Application Report

SWRA502C–February 2017–Revised November 2018

CC3120, CC3220 SimpleLink™ Wi-Fi® Internet-on-a chip™ Networking Subsystem Power Management

ABSTRACT

The CC3120 and CC3220 devices are part of the SimpleLink™ microcontroller (MCU) platform, which consists of Wi-Fi®, *Bluetooth*® low energy, Sub-1 GHz, and host MCUs. All share a common, easy-to-use development environment with a single core software development kit (SDK) and rich tool set. A one-time integration of the SimpleLink™ platform enables you to add any combination of the portfolio's devices into your design. The ultimate goal of the SimpleLink™ platform is to achieve 100 percent code reuse when your design requirements change. For more information, visit www.ti.com/simplelink.

This application report describes the best practices for power management and extended battery life for embedded low-power Wi-Fi® devices such as the SimpleLink™ Wi-Fi® Internet-on-a chip™ solution from Texas Instruments™.

Contents

List of Tables

Trademarks

1 Introduction

Power management and extended battery life are primary focus areas for embedded low-power Wi-Fi® devices such as the SimpleLink™ Wi-Fi® Internet-on-a chip™ solution. Handling power regimes effectively is fundamental for any battery-operated device. This problem is especially challenging for standards-based wireless devices that need to comply with certain requirements for Transmit power, Beacon interval, and data rates, as is the case with embedded Wi-Fi devices. This application note covers the CC3120 and CC3220 Wi-Fi® internet-on-a chip™ Networking subsystem power management (PM) capabilities, describes the basics of system behavior, and provides the basic toolbox for developers to design an optimal system.

The CC3120 device contains the Networking subsystem only and is driven by an external MCU host. The CC3220 contains the same Networking subsystem along with an internal MCU application processor. This document describes the Wi-Fi® subsystem power management aspects, making it applicable for both devices.

1.1 Terminology

This document refers to several power related measurement units:

- Current [ampere] – Refers to average current at 3.3 V, unless otherwise specified.
- Charge [coulomb] – [1 C = 1 ampere × 1 second], refers to 3.3 V unless otherwise specified. Charge is used when discussing finite processes (such as connection). Battery capacity is referred as mA × hour.
- Energy [joule] – [1 joule = 1 coulomb × 1 volt]

1.2 Low-Power Internet Systems Considerations

When designing a low-power networking system specifically based on 802.11 protocols, a designer should consider several key aspects.

1.2.1 Power Supply

Lined-power or battery-powered device: line-powered systems are typically less sensitive to power consumption than battery-operated systems. Battery-operated systems must consider the battery capacity, along with the average usage profile against the lifetime target. However, there are cases where power considerations apply to line-powered systems, such as when an energy budget is associated with each endpoint. This is more common in managed deployment scenarios, such as enterprise environments.

1.2.2 Traffic Properties

- Traffic Model – The payload to be transmitted or received and the frequency of this activity. For example, a sensor may typically deliver 100B every few minutes; alternatively, a video camera may stream a throughput of 4 Mbps on average, delivering 1500B every ≈ 3 ms.
- Protocol Properties – This includes the transport protocol (UDP/TCP), the application protocol running on top, and associated overheads.
- Type of Connection – The security protocols at the link layer and the transport layer, if any.
- System Latency – The expected latency time responding to incoming traffic. This parameter is dominant in determining the sleep policy of the networking device.

1.2.3 Network Topology

The following questions should be answered when approaching a low-power networking system design. These considerations apply to the specific problem the system will solve, before any specific considerations of the networking device.

- Is the system expected to work as a server or a client?

  As a server, it might be expected to be always responsive, while as a client it may initiate the communication events on its own timing, and may be off for a defined period of time.

- Does the system always need to be connected to the Access Point?

  Maintaining the connection with the Access Point (AP) enables the system to receive messages with low latency. However, maintaining the connection usually consumes a significant amount of energy.
- Does the system always need to be connected to the server?

  Maintaining a TCP or SSL connection with periodic keep alive messages consumes significant amounts of energy. The alternative of initiating a TCP/SSL connection every time implies overhead energy for the connection setup.
- Does the system communicate with a remote server or with a local one?

  In case of a local system (client or server), consider all related background processes. For instance, service discovery protocols such as mDNS tend to create excessive current draw owing to their defined nature.
- Is the IP acquisition method static IP or DHCP?

  DHCP process may consume a significant amount of energy.
- Is the IP addressing method IPv6 or IPv4?

  The IPv6 method may consume more energy; for applications that are sensitive to power consumption and where IPv6 is not mandatory, TI recommends disabling the IPv6 interface.
- Is a provisioning process required?

  AP provisioning is the method used to obtain initial connection between the device and the AP. One popular method is using the device as an AP, accessing it from a mobile device, or using a PC to access the device as an AP for its configuration. Although provisioning is an infrequent event, the power consumed during the process may be much higher than the power in regular device operation.
- What is the Round Trip Time (RTT), the typical delay between the end device and its peer?

  This delay affects the maximum throughput of the system and the power consumption required per KB.

For example, consider a sensor system that both communicates with a server for remote online capabilities (configuration, observation and alert indications), and provides the option for local access directly from smartphone or PC while at home. The first requirement can be supported by a system that is off for long period of times (minutes), checks the server periodically, or accesses the server whenever it detects an environmental change. The second requirement of local accessibility is more challenging from a power perspective. The system must always be responsive, thus it must be connected to the AP at all times.

1.2.4 Solution-Specific Parameters

This section describes some of the critical factors and parameters of the networking system that become the building blocks of each use case, and defines the total power consumption.

- Maximum throughput – This factor corresponds to the time it takes the system to receive or transmit a fixed amount of data, and thus the power consumption required per KB. This number should reflect the lowest common denominator in the system. For example, assume that the host interface can deliver 12 Mbps and a Wi-Fi connection can deliver 60 Mbps. For UDP TX, expect to get closer to 12 Mbps. For a TCP connection with very long RTT, the maximum TP may degrade to 2 Mbps or lower.
- Initialization time and energy – For periodic use cases that involve frequent short activity cycles with a long off-time in between, energy consumed during initialization may become a major contributor.
- Static current consumption – Any system use case is constructed from several building blocks, such as RX current, TX current, sleep current, and so forth.
- Dynamic energy consumption – System optimization for low power, and the system's efficiency with respect to power consumption while performing a specific activity. For example, connecting to an AP, initiating TCP connection, maintaining a connection with AP, and sending or receiving a fixed amount of data.

1.3 Networking Subsystem Quick Overview

The CC3120 and CC3220 devices belong to a family of wireless networking devices. Devices that belong to this product line consist of a full network stack over 802.11b/g/n. These devices are highly-integrated solutions both from the hardware and software standpoint, including internal PA and DC-DC circuit hardware accelerators for software offloading. The device supports industry-standard BSD socket API, internal SSL/TLS security protocols, very low footprint driver requirements, and advanced low-power modes.

The two main variants for Simplelink™ Internet-on-a chip™ devices are:

- CC3120 – Full networking device controlled by an external host MCU.
- CC3220 – Full system containing a networking subsystem, along with Arm® Cortex®-M4 MCU (CM4) and peripherals.

In the CC3220 device, the user programming the CM4 controls the power mode of the CM4 subsystem, however in the networking subsystem, the power management is handled autonomously. Once enabled, the subsystem always seeks to consume as little power as possible. The focus in this document is on the networking subsystem to better equip the programmer of the CM4 in the CC3220 device or the external host MCU in the CC3120 device with the required knowledge to build an optimal power efficient system.

1.3.1 CC3120 and CC3220 Main Blocks

The Networking subsystem contains two main blocks (Figure 1):

- Wi-Fi block
- NWP (network processor) block

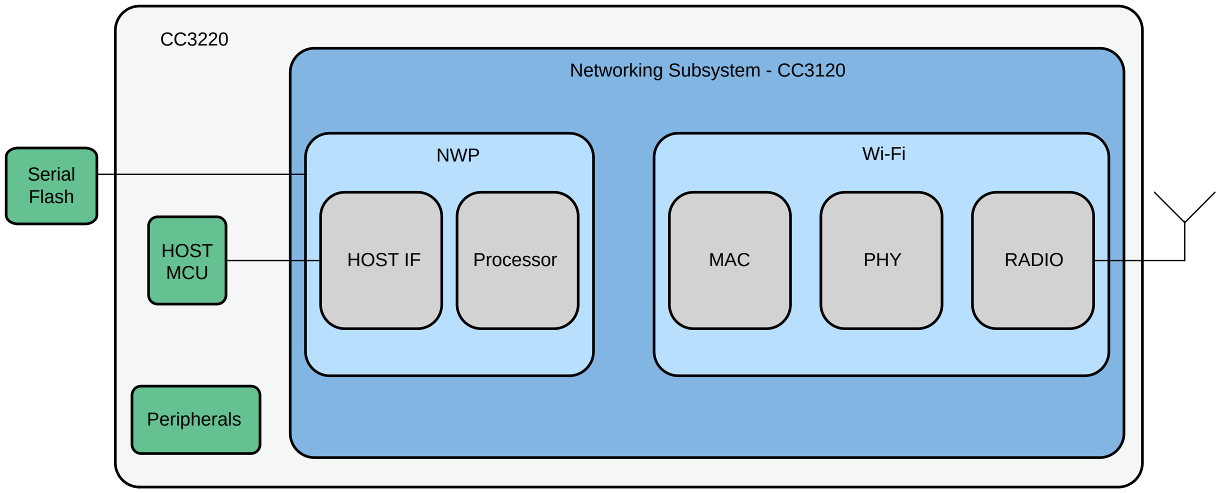


Figure 1. CC3120 and CC3220 Device Block Diagram

1.4 CC3120 Device Power Modes

The CC3120 device contains only the Networking subsystem and is self-contained in terms of power optimization. The CC3120 device has four power modes:

- **Shutdown** – Lowest power mode. Memories are not retained. RTC is not running. Requires cold boot initialization, including slow clock stabilization. Device is powered off.
- **Hibernate** – Lowest power mode which keeps the RTC clock and RTC counter running. Requires cold boot initialization. Device is powered off except for the hibernate logic.
- **Low-Power Deep Sleep (LPDS)** – Voltage levels are lowered, fast clocks (40-MHz crystal and internal PLL) are off. Memories are in retention mode. Device stays in low-power deep sleep mode if the Wi-Fi and NWP blocks have no immediate activity. Each of them manages its own sleep and wakeup events, and when both are in their low-power mode, the entire Networking subsystem is in LPDS mode.
- **Active** – Device is fully active, voltage levels are at their operational value, and all clocks are active. This mode may represent a wide variety of intermediate power states which are of transient nature and not explicitly controlled by the system.

Table 1 lists the CC3120 device power modes and the block states.

Table 1. CC3120 Device Power Modes

Networking Subsystem Power Mode	CC3120 Device Power Mode	CC3120 Device Block State	
		NWP	**Wi-Fi**
Shutdown	Shutdown	Off	Off
Disabled (off)	Hibernate	Off	Off
LPDS	LPDS	Retention	Retention
Active[1]	Active	Active	Retention
		Retention	Active
		Active	Active

[1] In Active mode, NWP and Wi-Fi can be in active or retention mode separately. If both are in retention, then the system is in LPDS mode.

1.5 CC3220 Power Modes

For complete understanding of the CC3220 power modes, consider these three aspects:

- MCU subsystem power mode – Controlled by the MCU application.
- Networking subsystem power mode – Maintains power modes automatically when enabled.
- Device level (chip) power mode – Derived from a combination of MCU subsystem and Networking subsystem power modes.

Table 2 lists the CC3220 device power modes according to the MCU power mode and the networking subsystem power mode.

Table 2. CC3220 Power Modes

MCU Power Mode	Networking Subsystem Power Mode		
	Disabled	**LPDS**	**Active**
Hibernate	Hibernate	N/A	N/A
LPDS	LPDS	LPDS	Active
Sleep	Active	Active	Active
Active	Active	Active	Active

1.5.1 Device Power Modes

The device can be in one of four modes:

- **Shutdown** – Lowest power mode. Memories are not retained. RTC is not running. Requires cold boot initialization, including slow clock stabilization. Device is powered off.
- **Hibernate** – Lowest power mode which keeps the RTC clock and RTC counter running. Requires cold boot initialization. Device is powered off except for the hibernate logic.
- **Low-Power Deep Sleep (LPDS)** – Voltage levels are lowered, fast clocks (40-MHz crystal and internal PLL) are off. Memories are in retention mode. Most of device logic is power-gated except for hibernate logic and top level logic.On wake-up, the MCU starts from the reset vector.
- **Active** – Device is fully active, voltage levels are at their operational value and all clocks are active.

1.5.2 Networking Subsystem Power Modes

Networking subsystem may be in one of three modes:

- **Disabled** – NWP is off and requires a cold or warm boot when enabled, depending on the power mode state (hibernate or shutdown).
- **Low-Power Deep Sleep (LPDS)** – Networking subsystem is in LPDS mode if the Wi-Fi and NWP blocks have no immediate activity. Each of them manages its own sleep and wakeup events, and when both are in their low-power mode, the entire networking subsystem is in LPDS mode.
- **Active** – At least one block (NWP or Wi-Fi) is running. This mode may represent a wide variety of intermediate power states that are of transient nature and are not explicitly controlled by the system.

1.5.3 MCU Power Modes

MCU can be in one of five power modes, as dictated by the MCU programmer:

- **Shutdown** – Lowest power mode. Memories are not retained. RTC is not running. Requires cold boot initialization, including slow clock stabilization. Device is powered off.
- **Hibernate mode** – Lowest power mode that still keeps the RTC running to enable faster wakeup.
- **Low-Power Deep Sleep mode (LPDS)** – While the MCU in this mode, two scenarios may occur:
  - The networking subsystem is disabled and the device mode is LPDS.
  - The networking subsystem is enabled and the device mode is LPDS (if networking subsystem is in LPDS mode) or active (if networking subsystem is active).
- **Sleep mode** – The MCU clocks are gated off in sleep mode and the entire state is retained. Device mode is active. The MCU application exercises this mode using CM4 assembly instructions, such as WFI or WFE.
- **Active mode** – MCU is running and the device mode is active.

1.6 Networking Subsystem Mode Switching

There are three modes that the networking subsystem may be in during normal operation: Active, LPDS, and disabled (OFF).

- Transitioning from OFF to active mode and from active mode to OFF mode is controlled by a host application using SimpleLink™ APIs.
  - **sl_Start** API takes the networking subsystem immediately to active mode.
  - **sl_Stop** API puts the networking subsystem in disable mode no later than the timeout parameter specified by the API.

    In the CC3120 device, **sl_Start** asserts the nHIB or nRESET pin of the device, and **sl_Stop** de-asserts this pin after sending a stop command to device.

    In the CC3220 device, **sl_Start** configures the internal register that enables the NWP. **sl_Start** initiates the cold boot initialization process of the networking subsystem. Initialization is finished when the networking subsystem asserts the IRQ line and the SL driver reads the **sl_Start** API status.
- Transitioning between LPDS and Active modes: in the context of the networking subsystem, entry to and exit from LPDS mode is dictated by activity. There is no direct intervention of the host application. Both Wi-Fi IP and NWP IP manage their activity according to their state (such as connected to AP, connected to server, and so forth). When one of the IPs has no immediate activity (within a near specific threshold) it may go to lower power mode. When both IPs are in their lower power mode, the entire networking subsystem is in the LPDS mode. Neither the host application nor the SL driver need to know whether the networking subsystem is in active mode or LPDS mode. When the host application wants to communicate with the NWP (assuming it is not disabled) it simply issues an API call. If it is in LPDS mode when the command is sent, then NWP wakes up and handles the command thereafter. Note that the networking subsystem does not enter LPDS mode while the IRQ line signaling an event to the host is active. The IRQ line is cleared once the host driver reads the event status back from the NWP. Therefore it is advised that power consumption-sensitive systems respond to NWP IRQ interrupts as quickly as possible.

Figure 2 shows the possible transitions between states, and the trigger for the transition.

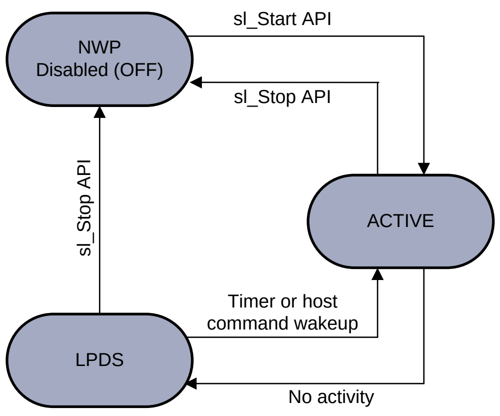


Figure 2. Transitions Between States

1.7 Power Policies

From the host application perspective, there are only two modes of operation explicitly selected by the host: NWP disabled (OFF) or NWP enabled (ON). Selection between Active or LPDS states is managed internally by the NWP, using power management algorithms.

The networking subsystem is equipped with a policy management entity that allows a developer (host application programmer) to guide the behavior of the power management algorithm through predefined power policies. The **sl_PolicySet** API configures the device power management policy. The available policies are:

- **Normal (Default)** – Best tradeoff between traffic delivery time and power performance. When connected to an AP, the Wi-Fi module wakes up for every beacon reception. Wi-Fi and NWP modules enter their low-power mode after considering current activities and predict future activities.
- **Low power** – The NWP power management algorithm is more opportunistic, exploiting opportunities to lower the power mode. Tradeoff tends toward power conservation performance (for example, tag application). The networking subsystem enters LPDS immediately once the activity is over, without predicting future activities. Almost every communication between the host and NWP takes the overhead of waking up the subsystem, without any time in idle mode predicting future events. Low Power policy is suitable primarily for unconnected applications (such as applications that use transceiver mode, and not connecting to AP). When used in a connected scenario, behavior and service is not assured.
- **Long Sleep Interval (LSI)** – When an 802.11 station is connected to the access point, it must receive the beacons transmitted by the AP. APs typically transmit a beacon every 102.4 ms. 802.11 standards define the DTIM (Delivery Traffic Indication Map) as a specific beacon that contains information regarding incoming packets for the STA. The AP may choose its DTIM interval (such as 1-every beacon, 2-every other beacon, and so forth). This special low-power policy instructs the networking subsystem to skip beacons and DTIM packets, and comes with a desired max sleep time parameter. The parameter reflects the desired sleep interval between two consecutive wakeups for beacon reception. The Wi-Fi module computes the desired time and wakes up to the next DTIM that does not exceed the specified time (see Table 3 for examples). The desired maximum sleep time parameter is 2 seconds. TI strongly recommends setting the LSI parameter to less than half a second to ensure reliable service while lowering current consumption.

NOTE: This policy only works in client mode and external connection (internet connection through gateway). It automatically terminates mDNS and internal HTTP server running on the device. TCP/UDP servers initiated by the user application lead to unpredictable system behavior and performance.

Table 3. Always Connected Power Policies

AP Beacon Interval [T.U]	AP DTIM Configuration	Desired Max Sleep Time [mSec]	Actual Wi-Fi Sleep Time [mSec]
100	DTIM = 1	200	204.8
100	DTIM = 2	500	409.6
100	DTIM = 1	500	512
100	DTIM = 3	1400	1228.8
100	DTIM = 1	2000	2048
100	DTIM = 4	800	819.2

- **Always On** – Both Wi-Fi and NWP modules remain active and do not enter their low-power modes. Wi-Fi does not enter 802.11 power save mode. Provides the lowest response time at the expanse of power consumption.

1.8 Low-Power Design With CC3120 and CC3220 Devices

This section merges the low power considerations with the detailed characteristics of the CC3120 and CC3220 products. The section introduces key features in the networking subsystem targeted to save power and explains the key tradeoffs that a designer should consider and resolve while working with this product.

1.8.1 Connection Policies

The networking subsystem is equipped with an advanced connection policy manager. The user may define up to seven connection profiles stored in the device NVMEM.

A profile (set using **sl_WlanProfileAdd**) is a structure of SSID, password, and a priority for the profile. Whenever the networking subsystem is enabled and not connected to an AP, it strives to connect to the AP with the higher priority profile. This behavior is enabled when the automatic connection policy is set (using **sl_WlanPolicySet**). While connected to an AP, the networking subsystem does not switch to another AP with higher priority.

A key feature of the connection manager is the fast connect feature (also set using **sl_WlanPolicySet**). Fast connect policy dictates that upon initialization (transition from NWP disabled to NWP enabled modes) or in the event of a disconnect, the networking subsystem immediately tries to connect to the last AP that it was connected to. It remembers the SSID, security credentials, and the channel of the last AP and automatically tries to connect to that network. The main advantage of fast connect over auto connection is that the system skips the scan. When both auto connection and fast connection are configured, the system first tries to connect to last AP according to fast connect policy; if it fails, it runs the scan process and looks for the highest priority profile, then connects to it.

By skipping the scan process, a significant amount of energy is preserved. The scan process may be long (few hundreds of milliseconds), during which the modem is active, switching between channels, sending probe request packets, and listening for responses. Therefore, for every system use case that contains multiple connections, using the fast and auto connect features is recommended.

1.8.2 Service Discovery

Upon connection to an AP, NWP automatically starts advertising itself by sending mDNS packets. Stopping the mDNS feature when it is not used by application helps saving power. Stopping the mDNS feature is done using the **sl_NetAppStop** API, and can be done once as an indication is stored in NVMEM.

1.8.3 Host Interrupt (IRQ) Handling

The networking subsystem handles its own power states, seeking to move to the lower power state (LPDS) whenever possible. However, the NWP does not move to LPDS mode while the host IRQ line is asserted (only applicable for CC3220 device). NWP asserts the IRQ toward the host to indicate on a command response or asynchronous event. The host is then expected to read the event from the NWP, and the NWP clears the IRQ line. Because the NWP does not enter LPDS with the IRQ line asserted (only applicable for CC3220 device) and does not clear it until the host reads the status, ensure that the host handles the IRQ line as soon as possible.

1.8.4 Reducing Host Wakeups using Device Built-in System Filters

The device is equipped with an advanced and configurable filtering mechanism in different layers. Smart usage of the filters can reduce the amount of unwanted packets transferred to the host, and cause host unwanted wakeup. By default, the device filters MAC layer broadcast and multicast and IP layer multicast that do not belong to multicast groups joined by the device.

1.8.5 Serial Flash Handling

The CC3120 and CC3220 devices use the serial flash (SF) for nonvolatile storage. With respect to power management, verify that the SF does not become a system bottleneck. SF may be in one of three modes:

- **Active** – When the device reads from the SF or writes to the SF. NWP uses the SF to store internal data and may write to SF asynchronously.
- **Standby** – When the device does not use the SF, the CS (chip select) line is held high. This is also the SF state during LPDS mode.
- **Power Down** – Before the CC3120 or CC3220 device enters hibernate, it sends the power down command to the SF. This command is handled by the NWP in the CC3120 device and is handled by the hibernate driver in the CC3220 device. For the CC3120 device, use the timeout parameter along with the **sl_Stop** command to allow the NWP to send the power down command.

1.8.6 TX Output Power

The networking subsystem transmits with its maximum output power by default. The programmer may reduce this parameter by setting a back off parameter using the **sl_WlanSet** API. This parameter defines the additional back off (in dB) that the radio takes. If the user sets the value to four or above, the networking subsystem radio switches to use a low power PA, and TX current is reduced significantly.

NOTE: The output power is reduced by the same factor to all rates. For example, a maximum output power for 1 Mbps is 18 dBm, and 14.5 dBm for 54 Mbps. If the value is set to 4, the output power of 1 Mbps is reduced to 14 dBm and the 54 Mbps to 10.5 dBm. Therefore, if the user decides to reduce the maximum output power to reduce overall power consumption, the transmission rate drops, the transmission length is higher, and overall power consumption may increase.

1.8.7 CC3120 Device (Networking Subsystem) Current Consumption

The CC3120 SimpleLink™ Wi-Fi® Wireless Network Processor, Internet-of-Things Solution for MCU Applications Data Sheet should be used as the main source of current consumption numbers.

1.8.8 CC3220 Device Current Consumption

The CC3220 SimpleLink™ Wi-Fi® Wireless and Internet-of-Things Solution, a Single-Chip Wireless MCU Data Sheet should be used as the main source of current consumption numbers.

1.8.9 Always-Connected Current

Idle connection current is an important parameter, especially for always-connected systems. This is the average current that the entire system consumes while connected to the access point without any traffic exchange. The Wi-Fi uses the 802.11 power save to reduce its power and only receive beacons. While in this mode, the networking subsystem is in LPDS mode most of the time. The Wi-Fi block wakes up to receive a beacon, and goes back to sleep. If the beacon carries an indication that the AP holds a packet for the device or an indication for a broadcast packet that follows the beacon, the system fully wakes up to handle the traffic. By default, the Wi-Fi IP wakes up for every beacon according to the AP TBTT parameter, and transmits a keep alive packet to the access point every 55 seconds while in this mode.

As mentioned in Section 1.7, the networking subsystem introduces an advanced feature of (LSI) Long Sleep Interval. In this mode the programmer sets the desired sleep time between beacons, understanding that beacons are missed in between. Table 4 shows the expected average current consumption per sleep interval.

Table 4. Expected Average Current Consumption Per Sleep Interval

PM Mode	Wakeup Interval Time [ms]	Average Current [mA]
Default	102	0.690
LSI	204	0.419
LSI	510	0.284
LSI	1020	0.233
LSI	1530	0.208
LSI	2000	0.2

Notes:

- Serial flash current in standby mode is not included.
- 2 seconds is the maximum allowed LSI, while the maximum recommended value is 500 ms.
- Tested with few AP models in clean RF environment.
- Measured at 3.3-V VBAT.
- Measurements are based on an R2 CC3220 device.

Referring to the default TBTT (102.4 ms), the CC3120 device average current during idle connected period is 0.69 mA. This performance is a result of over of ten years of Wi-Fi experience.

NOTE: This parameter may vary between different access points. Some access points and networks send many broadcast packets that force the networking subsystem to stay active longer. Some access points do not send their beacons with exact timing, missing beacons and other phenomena. Networking subsystem algorithms try to overcome many of the above phenomena, but the average current consumption may still vary.

1.8.10 NWP Initialization Time

This parameter is critical for several use cases that frequently enable and disable the networking subsystem. For example, a system that spends most of the time in hibernate mode and every T minute initiates a connection to the cloud. Initialization time is measured from the point the NWP was enabled, using the **sl_Start** command to the time the NWP asserts the IRQ line. This time may vary based on the following:

- Size of the service pack loaded from serial flash.
- Number of configurations stored in the serial flash. For example, connection profiles, static IP, MAC address, and so forth.
- Calibration (if running and not using a one time calibration option) adds around 200 ms to the NWP initialization time.

The typical initialization time for the NWP is 60 ms, with a charge of approximately 1000 µC at 3.3 V. See the CC3120, CC3220 SimpleLink™ Wi-Fi® and Internet of Things Network Processor Programmer's Guide.

1.8.11 Wi-Fi® Connection Charge

When using the fast connect feature (skipping the scan process), the networking subsystem automatically connects to the access point. The charge that connection process consumes is an important parameter for systems that are not in always-connected use case. Connection time and charge may vary significantly from one access point to another, and can also vary within the same access point from one connection attempt to another based on the access point load and activity.

The typical charge for WPA2 connection is 1000 µC to 2000 µC at 3.3 V, and is highly depending on the AP used. The measurements listed in Table 5 were done on the CC3120 device R1 SP1 and MSP430™ host using fast connect feature.

Table 5. Wi-Fi® Connection Charge

AP	WPA2 Connection Charge [µC]
Cisco 1250	1000
Netgear 3500	1400
TP-Link WR2041	1300

1.8.12 TCP and SSL/TLS Connection Charge

This parameter varies significantly from one server to another and according to the selected cypher suite. The typical connection time and charge to a server that responds quickly are presented in Table 6.

Table 6. TCP and SSL/TLS Connection Charge

Cipher Suite	Connection Charge [µC]
Open (non secure TCP)	3500
SSL/TLS (RSA RC4-128 MD5)	4500
SSL/TLS (RSA RC4-128 SHA)	4200
SSL/TLS (RSA AES-256 CBC SHA)	4000
SSL/TLS (DHE RSA AES-256 CBC SHA)	9700
TLS (ECDHE RSA RC4/AES SHA)	44500

Notes:

- A local PC server was used for the measurement.
- Zero round trip delay time assumed. Larger round trip delay time increases the connection time significantly. However, the charge per connection may remain closer to the specified result here because most of the time increase is spent in power save mode.

1.8.13 Traffic Exchange Charge and Considerations

Application traffic patterns vary from one use case to another. From a power consumption perspective, the characteristics of the traffic make a great difference.

While there is no traffic, the system is in idle connected mode. The system tries to stay in low-power mode as long as possible, and every traffic event initiated by the host (TX) or by the AP (RX) takes the system out of idle connected mode.

The overhead when waking the system for traffic and exiting 802.11 power save mode is not negligible. Therefore, the amount of times the system wakes up for traffic should be minimized. Minimize the amount of wakeups over the duration of each wakeup. For example, sending 10 packets as fast as possible is much better than sending same packets with 30 ms delay between them.

The following guidelines improve overall power consumption:

- Consolidate application data into large packets.
- Concentrating traffic in bursts is better than uniform distribution of the traffic.
- If a system includes the server, than the server should also concentrate the traffic in bursts. (Reduces number of node wakeups)

UDP TX is an exception to the above guidelines. If the use case requires transmission of a small amount of packets at each time (up to 5), better results are achieved when delay between packets is greater than 50 ms.

Table 7 shows the average current consumption of TX/RX in UDP/TCP use cases when traffic is transmitted and received as burst. The specified payload is transmitted or received as a burst, then the device is in idle for the remaining time of the second. The process repeats every second.

Table 7. Average Current for TX/RX in Burst

TP [Mbps]	UDP		TCP	
	TX [mA]	RX [mA]	TX [mA]	RX [mA]
0.1	5.8	5.6	6.5	6
0.5	9.1	8.05	9.5	10
1	13.5	11.5	14	15.5
2	22	18	22	26
3	30	24	31	36
5	46.5	36.5	47.5	57
12	105	80	107	94

Notes:

- 1460B packet length
- Excellent link quality assumed
- Tested with few types of AP in clean RF environment

1.9 Battery Powered System Considerations

For battery-operated systems, some considerations should be understood.

1.9.1 Minimum Operating Voltage

The CC3120 or CC3220 device specifies that the minimum operating voltage is 2.1 V. To ensure proper operation, the battery voltage should always be above this threshold. Battery voltage tends to drop while the battery discharges, according to battery specification and chemistry. The battery voltage also drops when a high current is drawn from it, according to its internal resistance. Typical internal resistance for an AA battery is 1 Ω. Considering the TX current and the calibration peak current, the voltage drop can reach 400 mV on alkaline batteries. Based on this system characteristic, the minimum operational for an AA battery is 2.5 V for 2AA on alkaline batteries.

1.9.2 Usable Battery Capacity

The usable battery capacity is the battery capacity that the battery supplies before the voltage drops below 2.5 V. It depends on the battery discharge characteristics. For Alkaline batteries (analysis is done using Duracell MN1500 AA) the capacity increases as the average current drops. The battery data sheet indicates the capacity listed in Table 8 while the voltage remains above 2.5 V.

Table 8. Battery Capacity

Average Current	Capacity Above 2.5 V
50 mA	1550 mAh
25 mA	1850 mAh
10 mA	1990 mAh
5 mA	2090 mAh

The trend is expected to continue down to an average current of 1 mA and below. The usable capacity for an average current of less than 1 mA reaches 2200 mAh for this specific battery.

Lithium batteries have better internal resistance (analysis based on the Energizer® L91) and their voltage and service hours curve is more flat. This gives an advantage to lithium batteries, as almost the entire capacity can be used. With the analyzed battery, more than 3000 mAh can be used.

1.10 Example Test Cases

1.10.1 Intermittently Connected Test Case

A CC3120-based system must transmit 100B and check messages on the server every few minutes. In this test case, the system initiates the communication and is not responsive at all times. Therefore, the system can initiate a new connection on every cycle and does not have to keep connection. If the cycle time is long enough, the system conserves more power if it switches to hibernate mode between transactions, and reconnects to the AP and server every time rather than maintain the connection.

In this test case, the total energy per activity cycle is a summation of the following activities:

- E1 is energy spent during system initialization.
- E2 is energy spent during reconnecting to the AP (802.11 association and authentication).
- E3 is energy spent while reconnecting to the server (IP layer and transport layer protocols per application requirements).
- E4 is energy spent during application traffic.
- E5 is energy spent by the CC3120 device between cycles while in hibernate mode.

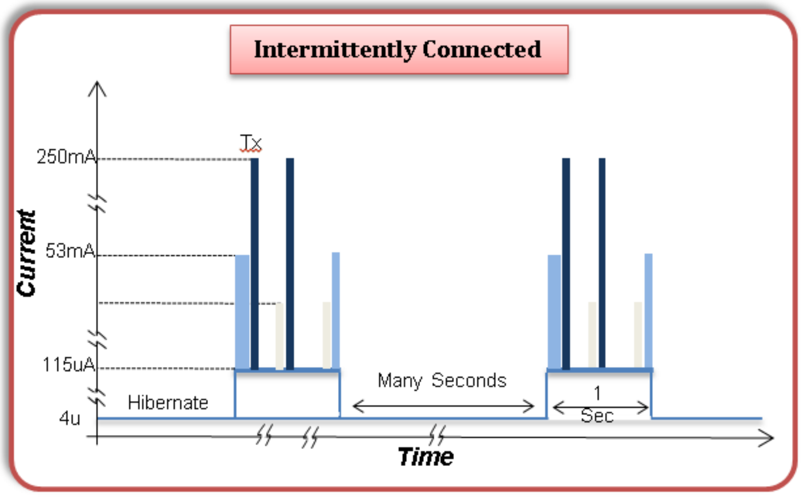


Figure 3. Intermittently Connected Test Case

In this test case, the application enables the CC3120 device, the device automatically reconnects to the AP, the application opens and binds a socket, initiates traffic and disables the device. While the CC3120 device is enabled, it manages its power state according to a defined policy. For example, it may be in LPDS state while waiting for a server response, or if the host delays between commands.

The total energy spent and system life span are given by Equation 1.

$E = \Sigma^{5}_{n=1} E_n$

$T = V \times B / E \times C \times 1 / 400$

where

- E is Total energy per cycle [joule]
- B is Battery capacity [mAh]
- C is Cycle time [min]
- V is Voltage [volts]
- T is Device life span [days] (1)

Example 1. Intermittently Connected Test Case Example

Energy

- C – (Activity period) = 2 min
- Typical average current draw over a single activity period:
  - E1 = 1700 µC at 3.3 V = 5.6 mJ
  - E2 = 2000 µC at 3.3 V = 6.6 mJ
  - E3 = 4000 µC at 3.3 V = 13.2 mJ
  - E4 = 5000 µC at 3.3 V = 16.5 mJ
  - E5 = 600 µC at 3.3 V = 2 mJ
  - E_{Total} = 44 mJ
  - Average Current = 111 µA

Battery: two AA alkaline batteries rated at 1.5 V, each connected in series with a capacity of 2000 mAh:

- B = 2000 mAh
- V = 3.0 V

T (device life span) = 3 × 2000 × 2 / 0.044 / 400 = 681 days

1.10.2 Always-Connected Test Case

In this test case, the application requires that the system maintain its connection (see Figure 4). The system must stay online and respond to notifications within certain latency (a few seconds). The traffic is scarce and most of the time the system is idle. For example, a server waiting for clients to connect or a client constantly connected to a cloud server. In this case, the host enables the NWP once and initiates a connection to the AP. As long as there is no activity in the system, the system is in the lowest possible mode (using the 802.11 power save protocol to handle low power at the link layer). The system manages the following activities:

- Wakeup for beacons to check if an incoming traffic exists at AP buffers
- Upon indication of incoming traffic, the system exits low-power mode, receives the traffic and transfers to the host if necessary, according to configured filters.
- Manages keep alive handshake with the AP to maintain a connection.

The average energy consumed by the system is a summation of the following activities:

- E1 is energy spent for connection maintenance
- E2 is energy spent during application traffic

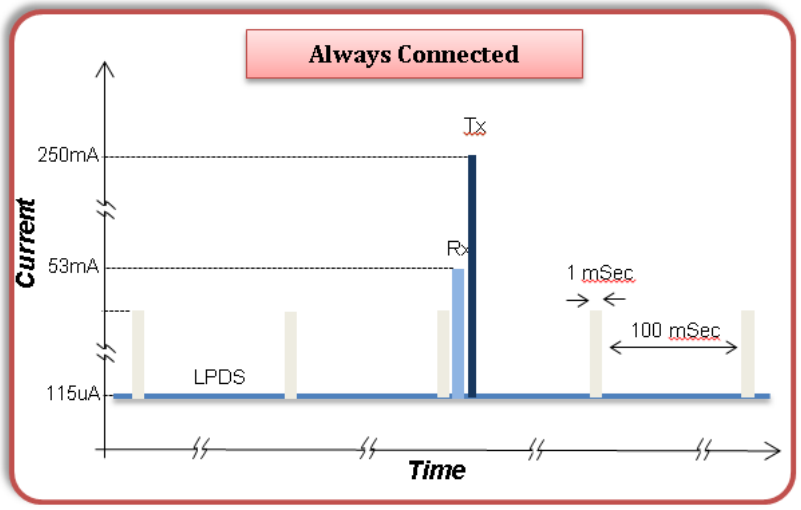


Figure 4. Always Connected Test Case

The total energy spent and the system life span is given by Equation 2.

$T = B \times V / W_1 + E_2 / P_2 \ (1 / 1000 \times 24)$ days

where

- B is battery capacity [mAh].
- V is voltage [volts].
- P_2 is the period cycle for application traffic activity [sec].
- W_1 is power consumed during idle connection time [Watt].
- E2 is energy spent during application traffic [Joule]. (2)

Energy spent during the initial phase of device initialization and connection establishment can be neglected, because this is a one-time event.

Example 2. Always-Connected Test Case Example

Energy

- P_2 = 60 s
- W_1 = 233 µA × 3.0 V = 0.0007 W (assuming a long sleep interval of 1 sec)
- E_2 = 25 mA × 200 ms × 3.0 V = 15 mJ (assuming part of the 200 ms is at RX current, very short time at TX current, and part in LPDS current resulting in 25-mA average over the 200 ms)

Battery: two, AA, alkaline batteries rated at 1.5 V, each connected in series with a capacity of 2000 mAh:

- B = 2000 mAh
- V = 3.0 V

T (device lifespan) = 2000 × 3.0 / (0.0007 + 0.015 / 60) / (1000 × 24) = approximately 263 days (3)

2 Power Measurement Guide

This section discusses the steps needed to experience the low-power modes and test cases of the CC3120 BoosterPack™ or the CC3220 LaunchPad™, using the *power_measurement* source code example in the CC3120 and CC3220 Software Development Kit (SDK) package provided by TI.

2.1 CC3120 and CC3220 Device Prerequisites

This following sections cover the standard equipment and hardware required of the CC3120 BoosterPack™ and CC3220 LaunchPad™ for reliable power measurements. Gather the following items before running the example code in Code Composer Studio™ (CCS) Integrated Development Environment (IDE) and IAR.

2.1.1 Standard Equipment for CC3120 and CC3220 Devices

The standard equipment for the CC3120 and CC3220 devices follows:

- PC capable of running the latest version of CCS or IAR as well as Uniflash
- Terminal emulator program, such as Tera Term, installed on the PC
- Micro-USB cable (included with EVM)
- DC Power Analyzer or other tool for plotting and measuring current consumption
- Access point (AP)

2.1.2 Required Hardware for CC3120 and CC3220 Devices

The required hardware for the CC3120 and CC3220 devices follows:

CC3120 device hardware:

- CC3120BOOST
- CC31xxEMUBOOST (for flashing the Service Pack on the CC3120 device)
- MSP432-EXP432P401R (to use as the host processor)

NOTE: See the CC3120 BoosterPack User's Guide for instructions on:
- Flashing the Service Pack using the CC31XXEMUBOOST device
- Connecting the CC3120BOOST device to the MSP432-EXP432P401R device

CC3220 device hardware:

- CC3220S-LAUNCHXL or CC3220SF-LAUNCHXL

2.2 Software Setup

The following section provides instructions for running the *power_measurement* software example and measuring the performance of the devices in each power mode or test case.

2.2.1 Download and Install

Follow these instructions to download and install the software required to run *power_measurement*.

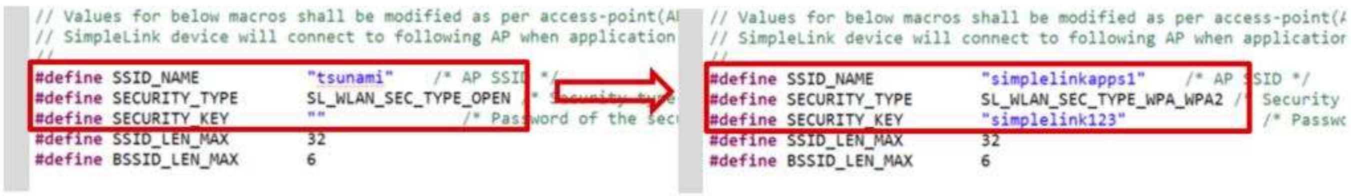
1. Download and install the latest version of CCS or IAR.
2. Download and install the SimpleLink™ SDKs (CC3220 SDK, or MSP432 SDK with the CC3120 Add-on).
   - For the CC3120 device, select Yes when prompted to install the FTDI drivers.
   - For the CC3220 device, see the CC3220 Getting Started Guide for more information on installing the XDS110 drivers.
3. Download and install the latest Uniflash tool.

NOTE: Users must see the CC3120 and CC3220 Getting Started Guide for instructions on:
- Building the application using CCS or IAR
- Using the UniFlash tool to flash the latest Service Pack and the precompiled binary file from the *power_measurement* project. These files are included in the SDK packages. Continue following this guide to Section 2.2.4 before flashing the latest Service Pack and application code.

2.2.2 Macro Definition

The *power_measurement* example allows users to quickly start measuring power consumption of CC3120 or CC3220 devices by easily configuring the application to exercise one of the following power mode and test case options: Hibernate, Low-Power Deep Sleep (LPDS), Transceiver mode, Intermittently Connected mode, or Always Connected mode. In the Intermittently Connected or Always Connected mode test cases, users must connect to an AP.

1. After importing the *power_measurement* example into your workspace, open the common.h file. Inside this file are macro definitions which manage application settings.
2. Users must define each macro with the information for the AP. Edit the SSID_NAME, SECURITY_TYPE, and SECURITY_KEY macros to contain the AP information, as shown in Figure 5.
   - For open security, define SECURITY_TYPE as SL_WLAN_SEC_TYPE_OPEN.
   - For WPA and WPA2 security, define SECURITY_TYPE as SL_WLAN_SEC_TYPE_WPA_WPA2.
   - For the SSID_NAME and SECURITY_KEY macros, the quotation marks must remain as part of the macro definition.



```
// Values for below macros shall be modified as per access-point(AI
// SimpleLink device will connect to following AP when application
//
#define SSID_NAME              "tsunami"      /* AP SSID */
#define SECURITY_TYPE          SL_WLAN_SEC_TYPE_OPEN /* Security type
#define SECURITY_KEY           ""             /* Password of the Sec
#define SSID_LEN_MAX           32
#define BSSID_LEN_MAX          6
```

```
// Values for below macros shall be modified as per access-point(A
// SimpleLink device will connect to following AP when applicatior
//
#define SSID_NAME              "simplelinkapps1"    /* AP SSID */
#define SECURITY_TYPE          SL_WLAN_SEC_TYPE_WPA_WPA2 /* Security
#define SECURITY_KEY           "simplelink123"      /* Passwc
#define SSID_LEN_MAX           32
#define BSSID_LEN_MAX          6
```

Figure 5. Editing common.h File

NOTE: The default settings for the Intermittently Connected or Always Connected power modes in the *power_measurement* example are listed as follows:

- DHCP: fast renew + no wait
- AP reconnect: auto + fast connect
- Number of packets: 1 (1000 bytes)
- LSI duration: 100 ms

See the CC3120, CC3220 SimpleLink™ Wi-Fi® and Internet of Things Network Processor Programmer's Guide for more information regarding these parameters.

2.2.3 Configuring the Socket

With the Intermittently Connected and Always Connected mode test cases, the user can configure a macro, specifying the socket type. This socket can be configured to establish either a UDP or TCP connection. A client and server are required to establish a TCP connection, and using Python scripts is a simple solution to set up a peer server.

2.2.3.1 Configuring the Client for TCP Connection

To configure the SimpleLink™ device for a TCP connection and run a peer server:

1. In the definition list of the power_measure.c file, identify the SOCKET_TYPE macro:

```
#define SOCKET_TYPE                    SocketType_UDP /* options -
> SocketType_UDP, SocketType_TCP, SocketType_SEC_TCP */
```

2. Define SOCKET_TYPE as:
   - SocketType_TCP (unsecured socket, TCP connection)

     or
   - SocketType_SEC_TCP (secure socket, TCP connection)

NOTE: For more information on secure sockets, see the following:

- Network Layer Security section of the CC3x20, CC3x35 SimpleLink™ Wi-Fi® Internet-on-a chip™ Solution Built-In Security Application Report application report for the CC3120 and CC3220 devices.
- Secure Socket section of the CC3120, CC3220 SimpleLink™ Wi-Fi® and Internet of Things Network Processor Programmer's Guide for the CC3120 and CC3220 devices.

3. Identify the macros shown as follows. Define the network port number to be used by updating the PORT macro.

```
#define PORT                                    (5001)
#define DEST_IP_ADDR                    SL_IPV4_VAL(192,168,39,200)
#define SRC_IP_ADDR                    SL_IPV4_VAL(192,168,1)
            /* relevant for Static IP mode */
#define GATEWAY_IP_ADDR                SL_IPV4_VAL(192,168,39,242)
            /* relevant for Static IP mode */
```

4. Update the definition for DEST_IP_ADDR with the IP address of the machine that will run the python script (explained in the following section). This is the IP address assigned by the AP being used. On Windows, this can be found using the console command *ipconfig*.
5. Update the other macros if configuring the SimpleLink™ device to use a static IP address:
   - SRC_IP_ADDR is the IP address that is assigned to the SimpleLink™ client device.
   - GATEWAY_IP_ADDR is the IP address that is assigned to the AP with which the user would like to connect.
6. When implementing a secure server, modify the *cipher* and *method* variables in the bsdTcpSecClient() function to change the cryptographic protocol to Transport Layer Security (TLS), as follows. The purpose of changing the protocol to TLS is explained in the next section.

```
uint32_t cipher = SL_SEC_MASK_TLS_RSA_WITH_AES_256_CBC_SHA;
uint8_t method = SL_SO_SEC_METHOD_TLSV1_2;
```

2.2.3.2 Configuring the Server With Python Scripts

Follow these steps to configure the server with python scripts.

1. Download and install Python 2.7.x (although there are Python 3.0 installers available, they were not tested). Install the package on your PC according to the instructions the installer prompts.
2. Download and install OpenSSL to implement a secure server. See CC31xx and CC32xx Generate Certificate for the OpenSSL download and certificate generation. Place all certificate files under the same directory when you are done.
3. The following are examples of Python scripts that can be used to implement an unsecured or secured TCP server:
   - Use tcp_server.py for an unsecured socket:

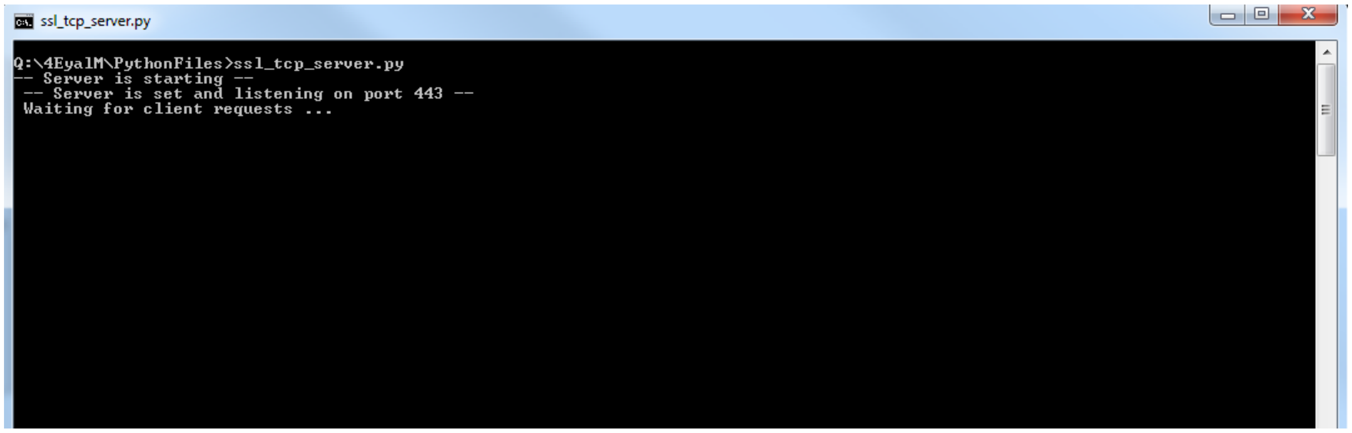
```
import socket,os

TCP_IP = raw_input ('Please enter the server IP address: ')
TCP_PORT = raw_input ('Please enter the port number: ')
BUFFER_SIZE = 1400
idx = 1

# open TCP socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((TCP_IP, int (TCP_PORT)))
print "waiting for RX"
while 1:
                    s.listen(1)
                    conn, addr = s.accept()
                    print 'Connection has been established with address: ', addr
                    while 1:
                                        data = conn.recv(BUFFER_SIZE)
                                        # data will be null when the client closes the socket
                                        if not data: break
                                        if data: print "received data:", idx
                                        #conn.sendall(data) # echo if needed, uncomment this
line if you want to reply
                                        idx = idx + 1
                    conn.close()
```

- Use ssl_tcp_server.py for a secured socket:

```
import socket, ssl
TCP_PORT = 443
idx = 1
print "-- Server is starting --"
bindsocket = socket.socket()
bindsocket.bind(('', TCP_PORT))
print "-- Server is set and listening on port " + str(TCP_PORT) + " --"

while True:
                    print " Waiting for client requests ... "
                    bindsocket.listen(1)
                    newsocket, fromaddr = bindsocket.accept()
                    connstream = ssl.wrap_socket(newsocket,

                                                serverside=True,

                                                certfile="cert.pem",

                                                keyfile="cert_privkey.pem"

                                             )
                    while True:
                                        data = connstream.read()
                                        if not data : break
                                        if data : print "packet number: ",idx
                                        idx += 1
                    connstream.close()
```

The cryptographic protocol is defined as TLS in the *ssl.py* Python script which is imported in *ssl_tcp_server.py* from the Python 2.7.x library. This is why the cryptographic protocol is changed to TLS in the previous section.

4. Open a command line terminal and run the script, as shown in Figure 6. Ensure that the PC is connected to the same AP as the SimpleLink™ device.



```
Q:\4EyalM\PythonFiles>ssl_tcp_server.py
-- Server is starting --
 -- Server is set and listening on port 443 --
 Waiting for client requests ...
```

Figure 6. Running the Command

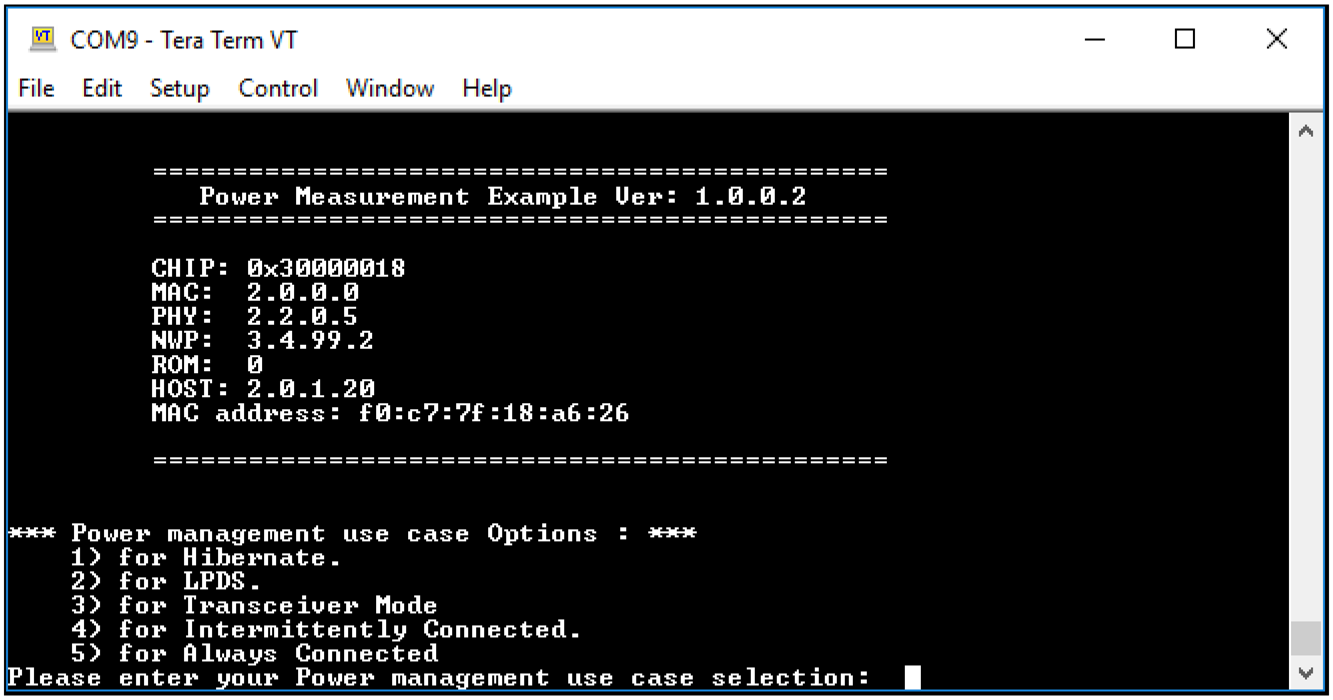
NOTE:

- When establishing a TCP connection, ensure that the server is launched first, before the client.
- When prompted, after running *tcp_server.py*, enter the IP address of your PC and network port number as defined by the PORT macro. The server will then wait to receive data.
- When running *ssl_tcp_server.py*, ensure that the PORT macro is 443, because this is the network port number defined in the Python script.

2.2.4 Running the Power Measurement Example

For the *power_measurement* example, CCS and IAR debug mode should not be used, because JTAG will disconnect in low-power modes. Instead, CCS and IAR should be used only to compile the code. After compiling, see the CC3220 SimpleLink™ Wi-Fi® and Internet of Things Solution, a Single-Chip Wireless MCU Getting Started Guide guide for flashing the latest Service Pack and application code to the device.

1. When the application code is successfully flashed on the SimpleLink™ device, open a terminal emulation program, like Tera Term, and select the UART serial port. If the terminal does not display the COM port name, see the device manager discussed in the CC3120 and CC3220 Getting Started Guide.
2. Configure the baud rate to 115200.
3. Press the reset button on the development kit.
4. If successful, the terminal emulation program should output the text shown in Figure 7. This indicates the device is ready for a power use-case selection.



```
         ==============================================
            Power Measurement Example Ver: 1.0.0.2
         ==============================================

         CHIP: 0x30000018
         MAC:  2.0.0.0
         PHY:  2.2.0.5
         NWP:  3.4.99.2
         ROM:  0
         HOST: 2.0.1.20
         MAC address: f0:c7:7f:18:a6:26

         ==============================================


*** Power management use case Options : ***
    1) for Hibernate.
    2) for LPDS.
    3) for Transceiver Mode
    4) for Intermittently Connected.
    5) for Always Connected
Please enter your Power management use case selection:
```

Figure 7. Serial Output: Power Measurement

2.3 Launching the Application

After viewing the menu that appears in the terminal program, the user must select which power mode or test case to run by inputting the appropriate number. After making the selection, the user must restart the program to make a new selection. Figure 8 shows a brief summary of what to expect for each power mode and test case.

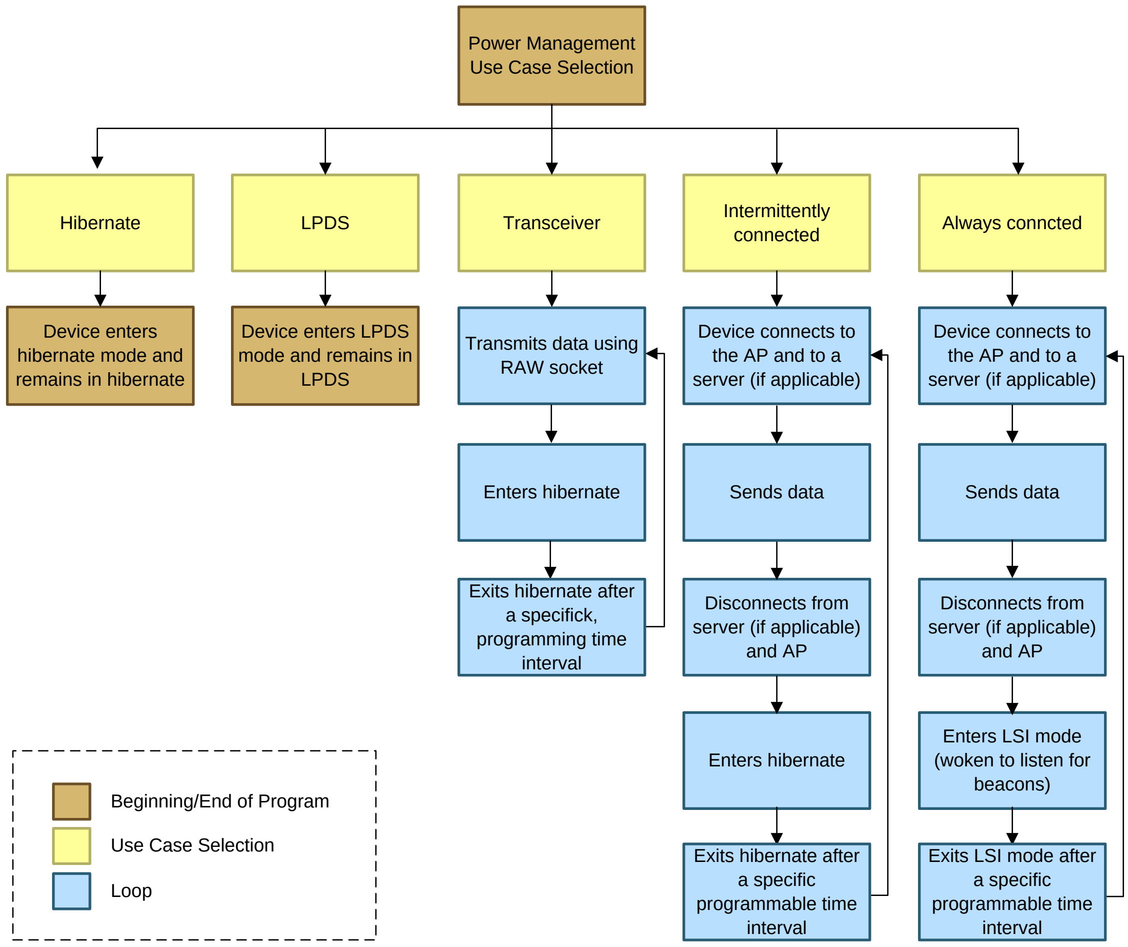

Power Management Use Case Selection
Hibernate
LPDS
Transceiver
Intermittently connected
Always conncted
Device enters hibernate mode and remains in hibernate
Device enters LPDS mode and remains in LPDS
Transmits data using RAW socket
Enters hibernate
Exits hibernate after a specifick, programming time interval
Device connects to the AP and to a server (if applicable)
Sends data
Disconnects from server (if applicable) and AP
Enters hibernate
Exits hibernate after a specific programmable time interval
Device connects to the AP and to a server (if applicable)
Sends data
Disconnects from server (if applicable) and AP
Enters LSI mode (woken to listen for beacons)
Exits LSI mode after a specific programmable time interval
Beginning/End of Program
Use Case Selection
Loop

Figure 8. Power Management Use-Case Selection Flowchart

2.4 Power Measurement Test Setup

This section describes the general setup required for performing power measurements. The current must be measured across time to properly calculate the average current consumption. That being the case, a basic multimeter is not sufficient for cases where the current fluctuates, and the current consumption of the device should be measured using one of the tools described in the following sections.

2.4.1 Measuring Current Consumption With a DC Power Analyzer

The easiest way to accurately measure and evaluate current consumption is using a DC Power Analyzer with built-in efficiency measurement capabilities. This setup is suited for both ultra-low and active current measurements.

Tools needed:

- DC Power Analyzer (for example, Keysight N6700 Power Modules)
- Short cables to connect the DC Power Analyzer to the SimpleLink™ device
- PC

Prerequisites and things to consider:

- Ensure the DC Power Analyzer has built-in sourcing and measuring, current-measuring, and data logging functions.
- Ensure the *power_measurement* example code is successfully flashed on the SimpleLink™ device before proceeding to the following steps.
- Calibrate for the voltage drop across the wire connecting the DC Power Analyzer to the device (use short and multistrand wire).

Procedure:

1. Make the modifications for each board.
   - CC3120 BoosterPack:
     1. With the USB cable disconnected and both the CC3120 and MSP432 devices turned off, remove jumpers J6 and J8.
     2. Connect the +output of the DC Power Analyzer to the VBAT_CC pin of J6, and use P2.1 for GND as shown in Figure 9.

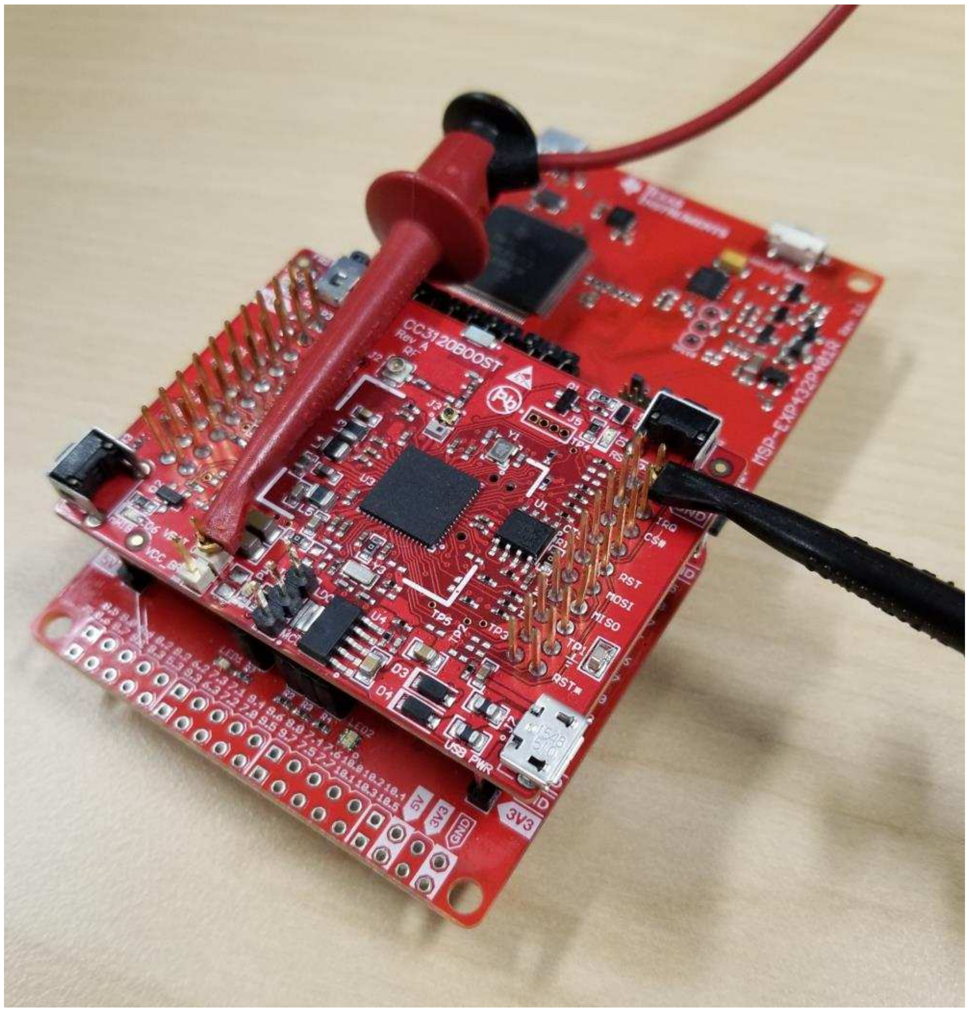


Figure 9. DC Power Analyzer Connection to CC3120 Device

- CC3220 LaunchPad:
  1. With the USB cable disconnected and the CC3220 LaunchPad turned off, remove jumper J19.
  2. Additionally, remove jumpers J14 and J24 to disable the onboard OP AMP and LEDs. This will reduce the total current consumption of the development kit.
  3. Connect the +output of the DC Power Analyzer to J19 and use P2.1 (or J22) for GND, as shown in Figure 10.

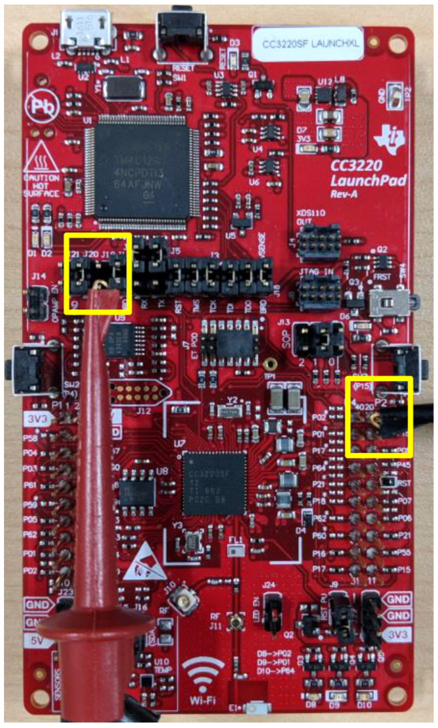


Figure 10. DC Power Analyzer Connection to CC3220 LaunchPad™

2. Connect the SimpleLink™ device to the PC using the USB cable.
3. Execute the code by powering on the device, and measure the current using the sourcing and measuring function of the DC Power Analyzer. For reference, Figure 11 shows the full overview.

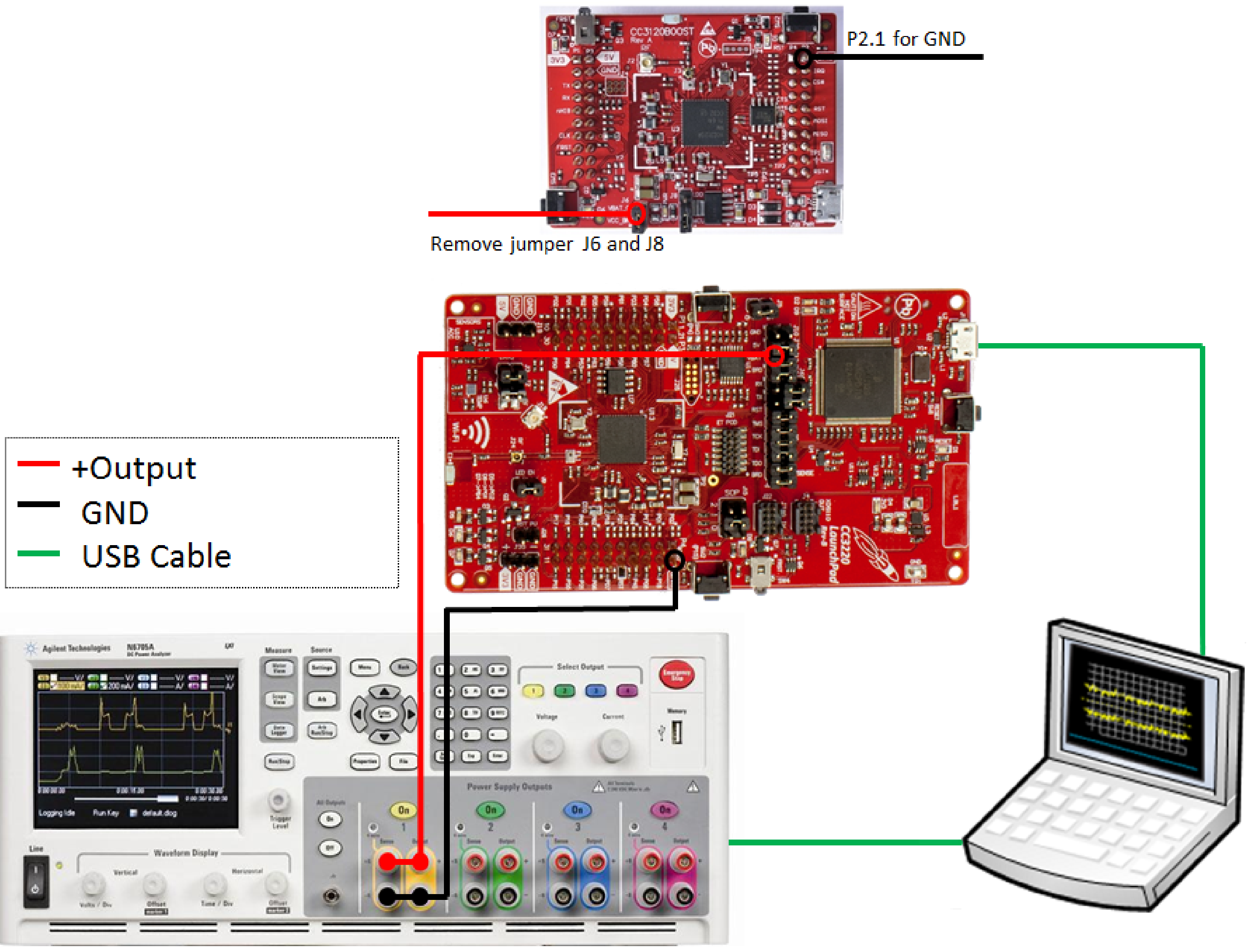
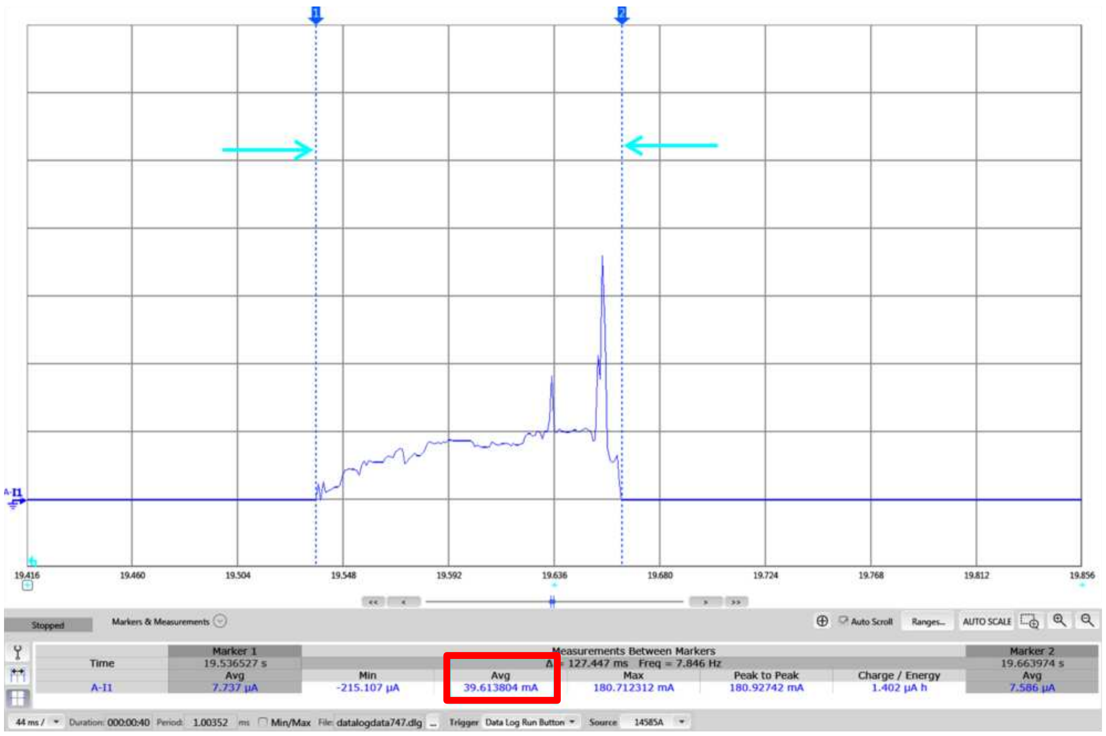


Figure 11. DC Power Analyzer Test Setup

4. When the current measurement completes, position the markers to select the waveform to be calculated. The result should show the average current consumption of the sample period located between the two markers. Figure 12 shows an example.



	Marker 1	Measurements Between Markers					Marker 2
Time	19.536527 s	Δ = 127.447 ms Freq = 7.846 Hz					19.663974 s
	Avg	Min	Avg	Max	Peak to Peak	Charge / Energy	Avg
A-I1	7.737 µA	-215.107 µA	39.613804 mA	180.712312 mA	180.92742 mA	1.402 µA h	7.586 µA

44 ms / Duration: 000:00:40 Period: 1.00352 ms Min/Max File: datalogdata747.dlg Trigger Data Log Run Button Source 14585A

Figure 12. Average Current Measurement With DC Power Analyzer

2.4.2 Measuring Current Consumption With the IMETER-BOOST

Another method to measure current consumption is using a tool that precisely measures and plots current, such as the IMETER-BOOST. This setup is suited for both ultra-low and active current measurements.

Tools needed:

- IMETER-BOOST (or any precision data acquisition device with a graphical user interface (GUI) to visualize and analyze data).
- CC3200-LAUNCHXL or CC3200MODLAUNCHXL
- Short cables to connect the data acquisition (DAQ) tool to the SimpleLink™ device
- PC

NOTE: The IMETER-BOOST must be connected to a CC3200-LAUNCHXL LaunchPad to graphically view the current and voltage measurements using the CC3200 HTTP Server abilities.

Prerequisites and things to consider:

- Ensure the *power_measurement* example code is successfully flashed on the SimpleLink™ device before proceeding to the following steps.
- Calibrate for the voltage drop across the wire connecting the supply to the device (use short and multistrand wire).

Procedure:

1. Make the modifications for each board.
   - CC3120 BoosterPack:
     1. With the USB cable disconnected and both the CC3120 BoosterPack and MSP432 LaunchPad turned off, remove jumper J6.
     2. Connect the IMETER-BOOST across J6, as shown in Figure 13.
     3. Use a jumper wire to connect the GND pins of the SimpleLink™ device and the IMETER-BOOST system.

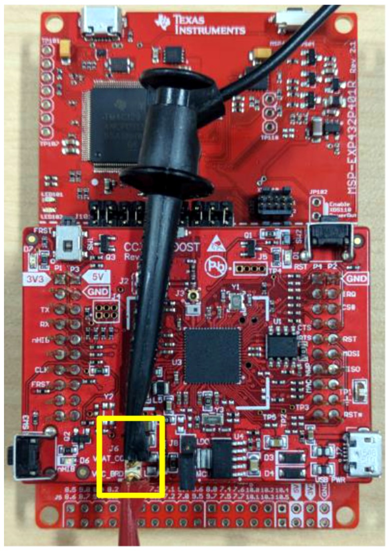


Figure 13. Current Measurement Tool Connection to CC3120 Device

- CC3220 LaunchPad:
  1. With the USB cable disconnected and the CC3220 LaunchPad turned off, remove jumper J19.
  2. Additionally, remove jumpers J14 and J24 to disable the on-board OP AMP and LEDs. This will reduce the total current consumption of the development kit.
  3. Connect the IMETER-BOOST across J19, as shown in Figure 14.
  4. Use a jumper wire to connect the GND pins of the SimpleLink™ device and the IMETER-BOOST system.

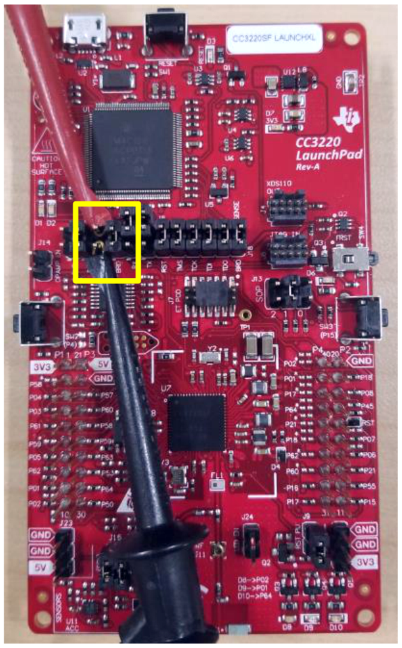


Figure 14. Current Measurement Tool Connection to CC3220SF Device

2. Connect the SimpleLink™ device to the PC using the USB cable.
3. Execute the code and measure the current using the data acquisition tool. For reference, Figure 15 shows the full overview.

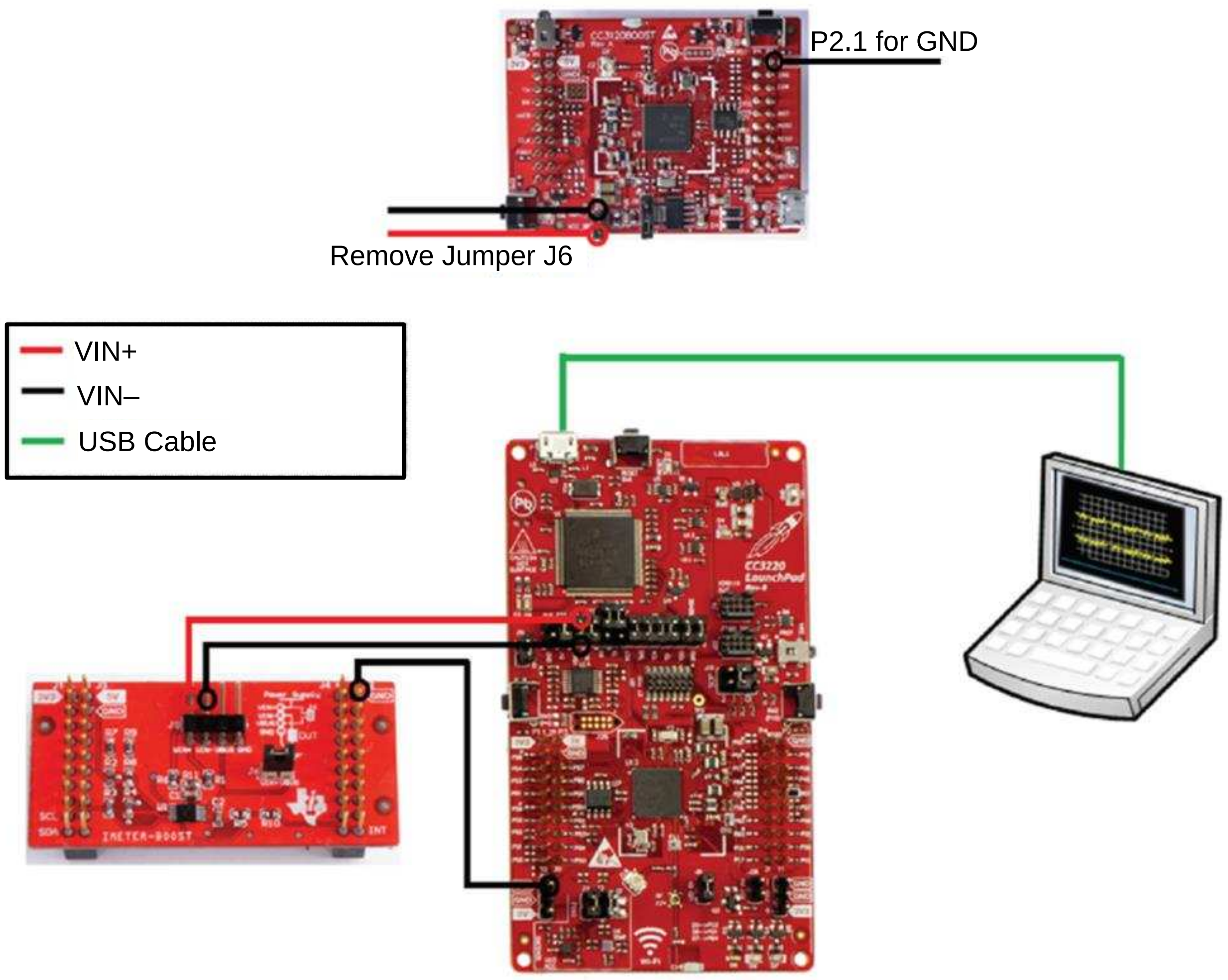
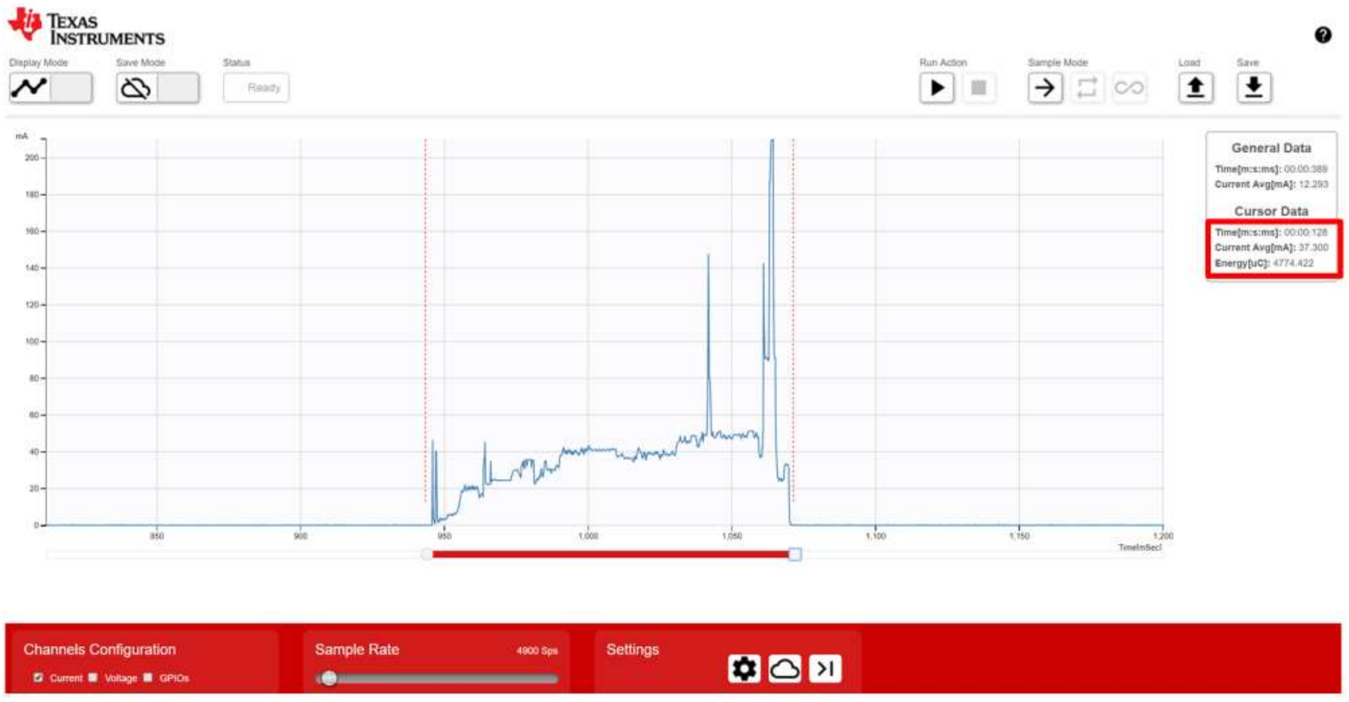


Figure 15. IMETER-BOOST Test Setup

4. When the current measurement completes, position the markers to select the waveform to be calculated. The result should show the average current consumption of the sample period located between the two markers. Figure 16 shows an example.



General Data
Time[m:s:ms]: 00:00:389
Current Avg[mA]: 12.293

Cursor Data
Time[m:s:ms]: 00:00:128
Current Avg[mA]: 37.300
Energy[uC]: 4774.422

Figure 16. Average Current Measurement With IMETER-BOOST Device

2.4.3 Measuring Current Consumption With an Oscilloscope and Current Probe

The most straightforward way to measure current with an oscilloscope is to use a current probe and directly monitor the current going into the system. This setup is suited for dynamic and active current measurements only (currents higher than 10 mA).

Tools needed:

- Oscilloscope
- Current probe with amplifier
- External 3.3-V supply source
- Short cables to connect the 3.3-V external supply to the SimpleLink™ device
- PC

Prerequisites and things to consider:

- Ensure the full bandwidth is selected for the channel where the current probe is connected. This is to ensure you do not miss any glitch or spike while measuring currents.
- Calibrate the current probe before connecting it across the supply line and align the current direction marker (arrow) on the current probe with the direction of the current going through the supply line.
- TI recommends a regulated, DC power supply as opposed to batteries, because this eliminates variables that could stem from using low or defective batteries.
- Check the current limit that you have set for the external supply. A lower current limit may RESET the SimpleLink™ device.
- Ensure the *power_measurement* example code is successfully flashed on the SimpleLink™ device before proceeding to the following steps.
- Calibrate for the voltage drop across the wire connecting the supply to the device (use short and multistrand wire).

Procedure:

1. Make the modifications for each board.
   - CC3120 BoosterPack:
     1. With the USB cable disconnected and both the CC3120 BoosterPack and MSP432 LaunchPad turned off, remove jumper J6 and J8.
     2. Connect the +output of the power supply to the VBAT_CC pin of J6 and use P2.1 for GND, as shown in Figure 9.
   - CC3220 LaunchPad:
     1. With the USB cable disconnected and the CC3220 LaunchPad turned off, remove jumper J19.
     2. Additionally, remove jumpers J14 and J24 to disable the on-board OP AMP and LEDs. This will reduce the total current consumption of the development kit.
     3. Connect the +output of the power supply to J19 and use P2.1 (or J22) for GND, as shown in Figure 10.

2. Connect the current probe to the positive supply wire and ensure that the direction of the current in the wire and the arrow on the current probe align.
3. Connect the SimpleLink™ device to the PC using the USB cable.
4. Execute the code and measure the current using the oscilloscope. For reference, Figure 17 shows the full overview.

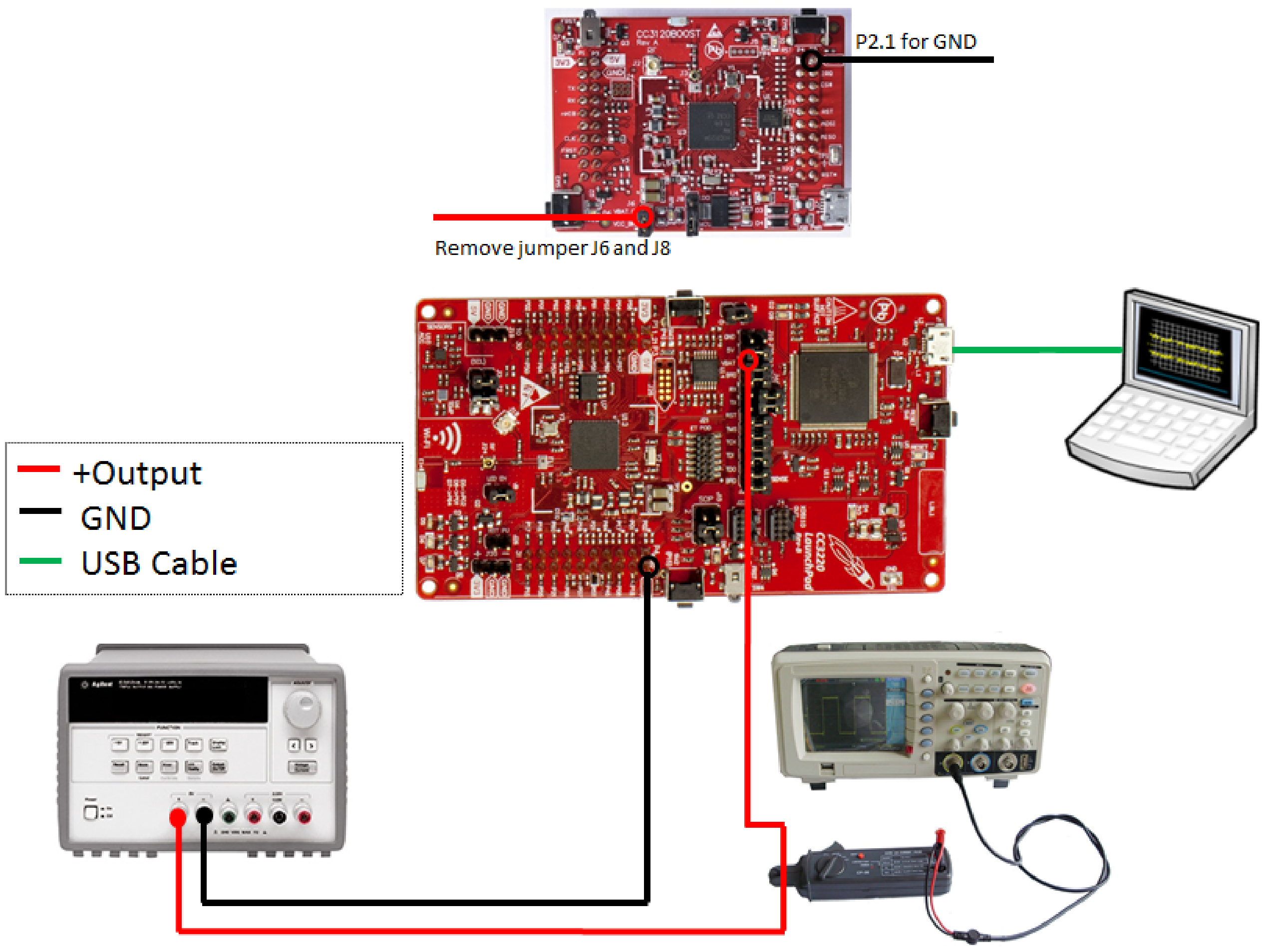


Figure 17. Oscilloscope With Current Probe Test Setup

5. When the current measurement completes, position the markers to select the waveform.

To measure the charge profile, use the *math* function to perform integral over time. The result will be the total profile charge in mA-seconds, or Coulombs, in between the cursors. Figure 18 shows an example.

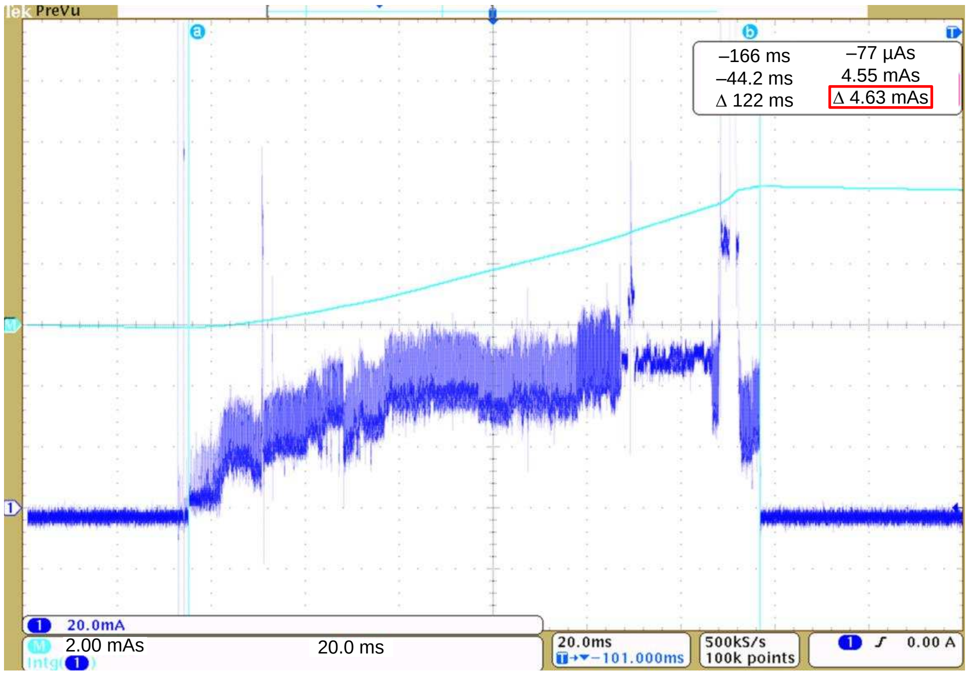


Figure 18. Current Measurement With Current Probe

2.4.4 Measuring Current Consumption With the XDS110 EnergyTrace™ HDR

An alternative method to measuring current consumption is using the EnergyTrace technology included in CCS. EnergyTrace™ technology is an energy-based code analysis tool that measures and displays the energy profile of the application. This setup is suited for both ultra-low and active current measurements.

Tools needed:

- XDS110 JTAG Debug Probe (TMDSEMU110-U)
- XDS110 EnergyTrace High Dynamic Range (HDR) Debug Probe Add-On (TMDSEMU110-ETH)
- 14-pin auxiliary cable (shipped with the TMDSEMU110-U device)
- Breakout board for the auxiliary cable (shipped with TMDSEMU110-U device)
- Jumper wires
- PC

Prerequisites and things to consider:

- The XDS110 EnergyTrace High Dynamic Range (ETHDR) provides an enhanced HDR EnergyTrace capability to the capability that exists on the base XDS110 Debug Probe.
- Connect the XDS110 ETHDR to the XDS110 Debug Probe by stacking the enclosures, see Figure 19.

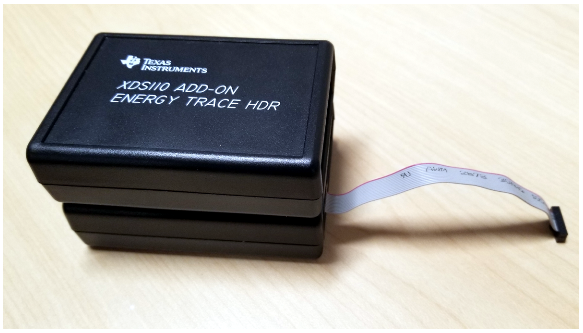
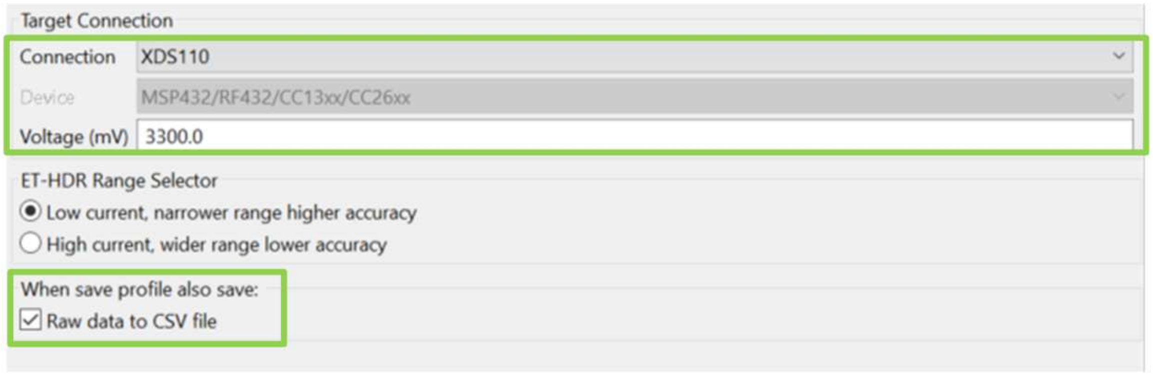


Figure 19. XDS110 Probe and ETHDR System

- The breakout board is only used when measuring current consumption with the CC3120 BoosterPack.
- Ensure the latest version of CCS is installed on the PC.
- Open CCS and go to Window → Preferences → Code Composer Studio → Advanced Tools → EnergyTrace Technology. Under Target Connection, configure Connection to be XDS110 and Voltage to 3300 mV. Additionally, check the box next to Raw data to CSV file. Figure 20 shows the specified fields.



Target Connection
Connection XDS110
Device MSP432/RF432/CC13xx/CC26xx
Voltage (mV) 3300.0

ET-HDR Range Selector
● Low current, narrower range higher accuracy
○ High current, wider range lower accuracy

When save profile also save:
✓ Raw data to CSV file

Figure 20. EnergyTrace™ Settings

- Ensure the *power_measurement* example code is successfully flashed on the SimpleLink™ device before proceeding to the following steps.

Procedure:

1. Make the modifications for each board.
   - CC3120 BoosterPack:
     1. With the USB cable disconnected and both the CC3120 and MSP432 devices turned off, remove jumper J6 and J8.
     2. Connect the jumper wires from VBAT_CC and P2.1 on the BoosterPack to TGTSUPPLYOUT and GND on the breakout board, respectively, as shown in Figure 21 and Figure 22.
     3. Last, connect the 14-pin auxiliary cable to the breakout board.

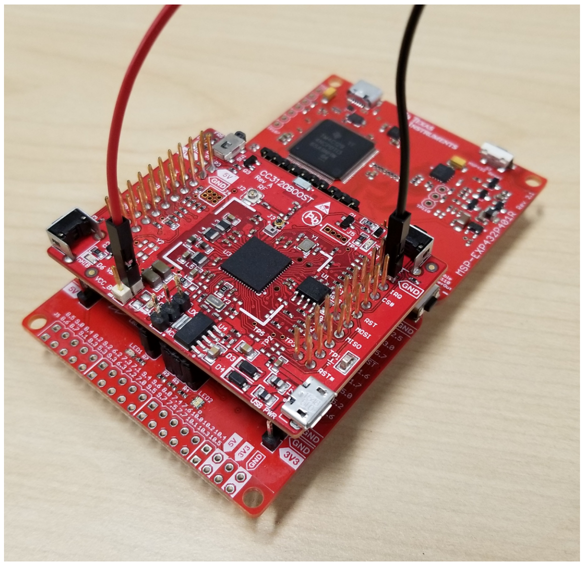


Figure 21. XDS110 ETHDR Connection to CC3120 Device

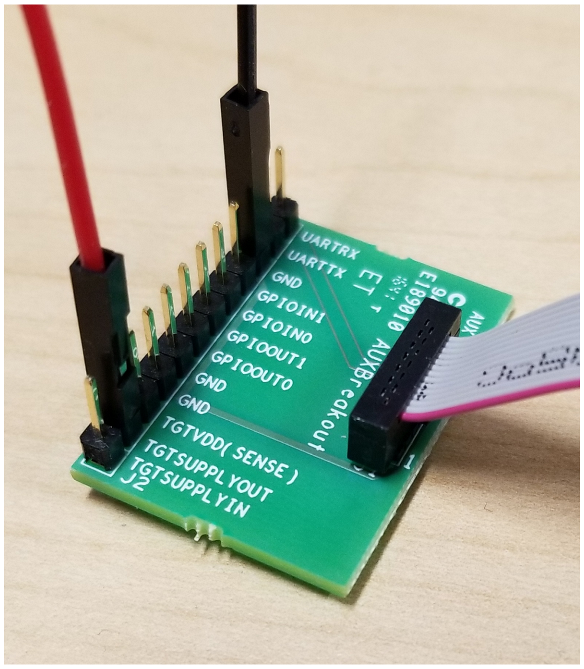


Figure 22. Breakout Board Connection

- CC3220 LaunchPad:
  1. With the USB cable disconnected and the CC3220 LaunchPad turned off, remove the jumpers across the isolation block headers.
  2. Additionally, remove jumpers J14 and J24 to disable the onboard OP AMP and LEDs. This will reduce the total current consumption of the development kit.
  3. Connect the 14-pin auxiliary cable to J7, as shown in Figure 23.

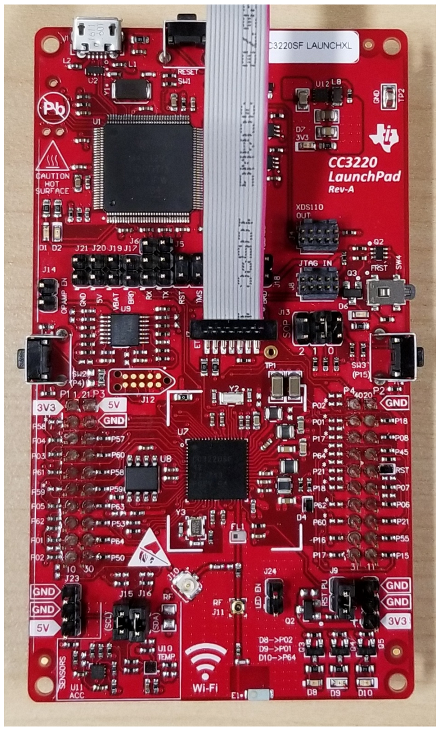


Figure 23. XDS110 ETHDR Connection to CC3220SF Device

2. Connect the opposite end of the 14-pin auxiliary cable to the AUX port of the XDS110 Debug Probe, as shown in Figure 24. Then connect the XDS110 Debug Probe to the PC using the USB cable.

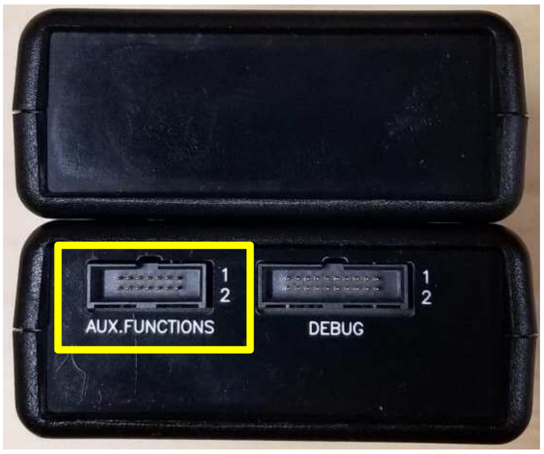


Figure 24. XDS110 Auxiliary Interface

3. Start the EnergyTrace tool by clicking on the blue icon with the EnergyTrace logo from the CCS Toolbar, as shown in Figure 25.

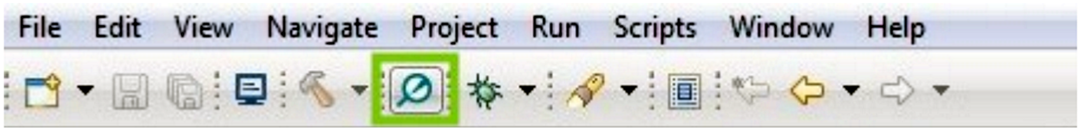


Figure 25. EnergyTrace™ and CCS Toolbar

4. Measure the current consumption.
   - CC3120 BoosterPack:
     1. Start the EnergyTrace capture by clicking on the green Start icon in the EnergyTrace Technology window.
     2. Then, connect the MSP432 LaunchPad to the PC using a second USB cable to execute the code.
   - CC3220 LaunchPad:
     1. Start the EnergyTrace capture and execute the code by clicking on the green Start icon in the EnergyTrace Technology window.

NOTE:

- The duration of the capture can be changed using the stopwatch icon in the main view.
- To stop trace collection while a trace capture is in progress, click the Pause icon that replaces the Start icon.

5. For reference, Figure 26 and Figure 27 show the test setup of the SimpleLink™ devices and XDS110 system.

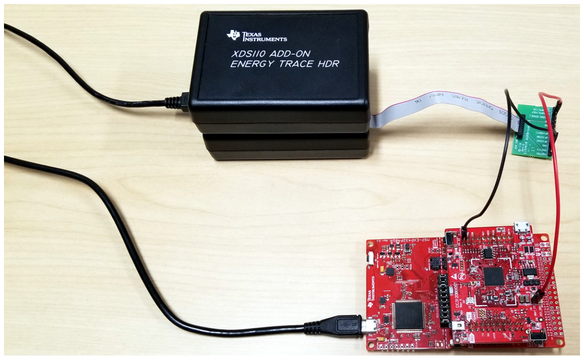


Figure 26. CC3120 Device Test Setup With XDS110 System

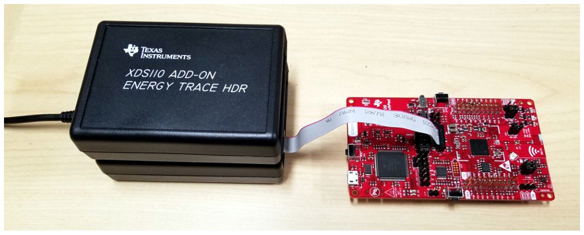


Figure 27. CC3220 Device Test Setup With XDS110 System

When the measurement completes, the average current consumption value can be viewed in the EnergyTrace Technology window. The measured current waveform can be viewed in the Current window, as shown in Figure 28. EnergyTrace does not have markers that can be used to select specific time frames of the measurement. Alternatively, the energy profile can be saved as a CSV file by clicking on the Save icon in the EnergyTrace Technology window.

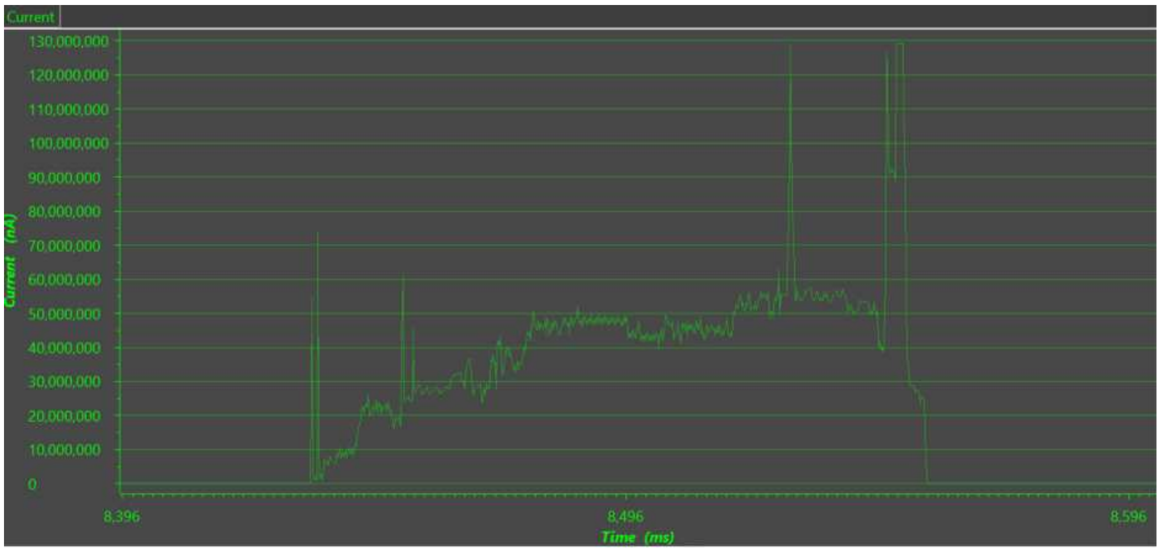


Figure 28. Average Current Measurement With EnergyTrace™

2.5 *Testing and Results*

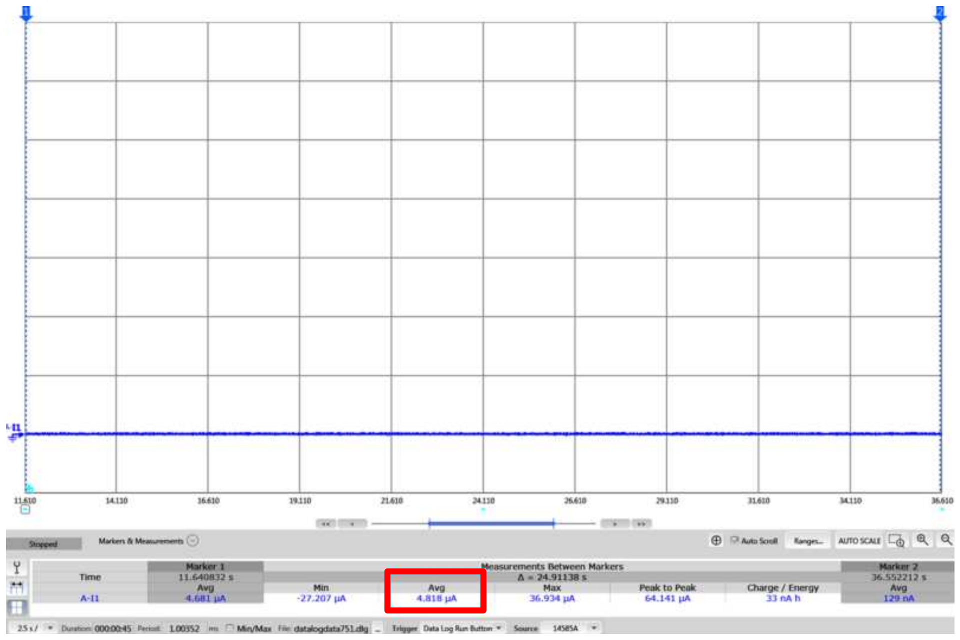
The following sections list the test results.

2.5.1 Low-Power Mode Current Measurements

The following sections describe current measurements of the CC3120 BoosterPack and CC3220SF LaunchPad when placed in low-power modes. For precise measurements, a Keysight N6705B DC power analyzer was used. The CC3120 BoosterPack and CC3220SF LaunchPad were both powered at 3.3 V.

2.5.1.1 Current Measurement in Hibernate Mode

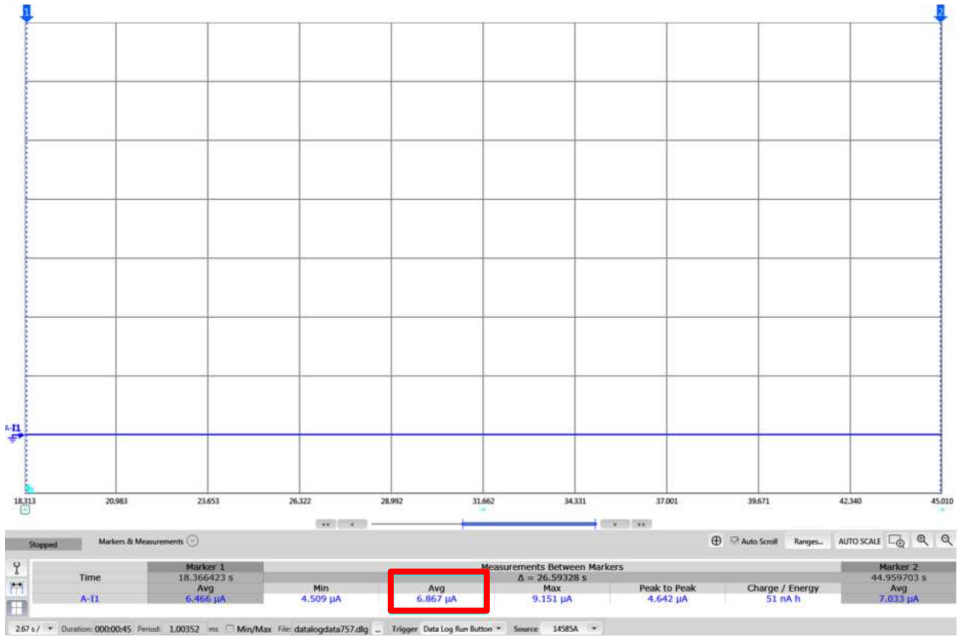
Figure 29 and Figure 30 show the current draw when the CC3120 and CC3220SF devices are in Hibernate mode. The average current measured was 4.818 µA and 6.867 µA, respectively.



	Marker 1	Measurements Between Markers				Marker 2
Time	11.640832 s	Δ = 24.91138 s				36.552212 s
	Avg	Min	Avg	Max	Peak to Peak	Charge / Energy	Avg
A-I1	4.681 µA	-27.207 µA	4.818 µA	36.934 µA	64.141 µA	33 nA h	129 nA

2.5 s / Duration: 000:00:45 Period: 1.00352 ms Min/Max File: datalogdata751.dlg Trigger: Data Log Run Button Source: 14585A

Figure 29. CC3120 Device in Hibernate Mode



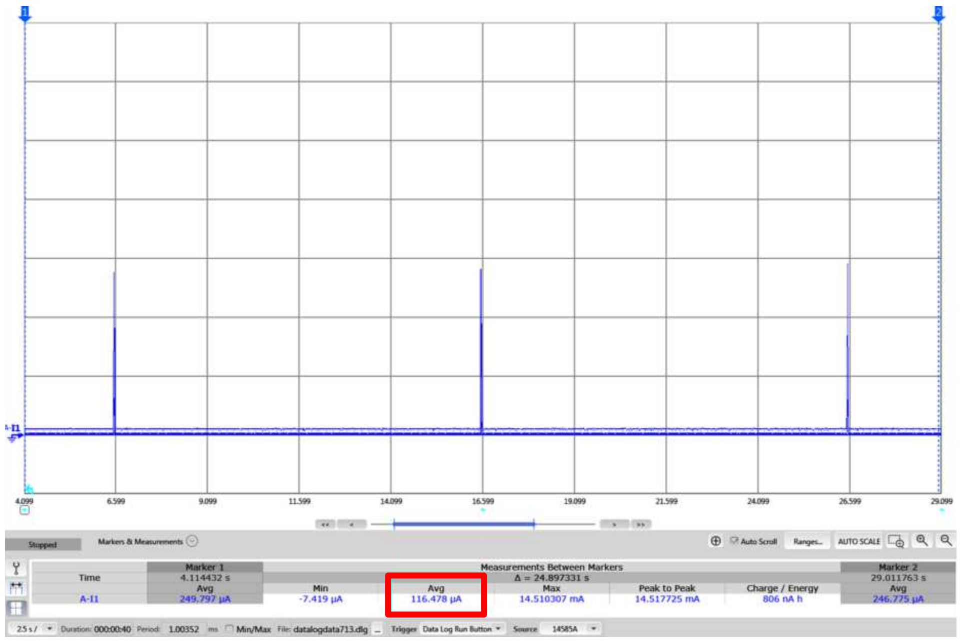
	Marker 1	Measurements Between Markers					Marker 2
Time	18.366423 s	Δ = 26.59328 s					44.959703 s
	Avg	Min	Avg	Max	Peak to Peak	Charge / Energy	Avg
A-I1	6.466 µA	4.509 µA	6.867 µA	9.151 µA	4.642 µA	51 nA h	7.033 µA

2.67 s / Duration: 000:00:45 Period: 1.00352 ms Min/Max File: datalogdata757.dlg Trigger: Data Log Run Button Source: 14585A

Figure 30. CC3220SF Device in Hibernate Mode

2.5.1.2 Current Measurement in LPDS Mode

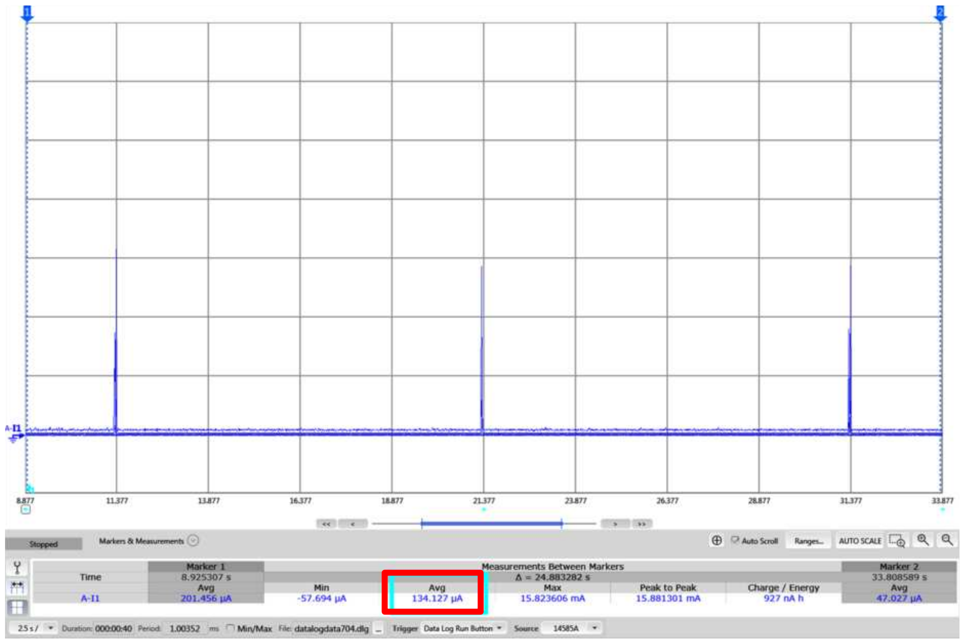
Figure 31 and Figure 32 show the current draw when the CC3120 and CC3220SF devices are in LPDS. The average current measured was 116.478 µA and 134.127 µA, respectively. sl_Start(), called on line 1540 of power_measure.c, configures the internal register that enables the NWP which manages its own sleep/wakeup events. The current spikes that can be observed from the following measurements are caused by the NWP periodically waking up to output logs.



	Marker 1	Measurements Between Markers					Marker 2
Time	4.114432 s	Δ = 24.897331 s					29.011763 s
	Avg	Min	Avg	Max	Peak to Peak	Charge / Energy	Avg
A-I1	249.797 µA	-7.419 µA	116.478 µA	14.510307 mA	14.517725 mA	806 nA h	246.775 µA

2.5 s / Duration: 000:00:40 Period: 1.00352 ms Min/Max File: datalogdata713.dlg Trigger Data Log Run Button Source 14585A

Figure 31. CC3120 Device in LPDS Mode



	Marker 1	Measurements Between Markers					Marker 2
Time	8.925307 s	Δ = 24.883282 s					33.808589 s
	Avg	Min	Avg	Max	Peak to Peak	Charge / Energy	Avg
A-I1	201.456 µA	-57.694 µA	134.127 µA	15.823606 mA	15.881301 mA	927 nA h	47.027 µA

2.5 s / Duration: 000:00:40 Period: 1.00352 ms Min/Max File: datalogdata704.dlg Trigger Data Log Run Button Source 14585A

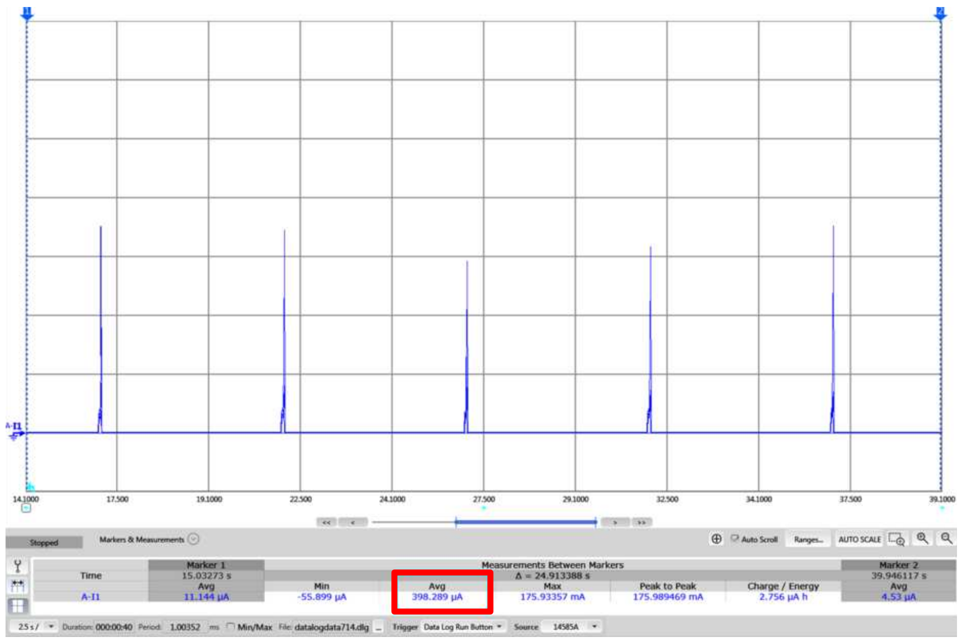
Figure 32. CC3220SF Device in LPDS Mode

2.5.2 Test Case Current Measurements

The following sections describe current measurements of the CC3120 BoosterPack and CC3220SF LaunchPad when placed in different test cases. For precise measurements, a Keysight N6705B DC power analyzer was used. The CC3120 BoosterPack and CC3220SF LaunchPad were both powered at 3.3 V.

2.5.2.1 Current Measurement in Transceiver Mode

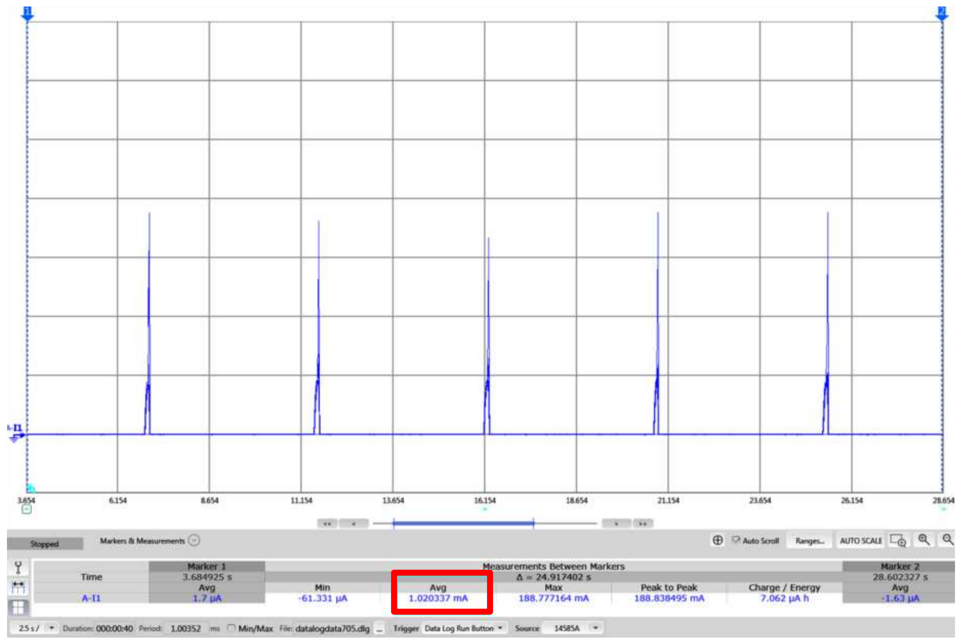
Figure 33 and Figure 34 show the current draw when the CC3120 and CC3220SF devices are placed in Transceiver mode. The average current measured was 398.289 µA and 1.02 mA, respectively.



	Marker 1	Measurements Between Markers					Marker 2
Time	15.03273 s	Δ = 24.913388 s					39.946117 s
	Avg	Min	Avg	Max	Peak to Peak	Charge / Energy	Avg
A-I1	11.144 µA	-55.899 µA	398.289 µA	175.93357 mA	175.989469 mA	2.756 µA h	4.53 µA

2.5 s / Duration: 000:00:40 Period: 1.00352 ms Min/Max File: datalogdata714.dlg Trigger Data Log Run Button Source 14585A

Figure 33. CC3120 Device in Transceiver Mode

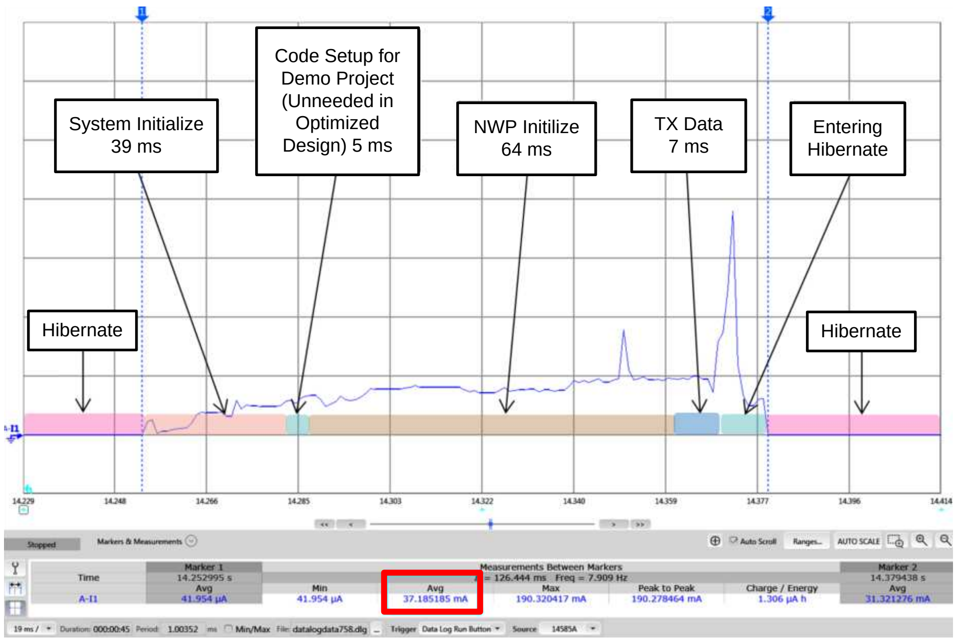


	Marker 1	Measurements Between Markers					Marker 2
Time	3.684925 s	Δ = 24.917402 s					28.602327 s
	Avg	Min	Avg	Max	Peak to Peak	Charge / Energy	Avg
A-I1	1.7 µA	-61.331 µA	1.020337 mA	188.777164 mA	188.838495 mA	7.062 µA h	-1.63 µA

2.5 s / Duration: 000:00:40 Period: 1.00352 ms Min/Max File: datalogdata705.dlg Trigger Data Log Run Button Source 14585A

Figure 34. CC3220SF Device in Transceiver Mode

Figure 35 shows a close-up view of the waveform when the CC3220SF device is in Transceiver mode, waking up from Hibernate mode, sending data, and then entering Hibernate mode again. The average current consumption of this transaction was 37.19 mA.



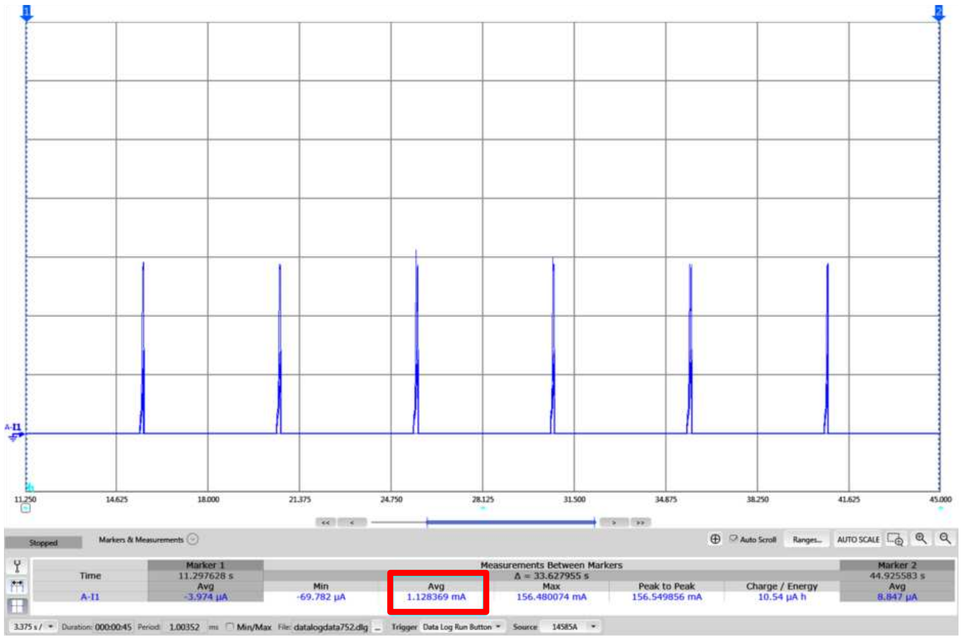
	Marker 1	Measurements Between Markers					Marker 2
Time	14.252995 s	Δ = 126.444 ms Freq = 7.909 Hz					14.379438 s
	Avg	Min	Avg	Max	Peak to Peak	Charge / Energy	Avg
A-I1	41.954 µA	41.954 µA	37.185185 mA	190.320417 mA	190.278464 mA	1.306 µA h	31.321276 mA

19 ms / Duration: 000:00:45 Period: 1.00352 ms Min/Max File: datalogdata758.dlg Trigger Data Log Run Button Source 14585A

Figure 35. Transceiver Mode Close-Up View

2.5.2.2 Current Measurement in Intermittently Mode

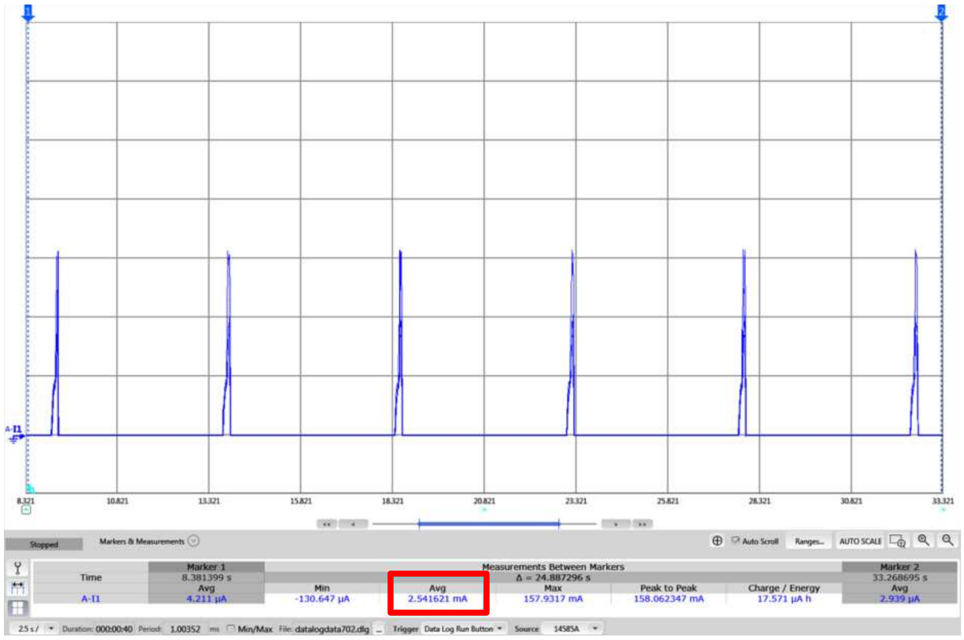
Figure 36 and Figure 37 show the current draw when the CC3120 and CC3220SF devices are placed in Intermittently Connected mode. The average current measured was 1.13 mA and 2.54 mA, respectively.



	Marker 1	Measurements Between Markers					Marker 2
Time	11.297628 s	Δ = 33.627955 s					44.925583 s
	Avg	Min	Avg	Max	Peak to Peak	Charge / Energy	Avg
A-I1	-3.974 µA	-69.782 µA	1.128369 mA	156.480074 mA	156.549856 mA	10.54 µA h	8.847 µA

3.375 s / Duration: 000:00:45 Period: 1.00352 ms Min/Max File: datalogdata752.dlg Trigger Data Log Run Button Source 14585A

Figure 36. CC3120 Device Intermittently Connected Mode

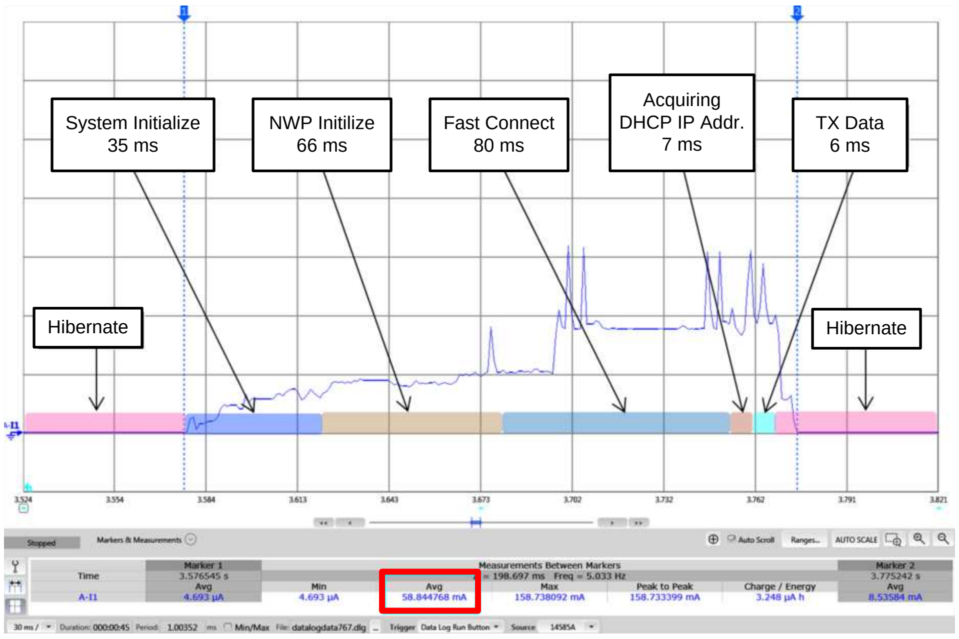


	Marker 1	Measurements Between Markers					Marker 2
Time	8.381399 s	Δ = 24.887296 s					33.268695 s
	Avg	Min	Avg	Max	Peak to Peak	Charge / Energy	Avg
A-I1	4.211 µA	-130.647 µA	2.541621 mA	157.9317 mA	158.062347 mA	17.571 µA h	2.939 µA

2.5 s / Duration: 000:00:40 Period: 1.00352 ms Min/Max File: datalogdata702.dlg Trigger Data Log Run Button Source 14585A

Figure 37. CC3220SF Device in Intermittently Connected Mode

Figure 38 shows a close-up view of the waveform, where the CC3220SF device is in Intermittently Connected Mode. After it wakes up from Hibernate, it sends data and enters Hibernate again.



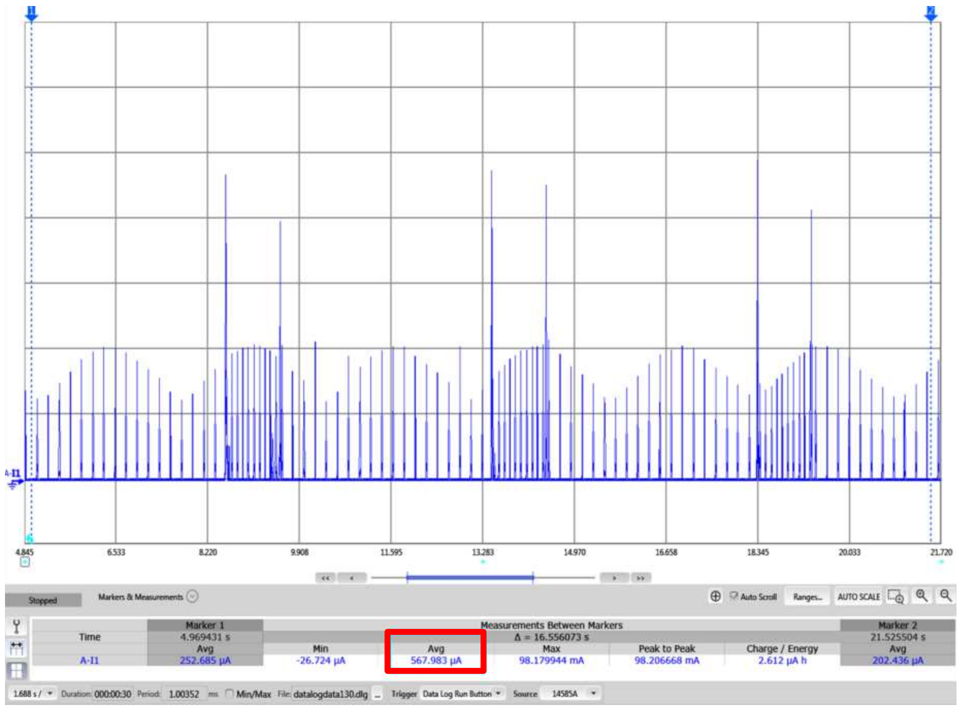
	Marker 1	Measurements Between Markers					Marker 2
Time	3.576545 s	Δ = 198.697 ms Freq = 5.033 Hz					3.775242 s
	Avg	Min	Avg	Max	Peak to Peak	Charge / Energy	Avg
A-I1	4.693 µA	4.693 µA	58.844768 mA	158.738092 mA	158.733399 mA	3.248 µA h	8.53584 mA

30 ms / Duration: 000:00:45 Period: 1.00352 ms Min/Max File: datalogdata767.dlg Trigger: Data Log Run Button Source: 14585A

Figure 38. Intermittently Connected Mode Close-Up View

2.5.2.3 Current Measurement in Always Connected Mode

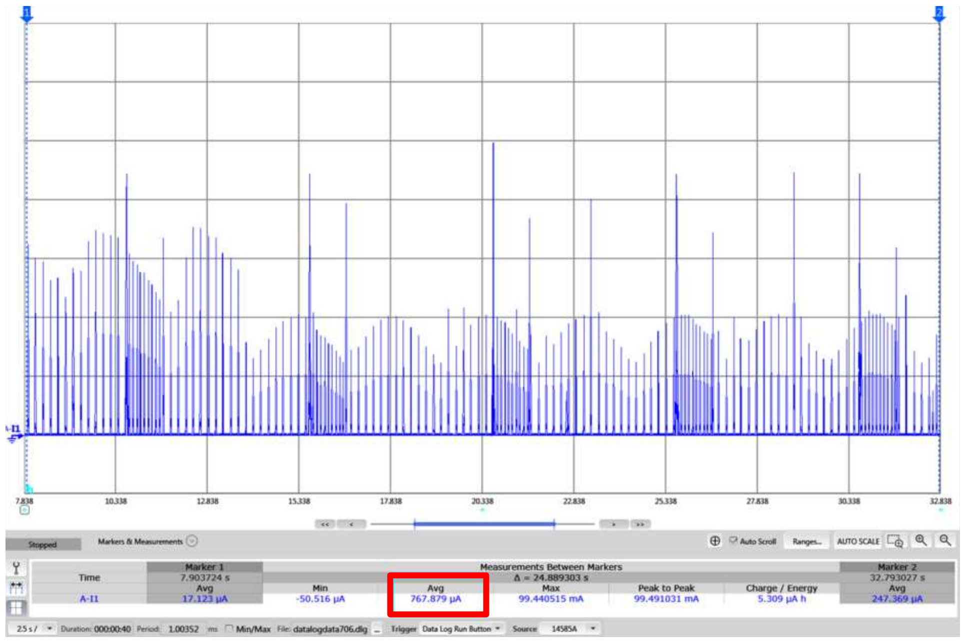
Figure 39 and Figure 40 show the current draw when the CC3120 and CC3220SF devices are placed in Intermittently Connected Mode. The average current measured was 567.98 µA and 767.87 µA, respectively.



	Marker 1	Measurements Between Markers					Marker 2
Time	4.969431 s	Δ = 16.556073 s					21.525504 s
	Avg	Min	Avg	Max	Peak to Peak	Charge / Energy	Avg
A-I1	252.685 µA	-26.724 µA	567.983 µA	98.179944 mA	98.206668 mA	2.612 µA h	202.436 µA

1.688 s / Duration: 000:00:30 Period: 1.00352 ms Min/Max File: datalogdata130.dlg Trigger Data Log Run Button Source 14585A

Figure 39. CC3120 Device in Always Connected Mode (UDP)

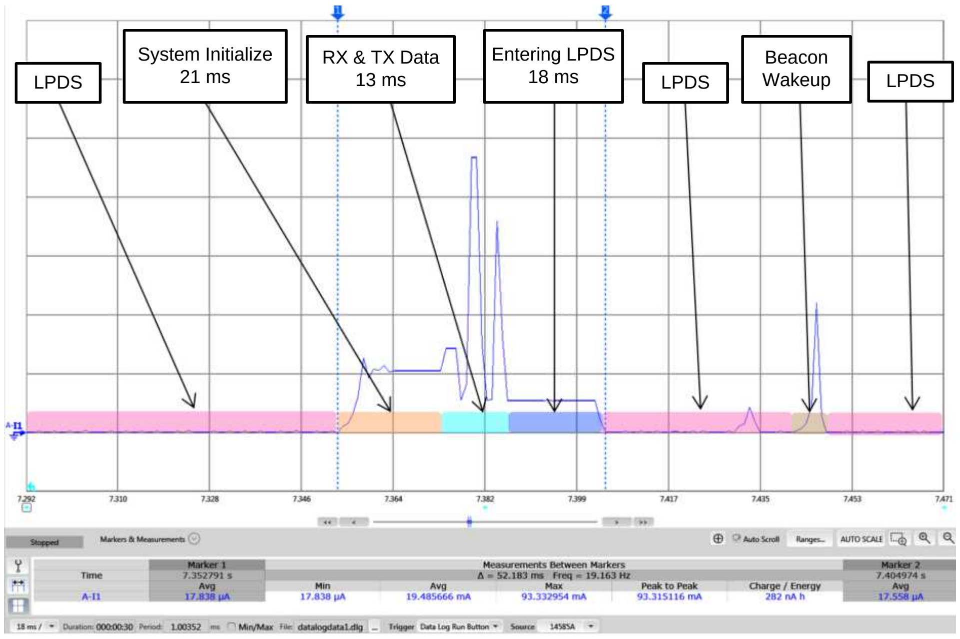


	Marker 1	Measurements Between Markers					Marker 2
Time	7.903724 s	Δ = 24.889303 s					32.793027 s
	Avg	Min	Avg	Max	Peak to Peak	Charge / Energy	Avg
A-I1	17.123 µA	-50.516 µA	767.879 µA	99.440515 mA	99.491031 mA	5.309 µA h	247.369 µA

2.5 s / Duration: 000:00:40 Period: 1.00352 ms Min/Max File: datalogdata706.dlg Trigger Data Log Run Button Source 14585A

Figure 40. CC3220SF Device in Always Connected Mode (UDP)

Figure 41 shows a close-up view of the waveform, where theCC3220SF device is in Always Connected Mode. After it wakes up from LPDS mode, it receives and sends data and enters LPDS again.



	Marker 1	Measurements Between Markers					Marker 2
Time	7.352791 s	Δ = 52.183 ms Freq = 19.163 Hz					7.404974 s
	Avg	Min	Avg	Max	Peak to Peak	Charge / Energy	Avg
A-I1	17.838 µA	17.838 µA	19.485666 mA	93.332954 mA	93.315116 mA	282 nA h	17.558 µA

18 ms / Duration: 000:00:30 Period: 1.00352 ms Min/Max File: datalogdata1.dlg Trigger Data Log Run Button Source 1458SA

Figure 41. Always Connected Close-Up View

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from May 25, 2018 to November 2, 2018	Page
• Changed description of Active in Section 1.4	6
• Added second paragraph in Section 1.5	6
• Added Table 2. Subsequent tables renumbered	6
• Deleted footnote from Table 3	9
• Changed description of Always On in Section 1.7	9
• Changed Notes after Table 4	12
• Changed Notes after Table 7	14