

CC3200 Power Management Optimizations and Measurements

Introduction

[Return to CC31xx & CC32xx Home Page](#)

The CC3200 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. 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 in the different power profiles use cases.

The various system's 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:

- Always connected (aka. Idle connected)
- Intermittently connected
- Transceiver mode

Note: that 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 activities.

Getting started

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, please refer to CC3200 getting started user guide ^[1], before proceeding.

Pre-requisites

Hardware

- CC3200 LaunchPad Rev4.1 or above
 - Oscilloscope with differential probe or current probe
 - Digital multimeter (capable of measuring down to 1 uA)
-

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.

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

```
150 //*****
151 // user defined //
152 signed short g_ActiveUseCase = SLEEP_MEASURE; //options --> RISEWAKE_MEASURE; SLEEP_MEASURE;
153 unsigned char g_RocketType = UDP_SOCKET; //option --> RAW_SOCKET; UDP_SOCKET; TCP_SOCKET; SB;
154 unsigned char g_TxOption = STATIC_IP; //option --> STATIC_IP; DHCP;
155 unsigned char g_CoBypass = 1; // default bypass, if CoA is required use the value 0;
```

Measurement Tool

Follow the steps mentioned in the static current measurements section, in order to record the LPDS current.

Expected Results

The expected power consumption numbers are described in the datasheet. ^[4]

Active (Rx & 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, etc...

Configuration Parameter

In order to individually measure the Tx and Rx current consumptions, please refer to the Appendix section of **Using radiotool for active modes**.

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

Expected Results

The expected power consumption numbers for each Rx and Tx states are described in the datasheet. ^[4]

Power Profiles Use Cases

The Power Profiles use cases combine various power states 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 drives 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.

The majority of the use cases for battery powered devices in the IoT context would necessitate the device to be in its lowest possible power state for most of its lifetime, only having to intermittently wake up and take some action. The choice of the suitable low power mode is primarily a trade-off between the response time and the power consumption. Two popular deployment use cases are considered here and the power consumption determined.

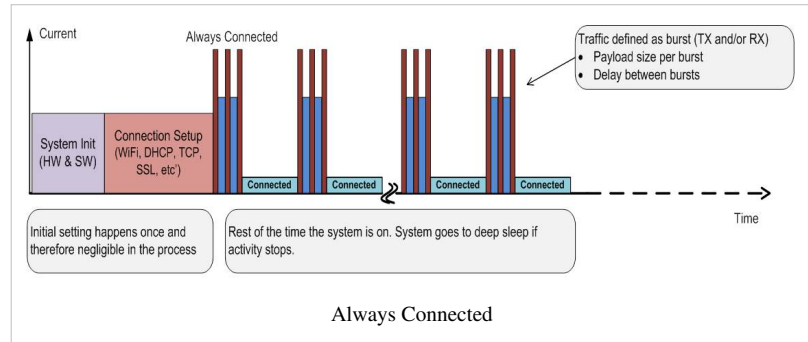
Idle Connected Profile – Use cases with responsiveness being of paramount importance would place the system in the LPDS. In this mode the device is always connected to the access point. The entire system is in LPDS with only the WLAN subsystem periodically waking up to service beacons. This system is remotely triggered to wakeup and takes action, usually by issuing commands over cloud based services.

Sensor profile - Non-time-critical use cases that have infrequent activity and need to take action either periodically or based on an external trigger (GPIO) would place the system in HIBernate. Most of the sub-systems are in its powered down state always and occasionally wakes up to perform all the necessary actions.

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

Use Case 1: Always Connected

The always connected use case deals with situations where the device must stay connected to AP in any time. Staying connected may cause a high power consumption due to periodic beacon functionality. In order to reduce the power penalty, the next improvement was implemented in the application:



- **LSI (Long Sleep Intervals)** – A configuration that enable to wake the device up only on every n-th beacon thus providing longer Sleep periods.

A good representation of a code for such case is:

```
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();
while (1) {
    sl_Send();           // can be UDP/TCP and/or secure connection
    sl_Recv();           // can be UDP/TCP and/or secure connection
    Delay();
}
```

Configuring Options

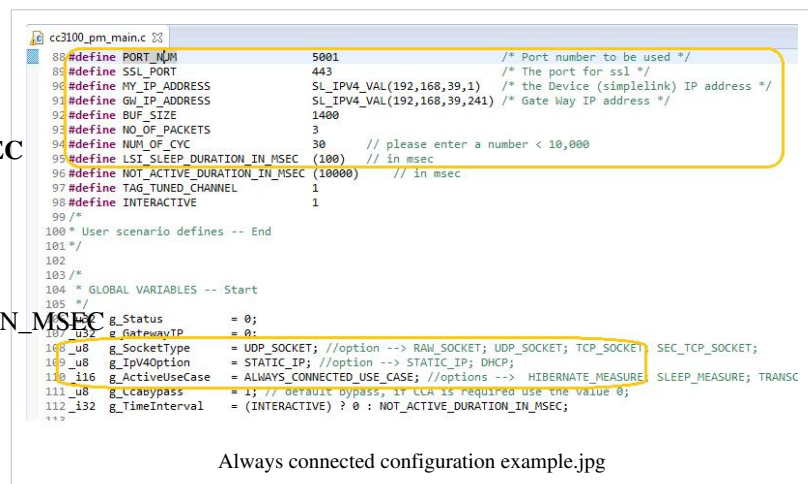
There is one define value that configures the behavior of the device in this case:

- **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.

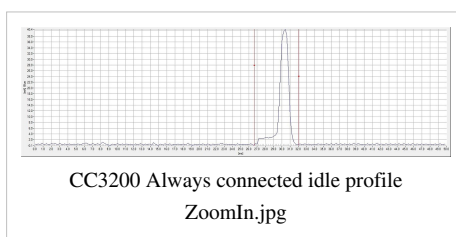
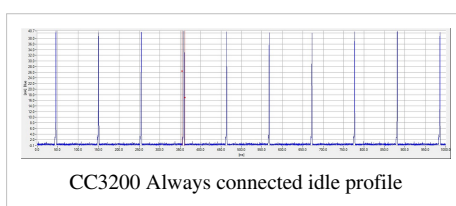


Always connected configuration example.jpg

Expected Results

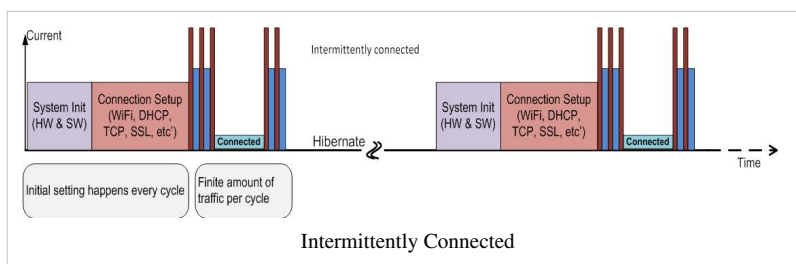
Parameter	Description
Setup & scenario info	The AP connection security is open. Device is in Idle profile (beacons receive only). Beacon interval is 102ms
Average current consumption (steady-state)	0.85 mA
Electric charge consumption of one active cycle	0.05 mC (one Beacon reception)
Remarks	By default, 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.

The following figure describes the use case profile



Use Case 2: Intermittently Connected

This mode 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 components are shut-down, hence when waking up a new connection needs to be established.



The next tweaks were taken in order to reduce the power consumption cost of this periodic establish connection action.

1. Working with static IP (when possible) in order avoid DHCP.
2. Set the connection policy to work with fast connect, which mean the device will try first to connect to the previous connection.
3. Disable Scan, as in this case we probably stay in the same channel & network (AP).
4. Disable mDNS.

A good representation of a code for such case is:

```

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
while (1) {
    sl_stop(10);               // 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();
}

```

Configuration Option

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.

```

85 * User scenario defines -- Start
86 */
87 #define IP_ADDR          SL_IPV4_VAL(192,168,39,200) /* Destination Ip address */
88 #define PORT_NUM         5001                       /* Port number to listen */
89 #define SSL_PORT         443                         /* The port for ssl */
90 #define MY_IP_ADDRESS    SL_IPV4_VAL(192,168,39,1)   /* the Device (simple) IP address */
91 #define GW_IP_ADDRESS    SL_IPV4_VAL(192,168,39,241) /* Gate Way IP address */
92 #define BUF_SIZE         1400
93 #define NO_OF_PACKETS    3
94 #define NUM_OF_CYC        30 // please enter a number < 10,000
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; TCP_SOCKET;
109 _u8 g_IpV4Option        = STATIC_IP; //option --> STATIC_IP; DHCP;
110 _i16 g_ActiveUseCase     = INTERMITTENTLY_CONNECTED; //options --> HIBERNATE_MEASURE;
111 _u8 g_CcAbypass         = 1, // default bypass, if CCA is required use the value 0;
112 _i32 g_TimeInterval     = (INTERACTIVE) ? 0 : NOT_ACTIVE_DURATION_IN_MSEC;
113

```

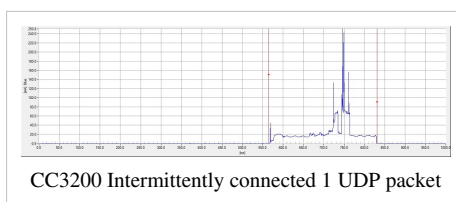
Intermittently Connected configurations example

Expected Results

UDP Packet

Parameter	Description
Setup & scenario info	Device wakes up from hibernate every 5 seconds, and transmit 1 packet on UDP socket. The AP connection security is open.
Average current consumption (steady-state)	1.3 mA
Electric charge consumption of one active cycle	6.5 mC

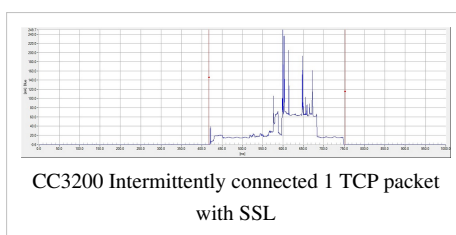
The following figure describes the use case profile:



TCP Packet

Parameter	Description
Setup & 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)	2.25 mA
Electric charge consumption of one active cycle	10.8 mC
Remarks	This examples requires a peer server. See Appendix section on how to setup a server with Python scripts.

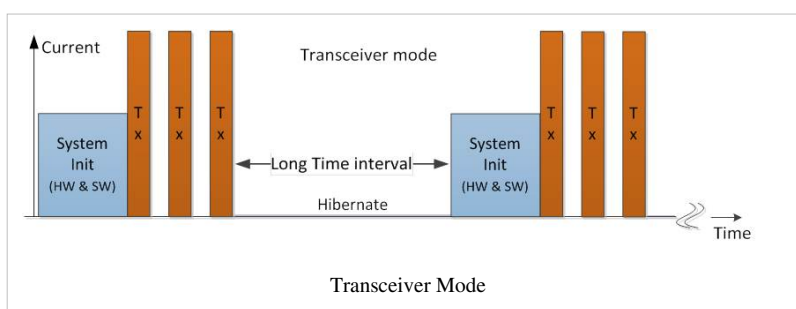
The following figure describes the use case profile:



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 we can configure some MAC/PHY attributes like:

- Ignore CCA (Clear channel Assessment).



- Set CCA threshold.
- Set Tx timeout.
- Do channel tune.
- Set TX power.
- Set TX rate.

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.

A good representation of a code for such case is:

```
sl_start(0,0,0);
sl_WlanPolicySet(...); // set "SL_LOW_POWER_POLICY"
sl_WlanPolicySet(...); // set all connection option to zero
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
}
```

Configuration Options

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

1. **TAG_TUNED_CHANNEL** – defines the channel number the device will work and do channel tune.

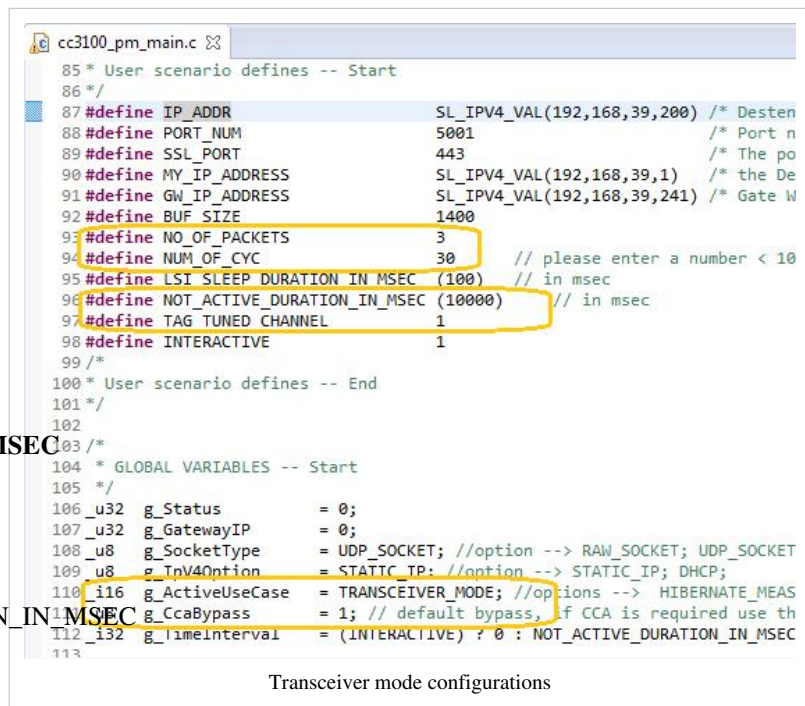
Example: " #define
TAG_TUNED_CHANNEL
1" will set the working
channel to be channel 1.

2. **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.

3. **TAG_FRAME_LENGTH** - Tag frame data length.



Example: "#define
TAG_FRAME_LENGTH
50" will set the data length
to 50.

4. 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.

5. TAG_FRAME_TRANSMIT_POWER

– TX Power to be used.

Example: "#define
TAG_FRAME_TRANSMIT_POWER
7" will set the TX power.

```
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
```

PHY/MAC/L2 packet attributes configuration example

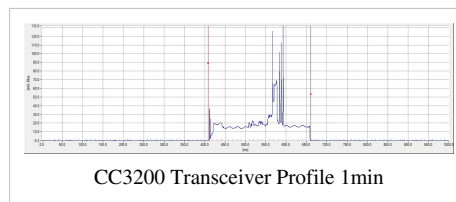
A detailed configuration example for transceiver mode use case is shown in

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.

Expected Results

Parameter	Description
Setup & 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)	0.940 mA
Electric charge consumption of one active cycle	5 mC
Note:	Channel tune is being carried out on every wake up (R1 limitation)

The following figure describes the use case profile:



Power Management Application BenchMark

The power management application is released as a stand alone CC3200 application, which is inserted within the CC3200 SDK folder. Import it into your environment (code composer /IAR), as any other project described in the getting started user guide.

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:

1. **IP_ADDR** – Destination IP address.
2. **MY_IP_ADDRESS** – The device IP address.
3. **GW_IP_ADDRESS** – The gateway IP address.
4. **NO_OF_PACKETS** – The number of packets to be transmitted
5. **NUM_OF_CYC** – defines how many times to repeat current use case.
6. **BUF_SIZE** - defines the packet data size.
7. **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.

How to use

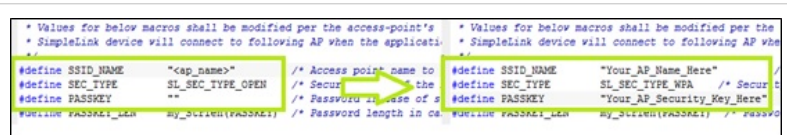
The power measurement application enables to switch between the various use cases and to configure the device settings through several global variables declared at the top of the main file. The table below describes how to use them. For the user's conveniences, some defines were already created for all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

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

Example Usage

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.
 1. Check to make sure that all pull-ups, pull-downs, and resistors are in place (according the board measurement setup ECO).
 2. 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.
6. Open the main file from the project explorer in the CCS and configure the general & use case settings.



```

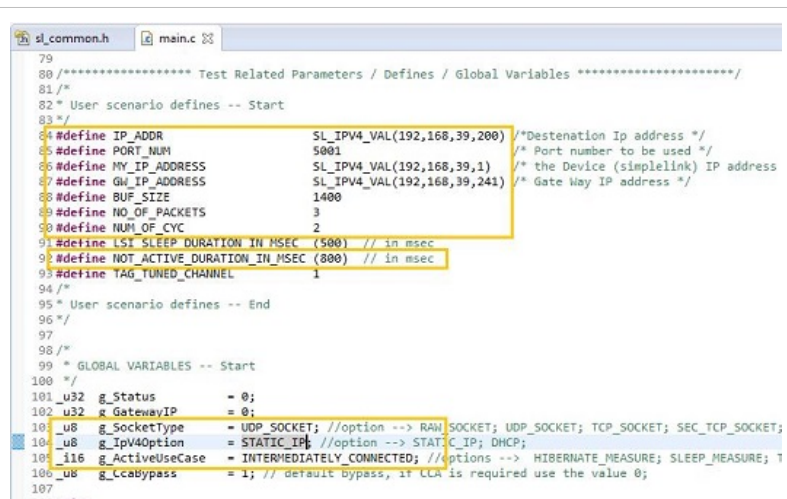
/* Values for below macros shall be modified per the access-point's
 * SimpleLink device will connect to following AP when the applicati
 */
#define SSID_NAME "cap_name" /* Access point name to
#define SEC_TYPE SL_SEC_TYPE_OPEN /* Secur
#define PASSKEY "" /* Password in case of s
#define PASSKEY_LEN my_strlen(PASSKEY) /* Password length in ch

/* Values for below macros shall be modified per the
 * SimpleLink device will connect to following AP whe
 */
#define SSID_NAME "Your_AP_Name_Here"
#define SEC_TYPE SL_SEC_TYPE_WPA /* Secur
#define PASSKEY "Your_AP_Security_Key_Here"
#define PASSKEY_LEN my_strlen(PASSKEY) /* Passwo

```

Setting AP info in the code

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.



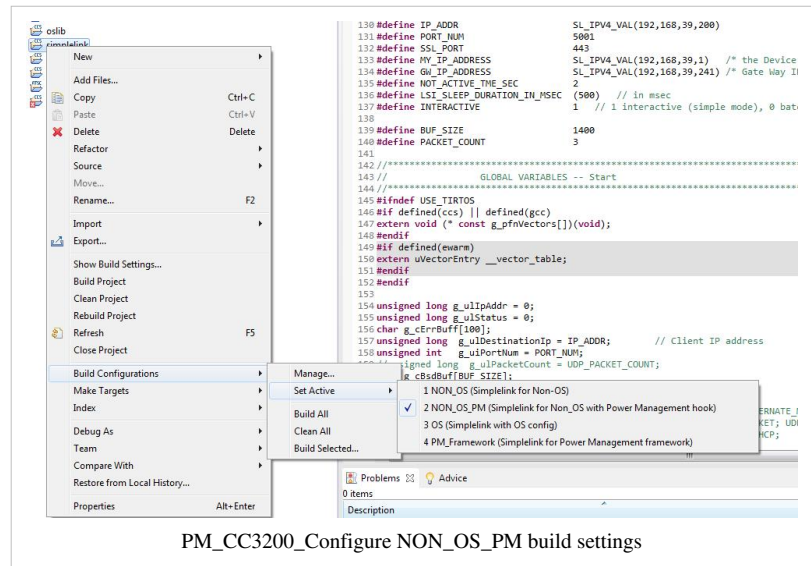
```

79
80 /***** Test Related Parameters / Defines / Global Variables *****/
81 /*
82 * User scenario defines -- Start
83 */
84 #define IP_ADDR SL_IPV4_VAL(192,168,39,200) /*Destenation Ip address */
85 #define PORT_NUM 5001 /* Port number to be used */
86 #define MY_IP_ADDRESS SL_IPV4_VAL(192,168,39,1) /* the Device (simplelink) IP address
87 #define GW_IP_ADDRESS SL_IPV4_VAL(192,168,39,241) /* Gate Way IP address */
88 #define BUF_SIZE 1400
89 #define NO_OF_PACKETS 3
90 #define NUM_OF_CYC 2
91 #define LSI_SLEEP_DURATION_IN_MSEC (500) // in msec
92 #define NOT_ACTIVE_DURATION_IN_MSEC (800) // in msec
93 #define TAG_TUNED_CHANNEL 1
94 /*
95 * User scenario defines -- End
96 */
97
98 /*
99 * GLOBAL VARIABLES -- Start
100 */
101 _u32 g_Status = 0;
102 _u32 g_GatewayIP = 0;
103 _u8 g_SocketType = UDP_SOCKET; //option --> RAW_SOCKET; UDP_SOCKET; TCP_SOCKET; SEC_TCP_SOCKET;
104 _u8 g_IpV4Option = STATIC_IP; //option --> STATIC_IP; DHCP;
105 _i16 g_ActiveUseCase = INTERMEDIATELY_CONNECTED; //options --> HIBERNATE_MEASURE; SLEEP_MEASURE; T
106 _u8 g_CcBypass = 1; // default bypass, if CLA is required use the value 0;
107

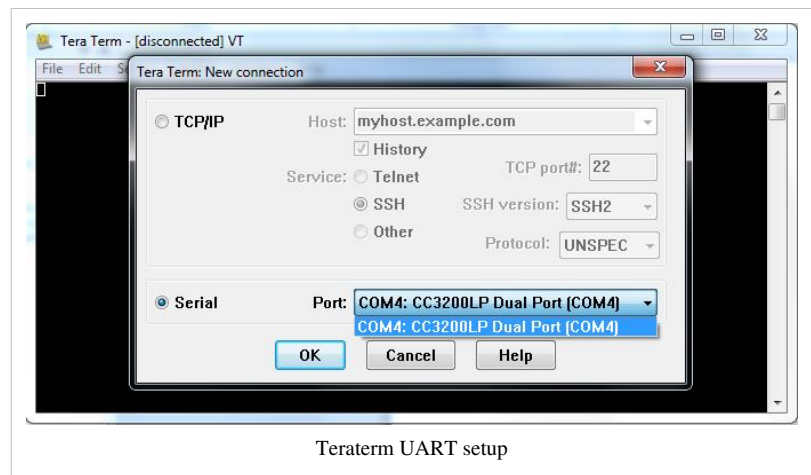
```

Global variables & define desired use cases

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 [5].
Important 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.
11. Open the J15 jumper (CC3200 Launchpad rev3p2) and press the reset button the application should start running.



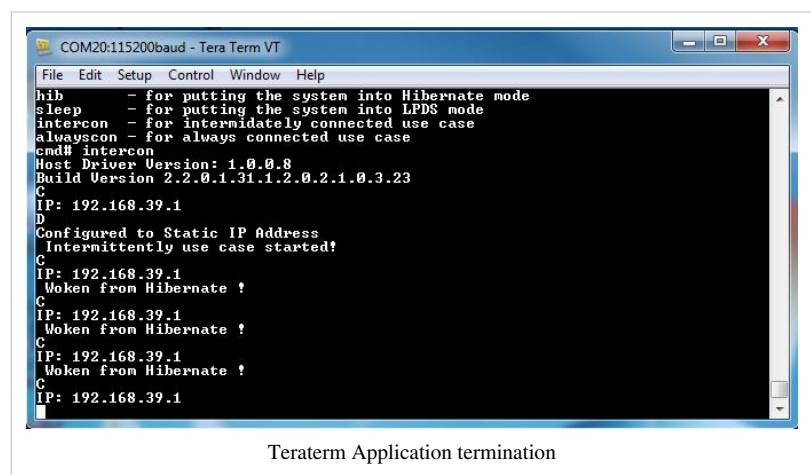
Current Consumption Measurements Setup

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:

$$\text{Energy: } E \text{ [Joules]} = \int V_{\text{supply}} \cdot I \, dt$$

Alternatively the charge consumption could be calculated:



Charge: $C [\text{coulomb}] = \int I dt$

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

Energy: $E [\text{Joules}] = \sum_{(x=1)}^n (V_{\text{supply}} * I_x * T_x)$

Alternatively the charge consumption could be calculated:

Charge: $C [\text{coulomb}] = \sum_{(x=1)}^n (I_x * T_x)$

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.

Oscilloscope with Current Probe

1. Tools Needed

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

2. FW Needed

1. Use TI PM benchmark code or your application FW

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 10mA).
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 3.3V supply at VBAT_CC pin and connect current probe across this supply line and measure current.

For the CC3200Launchpad : **Connecting wire across jumper(J12)** is not a good method, since this will alter the DCDCs inductor values.

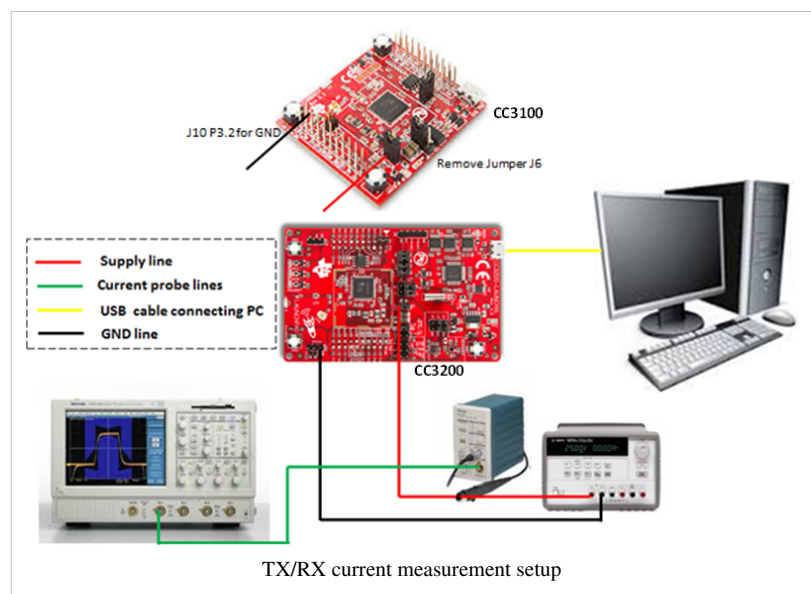
5. Check the current limit that you have set for the external supply, lower current limit may RESET the device.
6. Calibrate for voltage drop across the wire connecting supply to the device (use short and multi strand wire).

4. Procedure

1. Modifications for each board

For CC3100Boosterpack: **Remove jumper J6, Connect at Jumper J10, Pin_3.2 for GND.**

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.
 1. 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.
 2. To measure the peak current of transmit (TX) and receive (RX) modes use the “**Y**” axis cursors.

Mobile Communications DC source

1. Tools Needed

1. Agilent 66319D
2. Install NI Measurement and Automation explorer and Agilent 66319D GUI
3. NI GPIB-USB connector
4. Wires to connect instrument to device

2. FW Needed

1. Use TI benchmark code or your application FW

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.

There are three ranges: Low, Mid and HiGH. Select based on the requirement.

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.

4. Procedure

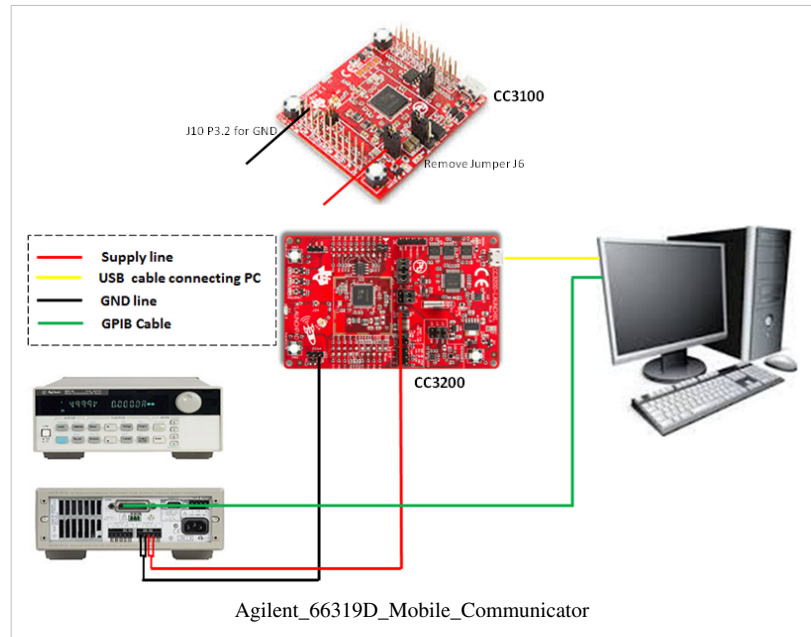
1. Modifications for each board

For CC3100Boosterpack: **Remove jumper J6, Connect at Jumper J10, Pin_3.2 for GND**

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...”.

On the pop window press “Auto-Detect”



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.

Static Current Measurements

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

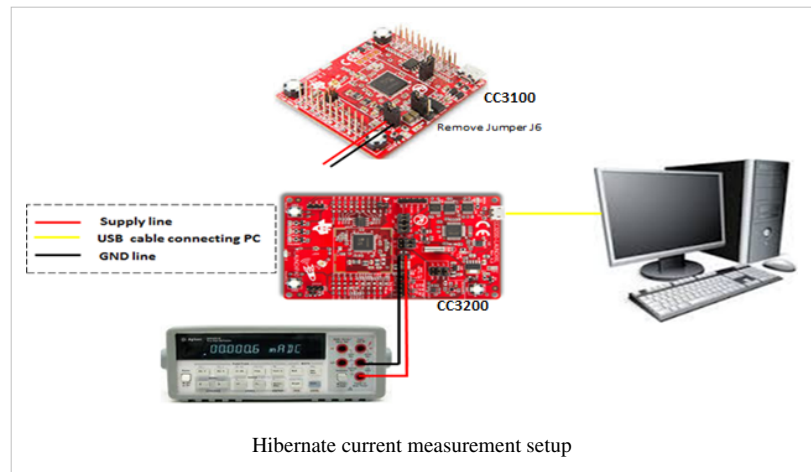
Digital Multimeter

1. Tools Needed

1. Digital multimeter(Agilent 34401A)
2. Short cables to connect multimeter in series with the device
3. Optional: Keysights Software [6]

2. FW Needed

1. Use TI benchmark code to generate the system modes
 - For Hibernate
 - For LPDS



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.

4. Procedure

1. Modifications for each board

For CC3100Boosterpack: **Remove jumper J6**

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.

Appendix

References

- [1] <http://www.ti.com/lit/pdf/swru376>
 - [2] <http://www.ti.com/tool/cc3200sdk>
 - [3] <http://www.ti.com/lit/pdf/swra462>
 - [4] <http://www.ti.com/product/CC3200/datasheet/specifications#SWAS03182390>
 - [5] http://processors.wiki.ti.com/index.php/CC31xx_%26_CC32xx_UniFlash#Service_Pack_Programming
 - [6] <http://www.keysight.com/main/software.jsp?ckey=2417463&lc=eng&cc=US&nid=-536902435.536880933&id=2417463>
-

Article Sources and Contributors

CC3200 Power Management Optimizations and Measurements *Source:* <http://processors.wiki.ti.com/index.php?oldid=202988> *Contributors:* A0221015, Beatrice, Jakesalmi, M-pahl, Made4engineering, Odaismadi

Image Sources, Licenses and Contributors

File:Cc31xx cc32xx return home.png *Source:* http://processors.wiki.ti.com/index.php?title=File:Cc31xx_cc32xx_return_home.png *License:* unknown *Contributors:* A0221015

File:Setting CC3200 to static hibernate current measure.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Setting_CC3200_to_static_hibernate_current_measure.jpg *License:* unknown *Contributors:* Beatrice

File:Setting CC3200 to static LPDS current measure.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Setting_CC3200_to_static_LPDS_current_measure.jpg *License:* unknown *Contributors:* Beatrice

File:PM_Always_Connected.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:PM_Always_Connected.jpg *License:* unknown *Contributors:* Beatrice

File:PM_Always_connected_config_example.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:PM_Always_connected_config_example.jpg *License:* unknown *Contributors:* Beatrice

File:CC3200 Always connected idle profile 1min.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:CC3200_Always_connected_idle_profile_1min.jpg *License:* unknown *Contributors:* Beatrice

File:CC3200 Always connected idle profile ZoomIn.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:CC3200_Always_connected_idle_profile_ZoomIn.jpg *License:* unknown *Contributors:* Beatrice

File:PM Intermittently Connected.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:PM_Intermittently_Connected.jpg *License:* unknown *Contributors:* Beatrice

File:PM Intermittently_connected_config_example.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:PM_Intermittently_connected_config_example.jpg *License:* unknown *Contributors:* Beatrice

File:CC3200 Intermittently connected 1 UDP packet.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:CC3200_Intermittently_connected_1_UDP_packet.jpg *License:* unknown *Contributors:* Beatrice

File:CC3200 Intermittently connected 1 SSL packet.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:CC3200_Intermittently_connected_1_SSL_packet.jpg *License:* unknown *Contributors:* Beatrice

File:PM Transceiver Mode.JPG *Source:* http://processors.wiki.ti.com/index.php?title=File:PM_Transceiver_Mode.JPG *License:* unknown *Contributors:* Beatrice

File:PM_transceiver_mode_config_example.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:PM_transceiver_mode_config_example.jpg *License:* unknown *Contributors:* Beatrice

File:PHY,MAC, L2 packet attributes config example.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:PHY,MAC,L2_packet_attributes_config_example.jpg *License:* unknown *Contributors:* Beatrice

File:CC3200 Transceiver Profile 1_min.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:CC3200_Transceiver_Profile_1_min.jpg *License:* unknown *Contributors:* Beatrice

File:Settings AP info in the code.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Settings_AP_info_in_the_code.jpg *License:* unknown *Contributors:* Beatrice

File:Global var and define desired use cases.jpeg *Source:* http://processors.wiki.ti.com/index.php?title=File:Global_var_and_define_desired_use_cases.jpeg *License:* unknown *Contributors:* Beatrice

File:PM_CC3200_Configure NON_OS_PM build settings.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:PM_CC3200_Configure_NON_OS_PM_build_settings.jpg *License:* unknown *Contributors:* Beatrice

File:CC3200_Teraterm UART setup.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:CC3200_Teraterm_UART_setup.jpg *License:* unknown *Contributors:* Beatrice

File:cc3200_Teraterm PM App running.jpg *Source:* http://processors.wiki.ti.com/index.php?title=File:Cc3200_Teraterm_PM_App_running.jpg *License:* unknown *Contributors:* Beatrice

File:CC3x00_Curr_meas_with_probe_and_oscilloscope.png *Source:* http://processors.wiki.ti.com/index.php?title=File:CC3x00_Curr_meas_with_probe_and_oscilloscope.png *License:* unknown *Contributors:* Beatrice

File:CC3x00_Curr_meas_with_Agilent_66319D_Mobile_Communicator.png *Source:* http://processors.wiki.ti.com/index.php?title=File:CC3x00_Curr_meas_with_Agilent_66319D_Mobile_Communicator.png *License:* unknown *Contributors:* Beatrice

File:CC3x00_Curr_meas_with_digital_multimeter.png *Source:* http://processors.wiki.ti.com/index.php?title=File:CC3x00_Curr_meas_with_digital_multimeter.png *License:* unknown *Contributors:* Beatrice