Application Report
SWRA659A–March 2020–Revised June 2020

CC3x00 Power Management Optimization and Measurements

Jesus Pintado

ABSTRACT

The CC3X00 SimpleLink™ Power Management Measurement application provides users the ability to configure the device in various low power uses cases, for the purposes of current consumption measurements.

Contents

List of Figures

List of Tables

Trademarks

1 Introduction

The CC3X00 SimpleLink Power Management Measurement application provides users the ability to configure the device in various low power modes and profiles, for the purposes of current consumption measurements. The user can choose to work in the Interactive (simple) mode or to use the advanced mode, where they can change the define values in the source code in order to control the behavior of the application. The application has recommended guidelines on how to optimize the power consumption for the different power profiles.

The system's various basic power modes are first explained. These are the building blocks of all power management profiles.

- Hibernate
- Low power deep sleep (LPDS)
- Active modes (Rx or Tx)

The various power profiles are the focus of this document:

- The various power profiles are the focus of this document:
- Intermittently connected
- Transceiver mode

NOTE: The focus is on periodic/repetitive events because one-time events, no matter how high they peak, do not need to be taken into account, since their influence is negligible for longer periods of activity.

1.1 Getting Started

1.1.1 For CC3100

All instructions in this application report are under the assumptions that the user already has a working environment of CC3100 BoosterPack attached to the MSP430 LaunchPad, and that he installed all the relevant software packages. Otherwise, see the *CC3100 SimpleLink Wi-Fi and IoT Solution Getting Started Guide*, before proceeding.

1.1.2 For CC3200

All Instructions in this document are under the assumptions that the user already has a working environment of CC3200 LaunchPad, and installed all the relevant software packages. Otherwise, see the *CC3200 SimpleLink Wi-Fi and IoT Solution w/ MCU LaunchPad Getting Started Guide*, before proceeding.

2 Prerequisites

2.1 For CC3100

Hardware

- CC3100 Boosterpack Rev 3.3 or above
- MSP430F5529LP
  - Oscilloscope with differential probe or current probe
  - Digital multimeter (capable of measuring down to 1 µA)

Software

- CC3100 SDK, which contains
  - CC3100 ServicePack
  - CC3100 SDK
  - CC3100 Power Management Optimizations and Measurements source code

The power management application is released as a stand-alone CC3200 application, which is integrated within the CC3200 SDK folder during installation. Import it into your environment (code composer /IAR), as any other project described in the *CC3200 SimpleLink Wi-Fi and IoT Solution w/ MCU LaunchPad Getting Started Guide*.

3 Basic System Power Modes for CC3X00

The following section describes the CC3X00 active and low power static modes. The CC3X00 devices contain only the Networking sub-system and is self-contained in terms of power optimization. The following system modes are defined, and could be measured using TI's EVM boards. The main three building blocks of most power related application profiles are:

- Hibernate
- Low Power Deep Sleep (LPDS)
- Active: Rx and Tx

More details can be found at: *SimpleLink™ CC3100/CC3200 Wi-Fi Internet-on-a-chip™ Networking Sub-system Power Management*

3.1 Hibernate State

Hibernate mode represents the lowest power state of the device. This mode is entered when the MCU subsystem requests the power management unit to shut off the voltage source itself. In hibernate, all the voltage sources, like DCDC or LDOs, within the power management unit are shut off. Very few logic, which works directly on battery power, is ON and they work on 32 KHz clock. Wakeup sources to exit this sleep mode can be a toggle event on the device's general purpose IO (GPIO) or based on timer expiry. In order to measure the static hibernate current consumption, the user needs to set the global use case variable to HIBERNATE_MEASURE.

The application enables the user to enter a constant Hibernate state for an easy measurement of the power consumption of this state. To do so, configure g_ActiveUseCase to "HIBERNATE_MEASURE" in the PM management benchmark code.

```
150 //*****************************************************************************
151 // user defined //
152 signed short   g_ActiveUseCase  = HIBERNATE_MEASURE; //options -->  HIBERNATE_MEASURE; SLEEP_MEASURE;
153 unsigned char  g_SocketType     = UDP_SOCKET; //option --> RAW_SOCKET; UDP_SOCKET; TCP_SOCKET; SEC_TCI
154 unsigned char  g_IpV4Option     = STATIC_IP; //option --> STATIC_IP; DHCP;
155 unsigned char  g_CcaBypass      = 1; // default bypass, if CCA is required use the value 0;
156
```

Figure 1. Hibernate State

To do so, configure g_ActiveUseCase to "SLEEP_MEASURE" in the PM management benchmark code.

3.1.1 Measurement Tool

Follow the steps mentioned in the *Static Current Measurements* section in order to record the current for hibernate mode.

3.1.2 Expected Results

The expected power consumption numbers are described in the CC3100 data sheet and CC3200 data sheet.

3.2 LPDS State

To enter into this mode each subsystem processor requests the clock management unit for shutting off their subsystem. When both the subsystems request for this mode, Clock management unit will turn off the PLL, 40 Mhz xtal, and the power management unit will shut off the power to each subsystem and scale down the voltage of always on domain to 0.9 V. Active logic in this mode will work at 32 KHz xtal. There is an option of retaining the memory content for each subsystem or not. Wakeup sources to exit this sleep mode can be a toggle event on the device's general purpose IO (GPIO) or based on timer expiry. In this mode, power saving happens because of turning off of power domain, turning off of PLL, turning off of 40 MHz xtal, memory not retained if opted, and voltage scale down. In order to measure the static Low Power Deep sleep (LPDS) current consumption, the user needs to set the global use case variable to SLEEP_MEASURE.

The application enables the user to enter a constant Sleep mode (no traffic) to enable an easy way to measure the power consumption of this state.

```
150  //*****************************************************************************
151  // user defined //
152  signed short   g_ActiveUseCase  = SLEEP_MEASURE; //options -->  HIBERNATE_MEASURE; SLEEP_MEASURE;
153  unsigned char  g_SocketType     = UDP_SOCKET; //option --> RAW_SOCKET; UDP_SOCKET; TCP_SOCKET; SE(
154  unsigned char  g_IpV4Option     = STATIC_IP; //option --> STATIC_IP; DHCP;
155  unsigned char  g_CcaBypass      = 1; // default bypass, if CCA is required use the value 0;
```

Figure 2. CC3200 to Static LPDS Current Measure

To do so, configure g_ActiveUseCase to "SLEEP_MEASURE" in the PM management benchmark code.

3.2.1 Measurement Tool

Follow the steps mentioned in the *Static Current Measurements*, in order to record the LPDS current.

3.2.2 Expected Results

The expected power consumption numbers are described in the CC3100 data sheet and CC3200 data sheet.

3.2.3 Active (Rx and Tx) States

The device is fully active, voltage levels are at their operational value, and all clocks are ticking. At least one block (MCU/NWP/Wi-Fi) is running. The two main active modes are transmitting (TX) and receiving (RX). The Tx and Rx active currents vary based on the channel, packets type, and so forth.

3.2.3.1 Configuration Parameter

In order to individually measure the Tx and Rx current consumptions, see the Appendix section of Using radio tool for active modes.

Follow the steps mentioned in the *Current Measurement for Profiles and Active States* section, in order to measure the average current consumption, for each of the active mode, and power profile use cases.

3.2.3.2 Expected Results

The expected power consumption numbers are described in the CC3100 data sheet and CC3200 data sheet.

4 Power Profiles Use Cases

The Power Profiles use cases combine various power state to emulate the behavior of a real application in an end product. This section describes how to emulate, optimize, and measure such profiles. We will focus on these three main profiles:

- Idle Connected
- Intermittently Connected
- Transceiver Mode

End product's applications have various power requirements, which drive the duration of the active states and inactive periods. The system's latency, or the response time, is another common requirement, which focuses on how fast a device can wake up from inactive mode and be fully functional. Thus, it is important to architect multiple low power inactive modes depending upon all the possible use cases of the end product. In other words, the low power mode used in the system is determined by the end product's application properties.

Follow the steps mentioned in the *Current Measurement for Profiles and Active States* section, in order to measure the average current consumption, for each of these profiles.

4.1 Use Case 1: Always Connected

The always connected profile deals with situations where the device must stay connected to the access point (AP) at any time. Staying connected may cause a high power consumption due to periodic beacon functionality. In this profile, the system will enter LPDS mode between wakeups for activity since the system state needs to be kept and low latency is required.

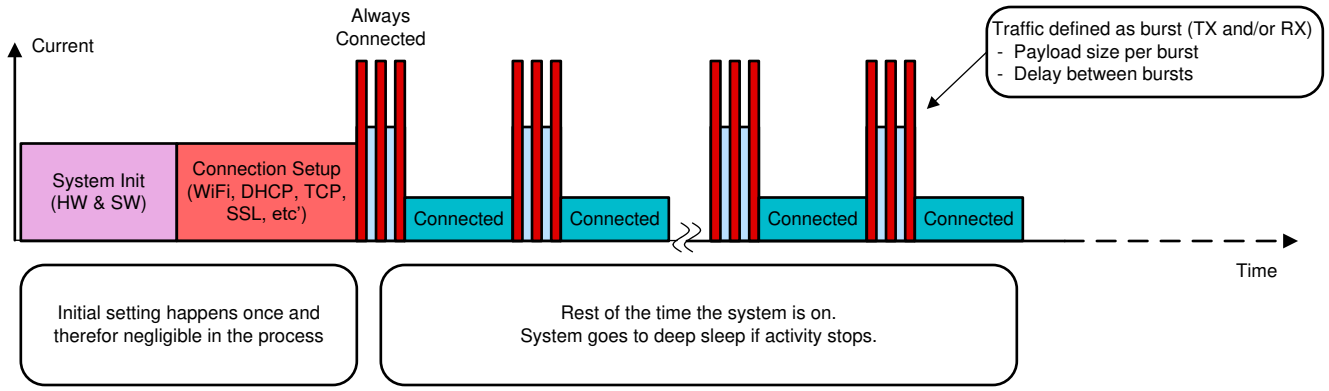


Figure 3. Use Case 1: Always Connected

To optimize your use case please implement below configurations:

- LSI (Long Sleep Intervals) – A configuration that enable to wake the device up only on every n-th beacon thus providing longer Sleep periods.
  - Implementation: Use the sl_WlanPolicySet and configure SL_POLICY_PM to LSI. Can be added to the “configureSimpleLinkToDefaultState” function” in the main.c file.
- mDNS - Upon connection to an AP, NWP automatically starts advertising itself by sending mDNS packets. Stop the mDNS feature when it is not required by the system application for power optimization.
  - Implementation: Stopping the mDNS feature is done using the sl_NetAppStop API, and can be done once as an indication is stored in NVMEM. Can be updated in the “configureSimpleLinkToDefaultState” function” in the main.c file.

A good representation of a code for such case is:

```
//Configurations - This section is done once
sl_start(0,0,0);
sl_WlanPolicySet(...); // configure the time interval between wakeups
sl_socket();
sl_SetSocketOpt();     // configure UDP/TCP Secured or not
sl_bind();

// Sending/Receiving Data - Done when packet need to be sent or received
while (1) {
    sl_Send();         // can be UDP/TCP and/or secure connection
    sl_Recv();         // can be UDP/TCP and/or secure connection
    Delay();
}
```

4.2 Configuring Options

There is one define value that configures the behavior of the device in this case:

- LSI_SLEEP_DURATION_IN_MSEC – define the sleep period between each wake-up for beacon reception.
  - Example: "#define LSI_SLEEP_DURATION_IN_MSEC 500" will set it to 500 milliseconds.

A detailed configuration example for always connected use case is shown in the always connected configuration example figure; the relevant settings are emphasized.

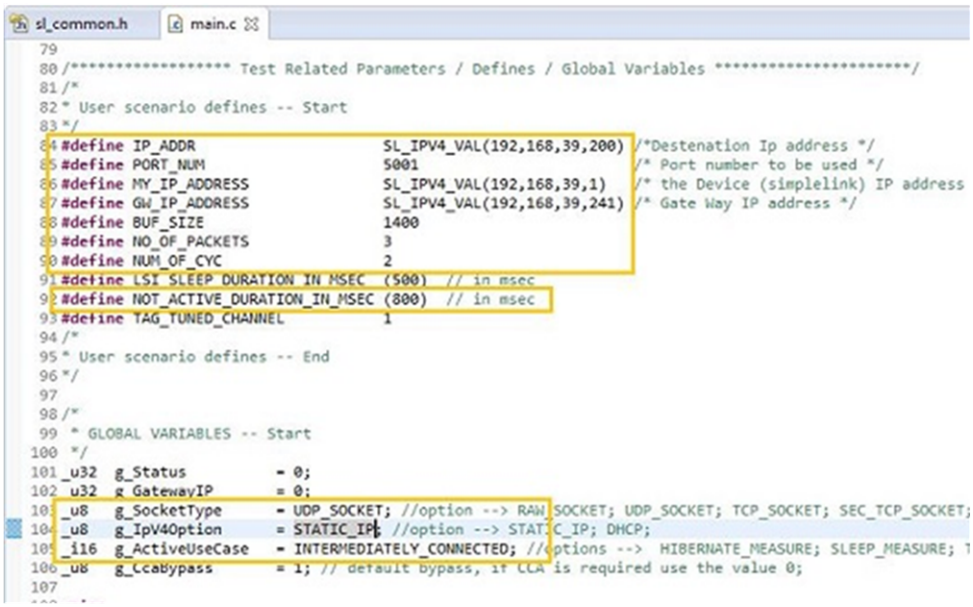


Figure 4. PM Always Connected config Example

4.2.1 Expected Results

Table 1. Expected Results

Parameter	Description
Setup and scenario info	The AP connection security is open. Device is in Idle profile (beacons receive only). Beacon interval is 102 mS
Average current consumption (steady-state)	0.695 mA
Electric charge consumption of one active cycle	0.05 mC (one Beacon reception)
Remarks	By default the device is listening on a UDP socket for incoming packets. The user can configure TCP with or without SSL socket as well. **Note:** The option with SSL sockets requires a peer server.

Figure 5 and Figure 6 describe the use case profile.

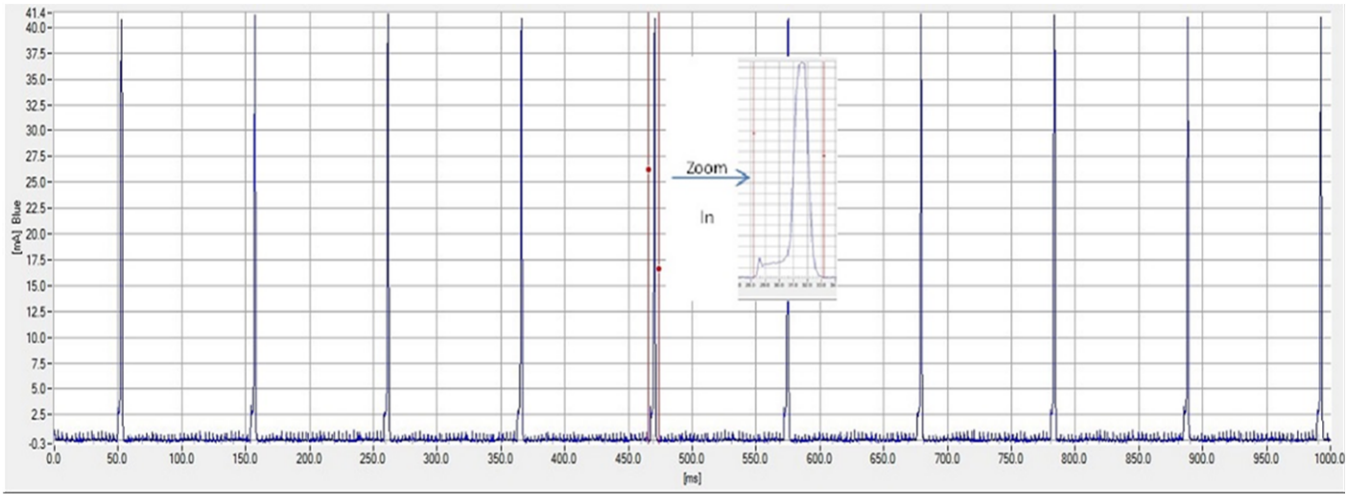


Figure 5. CC3100 Always Connected Idle Profile

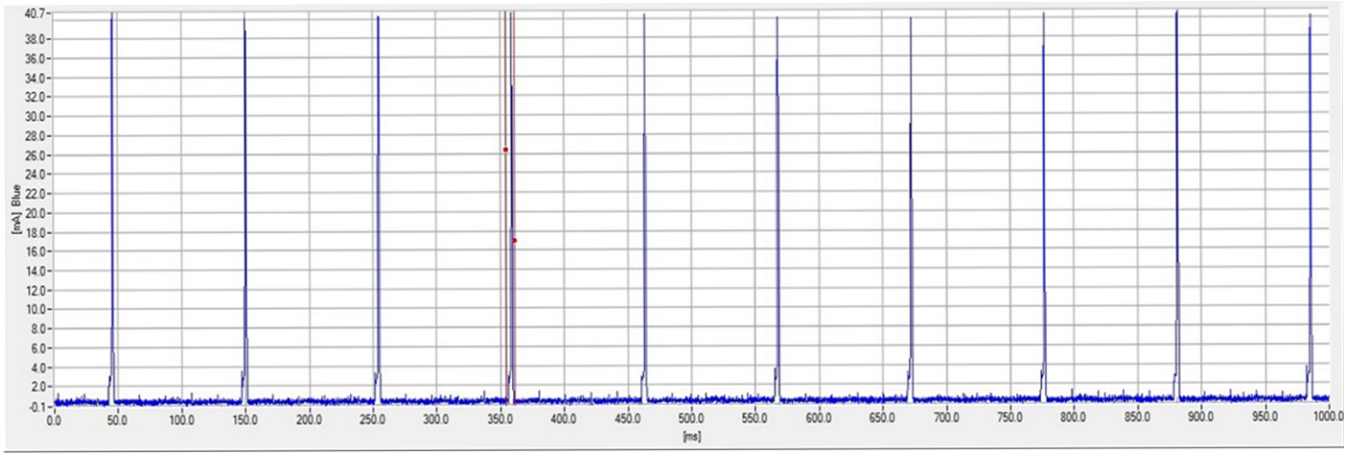


Figure 6. CC3200 Always Connected Idle Profile

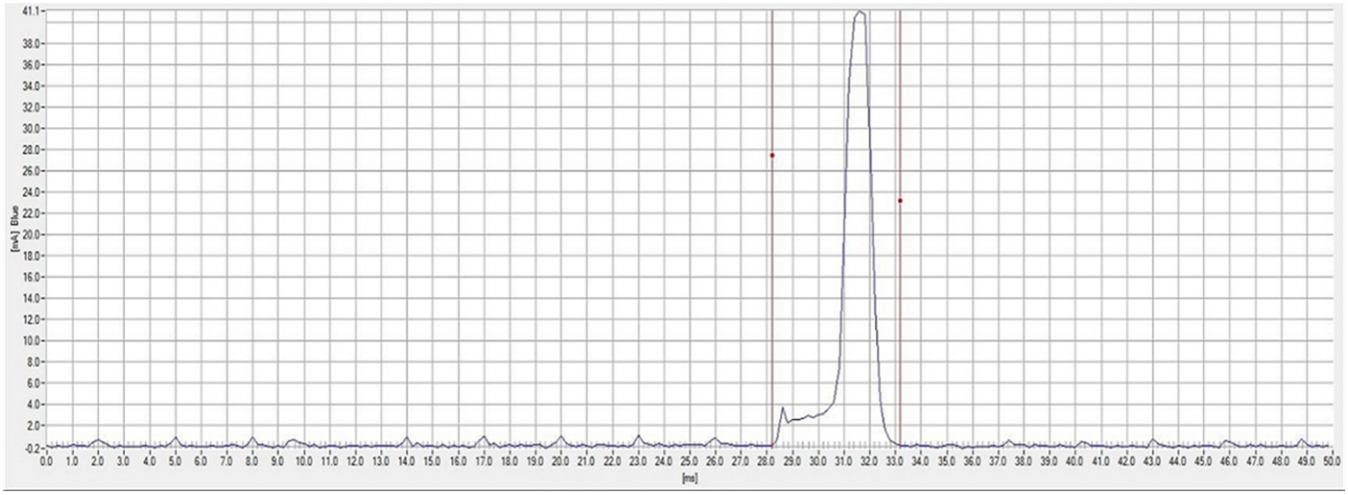


Figure 7. CC3100 Always Connected Idle Profile Zoom In

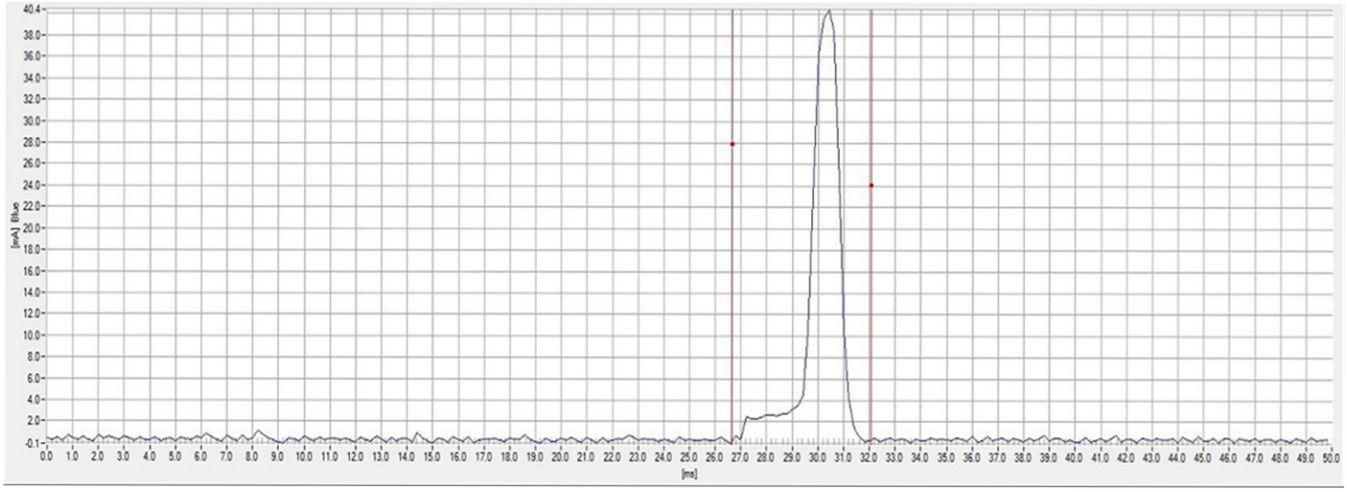


Figure 8. CC3200 Always Connected Idle Profile Zoom In

4.3 Use Case 2: Intermittently Connected

This profile is for devices that need to operate between long time intervals. In this mode, the device is in Hibernate state between working cycles, the lowest power consumption possible state. Almost all the device's components are shut-down, hence when waking up a new connection needs to be established. Optimizations were integrated in order to reduce the power consumption cost of this periodic reconnection, these are:

- Working with static IP (when possible) in order avoid DHCP.
- Set the connection policy to work with fast connect, which means that the device will first try to re-establish the previous connection.
- Disable Scan, as in this case we probably stay in the same channel and network (AP).
- Disable mDNS.

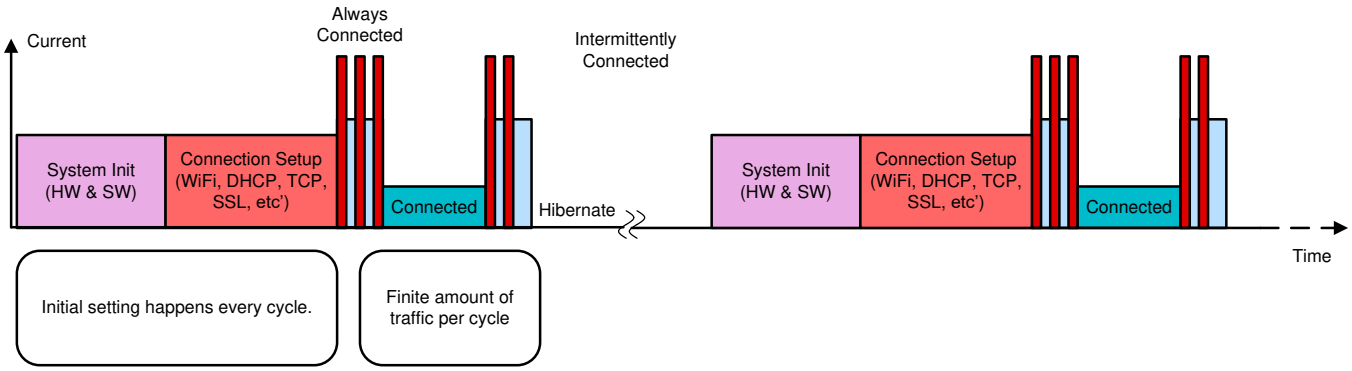


Figure 9. Use Case 2: Intermittently Connected

A good representation of a code for such case is:

```
//Configurations Done once – every exit from hib
sl_start(0,0,0);
sl_NetCfgSet(…) ;      // set static IP address to the device
sl_WlanPolicySet(...); // disable scan
sl_WlanPolicySet(…);   // set fast connect
sl_NetAppStop(..);     // disable mDNS

//Sending/Receiving Data - Done when packet need to be sent or receive
while (1) {
    sl_stop(10);       // to enter hibernate mode
    Delay();           // Long hibernate Time period
    sl_Start();        // device wake up and connect to network with previous setting
    sl_socket();       // usually UDP
    sl_SetSocketOpt(); // configure UDP/TCP Secured or not
    sl_bind();
    sl_RecvFrom();
    sl_SendTo();
    sl_Close();
}
```

4.3.1 Configuration Options

The define value that configures the behavior of the device in this case:

- NOT_ACTIVE_DURATION_IN_MSEC – defines the hibernate time period between 2 active states.
  - Example: "#define NOT_ACTIVE_DURATION_IN_MSEC 800"' will set 800 milliseconds hibernate time periods.

A detailed configuration example for intermittently connected use case is shown in Intermittently Connected configurations example figure; the relevant settings are emphasized.

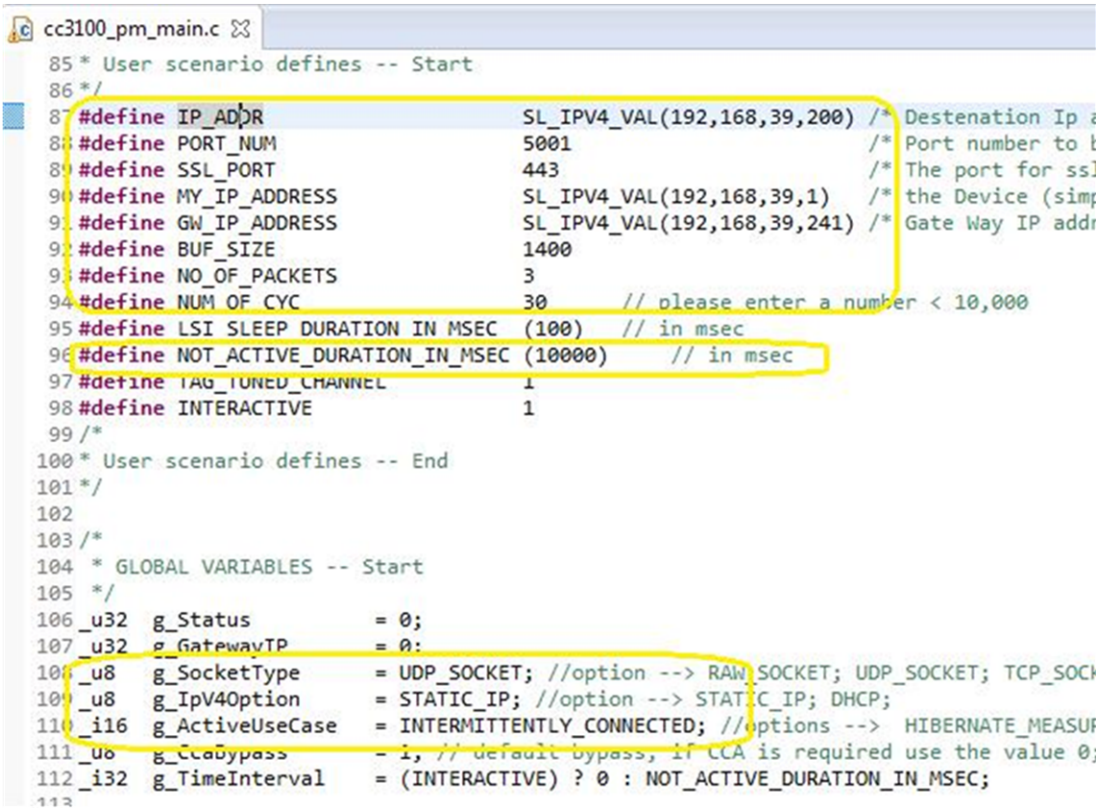


Figure 10. Intermittently Connected Configurations Example

4.3.2 Expected Results

Table 2. UDP Packet

Parameter	Description
Setup and scenario info	Device wakes up from hibernate every 5 seconds, and transmits 1 packet on UDP socket. The AP connection security is open.
Average current consumption (steady-state)	0.98 mA
Electric charge consumption of one active cycle	3.8 mC

Figure 11 and Figure 12 describe the use case profile.

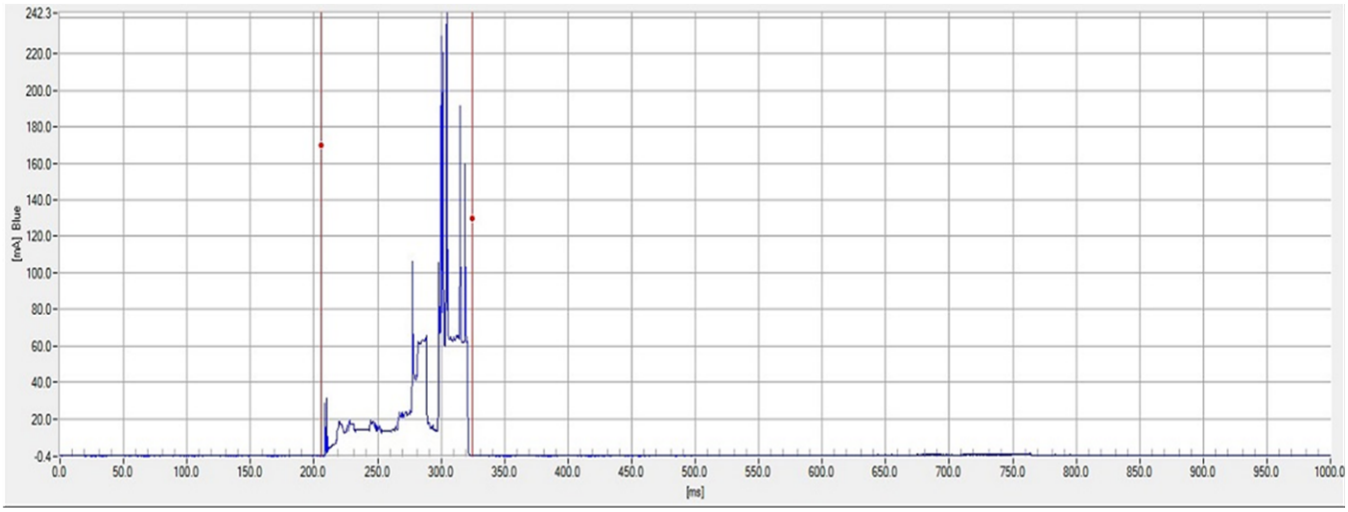


Figure 11. CC3100 Intermittently Connected 1 UDP Packet

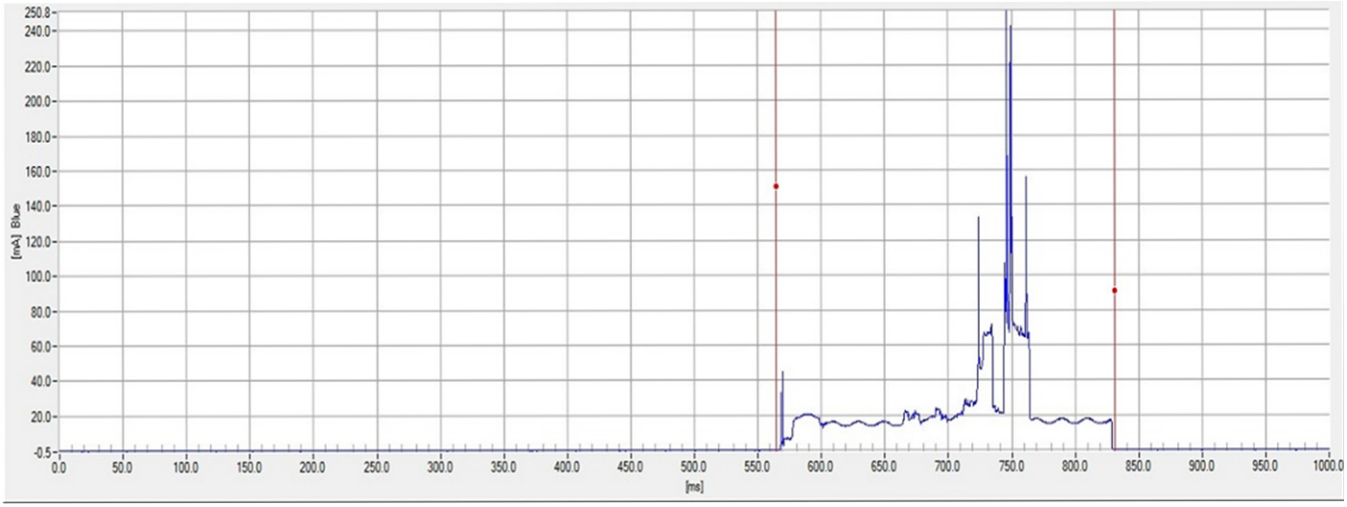


Figure 12. CC3200 Intermittently Connected 1 UDP Packet

Table 3. TCP Packet

Parameter	Description
Setup and scenario info	The AP connection security is open. Device wakes up from hibernate every 5 seconds. Establishes a TCP and SSL connection, and then transmit 1 TCP secured packet. The SSL version is SSLv3, and the cipher is "RSA_WITH_RC4_128_SHA".
Average current consumption (steady-state)	2.4 mA
Electric charge consumption of one active cycle	13.3 mC
Remarks	This examples requires a peer server. For more information on how to setup a server with Python scripts, see the Appendix section.

Figure 13 and Figure 14 describe the use case profile.

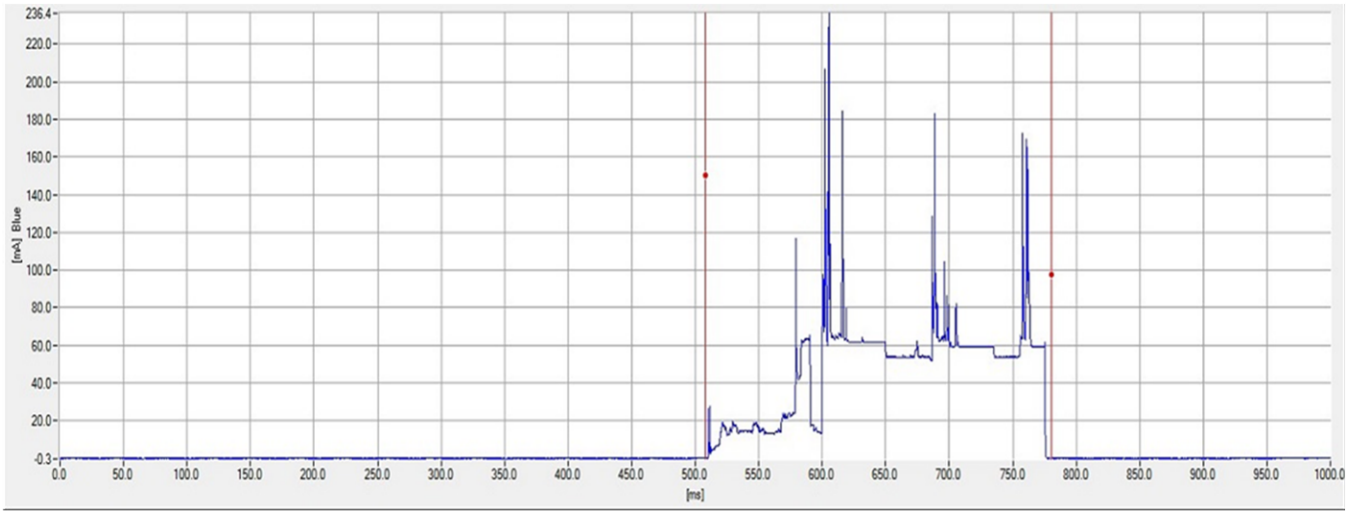


Figure 13. CC3100 Intermittently Connected 1 TCP Packet With SSL

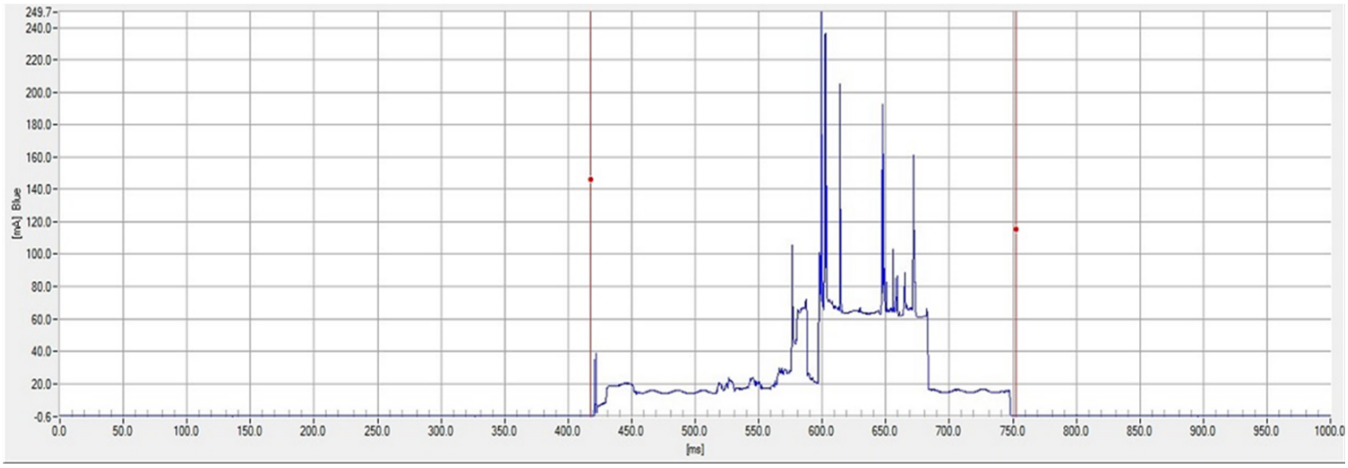


Figure 14. CC3200 Intermittently Connected 1 TCP Packet With SSL

4.4 Use Case 3: Transceiver Mode

For Transceiver mode a connection to standard Wi-Fi network is not required. The device is in hibernate state between operation times, and the socket in use is RAW hence not requires use of networking services. In this mode, you can configure some MAC/PHY attributes like:

1. Ignore CCA (clear channel assessment).
2. Set CCA threshold.
3. Set Tx timeout.
4. Do channel tune.
5. Set TX power.
6. Set TX rate.

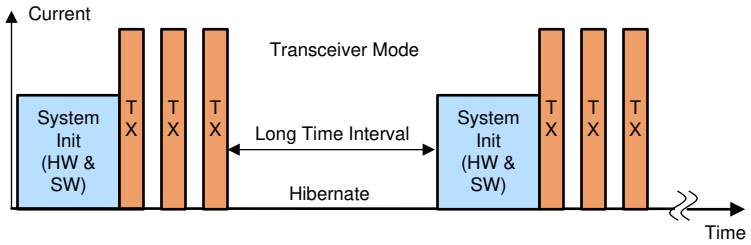


Figure 15. Use Case 3: Transceiver Mode

The tweaks made in this mode for power optimization are:

- Setting the PM policy to "SL_LOW_POWER_POLICY".
- Setting the connection policy to all zeroes, as connection is not required.
- The device is in hibernate state between operation.

To optimize transceiver mode, use the below configurations:

- Ignore CCA – While sending a packet in transceiver mode, CCA can be checked or bypassed. For power optimized configure device to bypass CCA.
  - Can be configured in the sl_Socket by choosing RAW or DGRAM socket.
- Set TX Power - Tx output power can be configured in order to reduce the current consumption, there are 15 steps:

NOTE: Can be set by configuring SL_RAW_RF_TX_PARAMS in the sl_Send command

- Set TX Rate – Transmission rate of the packet can be configured to reduce TX period and thus saving current.

NOTE: Can be set by configuring SL_RAW_RF_TX_PARAMS in the sl_Send command.

A good representation of a code for such case is:

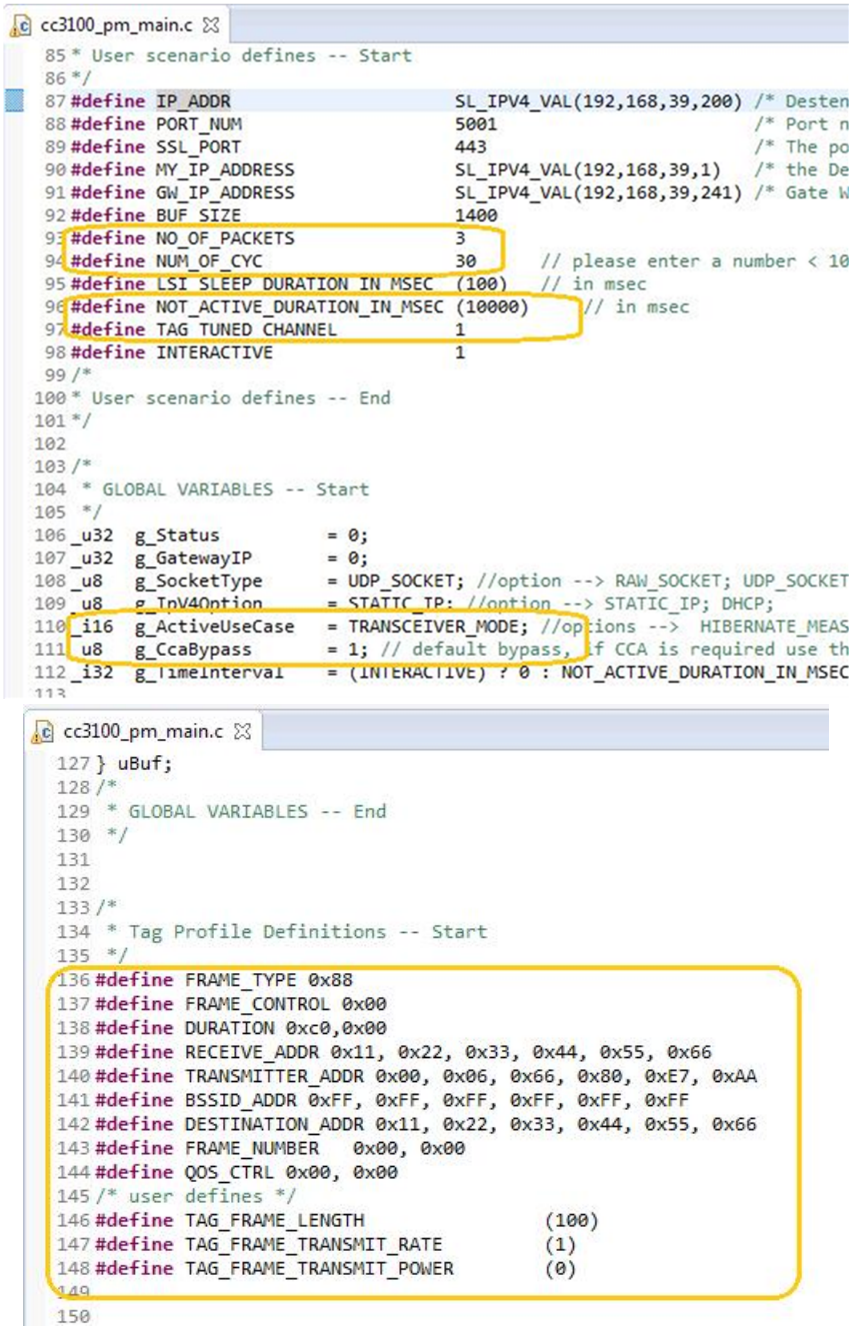
```
//Configurations - Done once
sl_start(0,0,0);
sl_WlanPolicySet(...);  // set "SL_LOW_POWER_POLICY"
sl_WlanPolicySet(…);    // set all connection option to zero

// Sending/Receiving Data - Done when packet need to be sent or received
while (1) {
    sl_stop(10);        // Enter hibernate mode
    Delay();            // Long hibernate Time period
    sl_Start();
    sl_socket();        // Raw socket
    sl_SetSockOpt(…);   // Optional setting CCA threshold & TX timeout
    sl_Send();          // TX param are passed also
}
```

4.4.1 Configuration Options

The define values that configure the behavior of the device in this case:

- TAG_TUNED_CHANNEL – defines the channel number the device will work and do channel tune.
  - Example: " #define TAG_TUNED_CHANNEL 1" will set the working channel to be channel 1
- NOT_ACTIVE_DURATION_IN_MSEC – defines the hibernate time period between 2 active states.
  - Example: "#define NOT_ACTIVE_DURATION_IN_MSEC 800" will set 800 milliseconds hibernate time periods.
- TAG_FRAME_LENGTH - Tag frame data length.
  - Example: "#define TAG_FRAME_LENGTH 50" will set the data length to 50.
- TAG_FRAME_TRANSMIT_RATE – the PHY rate to be used.
  - Example: "#define TAG_FRAME_TRANSMIT_RATE 6" will set mcs 6 or equivalent legacy rate.
- TAG_FRAME_TRANSMIT_POWER - TX Power to be used.
  - Example: "#define TAG_FRAME_TRANSMIT_POWER 7" will set the TX power.

A detailed configuration example for transceiver mode use case is shown in Transceiver mode configurations figure; the relevant settings are emphasized. A detailed configuration of MAC/PHY and L2 packet attributes is shown in PHY/MAC/L2 packet attributes configuration example figure.



cc3100_pm_main.c

```
85 * User scenario defines -- Start
86 */
87 #define IP_ADDR                    SL_IPV4_VAL(192,168,39,200) /* Desten
88 #define PORT_NUM                   5001                        /* Port n
89 #define SSL_PORT                   443                         /* The po
90 #define MY_IP_ADDRESS              SL_IPV4_VAL(192,168,39,1)   /* the De
91 #define GW_IP_ADDRESS              SL_IPV4_VAL(192,168,39,241) /* Gate W
92 #define BUF_SIZE                   1400
93 #define NO_OF_PACKETS              3
94 #define NUM_OF_CYC                 30      // please enter a number < 10
95 #define LSI_SLEEP_DURATION_IN_MSEC (100)   // in msec
96 #define NOT_ACTIVE_DURATION_IN_MSEC (10000)      // in msec
97 #define TAG_TUNED_CHANNEL          1
98 #define INTERACTIVE                1
99 /*
100 * User scenario defines -- End
101 */
102
103 /*
104  * GLOBAL VARIABLES -- Start
105  */
106 _u32  g_Status          = 0;
107 _u32  g_GatewayIP       = 0;
108 _u8   g_SocketType      = UDP_SOCKET; //option --> RAW_SOCKET; UDP_SOCKET
109 _u8   g_IpV4Option      = STATIC_IP; //option --> STATIC_IP; DHCP;
110 _i16  g_ActiveUseCase   = TRANSCEIVER_MODE; //options -->  HIBERNATE_MEAS
111 _u8   g_CcaBypass       = 1; // default bypass, if CCA is required use th
112 _i32  g_TimeInterval    = (INTERACTIVE) ? 0 : NOT_ACTIVE_DURATION_IN_MSEC
113
```

cc3100_pm_main.c

```
127 } uBuf;
128 /*
129  * GLOBAL VARIABLES -- End
130  */
131
132
133 /*
134  * Tag Profile Definitions -- Start
135  */
136 #define FRAME_TYPE 0x88
137 #define FRAME_CONTROL 0x00
138 #define DURATION 0xc0,0x00
139 #define RECEIVE_ADDR 0x11, 0x22, 0x33, 0x44, 0x55, 0x66
140 #define TRANSMITTER_ADDR 0x00, 0x06, 0x66, 0x80, 0xE7, 0xAA
141 #define BSSID_ADDR 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
142 #define DESTINATION_ADDR 0x11, 0x22, 0x33, 0x44, 0x55, 0x66
143 #define FRAME_NUMBER   0x00, 0x00
144 #define QOS_CTRL 0x00, 0x00
145 /* user defines */
146 #define TAG_FRAME_LENGTH                (100)
147 #define TAG_FRAME_TRANSMIT_RATE         (1)
148 #define TAG_FRAME_TRANSMIT_POWER        (0)
149
150
```

Figure 16. Transceiver Mode Configurations Plus PHY/MAC/L2 Packet Attributes

4.4.1.1 Expected Results

Table 4. Expected Results

Parameter	Description
Setup and scenario info	Device wakes up from hibernate every 5 seconds, and transmit 3 packets of 100 bytes each, on raw socket. The packet is OFDM at rate 6 Mbps & TX power is 7 (out 15).
Average current consumption (steady-state)	0.64 mA
Electric charge consumption of one active cycle	3.15 mC

NOTE: Channel tune is being carried out on every wake up (R1 limitation).

Figure 17 and Figure 18 describe the use case profile.

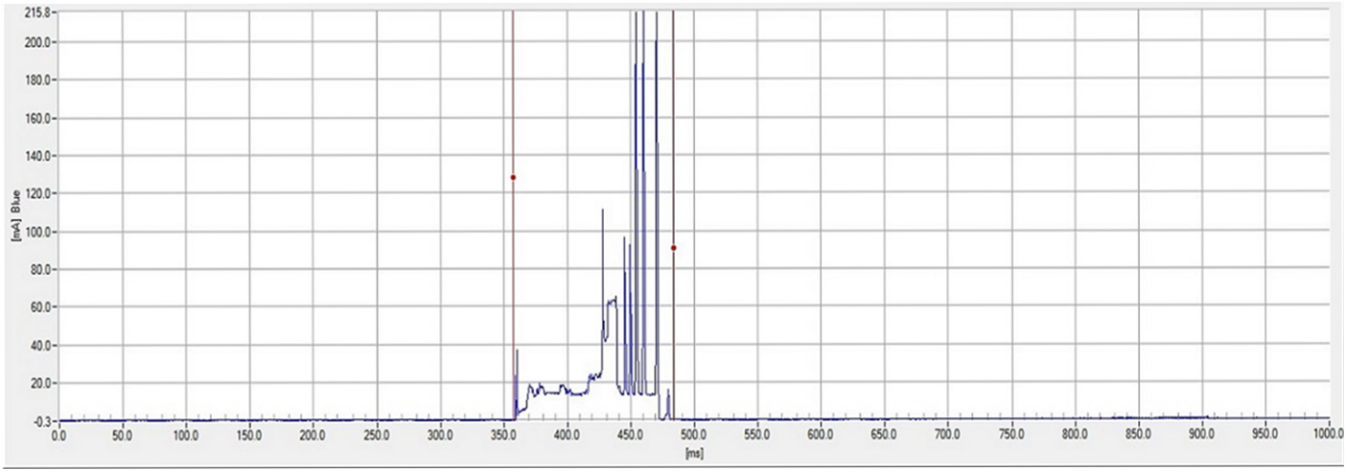


Figure 17. CC3100 Transceiver Profile 1 Min

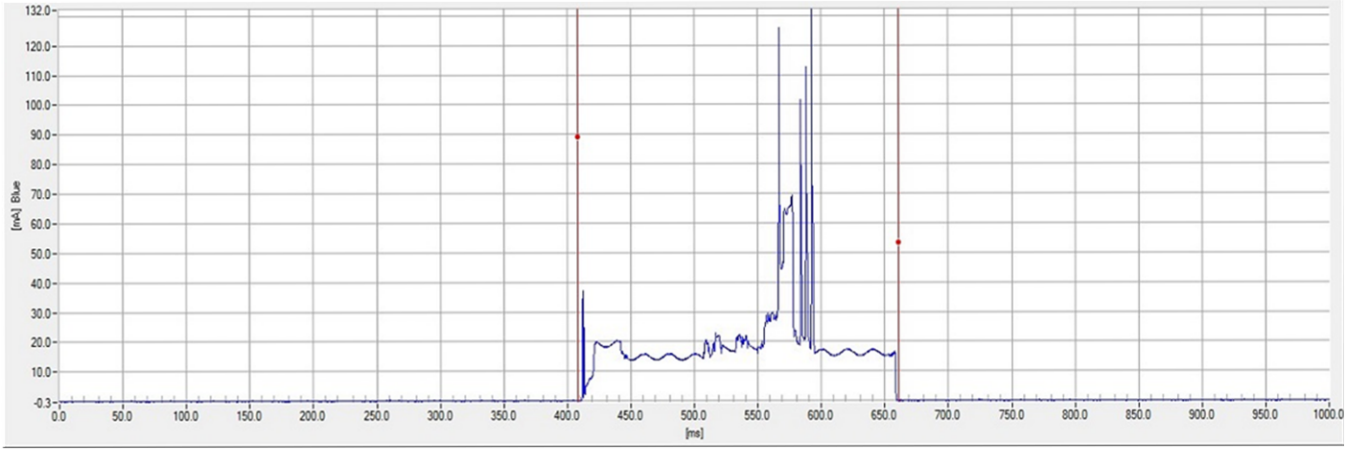


Figure 18. CC3200 Transceiver Profile 1 Min

5 Power Management Application Bench Mark

The power management application is released as a stand-alone CC3200 application, which is inserted within the CC3200 SDK folder. For the CC3100 the power management application is released as a stand-alone CC3100 application, which is integrated within the CC3100 SDK folder during installation. Import it into your environment (code composer /IAR), as any other project described in the *Getting Started User's Guide* for the device-specific documents.

The application requires some preliminary action to be taken/defined within the code. The next values need to be defined; the pre-define lines are already in the code, just fill the relevant values:

- IP_ADDR – Destination IP address
- MY_IP_ADDRESS – The device IP address
- GW_IP_ADDRESS – The gateway IP address
- NO_OF_PACKETS – The number of packets to be transmitted
- NUM_OF_CYC – defines how many times to repeat current use case
- BUF_SIZE - defines the packet data size
- INTERACTIVE – defines if we use the simple mode where small amount configuration/settings can be set interactively through sets of menus displayed on the terminal screen. The value 1 for this define (this is the default) will set the application to simple/interactive mode while 0 to the advanced one.

Moreover, there is also a need for a destination peer. A python script that implements TCP/UDP client or server is a simple solution for such peer. There are also good examples of DHCP server PC application that can be found in the internet, that the user can use.

5.1 How to Use

The power measurement application enables to switch between the various use cases and to configure the device settings through several global variables declared at the top of the main file. The table below describes how to use them. For your convenience, some defines were already created for all possible values. Note that if the user works in interactive mode the use case and socket type are set interactively, while the other (advanced setting) still needs to be defined in source code.

Table 5. Power Management Configurations

Variable Name	Value	Remarks
g_ActiveUseCase	HIBERNATE_MEASURE	Hibernate Current measurement mode
	SLEEP_MEASURE	Sleep current measurement mode
	TRANSCEIVER MODE	
	ALWAYS CONNECTED USE CASE	
	INTERMEDIATELY CONNECTED	
g_SocketType	UDP SOCKET	
	TCP SOCKET	
	SEC_TCP_SOCKET	Note: For secure socket there are default values for cipher & method variables.
g_IpV4Option	STATIC_IP	The Address value is defined in the "MY_IP_ADDRESS" define statement.
	DHCP	Obtain IP address through DHCP process.
g_CcaBypass	0	This variable is relevant only on Transceiver mode, the default value is 1, aka bypass the CCA
	1	Example

5.2 Example Usage for CC3200

This example will show how to run the application in "Intermittently connected" use case with static IP address and communicating via UDP socket.

1. Connect the CC3200 Launchpad to a windows PC with a Micro-USB cable.
2. Connect the current measurement tool.
   a. Check to make sure that all pull-ups, pull-downs, and resistors are in place (according the board measurement setup ECO).
   b. Remove the jumper from J6 and connect your current measurement tool.
3. Open CCS (Code Composer Studio) and Choose File->import from the menu, choose the CCS project.
4. Check the project "power_management" and press Finish.
5. Open the sl_common.h file under "[sdk-path\example\common\]" , and configure your network parameters. Set values for the: "SSID_NAME","SECURITY_TYPE" and "SECURITY_KEY" defines.

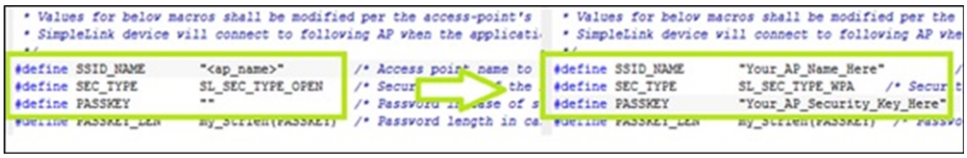


Figure 19. Setting AP Information in the Code

6. Open the main file from the project explorer in the CCS and configure the general & use case settings.

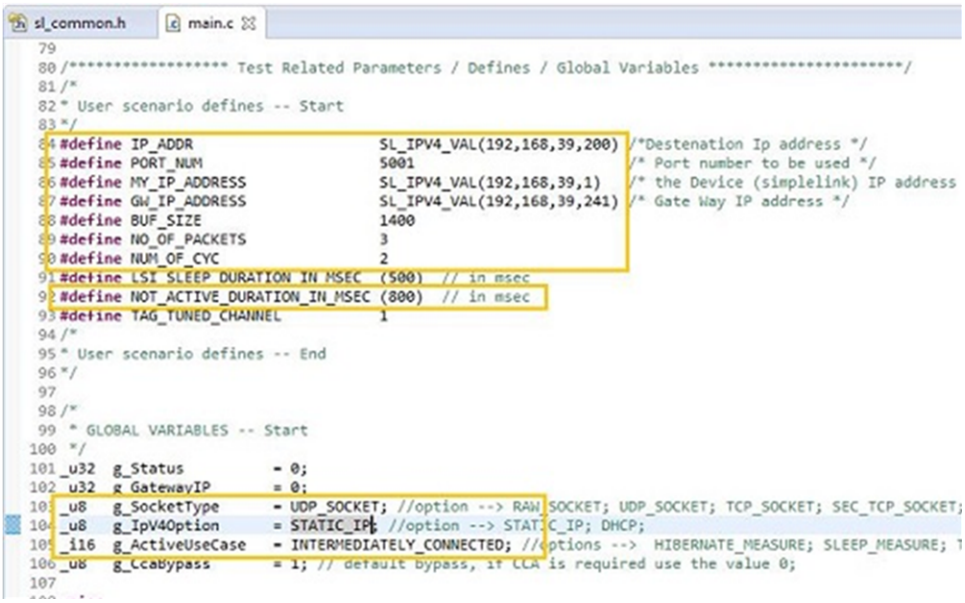


Figure 20. Global Variables and Define Desired Use Cases

7. Right click on the simplelink project and under the build configuration → set active select the "NON_OS_PM" option and recompile the simplelink project.

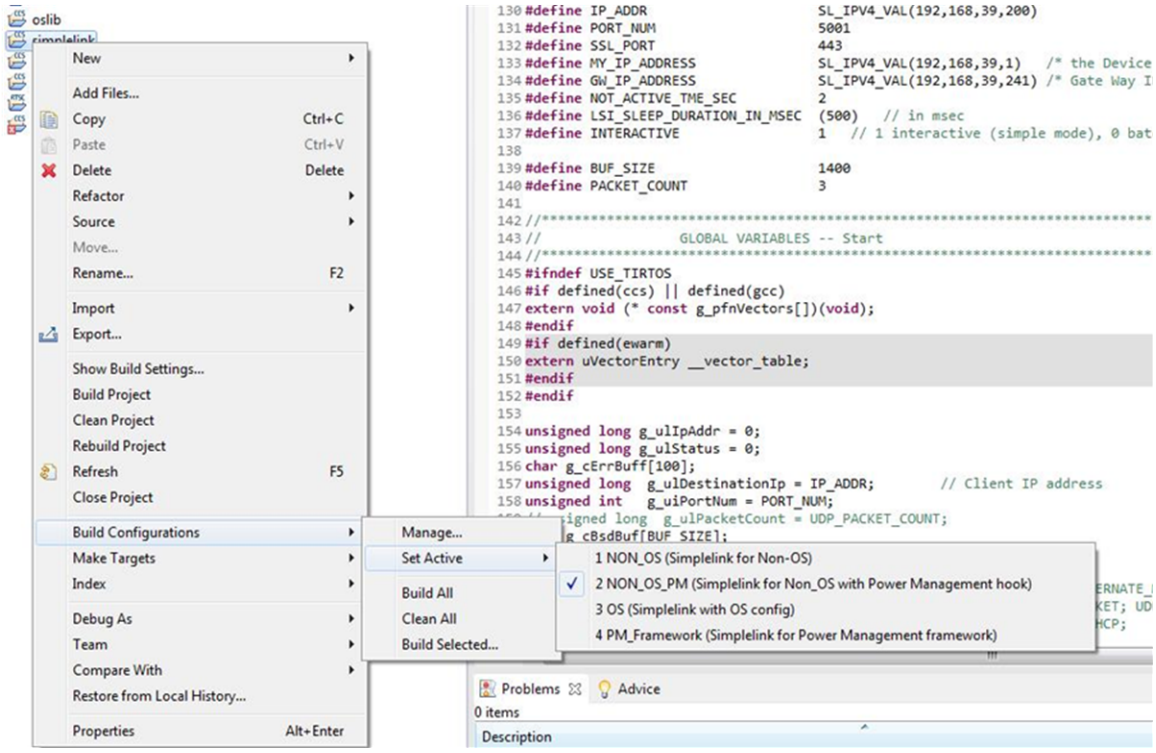


Figure 21. PM_CC3200_Configure NON_OS_PM Build Settings

8. Build the power management project. A bin file will be created under the "power_managment/Release/" directory.
9. Flash the device with the created bin file using UniFlash tool. For newbie with UniFlash please refer to the UniFlash user guide wiki.

NOTE: While flashing make sure that j15 jumper is shortened and no TeraTerm windows is open, otherwise the flashing process will fail.

10. Launch Tera Term on the relevant com port, the serial number of the com port may differ. The baud rate should be set to 115200.

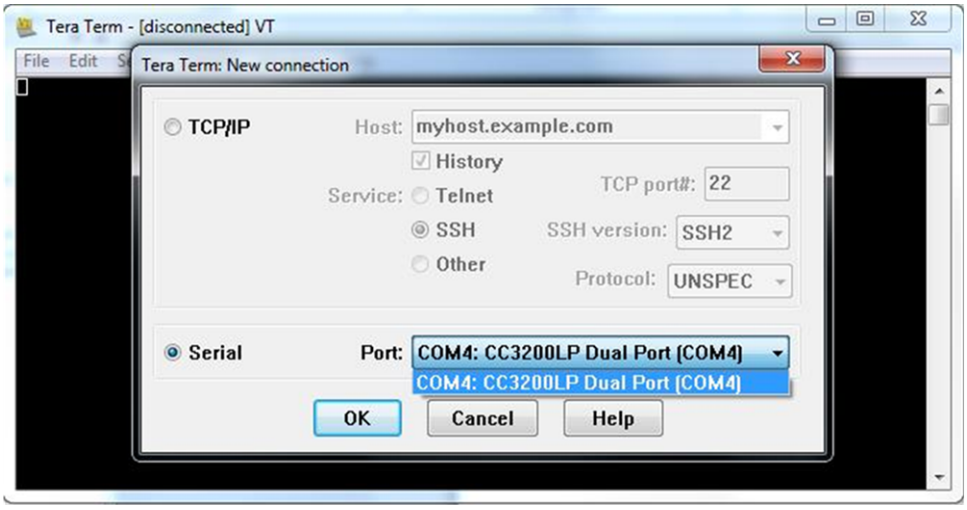


Figure 22. Teraterm UART Setup

11. Open the J15 jumper (CC3200 Launchpad rev3p2) and press the reset button the application should start running.

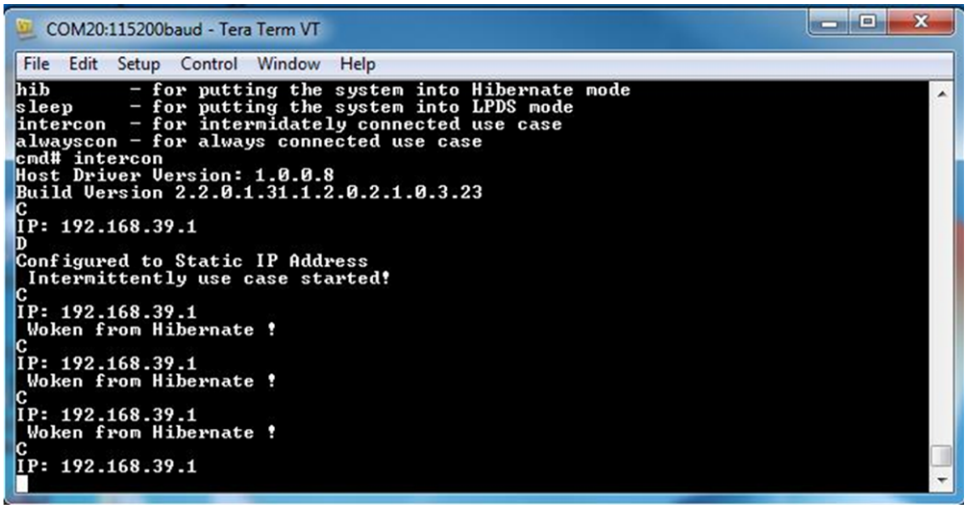


Figure 23. Teraterm Application Termination

5.3 Example Usage for CC3100

This example will show step by step how to run the application in "Intermittently connected" use case with static IP address and communicating via UDP socket.

1. Connect Micro-USB cable from J401 on the MSP430 Launchpad to a windows PC.
2. Connect the current, measurement tool.
   a. Check to make sure that all pull-ups, pull-downs, and resistors are in place (according the board measurement setup ECO).
   b. Remove the jumper from J6 and connect your current measurement tool.
3. Open CCS (Code Composer Studio) and Choose File->import from the menu, choose CCS project.
4. Under select search directory, enter the path: C:\ti\CC3100SDK_1.0.0\cc3100-sdk\platform\msp430f5529lp\example_project_ccs.
5. Check the project "power_management" and press Finish.
6. Open the sl_common.h file located at the path C:\TI\CC3100SDK_1.0.0\cc3100-sdk\examples\common\, and configure your network parameters.

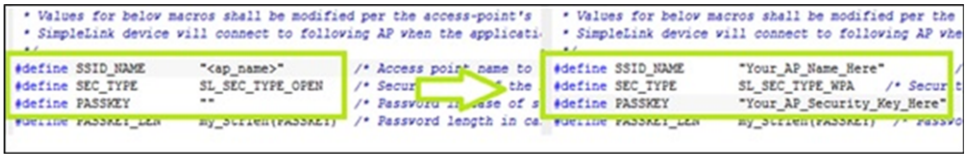


Figure 24. Setting AP Information in the Code

7. Open the main file from the project explorer in the CCS and configure the general & use case settings.

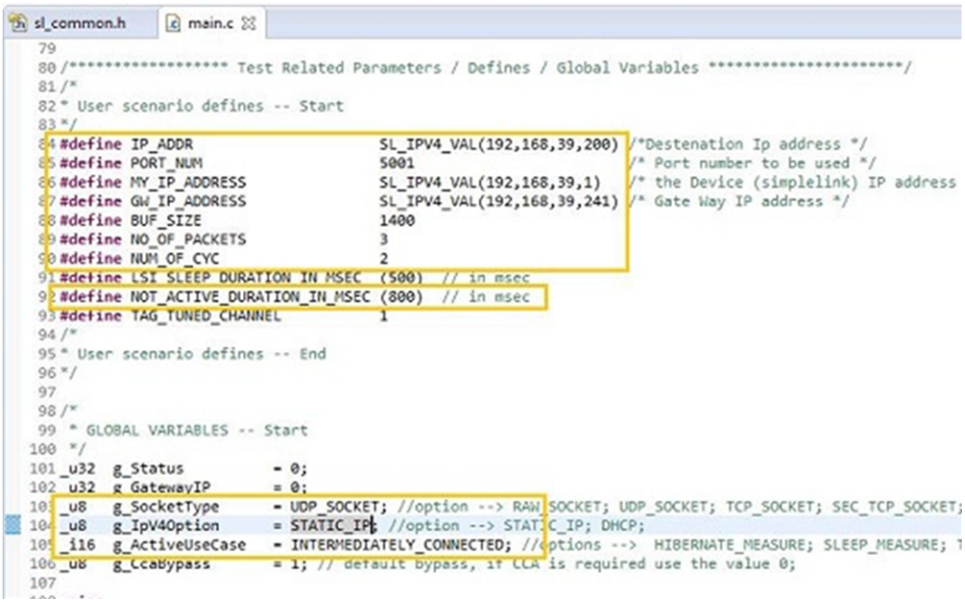


Figure 25. Global Variables and Define Desired Use Cases

8. Launch Tera Term on the relevant com port , the serial number of the com port may differ.

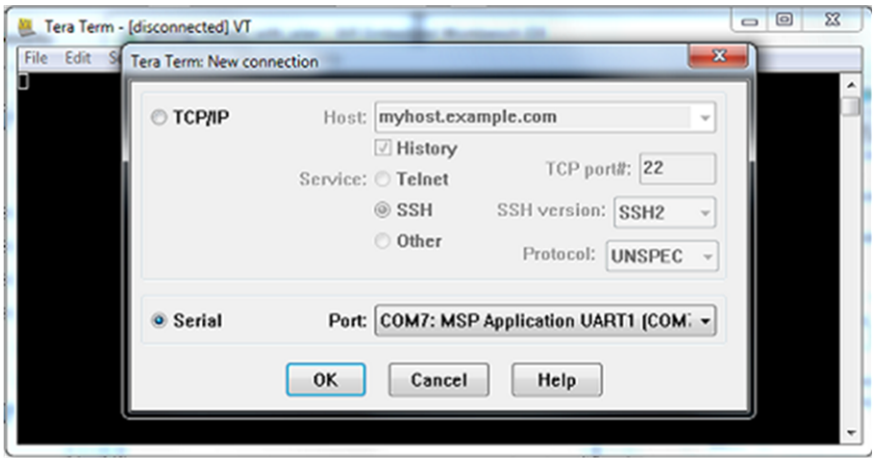


Figure 26. Teraterm UART Setup

9. Press F11 on the CCS window, and let the application run. On successfully termination you should see in the Tera Term outputs similar to Figure 27.

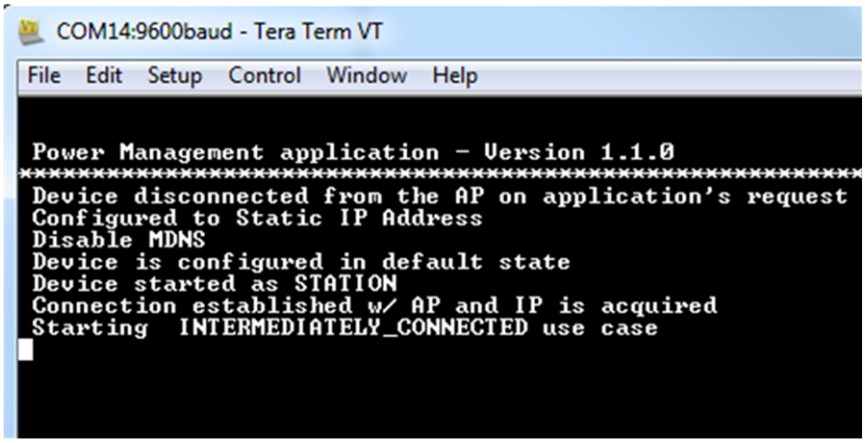


Figure 27. Teraterm Application Termination

10. The next sniffer snapshot corresponds to the traffic initiated by the application in one hibernate → active → hibernate cycle:

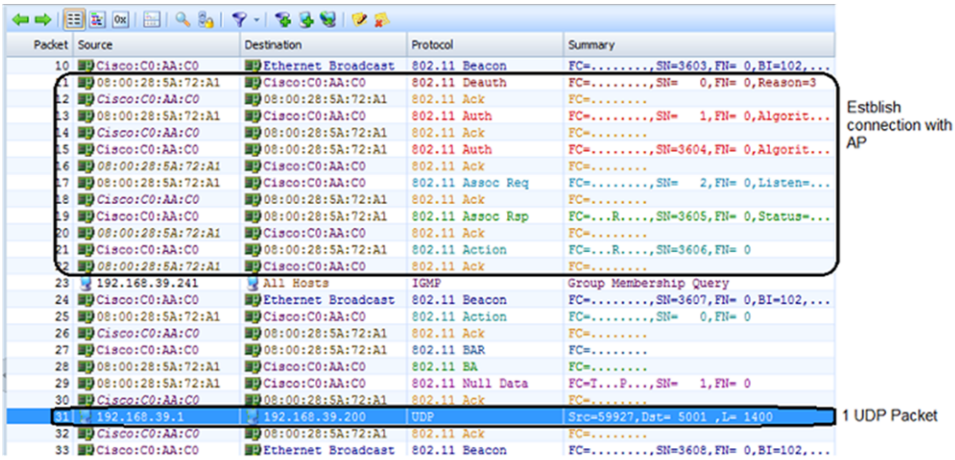


Figure 28. Sniffer Snapshot of Intermittently Connected Mode

The current measurement graph should look like this graph of Graph of 3 active cycles for intermittently connected mode. In each state connection is reestablished and one UDP packet is sent.

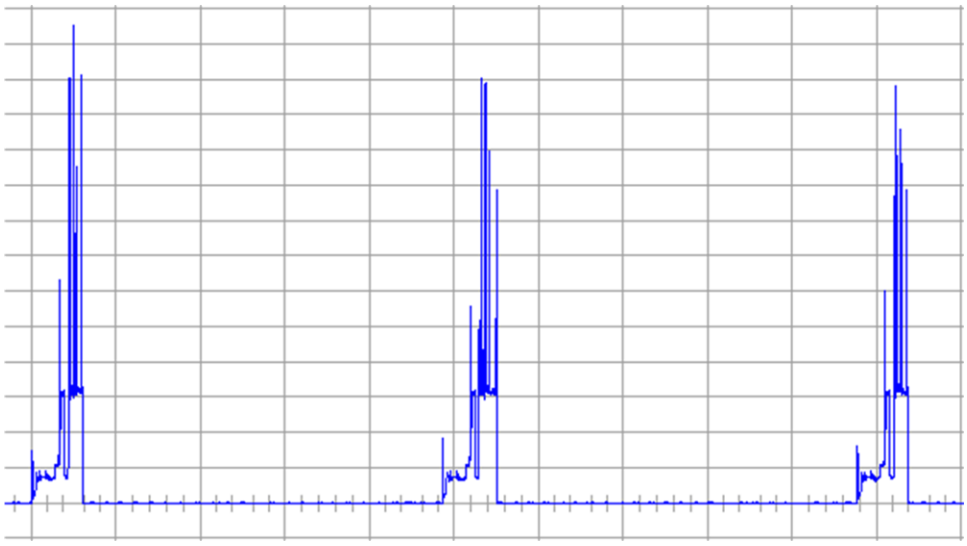


Figure 29. Active Cycles for Intermittently Connected Mode

6 Current Consumption Measurements Setup

6.1 Current Measurement for Profiles and Active States

Application energy consumption profile describes the consumption of the system for different system modes over time. The profile also includes the energy consumed for the transition between system modes. The total energy consumed is an integral of the current consumed, measured in units of Amperes, over time from the supply:

- Energy: E [Jouls]= ∫Vsupply*Idt

Alternatively the charge consumption could be calculated:

- Charge: C [coulomb]= ∫ Idt

Commonly, a simpler, piece-wise linear approximation description of energy consumption is used:

- Energy: E [Jouls]= ∑_(x=1)^n(Vsupply*Ix*Tx)

Alternatively the charge consumption could be calculated:

- Charge: C [coulomb]= ∑_(x=1)^n(I_x*T_x)

The application power consumption could be measured in various methods, using:

- Oscilloscope with Current Probe
- Mobile Communications DC source (Example presented using Agilent 66319D)

These methods are recommended for dynamic and active current consumption measurements.

6.2 Oscilloscope With Current Probe

6.2.1 Tools Needed

- Oscilloscope(Tektronix TDS7404/TDS5104B)
- Current probe with Amplifier(Tektronix TCP312 and TCPA300)
- External 3.3 V supply source(Agilent E3631A)
- Short cables to connect 3.3 V external supply to device

6.2.2 FW Needed

- Use TI PM benchmark code or your application FW

6.2.3 Pre-Requisites / Things to Consider

1. This setup is suited for active current measurements only or current numbers, which are higher than min measurable currents using particular current probe (with current probe and oscilloscope which we have in our lab, we could measure current numbers accurately which are higher than 10mAmp).
2. Make sure FULL bandwidth is selected for the channel where current probe is connected this is to make sure you do not miss out any glitch/spike while measuring currents.
3. Calibrate current probe before connecting across supply line and align current direction mark (arrow) on current probe with current direction in supply line (if not you will see negative currents on the oscilloscope).
4. Always feed external 3.3 V supply at VBAT_CC pin and connect current probe across this supply line and measure current.
   a. For the CC3200Launchpad : Connecting wire across jumper(J12) is not a good method, since this will alter the DCDCs inductor values.
5. Check the current limit that you have set for the external supply, lower current limit may RESET the device.
6. Calibrate for voltage drop across the wire connecting supply to the device (use short and multi strand wire).

6.2.4 Procedure

1. Modifications for each board
   a. For CC3100Boosterpack: Remove jumper J6, Connect at Jumper J10, Pin_3.2 for GND.
   b. For CC3200LaunchPad: Remove resistor R62 across jumper J12 and also jumper J12.
2. Connect external supply to the pin, which connects to device(on other pin we can measure 3.3V which is coming from board supply).
3. Connect current probe to the positive supply wire and make sure current direction in the wire and arrow on the current probe are in same direction.
4. Start executing the code and measure the active currents on the oscilloscope.
   a. To measure the charge of a profile, place the cursors to limit the profile and use “math” function to perform integral over time. The result will be the total profile charge in [Coulombs] in between the cursors.
   b. To measure the peak current of transmit (TX) and receive (RX) modes use the “Y” axis cursors.

6.3 Mobile Communications DC Source

6.3.1 Tools Needed

- Agilent 66319D
- Install NI Measurement and Automation explorer and Agilent 66319D GUI
- NI GPIB-USB connector
- Wires to connect instrument to device

6.3.2 FW Needed

- Use TI benchmark code or your application FW

6.3.3 Pre-Requisites /Things to Consider

1. This particular setup can be used to measure transient currents.
2. Set proper supply Voltage, Current and series resistor value before turning ON the supply Output.
3. Make sure you select proper range in the “Range” tab before you start the measurements.
   a. There are three ranges: Low, Mid and HiGH. Select based on the requirement.
   b. Select the proper time and current divisions, these will be at top left and top right corner of plot window.
4. Plot in the GUI might not match the current numbers due to software issue. But the Minimum, Average and Maximum values displayed at the bottom are accurate.
5. In order to measure max current of particular spike/peak use marker, place markers on either side of the spikes and look for max currents.
6. There is no necessity to tweak settings for the measurements accuracy. Hence Set it to default, software automatically sets the best possible accuracy with the current configuration.
7. If you select the 4wire connection mode then software will take care of drop across the wire, there is no need to compensate for the drop across the wire.

6.3.4 Procedure

1. Modifications for each board.
   a. For CC3100Boosterpack: Remove jumper J6, Connect at Jumper J10, Pin_3.2 for GND.
   b. For CC3200LaunchPad: Remove resistor R62 across jumper J12 and also jumper J12.
2. Install necessary software for the instrument 66319D.
3. Open 66319D GUI and go to “Source” tab and select “I/O Configuration…” .
   a. On the pop window press “Auto-Detect”.
   b. This will show instrument details, which are connected to your PC and prompt if you want to use 66319D as power supply, click “Yes”.
4. Connect positive terminal of power supply to VBAT_CC pin and GND to board GND.
5. Configure supply voltage, current and series resistor value and then turn on the power supply.
6. Start executing code and press “DLOG” tab if you want to log the data or press “Measure” if you want to measure instantaneous current numbers.
7. To measure average current over certain period of time or over one complete cycle of active + low power mode, stop measurement after desired time period or one cycle and place markers on either ends and measure currents.

6.4 Static Current Measurements

The static low power modes current consumption could be measured using digital multi-meter. Here is a description of the setup:

6.4.1 Tools Needed

- Digital multimeter(Agilent 34401A)
- Short cables to connect multimeter in series with the device
- Optional: Keysights Software

6.4.2 FW Needed

- Use TI benchmark code to generate the system modes

6.4.3 Pre-Requisites/Things to Consider

1. This particular setup should be used to measure constant currents(both low and high currents can be measured provide these currents are constant over time or long interval).
2. Start all current measurements with higher current range setting on ammeter and then depending on low power mode configured, reduce the current measurement range.
3. Take care of the current direction through the ammeter, wrong current direction may lead to negative current numbers.
4. Use short cables for connecting ammeter to device.

6.4.4 Procedure

1. Modifications for each board.
   a. For CC3100Boosterpack: Remove jumper J6
   b. For CC3200LaunchPad: Remove resistor R62 and connect ammeter across jumper J12
2. Configure multimeter to measure DC current and set higher current range.
3. Check for the current direction, wrong current direction will lead to negative currents.
4. Release RESET and execute the code.
5. Once device enters into the configured low power mode, reduce current range and note down the current numbers.
6. Before you exit from low power mode/RESET the device increase the current range on multimeter to higher value.

7 References

- Texas Instruments: *CC3100 SimpleLink Wi-Fi and IoT Solution Getting Started Guide*
- Texas Instruments: *CC3200 SimpleLink Wi-Fi and IoT Solution w/ MCU LaunchPad Getting Started Guide*
- Texas Instruments: *SimpleLink™ CC3100/CC3200 Wi-Fi Internet-on-a-chip™ Networking Sub-system Power Management*
- CC3100 & CC3200 Radio Tool wiki

Revision History

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

Changes from Original (March 2020) to A Revision	Page
• Update was made in .	1
• Update was made in Section 1.	2
• Updates were made in Section 4.2.1.	7
• Updates were made in Section 4.3.2.	11
• Updates were made in Section 4.4.1.1.	16

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265