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

This document describes how to exploit the different features of Bluetooth technology to achieve the minimum possible energy consumption for a given use case. The document lists the most common use cases a Bluetooth developer may encounter and suggests a solution for each of them. This provides an ideal starting point if you are already somewhat familiar with Bluetooth, but still not sure about where to start your development.

KEY POINTS

- Power optimization
- Bluetooth features suited for different use cases

1 Introduction

Bluetooth Low Energy has the potential to add wireless connectivity to your device while keeping its energy consumption low. To achieve low energy consumption, however, you must be aware of the possibilities, and choose wisely. Different use cases may require different approaches and different Bluetooth features. In many cases, for example, you do not even need to create a Bluetooth connection to communicate, and even if you choose to create a connection, you should tune a number of parameters for optimum power reduction.

This guide lists the most common use cases a Bluetooth developer may encounter and suggests a solution for each of them. If you find one on this list that suits your needs, we strongly recommend you start your development here. The following use cases are discussed:

- I want to signal the presence of a device.

  This use case discusses Bluetooth beacons, where a single device broadcasts a short message (usually an ID) to be picked up by other Bluetooth-capable devices. Often used for finding a given asset that transmits the message or for geo-fencing (where the presence of the message signals that you entered an area).
- I want to broadcast data (such as information about a product/artwork).

  In this use case not only an ID is broadcasted by the device, but also some additional information, for example the price of a showcased product, a description of a painting, a URL that helps you find more information about a company that advertises itself, and so on. The data may also be dynamic, changing slowly or rapidly.
- I want to broadcast / send data to low-power receivers.

  You may have hundreds of devices that need to be updated with some information from time to time (for example. updating their real time). Data broadcast can be solved with Bluetooth beacons (advertisements). But what if the receivers are not smartphones and they also need to be low-power? For example, you have hundreds of asset tags that are being tracked with Bluetooth Direction Finding, and you want to update each of them with some information.
- I want to turn lamps on and off.

  This is a very common use case, where you want to issue short commands to nearby devices over Bluetooth. Your first thought might be to use connections, but sometimes it is much faster and less energy-demanding to send commands in advertisements. Low-power switches/controllers are discussed as well as low-power controlled devices.
- I want to connect to a device occasionally to exchange some data.

  Sometimes you need a connection even for short messages, either because of security, privacy, reliability, or simply because you need two-way communication. In this use case we discuss short-term connections, when a connection is only made to exchange some data quickly.
- I want to maintain a long-term connection with occasional data exchange.

  Keeping a connection alive may actually require less power than disconnecting, advertising, and then creating a new connection. This approach may be useful between fixed installation devices (such as a thermostat and an air-conditioner) that exchange some data from time to time, or between a smart phone and a smart watch. Manually-controlled connections (such as configuring a device with a smartphone app) are also considered as long-term connections with occasional data exchange.
- I want to send a lot of data on a long-term connection.

  You may need a continuous wireless communication, for example with a wireless keyboard or other HID (human interface device). File transfer and audio transfer also fall under this case, although they are not typical Bluetooth Low Energy use cases. In this case you can still tune and dynamically change the connection parameters and the TX power to achieve the lowest possible energy consumption.
- I want to exchange data with many peripheral devices.

  A very common use case is that your central device needs to communicate with many peripheral devices. The ideal solution in this case depends on the number of peripheral devices, which may be two or two thousand.

If you have a different use case, and you need some tips for low energy consumption, you can always find help at https://community.silabs.com/.

The figures in Use cases 2, 3, and 5 are from the Bluetooth Core Specification at https://www.bluetooth.com/specifications/specs/core-specification/.

2 Use Case #1: I want to signal the presence of a device

Although Bluetooth is primarily used for creating a point-to-point wireless connection between two devices, and Bluetooth advertisements are mainly used to make this possible, there are use cases where an advertisement is used for simply advertising the presence of the device. This may be useful if you want to:

1. Find the device or
2. Mark a spot in space or signal that you entered a specific zone.

In both cases, the only thing you want to broadcast in the advertisements is an ID that identifies the device or the spot/zone.

Bluetooth feature to be used: Legacy non-connectable advertisements (beacons).

Legacy Bluetooth advertisements can transmit up to 31 bytes of data with a minimum interval of 20 ms. The two main types of advertisements are connectable and non-connectable advertisement. Connectable advertisements typically advertise the device name and optionally a service UUID that describes the main functionality of the device. Non-connectable advertisements (Bluetooth beacons) typically advertise an ID or URL and sometimes their TX power (or the RSSI level at 1 m distance) so that a receiver can roughly calculate their distance based on the measured RSSI.

The format of the payload is specified by the Bluetooth advertising data format, as described in the following article: https://docs.silabs.com/bluetooth/3.2/general/adv-and-scanning/bluetooth-adv-data-basics.

The advertising data types are specified by the Bluetooth SIG here: https://btprodspecificationrefs.blob.core.windows.net/assigned-numbers/Assigned%20Number%20Types/Generic%20Access%20Profile.pdf

Although the content of the advertisement can be anything as long as it complies with the advertising data format, two formats are widely used.

- iBeacon is a standard format defined by Apple: https://developer.apple.com/ibeacon/
- Eddystone beacon is a standard format defined by Google: https://github.com/google/eddystone

Bluetooth API to be used:

- `sl_bt_advertiser_create_set()`
- `sl_bt_advertiser_set_timing()`
- `sl_bt_advertiser_set_channel_map()`
- `sl_bt_advertiser_set_data()`
- `sl_bt_advertiser_start()`
- `sl_bt_system_set_tx_power()`

Tips for low power consumption:

- Set the advertising interval to as long as your application allows. For example, if you want to signal that you entered a zone, a one-second interval is more than enough. Obviously, shorter intervals result in lower latency, but the shorter the interval the higher the power consumption. Advertising interval can be set with `sl_bt_advertiser_set_timing()`.
- Make sure that you put the device into deep sleep mode (EM2) between two advertisements by installing the *Power Manager* software component and not installing any component that requires EM1 (such as UART or logging). If you start with the **Bluetooth – SoC iBeacon** or **Bluetooth – SoC Empty** example applications, this is pre-configured.
- If you plan to wait many seconds between beacons, consider putting the device into EM4 mode, which saves even more energy. See the following software example:

  https://github.com/SiliconLabs/bluetooth_applications/tree/master/bluetooth_using_em4_energy_mode_in_bl_ibeacon_app
- Make sure that you use non-connectable mode (use *sl_bt_advertiser_non_connectable* as the *connect* parameter when you call `sl_bt_advertiser_start`). Although connectable advertisements can also be used for broadcasting an ID, they consume a bit more energy since they are listening for connection requests after each advertisement.
- By default Bluetooth advertisements are sent on 3 channels, so that they can still be received if there is interference on one or two of the channels. If you know which Bluetooth channels are free of interference and which are blocked (for example because you know the Wi-Fi channels used in your area), you can set the channel map so that advertisements are only sent on free channels. This saves a lot of energy. Use `sl_bt_advertiser_set_channel_map()`.
- Set the TX power so that the beacon can be seen in the area you want it to be seen. Do not set it too high if not necessary. TX power can be set with `sl_bt_system_set_tx_power()`.

3 Use Case #2: I want to broadcast data (such as information about a product/artwork)

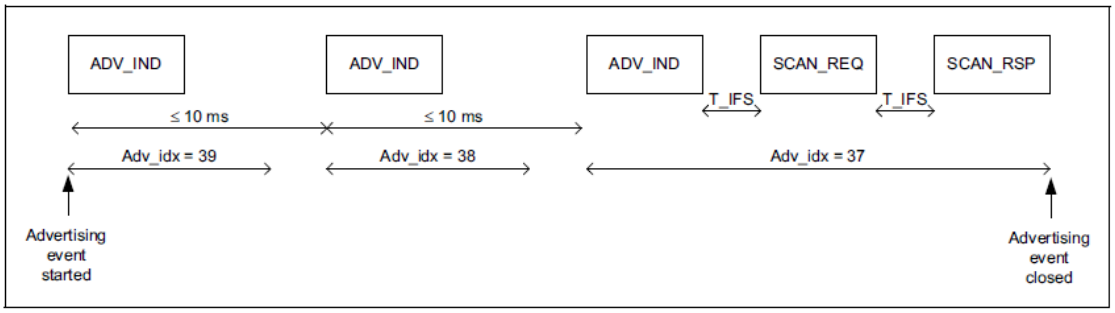
As discussed in the previous section, Bluetooth beacons are the perfect solution for signalling the presence of a device or anything the device is attached to. However the 31 B payload length might be too short if you want to tell more about the product (for example, if you want to share descriptive information about a product/artwork to which the Bluetooth tag is attached).

If you need just a bit more payload in your advertisement, there is an option to use active scanning and scan reports to extend your advertisement payload with another 31 B.

If you want to broadcast much more data, Bluetooth 5 offers the possibility to use advertisement extensions with payload size up to 1650 B.

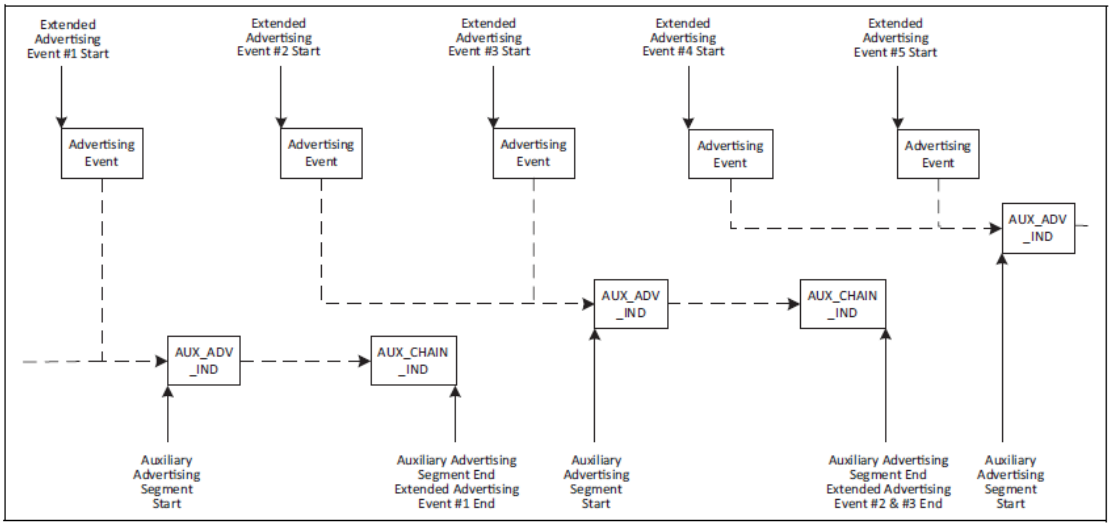
Bluetooth features to be used: legacy scannable advertisements, extended advertisements.

The length of **legacy advertisements** is limited to 31 B. However, if the advertisement type is set to *connectable-scannable* or *scannable-non-connectable*, then the device will listen for scan requests after each advertisement. Once a scan request is received (from an *active* scanner), the advertiser sends a scan report, which extends the original advertisement with an additional 31 B. This might be a good solution if you want to advertise just a bit more data than 31 B. A big advantage of this solution (from the perspective of energy consumption) is that the second half of the advertisement is only sent when requested. The disadvantage is obviously the limited payload length. The usage is similar to beacons except that the mode is different and you must also define a scan response packet when you set the data for the advertisement. Data can be set with `sl_bt_advertiser_set_data()` both for advertising and scan response packets.



Extended advertisements (or advertisement extension) is a new concept introduced with Bluetooth 5. Bluetooth 5 defines advertisement extension packets that can be sent on the data channels (channel 0 to 36) to offload the advertising channels (channel 37, 38, 39). In this context the data channels are also called secondary advertising channels and the classic advertising channels are called primary advertising channels.

An extended advertisement always starts with a packet sent on the primary advertising channels, but this packet is only used to point to a specific time instance and specific secondary advertising channel where the actual advertising data will be sent. Since the traffic on the secondary advertising channels can be spread well, there is no 31 B limit on the packet length. Instead, a packet with 255 B payload can be sent, which can be further extended with additional packets, chained after the first one. The maximum length of data that can be sent on an extended advertisement chain is 1650 B. This allows you to broadcast a large amount of data. If it is still not enough, you can start multiple advertising sets in parallel, or you can periodically change the data in the advertisement.



Bluetooth API to be used:

- `sl_bt_advertiser_create_set()`
- `sl_bt_advertiser_set_timing()`
- `sl_bt_advertiser_set_phy()`
- `sl_bt_advertiser_set_channel_map()`
- `sl_bt_advertiser_set_data()`
- `sl_bt_advertiser_start()`
- `sl_bt_system_set_tx_power()`
- `sl_bt_advertiser_set_report_scan_request()`
- `sl_bt_advertiser_set_configuration()`
- `sl_bt_advertiser_set_long_data()`

Tips for low power consumption:

- Consider the recommendations that are described in section 2 Use Case #1: I want to signal the presence of a device.
- Consider enabling scan request reporting with `sl_bt_advertiser_set_report_scan_request()`. In this case all scan requests sent by active scanners will be reported to the application (via the *sl_bt_evt_advertiser_scan_request* event), and this can be used to trigger actions. For example a good practice is using a long advertising interval by default and change it to a shorter one once a scan request is detected. This ensures low power consumption while no active scanners are present and a low latency while scanners are present. You may also start an extended advertisement with more data, once a scan request is received.
- On secondary advertising channels you can use different PHYs with different data rates. If your scanner supports 2M PHY, it is recommended to use 2M PHY on the secondary channels. 2M PHY uses a 2x higher data rate than the default 1M PHY. This means almost 2x less power consumption because of 2x shorter packets. Set the PHY with `sl_bt_advertiser_set_phy()`.

  Note: you can also use Coded PHY, which will result in higher power consumption, but ensures a much longer range.

4 Use Case #3: I want to broadcast / send data to low power receivers

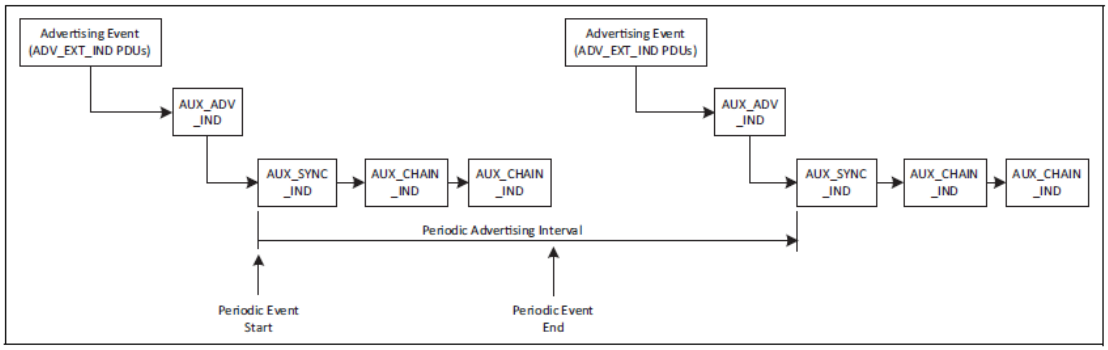
When you broadcast data with legacy or extended advertisements, as described in section 3 Use Case #2: I want to broadcast data (such as information about a product/artwork), the transmitter might be low power, but the receivers must be scanning all the time to capture those advertisements, which consumes a lot of energy. In some use cases, however, receivers must be low power, for example in the case of electronic shelf labels that are running from coin cell batteries and are receiving updates from a central device. In this use case periodic advertisements provide a solution. Periodic advertisements can broadcast data to many devices while ensuring low power consumptions on both sides.

Bluetooth feature to be used: periodic advertisements.

Periodic advertisements take extended advertisements to a higher level. With extended advertisement you always have to scan the primary advertising channels to find an advertisement. Periodic advertisements make it possible to sync on the advertisement once, and never go back to the primary channels. This is achieved by predictable timing of the advertisements sent on the secondary channels. While on the primary channels advertisement events are intentionally timed inaccurately (to avoid constant overlapping between two advertisements), secondary channels can use accurate timing. Therefore, once you find a periodic advertisement, and you know the advertising interval as well as the channel hopping sequence, you can follow the advertisement without going back to the primary channels. All the information that you need for syncing on a periodic advertisement (such as the periodic advertising interval) is included in an extended advertisement.

A huge benefit of periodic advertising is that the scanner device can go to sleep between two advertising events, because the time of the next event is known. Of course this is true only after the first periodic advertising event was found. Finding the first event still needs continuous scanning on the primary channels.

The maximum length of data that can be sent in a periodic advertisement is 1650 B, just as for extended advertisements. To broadcast more data, either start more periodic advertisements simultaneously, or change the data periodically.



Bluetooth API to be used:

- `sl_bt_system_set_tx_power()`
- `sl_bt_advertiser_create_set()`
- `sl_bt_advertiser_set_timing()`
- `sl_bt_advertiser_set_phy()`
- `sl_bt_advertiser_set_data()`
- `sl_bt_advertiser_start()`
- `sl_bt_advertiser_set_long_data()`
- `sl_bt_advertiser_start_periodic_advertising()`
- `sl_bt_sync_set_parameters()`
- `sl_bt_sync_open()`

Tips for low power consumption:

- Consider the recommendations that are described in section 2 Use Case #1: I want to signal the presence of a device and section 3 Use Case #2: I want to broadcast data (such as information about a product/artwork).
- It is important to know that the extended advertisements that carry the information about the periodic advertising do not have to have the same advertising interval that the periodic advertising has. In other words, sync info (sent in the extended advertisement) can be broadcasted more frequently than the periodic advertising interval, which makes it possible for the receiver to scan for a shorter time

on the primary advertising channels, thereby saving some energy on the receiver side. Also, the sync info can be broadcasted less frequently than the periodic advertising interval, which saves energy on the transmitter side.

- The less accurate your Low Frequency Oscillator, the earlier the receiver device must wake up from sleep mode to ensure it receives the packets. Using a 32 kHz crystal oscillator (instead of the RC oscillator) is strongly suggested to save energy.

5 Use Case #4: I want to turn lamps on and off

Bluetooth Low Energy is very often used to control devices around you, such as your lamp, LED strip, or garage door. Usually, this is achieved by creating a connection to the controlled device and sending different commands over this connection (discussed in section 6 Use Case #5: "I want to occasionally connect to my device to exchange some data"). However, in many cases the switch is running on a battery and requires very low power consumption. In this case, a good approach may be to use simple advertisements to send the commands instead of the complicated connection process. This not only gives the lowest possible power consumption, but also provides the lowest possible latency.

Bluetooth features to be used: legacy advertisements, extended advertisements.

If you have a controlled device (for example a lamp) that has a continuous power supply and a switch that runs on a battery, the simplest and probably best solution is to put the controlled device into continuous scanning mode, and send a single advertisement packet from the switch whenever an action is needed. Since the controlled device is scanning continuously, it will receive the command immediately. Also, the switch does not have to spend energy on creating a connection, since it sends a single packet only – maybe repeating it a few times for redundancy. This is the same approach that is used by wireless rings or simple garage door openers. The only difference is that usually such solutions apply a proprietary protocol, and not a standard one like Bluetooth.

Obviously, this solution has drawbacks compared to Bluetooth connections:

- The devices cannot discover each other and pair.
- While the BT address of the switch is present in the advertisement, the address of the controlled device is not specified.
- Anyone can listen to the activities and anyone can control the controlled device.

However, these problems can be easily overcome.

- While the controlled device is scanning and the switch is waiting for a trigger during normal duty, both devices can be put into connectable advertising mode, and a connection created between them (or between them and a smartphone). This connection must be created only once. During this connection the switch learns the BT address of the controlled device, and they can also share a common secret which can be used later to secure the communication between them.
- Once the BT address of the controlled device is known, it can simply be added to the advertisement data.
- Once a common secret is shared, the commands provided in the advertisements can be encrypted and signed. Adding a counter to the command also protects againts replay attacks. Applying rolling codes is also a good solution.

Since 31 B is usually not enough to send encrypted custom commands, it is recommeded to use extended advertisements, described in section 3 Use Case #2: I want to broadcast data (such as information about a product/artwork).

Use the following advertising data types in the advertising data (https://btprodspecificationrefs.blob.core.windows.net/assigned-numbers/Assigned%20Number%20Types/Generic%20Access%20Profile.pdf):

- Public Target Address (0x17)
- Service Data - 128-bit UUID (0x21) – define a random 128-bit UUID for your application, and add your custom data
- Manufacturer Specific Data (0xFF) – if your company has a manufacturer ID registered at Bluetooth SIG

If not only the switch but also the controlled device requires low power consumption, there are three solutions:

- Use periodic advertisements as described in section 4 Use Case #3: I want to broadcast / send data to low power receivers.
- Let the controlled device advertise and create a connection as described in section 6 Use Case #5: "I want to occasionally connect to my device to exchange some data".
- Use the same approach as described in this section, but set the scan window to a short interval while setting the scan interval to a long interval on the scanner device. On the switch, repeat the advertisement many times. For example, set the scan window to 30 ms, scan interval to 1 sec, set the advertisement interval to 20 ms and repeat the advertisements for 1.2 sec. This ensures that at least one advertisement will be received by the controlled device, while it is scanning with a 3% duty cycle.

Bluetooth API to be used:

- `sl_bt_advertiser_create_set()`
- `sl_bt_advertiser_set_timing()`
- `sl_bt_advertiser_set_phy()`
- `sl_bt_advertiser_set_channel_map()`
- `sl_bt_advertiser_set_data()`
- `sl_bt_advertiser_start()`

- `sl_bt_system_set_tx_power()`
- `sl_bt_advertiser_set_configuration()`
- `sl_bt_scanner_set_timing()`
- `sl_bt_scanner_start()`

Tips for low power consumption:

- Consider the same recommendations that are described in section 2 Use Case #1: I want to signal the presence of a device, section 3 Use Case #2: I want to broadcast data (such as information about a product/artwork), and section 4 Use Case #3: I want to broadcast / send data to low power receivers.

6 Use Case #5: I want to connect to a device occasionally to exchange some data

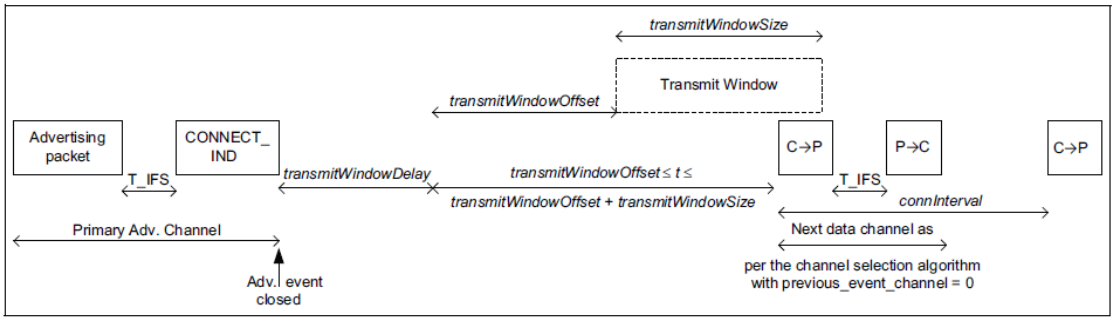
Bluetooth Low Energy is a perfect solution if you want to query some data from a low energy device (such as a sensor), and process/display that data on a more powerful device. Similarly, it can be used to set configurations on low energy devices, or issue simple commands to the low energy devices. This is extremely useful if you want to spare the user interface on the low energy device.

If you want to issue simple commands, consider the approach described in section 5 Use Case #4: I want to turn lamps on and off. In all other cases you will need to create a Bluetooth connection with your low energy device to exchange data.

Bluetooth feature to be used: connections (also needs advertising and scanning).

Bluetooth connections are used to create a reliable two-way data link between two devices. The connection process is simple:

- The peripheral device is advertising and listening for connection requests for a limited time after each advertisement.
- The central device sends a connection request (right after an advertisement was received) and defines the connection interval.
- The central device sends at least one packet in each connection interval and the peripheral device responds with at least one packet in every Nth connection interval (where N-1 is the peripheral latency). If there is no data to send, an empty packet must be sent to keep the connection alive. Both devices can go into sleep mode between two connection events.



Bluetooth connections are reliable (packet loss is detected, and re-transmission is applied) and can also be secured with encryption and authentication. They are also low power because the devices must wake up only for a short time in every connection interval. Nevertheless, there are many settings that should be considered to make your application really low power.

This section focuses on connections that are meant to be kept alive for a short time.

Bluetooth API to be used:

- All the advertiser and scanner APIs
- `sl_bt_system_set_tx_power()`
- `sl_bt_system_linklayer_configure()`
- `sl_bt_gatt_set_max_mtu()`
- `sl_bt_gatt_server_set_max_mtu()`
- `sl_bt_connection_set_default_parameters()`
- `sl_bt_connection_set_default_preferred_phy()`
- `sl_bt_connection_open()`

Tips for low power consumption:

- Find the best advertising interval for your application. Too short an advertising interval mean high power consumption while too long an advertising interval mean high latency. Set the advertising interval with `sl_bt_advertiser_set_timing()`.
- Scanners usually parse the advertisements first and send the connection request after a subsequent advertisement. Therefore, it may be a good idea to send advertisements in bursts. For example, wait for 3 seconds, then send 10 advertisements with a 20-ms difference. Here you will need a timer (SleepTimer is recommended) to schedule the bursts, while the duration of the bursts can be set with `sl_bt_advertiser_set_timing()`.
- It may be also good to temporarily shorten the advertising interval when a scan request is detected (see section 3 Use Case #2: I want to broadcast data (such as information about a product/artwork)). Listen for the *sl_bt_evt_advertiser_scan_request* event, stop advertising, adjust the timing, and restart advertising.

- The main parameters of a Bluetooth connection are connection interval, peripheral latency, and timeout. These parameters can be set with `sl_bt_connection_set_default_parameters()`.
  - If your connection is short (that is, it does not take more than a few connection events), choose a short connection interval and no peripheral latency to ensure lower latency.
  - If your connection is a bit longer (several seconds, for example because of manual triggers), then probably there will be periods where only empty packets are sent. Consider using a somewhat longer connection interval here, because too many empty packets will result in an unnecessary power consumption increase. Also apply peripheral latency so that the peripheral does not have to respond to all packets from the central device. See details in section 7 Use Case #6: "I want to maintain a long-term connection with occasional data exchange".
- Although the connection process is simple and fast, usually every connection starts with some parameter exchange. The peer devices usually exchange:
  - Supported features
  - Connection parameters
  - MTU size

  These parameter exchanges take a while, and increase both latency and energy consumption. To save the most power you can disable these procedures:
  - Feature exchange can be disabled with `sl_bt_system_linklayer_configure()`, see the API reference,
  - Connection parameter exchange can be disabled by setting the same default connection parameters on both devices,
  - MTU size exchange can be disabled by setting the MTU size to the default 23 B on both devices. Use `sl_bt_gatt_set_max_mtu()` and `sl_bt_gatt_server_set_max_mtu()`.

  Note: Disabling these procedures will result in limited functionality, but this is usually not an issue if you want to exchange some data quickly.
- After parameter exchange, the central device usually discovers the services in the GATT database of the peripheral device. Again, This causes higher latency and more energy consumption. To avoid this, either
  - Use hard-coded attribute handles when reading/writing characteristics on the remote device, or
  - Discover the database once, save the discovered attribute handles, and check the database hash on the subsequent connections to see if the attribute handles changed since the last connection (see https://docs.silabs.com/bluetooth/3.2/general/gatt-protocol/gatt-caching for more details).
- Bluetooth 5 supports multiple data rates via 1M, 2M and Coded PHY. Consider changing the preferred PHY from the default 1M to 2M PHY with `sl_bt_connection_set_default_preferred_phy()`. The increased datarate will result in shorter packet transmission times, which means lower power consumption.
- Set the TX power according to the expected distance between the devices with `sl_bt_system_set_tx_power()`. If the devices are expected to be close to each other, TX power can be decreased to save energy. You may also set the TX power according to the received RSSI level.
  - On the scanner side (central), check the RSSI level of the advertisements (see *sl_bt_evt_scanner_scan_report* event).
  - On the advertiser side (peripheral), check the RSSI level of the scan requests (see *sl_bt_evt_advertiser_scan_request* event).
- Consider applying application-level security. Although Bluetooth security is able to protect your connections very well against malicious attackers, it needs some data exchange on the Bluetooth connection. In some cases it might be better to send encrypted / signed content over a non-secured connection, and do the decryption / signature check in the application, just to save some energy by not sending encryption requests and responses.
- The GATT layer supports both acknowledged (read, write) and un-acknowledged (notify, write_without_response) data transfer. The latter one needs less data transmission and hence less energy. Note: Although un-acknowledged data tranfser means there is no acknowledgement from the other side, the link layer still ensures reliable data transfer by detecting packet loss and re-transmitting lost packets.

7 Use Case #6: I want to maintain a long-term connection with occasional data exchange

Whether it is better to open a short-term connection every time you want to read data from your peripheral device or to keep a long-term connection alive with occasional data exchange depends on the use case. Here are some aspects to be considered:

- Short-term connections allow you to move away from the peripheral device between two data exchanges. Long-term connections suppose that the peer devices stay close to each other.
- Short-term connections use more energy on the peripheral side because peripheral devices must advertise all the time to be connectible. Long-term connections use more energy on the central side because the central device must send a packet in every connection interval, while the peripheral latency allows the peripheral device to respond only every Nth packet from the central.
- Short-term connections have higher latency introduced by the connection process. Long-term connections have higher sum energy consumption.

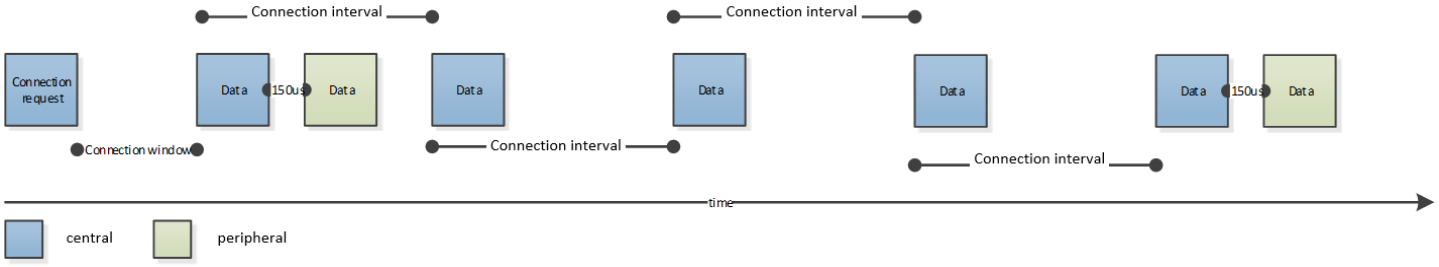
In addition to these, if you connect to your peripheral device with a mobile phone, and trigger actions (read/write) manually, it also counts as a long-term connection because manual triggers are usually slow.

This chapter focuses on how to save energy while keeping alive a long-term connection. If you decide on using short-term connections, refer to section 6 Use Case #5: “I want to occasionally connect to my device to exchange some data”.

Bluetooth feature to be used: connections (also needs advertising and scanning).

While in the case of short-term connections the focus is on the connection creation process, long-term connections need optimized power consumption while keeping the connection alive. A huge benefit of Bluetooth Low Energy is that the devices do not have to listen on a channel all the time to receive packets, which would result in high energy consumption. Instead packets are always sent and received at the beginning of connection intervals. The longer the connection interval, the lower the consumption – but it also results in a higher latency. The connection interval, therefore, must be set as a trade-off between power consumption and latency.

One way to further decrease power consumption is using peripheral latency. Setting peripheral latency to N means that the peripheral device can skip N connection intervals and send a response packet to the central device in every (N+1)th connection interval. Note that this is just an option. The peripheral can always decide to respond a packet from the central earlier. This ensures both low latency and low energy consumption on the peripheral side.



Bluetooth API to be used:

- All the advertiser and scanner APIs
- `sl_bt_system_set_tx_power()`
- `sl_bt_connection_set_default_parameters()`
- `sl_bt_connection_set_default_preferred_phy()`
- `sl_bt_connection_open()`
- `sl_bt_connection_set_parameters()`
- `sl_bt_connection_set_preferred_phy()`
- `sl_bt_connection_disable_slave_latency()`
- `sl_bt_connection_set_power_reporting()`
- `sl_bt_connection_set_remote_power_reporting()`

Tips for low power consumption:

- Too short an advertising interval mean high power consumption while too long an advertising interval mean high latency. Set the advertising interval with `sl_bt_advertiser_set_timing()`.

- Use peripheral latency. If the data exchanged is triggered by the peripheral, then the peripheral latency can be set to a high number, because it saves energy and the peripheral is still able to send data in any connection interval. If the data exchange is triggered by the central, then you should be more careful, because the central cannot reach the peripheral in any connection interval. Set the peripheral latency with `sl_bt_connection_set_default_parameters()`.
- Decrease the connection interval during data exchange and increase it while no data exchange is expected. There are two methods to achieve this:
  - If you know in advance (6 connection intervals earlier) when the data exchange is due, you can update the connection parameters (and hence the connection interval). As an example, the connection interval can be set to one second in 'stand-by' state, while it can be temporarily decreased to 20 ms during data exchange. Note that the parameter update process takes 6 connection intervals, therefore an instant switch cannot be achieved. Connection parameters can be updated with `sl_bt_connection_set_parameters()`. Note that the peer devices must agree on the parameters. If there is no common range for the parameters, the central device decides.
  - Another solution is to set the connection interval to be short, and the peripheral latency to be high. For example, a 20 ms connection interval and a peripheral latency of 49 connection events means that the peripheral needs to wake up once a second. In this case the peripheral can apply the peripheral latency in the 'stand-by' state, while it can temporarily disable peripheral latency during data exchange with `sl_bt_connection_disable_slave_latency()`. This method means a high power consumption on the central side, but also ensures an instant switch in the transmission rate when needed.
- Bluetooth 5 supports multiple data rates via 1M, 2M and Coded PHY. Consider changing the preferred PHY from the default 1M to 2M PHY with `sl_bt_connection_set_default_preferred_phy()`. The increased data rate will result in shorter packet transmission time, which means lower power consumption.
- Set the TX power according to the expected distance between the devices with `sl_bt_system_set_tx_power()`. If the devices are expected to be close to each other, TX power can be descreased to save energy.
- The Silicon Labs Bluetooth stack also supports LE Power control which means that the TX power is adjusted automatically (by the stack) so that the received signal strength falls into a given range. If the positions of your devices are not fixed, it is worth enabling this feature by installing the *PowerControl* software component. Note: This feature must be supported by both devices. You should also set the minimum and maximum enabled TX power with `sl_bt_system_set_tx_power()` and configure the so-called Golden Range for the RSSI in the configuration of the *PowerControl* software component. The Golden Range is the range to keep the RSSI in in on the receiver side. If the RSSI goes below this range, the receiver instructs the transmitter to increase its TX power (if possible). If the RSSI goes above this range, the receiver instructs the transmitter to decrease its TX power (if possible).
- The GATT layer supports both acknowledged (read, write) and un-acknowledged (notify, write_without_response) data transfer. The latter needs less data transmission and hence uses less energy. Note: Although un-acknowledged data tranfser means there is no acknowledgement from the other side, the link layer still ensures reliable data transfer by detecting packet loss and re-transmitting lost packets.

8 Use Case #7: I want to send a lot of data on a long-term connection

Sending a lot of data on a Bluetooth connection is not a low energy use case by its nature. However, there are a few things to consider to save energy:

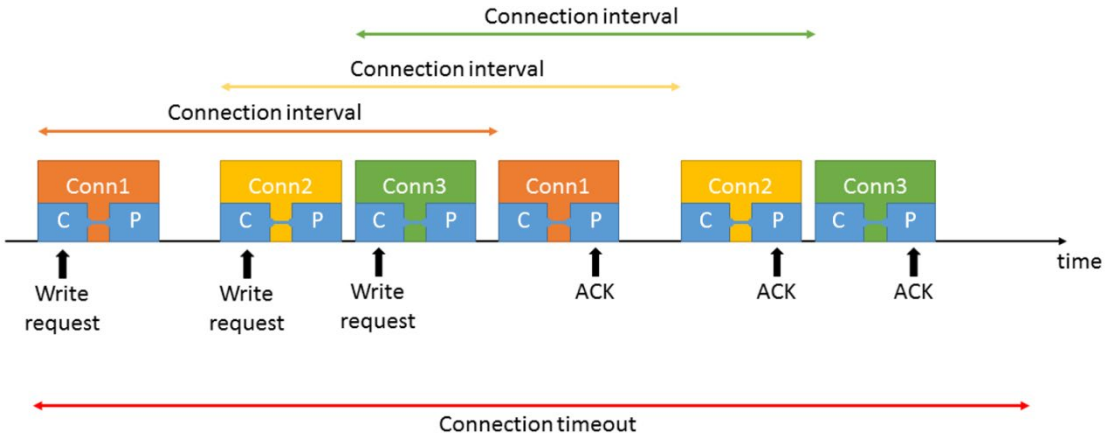
- First, using a 2 Mbps data rate with 2M PHY is not only good because it ensures a higher transmission speed, but it also decreases power consumption because of the shorter transmission time. Therefore it is strongly suggested to apply 2M PHY with `sl_bt_connection_set_default_preferred_phy()`, unless you need long-range communication.
- Another important setting is the TX power. To use optimal TX power take advantage of the LE Power Control feature of the Bluetooth stack, which automatically adjusts the TX power based on the RSSI level reported by the receiver. LE Power Control is described in section 7 Use Case #6: I want to maintain a long-term connection with occasional data exchange.
- In a busy environment, it may be worth enabling Adaptive Frequency Hopping (AFH). AFH scans through the 2.4 GHz frequency band, and disables Bluetooth channels where interference (for example with Wi-Fi) is detected. Although the scanning means additional energy consumption, avoiding packet retransmission caused by interference may save more than the added energy. For details about AFH see https://docs.silabs.com/bluetooth/latest/general/system-and-performance/adaptive-frequency-hopping.

9 Use Case #8: I want to exchange data with many peripheral devices

Bluetooth Low Energy allows a device to connect to many other devices at the same time. One central device can connect to multiple peripheral devices, and one peripheral device can also connect to many central devices. Some devices may even work as a central and a peripheral at the same time. Therefore exchanging data with multiple devices is not an issue, at least not up to a certain number of devices. Keeping many parallel connections alive will suffer from many collisions and requires heavy scheduling tasks from the stack. Silicon Labs' Bluetooth stack supports up to 32 simultaneous connections. If more devices are to be served, the active connections can be rotated among the devices or other solutions should be found like communicating using advertisements. This section gives recommendations for such use cases.

Bluetooth feature to be used: connections, legacy advertising, extended advertising, periodic advertising.

Parallel Bluetooth connections are achieved by interleaving connection events. Since the devices are passive between two connection events, it is easy to schedule them without overlapping. To make this possible, however, you should set the connection parameters (connection interval, supervision timeout) accordingly. For example, a 7.5 ms connection interval should not be used if there are 32 active connections. Also, the connections should preferably use the same interval, or the intervals should be the multiples of a common base value.



Since the central device decides when to start the first connection event, centrals can manage parallel connections better than peripherals.

If you need to communicate with more than 32 devices or you just want to save power, consider using periodic advertising. Any number of devices can sync on a periodic advertisement, and it also means low power consumption (see section 4 Use Case #3: I want to broadcast / send data to low power receivers). Periodic advertising, however, also means a one-way communication, so if you want to report back data, you can either send it in an advertisement, or you can create a temporary connection.

Remember that, although periodic advertising is broadcasting data, you can still add Bluetooth addresses to the data to mark the intended recipients. While it is not a secure way of communication, it can be very efficient – and it is still possible to encrypt the data with a shared secret that only the recipient knows.

Bluetooth API to be used:

- Check previous sections based on the chosen solution.

Tips for low power consumption:

- When using multiple parallel connections, make sure there are no overlapping connection events. Overlapping results in retransmission and therefore higher consumption.
  - Choose the connection parameters wisely as described above, and
  - install the *Even Connection Scheduling Algorithm* software component in your projects.
- Sending data on a periodic advertisement might be a good solution, but remember that when you send data to one (or some) recipients, all the synced devices must be receiving and hence consume energy. To solve this issue, either:
  - Send only a short command in the advertisement, for example a command that instructs one recipient to start connectable advertising, and then create a connection to send more data.
  - Create multiple periodic advertisements, for example one for devices with odd Bluetooth addresses and one for devices with even Bluetooth addresses. This will halve the devices that must receive data sent to a given recipient.
  - Create one periodic advertisement for commands and one for data. In this case all devices can listen for commands and only active recipients can sync on the data-carrier advertisement.

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