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

The SLWSTK-COEXBP is a development kit designed by Silicon Labs to demonstrate coexistence between different radios transmitting in the 2.4 GHz ISM band. The Coexistence Development Kit requires the EFR32™ Mighty Gecko Wireless Starter Kit (SLWSTK6000B), which contains the following:

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

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

The software requirements for running the coexistence features are as follows:

- Silicon labs EmberZNet PRO (Zigbee) stack
- Silicon Labs Flex SDK (includes RAIL)
- Software required to run and view logic analyzer waveforms

The software images required to create a Zigbee network with coexistence features enabled must be built through sample applications in the EmberZNet PRO stack, with added modifications.

The software required to exercise the coexistence features is available as a custom application built on top of the Flex stack.

This User Guide refers to Silicon labs EmberZNet PRO release version 6.2.3 and Silicon Labs Flex SDK release version 2.2.2.

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

- *AN1017: Zigbee® and Thread Coexistence with Wi-Fi*
- *QSG106: Getting Started with EmberZNet PRO*

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 plugins and packages used for advanced testing.

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. This User Guide details coexistence features in the EmberZNet stack.

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) Master: Acts as a Wi-Fi emulator and responds to activity on the co-located Z3Light running the coexistence features.
  - Z3Light: Zigbee router/coordinator with coexistence features enabled and connected to the PTA Master.
  - Z3Switch: Used to create a network with the Z3Light to generate Zigbee traffic.
- Coexistence Backplane Board

A logic analyzer is optional but highly recommended to enable signal analysis and debugging.

To set up the recommended coexistence test configuration, the PTA Master and Z3Light 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 of the signals for debugging and evaluation purposes. The Z3Switch 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.

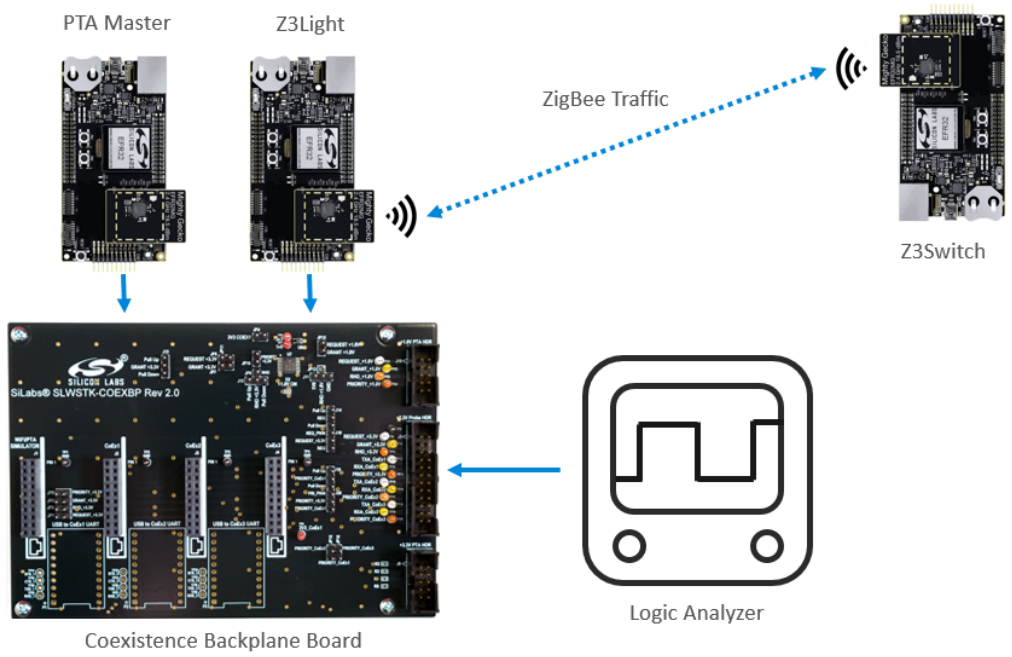


Figure 1.1. Typical Zigbee Coexistence Test Configuration

2. Coexistence Features Test Setup

This section describes the steps required to bring up a Zigbee 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. This step 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. In this step, the applications are rebuilt with the throughput library, flashed, and network traffic created between them.
- 2.3 Adding the Coexistence Feature to Zigbee Devices. In this step, 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 Master within the Test Setup. Now the PTA Master application is built and flashed to a device, which then is introduced to the test setup as a coexistence master device.
- 2.5 Replacing PTA Master with an External Wi-Fi Device. In this step the PTA Master device is replaced with an external Wi-Fi chipset.

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 *QSG106: Getting Started with EmberZNet PRO* 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 QSG106 as the document will be referred to extensively in the following steps. QSG106 can be found through the list of documents on the Simplicity Studio Launcher perspective, and on the Silicon Labs website under Zigbee documentation. A direct link to the document is below:

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

1. If you have not already done so in the previous section, follow the steps in QSG106 to start the Z3Light and Z3Switch example applications. Otherwise return to the Simplicity IDE perspective for one of the examples. If the .isc file is not open, double-click the .isc file in the Project Explorer to open it.
2. Do not generate the application yet. Additional plugins need to be added.
3. Go to the Plugins tab and enter the keyword “throughput” to search for the throughput library plugin.
4. Mark the checkbox under “Use?” to enable the plugin.

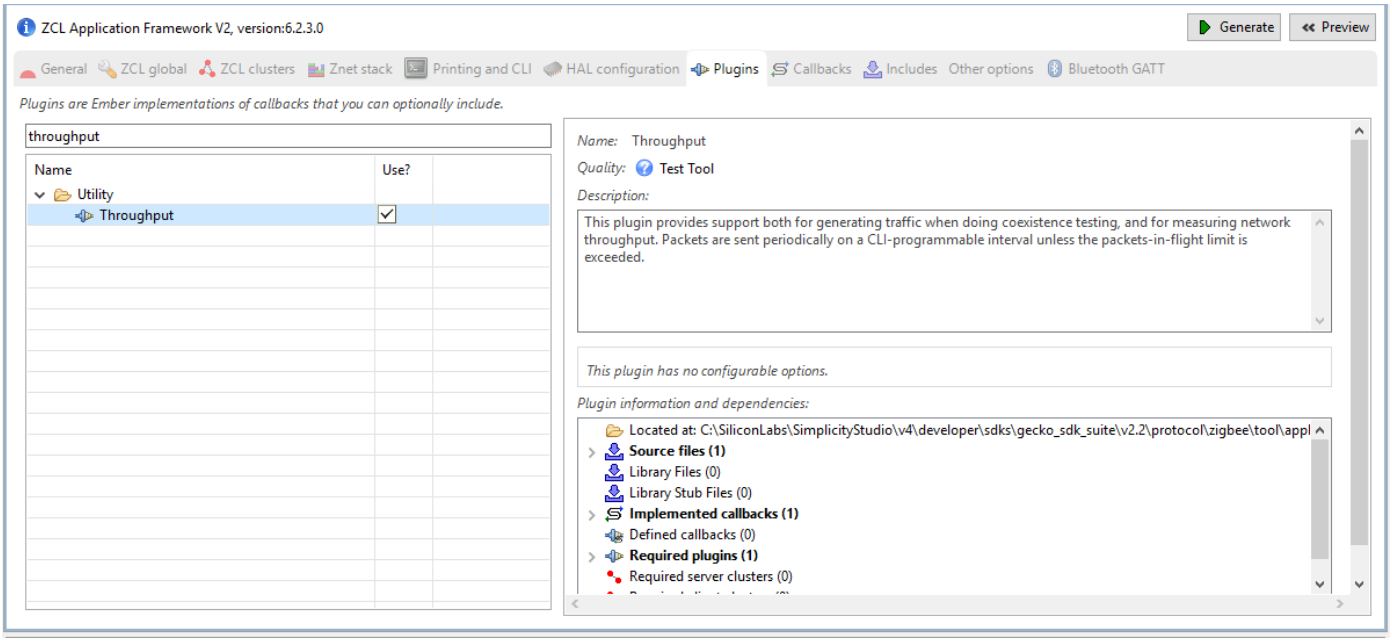


Figure 2.1. Enabling the Throughput Plugin

5. Repeat for the other example application.
6. Follow the rest of the steps in QSG106 to generate, build, and flash both the Z3Light and Z3Switch application images.

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 QSG106 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 commands response.
3. In the console of the Z3Light, send the following command

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

   where NodeID is the node ID of the Z3Switch noted in step 2 above. 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 are 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. The coexistence features are contained in an additional plugin available in the EmberZNet PRO stack. More information regarding the coexistence plugin can be found in section 4 of document *AN1017: Zigbee® and Thread Coexistence with Wi-Fi*.

It is also recommended to enable the FEM driver plugin. This plugin 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

1. Return to the Simplicity IDE perspective for one of the example .isc files.
2. Go to the Plugins tab and search for the keyword “coexistence” to find the coexistence configuration plugin.
3. Mark the checkbox under “Use?” to enable the plugin.

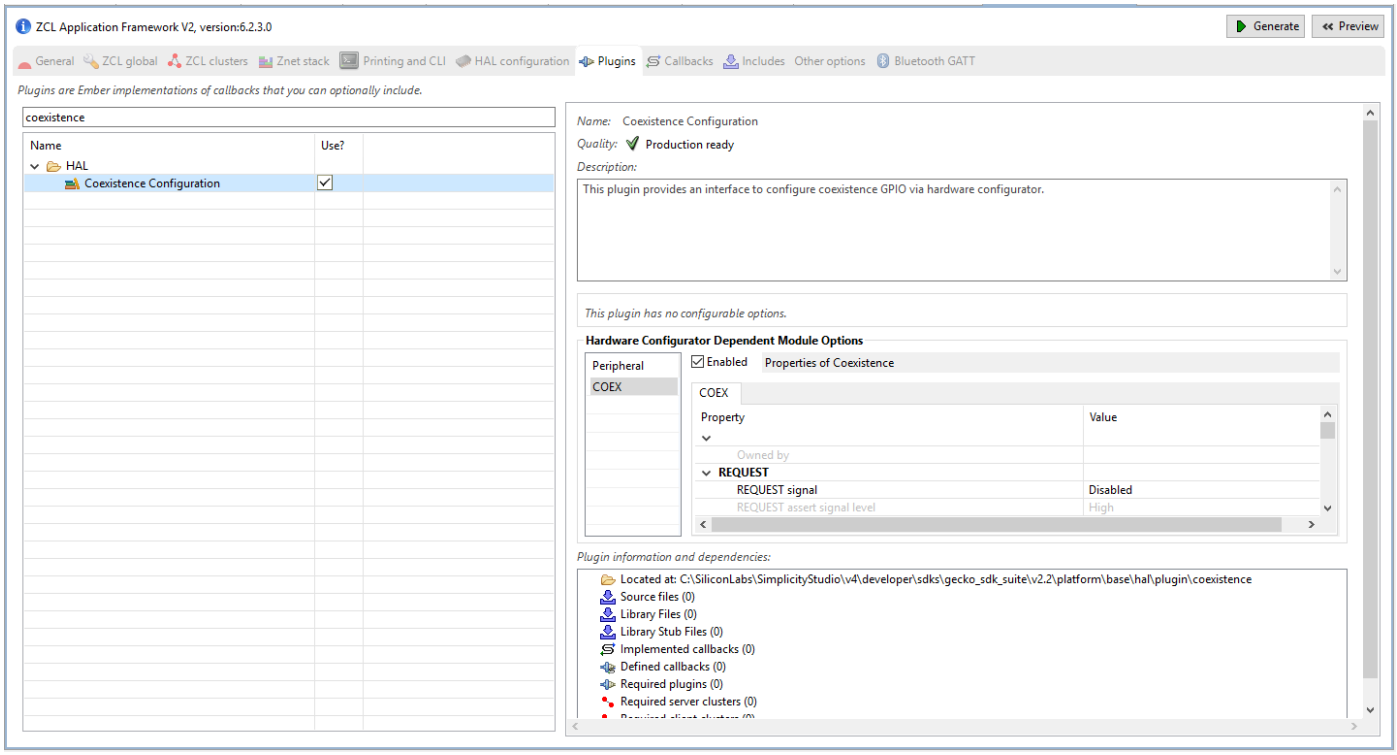


Figure 2.2. Enabling the Coexistence Configuration Plugin

4. In the menu on the right, make sure the Enabled box next to **Properties of Coexistence** is checked.
5. 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.
6. Still in the Plugins tab, search for keyword “fem” to bring the FEM driver plugin.
7. Mark the checkbox under “Use?” to enable the plugin.

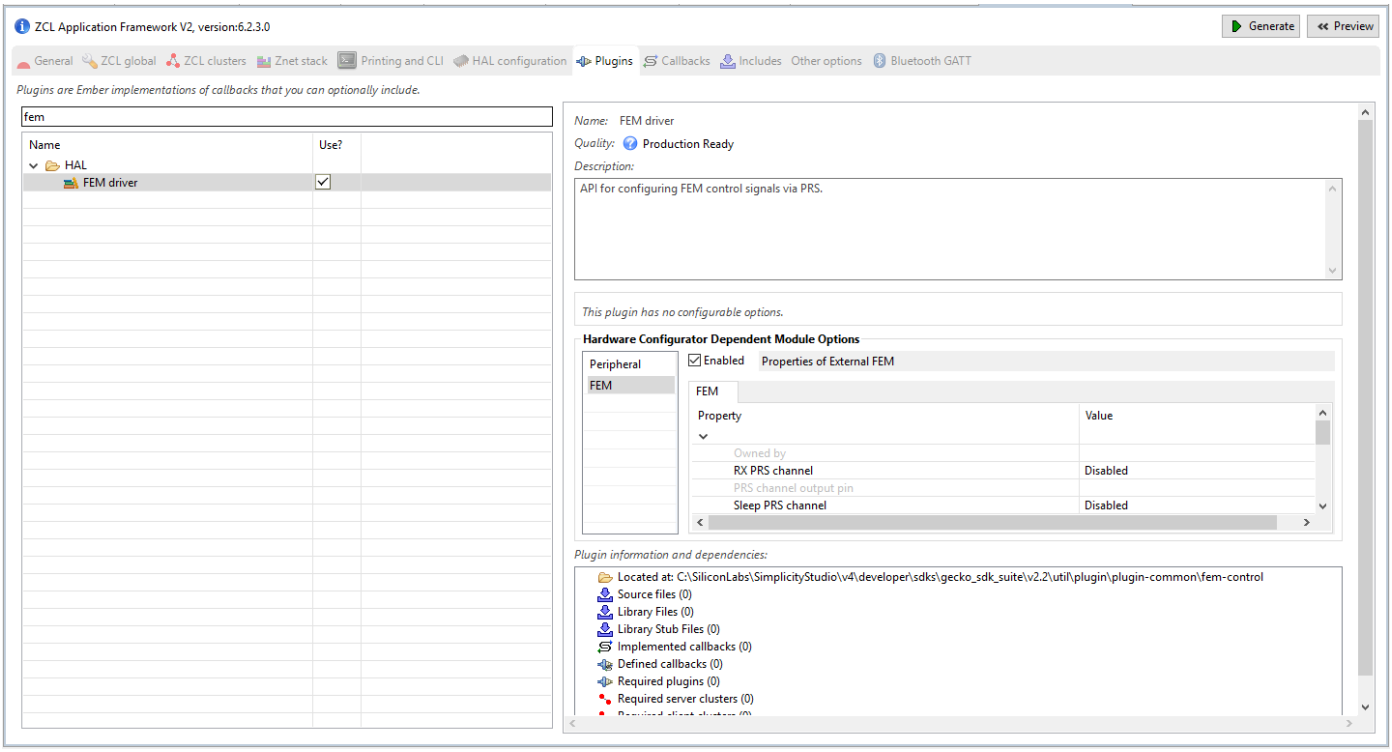


Figure 2.3. Enabling the FEM Driver Plugin

8. In the menu on the right, make sure the Enabled box next to **Properties of External FEM** is checked.
9. Modify the fields in the menu on the right as follows:
   - RX PRS channel: CH6
   - PRS channel 6 output pin: PD11
   - Sleep PRS channel: CH7
   - PRS channel 7 output pin: PA4
   - TX PRS channel: CH5
   - PRS channel 5 output pin: PD10

   Leave all other settings with their default values.
10. Repeat for the other example application.
11. Follow the rest of the steps in QSG106 to generate, build, and flash both the Z3Light and Z3Switch application images.

2.3.2 Setting up the Coexistence Backplane Board

1. Plug the Z3Light board into the CoEx1 position (J2) in the Coexistence Backplane Board. Follow the silk label for proper orientation.
2. Add the appropriate jumpers as shown in the following figure. This sets the GRANT line to always asserted, and allows the Z3Light to exercise the coexistence signals.

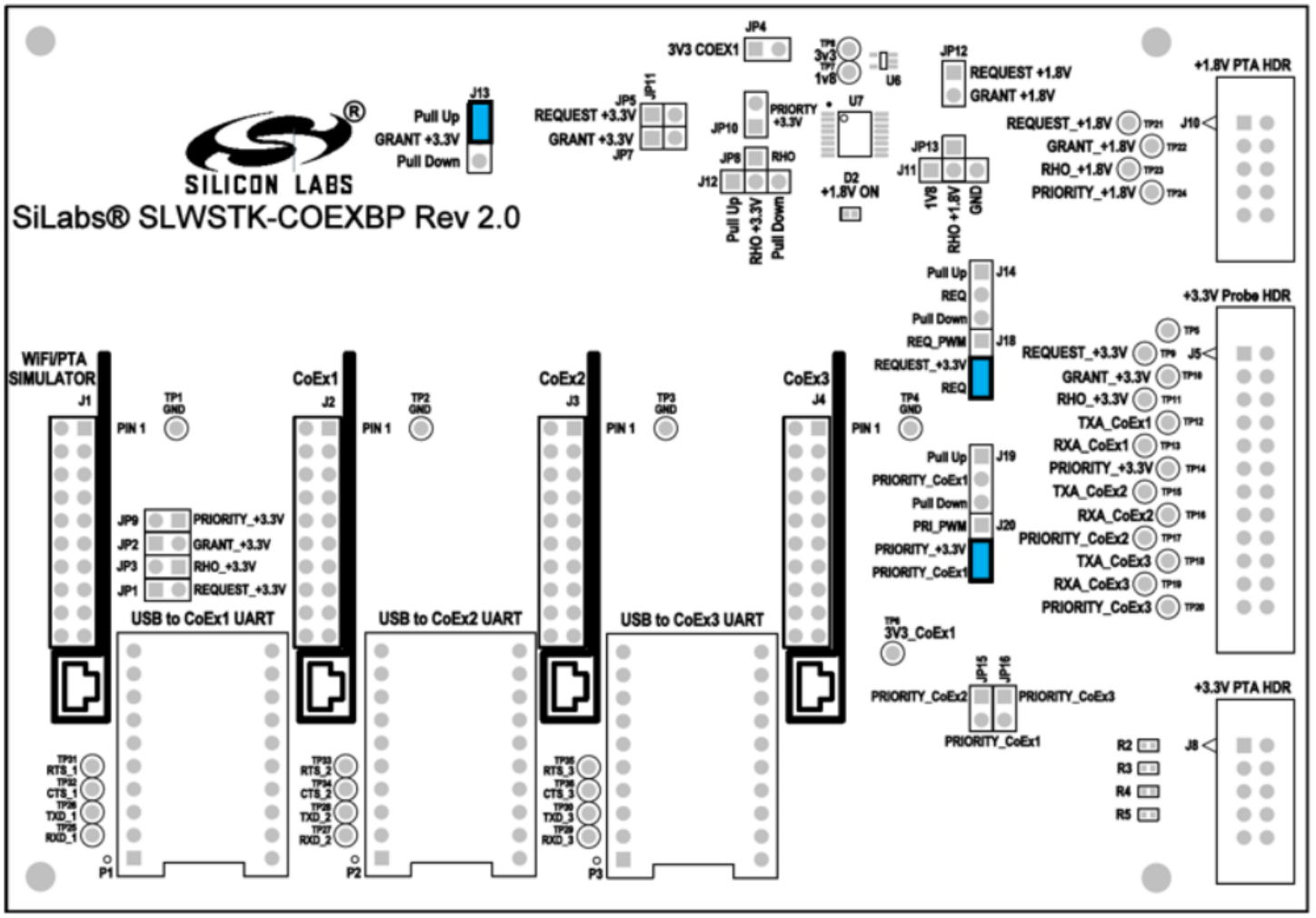


Figure 2.4. Jumper Configurations for Coexistence Testing

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 1

   GRANT_+3.3V – TP10 or J5 pin 3

   PRIORITY_+3.3V – TP14 or J5 pin 11

   TXA_CoEx1 – TP12 or J5 pin 7

   RXA_CoEx1 – TP13 or J5 pin 9

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 commands response.
3. In the console of the Z3Light, send the following command

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

   where NodeID is the node ID of the Z3Switch noted in step 2 above. 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 are 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.

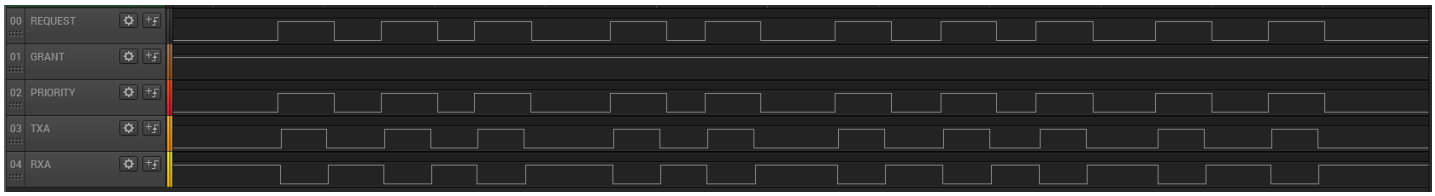


Figure 2.5. Logic Analyzer Result for Coexistence Testing

   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 representing the radio is in receive mode.

2.4 Using PTA Master within the Test Setup

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

To obtain the PTA Master application project, you must have the correct version of the Flex SDK (v2.2.2) installed. You must also add the correct board (BRD4161A) to the **My Products** list in Studio.

To add the correct board, go to Simplicity Studio's Launcher Perspective, and in the My Products section search for "BRD4161A" and click it to add it to the list.

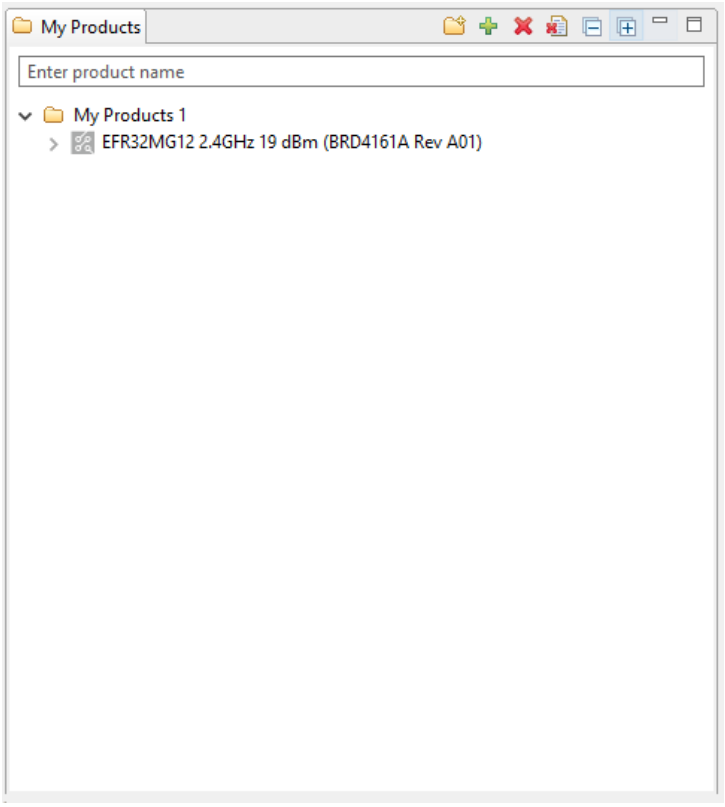


Figure 2.6. Selected Board Added to My Products

Next, make sure the Preferred SDK is set to Flex SDK 2.2.2. If this is not the case, click on the link to change the preferred SDK and select Flex SDK 2.2.2.

Figure 2.7. Preferred SDK Selected

If everything is set correctly, you should see the **IoTS PTA Master** application in the list of Examples.

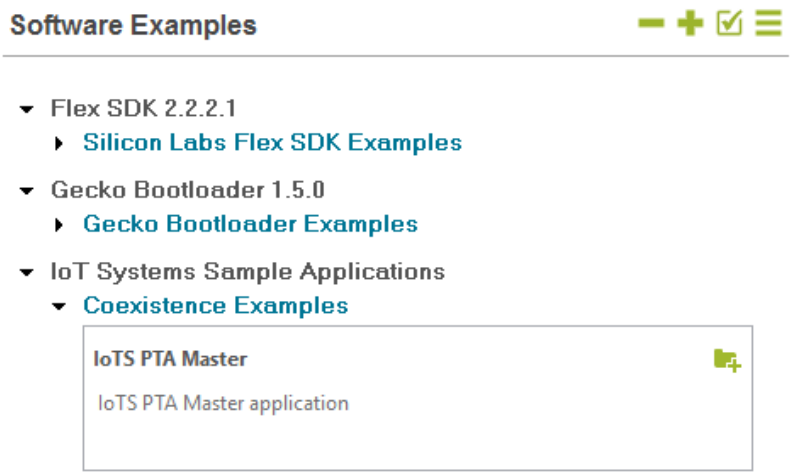


Figure 2.8. PTA Master Application in the Examples List

2.4.2 Building and Flashing the PTA Master Application

Click the PTA Master application shown in the Examples section. You are prompted whether you would like to create the application. Click [**Yes**].

Simplicity Studio creates a project with the application, and switches to the Simplicity IDE perspective.

An .isc file also opens. From this point on, the process is similar to that used for previous applications.

Follow the steps in *QSG106: Getting Started with EmberZNet PRO* to generate, build, and flash the PTA Master application onto one of the boards.

2.4.3 Setting up the PTA Master Device with the Coexistence Backplane Board

1. Add additional jumpers required to communicate with the PTA Master, as shown in the following diagram:

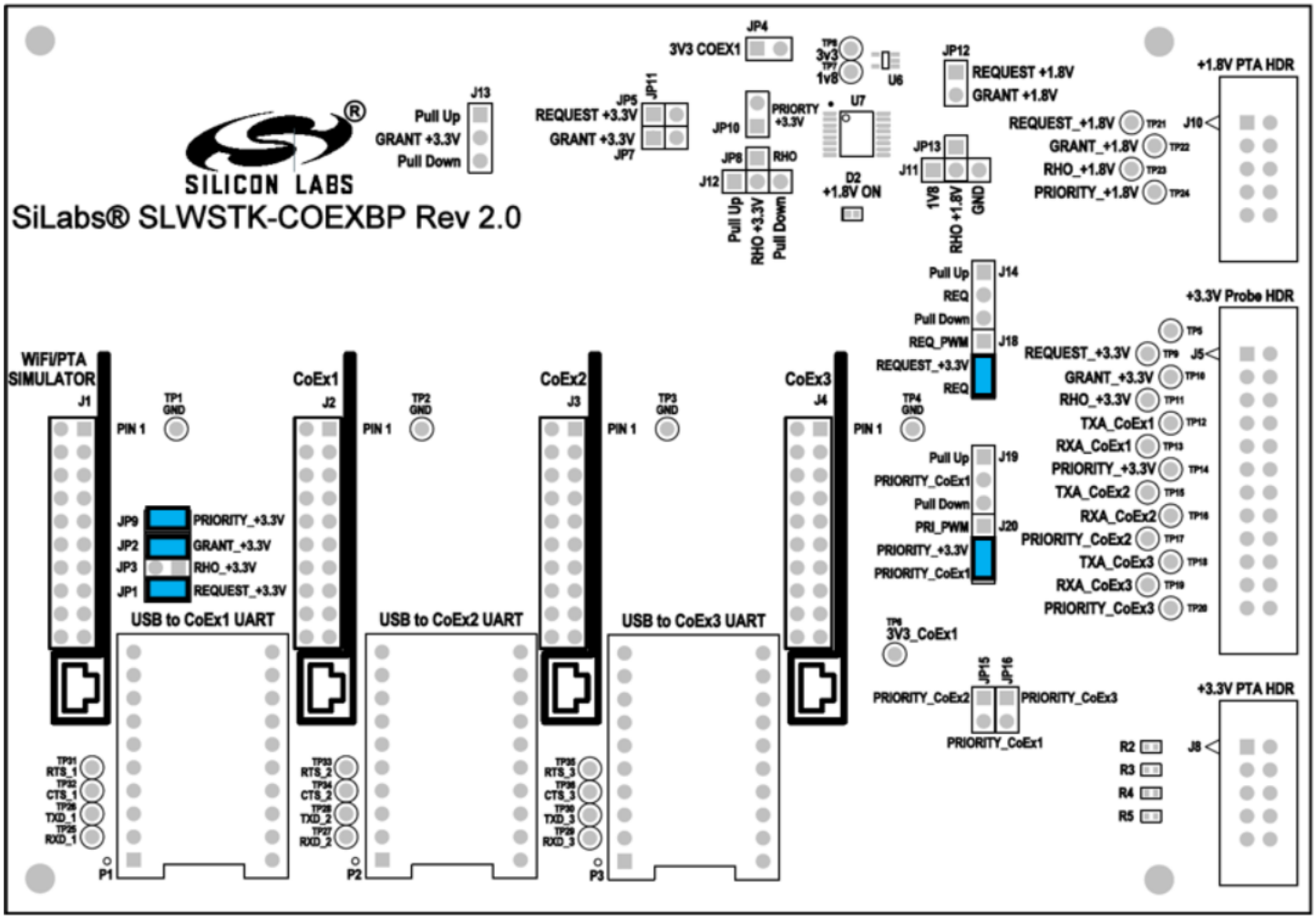


Figure 2.9. Jumper Configuration for Coexistence Testing with PTA Master Device

2. Plug PTA Master 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.

2.4.4 Configuring the PTA Master application to work with the Z3Light

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

2.4.5 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 commands response.
3. In the console of the Z3Light, send the following command

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

   where NodeID is the node ID of the Z3Switch noted in step 2 above. 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 are 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.

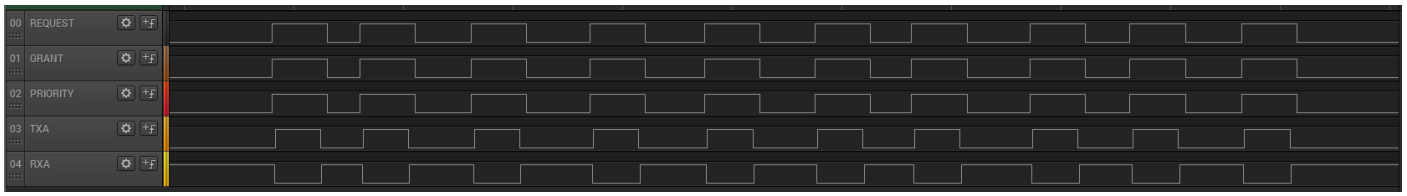


Figure 2.10. Logic Capture for Coexistence Testing with PTA Master Device

   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 Master is also asserted and de-asserted with REQUEST. In default settings, the PTA Master is configured to always GRANT when the slave 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 representing the radio is in receive mode.

2.5 Replacing PTA Master with an External Wi-Fi Device

Once evaluation has been completed using the PTA Master 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 Master device.
2. Connect the external Wi-Fi device coexistence signals to the appropriate device.
   - 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 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 EFR32MG1P and EFR32MG13P devices as well as the EFR32MG12P devices used with the Coexistence Development Kit, so this section also covers how to set up any of the other devices to work on the board.

When using the Coexistence Backplane Board for coexistence with the QFN48 radio modules, BRD4151A (EFR32MG1 in QFN48 package) or BRD4168A (EFR32MG13 in QFN48 package), or with the BGA125 radio module, BRD4161A (EFR32MG12 in BGA125 package), the AppBuilder coexistence-configuration must have any enabled PTA signals connected to the following GPIO:

EFR32 PTA Signal	EFR32MG1/MG13 GPIO	EFR32MG12 GPIO
RHO	PC11	PC11
REQUEST	PC10	PC10
GRANT	PF3	PC9
PRIORITY	PD12	PD12

It is also helpful to observe the FEM control signals by using the FEM driver plugin and directing TXA and RXA as follows:

FEM Driver Signals	EFR32 PRS Channel	EFR32 PRS Location	EFR32MG1/MG13 GPIO	EFR32MG12 GPIO
TXA	5	0	PD10	PD10
RXA	6	13	PD11	PD11

It is also helpful to observe the FRC_DFRAME and FRC_DOUT packet trace signals on the WSTK J101 header:

FRC Signal	EFR32MG1/MG13 WSTK J101	EFR32MG12
FRC_DFRAME	P20	N/A
FRC_DOUT	P22	N/A

The Coexistence Backplane Board has the connector and jumper options are shown in the following figure.

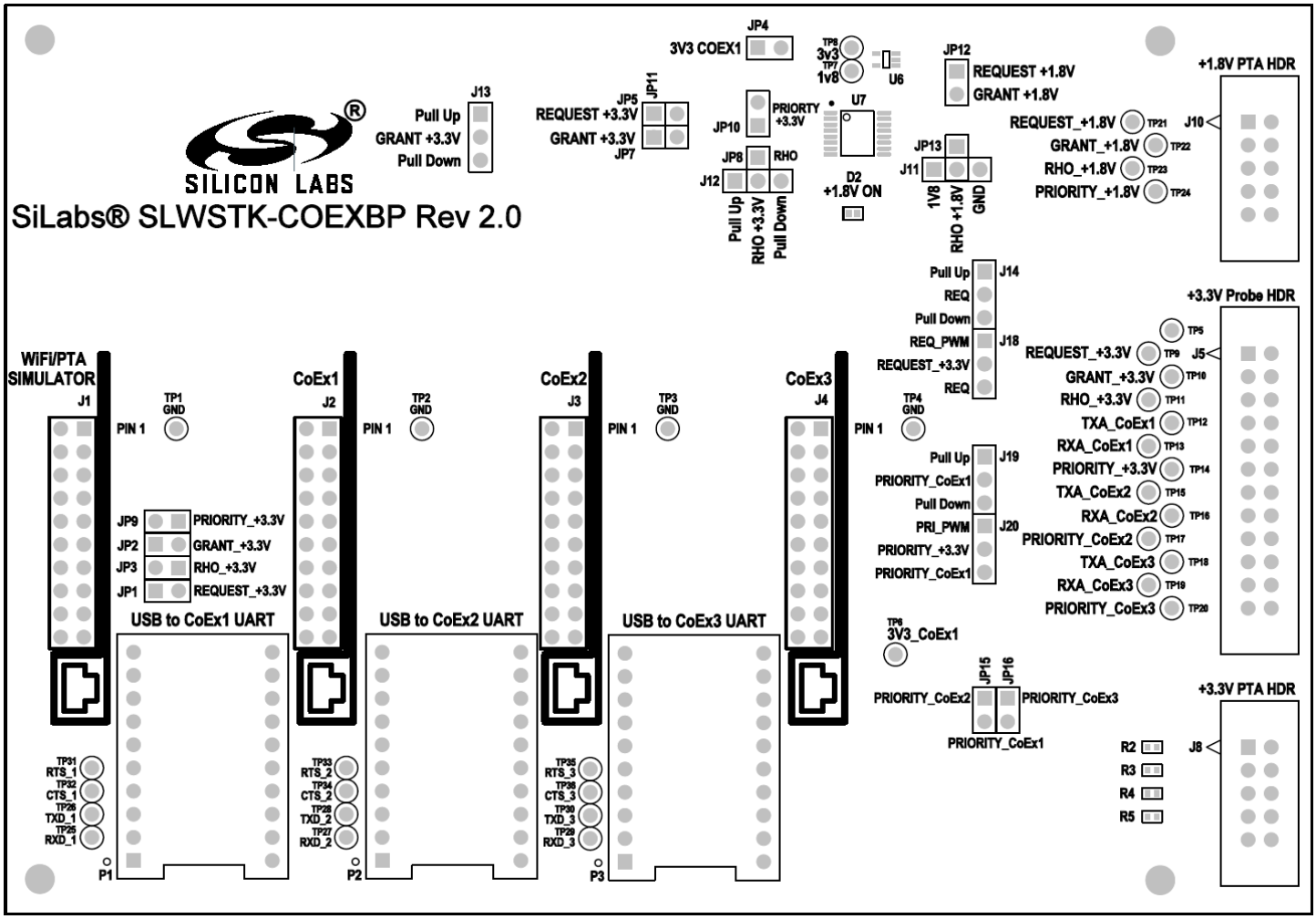


Figure 3.1. Coexistence Backplane Board Connector and Jumper Options

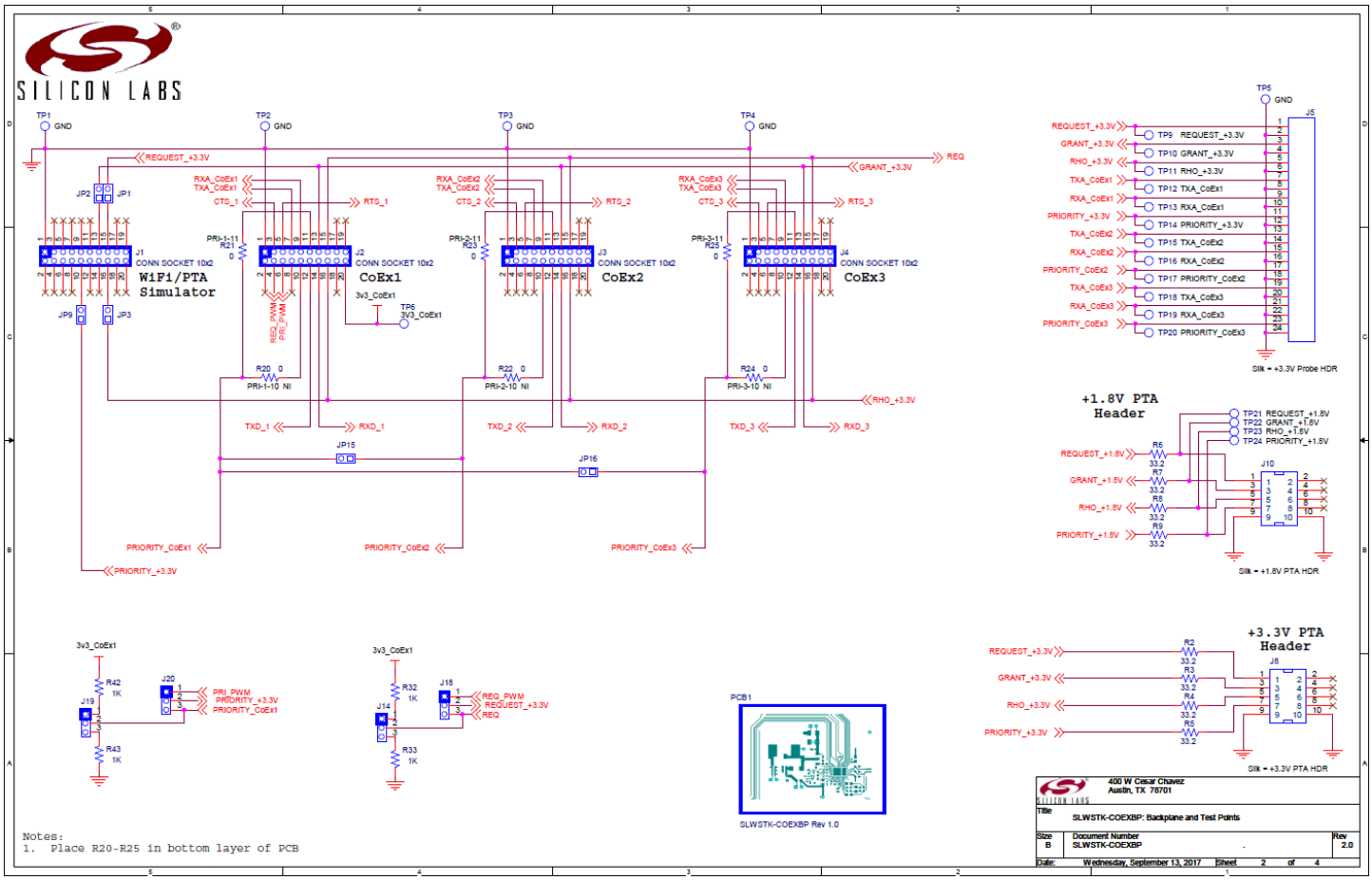


Figure 3.2. SLWSTK-COEXBP Schematic (1/3)

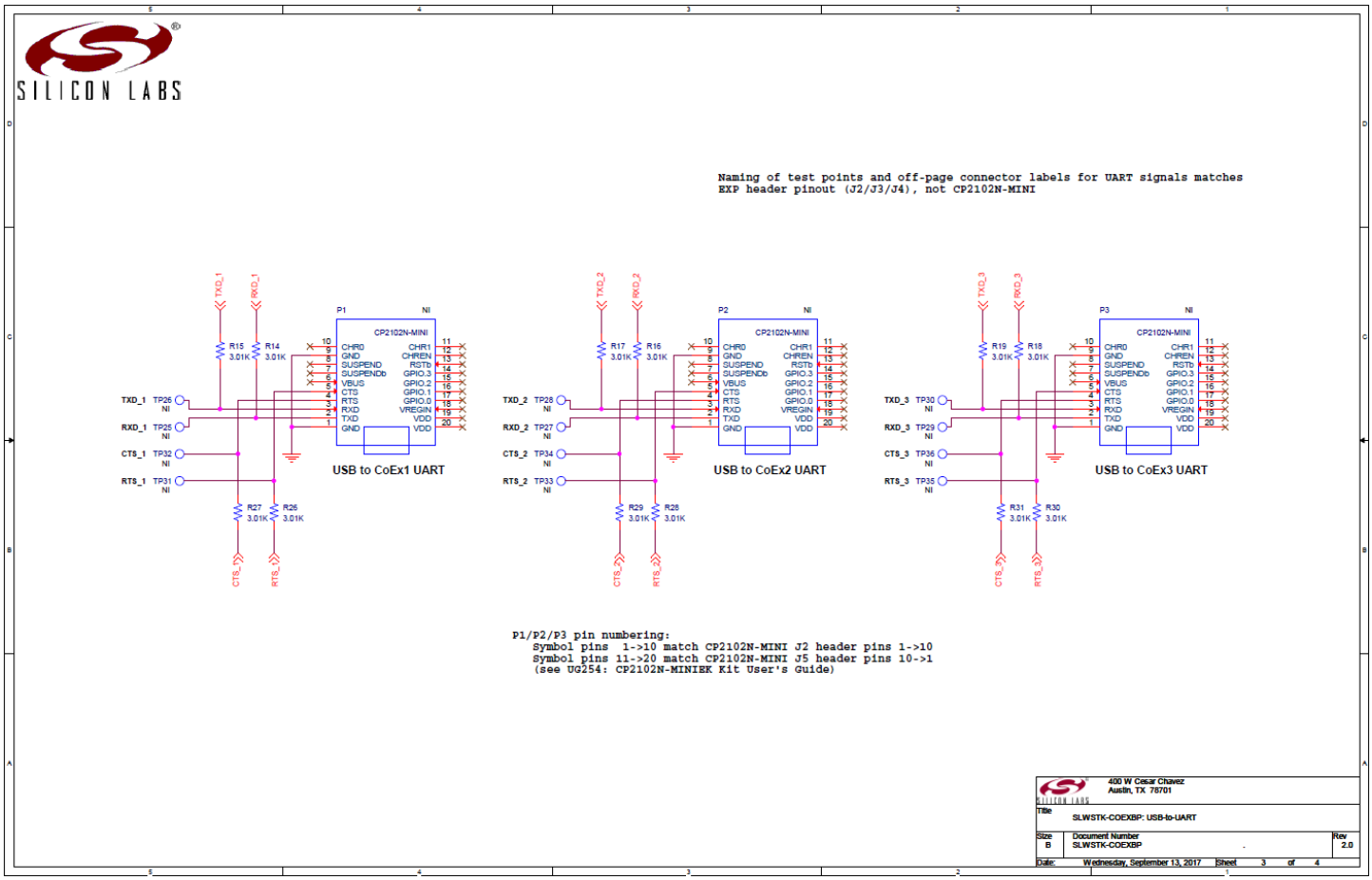


Figure 3.3. SLWSTK-COEXBP Schematic (2/3)

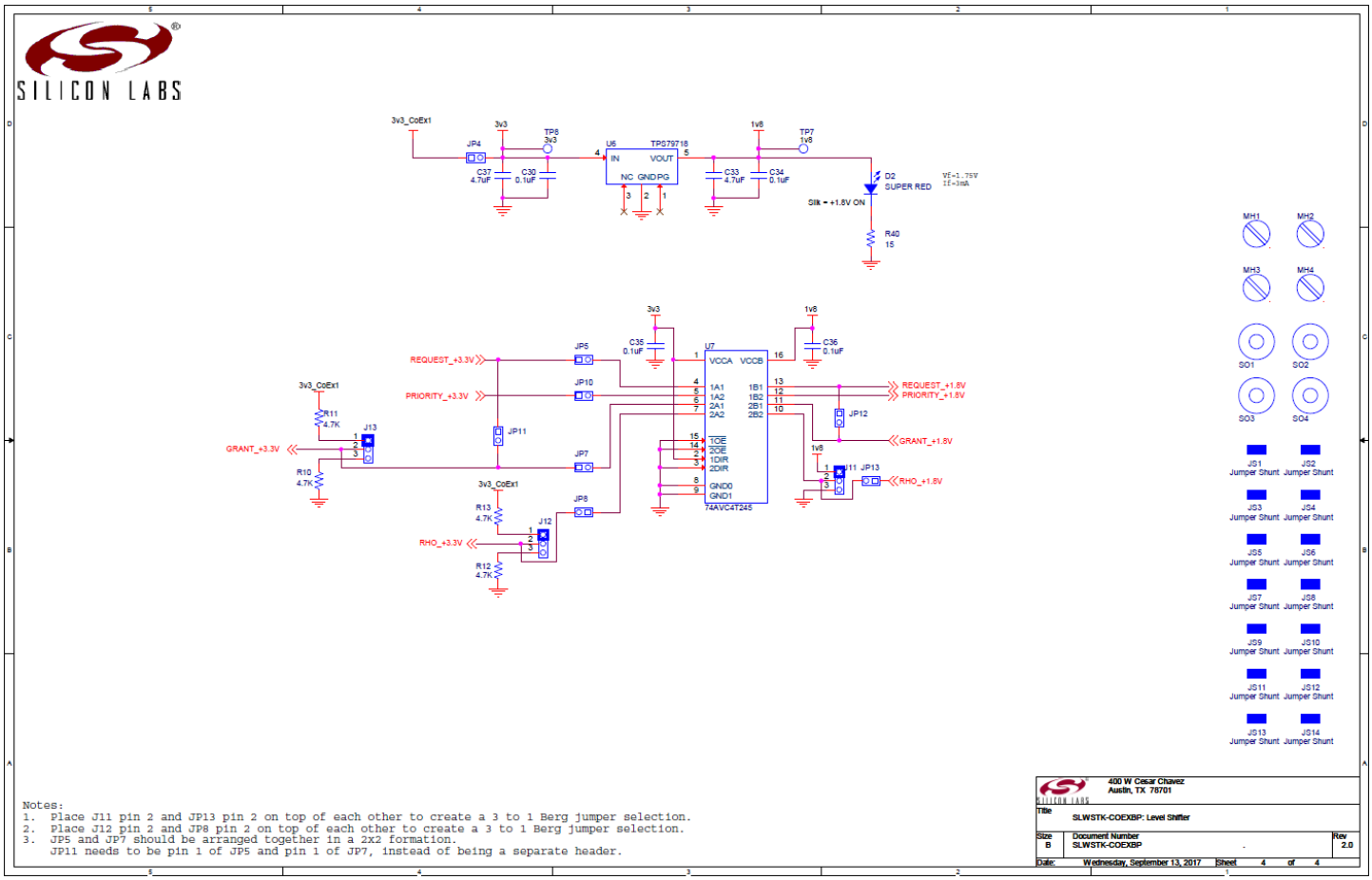


Figure 3.4. SLWSTK-COEXBP Schematic (3/3)

3.1 +3.3V or +1.8V I/O to Wi-Fi/PTA Device

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.

3.1.1 +3.3V I/O

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 figure.

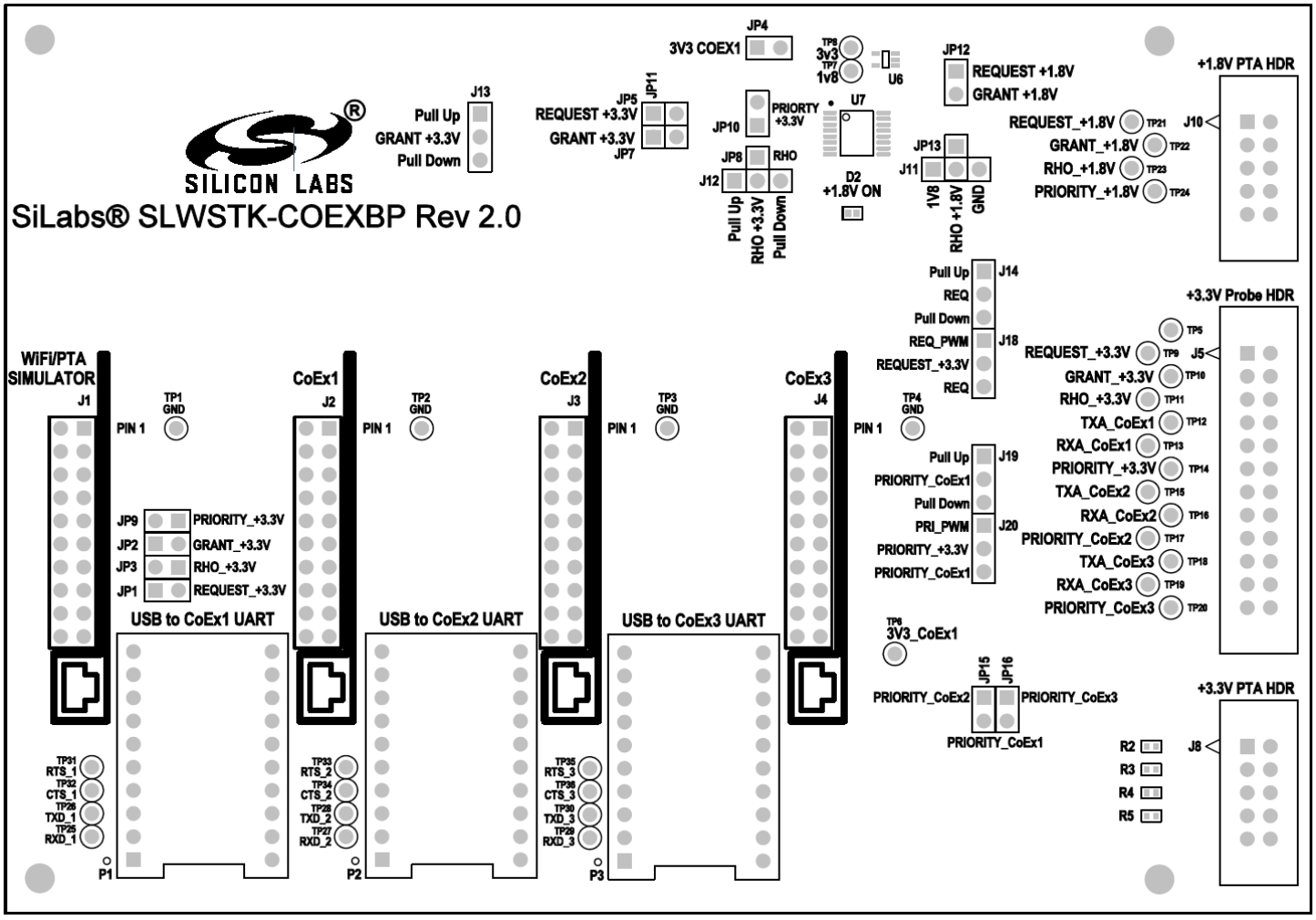


Figure 3.5. Coexistence Backplane Board Jumpers for +3.3V I/O Operation

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

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 3.3 Single-EFR32 Radio and section 3.4 Multi-EFR32 Radio for 2-Wire PTA with active-low REQUEST for these jumper options.

3.1.2 +1.8V I/O

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.

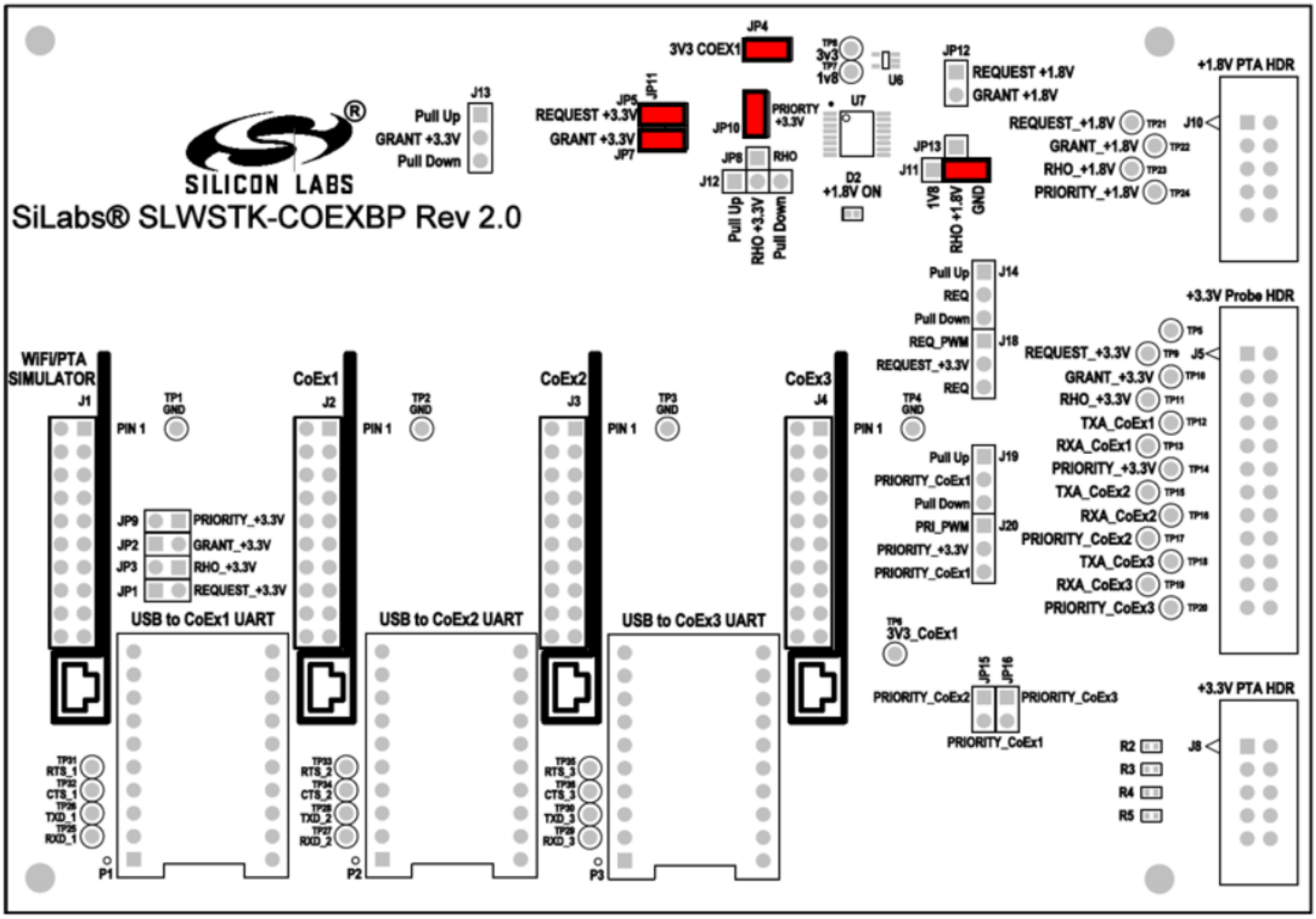


Figure 3.6. PTA Backplane Board Jumpers for +1.8V I/O Operation

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

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 3.3 Single-EFR32 Radio and section 3.4 Multi-EFR32 Radio for 2-Wire PTA with active-low REQUEST for these jumper options.

3.2 Logic Analyzer Observation

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

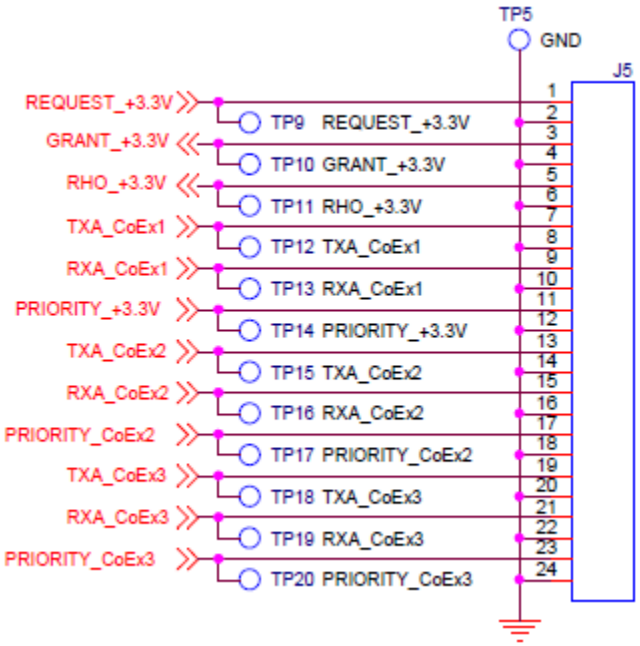


Figure 3.7. Coexistence Backplane Board Header for PTA Signal Logic Analyzer Connection

Where:

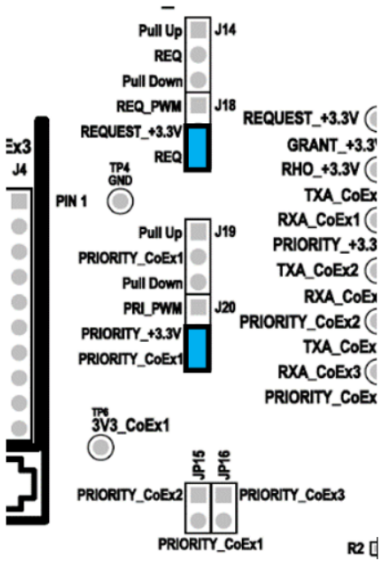
- CoEx1 refers to EFR32/WSTK connected into CoEx1 header (J2)
- CoEx2 refers to EFR32/WSTK connected into CoEx2 header (J3)
- CoEx3 refers to EFR32/WSTK connected into CoEx3 header (J4)

Note: Depending on jumper configurations, PRIORITY_+3.3 V can be PRIORITY_CoEx1 or wired-OR/-AND of multiple PRIORITY signals. See section 3.3 Single-EFR32 Radio and section 3.4 Multi-EFR32 Radio for 2-Wire PTA with active-low REQUEST for these PRIORITY_+3.3 V jumper options.

3.3 Single-EFR32 Radio

For a single EFR32 radio, be sure to:

1. Configure Board REQUEST and PRIORITY jumpers as follows:



2. Develop the desired PTA test application using AppBuilder.
3. Add the coexistence-configuration 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 Thread Coexistence with Wi-Fi*).
4. Add FEM driver 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) as shown in the following figure.

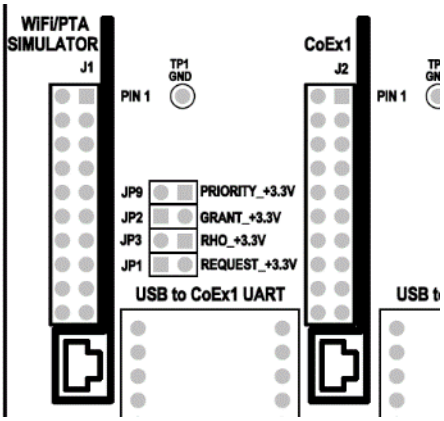


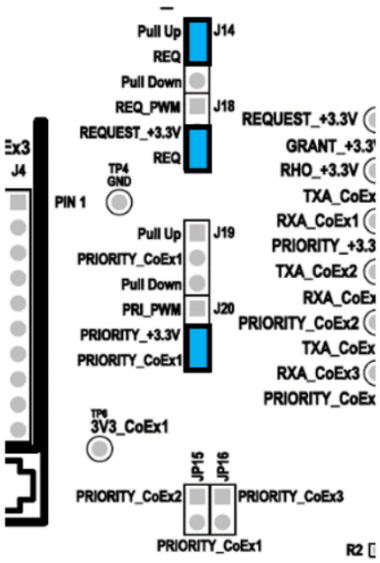
Figure 3.9. Coexistence Backplane Board Header for Single EFR32 Radio Testing

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-plugin only needs to be modified for the PTA GPIOs assigned on target hardware.

3.4 Multi-EFR32 Radio for 2-Wire PTA with active-low REQUEST

For multi- EFR32 radio using only REQUEST (active-low) and GRANT, be sure to do the following:

1. Configure Board REQUEST and PRIORITY (optional for debug) jumpers as follows:



2. Develop desired PTA test applications using AppBuilder.
3. Add coexistence-configuration 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 Thread Coexistence with Wi-Fi*).
4. Add the FEM driver 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) as shown in the following figure.

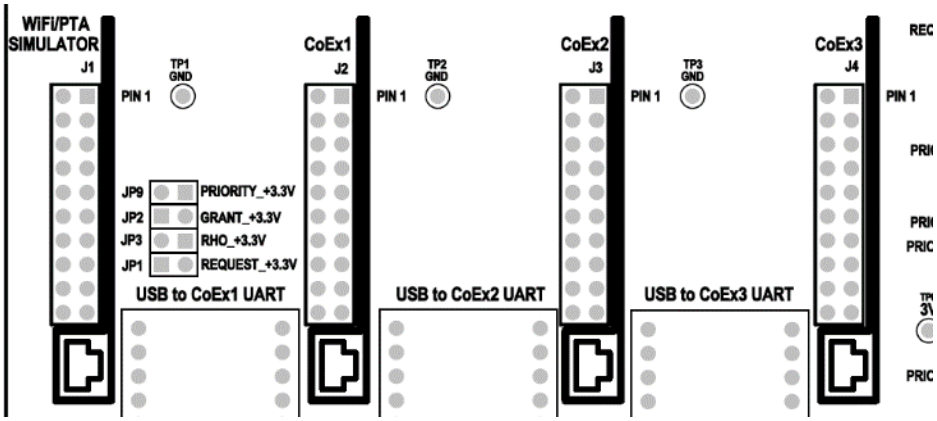


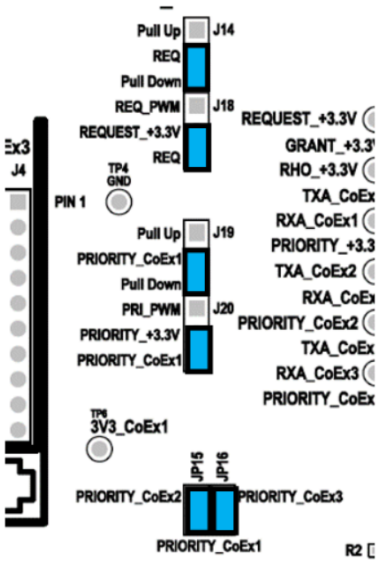
Figure 3.10. Coexistence Backplane Board Headers for Multi-EFR32 Radio Testing for 2-Wire PTA with active-low REQUEST

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-plugin only needs to be modified for the PTA GPIOs assigned on target hardware.

3.5 Multi-EFR32 Radio for typical 3-Wire PTA

For multi- EFR32 radio using typical 3-Wire PTA, REQUEST (active-high), PRIORITY (active-high), and GRANT (active-low), be sure to:

1. Configure Board REQUEST and PRIORITY jumpers as follows:



2. Develop desired PTA test applications using AppBuilder.
3. Add coexistence-configuration 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 Thread Coexistence with Wi-Fi*).
4. Add the FEM driver 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) as shown in the following figure.

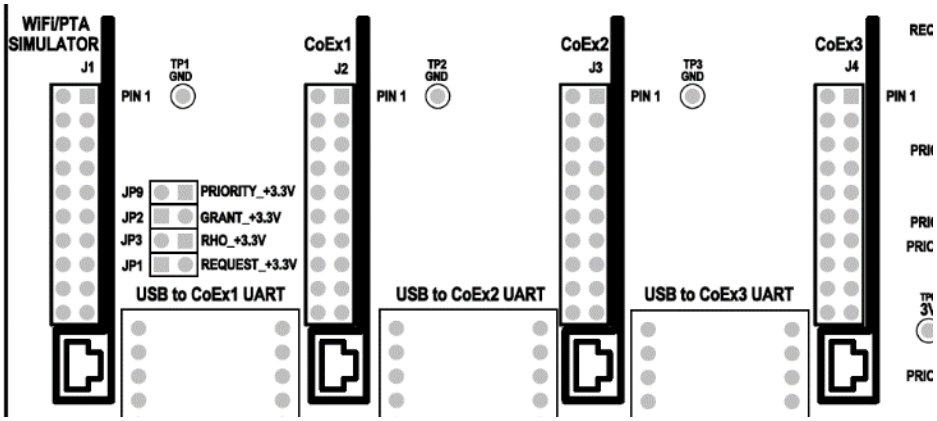


Figure 3.11. Coexistence Backplane Board Headers for Multi-EFR32 Radio Testing for typical 3-Wire PTA

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-plugin only needs to be modified for the PTA GPIOs assigned on target hardware.

3.6 Testing xNCP Applications

If testing xNCP (customized NCP) images with PTA support, initial testing may benefit from enabling the EFR32 serial connection through the WSTK EXP header, connecting to a UART-to-USB adapter, and testing using a PC-based gateway host application to drive the EFR32's xNCP image. This can be accomplished through the following procedure:

1. Order a CP2102N-MINIEK for each EFR32 xNCP on the Coexistence Backplane Board.

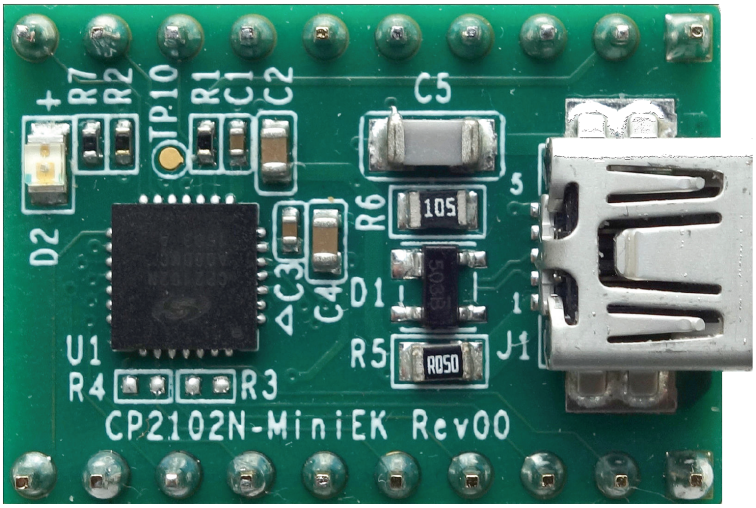


Figure 3.11. CP2102N-MINIEK

2. Install the CP2102 drivers onto the test PC.
3. Solder each CP2102N-MINIEK to Coexistence Backplane Board's CoEx1 (P1), CoEx2 (P2), and/or CoEx3 (P3) as follows:

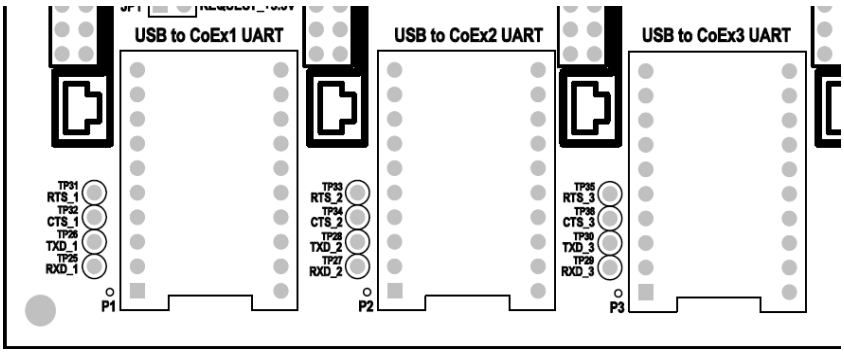


Figure 3.12. Coexistence Backplane Board Headers for Testing NCP Applications

Note: Each CP2102N-MINIEK is connected to its respective EFR32/WSTK Board as follows:

CP2102N-MINIEK Signal	J2, J3, or J4 header
GND	Pin 1 (GND)
TXD	Pin 14 (UART_RX, PA1)
RXD	Pin 12 (UART_TX, PA0)
CTS	Pin 5 (UART_RTS, PA3)
RTS	Pin 3 (UART_CTS, PA2

Note: On BRD4161A (EFR32MG12), PB6, PB7, PD8, and PD9 drive these same signals. However, PB6, PB7, PD8, and PD9 cannot be driven by USART0. As such, while xNCP with PTA is supported on EFR32MG12, xNCP with PTA is more easily evaluated on Coexistence Backplane Board using BRD4151A or BRD4168A.

4. Within AppBuilder, under **Core plugins** enable the NCP-UART plugin and set **Flow Control Type** to desired (**Hardware or Software)**.
5. Enable and configure the Coexistence-Configuration, FEM driver, and HAL-Configuration plugins.
6. On the **Other** tab, under **Additional Macros**, ensure that the **EZSP_UART** and **NO_USB** macros are defined and selected.

7. After generating and compiling the xNCP application, program the EFR32.
8. Install the EFR32/WSTK into the Coexistence Backplane Board, then connect the associated CP2102N-MINIEK to the PC.
9. Note the COM port assigned to CP2102N-MINIEK.
10. Start the PC-based gateway application as appropriate for the desired flow control:
    - Hardware flow control at 115.2 Kb: using `-n 0 –p COMx` arguments, where `x` is the assigned CP2102N-MINIEK COM port.
    - Software flow control at 57.6 Kb: using `-n 1 –p COMx` arguments, where `x` is the assigned CP2102N-MINIEK COM port.
11. Test the PTA solution using the PC-gateway application.

4. Throughput Library

The throughput commands allow you to test your application's network throughput.

4.1 Throughput Library Commands

Table 4.1. List of Commands Available in the Throughput Library

Command	Command Description	Arguments		
		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	INT8U	Interval in ms
`plugin throughput set-timeout`	Set the test timeout	timeout	INT32U	Timeout in ms
`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 all parameters: • NodeID • Count • Interval • Packet Length • Max In-Flight • APS Options • Test Timeout	nodeId	INT16U	Destination nodeId
		count	INT32U	Packets to send
		interval	INT32U	Interval in ms
		packet	INT8U	Packet length in bytes
		inflight_count	INT8U	Packets in flight
		apsOptions	INT16U	APS Options
		timeout	INT32U	Timeout in ms
`plugin throughput set-count`	Set the number of packets to send	count	INT32U	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

Command	Command Description	Arguments		
		Name	Type	Description
`plugin throughput set-all`	Set all parameters: • NodeID • Count • Interval • Packet Length • Max In-Flight • APS Options • Test Timeout	nodeId	INT16U	Destination nodeId
		count	INT32U	Packets to send
		interval	INT32U	Interval in ms
		packet	INT8U	Packet length in bytes
		inflight_count	INT8U	Packets in flight
		apsOptions	INT16U	APS Options
		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 need to be defined. The simplest way to achieve this is by using 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 28.

A sample command is `plugin throughput set-all <NodeID> 10 0 127 1 0 100000`, where `NodeID` needs to be defined as the nodeID of the receiving node. Note: to achieve maximum throughput, the interval field should to be set 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 command `plugin throughput stop`. Once the test is complete, you can print the results using `plugin throughput print-result`. This outputs 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 plugin also tracks some counters to provide visibility and help with debugging purposes. Use `plugin throughput print-counters`. The results are outputted 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 Thread Coexistence with Wi-Fi* and are also documented in the stack API documentation.

5. PTA Master Application

The PTA Master application is custom application built on the Flex stack, designed to help customers evaluate coexistence features on the silicon labs devices. The Zigbee radios act as slave devices, and require a master device to react and respond to coexistence requests. The PTA Master application implements this and allows users control to the behavior of the PTA Master to evaluate the corresponding behavior of the slave device.

5.1 Commands in the PTA Master Application

Table 5.1. List of Commands Available in the PTA Master 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 Master Application

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

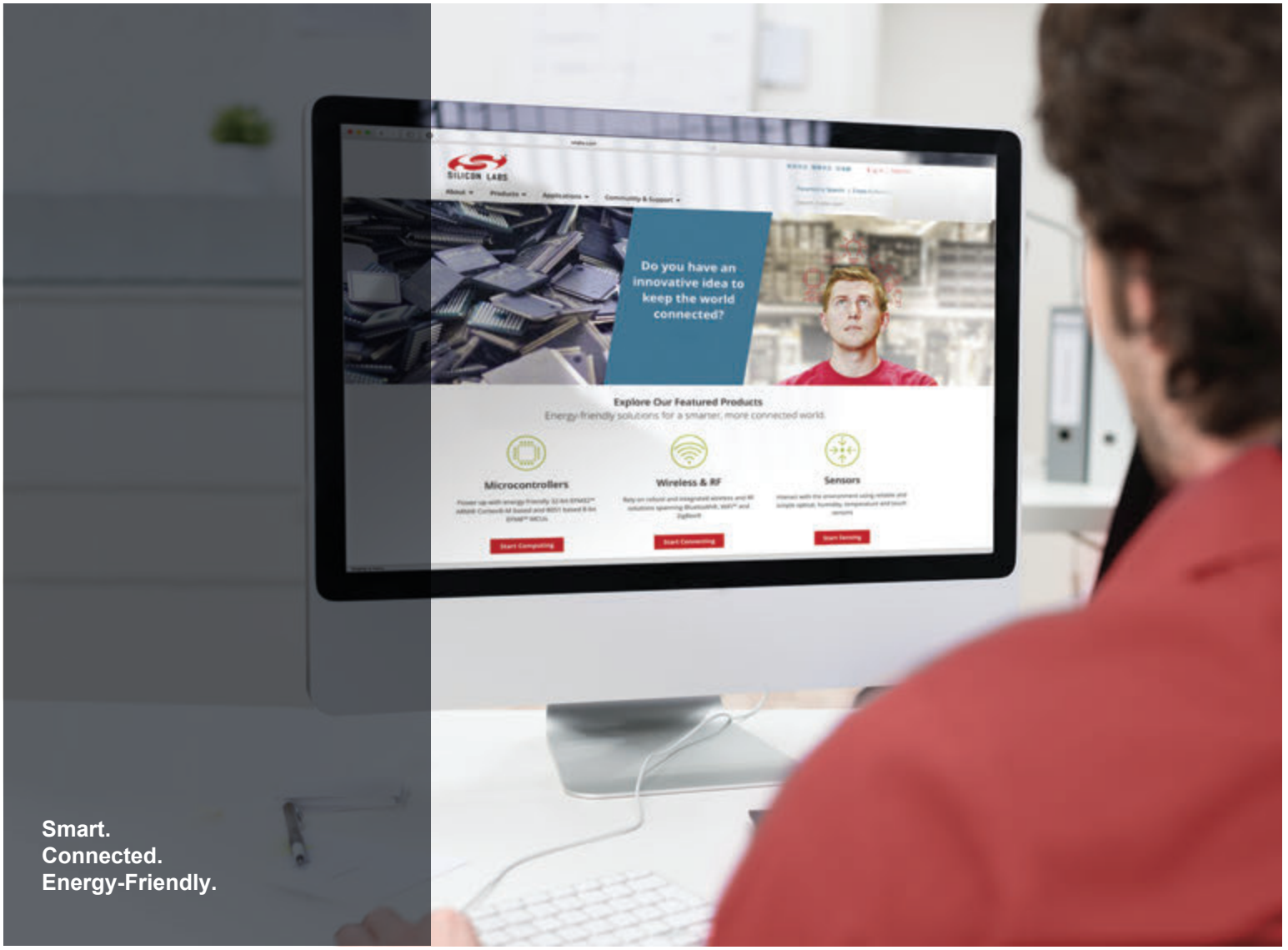
- REQUEST is set to Active HIGH
- GRANT is set to Active HIGH
- PRIORITY is set to Active HIGH
- RHO is set to Active HIGH, but the feature is disabled and the pin is set to High Z mode
- GRANT Abort, GRANT Delay, and GRANT Deny are disabled
- No stream is active

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

Use the `configuration` command to print out 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 Master application can also hinder network traffic to test behavior of the slave 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.



Products
www.silabs.com/products

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

SILICON LABS®

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

http://www.silabs.com