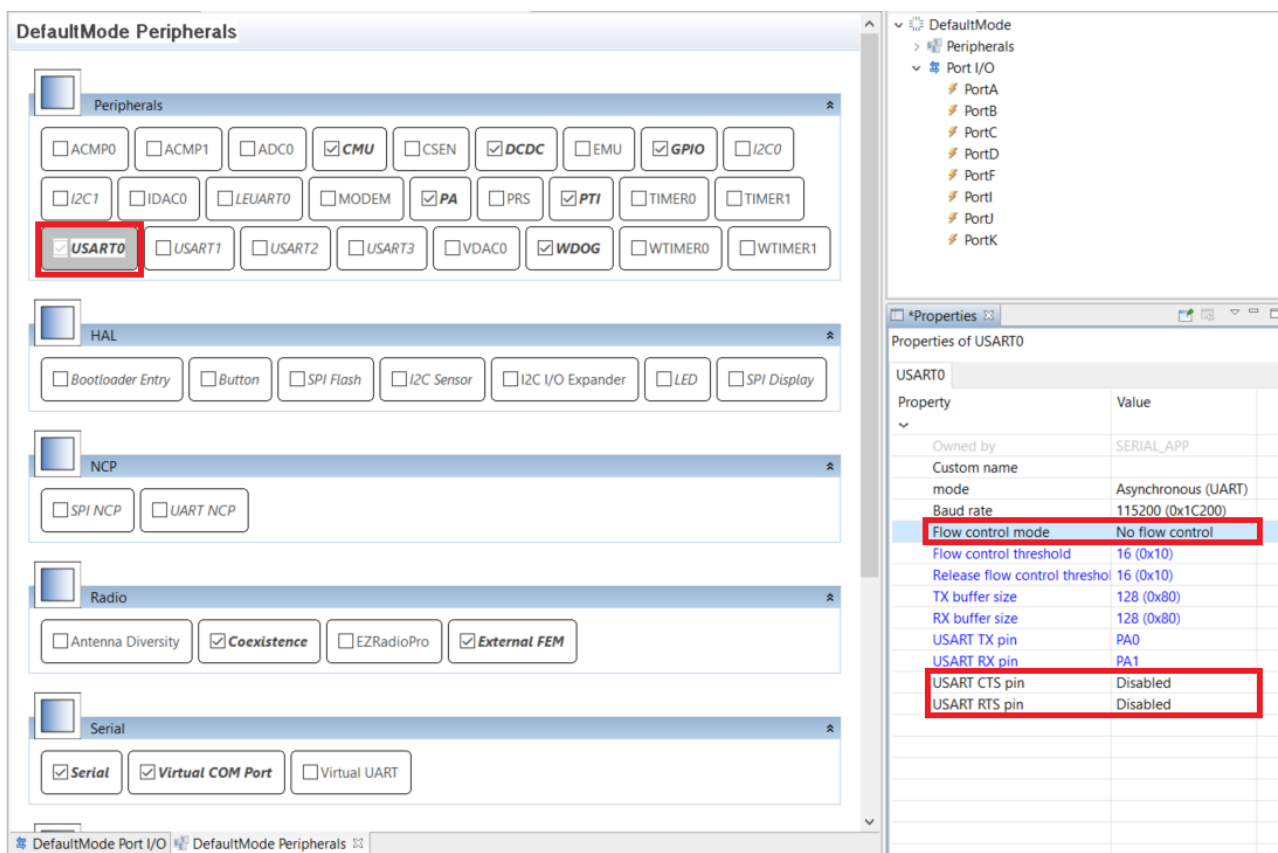
4. To disable the USART CTS/RTS support, select the USART0 box under Peripherals, and modify the properties as follows:

Flow Control Mode: No flow control

USART CTS pin: Disabled

USART RTS pin: Disabled

Click [**Save**] (or press Ctrl+S). The hal-config.h file will be automatically updated with the latest changes.



## 3.2  Configuring for Different PTA Scenarios

### 3.2.1 Single-EFR32 Radio

For a single EFR32 radio, be sure to:

1. Configure Board jumpers and jumper wires as follows. Make sure the jumper wire selections are based off the board and device being used. Refer to Table 3.1 Jumper Wire Recommendations for PTA on Different Devices on page 20 and Table 3.2 Jumper Wire Recommendations for Radio State on Different Devices on page 20 for details.
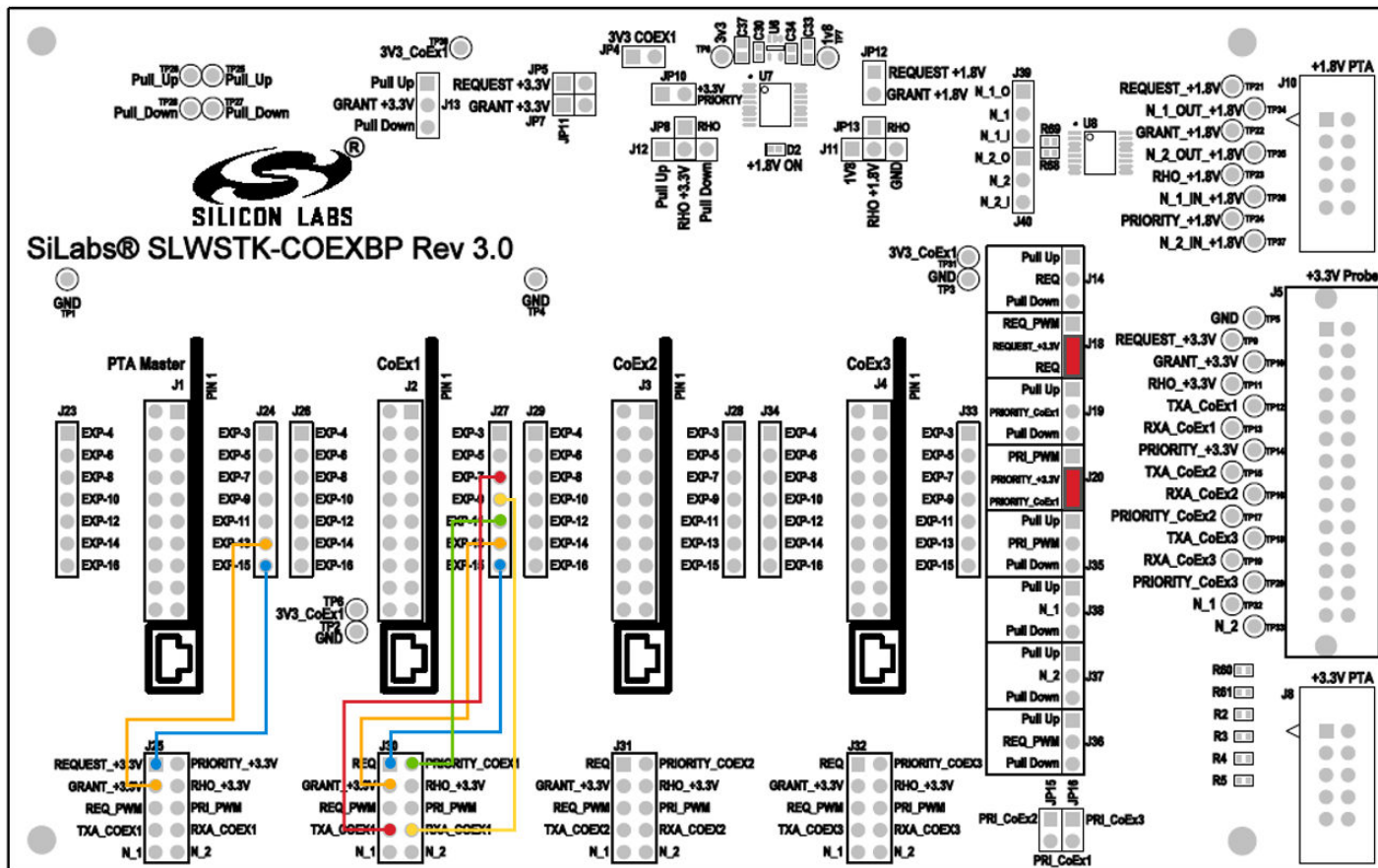


**Figure 3.1. Coexistence Backplane Board Jumpers for Single-EFR32 Radio**

2. Develop the desired PTA test application using AppBuilder.

3. Add the coexistence-configuration component/plugin and configure for target Wi-Fi/PTA device as per single-radio description (see the section "PTA Support Software Setup" in *AN1017: Zigbee® and OpenThread Coexistence with Wi-Fi*).

4. Add FEM driver component/plugin (configured as described above) to enable TXA and RXA observation.

5. Build the EFR32 application and program the WSTK.

6. Plug the EXP header on the EFR32/WSTK Board into Coexistence Backplane Board CoEx1 header (J2) (ensure pin 1 to pin 1).

7. Enable PTA on the Wi-Fi/PTA device.

8. Execute Wi-Fi stream and 802.15.4 stream.

9. Observe Wi-Fi error rate and 802.15.4 message success.

10. Observe PTA signals to debug and adjust as necessary.

11. With the PTA solution confirmed, the coexistence-component/plugin only needs to be modified for the PTA GPIOs assigned on target hardware.

### 3.2.2 Multi-EFR32 Radios with Unshared PRIORITY

For multi- EFR32 radios with unshared PRIORITY, be sure to do the following:

1. Configure Board jumpers and jumper wires as follows. If using two radios, do not configure jumper wires for Coex 3. Make sure the jumper wire selections are based off the board and device being used. Refer to Table 3.1 Jumper Wire Recommendations for PTA on Different Devices on page 20 and Table 3.2 Jumper Wire Recommendations for Radio State on Different Devices on page 20 for details.
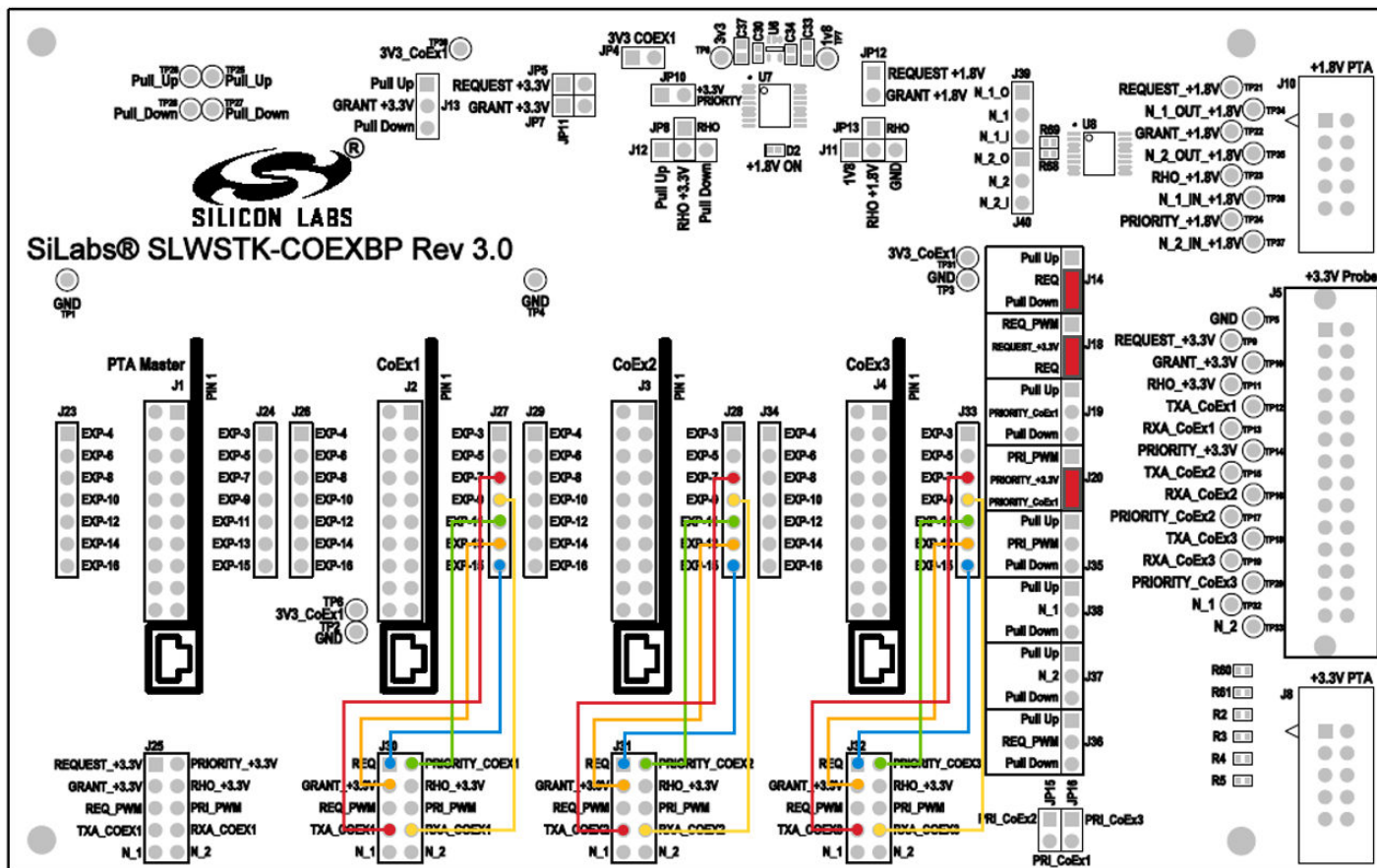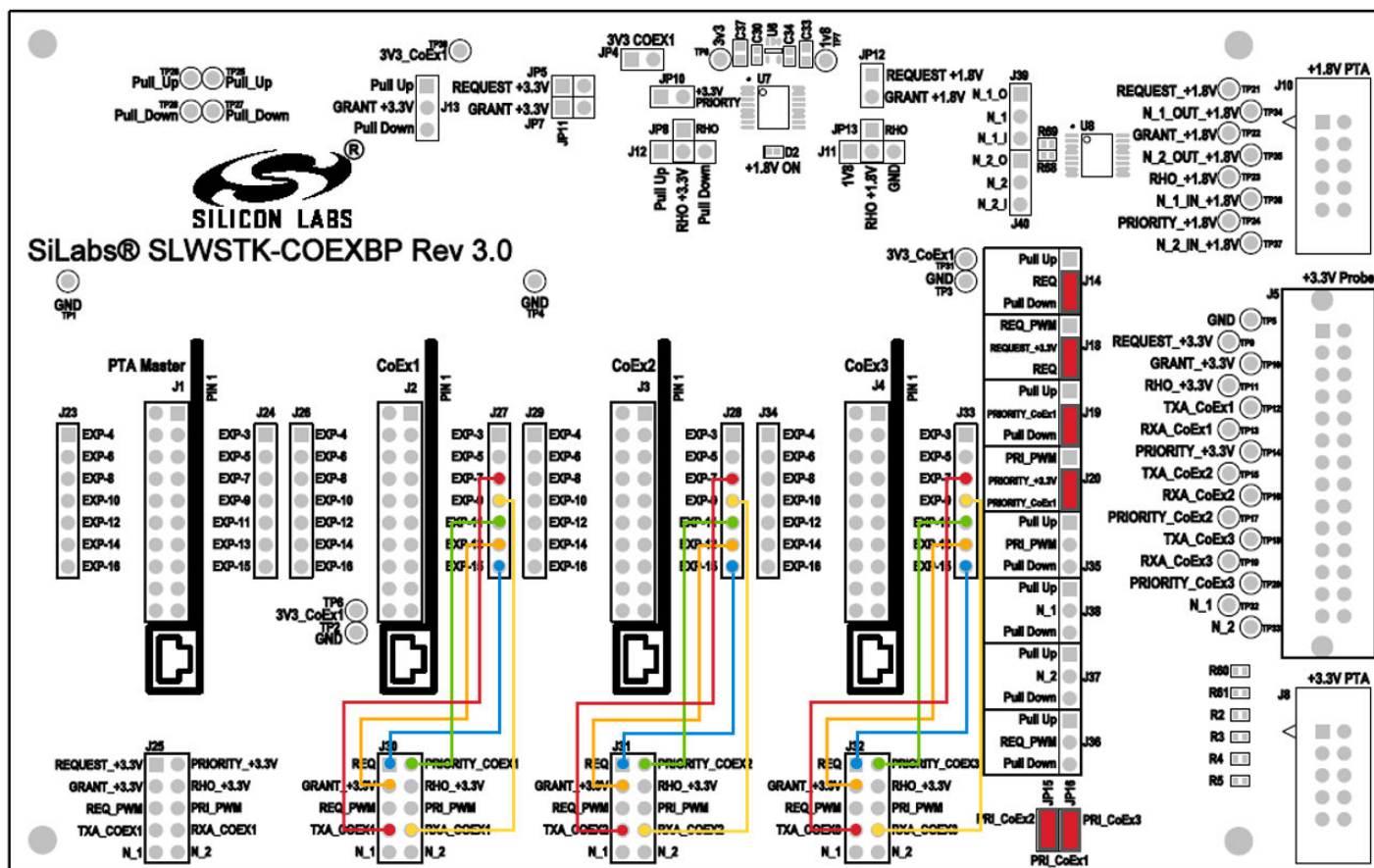


**Figure 3.2. Coexistence Backplane Board Jumpers for Multi-EFR32 Radios with Unshared PRIORITY**

2. Develop the desired PTA test applications in Simplicity Studio.

3. Add coexistence-configuration components/plugins and configure for target Wi-Fi/PTA device as per multi-radio description (see the section "PTA Support Software Setup" in *AN1017: Zigbee® and OpenThread Coexistence with Wi-Fi*).

4. Add the FEM driver component/plugin (configured as described above) to enable TXA and RXA observation.

5. Build the EFR32 applications and program the WSTKs.

6. Plug the EXP header on one EFR32/WSTK Board into Coexistence Backplane Board CoEx1 header (J2) and the second and third EFR32/WSTK Boards, if used, into CoEx2 and CoEx3 headers (J3 and/or J4) (ensure pin 1 to pin 1).

7. Enable PTA on the Wi-Fi/PTA device.

8. Execute Wi-Fi stream and 802.15.4 streams.

9. Observe Wi-Fi error rate and 802.15.4 message success.

10. Observe PTA signals to debug and adjust as necessary.

11. With the PTA solution confirmed, the coexistence-component/plugin only needs to be modified for the PTA GPIOs assigned on target hardware.

### 3.2.3 Multi-EFR32 Radios with Shared PRIORITY

For multi-EFR32 radios with shared PRIORITY, be sure to do the following.

1. Configure Board jumpers and jumper wires as follows. If using 2 radios, do not configure jumper wires for Coex 3. Make sure the jumper wire selections are based off the board and device being used. Refer to Table 3.1 Jumper Wire Recommendations for PTA on Different Devices on page 20 and Table 3.2 Jumper Wire Recommendations for Radio State on Different Devices on page 20 for details.



**Figure 3.3. Coexistence Backplane Board Jumpers for Multi-EFR32 Radios with Shared PRIORITY**

2. Develop desired PTA test applications in Simplicity Studio.
3. Add coexistence-configuration components/plugins and configure for target Wi-Fi/PTA device as per multi-radio description (see the section "PTA Support Software Setup" in *AN1017: Zigbee® and OpenThread Coexistence with Wi-Fi*).
4. Add the FEM driver component/plugin (configured as described above) to enable TXA and RXA observation.
5. Build the EFR32 applications and program the WSTKs.
6. Plug the EXP header on one EFR32/WSTK Board into Coexistence Backplane Board CoEx1 header (J2) and the second and third EFR32/WSTK Boards, if used, into CoEx2 and CoEx3 headers (J3 and/or J4) (ensure pin 1 to pin 1).
7. Enable PTA on the Wi-Fi/PTA device.
8. Execute Wi-Fi stream and 802.15.4 streams.
9. Observe Wi-Fi error rate and 802.15.4 message success.
10. Observe PTA signals to debug and adjust as necessary.
11. With the PTA solution confirmed, the coexistence-component/plugin only needs to be modified for the PTA GPIOs assigned on target hardware.

### 3.2.4 PWM Feature

Refer to *AN1017: Zigbee® and OpenThread Coexistence with Wi-Fi* on how to set up the board to use the PWM feature.

### 3.2.5 Signal Identifier Feature (EFR32xG24)

Refer to *AN1017:Zigbee® and OpenThread Coexistence with Wi-Fi* on how to set up the board to use the Signal Identifier feature.

## 3.3 I/O to Wi-Fi/PTA Devices for +3.3V or +1.8V

The Coexistence Backplane Board provides the capability of interfacing to Wi-Fi/PTA devices via +3.3 V or +1.8 V I/O as needed.

### 3.3.1 I/O for +3.3V

For +3.3 V I/O to Wi-Fi/PTA device, PTA signal connections to Wi-Fi/PTA should be made to J8 as shown in the following table. Add appropriate jumpers and jumper wires based on device configuration explained in section 3.2 Configuring for Different PTA Scenarios.
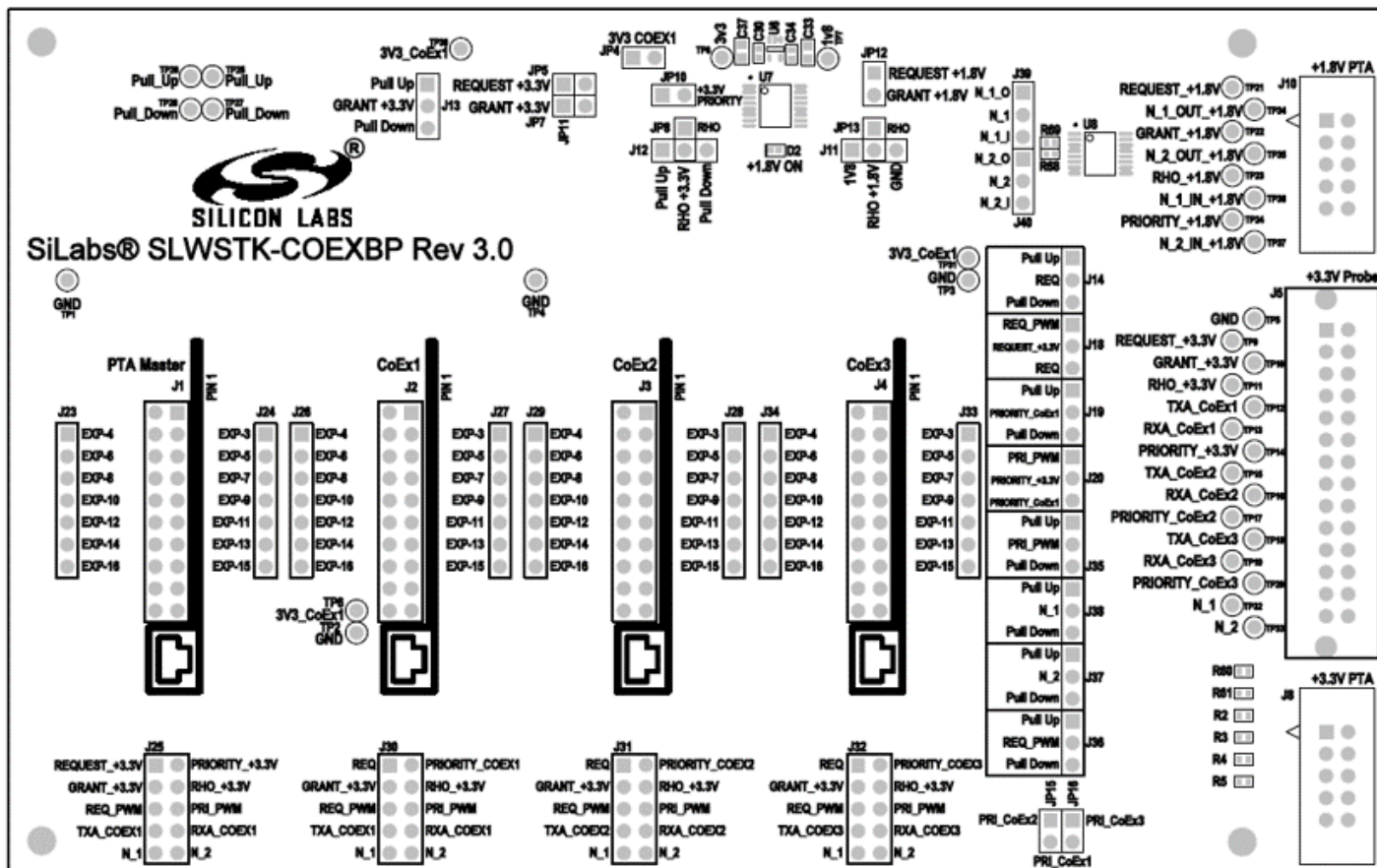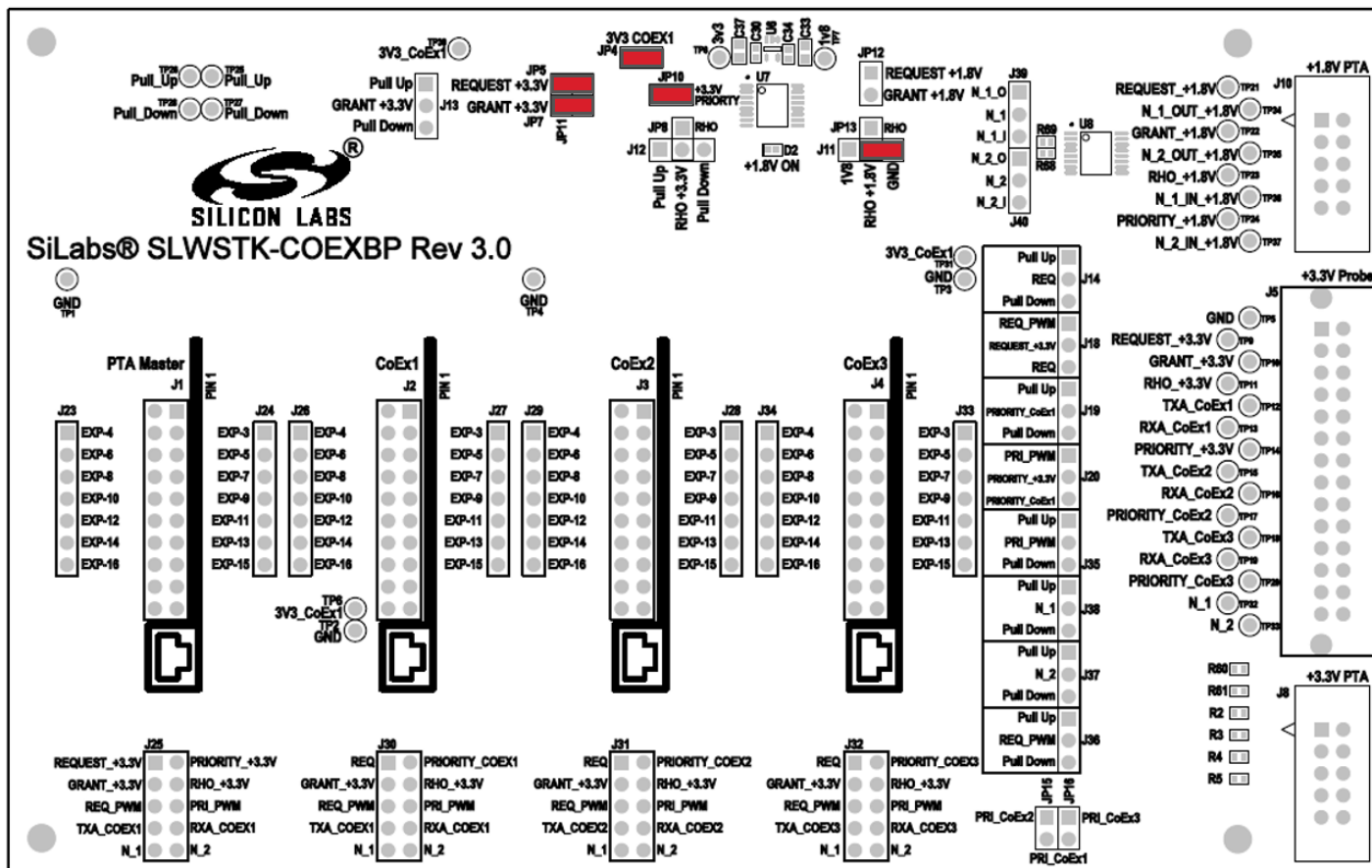


**Figure 3.4. Coexistence Backplane Board Jumpers for +3.3V I/O Operation**

The +3.3V /IO connections to Wi-Fi/PTA device can be made via +3.3V PTA HDR (J8) as follows:

**Table 3.3. Header (J8) Pins for +3.3V I/O Connections**

| EFR32 PTA Signal | J8 Pin |
|---|---|
| REQUEST_+3.3V | 1 |
| GRANT_+3.3V | 3 |
| RHO_+3.3V | 5 |
| PRIORITY_+3.3V | 7 |
| GND | 9 and 10 |

**Note:** Additional jumpers are required to configure the REQUEST and PRIORITY signals. Jumper configuration varies with single-EFR32 radio and multi-EFR32 radio configurations and active-high/active-low polarities. See section 3.2 Configuring for Different PTA Scenarios for these jumper options.

### 3.3.2  I/O for +1.8V

For +1.8V I/O to Wi-Fi/PTA device, assuming typical 3-Wire PTA with RHO unused) PTA signal connections to Wi-Fi/PTA should be made to J10 and all red jumpers shown below are required. Add additional jumpers and jumper wires based on device configuration explained in section 3.2 Configuring for Different PTA Scenarios.



**Figure 3.5.  PTA Backplane Board Jumpers for +1.8V I/O Operation**

The +1.8 V /IO connections to Wi-Fi/PTA device can be made via +1.8 V PTA HDR (J10) as follows:

**Table 3.4.  Header (J10) Pins for +1.8V I/O Connections**

| EFR32 PTA Signal | J10 Pin |
|---|---|
| REQUEST_+1.8V | 1 |
| GRANT_+1.8V | 3 |
| RHO_+1.8V; (If RHO is used, move J11/JP13 jumper and add J8/J12 jumper) | 5 |
| PRIORITY_+1.8V | 7 |
| GND | 9 and 10 |

**Note:** Additional jumpers are required to configure the REQUEST and PRIORITY signals. Jumper configuration varies with single-EFR32 radio and multi-EFR32 radio configurations and active-high/active-low polarities. See section 3.2 Configuring for Different PTA Scenarios for these jumper options.

## 3.4  Probe Header

The EFR32 PTA signals (+3.3V I/O) are observable via a logic analyzer connected to J5 as shown below:
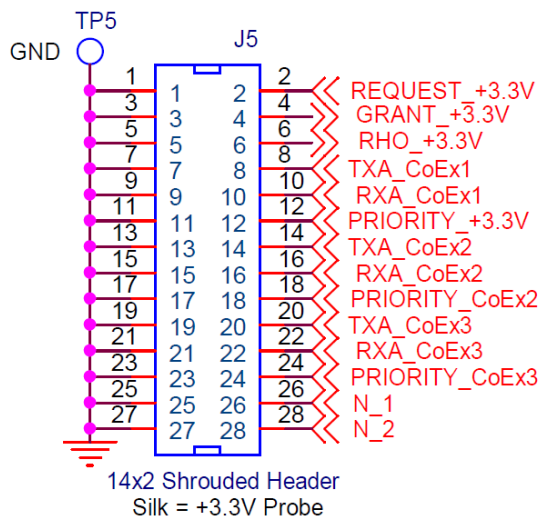


**Figure 3.6.  Coexistence Backplane Board Header for PTA Signal Logic Analyzer Connection**

Where:

- CoEx1 refers to EFR32/WSTK connected into CoEx1 header (J2)
- CoEx2 refers to EFR32/WSTK connected into CoEx2 header (J3)
- CoEx3 refers to EFR32/WSTK connected into CoEx3 header (J4)

**Note:** Depending on jumper configurations, PRIORITY_+3.3 V can be PRIORITY_CoEx1 or wired-OR/-AND of multiple PRIORITY signals. See section 3.2 Configuring for Different PTA Scenarios for these PRIORITY_+3.3 V jumper options.

## 3.5 Board Schematics



**Figure 3.7. SLWSTK-COEXBP Schematic (1/3)**
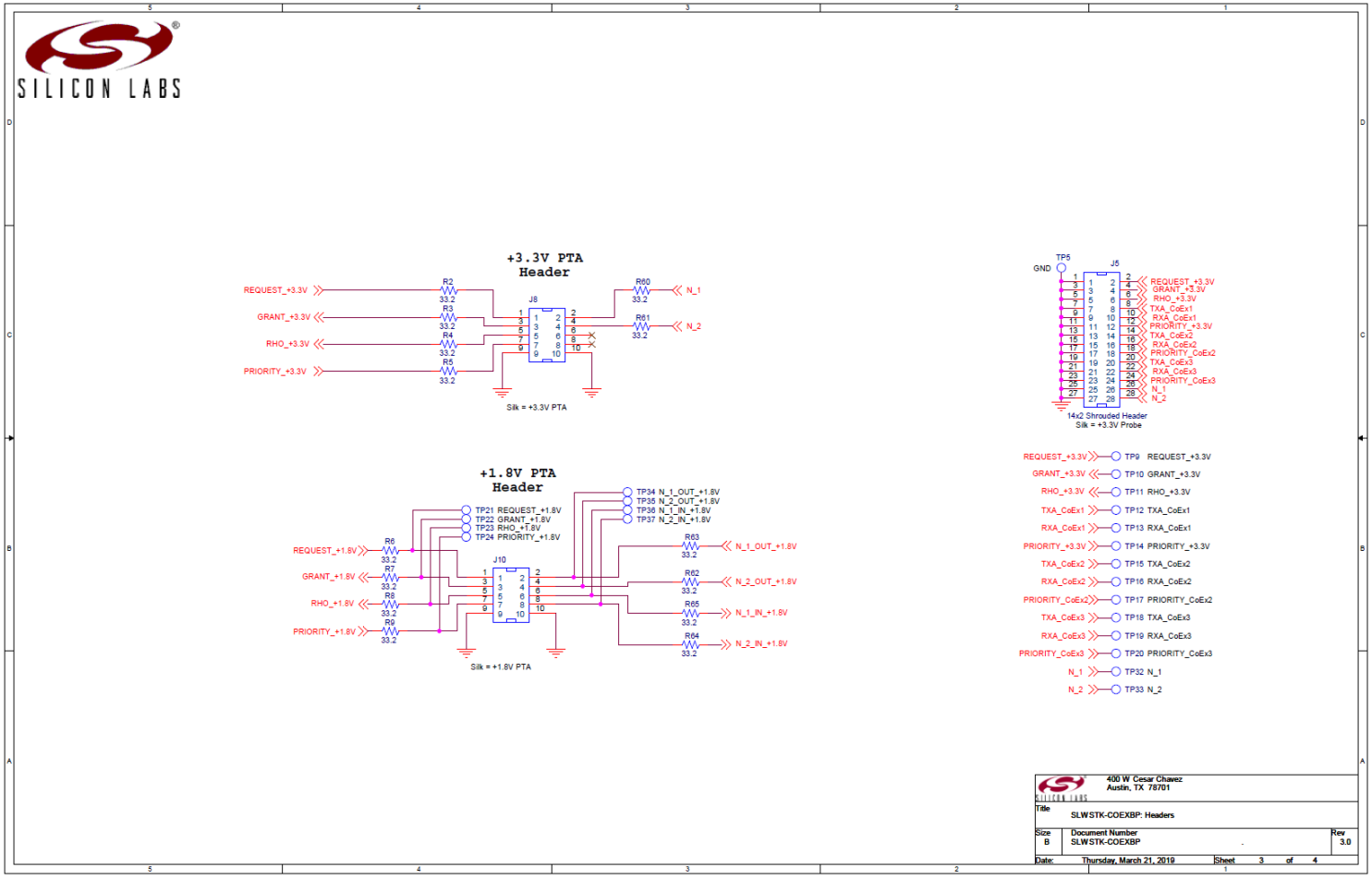
**Figure 3.8. SLWSTK-COEXBP Schematic (2/3)**
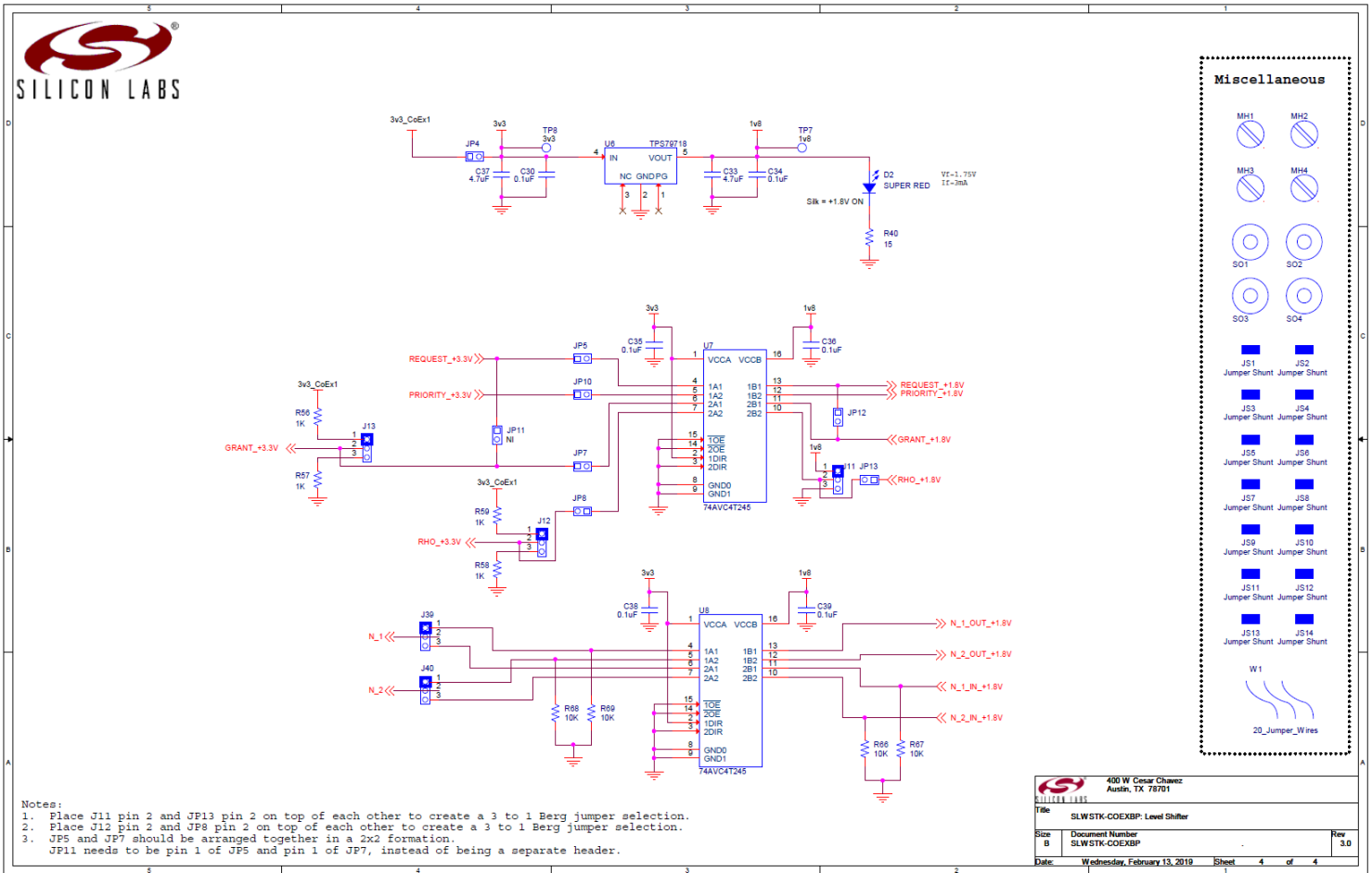
**Figure 3.9. SLWSTK-COEXBP Schematic (3/3)**

### 3.6 Packet Trace

For the purposes of debugging, it can be useful to monitor the packet trace (PTI) signals generated by the radio device. These signals, when viewed with TXA and RXA to indicate radio state, can help determine when a packet was received or transmitted, and whether the radio transaction was successful. To access the PTI signals, some modifications need to be made to the coexistence device's WSTK board.

1. Solder a header into the pin locations currently marked as NC. These pins are not connected to any functions on the WSTK board. It is also recommended to add header pins to the adjacent GND pin locations to provide proper ground reference to the logic analyzer.
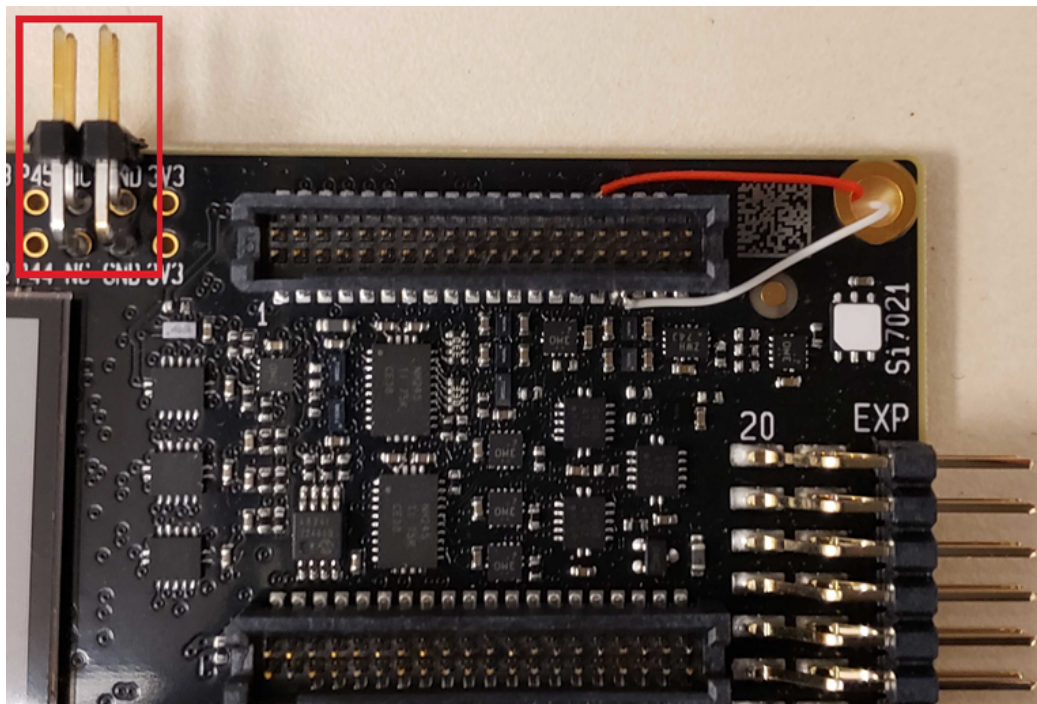


**Figure 3.10. PTI Header Location**

2. Solder wires to pin 32 and pin 33 of the 40-pin connector. Pin 32 corresponds to the PTI_Frame/PTI_Sync signal, and Pin 33 corresponds to the PTI_DATA signal.
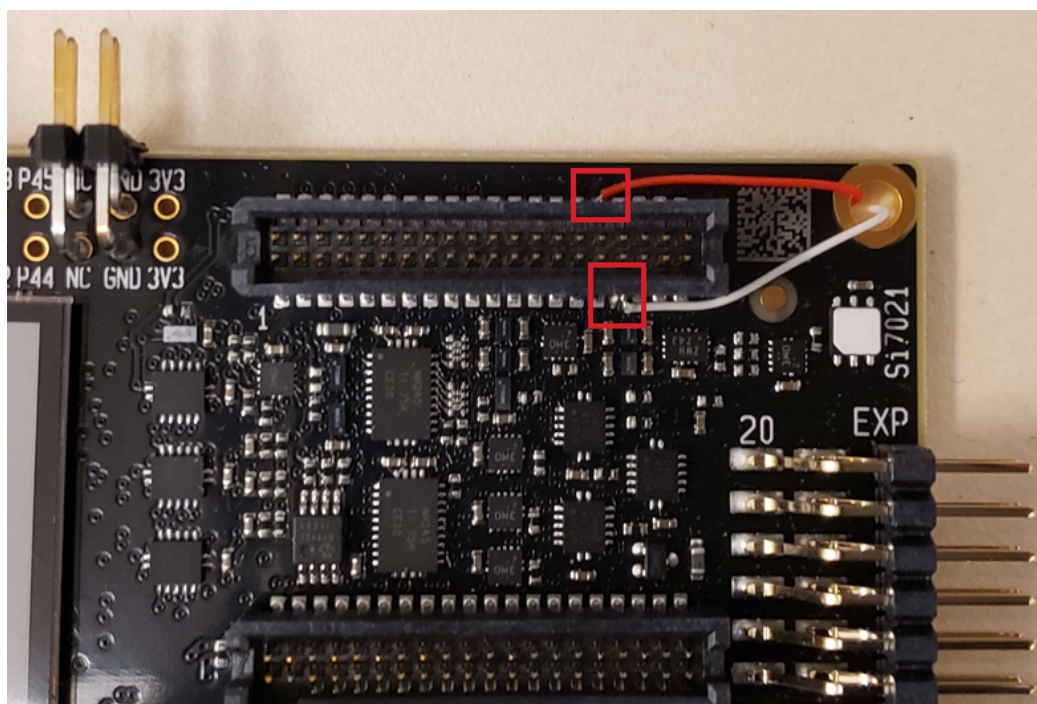


**Figure 3.11. PTI Pin Locations**

3. Route these wires to the back of the board and solder onto the header pins that are in the NC pin locations.
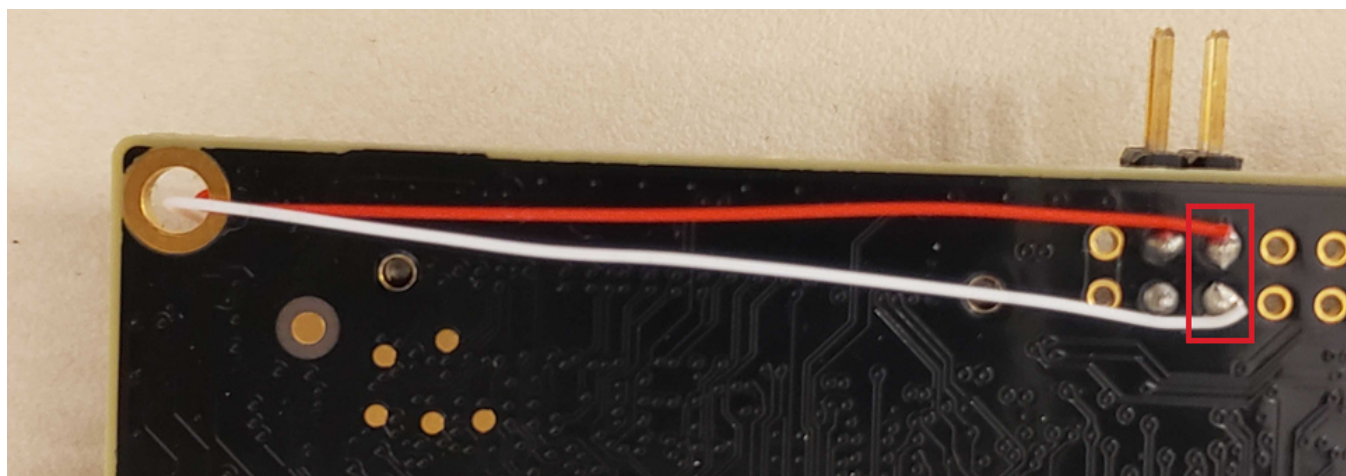


**Figure 3.12. PTI Header Soldering**

4. Connect a logic analyzer to the two PTI signals.

The logic analyzer will now be able to capture Packet Trace information alongside coexistence activity from the radio.
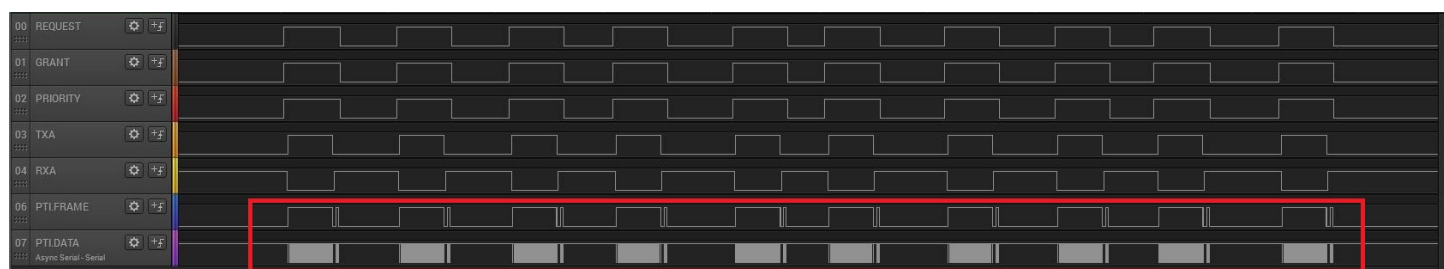


**Figure 3.13. PTI Logic Capture**

For information on how to analyze the PTI data captured, refer to *AN1017: Zigbee® and OpenThread Coexistence with Wi-Fi®*, *AN1128: Bluetooth® Coexistence with Wi-Fi®*, or contact the Silicon Labs support team.

# 4. Throughput Library

## 4.1 Throughput Library Commands

**Table 4.1. List of Commands Available in the Throughput Library**

| Command | Command Description | Name | Type | Description |
|---|---|---|---|---|
| plugin throughput start | Start the throughput test | - | - | - |
| plugin throughput stop | Abort the test while running | - | - | - |
| plugin throughput print-result | Show the results of the last test | - | - | - |
| plugin throughput set-destination | Set the destination nodeId | nodeId | INT16U | Destination nodeId |
| plugin throughput set-interval | Set the interval | interval | INT16U | Interval in ms |
| plugin throughput set-timeout | Set the test timeout | timeout | INT32U | Timeout in ms |
| plugin throughput set-count | Set the number of packets to send | count | INT16U | Packets to send |
| plugin throughput set-inflight | Set the number of packets in flight during the test | inflight_count | INT8U | Packets in flight |
| plugin throughput set-packet-size | Set the packet length | packet | INT8U | Packet length in bytes |
| plugin throughput set-all | Set the NodeID | nodeId | INT16U | Destination nodeId |
| plugin throughput set-all | Set the Count | count | INT16U | Packets to send |
| plugin throughput set-all | Set the Interval | interval | INT16U | Interval in ms |
| plugin throughput set-all | Set the Packet Length | packet | INT8U | Packet length in bytes |
| plugin throughput set-all | Set the Max In-Flight | inflight_count | INT8U | Packets in flight |

| Command | Command Description | Name | Type | Description |
|---|---|---|---|---|
| plugin throughput set-all | Set the APS Options | apsOptions | INT16U | APS Options |
| plugin throughput set-all | Set the Test Timeout | timeout | INT32U | Timeout in ms |
| plugin throughput set-aps-ack-off | Turn off APS acks | - | - | - |
| plugin throughput set-aps-ack-on | Turn on APS acks | - | - | - |
| plugin throughput print-parameters | Print all the test parameters | - | - | - |
| plugin throughput print-counters | Print the stack counters | - | - | - |
| plugin throughput clear-counters | Clear all the stack counters | - | - | - |

## 4.2 Using the Throughput Library

To begin using the throughput library, the required parameters must be defined. The simplest way to achieve this is by using the command `plugin throughput set-all <parameters>`. Details regarding the command and order of parameters can be found in Table 4.1 List of Commands Available in the Throughput Library on page 37.

A sample command is `plugin throughput set-all <0xNodeID> 10 0 127 1 0 100000`, where `NodeID` must be defined as the no-deID of the receiving node, typed as a hex number. Note: To achieve maximum throughput, set the interval field to 0.

To print the parameters currently set, use the command `plugin throughput print-parameters`. This returns the parameters currently defined, in the following format:

```
TEST PARAMETERS
Destination nodeID: 0x1234
Packets to send: 10
Transmit interval: 0 ms
Payload size: 77B
Packet size: 127B
Packets in flight: 1
APS Options = 0x0000
Timeout: 100000 ms
```

To start the throughput process, use the command `plugin throughput start`. This starts sending packets to the other device. To stop the test during the run, use the command `plugin throughput stop`. Once the test is complete, you can print the results using `plugin throughput print-result`. This produces the results of the most recent run, in the following format:

```
THROUGHPUT RESULTS
Total time 96 ms
Success messages: 10 out of 10
Payload Throughput: 64166 bits/s
Phy Throughput: 105833 bits/s
Min packet send time: 6 ms
Max packet send time: 8 ms
Avg packet send time: 6 ms
STD packet send time: 2 ms
```

The throughput is calculated by dividing the total number of bits sent by the total time.

For Phy Throughput:

$$Phy\ Throughput\ (bits/s)\ = \frac{Packet\ size\ (Bytes) * 8\frac{bits}{Byte} * No.\ of\ packets}{Total\ Time\ (s)}$$

For Payload Throughput:

$$Payload\ Throughput\ (bits/s)\ = \frac{Payload\ size\ (Bytes) * 8\frac{bits}{Byte} * No.\ of\ packets}{Total\ Time\ (s)}$$

The throughput component/plugin also tracks some counters to provide visibility and help with debugging purposes. Use `plugin throughput print-counters`. The results are produced in the following format:

```
COUNTERS
CCA Failures: 0
Mac Tx Ucast: 10
Mac Tx Ucast Retry: 0
Mac Tx Ucast Fail: 0
APS Tx Ucast Success: 10
APS Tx Ucast Retry: 0
APS Tx Ucast Fail: 0
```

Details regarding the MAC and APS stack counters can be found in *AN1017: Zigbee® and OpenThread Coexistence with Wi-Fi* and are also documented in the stack API documentation.

## 5. PTA Controller Application

The PTA Controller application is a custom application designed to help customers evaluate coexistence features on Silicon Labs devices. The Zigbee radios act as peripheral devices, and require a control device to react and respond to coexistence requests. The PTA Controller application implements this and allows users to control the behavior of the PTA Controller to evaluate the corresponding behavior of the peripheral device.

## 5.1 Commands in the PTA Controller Application

**Table 5.1. List of Commands Available in the PTA Controller Application**

| Command | Command Description | Argument and Description |
|---|---|---|
| configuration | Print out the current configuration of features | - |
| request | Set REQUEST signal polarity | activehigh: sets the signal polarity active HIGH |
| " | " | activelow: sets the signal polarity active LOW |
| grant | Set GRANT signal polarity | activehigh: sets the signal polarity active HIGH |
| " | " | activelow: sets the signal polarity active LOW |
| priority | Set PRIORITY signal polarity | activehigh: sets the signal polarity active HIGH |
| " | " | activelow: sets the signal polarity active LOW |
| rho | Set RHO signal polarity | activehigh: sets the signal polarity active HIGH |
| " | " | activelow: sets the signal polarity active LOW |
| grantabort | Enable/Disable GRANT Abort feature, and set abort percentage | 0: disables GRANT Abort |
| " | " | 1-100: enables GRANT Abort and sets percentage of GRANTS that are aborted |
| grantdelay | Enable/Disable GRANT Delay feature, and set delay percentage | 0: disables GRANT Delay |
| " | " | 1-100: enables GRANT Delay and sets percentage of GRANTS that are delayed |
| grantdeny | Enable/Disable GRANT Deny feature, and set deny percentage | 0: disables GRANT Deny |
| " | " | 1-100: enables GRANT Deny and sets percentage of GRANTS that are denied |
| rhofeature | Enable/Disable Radio Hold Off (RHO) feature, and set rho percentage | 0: disables RHO feature, and sets pin to High Z mode. |
| " | " | 1-100: enables RHO and sets percentage of Radio Hold Off requests |
| txStreamToggling | Output a toggling stream for a specified time, up to 60 seconds. Note: This feature sets GRANT to always asserted | 1-60: time for which to output stream |
| setTxStream | Output a stream | 1: Enable stream |
| " | " | 0: Disable stream |
| setPower | Set the current transmit power in deci dBm, or raw units if 'raw' is specified | Output power to set (range depends on capabilities of radio device. Refer to datasheet of device for more info) |
| getPower | Get the current transmit power in deci dBm | - |
| setchannel | Set the current radio channel | 0-15: Value 0 corresponds to Zigbee channel 11 |
| getchannel | Get the current radio channel | - |

## 5.2 Using the PTA Controller Application

The PTA Controller application is ready to use after compilation with the following default features:

- REQUEST is set to Active HIGH
- GRANT is set to Active HIGH
- PRIORITY is set to Active HIGH
- RHO is set to Active HIGH, but the feature is disabled and the pin is set to High Z mode
- GRANT Abort, GRANT Delay, and GRANT Deny are disabled
- No stream is active

To change any of the default configurations, use the commands previously listed.

Use the `configuration` command to print out the current configuration. An example output returned through the CLI is:

```
Request:        Active High
Grant:          Active High
Priority:       Active High
Radio Hold Off: Active High
Grant Abort:    Disabled
Grant Delay:    Disabled
Grant Deny:     Disabled
RHO Feature:    Disabled
```

The PTA Contoller application can also hinder network traffic to test behavior of the peripheral device in such conditions. The `txStreamToggling` command outputs a periodic stream for a specified time, causing packets or sections of packets on the network to get corrupted.

# Smart. Connected.
# Energy-Friendly.

**IoT Portfolio**
www.silabs.com/products
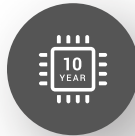
**Quality**
www.silabs.com/quality

**Support & Community**
www.silabs.com/community

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

# SILICON LABS

**www.silabs.com**