AN1017: Zigbee® and Silicon Labs® Thread Coexistence with Wi-Fi

This application note describes the impact of Wi-Fi on Zigbee and Silicon Labs Thread and methods to improve coexistence with Wi-Fi. These techniques are applicable to the Mighty Gecko family (EFR32MGx). First, design considerations to improve coexistence without direct interaction between Zigbee/Silicon Labs Thread and Wi-Fi radios are described. Next, Silicon Lab's Packet Traffic Arbitration (PTA) support to coordinate 2.4GHz RF traffic for co-located Zigbee/Silicon Labs Thread and Wi-Fi radios is described. Finally, methods for using the coexistence CLI commands are discussed.

KEY POINTS

- Wi-Fi impact on Zigbee/Silicon Labs Thread
- Improving unmanaged coexistence
- Implementing managed coexistence
- Using the coexistence CLI commands

This application note describes EFR32 Zigbee and Silicon Labs Thread coexistence support for EmberZNet PRO 6.6.1.0 and Silicon Labs Thread 2.10.1.0. See Section 9 Revision History for a summary of key changes in previous revisions of this application note.

Additional details about the implementation of managed coexistence and test results are available in an expanded version of this application note, AN1017-NDA: ZigBee and Thread Coexistence with Wi-Fi, available under non-disclosure from Silicon Labs technical support.

Contents

1 Introduction

The 2.4GHz ISM (Industrial Scientific and Medical) band supports Wi-Fi (IEEE 802.11b/g/n), Zigbee/Silicon Labs Thread (IEEE 802.15.4), *Bluetooth*®, and Bluetooth low energy. The simultaneous and co-located operation of these different 2.4GHz radio standards can degrade performance of one or more of the radios. To improve interference robustness, each of the 2.4GHz ISM radio standards support some level of collision avoidance and/or message retry capability. At low data throughput rates, low power levels, and/or enough physical separation, these 2.4GHz ISM standards can co-exist without significant performance impacts. However, recent customer trends are making coexistence more difficult.

- Increased Wi-Fi transmit power level for “extended range”
  - +30dBm Wi-Fi Access Points are now common.
- Increased Wi-Fi throughput
  - Depending on achievable SNR, high throughput requirements for file transfers and/or video streaming may result in high Wi-Fi duty cycle within 2.4GHz ISM band.
- Integrating Wi-Fi, Zigbee, Silicon Labs Thread, and Bluetooth low energy into the same device for gateway functionality
  - This is required by Home Automation and Security applications and provides easier end-node commissioning using Bluetooth low energy.

This application note describes the impact of Wi-Fi on Zigbee and Silicon Labs Thread and methods to improve coexistence with Wi-Fi on two Silicon Labs integrated circuits—the EM35x/EM358x and the Mighty Gecko family (EFR32MGx).

- Section 3 Unmanaged Coexistence describes design considerations to improve coexistence without direct interaction between Zigbee/Silicon Labs Thread and Wi-Fi radios.
- Section 4 Managed Coexistence describes Silicon Lab’s PTA (Packet Traffic Arbitration) support to coordinate 2.4GHz RF traffic for co-located Zigbee/Silicon Labs Thread and Wi-Fi radios. PTA support is available for the EFR32MGx only.
- Section 5 Coexistence CLI Console Commands describes how to use CLI commands in Host and SoC applications.

Note: Current Bluetooth and Bluetooth low energy solutions operate at less than +10dBm transmit power level and implement automatic frequency hopping (AFH). Even with Bluetooth at +10dBm, Silicon Labs’ testing indicates minimal impact from co-located Bluetooth on 802.15.4 performance. 802.15.4 device join success remains at 100% and 802.15.4 message loss remains at 0%. 802.15.4 MAC retries increase slightly due to occasional co-channel and adjacent channel collisions between the Bluetooth AFH and fixed 802.15.4 channel. However, the Wi-Fi coexistence discussion and solutions described in this application note can be applied equally well to Bluetooth coexistence. As such, all following solutions presented for “Wi-Fi Coexistence” can be applied to “Wi-Fi/Bluetooth Coexistence”.

2 Wi-Fi Impact on Zigbee/Silicon Labs Thread

World-wide, Wi-Fi (IEEE 802.11b/g/n) supports up to 14 overlapping 20/22MHz bandwidth channels across the 2.4GHz ISM band with transmit power levels up to +30dBm. Similarly, 2.4GHz Zigbee and Silicon Labs Thread (based on IEEE 802.15.4) support 16 non-overlapping 2MHz bandwidth channels at 5MHz spacing with transmit powers up to +20dBm. These Wi-Fi and Zigbee/Silicon Labs Thread channel mappings are shown in the following figure.

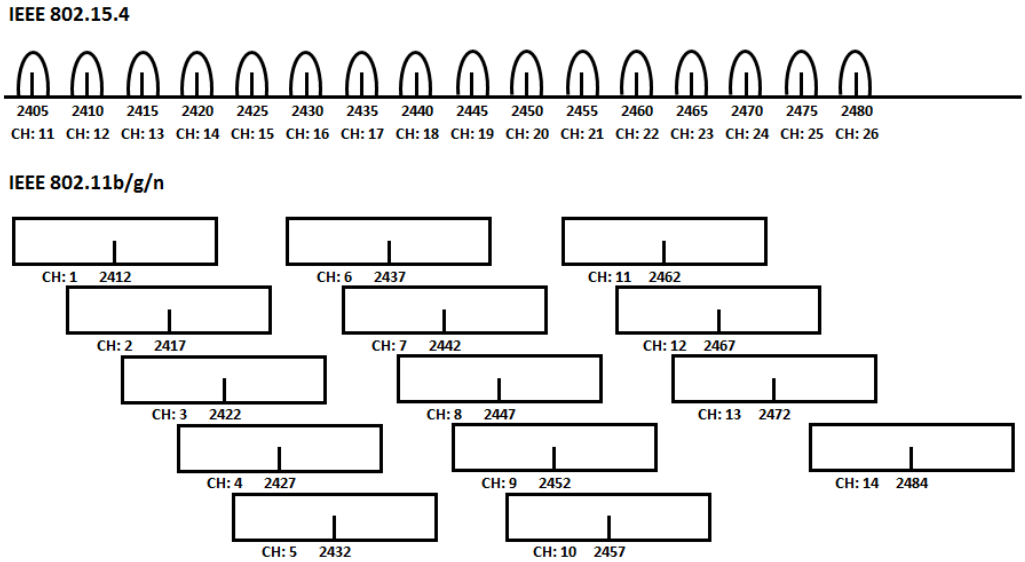


Figure 1. 802.15.4 and 802.11b/g/n Channel Mapping (World-Wide)

Actual channels available vary by country. For example, in the USA, Wi-Fi channels 1 through 11 are available and Zigbee channels 11 through 26 are available, although channels 25 and 26 require reduced transmit power levels to meet FCC requirements (North America only).

To better understand the effects of Wi-Fi on Zigbee/Silicon Labs Thread, Silicon Labs measured the impact of a 100% duty-cycled 802.11n (MCS3, 20MHz bandwidth) blocker transmitting at various power levels while receiving an 802.15.4 message transmitted at power level sufficient to achieve 1% PER (receive sensitivity). The results for co-channel, adjacent channel, and “far-away” channel are shown in the following figure. All 802.11n and 802.15.4 power levels are referenced to the Silicon Labs’ Mighty Gecko (EFR32MG1P232F256GM) RF input. The test application was developed using Silicon Labs’ EmberZNet PRO (Zigbee) stack with NodeTest running on the EFR32 DUT (Device Under Test) and a test script to control the DUT and RF test equipment. Because this is an 802.15.4 focused test, results are identical for Wi-Fi blocking Silicon Labs Thread.

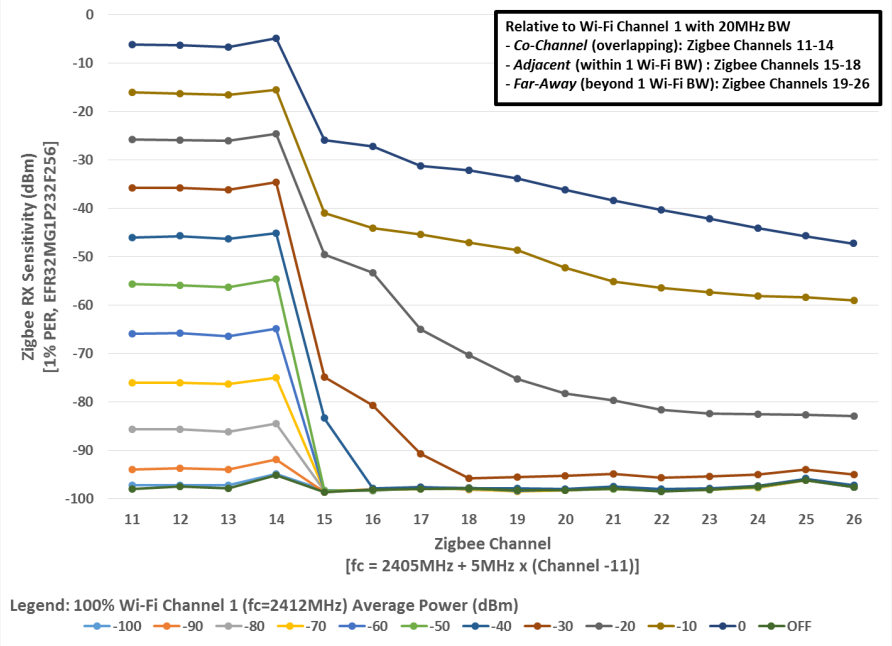


Figure 2. 802.15.4 Receive Sensitivity with 100% Duty Cycled 802.11n (MCS3/20MHz) Wi-Fi Blocker

These are the key observations about the impact of Wi-Fi on Zigbee/Silicon Labs Thread from the figure above.

Co-Channel (Zigbee overlapping Wi-Fi):

- At channel 12, Zigbee can receive an 802.15.4 signal down to 6dB weaker than aggregate Wi-Fi transmit power (100% duty cycle).
  - This receive sensitivity limitation impacts both co-located and remote, not co-located, 802.15.4 radios.
- At channel 12, Zigbee with and without LNA (Low Noise Amplifier) can receive an 802.15.4 signal down to 6dB weaker than aggregate Wi-Fi transmit power (100% duty cycle).
- 802.15.4 transmits can also be blocked by Wi-Fi transmit power tripping the 802.15.4 -75dBm CCA (Clear Channel Assessment) threshold.

Adjacent Channel (Zigbee within one Wi-Fi bandwidth):

- At channel 15, Zigbee can receive a -85dBm 802.15.4 signal with -42dBm or weaker Wi-Fi transmit power (100% duty cycle).
  - Maximum receive sensitivity is attained at -45dBm or weaker Wi-Fi transmit power (100% duty cycle).
- At channel 15, Zigbee without LNA can receive a -85dBm 802.15.4 signal with -42dBm or weaker Wi-Fi transmit power (100% duty cycle); -46dBm or weaker with LNA enabled.

"Far-Away" Channel (Zigbee beyond one Wi-Fi bandwidth):

- At channel 25, Zigbee can receive a -85dBm 802.15.4 signal with -22dBm or weaker Wi-Fi transmit power (100% duty cycle).
  - Maximum receive sensitivity is attained at -31dBm or weaker Wi-Fi transmit power (100% duty cycle).
- At channel 25, Zigbee without LNA can receive a -85dBm 802.15.4 signal with -25dBm or weaker 100% Wi-Fi transmit power (100% duty cycle); -29dBm or weaker with LNA enabled.

In a real-world environment, Wi-Fi is typically not 100% duty cycle and only approaches 100% duty cycle during file transfers or video stream in low Wi-Fi SNR (Signal to Noise Ratio) conditions. As seen in the figure above, the EFR32MGx receive sensitivity varies as the Wi-Fi blocker turns ON/OFF. The net result is the ability to see weaker signals when Wi-Fi is OFF, but not when strong Wi-Fi is ON (actively transmitting).

The following figure illustrates the receive range of a node (blue node) near a strong Wi-Fi transmitter. Relative to the blue 802.15.4 node, the area inside the green circle represents the receive range when Wi-Fi is ON. The area between the green and yellow circles represents the receive range when Wi-Fi is OFF. From this figure:

- The green node is always receivable by the blue node.
- The yellow node is only receivable by the blue node when Wi-Fi is OFF.
- The red node is never receivable by the blue node.
- The yellow and red nodes are always receivable by the green node.

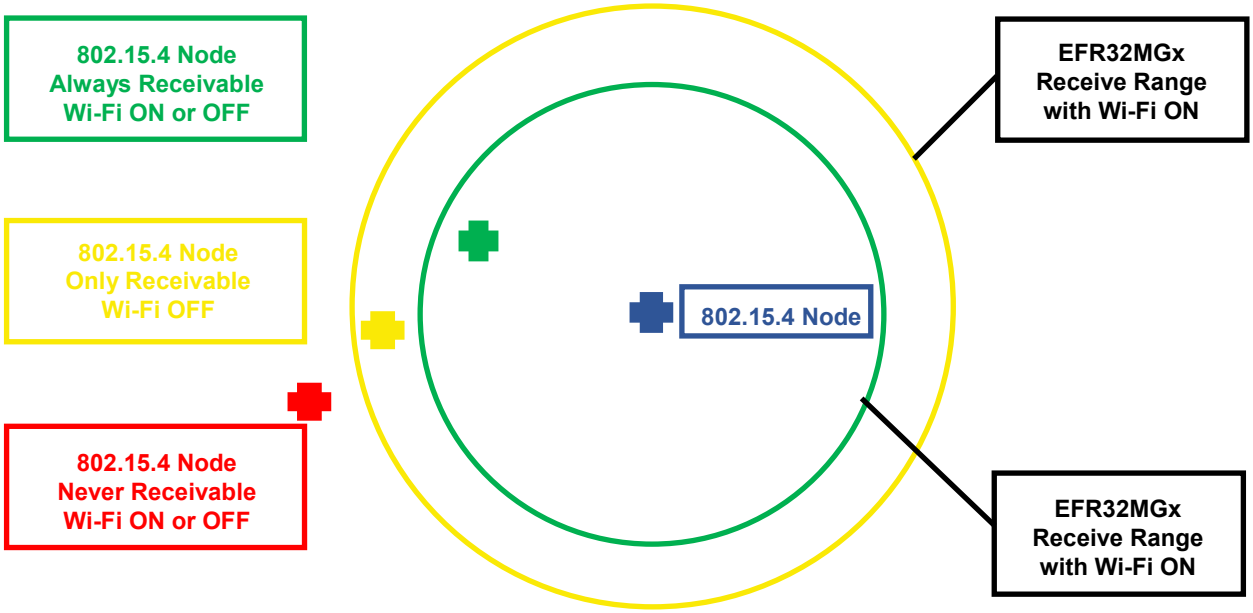


Figure 3. EFR32MGx Receiver Desensitized when Wi-Fi Transmitting

Depending on each node's type (Coordinator, Router, or End Device) and the Wi-Fi duty cycle, the impact of strong Wi-Fi turning ON/OFF will vary.

In a Zigbee network:

- Coordinator: Tasked with network creation, the control of network parameters, and basic maintenance, in addition to performing an application function, such as aggregating data or serving as a central control point or gateway.
- Router: In addition to running an application function, a Router can receive and retransmit data from other nodes.
- End Device: Typically, a battery-powered device (Sleepy End Device or SED) running an application function and able to talk to a single parent node (either the Coordinator or a Router). End Devices cannot relay data from other nodes.

In a Silicon Labs Thread network:

- Border Router: Provides Silicon Labs Thread network node connectivity to other devices in external networks (for example, Internet access).
- Router: In addition to running an application function, a Router can receive and retransmit data from other nodes and provide join and security capability. When routing function is not needed by network, a Router can downgrade to a Router-Eligible End Device (REED).
- REED: In addition to running an application function, a REED can receive and retransmit data from other nodes. When additional routers needed by network, a REED can upgrade to a Router.
- SED: Typically, a battery-powered device running an application function and able to talk to a single parent node (either a Border Router, Router, or REED). SEDs cannot relay data from other nodes.

Two Zigbee cases are considered below, but many other cases are possible.

Case 1: Zigbee Coordinator near strong Wi-Fi plus three end-nodes

For this case, the figure above is composed of:

- Coordinator: Blue node
- End Devices: Green, Yellow, and Red nodes

In this simple network, each end device attempts to join the network formed by the coordinator. However, the red node is outside of receive range and cannot join. With Wi-Fi OFF, both the green and yellow nodes successfully join the network and have no issues sending messages to the Coordinator. Regardless of Wi-Fi ON/OFF duty cycle, the green node remains successful sending messages to the Coordinator.

With Wi-Fi ON/OFF at low-duty cycle, some messages from the yellow node are periodically blocked, but Zigbee retry mechanisms are effective in getting the messages to the coordinator. However, with Wi-Fi ON/OFF at high-duty cycle, many messages from the yellow node are blocked and Zigbee retry mechanisms may be exhausted. Even when retry mechanisms are successful, the message latency

increases. If the yellow node is a battery-powered Sleepy End Device, it must remain active longer to execute retries, reducing battery life.

Case 2: Zigbee Coordinator near strong Wi-Fi, Router within always receive range, plus two end-nodes

For this case, the figure above is composed of:

- Coordinator: Blue node
- Router: Green node
- End Devices: Yellow and Red nodes

In this simple network, the green Router forms a route directly to the Coordinator, maintained regardless of Wi-Fi ON/OFF duty cycle. With Wi-Fi OFF, the yellow node forms a route directly to the blue Coordinator at a lower route cost than a route via the green Router. The red node cannot be received by the Coordinator and its messages are also routed through the Router to the Coordinator.

With Wi-Fi OFF, the green Router, the yellow node, and the red node (via the green Router) have no issues sending messages to the Coordinator. Regardless of Wi-Fi ON/OFF duty cycle, the green Router and the red node (via the green Router) remain successful sending messages to the Coordinator. With Wi-Fi ON/OFF at low-duty cycle, some messages from the yellow node are periodically blocked, but Zigbee retry mechanisms are effective in getting the messages to the Coordinator.

With Wi-Fi ON/OFF at high-duty cycle, many messages from the yellow node are blocked and Zigbee retry mechanisms may be exhausted. If Wi-Fi ON/OFF stays at high-duty cycle for enough time, the network responds by restructuring the yellow node to route messages to the Coordinator via the Router. However, this route rediscover takes time and messages may be lost. If Wi-Fi ON/OFF remains high-duty cycle, the yellow node messages will continue to go through the Router, which forwards messages to the Coordinator.

However, when Wi-Fi ON/OFF returns to low-duty cycle, the network will, due to lower route cost, return to the original structure with the yellow node sending messages directly to the Coordinator.

Under conditions with Wi-Fi ON/OFF switching between low and high duty cycles, the network may switch back and forth between these two route states. During these switching events, messages from the yellow end-node to the Coordinator are lost.

3 Unmanaged Coexistence

The unmanaged coexistence recommendations that follow provide guidance on how to maximize the EFR32MGx message success with strong nearby Wi-Fi.

3.1 Implement Frequency Separation

From the observations in the previous section, co-channel operation of 802.15.4 with 100% duty cycle Wi-Fi blocks most of the 802.15.4 messages and must be avoided. Also, EFR32MGx tolerates up to 20dB stronger Wi-Fi signal in “far-away” channel case than in adjacent channel case. The 802.15.4 network performance is improved by maximizing the frequency separation between the Wi-Fi network and the 802.15.4 network.

If the Wi-Fi and 802.15.4 radios are implemented with a common host (MCU controlling both radios), then the host should attempt to maximize the frequency separation. For Wi-Fi networks, the Access Point (AP) establishes the initial channel and, in auto channel configuration, is free to move the network to another channel using the Channel Switch Announcement (CSA), introduced in 802.11h, to schedule the channel change.

For Silicon Labs Thread networks, frequency separation implementation depends on the application layer. For Zigbee networks, the Coordinator establishes the initial channel. However, when implemented by the product designer, a Network Manager function, which can be on the Coordinator or on a Router, can solicit energy scans from the mesh network nodes and initiate a network channel change as necessary to a quieter channel.

Note: The Network Manager function is not a mandatory feature, but rather must be implemented using tools/functions provided by the EmberZNet PRO stack.

Details on mesh network channel frequency agility can be found in:

- https://www.silabs.com/community/wireless/zigbee-and-thread/knowledge-base.entry.html/2012/06/29/how_does_frequencya-x5lU
- https://www.silabs.com/community/wireless/zigbee-and-thread/knowledge-base.entry.html/2012/06/29/how_can_i_test_frequ-vrgk

3.2 Operate Wi-Fi with 20MHz Bandwidth

Since Wi-Fi 802.11n uses OFDM (Orthogonal Frequency-Division Multiplexing) sub-carriers, third-order distortion products from these sub-carriers extend one bandwidth on each side of the Wi-Fi channel. 802.11n can operate in 20MHz or 40MHz modes. If operated in 40MHz mode, 40MHz of the 80MHz ISM band is consumed by the Wi-Fi channel. However, an additional 40MHz on each side can be affected by third-order distortion products. These third-order products can block the 802.15.4 receiver and is the primary reason adjacent channel performance is up to 20dB worse than “far-away” channel performance.

In proposing 40MHz mode for 802.11n, the Wi-Fi standard anticipated potential issues with other 2.4GHz ISM devices when Wi-Fi operated in 40MHz mode. During association, any Wi-Fi station can set the **FortyMHz Intolerant** bit in the HT Capabilities Information. This bit informs the Wi-Fi access point that other 2.4GHz ISM devices are present, forcing the entire Wi-Fi network to 20MHz mode.

If the Wi-Fi and 802.15.4 radios are implemented with a common host, then the host should have the Wi-Fi radio set the **FortyMHz Intolerant** bit during association to force the Wi-Fi to 20MHz mode, improving the 802.15.4 performance.

If the application requires Wi-Fi to operate in 40MHz mode, frequency separation must be maximized by placing Wi-Fi channels and 802.15.4 channel at opposite ends of the 2.4GHz ISM band.

From Silicon Labs’ managed coexistence testing, 802.15.4 performance with 40MHz Wi-Fi, *for the same Wi-Fi RF duty cycle*, is comparable to 802.15.4 performance with 20MHz Wi-Fi. While 802.15.4 performance with 100% Wi-Fi RF duty cycle is inherently impaired, 40MHz Wi-Fi, *for the same target Wi-Fi data rate*, has a lower RF duty cycle than 20MHz Wi-Fi, providing the 802.15.4 radio more frequent and longer time gaps for successful transmits and receives.

3.3 Increase Antenna Isolation

From the observations in Section 2 Wi-Fi Impact on Zigbee/Silicon Labs Thread, minimizing the Wi-Fi energy seen by the 802.15.4 RF input improves the 802.15.4 receive range. For example, in the “far-away” channel case (Wi-Fi channel 1 and Zigbee channel 25) with 100% Wi-Fi duty cycle, a -85dBm 802.15.4 message can be received when the average Wi-Fi energy at EFR32MGx input is -22dBm or less. If the Wi-Fi average transmit power level is +10dBm, 32dB or more antenna isolation between the Wi-Fi transmitter and 802.15.4 RF input is required to always receive a -85dBm 802.15.4 signal, Wi-Fi ON or OFF.

Increased antenna isolation can be achieved by:

- Increasing the distance between antennas. In open-space, far-field, power received is proportional to $1/R^2$, where R is the distance between antennas.
- Taking advantage of antenna directionality. A monopole antenna provides a null along the axis of the antenna, which can be directed toward the Wi-Fi antenna(s).

3.4 Use Zigbee/Silicon Labs Thread Retry Mechanisms

The 802.15.4 specification requires retries at the MAC (Media Access Control) layer, which are implemented in Silicon Labs' EmberZNet PRO stack. To further improve message delivery robustness, Silicon Labs EmberZNet PRO stacks also implements NWK (Zigbee network layer) retries, wrapping the MAC retries. The user application can also take advantage of APS (Zigbee Application Support (APS) Sub-Layer) retries, wrapping the NWK retries. More information on the retry mechanisms can be found at:

- https://www.silabs.com/documents/public/user-guides/ug103-02-appdevfundamentals-zigbee.pdf (Section 4.4)
- https://www.silabs.com/community/wireless/zigbee-and-thread/knowledge-base.entry.html/2012/06/29/how_does_the_emberzn-po1M

These retry mechanisms are effective at improving message delivery. However, under high interference conditions, message latency increases.

For Silicon Labs Thread networks, 802.15.4 retries at the MAC layer still apply. However, other message retry mechanisms depend on the application layer.

3.5 Remove FEM (or Operate FEM LNA in Bypass)

EFR32MGx can deliver nearly +20dBm transmit power and has excellent receiver sensitivity without an external FEM (Front End Module). The additional FEM LNA receive gain also improves sensitivity. However, this additional gain also degrades the EFR32xGx linearity performance in the presence of strong Wi-Fi.

For best receive sensitivity in the presence of strong Wi-Fi blockers, either eliminate the FEM or operate the FEM LNA in bypass mode. This recommendation is a trade-off as receive sensitivity without Wi-Fi blockers is improved with FEM LNA gain enabled.

4 Managed Coexistence

The market trends of higher Wi-Fi transmit power, higher Wi-Fi throughput, and integration of Wi-Fi and 802.15.4 radios into the same device have the following impacts.

- Advantages:
  - Host can implement frequency separation between Wi-Fi and 802.15.4.
  - Co-located Wi-Fi radio can force the Wi-Fi network to operate with 20MHz bandwidth.
  - Co-located Wi-Fi and 802.15.4 radios can communicate pending and/or in-progress activity on 2.4GHz ISM transmits and receives.
- Disadvantages:
  - Higher Wi-Fi transmit power requires greater antenna isolation.
  - Higher Wi-Fi throughput results in higher Wi-Fi duty cycle.
  - Antenna isolation is usually limited by the size of the product (only 15-20dB isolation is not unusual).

Assuming frequency separation achieves the “far-away” channel case and Wi-Fi only uses 20MHz bandwidth, a +30dBm Wi-Fi transmit power level at 100% duty cycle requires 45dB antenna isolation to receive -80dBm 802.15.4 messages. This is generally not achievable in small devices with co-located Wi-Fi and 802.15.4.

Managed Coexistence takes advantage of communication between the co-located Wi-Fi and 802.15.4 radios to coordinate each radio’s access to the 2.4GHz ISM band for transmit and receive. For the EFR32, Silicon Labs has implemented a coordination scheme compatible with Wi-Fi devices supporting PTA. This PTA-based coordination allows the EFR32 to signal the Wi-Fi device when receiving a message or wanting to transmit a message. When the Wi-Fi device is made aware of the EFR32 requiring the 2.4GHz ISM band, any Wi-Fi transmit can be delayed, improving Zigbee/Silicon Labs Thread message reliability.

Section 4.1 discusses PTA support hardware options, Section 4.1.17 discusses PTA support software setup, and Section 4.3 demonstrates coexistence configuration setup examples for different Wi-Fi/PTA applications.

4.1 PTA Support Options

PTA is described in IEEE 802.15.2 (2003) Clause 6 and is a recommendation, not a standard. 802.15.2 originally addressed coexistence between 802.11b (Wi-Fi) and 802.15.1 (Bluetooth Classic) and does not describe an exact hardware configuration. However, 802.15.2 recommends that the PTA implementation consider the following:

- TX REQUEST from 802.11b to PTA and TX REQUEST from 802.15.1 to PTA
- TX CONFIRM from PTA to 802.11b and TX CONFIRM from PTA to 802.15.1
- STATUS information from both radios:
  - Radio state [TX, RX, or idle]
  - Current and future TX/RX frequencies
  - Future expectation of a TX/RX start and duration
  - Packet type
  - Priority (Fixed, Randomized, or Quality of Service (QoS) based)

The radio state, transmit/receive, and frequencies described in 802.15.2 are summarized in the following table.

Table 1. IEEE 802.15.2 2.4GHz ISM Co-Located Radio Interference Possibilities

Co-located 802.11b State	Co-Located 802.15.1 State			
	Transmit		Receive	
	In-Band	Out-of-Band	In-Band	Out-of-Band
Transmit	Conflicting Transmits Possible packet errors	No Conflict	Conflicting Transmit-Receive Local packet received with errors	Conflicting Transmit-Receive Local packet received with errors or no errors if sufficient isolation for frequency separation
Receive	Conflicting Transmit-Receive Local packet received with errors	Conflicting Transmit-Receive Local packet received with errors or no errors if sufficient isolation for frequency separation	Conflicting Receives Possible packet errors	No Conflict

From the above table, the frequency separation recommendations from Section 3 Unmanaged Coexistence remain required for managed coexistence:

- 802.15.2 “In-Band” is equivalent to Co-Channel operation, which showed significant Wi-Fi impact on co-channel 802.15.4.
- 802.15.2 “Out-of-Band” covers both Adjacent and “Far-Away” Channel operation, which showed ~20dB improvement in “Far-Away” Channel vs. Adjacent Channel.

As such, for Managed Coexistence, Silicon Labs recommends continuing to implement all the Unmanaged Coexistence recommendations.

- Frequency Separation
- Operate Wi-Fi in 20MHz Bandwidth
- Antenna Isolation
- Zigbee/Silicon Labs Thread Retry Mechanisms
- FEM LNA in Bypass

In reviewing existing PTA implementations, Silicon Labs finds the PTA master implementation has been integrated into many Wi-Fi devices, but not all Wi-Fi devices support a PTA interface. The following figure shows the most common Wi-Fi/PTA implementations supporting Bluetooth.

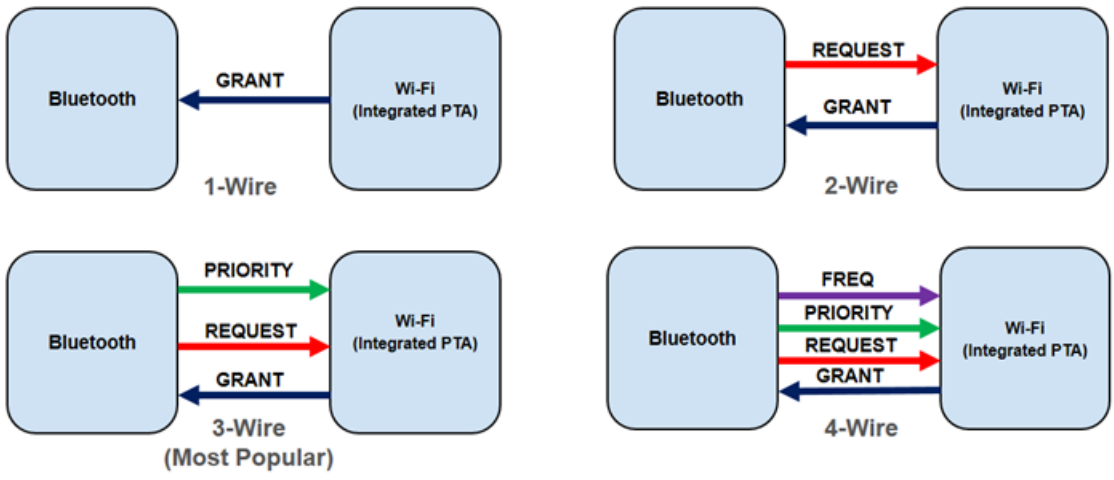


Figure 4. Typical Wi-Fi/Bluetooth PTA Implementations

4.1.1 1-Wire PTA

In 1-Wire PTA, the Wi-Fi/PTA device asserts a GRANT signal when Wi-Fi is not busy transmitting or receiving. When GRANT is asserted, the Bluetooth radio can transmit or receive. This mode does not allow external radio to request the 2.4GHz ISM and is not recommended.

4.1.2 2-Wire PTA

In 2-Wire PTA, the REQUEST signal is added, allowing the Bluetooth radio to request the 2.4GHz ISM band. The Wi-Fi/PTA device internally controls the prioritization between Bluetooth and Wi-Fi and, on a conflict, the PTA can choose to either GRANT Bluetooth or Wi-Fi.

4.1.3 3-Wire PTA

In 3-Wire PTA, the PRIORITY signal is added, allowing the Bluetooth radio to signify a high- or low-priority message is either being received or transmitted. The Wi-Fi/PTA device compares this external priority request against the internal Wi-Fi priority, which may be high/low or high/mid/low and can choose to either GRANT Bluetooth or Wi-Fi. PRIORITY can be implemented as either static or directional (enhanced) priority:

- Static: PRIORITY is either high or low during REQUEST asserted for the transmit or receive operation.
- Directional: PRIORITY is either high or low for a typically 20µs duration after REQUEST asserted, but switches to low during receive operation and high during transmit operation.
- For platforms, such as Wi-Fi data routers that can achieve high Wi-Fi duty-cycles, as well as IoT hubs that stream Bluetooth audio, implementing PRIORITY is highly recommended as it provides the Wi-Fi/PTA device with insight on the EFR32 REQUEST. PRIORITY is also configurable, both at compile-time and at run-time, to address various product optimization requirements. However, given the relatively low RF duty cycle of 802.15.4, static PRIORITY may not be necessary for platforms that do not experience high Wi-Fi duty cycles nor support Bluetooth audio streaming, freeing a GPIO pin on the EFR32 and SoC.

4.1.4 4-Wire PTA

In 4-Wire PTA, the FREQ signal is added, allowing the Bluetooth radio to signify an “in-band” or “out-of-band” message is either being received or transmitted. Silicon Labs recommends maximizing frequency separation, making the FREQ signal mute. For this reason, Silicon Labs’ EFR32 does not support the FREQ signal. Silicon Labs therefore recommends asserting the FREQ input to the Wi-Fi/PTA.

4.1.5 Single EFR32 Connected to Wi-Fi/PTA Transmit Timing

Additional details about the implementation of managed coexistence and test results are available in an expanded version of this application note, *AN1017-NDA: ZigBee and Thread Coexistence with Wi-Fi*, available under non-disclosure from Silicon Labs technical support.

4.1.6 Single EFR32 Connected to Wi-Fi/PTA Receive Timing

Additional details about the implementation of managed coexistence and test results are available in an expanded version of this application note, *AN1017-NDA: ZigBee and Thread Coexistence with Wi-Fi*, available under non-disclosure from Silicon Labs technical support.

4.1.7 Directional PRIORITY Transmit Timing (Single EFR32)

Additional details about the implementation of managed coexistence and test results are available in an expanded version of this application note, *AN1017-NDA: ZigBee and Thread Coexistence with Wi-Fi*, available under non-disclosure from Silicon Labs technical support.

4.1.8 Directional PRIORITY Receive Timing (Single EFR32)

Additional details about the implementation of managed coexistence and test results are available in an expanded version of this application note, *AN1017-NDA: ZigBee and Thread Coexistence with Wi-Fi*, available under non-disclosure from Silicon Labs technical support.

4.1.9 Multiple EFR32s Connected to Wi-Fi/PTA

The 802.15.2 recommendation only addresses a single 802.11b radio connected to a single 802.15.1 radio. However, market trends are requiring multiple co-located 2.4GHz ISM radios (Zigbee, Silicon Labs Thread, and Bluetooth low energy) to operate with a Wi-Fi/PTA device only designed for one external radio. Silicon Labs has addressed this requirement by enhancing the REQUEST signal with the "shared" REQUEST feature, implemented through the AppBuilder module in Simplicity Studio. The Silicon Labs EFR32 PRIORITY can also be configured as "shared" to support additional multi-EFR32 radio configurations.

The "shared" REQUEST and "shared" PRIORITY features have the following characteristics:

- Operate REQUEST and PRIORITY in open-drain or open-source (an external 1 kΩ ±5% pull-up for open-drain or pull-down for open source is required).
- Before asserting REQUEST, test REQUEST to determine if another EFR32 has already asserted REQUEST.
- If not already asserted, assert REQUEST.
- If already asserted:
  - Wait for REQUEST to deassert.
  - Delay random time (programmable).
  - Re-test REQUEST.
  - If not already asserted, assert REQUEST.
  - If already asserted:
    - Another radio secured REQUEST.
    - Return to wait for REQUEST to deassert.

With these enhanced REQUEST and PRIORITY features, multiple EFR32s implement a PTA interface, which to the Wi-Fi/PTA device appears to be a single external radio.

An example of three EFR32 radios (Zigbee, Silicon Labs Thread, and BLE) using 2-Wire PTA interfaced to one Wi-Fi/PTA interface using 2-Wire PTA is shown in the following figure.

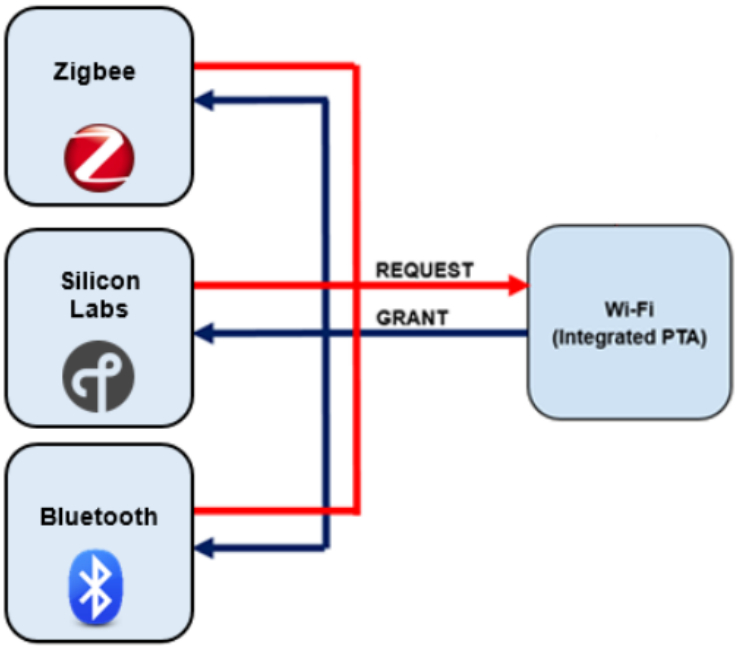


Figure 5. Three EFR32s Supporting Single Wi-Fi 2-Wire PTA Interface
(REQUEST pull-up or pull-down not shown)

With these enhanced REQUEST features and "shared" PRIORITY, an example of three EFR32 radios (Zigbee, Silicon Labs Thread, and BLE) using 3-Wire PTA interfaced to one Wi-Fi/PTA interface using 3-Wire PTA is shown in the following figure.

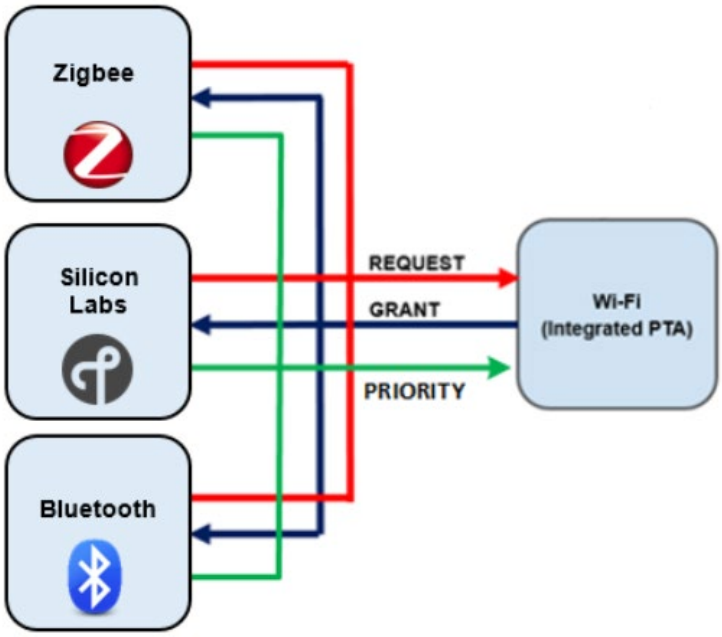


Figure 6. Three EFR32s Supporting Single Wi-Fi/PTA 3-Wire PTA Interface
(REQUEST and PRIORITY pull-ups or pull-downs not shown)

4.1.10 Multiple EFR32s Connected to Wi-Fi/PTA Transmit Timing

Additional details about the implementation of managed coexistence and test results are available in an expanded version of this application note, *AN1017-NDA: ZigBee and Thread Coexistence with Wi-Fi*, available under non-disclosure from Silicon Labs technical support.

4.1.11 Multiple EFR32s Connected to Wi-Fi/PTA Receive Timing

Additional details about the implementation of managed coexistence and test results are available in an expanded version of this application note, *AN1017-NDA: ZigBee and Thread Coexistence with Wi-Fi*, available under non-disclosure from Silicon Labs technical support.

4.1.12 Directional PRIORITY Transmit Timing (Multiple EFR32s)

Additional details about the implementation of managed coexistence and test results are available in an expanded version of this application note, *AN1017-NDA: ZigBee and Thread Coexistence with Wi-Fi*, available under non-disclosure from Silicon Labs technical support.

4.1.13 Directional PRIORITY Receive Timing (Multiple EFR32s)

Additional details about the implementation of managed coexistence and test results are available in an expanded version of this application note, *AN1017-NDA: ZigBee and Thread Coexistence with Wi-Fi*, available under non-disclosure from Silicon Labs technical support.

4.1.14 Wi-Fi/PTA Considerations

From Silicon Labs' testing, Wi-Fi/PTA implementations vary. Best 802.15.4 results are attained when the Wi-Fi/PTA implementation has the following characteristics.

4.1.14.1 Wi-Fi/PTA Supports Wi-Fi TX Pre-emption

Wi-Fi RF duty cycle varies based on MCS (Modulation and Coding Scheme) index, guard interval, bandwidth, and target data rate. Wi-Fi signal strength and SNR vary with distance and objects/walls between AP and STA (Station), particularly with mobile Wi-Fi devices. When Wi-Fi signal strength and SNR drop, the MCS index is lowered to modulations able to operate at the lower SNR. However, lowering the MSC index also lowers the maximum data rate and the Wi-Fi RF duty cycle increases as the AP and STA attempt to maintain the target data rate.

Wi-Fi TX pre-emption impacts Wi-Fi performance as in-progress Wi-Fi TX packets are aborted and corrupted. In response to a corrupted packet, most Wi-Fi devices assume SNR or interferer and temporarily lower the MCS index to maintain connection but recover to optimum MSC index as subsequent messages are successful. However, 802.15.4 networks are low duty cycle and the infrequent Wi-Fi TX pre-emption events have a low impact on Wi-Fi performance.

After REQUEST asserted, Wi-Fi/PTA devices not supporting Wi-Fi TX pre-emption delay GRANT to EFR32 until the end of the in-progress packet. At MCS0 with 20MHz bandwidth and large aggregated Wi-Fi packets, this delay can be as long as 16ms. This delay is much longer than the typical ~200µs REQUEST to end of CCA, and much longer than the maximum received 802.15.4 packet. As such, lack of Wi-Fi TX pre-emption during high Wi-Fi RF duty cycle increases 802.15.4 retries, increases end-node battery consumption, and increases 802.15.4 message loss.

After REQUEST asserted, Wi-Fi/PTA devices supporting TX pre-emption stop the Wi-Fi transmit, ramp down the Wi-Fi PA (within-gate delays up to tens of µs), and quickly GRANT the 2.4GHz band to the 802.15.4 radio. The short time from REQUEST to quiet Wi-Fi and GRANT improves the 802.15.4 TX and RX success on first attempt, which decreases 802.15.4 retries, decreases end-node battery consumption, and decreases 802.15.4 message loss.

4.1.14.2 Wi-Fi/PTA Prevents Wi-Fi Transmit when GRANT Asserted

Some Wi-Fi/PTA devices have been optimized to work with classic Bluetooth devices and have also been optimized to continue Wi-Fi TX events for a short time after asserting GRANT. This works only for that particular Wi-Fi/PTA and classic Bluetooth combination, as the minimum time from REQUEST asserted to Bluetooth TX start was used to determine how long Wi-Fi TX can continue after GRANT asserted. For 802.15.4 receives, the Wi-Fi/PTA device must stop all Wi-Fi TX when GRANT asserted.

Some Wi-Fi/PTA devices also attempt Wi-Fi TX based on the FREQ feature described in the 4-Wire PTA. If the FREQ feature is set by GPIO pin (or Wi-Fi/PTA internal register) to different frequencies, Wi-Fi/PTA will assert GRANT, but continue to Wi-Fi transmit. Given co-located Wi-Fi and 802.15.4 radios with insufficient antenna isolation, the FREQ feature must be set to always blocking, FREQ GPIO pin (or Wi-Fi/PTA internal register) held asserted.

4.1.14.3 Wi-Fi/PTA and Application Implements Reasonable Prioritization

802.15.4 RX packets are asynchronous and missing a packet ensures retries, increased message latency, and decreased end-node battery life. As such, 802.15.4 RX priority should be set as high as possible for the application.

802.15.4 TX packets, originating from the co-located Wi-Fi/802.15.4 may have more discretion, as these devices are typically wall-powered. However, 802.15.4 transmits showing high latency can result in an unsatisfactory user experience. As such, 802.15.4 TX should also be set as high as possible for the application.

4.1.14.4 Wi-Fi/PTA Implements Aggregation

Ensure aggregation is enabled on Wi-Fi/PTA device. During suitable conditions, aggregation combines multiple smaller packets into fewer larger packets, reducing total packet overhead. As such, aggregation aids Zigbee and Silicon Labs Thread in-coming packet detection as Wi-Fi TX idle periods combine into larger idle periods for improved 802.15.4 PREAMBLE/SYNCH detection.

4.1.14.5 Wi-Fi/PTA Supports Directional PRIORITY

Directional PRIORITY is a common feature that can be enabled on Wi-Fi/PTA devices and is recommended due to the additional radio state information provided by the Bluetooth radio which allows the possibility for concurrent Wi-Fi and Bluetooth reception when utilizing channel separation as well as concurrent Wi-Fi and Bluetooth transmission, also when utilizing channel separation. In this way, implementing Pulsed Directional PRIORITY maximizes Wi-Fi and Bluetooth throughput.

4.1.15 TX PRIORITY Escalation

To improve Wi-Fi performance with Zigbee/Silicon Labs Thread coexistence, it is possible to start all Zigbee/Silicon Labs Thread TX messages at low priority. However, to avoid blocking, all Zigbee/Silicon Labs Thread TX messages during busy Wi-Fi, TX PRIORITY Escalation will escalate TX to high priority after a programmable number of CCA/GRANT or MAC failures. Then, after a successful TX message, de-escalate TX back to low priority.

As described in Section 4.1.9 Multiple EFR32s Connected to Wi-Fi/PTA, TX PRIORITY Escalation can be controlled at run-time via "CCA/GRANT TX PRIORITY Escalation Threshold" and "MAC Fail TX PRIORITY Escalation Threshold" fields of pta Options. When using this feature, "TX high PRIORITY" field must be set to 0 to avoid driving PRIORITY high on all TX messages.

4.1.16 PWM for High Duty Cycle Wi-Fi

PWM is a feature where the EFR32 reserves a timed slot for REQUEST. This is important in high duty-cycle Wi-Fi because planned EFR32 TX and RX, allows Wi-Fi to be more amenable to GRANT airtime when it is expected. Note that GRANT is not guaranteed; Wi-Fi still makes the decision based on its discretion.

Zigbee and Silicon Labs Thread have no knowledge of when an incoming packet will arrive and must capture the 802.15.4 PREAMBLE/SYNCH (160µs duration) to detect an incoming packet. As the co-located Wi-Fi TX RF duty cycle increases, the Wi-Fi TX idle periods become smaller and there are fewer idle periods exceeding 160µs. As such, the probability of a 160µs PREAMBLE/SYNCH arriving during an acceptably large Wi-Fi TX idle period decreases significantly.

The PWM feature implements regular PTA REQUESTs to interrupt the high Wi-Fi RF duty cycle and ensure adequate windows for PREAMBLE/SYNCH detection. However, this feature does degrade Wi-Fi performance. Additionally, the exact PWM period and duty cycle must be carefully selected to not alias PWM period with Wi-Fi beacons to avoid collapsing the Wi-Fi network.

Note: Ensure aggregation is enabled on Wi-Fi/PTA device. During suitable conditions, aggregation combines multiple smaller packets into fewer larger packets which reduces the total packet overhead. As such, aggregation aids Zigbee and Silicon Labs Thread packet detection as Wi-Fi TX idle periods combine into larger idle periods for improved PREAMBLE/SYNCH detection.

4.1.16.1 Background

The following is a 15.485ms capture of a Wi-Fi TX Active signal while Wi-Fi is transmitting iperf TCP at maximum throughput. Within this waveform, there are multiple Wi-Fi TX idle periods of which the first group of idle periods is expanded.

Figure 7. High-Duty Cycle Wi-Fi TX Active

From the first group, only the 0.306ms idle period is sufficient for PREAMBLE/SYNCH detection with a 0.146ms detection window (0.306ms–0.160ms). The following table lists all Wi-Fi TX idle period durations in the above 15.485ms capture.

Table 2. High-Duty Cycle Wi-Fi TX Active Durations

Wi-Fi TX Idle (ms)	Detection Window (Idle–0.160ms) (ms)
0.306	0.146
0.113	0.000
0.061	0.000
0.246	0.086
0.077	0.000

Wi-Fi TX Idle (ms)	Detection Window (Idle–0.160ms) (ms)
0.260	0.100
0.061	0.000
0.065	0.000
0.016	0.000
0.065	0.000
0.016	0.000
0.065	0.000
0.016	0.000
0.245	0.085
0.105	0.000
0.098	0.000
0.016	0.000
0.171	0.011

The sum of all Wi-Fi TX Idle periods is 2.002ms. This Wi-Fi TX Activity indicates 87.1% duty cycle [(15.485–2.002) / 15.485 x 100%], but this calculation does not include the additional Wi-Fi RX activity.

The sum of all Detection Windows is 0.428ms, providing only 2.8% probability of PREAMBLE/SYNCH detection [0.428 / 15.485 x 100%], but, again, this calculation does not include Wi-Fi RX Activity. With only 2.8% probability of detection and a target 1% or less message loss, 165 retries, per 802.15.4 message, are required [CEILING(LOG(1%)/LOG(100%–2.8%)]. This large number of required retries is far beyond the MAC, NWK, and APS retries and, even if retries are extended, the message latency becomes impractical. Large retries also quickly degrade life-time for battery power end-nodes.

To resolve, 802.15.4 radios need more RF air-time to listen for incoming packets without being blocked by co-located Wi-Fi TX. Two options are possible:

- Co-located Wi-Fi device enforces a maximum allowed Wi-Fi TX RF duty cycle.
- Co-located 802.15.4 radio asserts REQUEST at high PRIORITY to force regular quiet Wi-Fi TX.

Silicon Labs has not found option #1 to be reliable over various Wi-Fi/PTA devices. Silicon Labs has developed a PWM extension feature to support option #2 and this PWM feature can be applied to any use case where EFR32 incoming Zigbee and Silicon Labs Thread packet detection is impaired by a co-located, controllable, external device. This includes, but is not limited to, the following use cases:

- High duty-cycle co-located Wi-Fi TX
- Zigbee and Silicon Labs Thread co-channel with Wi-Fi
- External T/R switch requiring Zigbee and Silicon Labs Thread to share a single antenna with Wi-Fi

4.1.16.2 PWM Feature Description

The PWM feature does not require any additional GPIOs for single EFR32 designs. However, multi -EFR32 designs require an additional GPIO to be available for each EFR32.

Under the same Wi-Fi TX Active throughput conditions as in Section 4.1.16.1 Background, the following figure shows the REQUEST||PWM (shown as REQUEST||TIMER1) signal interrupting the Wi-Fi TX with a 19.5ms period and 20% duty cycle, sufficient to allow 802.15.4 RX with < 1% message loss.

Figure 8. EFR32 RX Success with PWM during High-Duty Cycle Wi-Fi TX

The green arrows show cases where the PREAMBLE/SYNCH was detected during a high PWM cycle, driving REQUEST||PWM high. This high PWM cycle stopped Wi-Fi TX Activity to allow detection of incoming 802.15.4 PRERAMBLE/SYNCH. As also seen, the REQUEST||PWM cycle is longer than all other cycles due to assertion of normal PTA REQUEST signal OR'ed with PWM cycle.

The red boxes are cases where the remote 802.15.4 radio executed packet transmits that were not detectable due to Wi-Fi TX Activity and packets' PREAMBLE/SYNCH not aligning with the PWM cycle.

Optimum PWM period and duty cycle can vary based on 802.15.4 operation and Wi-Fi activity. If Wi-Fi activity is low, PWM could be disabled and fall back to normal PTA activity as described earlier. During very high Wi-Fi duty cycle for unicast 802.15.4 traffic, a PWM programmed to 19.5ms period and 20% duty is effective in reducing message loss to less than 1% with APS retries disabled. As noted, Wi-Fi throughput is impacted and, under 19.5ms/20% condition, the high duty cycle Wi-Fi TX throughput noted above dropped ~20% (~270Mbps without PWM to ~220Mbps with PWM).

During very high Wi-Fi duty cycle for broadcast 802.15.4 traffic without ACKs (for example, as occurs during network join sequence), a higher PWM duty cycle is required (e.g., 19.5ms period and 80% duty cycle). This higher duty cycle can be disruptive to Wi-Fi but is only needed during expected broadcast message activity. This higher duty cycle can be reduced as soon as broadcast activity (for example, device join) completes.

4.1.16.3 PWM Implementation

The optimum PWM implementation varies with single-EFR32 vs. multi-EFR32 applications. This implementation difference is primarily due to the REQUEST sharing feature used in multi-EFR32 applications. However, if addressed during circuit board design, this difference is easily resolved. For details of the Application Builder implementation, see Section 4.2.1.4 PWM.

Use of an API or coexistence CLI command are needed to allow the host application to control the PWM period, duty cycle, and priority.

4.1.16.3.1 Multi-EFR32 Radio with PWM

For multi-EFR32 radio applications not implementing PWM, the Shared REQUEST signal is used to arbitrate which EFR32 radio has access to the 2.4GHz ISM band when GRANT is asserted from the Wi-Fi/PTA. If the PWM signal were to be applied to the shared REQUEST pin, other EFR32 radios would interpret the REQUEST as busy when it might just be forcing Wi-Fi TX idle to allow any of the IoT radios to detect an incoming packet. Instead, the Shared REQUEST feature is accommodated by separating the Shared REQUEST signal from the REQUEST||PWM signal resulting in a separate non-Wi-Fi/PTA signal connection between the EFR32 devices for Shared REQUEST.

The PWM enabled Wi-Fi/PTA connections between the Wi-Fi and Zigbee, Silicon Labs Thread, and BLE EFR32 radios are the same as the multi-EFR32 radio with typical 3-Wire PTA schema. Only one radio, selected at run time, controls the PWM and ORs the PWM signal in software with its own REQUEST (arbitrated by the Shared REQUEST signal). The other EFR32 radios must also implement PWM in Hardware Configurator to provide their REQUEST to the PTA interface, (also arbitrated by Shared REQUEST). Each EFR32 radio's REQUEST||PWM GPIO is set to open-drain/open-source, allowing a hardware OR function of any EFR32 radio's REQUEST with REQUEST||PWM.

If the radio driving REQUEST||PWM needs to go off-line (for example, for a firmware update), its PWM outputs can be halted and another EFR32 radio's PWM outputs can then be activated for continued PWM operation. In the following figure, the Zigbee radio implements the PWM feature, the Silicon Labs Thread radio provides PWM backup, and BLE radio implements multi-EFR32 coexistence.

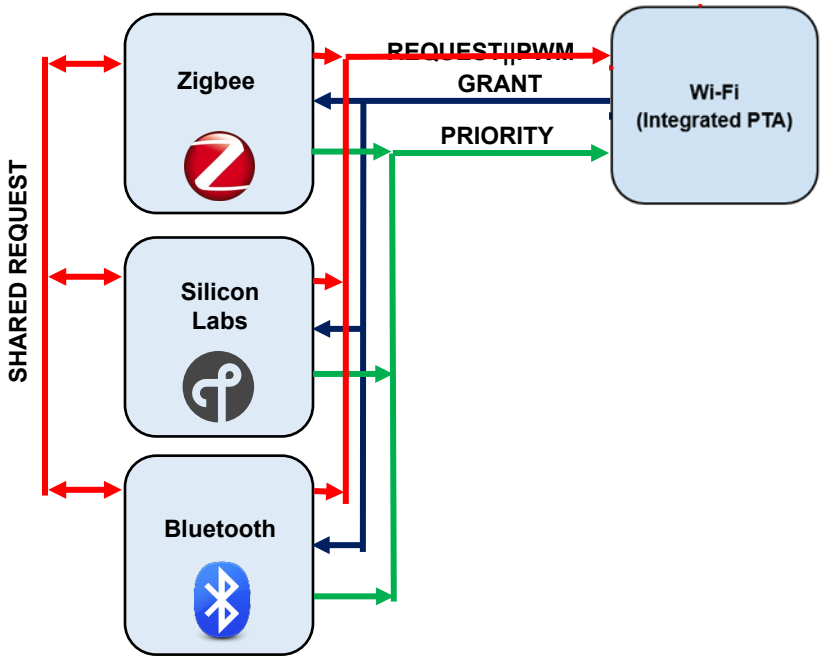


Figure 9. Three EFR32s Supporting PWM and Wi-Fi/PTA with Single 3-Wire PTA Interface
(REQUEST and PRIORITY pull-ups and pull-downs not shown)

4.1.17 Directional PRIORITY

Directional PRIORITY provides radio state information on the same signal line as PRIORITY by a timed pulse indicating a priority transaction, followed by a radio state indicating to the Wi-Fi part that the EFR32 is transmitting or receiving.

PRIORITY signal must be enabled for implementing **Directional PRIORITY**.

Directional PRIORITY requires the use of an additional GPIO, one Timer, and either four PRS channels (EFR32xG2x) or six PRS channels (EFR32xG1x). When Directional PRIORITY is enabled, the designer will need to verify the Timer and PRS channels selected for Directional PRIORITY are not used by the protocol SDK stack, plugins or custom application code.

In addition, RACLNAEN and REQUEST is either one PRS channel which inverts ORs in one operation (EFR32xG2x) or two separate PRS channels (EFR32xG1x).

4.1.17.1 Single-EFR32 PTA with Directional PRIORITY

The following figures show the GPIO, Radio, and Timer signals for Single-EFR32 PTA with or without SDK PWM and with Directional PRIORITY.

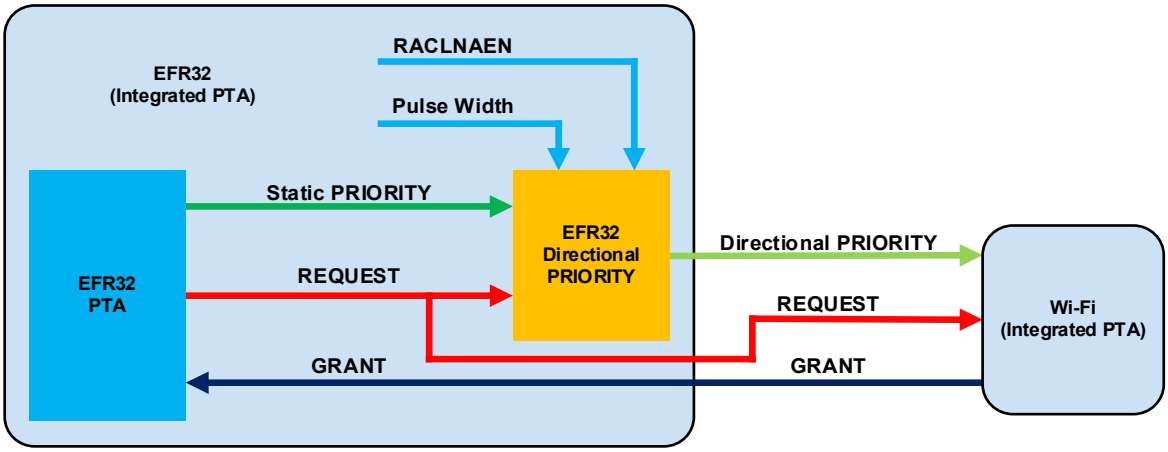


Figure 10. Single-EFR32 PTA without SDK PWM, with Directional PRIORITY

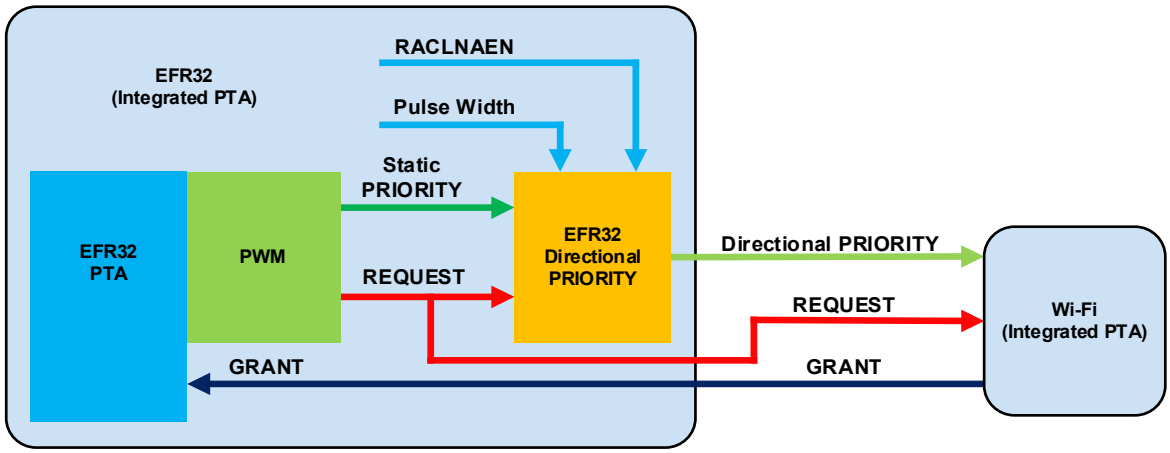


Figure 11. Single-EFR32 PTA with SDK PWM, with Directional PRIORITY

Directional PRIORITY

- EFR32 Directional Priority output.
- GPIO connects to Wi-Fi PTA.
- Requires active high Static PRIORITY and active high REQUEST

Static PRIORITY

- EFR32 PTA output.
- Directional PRIORITY input.
- Assign to any unused GPIO.
- Not connected to any external circuit.

REQUEST

- EFR32 PTA output.
- Directional PRIORITY input.
- Compared in Pulse Width Timer.
- GPIO connects to Wi-Fi PTA.

GRANT

- EFR32 PTA input.
- GPIO connects to Wi-Fi PTA.

RACLNAEN

- EFR32 radio receives LNA enable output.
- Directional PRIORITY input.

Pulse Width

- EFR32 Timer compared with REQUEST GPIO.
- Directional PRIORITY input.

4.1.17.2 Multi-EFR32 PTA with Directional PRIORITY

The following figure shows the GPIO, Radio, and Timer signals for Multi-EFR32 PTA with SDK PWM and Directional PRIORITY.

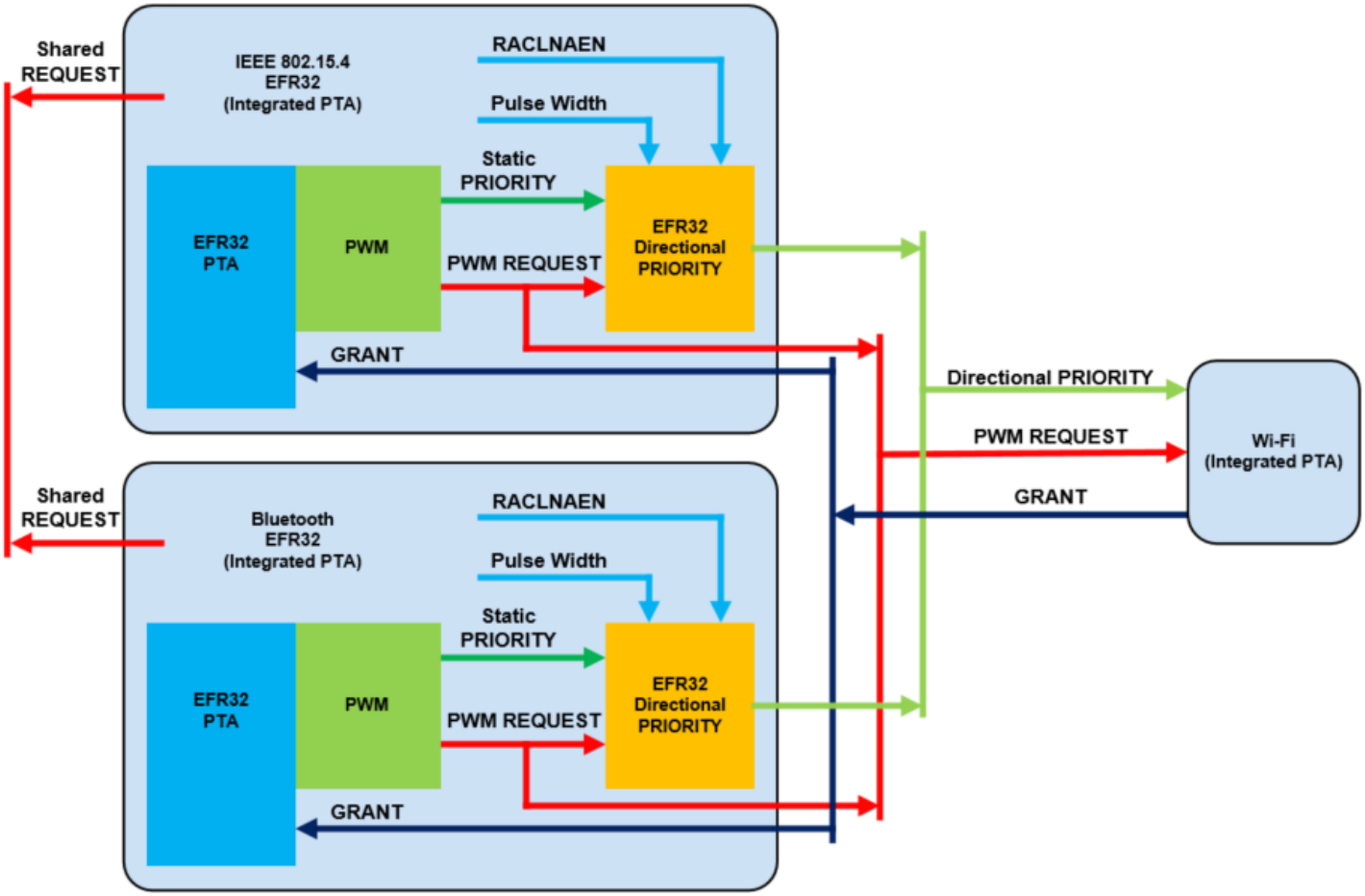


Figure 12. Multi-EFR32 PTA with SDK PWM and with Directional PRIORITY (1.5K pull-ups and pull-downs not shown)

Shared REQUEST

- EFR32 PTA input / output.
- GPIO connects to all EFR32s for PTA bus arbitration between EFR32s
  - Configured as open source / drain.
  - The active high Shared REQUEST is open-source and an external 1 kΩ ±5% pull-down is required.

Directional PRIORITY

- EFR32 Directional Priority output
- GPIO Connects to Wi-Fi PTA and all EFR32s
  - Configured as open source / drain.
    - The active high Directional PRIORITY is open-source and an external 1 kΩ ±5% pull-down is required.

Static PRIORITY

- EFR32 PTA output.
- Directional PRIORITY input.
- Assign to any unused GPIO.
- Not connected to any external circuit.

PWM REQUEST

- EFR32 PTA output.
- Directional PRIORITY input.
- GPIO Connects to Wi-Fi PTA and all EFR32s.
  - Configured as open source / drain.
    - If active high, PWM REQUEST is open-source and an external 1 kΩ ±5% pull-down is required.
  - Compared with Pulse Width in EFR32 Timer.

GRANT

- EFR32 PTA input.
- GPIO connects to Wi-Fi PTA and all EFR32s.

RACLNAEN

- EFR32 Radio Receives LNA enable output.
- Directional PRIORITY input.

Pulse Width

- EFR32 Timer compared with PWM REQUEST GPIO.
- Directional PRIORITY input.

The following figure shows the PRS and Timer logic diagram for Directional PRIORITY for EFR32xG1x.

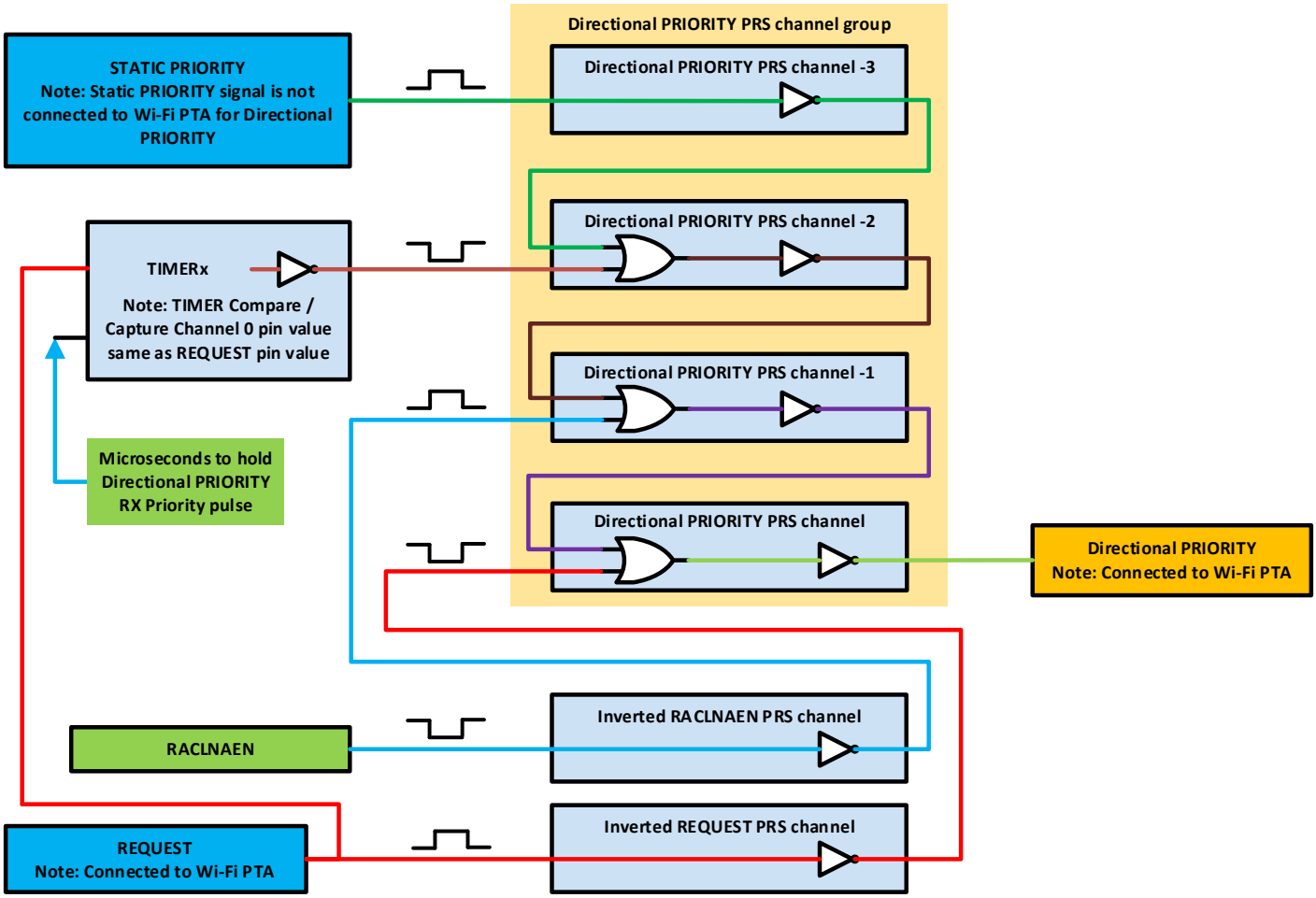


Figure 13. EFR32xG1x Directional PRIORITY Logic Diagram

The following figure shows the PRS and Timer logic diagram for Directional PRIORITY for EFR32xG2x.

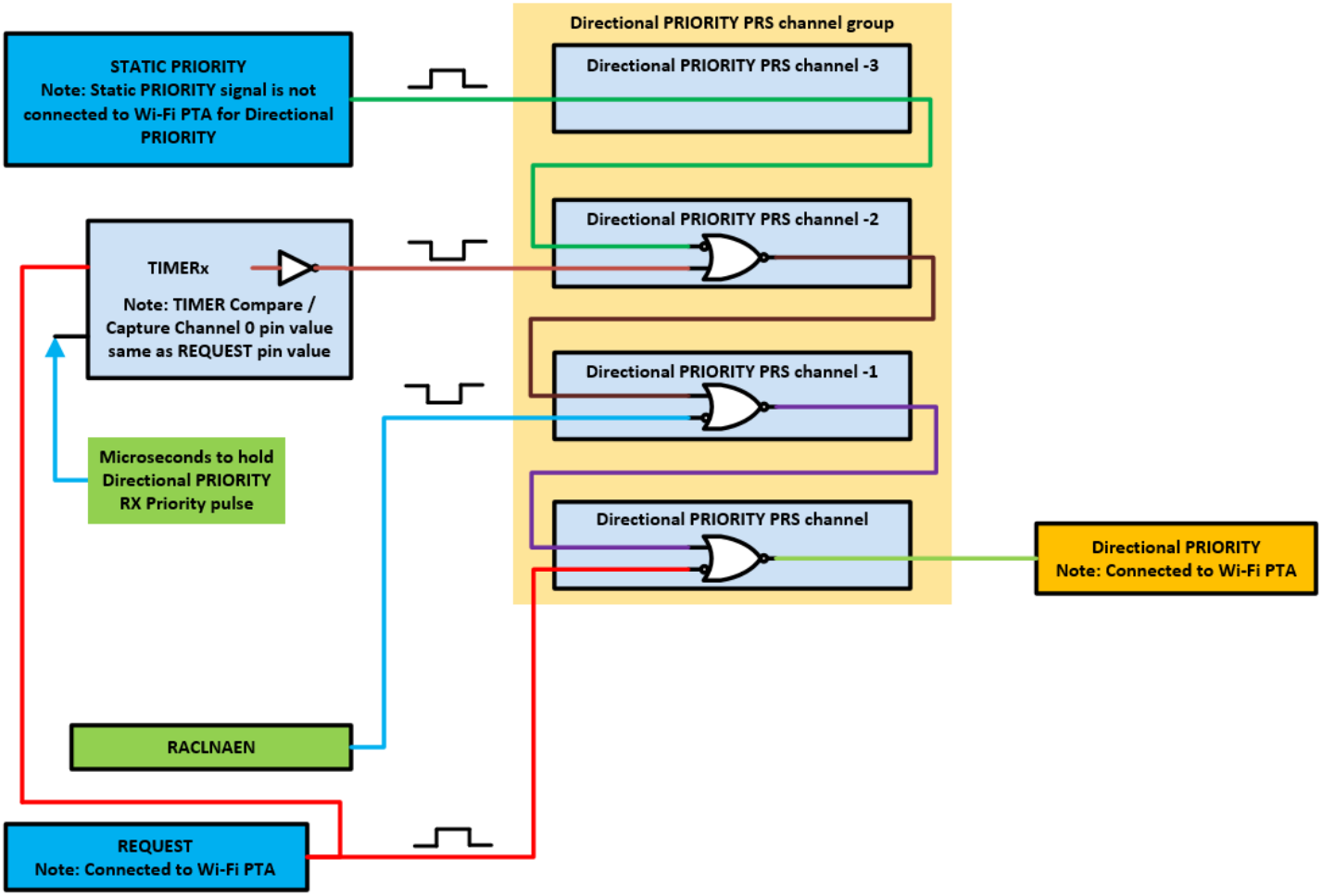
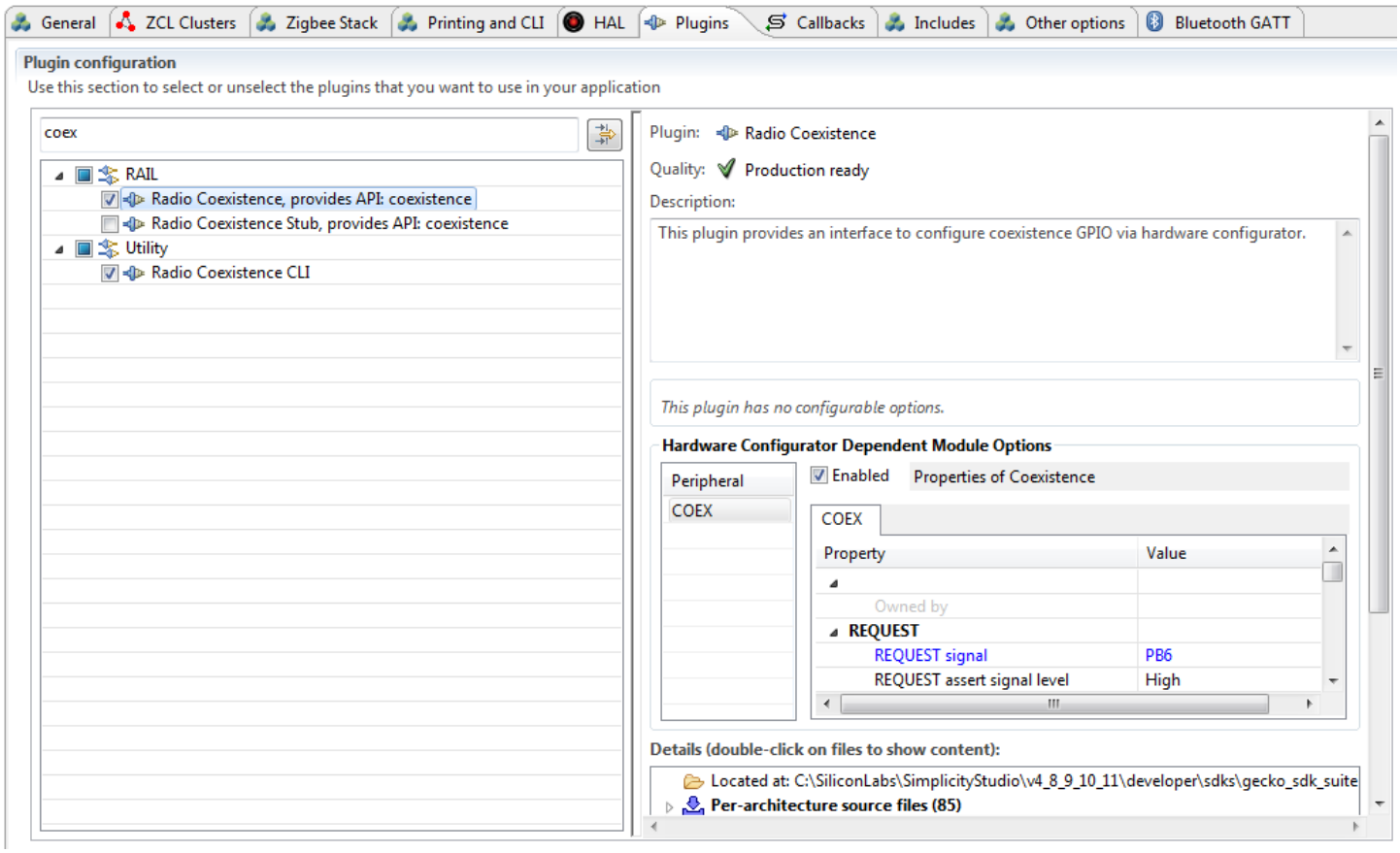


Figure 14. EFR32xG2x Directional PRIORITY Logic Diagram

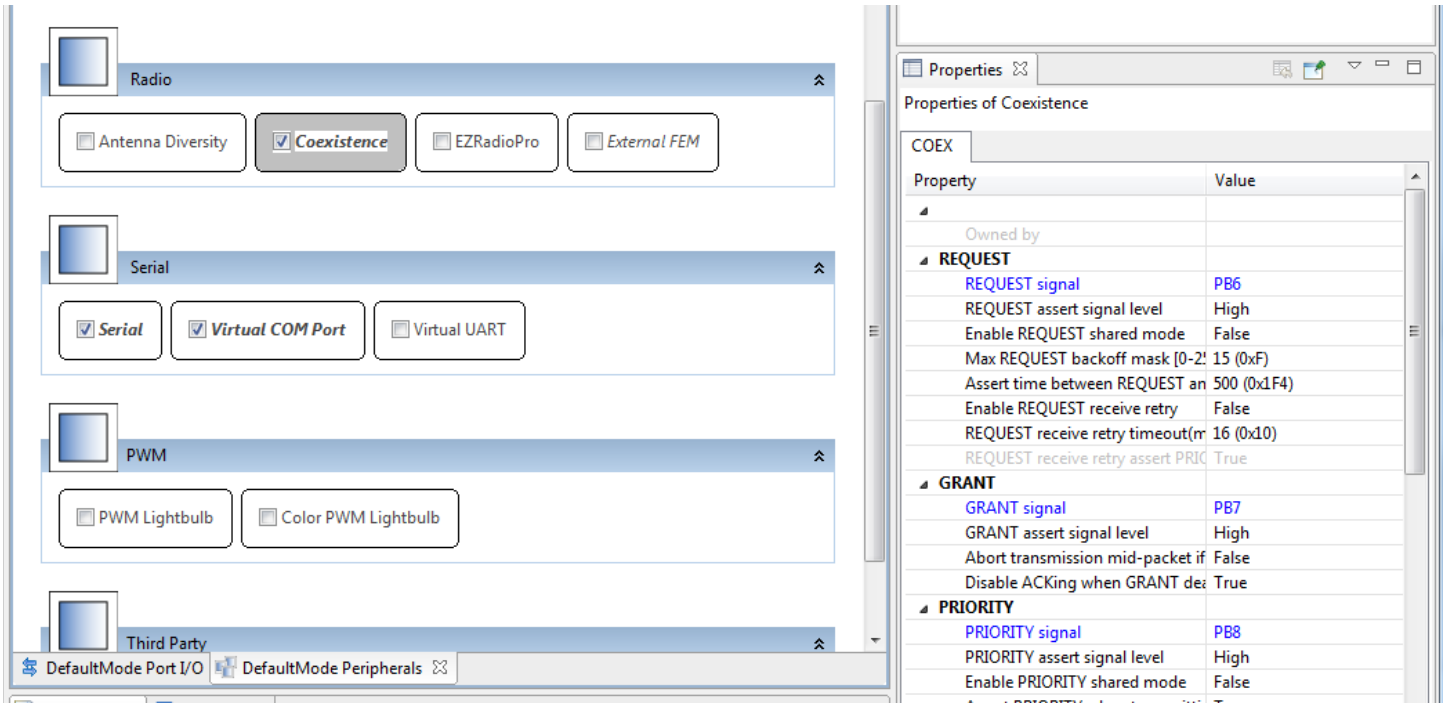
4.2 PTA Support Software Setup

Note: GPIO interrupt numbers are based on the GPIO pin numbers and not the port. This can cause conflicts if the same pin is selected for different ports—for example, PD15 will conflict with PB15. Silicon Labs recommends avoiding these conflicts. If the conflict exists in hardware, manual macros can be added with assistance of Silicon Labs Support.

1. Enable the **Radio Coexistence** plugin under the **RAIL** section of the **Plugins** tab. **Radio Coexistence CLI** is optional but recommended.



2. Open the project's .hwconf file in Hardware Configurator and select **Default Mode Peripherals** view.
3. Select **Coexistence** in the **Radio** section to open coexistence properties and ensure **Coexistence** is enabled.



Properties of Coexistence

COEX

Property	Value
REQUEST	
REQUEST signal	PB6
REQUEST assert signal level	High
Enable REQUEST shared mode	False
Max REQUEST backoff mask [0-255]	15 (0xF)
Assert time between REQUEST and RX/TX start (us) [BLE only]	500 (0x1F4)
Enable REQUEST receive retry	False
REQUEST receive retry timeout(ms)	16 (0x10)
REQUEST receive retry assert PRIORITY	True
GRANT	
GRANT signal	PB7
GRANT assert signal level	High
Abort transmission mid-packet if GRANT is lost	False
Disable ACKing when GRANT deasserted, RHO asserted, or REQUEST not secured (shared REQUEST only)	True
PRIORITY	
PRIORITY signal	PB8
PRIORITY assert signal level	High
Enable PRIORITY shared mode	False
Assert PRIORITY when transmitting packet	True
Assert PRIORITY when receiving packet	True
Include TX PRIORITY Escalation	True
CCA/GRANT TX PRIORITY Escalation Threshold	4 (0x4)
MAC Fail TX PRIORITY Escalation threshold	0 (0x0)
PWM	
PWM REQUEST signal (shared REQUEST only)	Disabled
PWM REQUEST signal level (shared REQUEST only)	High
Enable PWM REQUEST at startup	False
PWM REQUEST Period (0.5ms)	78 (0x4E)
PWM REQUEST Duty-Cycle (%)	20 (0x14)
Assert PRIORITY when PWM REQUEST asserted	Low
Radio Hold Off	
RHO signal	PB10
RHO assert signal level	High
Directional PRIORITY	
Enable Directional PRIORITY	True
Directional PRIORITY Timer	None
Directional PRIORITY pulse width	20 (0x14)
Directional PRIORITY PRS channel	Disabled
PRS channel output pin	
Inverted REQUEST PRS channel	CH10
Inverted RACLNAEN PRS channel	CH1
Radio Blocker Optimization	
Optimize for Wi-Fi Blocker	True
Enable Runtime Control	False
Optional 'Wi-Fi Select' Control of Blocker Optimization	PC0
Assert	High
Timeout (ms)	100 (0x64)
RX Active	
RX active PRS channel	CH5
PRS channel 5 output pin	PD13
RX active assert signal level	High

Figure 15. Coexistence Properties

4.2.1 AppBuilder Configurable Options

The coexistence configuration AppBuilder configurable options include the following.

4.2.1.1 REQUEST

REQUEST signal enabled

- If selected, REQUEST is mapped to GPIO pin and is used by PTA implementation.
- If not selected, REQUEST is not mapped to GPIO pin.

REQUEST signal is shared

- If selected. REQUEST is shared and implements open-drain or open-source I/O for multi-EFR32 radio applications.
- If active low, REQUEST is open-drain and an external 1 kΩ ±5% pull-up is required.
- If active high, REQUEST is open-source and an external 1 kΩ ±5% pull-down is required.
- If not selected, REQUEST is not shared and implements a push-pull output for single EFR32 radio applications.

REQUEST signal active high

- If selected, REQUEST GPIO pin is driven high (> Voh) when REQUEST is asserted.
- If not selected, REQUEST GPIO pin is driven low (< Vol) when REQUEST is asserted.

REQUEST signal GPIO port and REQUEST signal GPIO pin

- Select REQUEST port and pin matching circuit board configuration.
- To minimize PTA impact to other EFR32 peripherals, recommended REQUEST port and pin are:

EFR32 Package	REQUEST Port/Pin
QFN48	PC10
QFN32	PC11

REQUEST signal max backoff mask[0-255]

- REQUEST signal max backoff determines the random REQUEST delay mask (only valid if REQUEST signal is shared).
- Random delay (in µs) is computed by masking the internal random variable against the entered mask.
- The mask should be set to a value of 2^n-1 to insure a continuous random delay range.

4.2.1.2 GRANT

GRANT signal enabled

- If selected, GRANT is mapped to GPIO pin and is used by PTA implementation.
- If not selected, GRANT is not mapped to GPIO pin and GRANT is always asserted.

GRANT signal active high

- If selected, GRANT is asserted when GRANT GPIO pin is high (> Vih).
- If not selected, GRANT is asserted when GRANT GPIO pin is low (< Vil).

GRANT signal GPIO port and GRANT signal GPIO pin

- Select GRANT port and pin matching circuit board configuration.
- To minimize PTA impact to other EFR32 peripherals, recommended GRANT port and pin are:

EFR32 Package	GRANT Port/Pin
QFN48	PF3
QFN32	PB15

Abort transmission mid packet if GRANT is lost

- If selected, losing GRANT during an 802.15.4 TX will abort the 802.15.4 TX.
- If not selected, losing GRANT after the initial evaluation at end of CCA will not abort the 802.15.4 TX.

4.2.1.2.1 ACK Disable

Disable ACKing when GRANT deasserted, RHO asserted, or REQUEST not secured (shared REQUEST only)

- If selected, the ACK to a valid RX packet, requiring an ACK, is **not** transmitted if GRANT is deasserted, RHO is asserted, or REQUEST is not secured (shared REQUEST only).

  Note: This feature allows completing an 802.15.4 message, regardless of PTA signals, to minimize additional retries from remote device, reducing 2.4GHz RF traffic and improving battery life.
- If not selected, the ACK to a valid RX packet requiring an ACK is transmitted regardless of GRANT, RHO, or REQUEST state.

4.2.1.3 PRIORITY

PRIORITY signal enabled

- If selected, PRIORITY is mapped to GPIO pin and is used by PTA implementation.
- If not selected, PRIORITY is not mapped to GPIO pin.

PRIORITY signal active high

- If selected, PRIORITY GPIO pin is driven high (> Voh) when PRIORITY is asserted.
- If not selected, PRIORITY GPIO pin is driven low (< Vol) when PRIORITY is asserted.

PRIORITY signal GPIO port and **PRIORITY signal GPIO pin**

- Select REQUEST port and pin matching circuit board configuration.
- To minimize PTA impact to other EFR32 peripherals, recommended PRIORITY port and pin are:

EFR32 Package	PRIORITY Port/Pin
QFN48	PD12
QFN32	PB14

Enable PRIORITY shared mode

- If enabled. PRIORITY is shared and implements open-drain or open-source I/O for multi-EFR32 radio applications.
- If active low, PRIORITY is open-drain and an external 1 kΩ ±5% pull-up is required.
- If active high, PRIORITY is open-source and an external 1 kΩ ±5% pull-down is required.
- If not enabled, PRIORITY is not shared and implements a push-pull output for single EFR32 radio applications.

TX high PRIORITY

- If selected, PRIORITY is asserted during 802.15.4 TX.
- If not selected, PRIORITY is deasserted during 802.15.4 TX.

RX high PRIORITY

- If selected, PRIORITY is asserted during 802.15.4 RX.
- If not selected, PRIORITY is deasserted during 802.15.4 RX.

Include TX PRIORITY Escalation

- If enabled. TX PRIORITY Escalation feature is compiled into firmware.
- If not enabled, TX PRIORITY Escalation feature is not compiled into firmware and CCA/GRANT TX PRIORITY Escalation Threshold and MAC Fail TX PRIORITY Escalation Threshold must be set to 0 when writing to ptaOptions via run-time API.

CCA/GRANT TX PRIORITY Escalation Threshold

- If set to 0 (000b, default):
  - CCA/GRANT TX PRIORITY Escalation is disabled.
  - PRIORITY during TX is asserted as per “TX high PRIORITY” setting.
- If set between n=1 (001b) to 7 (111b) [requires “TX high PRIORITY” set to low priority (0)]:
  - CCA/GRANT TX PRIORITY Escalation is enabled.
  - PRIORITY during TX becomes asserted high after n MAC failures due to four CCA and/or GRANT denial failures.
  - PRIORITY during TX remains asserted high until a successful MAC TX and RX ACK.

MAC Fail TX PRIORITY Escalation Threshold

- If set to 0 (00b, default):
  - CCA/GRANT TX PRIORITY Escalation is disabled.
  - PRIORITY during TX is asserted as per “TX high PRIORITY” setting.
- If set to n=1 (01b) to 3 (11b) [requires “TX high PRIORITY” set to low priority (0)]:
  - CCA/GRANT TX PRIORITY Escalation is enabled.
  - PRIORITY during TX is asserted high after n MAC failures due to CCA (four CCA failures) or MAC ACK fails (four MAC TX and RX ACK failures).
  - PRIORITY during TX remains asserted high until a successful MAC TX and RX ACK.

4.2.1.4 PWM

PWM REQUEST signal (shared REQUEST only)

- If REQUEST signal is NOT shared, PWM REQUEST signal must be set to “Disabled”.
- If REQUEST signal is shared, the REQUEST GPIO pin is used to arbitrate REQUEST between multiple EFR32 radios and a second GPIO is required to drive REQUEST||PWM. PWM REQUEST signal specifies the REQUEST||PWM GPIO.

PWM REQUEST signal level (shared REQUEST only)

- If PWM REQUEST signal is “Disabled”, then PWM REQUEST signal level selection is ignored. Else, PWM REQUEST signal is shared and implements open-drain or open-source I/O for multi-EFR32 radio applications.
- If active low, PWM REQUEST is open-drain, an external 1 kΩ ±5% pull-up is required, and PWM REQUEST GPIO pin is driven low (< Vol) when REQUEST||PWM is asserted.
- If active high, PWM REQUEST is open-source, an external 1 kΩ ±5% pull-down is required, and PWM REQUEST GPIO pin is driven high (> Voh) when REQUEST||PWM is asserted.

Enable PWM REQUEST at startup

- If enabled, PWM REQUEST is enabled at firmware startup as per specified period, duty-cycle, and priority.
- If not enabled, PWM REQUEST is enabled at firmware startup, but can be enabled via run-time API.

PWM REQUEST Period (0.5ms)

Note: PWM REQUEST Period selection cannot be an integer sub-multiple of the Wi-Fi beacon or a significant number of consecutive Wi-Fi beacons may be missed, causing AP to collapse Wi-Fi network or STA to disassociate. Silicon Labs achieves <1% 802.15.4 message receive loss with PWM REQUEST set to 39ms (or 78 half-ms) period, 20% duty-cycle, and high priority, which results in ~30% reduction in Wi-Fi TCP throughput over MCS0 to MCS7 and 20 or 40MHz bandwidth.

- Sets PWM REQUEST Period from 5ms (10) to 109ms (218) in 0.5ms steps.

PWM REQUEST Duty-Cycle (%)

Note: Large PWM REQUEST Duty-Cycle selection will substantially impact the Wi-Fi throughput as it reserved more time for 802.15.4 listening. Silicon Labs achieves <1% 802.15.4 message receive loss with PWM REQUEST set to 39ms (or 78 half-ms) period, 20% duty-cycle, and high priority, which results in ~30% reduction in Wi-Fi TCP throughput over MCS0 to MCS7 and 20 or 40MHz bandwidth.

- Sets PWM REQUEST Duty-Cycle from 1% to 95% in 1% steps.

Assert PRIORITY when PWM REQUEST asserted

- Sets PWM REQUEST PRIORITY to assert or not when PWM REQUEST asserts.

Receive Retry

- **Receive retry REQUEST enabled**
  - If selected, REQUEST is held after a corrupted receive packet or after a successful receive packet with GRANT denied until time-out expires or another packet is received.
    Note: This feature is useful to hold 2.4GHz band clear while remote device re-transmits a packet, maximizing the opportunity to receive an uncorrupted retry packet from remote device, reducing 2.4GHz RF traffic and improving battery life.
    - If not selected, REQUEST is not held after a corrupted receive packet or after a successful receive packet with GRANT denied.
- **Receive retry timeout (milliseconds) [0-255]**
  - Selects the timeout for REQUEST hold after a corrupted receive packet.
    Note: 16ms is recommended to allow for maximum 802.15.4 packet duration and MAC retry random delay.
    Note: Many Wi-Fi/PTA implementations have a maximum GRANT timeout, which should be set to received retry timeout plus 6ms to allow for maximum size corrupted packet, maximum random delay, and maximum size retry packet.
- **REQUEST high PRIORITY on receive retry**
  - If selected, PRIORITY is asserted during REQUEST hold after a corrupted receive packet or after a successful receive packet with GRANT denied.
  - If not selected, PRIORITY is deasserted during REQUEST hold after a corrupted receive packet or after a successful receive packet with GRANT denied.

4.2.1.5 Radio Hold Off

RHO (Radio Hold Off) signal enabled

- If selected, RHO is mapped to GPIO pin and is used by PTA implementation.
- If not selected, RHO is not mapped to GPIO pin and RHO is always deasserted.

RHO (Radio Hold Off) active high

- If selected, RHO is asserted when RHO GPIO pin is high (> Vih).
- If not selected, RHO is asserted when RHO GPIO pin is low (< Vil).

RHO (Radio Hold Off) signal GPIO port and RHO (Radio Hold Off) signal GPIO pin

- Select RHO port and pin matching circuit board configuration.
- To minimize PTA impact to other EFR32 peripherals, recommended RHO port and pin are:

EFR32 Package	RHO Port/Pin
QFN48	PC11
QFN32	PC10

4.2.1.6 Directional PRIORITY

Enable Directional PRIORITY

- If True:
  - Directional PRIORITY signal is connected to the Wi-Fi PTA and multiplexes priority state and radio state information.
  - Allows the Wi-Fi PTA master to obtain radio state information from the EFR32 using the Directional PRIORITY signal.
  - When requesting network air time, the EFR32 will assert a pulse on the Directional PRIORITY line depending on the requirement of that transaction and then switch to communicating the state of the radio on the same Directional PRIORITY line.
  - The Directional PRIORITY line is held low when the radio is in receive mode and is held high when the radio is in transmit mode.
  - The PRIORITY signal is not connected to the Wi-Fi PTA and is used as a Static PRIORITY input to the Directional PRIORITY logic block.
- If False:
  - The Directional PRIORITY signal is not used or connected to the Wi-Fi PTA.

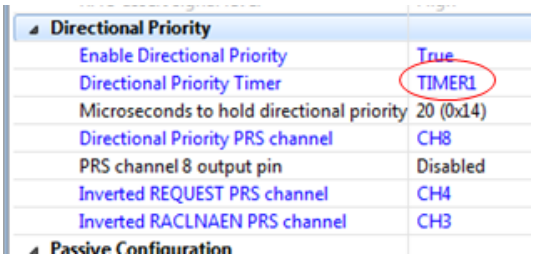
- The PRIORITY signal is connected to the Wi-Fi PATA and operates as Static PRIORITY and is either high or low during REQUEST asserted for the transmit or receive operation.

Directional PRIORITY Timer

To configure the Directional PRIORITY Timer:

1. Choose an unused Timer.

   TIMER1 is recommended for most EmberZNet PRO or Silicon Labs Thread applications due to TIMER0 is used by the IEEE 802.15.4 software stacks.



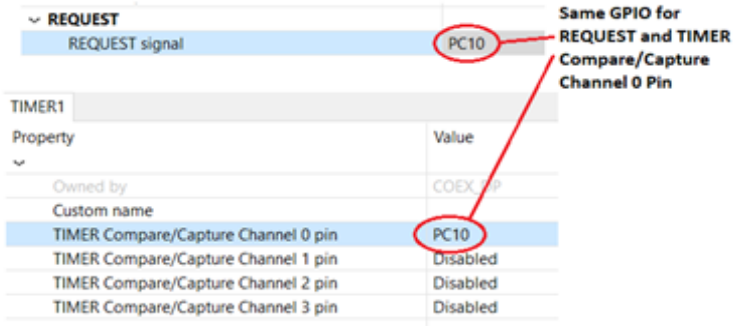
For EFR32xG2x devices, the **Timer Compare / Capture Channel** is selected by the stack code. However, for EFR32xG1x devices, the stack code does not select the Timer Compare / Capture Channel pin. Follow the steps below to select the Timer Compare / Capture Channel pin for EFR32xG1x devices.

1. Select the **Timer Compare / Capture Channel** pin.
2. Open Hardware Configurator.
3. Open the corresponding .hwconf file for the application being built.
4. On the **Configurator** tab, select the **Default Mode Peripherals** view.
5. Select the same TIMERx as selected in the Directional PRIORITY section of the coexistence plugin.

   TIMER1 is shown in this example but any available Timer can be used.

6. Change the **TIMER Compare/Capture Channel 0 pin** to match the same GPIO pin used for REQUEST.

   PC10 is shown in this example but other GPIOs can be selected. Refer to the EFR32 datasheet or reference manual to confirm GPIO availability for the TIMERx Compare/Capture Channel 0 pin.



Directional PRIORITY Pulse Width [0-255]

- Set to 20 (0x14) by default.

- Selects the hold time of the Directional PRIORITY RX Priority pulse in microseconds for a range of 1 to 255 depending on the requirement of the Wi-Fi PTA. Silicon Labs recommends the default of 20 microseconds for typical Wi-Fi PTA implementations.
- Set to 0 to bypass Directional PRIORITY.

Directional Priority PRS Channel

To configure the Directional PRIORITY PRS Channel:

1. Choose any group of four PRS channels not currently used by the SDK stack software, other plugins, or custom code.
2. Assign the highest PRS channel number from this group to the Directional Priority PRS Channel value.

   Note: The External FEM plugin is recommended to be enabled for monitoring the EFR32 radio TX activity and radio RX activity for the custom coexistence application test and development purposes and can use up to two additional PRS channels.
3. The SDK stack software automatically selects the preceding three PRS channels from the group.

   Example: Designer selects PRS channel 2 as the Directional Priority PRS Channel value. From the designer's choice, the SDK stack software automatically selects PRS channel 1, PRS channel 0 and PRS channel 11 for use in the Directional Priority PRS Channel group. In this example, the SDK stack software automatically wraps around from the lowest PRS channel number to the highest PRS channel number until all three additional required PRS channels are assigned.

PRS Channel Output Pin

- Directional PRIORITY output GPIO pin
- Connects to the Wi-Fi PTA.

Inverted Request PRS Channel

For EFR32xG2x Series 2 EFR32 devices, the **Inverted Request PRS Channel** is selected by the stack code. However, the stack code does not select the Inverted Request PRS Channel for EFR32xG1x Series 1 EFR32 devices.

- Choose any PRS channel not used by the SDK stack software, other plugins, custom code or the Directional Priority PRS Channel option.
- For EFR32xG2x, leave this option Disabled.

Inverted RACLNAEN PRS Channel

For EFR32xG2x Series 2 EFR32 devices, the **Inverted RACLNAEN PRS Channel** is selected by the stack code. However, the stack code does not select the Inverted Request PRS Channel for EFR32xG1x Series 1 EFR32 devices.

- Choose any PRS channel not used by the SDK stack software, other plugins, custom code, the Directional Priority PRS Channel option or the Inverted Request PRS Channel option.
- For EFR32xG2x, leave this option Disabled.

4.2.1.7 Radio Blocker Optimization

Notes

1. Radio Blocker Optimization is available for applications using EmberZNet PRO and Silicon Labs Thread and is available only for EFR32xG12, EFR32xG13 and EFR32xG14 EFR32 devices. Radio Blocker Optimization is not available for all other EFR32xG variants.
2. It is highly recommended to enable **Radio Blocker Optimization** on **EFR32xG12, EFR32xG13 and EFR32xG14** devices when co-located with a Wi-Fi device such as in a Gateway.
3. Run time control of **Radio Blocker Optimization** can be enabled in the hardware configurator.
4. Wi-Fi control of **Radio Blocker Optimization** is optionally enabled in the hardware configurator when run time control is enabled.

Optimize for Wi-Fi Blocker

- If True:
  - IEEE802.15.4 PHY performs better in the presence of Wi-Fi interference through improved adjacent channel rejection and improved far channel rejection.
  - IEEE802.15.4 PHY performance is slightly reduced in the presence of 802.15.4 and other narrow band interferers.
- If False: IEEE 802.15.4 PHY is optimized for Narrow band interference and will result in reduced adjacent channel rejection and far channel rejection performance in the presence of Wi-Fi interference.

This setting can be overridden by the Enable Runtime Control setting.

Enable Runtime Control

- If True: runtime control of **Wi-Fi Blocker Optimization** is available.
- If False: runtime control of **Wi-Fi Blocker Optimization** is not available.

Optional "Wi-Fi Select" Control of Blocker Optimization

- Selects the GPIO port and pin – defaults to Disabled.
- The selected GPIO pin is connected to Wi-Fi PA enable pin.
    - This feature is not available if Enable Runtime Control is False.

PHY SELECT assert signal level

- Assert high if Wi-Fi PA Enable is active high.
- Assert low if Wi-Fi PA Enable is active low.

PHY SELECT timeout (ms)

- 0 = **(Default) Optimize for Wi-Fi Blocker = false** and is always optimized for narrow band interferers. The GPIO state is ignored.
- 255 = **Optimize for Wi-Fi Blocker = true** and is always optimized for co-located Wi-Fi. The GPIO state is ignored.
- If 1 - 254 and GPIO is defined:
    - Wi-Fi Blocker Optimization is initially set per the assert state of the **PHY SELECT assert signal level** GPIO.
    - If GPIO is asserted, **Optimize for Wi-Fi Blocker** is enabled and the 802.15.4 PHY is optimized for Wi-Fi interferers.
    - If GPIO is not asserted, **Optimize for Wi-Fi Blocker** is disabled and the 802.15.4 PHY is optimized for narrow band interferers.
    - Wi-Fi's PA Enable assert triggers the **Wi-Fi Blocker Optimization**.
    - Wi-Fi's PA Enable de-assert triggers the countdown in milliseconds (1-254). When the countdown completes, the **Optimize for Wi-Fi Blocker** option is set to false.

4.2.1.8 RX Active

Note: The RX Active feature passes the 802.15.4 MODEM_FRAME_DETECT radio signal to a GPIO pin using a PRS channel. The GPIO port, pin and assert level are selected in Hardware Configurator. The signal output from the selected GPIO is then used to drive the RHO pin on other radios.

RX active PRS channel

- Selects PRS channel used to assign MODEM_FRAME_DETECT signal to an output GPIO.

PRS channel output pin

- Choose the GPIO to output the MODEM_FRAME_DETECT signal.

RX active assert signal level

- Selecting High results in a high signal output when the receive packet is detected and a low output otherwise.
- Selecting Low results in a low signal output when the receive packet is detected and a high output otherwise.

This completes the Coexistence Plugin / Configurator setup. Complete other AppBuilder application setups and generate. The coexistence configuration is saved in the application's .h file.

4.2.2 Run-Time PTA Re-configuration

The following PTA options, which can be configured at compile time via AppBuilder, can also be re-configured at run-time:

- Receive retry timeout (milliseconds) [0-255]
- Disable ACKing when GRANT deasserted, RHO asserted, or REQUEST not secured (shared REQUEST only)
- Abort transmission mid packet if GRANT is lost
- TX high PRIORITY
- RX high PRIORITY
- REQUEST high PRIORITY on receive retry
- Receive retry REQUEST enabled
- RHO (Radio Hold Off) signal enabled

- CCA/GRANT TX PRIORITY Escalation Threshold
- MAC Fail TX PRIORITY Escalation Threshold
- PWM REQUEST

Note: For descriptions of the above PTA options fields, see Section 4.2.2.1.1, PTA Option Descriptions.

The following PTA options cannot be configured via AppBuilder and can only be configured at run-time:

- Enable or disable PTA
- Disable REQUEST (force holdoff)
- Synch MAC to GRANT (MAC holdoff)
- REQUEST/PRIORITY Assert (Preamble/Synch or Address Detection)

4.2.2.1.1 PTA Option Descriptions

The descriptions of the above PTA options fields follow.

Disable REQUEST (force holdoff)

- If not set (default), REQUEST operates as per descriptions in Section 4.1.5 Single EFR32 Connected to Wi-Fi/PTA and Section 4.1.9 Multiple EFR32s Connected to Wi-Fi/PTA.
- If set, REQUEST stays disabled, effectively halting all radio TX/RX functions.

Synch MAC to GRANT (MAC holdoff)

- If not set (default), Synch MAC to GRANT is disabled for 802.15.4-compliant random MAC delays.
- If set, MAC CCA/TX is delayed until GRANT is asserted, synching all Zigbee TX operations with GRANT.
  - Synch MAC to GRANT is not strictly 802.15.4 compliant as it prevents random MAC delay execution.
  - Synch MAC to GRANT should only be enabled during known, higher priority, Wi-Fi or BT interfering activity and disabled as soon as such activity completes.

REQUEST/PRIORITY Assert (Preamble/Synch or Address Detection)

- If set to 0 (00b, default and recommended):
  - REQUEST during RX is asserted at Preamble/Synch.
  - PRIORITY during RX is asserted at Preamble/Synch, as per “RX high PRIORITY” setting.
- If set to 1 (01b) or 3 (11b) [requires “RX high PRIORITY” set to high priority (1)]:
  - REQUEST during RX is asserted at Address Detection (for this radio).
  - PRIORITY during RX is asserted at Address Detection (for this radio).
- If set to 2 (10b) [requires “RX high PRIORITY” set to low priority (0)]:
  - REQUEST during RX is asserted at Preamble/Synch.
  - PRIORITY during RX is asserted at Address Detection (for this radio).

The API function calls for re-configuring coexistence PTA vary based on SoC, EZSP, or TMSP application.

Note: For Run-Time API options not supported by selected EmberZNet PRO or Silicon Labs Thread release, the corresponding ptaOptions bit fields are RESERVED and must be written to 0.

4.2.2.2 SoC Application API

Note: For EmberZNet PRO 6.5.0 and Silicon Labs Thread 2.9.0 or later, add the following # include into the application's <xxx>-callbacks.c file to avoid warnings and errors at build time associated with API function calls described in this section:

```
#include "platform/radio/rail_lib/plugin/coexistence/protocol/ieee802154/coexistence-802154.h";
```

The following two SoC API function calls enable and disable the PTA at run-time:

```
bool halPtaIsEnabled(void);
EmberStatus halPtaSetEnable(bool enabled);
```

The following two SoC API function calls re-configure the PTA at run-time:

```
HalPtaOptions halPtaGetOptions(void);
EmberStatus halPtaSetOptions(HalPtaOptions options);
```

Where `HalPtaOptions` is a uint32_t with the following bitmap definition.

PTA Feature	Bit Position	Size (bits)
Receive retry timeout (milliseconds) [0-255]	0	8
Disable ACKing when GRANT deasserted, RHO asserted, or REQUEST not secured (shared REQUEST only)	8	1
Abort transmission mid packet if GRANT is lost	9	1
TX high PRIORITY	10	1
RX high PRIORITY	11	1
REQUEST high PRIORITY on receive retry	12	1
RHO (Radio Hold Off) signal enabled	14	1
Reserved (Reserved bits MUST be written 0)	15	1
Disable REQUEST (force holdoff)	16	1
Synch MAC to GRANT (MAC holdoff)	17	1
REQUEST/PRIORITY Assert (Preamble/Synch or Address Detection)	18	2
CCA/GRANT TX PRIORITY Escalation Threshold	20	3
Reserved (Reserved bits MUST be written 0)	23	2
MAC Fail TX PRIORITY Escalation Threshold	25	2
Reserved (Reserved bits MUST be written 0)	27	5

The following two SoC API function calls re-configure the PWM REQUEST at run-time:

```
const HalPtaPwmArgs_t *halPtaGetRequestPwmArgs(void);
EmberStatus halPtaSetRequestPwm(halPtaReq_t ptaReq, halPtaCb_t ptaCb, uint8_t dutyCycle, uint8_t
periodHalfMs);
```

Where:

```
typedef struct HalPtaPwmArgs {
  halPtaReq_t req;
  uint8_t dutyCycle;
  uint8_t periodHalfMs;
} HalPtaPwmArgs_t;
```

and

```
ptaReq/req:   0x00 => PWM REQUEST disabled
              0x80 => PWM REQUEST enabled at low priority
              0x82 => PWM REQUEST enabled at high priority
ptaCb: NULL
```

Where:

`dutyCycle`: PWM REQUEST duty-cycle from 5% to 95% in 1% steps

`periodHalfMs`: PWM REQUEST Period from 5ms (10) to 109ms (218) in 0.5ms steps

The following two SoC API function calls re-configure the Directional PRIORITY at run-time:

```
dp_pulse = halPtaGetDirectionalPriorityPulseWidth();
halPtaSetDirectionalPriorityPulseWidth(dp_pulse);
```

Where:

`uint8_t dp_pulse`: Pulse width (0 to disable, 1-255µs)

4.2.2.3 Zigbee Network Coprocessor Application using EZSP API

The following two EZSP (EmberZNet Serial Protocol) API function calls enable and disable the PTA, re-configure the PTA, and reconfigure the PWM REQUEST at run-time:

```
EzspStatus ezspGetValue(EzspValueId valueId, uint8_t *valueLength, uint8_t *value);
EzspStatus ezspSetValue(EzspValueId valueId, uint8_t valueLength, uint8_t *value);
```

Where valueId and valueLength have the following PTA related options:

EZSP Value ID	Value	Length (bytes)	Description
EZSP_VALUE_ENABLE_PTA	0x31	1	Enable (1) or disable (0) packet traffic arbitration.
EZSP_VALUE_PTA_OPTIONS	0x32	4	Set packet traffic arbitration (PTA) configuration options.
EZSP_VALUE_PTA_PWM_OPTIONS	0x35	3	Configure PWM REQUEST options.
EZSP_VALUE_PTA_DIRECTIONAL_PRIORITY_PULSE_WIDTH	0x36	1	Pulse width (0 to disable, 1-255µs)

Where PTA configuration options are a u int32_t with the following bitmap definition:

PTA Feature	Bit Position	Size (bits)
Receive retry timeout (milliseconds) [0-255]	0	8
Disable ACKing when GRANT deasserted, RHO asserted, or REQUEST not secured (shared REQUEST only)	8	1
Abort transmission mid packet if GRANT is lost	9	1
TX high PRIORITY	10	1
RX high PRIORITY	11	1
REQUEST high PRIORITY on receive retry	12	1
Receive retry REQUEST enabled	13	1
RHO (Radio Hold Off) signal enabled	14	1
Reserved (Reserved bits MUST be written 0)	15	1
Disable REQUEST (force holdoff)	16	1
Synch MAC to GRANT (MAC holdoff)	17	1
REQUEST/PRIORITY Assert (Preamble/Synch or Address Detection)	18	2
CCA/GRANT TX PRIORITY Escalation Threshold • Requires additional application code. • See Section 4.1.15 TX PRIORITY Escalation for details.	20	3
Reserved (Reserved bits MUST be written 0)	23	2
MAC Fail TX PRIORITY Escalation Threshold • Requires additional application code. • See Section 4.1.15 TX PRIORITY Escalation for details.	25	2
Reserved (Reserved bits MUST be written 0)	27	5

PWM REQUEST configuration options is a three-byte uint8_t array:

```
{uint8_t ptaReq, uint8_t dutyCycle, uint8_t periodHalfMs}
```

Where:

`ptaReq`: 0x00 => PWM REQUEST disabled

0x80 => PWM REQUEST enabled at low priority

0x82 => PWM REQUEST enabled at high priority

`dutyCycle`: PWM REQUEST duty-cycle from 5% to 95% in 1% steps

`periodHalfMs`: PWM REQUEST Period from 5ms (10) to 109ms (218) in 0.5ms steps

And the Directional PRIORITY parameter is:

`uint8_t dp_pulse`: Pulse width (0 to disable, 1-255µs)

4.2.2.4 Silicon Labs Thread Network Coprocessor Application using TMSP API

The following two TMSP (Thread Management Serial Protocol) API function calls enable and disable the PTA at run-time:

```
bool halPtaIsEnabled(void);
EmberStatus halPtaSetEnable(bool enabled);
```

The following two TMSP API function calls re-configure the PTA at run-time:

```
HalPtaOptions halPtaGetOptions(void);
EmberStatus halPtaSetOptions(HalPtaOptions options);
```

Where `HalPtaOptions` is a uint32_t with the following bitmap definition:

PTA Feature	Bit Position	Size (bits)
Receive retry timeout (milliseconds) [0-255]	0	8
Disable ACKing when GRANT deasserted, RHO asserted, or REQUEST not secured (shared REQUEST only)	8	1
Abort transmission mid packet if GRANT is lost	9	1
TX high PRIORITY	10	1
RX high PRIORITY	11	1
REQUEST high PRIORITY on receive retry	12	1
Receive retry REQUEST enabled	13	1
RHO (Radio Hold Off) signal enabled	14	1
Reserved (Reserved bits MUST be written 0)	15	1
Disable REQUEST (force holdoff)	16	1
Synch MAC to GRANT (MAC holdoff)	17	1
REQUEST/PRIORITY Assert (Preamble/Synch or Address Detection)	18	2
CCA/GRANT TX PRIORITY Escalation Threshold	20	3
Reserved (Reserved bits MUST be written 0)	23	2
MAC Fail TX PRIORITY Escalation Threshold	25	2
Reserved (Reserved bits MUST be written 0)	27	5

The following two TMSP API function calls re-configure the PWM REQUEST at run-time:

```
const HalPtaPwmArgs_t *getPtaPwmOptions(void);
EmberStatus setPtaPwmOptions(uint8_t request, uint8_t dutyCycle, uint8_t periodHalfMs);
```

Where:

```
typedef struct HalPtaPwmArgs {
  halPtaReq_t req;
  uint8_t dutyCycle;
  uint8_t periodHalfMs;
} HalPtaPwmArgs_t;
```

and

`req/request`: 0x00 => PWM REQUEST disabled

0x80 => PWM REQUEST enabled at low priority

0x82 => PWM REQUEST enabled at high priority

`dutyCycle`: PWM REQUEST duty-cycle from 5% to 95% in 1% steps

`periodHalfMs`: PWM REQUEST Period from 5ms (10) to 109ms (218) in 0.5ms steps

The following two TMSP API function calls re-configure the Directional PRIORITY at run-time:

```
dp_pulse = halPtaGetDirectionalPriorityPulseWidth();
halPtaSetDirectionalPriorityPulseWidth(dp_pulse);
```

Where:

`uint8_t dp_pulse`: Pulse width (0 to disable, 1-255µs)

4.3 Coexistence Configuration Setup Examples for Different Wi-Fi/PTA Applications

Example 1: Configure EFR32 PTA support to operate as single EFR32 with typical 3-Wire Wi-Fi/PTA

- Single EFR32 radio
- RHO unused
- REQUEST unshared, active high, PC10
  - Compatible 3-Wire Wi-Fi PTA devices sometimes refer to this signal as RF_ACTIVE or BT_ACTIVE (active high).
- GRANT, active low, PF3
  - Compatible 3-Wire Wi-Fi PTA devices sometimes refer to this signal as WLAN_DENY (deny is active high, making grant active low).
- PRIORITY, active high RX and TX (escalation disabled), PD12
  - Compatible 3-Wire W-Fi PTA devices sometimes refer to this signal as RF_STATUS or BT_STATUS (active high).

  Note: PRIORITY is static, not directional. If operated with a 3-Wire PTA expecting directional:

  - Static high PRIORITY is interpreted as high PRIORITY and always in TX mode, regardless of actual TX or RX.
  - Static low PRIORITY is interpreted as low PRIORITY and always in RX mode, regardless of actual TX or RX.
- PWM REQUEST disabled.
- Other options enabled to maximize 802.15.4 performance:
  - 802.15.4 RX and TX both at high priority
  - Receive retry REQUEST enabled with 16ms time-out and high priority.
  - Enabled ACKing when GRANT deasserted.

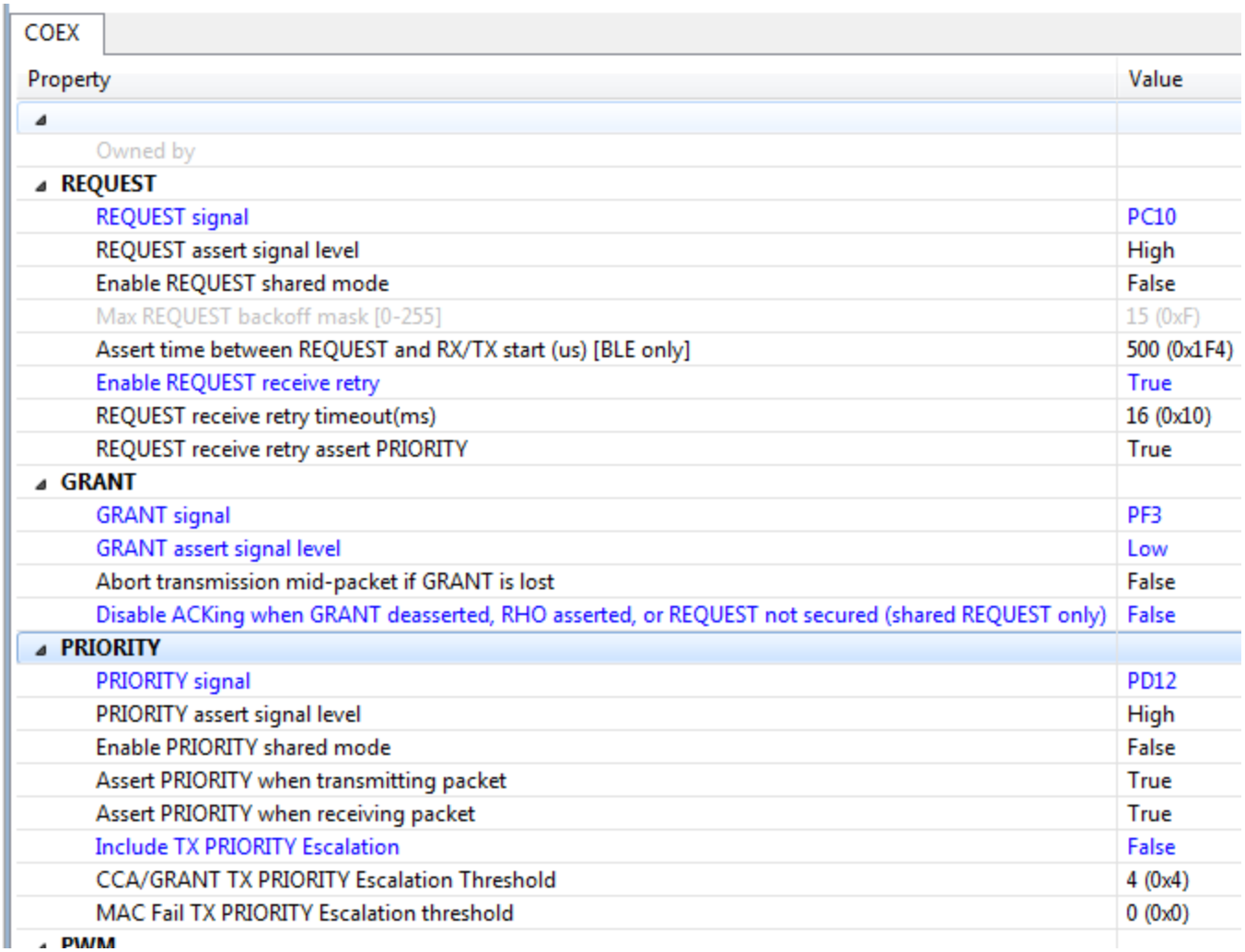


Figure 16. EFR32 PTA Support Configured to Operate as Single EFR32 with Typical 3-Wire W-Fi PTA

The logic analyzer capture in the following figure shows the PTA interface, Wi-Fi radio state, and EFR32 radio state for an EFR32 radio configured for typical 3-Wire Wi-Fi/PTA.

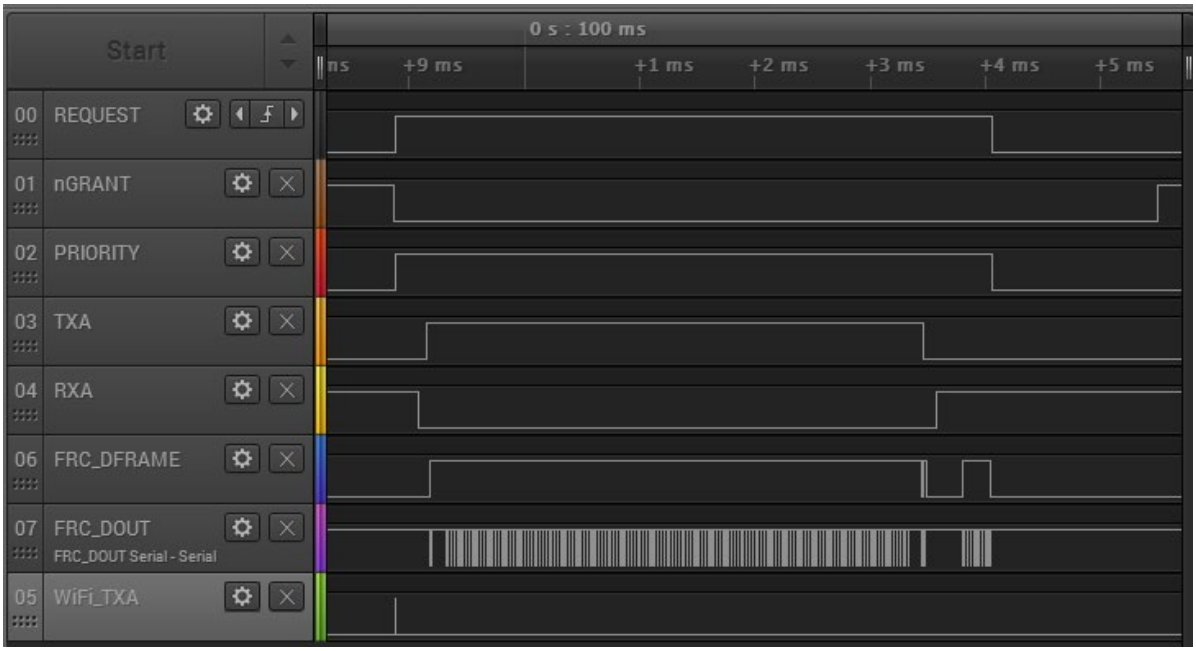


Figure 17. Example 802.15.4 TX for Single EFR32 typical 3-Wire Wi-Fi/PTA Logic Analyzer Capture

Where:

- **REQUEST**: active high, push-pull REQUEST output
- **nGRANT**: active low GRANT input
- **PRIORITY**: active high PRIORITY output
- **TXA**: EFR32 FEM TX Active control signal (configured via FEM Control plugin)
- **RXA**: EFR32 FEM RX Active control signal (configured via FEM Control plugin)
- **FRC_DFRAME**: EFR32 Frame Control Data Frame signal (packet trace frame/synch)
- **FRC_DOUT**: EFR32 Frame Control Data Out signal (packet trace data)
- **WiFi_TXA**: Wi-Fi TX Active signal

This logic analyzer sequence shows:

1. Wi-Fi starts a transmit but is immediately pre-empted (WiFi_TXA pulse) by higher priority 802.15.4 transmit asserting REQUEST and PRIORITY.
2. GRANT is asserted by Wi-Fi/PTA.
3. EFR32 radio completes CCA and CCA passes and GRANT is asserted.
4. EFR32 radio proceeds with transmit (RXA deasserts, followed by TXA assert).
5. After transmitting, EFR32 waits for ACK (TXA deasserts, followed by RXA assert).
6. EFR32 receives ACK (second FRC_DFRAME pulse). <= *802.15.4 TX message successfully completed*
7. EFR32 deasserts PRIORITY and REQUEST.
8. Wi-Fi/PTA deasserts GRANT.

Example 2: Configure EFR32 PTA support to operate with multi-radio 2-Wire Wi-Fi/PTA with active-low REQUEST

- Multiple EFR32 radios (external 1 kΩ ±5% pull-up required on REQUEST)
- RHO unused
- REQUEST shared, active Low, PC10
- GRANT, active Low, PF3
- PRIORITY unused
- PWM REQUEST disabled
- Other option settings to maximize 802.15.4 performance:
  - Enable REQUEST receive retry with 16ms time-out.
  - Do not Disable (Enable) ACKing when GRANT is deasserted RHO asserted, REQUEST not secured.

REQUEST	
REQUEST signal	PC11
REQUEST assert signal level	Low
Enable REQUEST shared mode	True
Max REQUEST backoff mask [0-255]	15 (0xF)
Assert time between REQUEST and RX/TX start (us) [BLE only]	500 (0x1F4)
Enable REQUEST receive retry	True
REQUEST receive retry timeout(ms)	16 (0x10)
REQUEST receive retry assert PRIORITY	True
GRANT	
GRANT signal	PF3
GRANT assert signal level	Low
Abort transmission mid-packet if GRANT is lost	False
Disable ACKing when GRANT deasserted, RHO asserted, or REQUEST not secured (shared REQUEST only)	False
PRIORITY	

Figure 18. EFR32 PTA Support Configures to Operate with Multi-radio 2-Wire Wi-Fi/PTA with active-low REQUEST

The logic analyzer capture in the following figure shows the PTA interface, Wi-Fi radio state, and EFR32 radio state for an EFR32 radio configured for multi-radio 2-Wire PTA with active-low REQUEST:

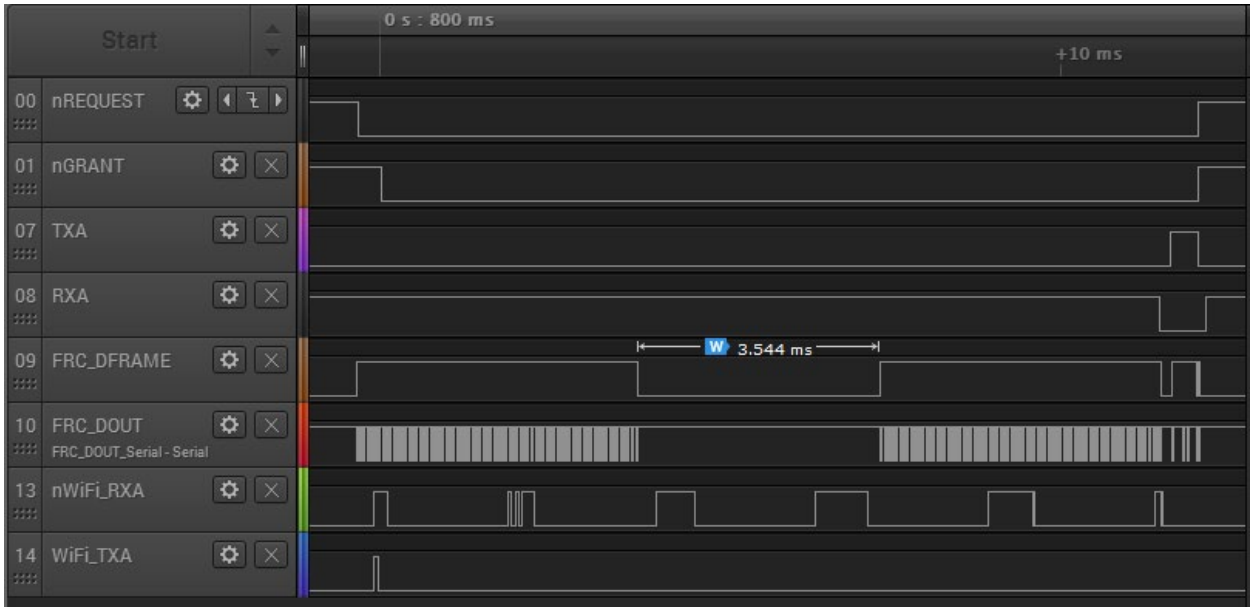


Figure 19. Example 802.15.4 RX for Multi-EFR32 2-Wire Wi-Fi/PTA with active-low REQUEST Logic Analyzer Capture

Where:

- **nREQUEST**: active low, shared (open-drain) REQUEST input/output
- **nGRANT**: active low GRANT input
- **TXA**: EFR32 FEM TX Active control signal (configured via FEM Control plugin)
- **RXA**: EFR32 FEM RX Active control signal (configured via FEM Control plugin)
- **FRC_DFRAME**: EFR32 Frame Control Data Frame signal (packet trace frame/synch)
- **FRC_DOUT**: EFR32 Frame Control Data Out signal (packet trace data)
- **nWiFi_RXA**: Wi-Fi RX Active signal
- **WiFi_TXA**: Wi-Fi TX Active signal

This logic analyzer sequence shows:

1. 802.15.4 packet is detected (FRC_DFRAME asserted) while Wi-Fi is receiving a packet (nWiFi_RXA asserted).
2. Shared REQUEST signal is tested and found not asserted by another EFR32 radio, so receiving EFR32 radio asserts REQUEST.
3. Wi-Fi ACK is transmitted (WiFi-TXA asserted) during 802.15.4 receive (no Wi-Fi TX pre-emption or higher priority Wi-Fi activity).
4. After Wi-Fi ACK completes, GRANT is asserted by Wi-Fi/PTA.
5. 802.15.4 receive is completed but CRC failed as packet was corrupted by co-located Wi-Fi ACK transmit during receive.
6. Since PTA configured with *Receive retry REQUEST enabled* using 16ms time-out, REQUEST is held up to 16ms for 802.15.4 retry with 2.4GHz quiet (Wi-Fi held off).
7. Wi-Fi continues to receive packets (nWiFi_RXA asserts) but does not ACK while 802.15.4 radio has GRANT.
8. After 3.5ms gap for end-node ACK time-out and MAC random delay, the 802.15.4 retry packet arrives and is received without error.
9. 802.15 ACK is transmitted (TXA asserted). <= *802.15.4 RX message successfully completed*
10. After 802.15.4 ACK completes, REQUEST is deasserted, followed by GRANT deassert.

Example 3: Configure EFR32 PTA support to operate with multi-radio typical 3-Wire Wi-Fi/PTA

- Multiple EFR32 radios (external 1 kΩ ±5% pull-down required on REQUEST and external 1 kΩ ±5% pull-down required on PRIORITY)
- RHO unused
- REQUEST shared, active High, PC10
- GRANT, active Low, PF3
- PRIORITY shared, active high RX and TX (escalation disabled), PD12
- PWM REQUEST disabled
- Other option settings to maximize 802.15.4 performance:
  - Enable REQUEST receive retry with 16ms time-out.
  - Do not Disable (Enable) ACKing when GRANT is deasserted RHO asserted, REQUEST not secured.

REQUEST	
REQUEST signal	PC11
REQUEST assert signal level	High
Enable REQUEST shared mode	True
Max REQUEST backoff mask [0-255]	15 (0xF)
Assert time between REQUEST and RX/TX start (us) [BLE only]	500 (0x1F4)
Enable REQUEST receive retry	True
REQUEST receive retry timeout(ms)	16 (0x10)
REQUEST receive retry assert PRIORITY	True
GRANT	
GRANT signal	PF3
GRANT assert signal level	Low
Abort transmission mid-packet if GRANT is lost	False
Disable ACKing when GRANT deasserted, RHO asserted, or REQUEST not secured (shared REQUEST only)	False
PRIORITY	
PRIORITY signal	PD12
PRIORITY assert signal level	High
Enable PRIORITY shared mode	True
Assert PRIORITY when transmitting packet	True
Assert PRIORITY when receiving packet	True
Include TX PRIORITY Escalation	False
CCA/GRANT TX PRIORITY Escalation Threshold	4 (0x4)
MAC Fail TX PRIORITY Escalation threshold	0 (0x0)

Figure 20. EFR32 PTA Support Configured to operate with Multi-radio typical 3-Wire Wi-Fi/PTA

5 Coexistence CLI Console Commands

In both Host and SoC applications, CLI commands can support run-time console control of PTA enable/disable, PTA run-time configuration (ptaOptions), and PTA debug counters. These custom CLI commands are useful in manual testing of various coexistence configurations, but also support run-time reconfiguration. The following figure shows a partial listing of the coexistence plugin commands. For more details about the CLI commands and parameters, refer to **coexistence-cli.c**.

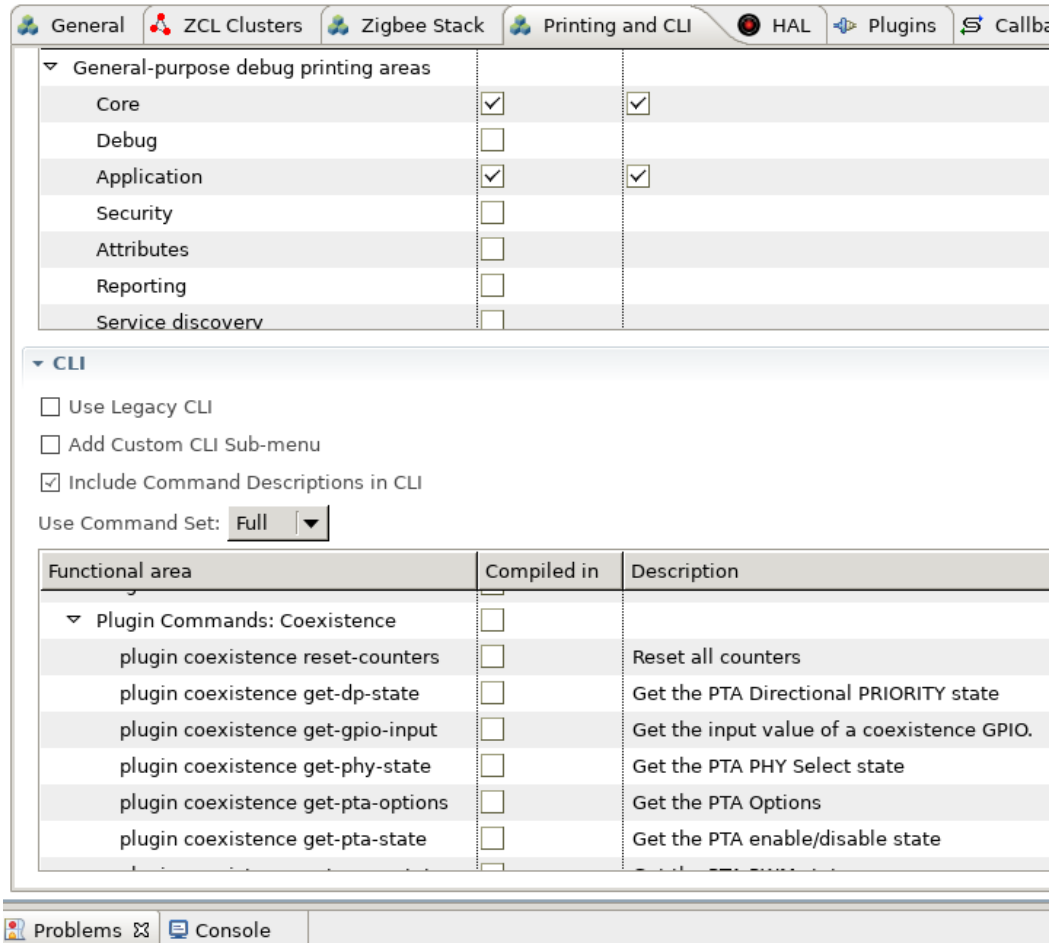


Figure 21. Coexistence CLI Console Commands (partial listing)

5.1 PTA-specific Debug Counters

MAC and APS stack counters are documented in the stack API documentation and can also be accessed via CLI. The following table describes the six coexistence PTA-specific debug counters.

Table 3. PTA-specific Debug Counters

Counter Index	Meaning
EMBER_COUNTER_PTA_LO_PRI_REQUESTED	Occurrences of REQUEST asserted with low priority
EMBER_COUNTER_PTA_HI_PRI_REQUESTED	Occurrences of REQUEST asserted with high priority
EMBER_COUNTER_PTA_LO_PRI_DENIED	Occurrences of GRANT denied with low priority REQUEST
EMBER_COUNTER_PTA_HI_PRI_DENIED	Occurrences of GRANT denied with high priority REQUEST
EMBER_COUNTER_PTA_LO_PRI_TX_ABORTED	Occurrences of TX aborted by GRANT deasserted with low priority REQUEST
EMBER_COUNTER_PTA_HI_PRI_TX_ABORTED	Occurrences of TX aborted by GRANT deasserted with high priority REQUEST

6 Coexistence Backplane Evaluation Board (EVB)

Silicon Labs' EFR32 coexistence solution can be evaluated by ordering an EFR32 Mighty Gecko Wireless SoC Starter Kit (WSTK) #SLWSTK6000B, and a Coexistence Backplane EVB (#SLWSTK-COEXBP). Please consult *UG350: Silicon Labs Coexistence Development Kit (SLWSTK-COEXBP)* for details.

7 Single-EFR32 Coexistence Test Results with a Typical 3-Wire Wi-Fi/PTA

Additional details about the implementation of managed coexistence and test results are available in an expanded version of this application note, *AN1017-NDA: ZigBee and Thread Coexistence with Wi-Fi*, available under non-disclosure from Silicon Labs technical support.

7.1 Measured Result Observations

Additional details about the implementation of managed coexistence and test results are available in an expanded version of this application note, *AN1017-NDA: ZigBee and Thread Coexistence with Wi-Fi*, available under non-disclosure from Silicon Labs technical support.

8 Conclusions

Co-located, strong Wi-Fi can have a substantial impact on 802.15.4 performance. 802.15.4 performance with co-located Wi-Fi can be improved through unmanaged and managed coexistence techniques. Unmanaged coexistence recommendations include:

1. Implement frequency separation.
2. Operate Wi-Fi with 20MHz bandwidth.
3. Increase antenna isolation.
4. Use Zigbee/Silicon Labs Thread Retry Mechanisms.
5. Remove FEM (or operate FEM LNA in bypass).

With market trends toward higher Wi-Fi TX power, higher Wi-Fi throughput, and integration of Wi-Fi and 802.15.4 radios into the same device, unmanaged techniques alone may prove insufficient, so that a managed coexistence solution is required. Even with a managed coexistence solution, all unmanaged coexistence recommendations are still necessary. Managed coexistence utilizes:

1. Wi-Fi/PTA devices providing 802.15.2-derived Packet Traffic Arbitration.
2. Silicon Labs' EFR32 PTA solution:
    1. Implement one to four GPIOs as a combination of REQUEST, GRANT, PRIORITY, and RHO (two additional GPIOs are required to implement the *PWM with High-Duty Cycle Wi-Fi* feature).
    2. Silicon Labs' AppBuilder coexistence-configuration plugin to configure EFR32 PTA support for available GPIO pins and for compatibility with the chosen Wi-Fi/PTA device.
    3. Silicon Labs' API, supporting run-time PTA reconfiguration.

Wi-Fi/802.15.4 coexistence test results show substantial 802.15.4 performance improvements when PTA is utilized:

1. Improved device join success:
    1. However, device join utilizes broadcast messages, which are not retried.
    2. *If possible, device join success can be further improved by temporarily reducing Wi-Fi traffic during devices joining 802.15.4 network.*
2. Substantially reduced MAC retries:
    1. Reduces message latency.
    2. Improves end-node battery life.
    3. Frequency separation remains important, as best managed coexistence performance is for "far-away" channels.

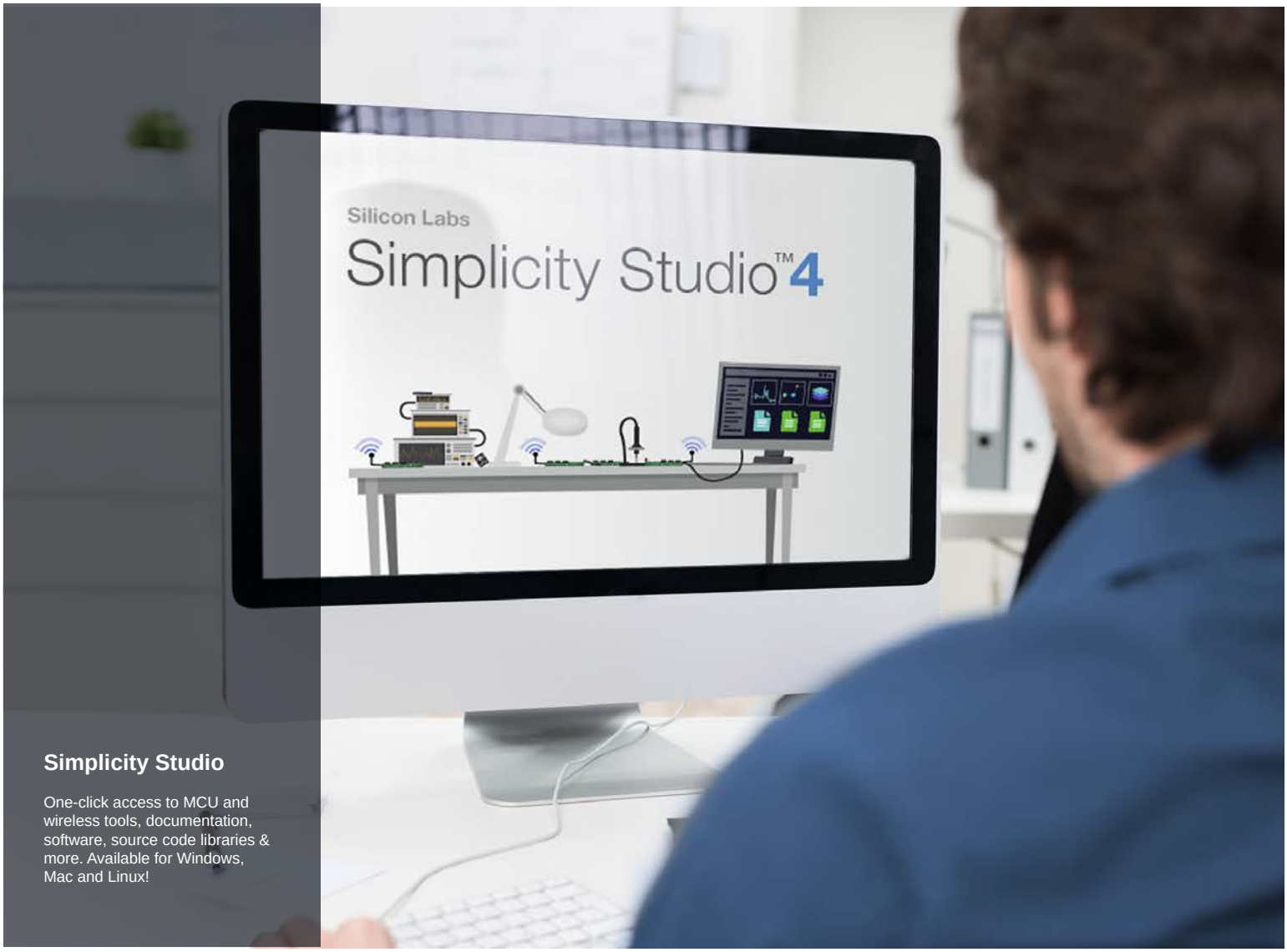
Substantially reduced message failure:

1. 802.15.4 network remains operational, even during high Wi-Fi duty cycles.

9 Revision History

Revision	Date	EmberZNet PRO	Silicon Labs Thread	Summary of Major SDK and Document Changes
1.7	July 2019	6.6.1.0	2.10.1.0	• Tied this application note to specific SDK versions of EmberZNet PRO and Silicon Labs Thread. Removed references to all earlier versions of each SDK and Table 2. • Application code coexistence extensions moved to EmberZNet PRO SDK. Deleted all text related to application code and entire Section 5. • Coexistence CLI commands added to EmberZNet PRO SDK. Added new Section 5. • Removed all references to EM35x/EM358x. • Updated Section 4.1.16 PWM for High Duty Cycle Wi-Fi.
1.6	Apr 2019	6.5.4	2.9.4	• Explained the different implementations of the Timer Compare / Capture Channel, the Inverted Request PRS Channel, and the Inverted RACLNAEN PRS Channel for EFR32xG2x and EFR32xG1x devices. • Added Figures 15–19 with associated text. • Updated Table 2.
1.5	Feb 2019	6.5.0	2.9.0	• Added Directional PRIORITY timing diagrams. • Directional PRIORITY and Passive Configuration are both integrated within the SDK. • Updated Table 2. • Added new section on how to implement Directional PRIORITY. • Added new section on Passive Configuration. • Added note in Section 4.2.2.1 SoC Application for how to avoid warnings and errors at build time. • Added Section 5, Application Code Coexistence Extensions. • Updated Table 2.
1.4	Sept 2018	6.4.0	2.8.0	• Updated Table 2.
1.3	May 2018	6.3.0	2.7.0	• TX PRIORITY Escalation and PWM for High Duty Cycle Wi-Fi are both integrated within the SDK. Application sample code is not required. • REQUEST GPIO pin and PRIORITY GPIO pins are always configured as push-pull regardless of the shared setting and should not be used in multi-EFR32 configurations. • GRANT deasserts occurring in the initial stages or slightly before a TX event may not be detected and the TX event can continue unaborted. • Updated Table 2 • Replaced Figure 2 and revised associated text.
1.2	Feb 2018	6.2.0	2.6.0	• PRIORITY signal can be implemented as static or directional (renamed from time-shared). • Updated Table 2.

Revision	Date	EmberZNet PRO	Silicon Labs Thread	Summary of Major SDK and Document Changes
1.1	Dec 2017	6.1.0	2.5.0	• EFR32 PRIORITY signal is implemented directly in the SDK and can be configured as “shared” with Hardware Configurator. There is no need for additional application code. • Removed instructions for adding custom CLI commands from AppBuilder and placed the instructions in pta-custom-cli.c. • Removed instructions from AppBuilder for adding an always active event to an application and placed the instructions in prevent-idle-sleep.c. • Removed instructions from AppBuilder for adding TX PRIORITY Escalation to an application and placed the instructions in tx-priority-escalation.c. • Application code to add PWM run-time control is available in pta-pwm.c. • Updated Table 2.
1.0	Sept 2017	6.0.0	2.4.0	• Moved coexistence feature setup from the Coexistence Configuration plugin to Hardware Configurator. • Updated Table 2.



Simplicity Studio

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

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

SILICON LABS

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

http://www.silabs.com