AN1252: Building Low Power Networks with the Silicon Labs Connect Stack v3.x

This version of AN1252 has been deprecated with the release of Simplicity SDK Suite 2025.6.1.

For the latest version, see docs.silabs.com.

****

This document illustrates techniques for reducing power consumption in a sensor network developed using Simplicity Studio and the Silicon Labs Connect Stack v3.x, distributed as part of the Silicon Labs Proprietary Flex Software Development Kit (SDK) v3.x with Simplicity Studio® 5. Techniques include creating sleepy end devices that, by their nature, consume less energy, reducing reporting frequency, and removing unnecessary peripheral energy use. The techniques are illustrated with the sensor and sink example applications provided with the Flex SDK. For more in-depth background on energy savings in Connect networks see *UG435.07: Energy Saving with Silicon Labs Connect v3.x*, part of the *Connect v3.x User's Guide* series.

Proprietary is supported on all EFR32FG devices. For others, check the device's data sheet under Ordering Information > Protocol Stack to see if Proprietary is supported. In Proprietary SDK version 2.7.n, Connect is not supported on EFR32xG22.

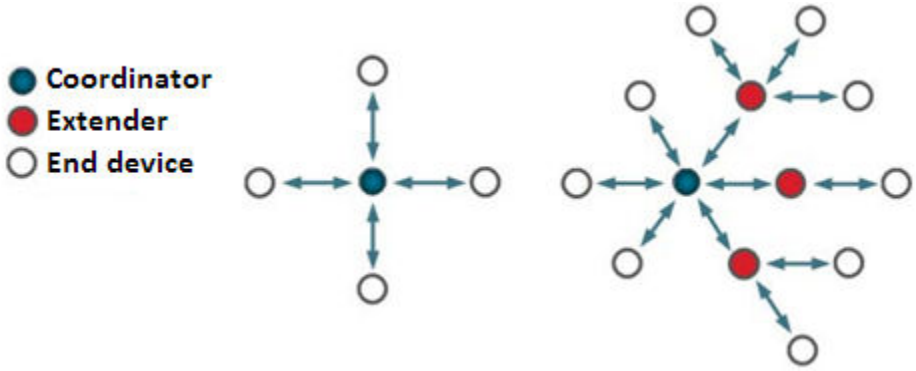
KEY POINTS

- Building example sensor and sink applications.
- Creating a sensor-sink network.
- Monitoring baseline average current consumption.
- Reducing energy use in the network.

1. Introducing Low Power Sensor Networks

A typical low power network includes a single “always-on” device that serves as the coordinator for the network and one or more end nodes operating as sleepy end devices. Such devices spend most of their time in deep sleep mode, waking only briefly to transmit data to the central coordinator. An example of a low power network is a simple sensor network where the sensors are all sleepy end devices and the central hub serves as the always-on coordinator, or sink, for the network.

The Silicon Labs Flex SDK includes example sensor and sink applications, based on the Connect stack, for exactly this scenario. Supported network structures are star and extended star topologies:



The example applications are:

- Connect (SoC): Sensor
- Connect (SoC): Sink

Together the applications demonstrate a star topology. Bi-directional communication is possible between the sensor(s) and the sink nodes. This document reviews using Simplicity Studio to build a sensor-sink network, and then to modify it for the lowest power consumption.

If you are not familiar with building the example applications in Simplicity Studio, please review *QSG168: Proprietary Flex SDK v3.x Quick Start Guide*. If you are not familiar with the features and functions of the Silicon Labs Connect stack, see *UG103.12: Silicon Labs Connect Fundamentals*.

2. Create the Baseline Network

This chapter describes how to:

1. Build and run the example coordinator/sink application.
2. Build and run the sensor application.
3. Create the network.
4. Evaluate baseline energy consumption.

You will need at least two devices connected to a computer running Simplicity Studio.

2.1 Build and Run the Coordinator/Sink Application

1. In the Simplicity Studio Launcher perspective, select the device to be used as the coordinator/sink.
2. Start a new project by clicking [**Create New Project**].
3. Create the project by selecting the **Connect: (SoC) Sink** example and completing the steps in the New Project wizard.
4. Compile (hammer icon) and debug (bug icon) the project. The coordinator application is now installed on the device.

To configure the coordinator, terminal software is necessary to set up the parameters. Use a separate application or the Simplicity Studio's built-in console tool. Issue `help` in the terminal to see some basic help from the firmware. When the firmware starts, issue `form 0` (form a network on channel 0) and `pjoin 255` (enable join of nodes for unlimited time) commands:

At this point, the coordinator is up and running, and ready to accept incoming joining requests, so sensor nodes can connect to the sink.

2.2 Build and Run the Sensor Application

1. Return to the Simplicity Studio Launcher perspective and select the device to be used as the sensor.
2. Start a new project by clicking [**Create New Project**].
3. Create the project by selecting the **Connect: SoC Sensor** example, and completing the steps in the New Project wizard. All the settings can be kept as default. Be sure the settings of the radio configuration of the sink and the sensor match.
4. Compile and debug/run the application.

2.3 Create the Network

The two most important terminal commands for the sensor are `join` and `join-sleepy`. On issuing `join`, the device attempts to connect to the coordinator and, if the join process is successful, the node becomes part of the network. The device will not try to enter to sleep or idle mode at all, keeping the device awake with relatively high current consumption.

The `join-sleepy` command enables the device to enter the sleep state. After a successful join process, the device starts sending the sensor data to the sink periodically (with a 1s rate by default).

At this point, a network consisting of a coordinator/sink and one node/sensor should work, the sensor data should be sent by the sensor and received by the sink with 1 s periodicity. More nodes can be added to the network by downloading the sensor firmware into additional devices. The status code of a successful join is 0x90: Network up, and the firmware prints Network up on the console.

```
Sensor
> join_sleepy 0

join sleepy 0x0
> Network up
TX: Advertise Request (unicast), status 0x0
RX: Advertise from 0x 0
TX: Pair Request to 0x 0: 0x0
RX: Pair Confirm from 0x 0
TX: Data to 0x 0: 41 7d 0 0 5c 93 0 0: 0x0
TX: Data to 0x 0: 41 7d 0 0 63 93 0 0: 0x0
TX: Data to 0x 0: 36 7d 0 0 6b 93 0 0: 0x0
```

The firmware prints numerical status codes on the console, instead of a textual description, if the sensor fails to join the network.

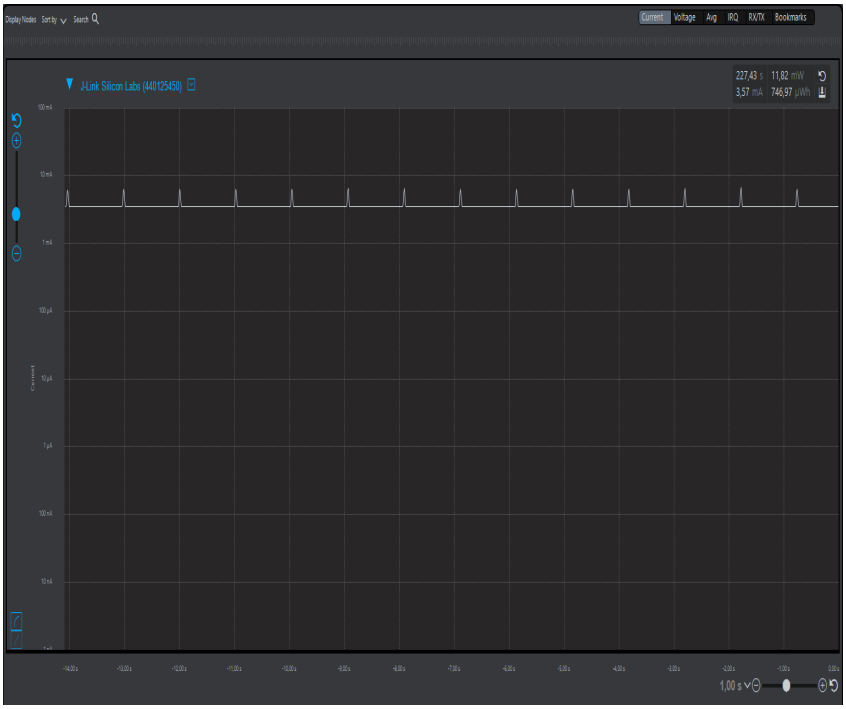
- `0x40`: No ACK received, the node to which a message was transmitted did not respond.
- `0x41`: Indirect MAC timeout, may occur if the sender did not get a poll request to send its pending message.
- `0x70`: Invalid call, usually when an unexpected operation is carried out (for example, joining when joined or sending when not joined, and so on).
- `0xAB`: No valid beacons, the other radio cannot be heard (for example, no antenna attached).

Before each step in the experiment with the sensor/sink examples it is a good practice to issue the `leave` command on both the coordinator/sink and sensors. For example, the coordinator maintains information about the previously connected nodes, based on their EUI64 ID. The nodes also store information and try to reconnect to the coordinator automatically at power-up based on this information. By issuing the `leave` command this information will be cleared.

2.4 Monitor Baseline Energy Use

The Energy Profiler is a useful tool to inspect current consumption real-time. For details on using the Energy Profiler see *UG343: Multi-Node Energy Profiler User's Guide*.

The following graph shows the consumption of a non-optimized sensor node.



3. Reduce Power Consumption

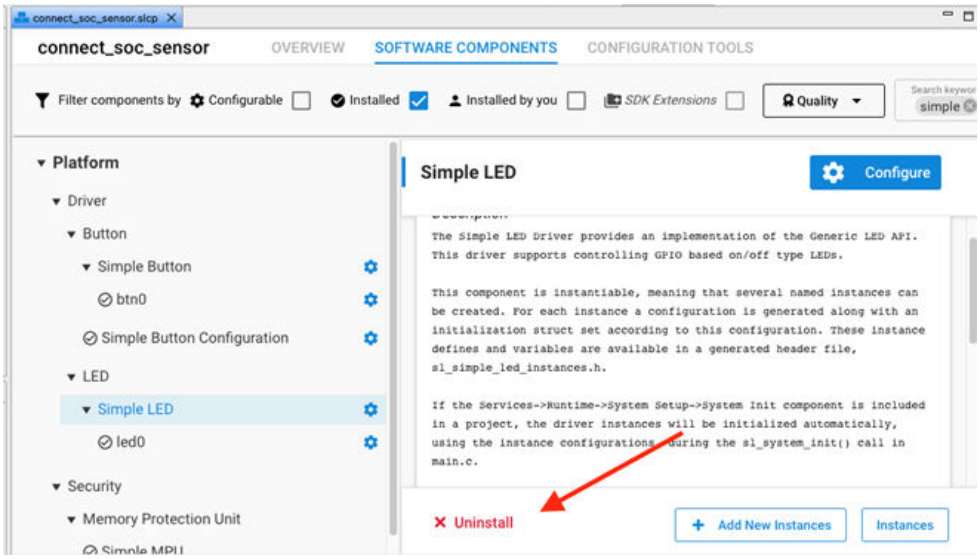
This chapter describes techniques for reducing power consumption, including:

- Disable Network LED
- Disable UART reception to enable sleep mode
- Change Report Periodicity
- Initialize the SPI Flash Chip

Finally, the reduced power consumption results are described.

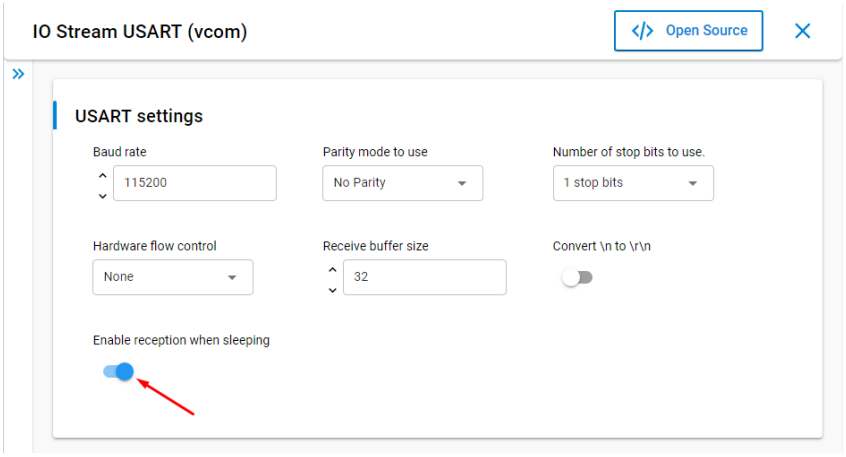
3.1 Disable Network LED

The sensor example sets an LED when the device is joined to a network, which draws a constant current, depending on the hardware (e.g., about 500uA on BRD4001). The easiest way to disable it is to remove the simple LED component.



3.2 Disable UART Reception during Sleep

Next, the step that reduces the consumption the most is disabling UART reception during sleep. This enables entering EM2 mode when it is possible. This option can be enabled in the Component Editor under **IO Stream USART (vcom)** settings.



Note: If reception is disabled during sleep, issuing commands through the CLI (UART) is not possible. The sensor example has a feature where PB0 toggles between enabled and disabled reception mode.

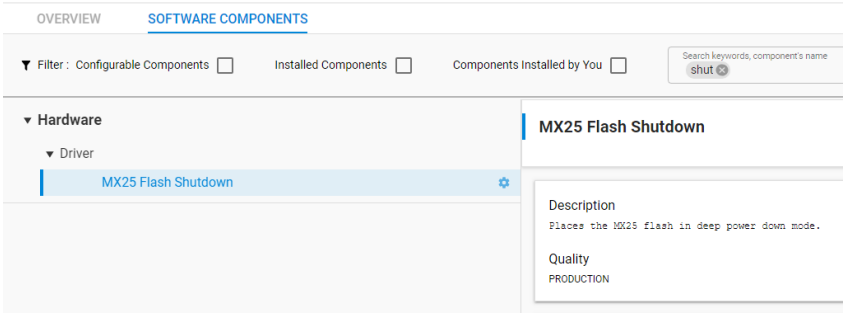
3.3 Change Report Periodicity

Changing the report period can significantly reduce the average consumption if the sleep consumption is significantly lower than consumption in active periods. To change the report period, open **app_process.c** and modify the `sensor_report_period_ms` value.

```
// -------------------------------------------------------------------------------
//                                Global Variables
// -------------------------------------------------------------------------------
/// report timing event control
EmberEventControl report_control;
/// report timing period
uint16_t sensor_report_period_ms =  (1 * MILLISECOND_TICKS_PER_SECOND);
/// TX options set up for the network
EmberMessageOptions tx_options = EMBER_OPTIONS_ACK_REQUESTED;
```

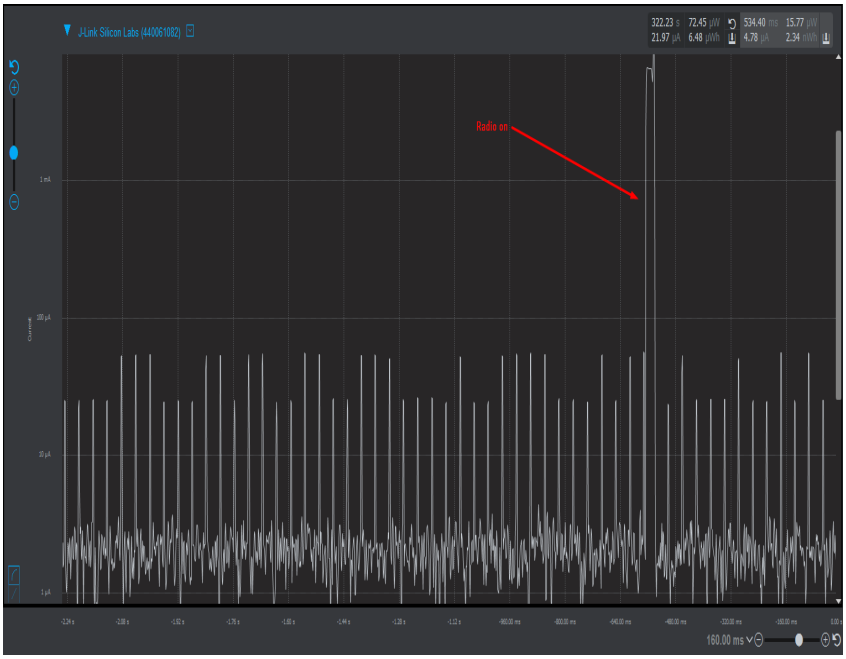
3.4 Initialize the SPI Flash Chip

This step shouldn't be necessary, as it is always implemented if you create the project for a Silicon Labs radio board. However, if your design includes an external flash, or if you decided to create the project for the given MCU part, not the Silicon Labs radio board, it might be necessary. All the radio boards designed by Silicon Labs contain an SPI flash chip that consumes ~8-10 µA if left uninitialized. To reduce the SPI Flash current consumption, the **MX25 Flash Shutdown** component must be installed. In the sensor application, it is installed by default.



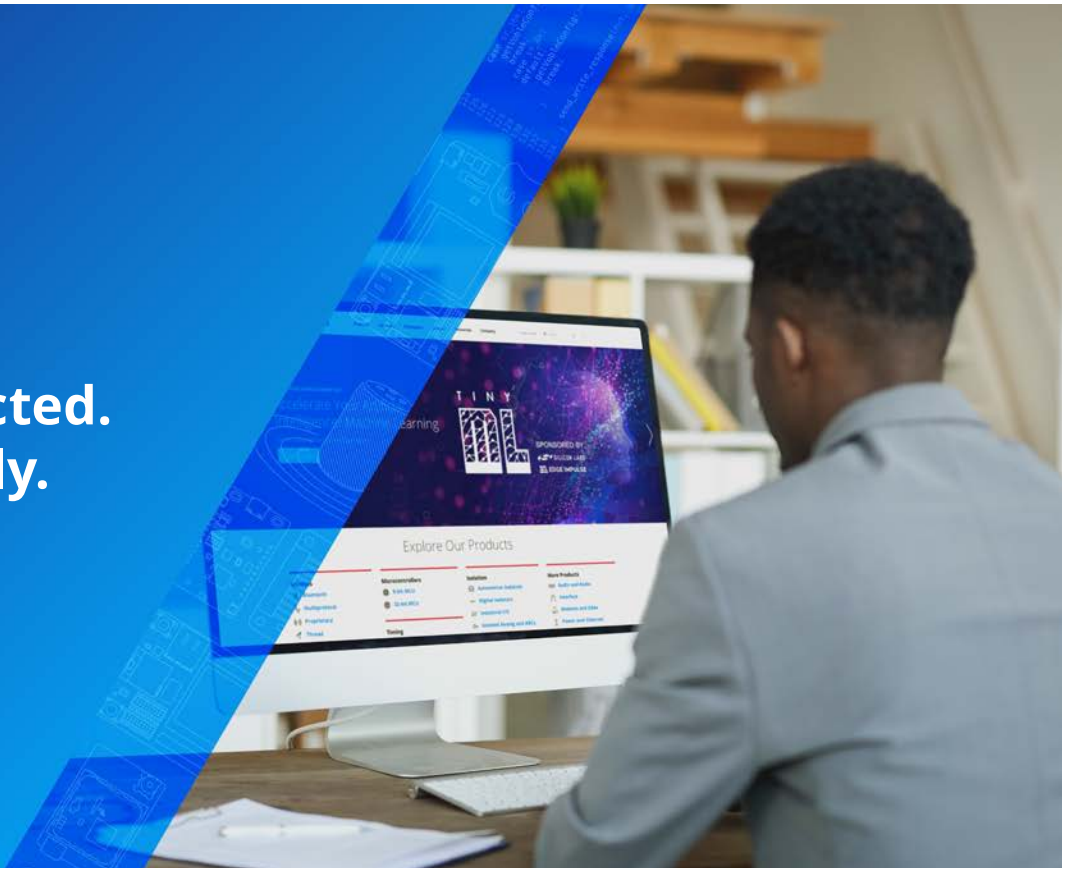
3.5 Reduced Power Consumption Results

Finally, if these modifications are applied, the sleep mode current consumption should be reduced significantly.



Note: The built-in current measurement of the WSTK board is not really accurate in the µA range and below. The sleep mode current consumption of the sensor node, once all modifications have been applied, should be in the 2-5 µA range.

Smart. Connected. Energy-Friendly.



IoT Portfolio
www.silabs.com/products

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