AN1315: *Bluetooth*® Mesh Device Power Consumption Measurements

Silicon Labs offers a complete portfolio of fully-certified modules and System On a Chip (SoC) solutions for Bluetooth® mesh connectivity that are known collectively as the EFR32BG family. This application note describes Low Power Node (LPN) and Friend node operation and the parameters related to power consumption. It also describes how to measure the power consumption of EFR32BG devices acting as Bluetooth mesh LPNs using the setup and procedures recommended in *AN969: Measuring Power Consumption on Wireless Gecko Devices*.

KEY POINTS

- One EFR32BG WSTK is required.
- Another device can be an EFR32BG WSTK or an equivalent product.
- Test examples are available in the Bluetooth Mesh SDK.
- Results from the test example are provided using an EFR32BG21.

1 Bluetooth Mesh Node Power Consumption

Bluetooth mesh nodes (other than Low Power nodes) are always listening for messages from the other nodes. Because the nodes are always on and receiving, they consume quite a lot of power and are typically connected mains or respective power sources (for example, coin or button cell batteries might have too short a lifespan for actual usage). You can learn about the radio/power operation and consumption in the datasheets for your part, SoC, or module.

The *Mesh Profile Bluetooth® Specification* Revision: v1.0.1 defines a Low Power Node, Friend node, and the concept of friendship as follows:

- Low Power Node: The ability to operate within a Bluetooth mesh network at significantly reduced receiver duty cycles. Minimizing the time the radio receiver is on leads to lower power consumption with the node only enabling the receiver when strictly necessary. LPNs achieve this through establishing a friendship with a Friend node.
- Friend Node: The ability to help an LPN operate by storing messages destined for the LPN and only forwarding them to it when the LPN explicitly requests messages from the Friend node.

1.1 Low Power Node and Friend Node

An LPN must first make a friendship with a Friend node. Once this has been established, the Friend node will receive and queue various messages going to the LPN on behalf of the LPN. There are configurable limitations for the messages that the Friend node can store for LPNs to control Friend node memory usage. Because the LPN does not have to be on and receiving all the time, it can sleep and periodically check the incoming messages from its Friend node. The LPN can send a mesh message to the Friend node and other nodes at any time. There are also some mechanisms like timeouts and maximum retries set for the communication and if those occur, the friendship will end. The friendship does not remain after restarting the device (LPN or Friend node). However, it is always possible to establish the friendship again.

1.2 Creating a Friendship

When the Friend node is started, it does not have any friendships with LPNs; it is just waiting for a Friend Request from LPNs. A Friend node can have several friendships with LPNs at the same time, depending on the Friends resources. After receiving Friend Requests with requirements that the Friend node can serve, the Friend node will reply with a Friend Offer. An LPN can also have requirements that are not compatible with the Friend node. In this case, the Friend node will not reply.

When the LPN is started, it also does not recall any friendships but starts transmitting Friend Request and waits for Friend Offer replies. Currently an LPN can have only one friendship at a time in the Silicon Labs implementation. When establishing a friendship, the LPN uses a great deal of power, so it usually wants to find a reliable friendship quite soon and not waste any power when searching for Friend nodes.

As the LPN sends Friend Requests, it will inform the Friend node of its requirements such as the PollTimeout and ReceiveDelay parameters (see 1.4.1 PollTimeout and 1.4.2 ReceiveDelay). As a return it can get several Friend Offers from Friend nodes. The Friend node provides information about itself such as ReceiveWindow size, message queue size, message subscription list size and RSSI measured from the Friend node.

The timing parameters are illustrated in Figure 3.17: Friendship timing parameters of the *Mesh Profile Bluetooth® Specification* Revision: v1.0.1.

After receiving Friend Offers, a very good Friend Offer, or waiting for Friend Offers for a while (depending on the implementation), the LPN will use some algorithm to select a Friend Offer that fits the best and sends a Friend Poll message to the selected Friend node. The Friend node replies with a Friend Update message which concludes the process by providing the security parameters and the friendship is established.

The friendship establishment is illustrated with one Friend node in Figure 3.19: Establishment of a friendship and for multiple Friend nodes in Figure 3.21: Friend establishment example of the *Mesh Profile Bluetooth® Specification* Revision: v1.0.1.

1.3 Maintaining the Friendship

A Friend node is always listening for messages. If it does not receive certain messages (typically a Friend Poll) from an LPN before the PollTimeout has passed, the Friend node will terminate the friendship. If the Friend node receives a correct message from an LPN, it will send a message back after ReceiveDelay during the ReceiveWindow. The Friend node responds to Friend Poll messages with queued Bluetooth mesh messages that could be normal messages, configuration messages, or security updates. The Friend node informs the LPN that it has no more queued messages using a Friend Update reply with the MD flag set to 0. Other messages or Friend Update replies with the MD flag set to 1 will continue being sent to the LPN.

An LPN can send a Bluetooth mesh message at any time to a Friend node or other nodes. But it has to send certain messages (typically a Friend Poll) to its Friend node before the PollTimeout has passed; otherwise, the Friend node terminates the friendship. Also, if the Friend node does not reply to LPN messages after several retries (the default is three times), the Friend node terminates the friendship.

PollTimeout timer is illustrated in Figure 3.18: Poll Timeout timer illustration of the *Mesh Profile Bluetooth® Specification* Revision: v1.0.1.

1.4 Low Power Node Settings

The following subsections describe the LPN settings related to the power consumption.

1.4.1 PollTimeout

The PollTimeout setting has the following characteristics:

- Establishes a maximum time which may elapse between two consecutive requests sent by the LPN to its Friend node. If no requests from an LPN are received by the Friend node before the PollTimeout timer expires, the Friend terminates the friendship.
- Duration: 10 to 3,455,999 in 100 ms = 1 sec to 345,600 sec = 5,760 min = 96 hours = 4 days
- The longer the PollTimeout, the more the LPN can sleep. On the other hand, this also means that the LPN receives messages less often and it takes more time for other nodes or the Provisioner to communicate with the LPN.
- This is the most important parameter when controlling the LPN power consumption.

1.4.2 ReceiveDelay

The ReceiveDelay setting has the following characteristics:

- The time which elapses between the LPN sending a request to the Friend node and the Friend node starting to listen for a response. This allows the Friend node time to prepare its response and send it back.
- Duration: 10 to 255 ms
- Requested by the LPN.
- If it is too small, the Friend node might not send a Friend Offer.
- Can be adjusted by the application.
- Does not have much effect on LPN power consumption.

1.5 Friend Node Settings

The following subsections describe the Friend node settings.

1.5.1 ReceiveWindow

The ReceiveWindow setting has the following characteristics:

- The time that the LPN spends listening for a response.
- Duration: 1 to 255 ms
- Defined by the Friend node.
- The smaller the ReceiveWindow, the less time the LPN needs to stay on and listen for messages if no message is received.
- If there is a response, listening stops immediately.
- Usually set by the system, not the application.
- Different Friend node brands have different ReceiveWindow sizes.
- Silicon Labs products normally have a ReceiveWindow of 10 to 20 ms.
- If the maximum of 255 ms is used for the ReceiveWindow setting in Friend node, the LPN power consumption can be much larger, as much as 25 times larger if no message is received in time.
- The LPN can affect power consumption by selecting a Friend node with the smallest ReceiveWindow.
- ReceiveWindow has a significant effect on LPN power consumption if the connection is not perfect. However, with a good signal and connection, the size of the ReceiveWindow does not matter.

1.6 Power Consumption-Related Guidelines

- Proxy feature uses a lot of power so avoid using this frequently. See the measurements in Figure 3.3 and Figure 3.4.
- Transmit as few messages as possible as transmission always increases power consumption noticeably.
- Subscribe as few messages as possible.
- Do not change subscriptions often.
- Target fewer unicasts to the LPN.
- Be sure that the LPN receives a Friend node, and the friendship stays (that is, the signal is sufficient). An LPN that is just on the signal edge could make a friendship, then lose it, and make it again. Going back and forth like this is not very efficient because establishing every friendship consumes more power than being in a friendship.
- Do not restart the LPNs or Friend nodes often as the friendship has to be established again.
- When an LPN is establishing a friendship, do not blindly keep trying in a busy loop. Have a backoff policy if no acceptable Friend Offers are received.
- Acknowledged messages and especially segmented acknowledged messages for other nodes might be slow as the acknowledgements return via the Friend node. This also might have an effect in power consumption.
- Do not keep the node in unprovisioned state because it will remain awake and consume high current, accordingly.

For more detailed information about the LPN, Friend node, friendship, and timing parameters, see chapter 3.6.6 Friendship of the *Mesh Profile Bluetooth® Specification* Revision v1.0.1.

2 Setup

1. Obtain two Bluetooth Wireless Starter Kits suitable for Bluetooth mesh.
2. Build and install the applications as described in 2.1 Build and Install the Applications.
3. Provision and control the operation using the Silicon Labs Bluetooth mesh mobile application.
4. See *AN969: Measuring Power Consumption on Wireless Gecko Devices* for the power measurement setups and how to make the measurements. You can either use a separate DC Power Analyzer device or WSTK Advanced Energy Monitoring (AEM) with the Simplicity Studio Energy Profiler tool which is used in this application note.

2.1 Build and Install the Applications

To observe power consumption, first program the devices with a suitable test application. Follow these steps:

1. If you have not already done so, install Simplicity Studio and the Bluetooth Mesh SDK. The Bluetooth Mesh SDK includes several software examples to create Bluetooth mesh application projects. Follow the directions in *QSG176: Silicon Labs Bluetooth® Mesh SDK v2.x Quick-Start Guide* to build and flash the example project to the device.
2. Program the Bluetooth Mesh - SoC Switch Low Power example onto the device with the WSTK. You can either use a ready-built demo (which includes the bootloader) or create a source code example, build it, and flash it onto the device.
3. Program the Bluetooth Mesh - SoC Light example onto another device. You can use a ready-built demo or create a source code example, build it, and flash it onto the device.

   Remember when using the source code examples that there is also a bootloader. The easiest way to get this is to install the demo first and then your own application if you do not want to create a bootloader example, build it, and flash it.

   If you create a source code example, the Low power node settings are in the Studio Component named Low Power Node (Bluetooth Mesh->Features->Low Power Node) as shown in the following figure.

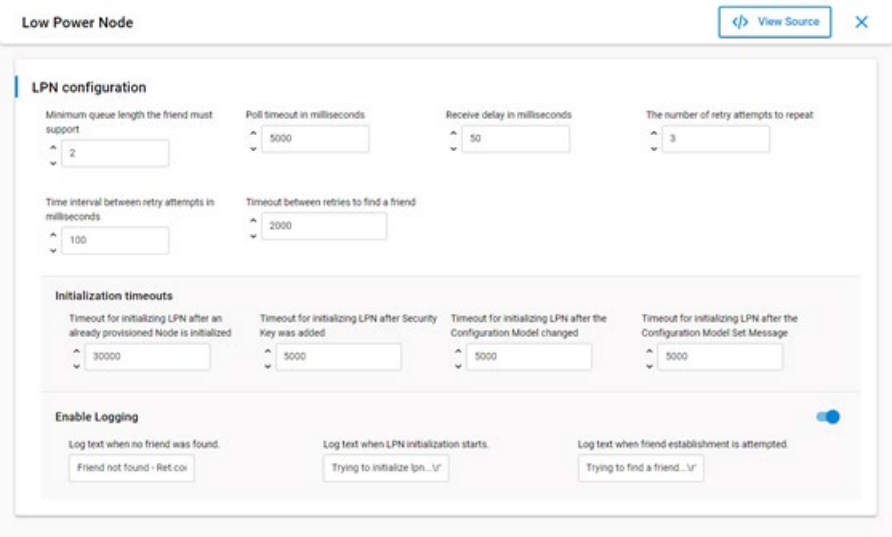

Figure 2.1. Low Power Node

You can also find these additional power consumption-related settings:

- During friendship: PollTimeout, Poll timeout in ms, LPN_POLL_TIMEOUT, default 5000 ms
- When finding friendship: Timeout in ms between retries to find a friend, LPN_FRIEND_FIND_TIMEOUT, default 2000 ms

There are also communication retry, Friend node, and initialization settings.

The Friend node source code has fewer settings. Those are located in the Studio Component named Bluetooth Mesh Stack (Bluetooth Mesh->Bluetooth Mesh Stack). The settings are memory consumption-related, not power consumption-related, such as Maximum number of Friendships allowed and others.

4. After flashing the example, both nodes should be in an unprovisioned state. If this is not the case (or you are not sure because there is no display or other issue), Silicon Labs recommends using Simplicity Commander to empty the flash before and then flash the demos and examples. For more information, see *UG162: Simplicity Commander Reference Guide*.

5. Now you are ready to start the power measurements with a separate DC Power Analyzer tool or using Simplicity Studio Energy Profiler. For more information, see *AN969: Measuring Power Consumption on Wireless Gecko Devices*.

3 Measurements

The following examples use this two-node Bluetooth mesh setup:

- Switch LPN: SLWSTK6023A EFR32xG21 Bluetooth Starter Kit with a main board and radio board running the Bluetooth Mesh - SoC Switch Low Power example. The example has LPN, Proxy (communication with Bluetooth LE devices like the mobile phone connection), and Relay features.
- Light node acting as a Friend node: SLWSTK6023A EFR32xG21 Bluetooth Starter Kit with a radio board running the Bluetooth Mesh – SoC Light example having Friend, Proxy, and Relay features. Also, some other Bluetooth mesh-compatible boards can be used instead of this. See *QSG176: Bluetooth® Mesh SDK v2.x Quick-Start Guide* for a list of compatible products.

3.1 Unprovisioned Low Power Node

Because the LPN using the Bluetooth Mesh - SoC Switch Low Power application is still unprovisioned, the current measurement should look like the following figure. Be sure to measure the Bluetooth Mesh - SoC Switch Low Power node not the Bluetooth Mesh - SoC Light node.

Figure 3.1. Current Profile for EFR32xG21 Series 2, unprovisioned

The current measurement is on a steady high level (average 10.68 mA in the example) because the node is running and waiting for provisioning. When using some a device family other than the EFR32xG21, the current might be different.

3.2 Low Power Node Looking for a Friend Node

Open the Bluetooth mesh mobile application and provision the Bluetooth Mesh - SoC Low Power Switch node. Refer *QSG176: Bluetooth® Mesh SDK v2.x Quick-Start Guide* for mobile application and provisioning instructions. At this point, provision only the Switch and do not change any Device Configuration settings at this time.

After provisioning, remember to return to the mobile application main level so that the Bluetooth connection with the mobile phone is closed. You should then see a current profile like the following figure.

Figure 3.2. Current Profile for EFR32xG21 Series 2, Looking for a Friend Node, with Proxy

On average the current measurement is now much lower (3.94 mA in this example). The approximately one second lasting high current part occurs when the LPN is looking for a Friend node by sending Friend Request messages. The lower current part with spikes that takes approximately two seconds occurs when the device is sleeping (no Bluetooth mesh communication) and using the Proxy feature (Bluetooth LE device connectivity).

By changing the parameter LPN_FRIEND_FIND_TIMEOUT it is possible to affect the power consumption when the LPN is looking for a Friend node. The LPN_FRIEND_FIND_TIMEOUT default is two seconds in the example (the part with lower current levels and spikes).

3.3 Low Power Node Friendship Established

Provision Bluetooth Mesh - SoC Light (the Light node) also acting as a Friend node. Now the Switch LPN establishes a friendship with the Light node.

The power measurements look like the following figure.

Figure 3.3. Current Profile for EFR32xG21 Series 2, Friendship established, with Proxy

The current consumption is now only 736 µA on average which is much less than the original over 10 mA measurement.

3.4 Low Power Node Friendship Established with Proxy turned Off

Turn the Proxy off: Turn off the Bluetooth Mesh - SoC Light node to make sure the device is connecting directly to the Bluetooth Mesh - SoC Low Power Switch node. Use the mobile application to connect the network, select the Bluetooth Mesh - SoC Low Power Switch node, refresh the Proxy feature, and then disable it. Also remember to return to the mobile application main level so the Bluetooth connection with the mobile phone is closed. Now turn the Bluetooth Mesh - SoC Light node back on to establish friendship again. The new measurements should look like the following figure.

Figure 3.4. Current Profile for EFR32xG21 Series 2, Friendship established, no Proxy

The current measurement is now only 6.6 µA. There is no communication visible as the poll period (PollTimeout) is so long that it is not detected. A longer—over 120-second measurement—appears in the following figure.

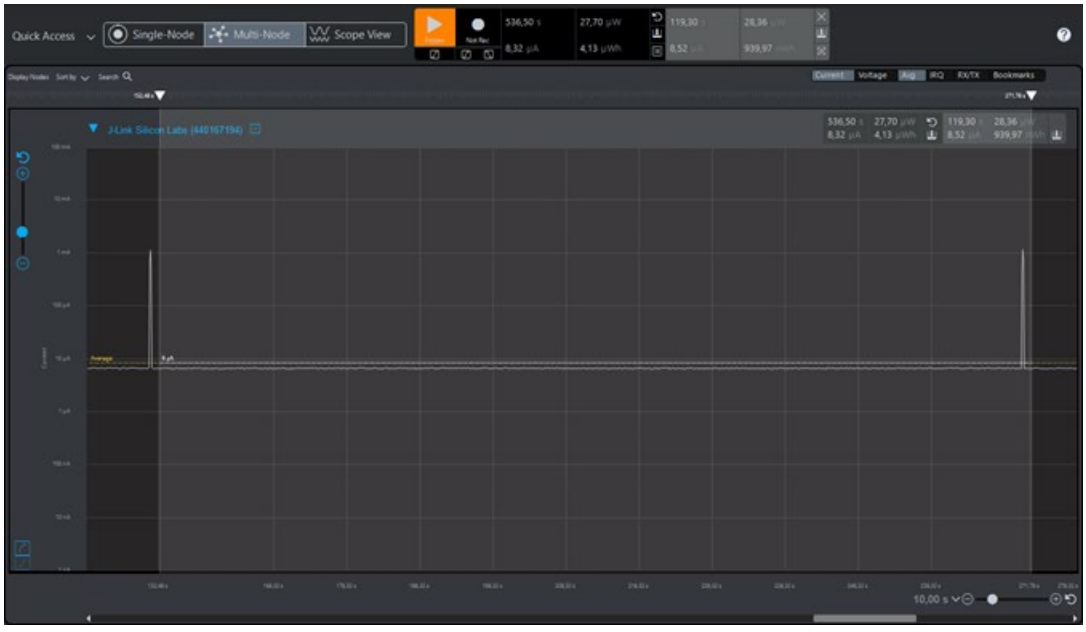

Figure 3.5. Current Profile for EFR32xG21 Series 2, Friendship established, no Proxy, long PollTimeout

There is now just a little less than 120 seconds between the Friend Polls and the current consumption is 8.5 µA on average. The communication current spike after every about 120 seconds looks like the following figure.

Figure 3.6. Current Profile for EFR32xG21 Series 2, Friendship established, no Proxy, Friend Poll & Update

There are two parts in the communication:

- Sleeping time of 119.3 seconds using average current of 6.6 µA.
- Friend Poll – Friend Update that takes about 67 ms and average current is 3.22 mA.

These are a few other examples with different PollTimeout settings:

- 120 seconds PollTimeout: (120 s * 0.0066 mA + 0.067 s * 3.22 mA) / (120 s + 0.067 s) = 8.4 µA (received what was measured which is correct)
- 10 seconds PollTimeout: (10 s * 0.0066 mA + 0.067 s * 3.22 mA) / (10 s + 0.067 s) = 28 µA
- 1 seconds PollTimeout: (1 s * 0.0066 mA + 0.067 s * 3.22 mA) / (1 s + 0.067 s) = 210 µA = 0.21 mA

As Bluetooth mesh uses Bluetooth communication, refer to *AN1246: EFR32BG SoC Bluetooth® Smart Device Power Consumption Measurements* for more detailed information.

Simplicity Studio

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

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

SILICON LABS

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

www.silabs.com