Measuring CC13xx and CC26xx current consumption

Joakim Lindh, Christin Lee, Marie Hernes and Siri Johnsrud

ABSTRACT

This application report describes the setup and procedures to measure power consumption on the CC13xx and CC26xx devices. It describes how this can be done both using a DC Power Analyzer or EnergyTrace™. Steps to analyze both a Bluetooth Low Energy peripheral and a device running the proprietary rfPacketTx example are included.

Contents

1 Introduction ................................................................................................................... 3
2 Standby ....................................................................................................................... 4
3 Understanding Bluetooth Low Energy Power Metrics .................................................. 7
4 SimpleLink Bluetooth Low Energy Wireless MCUs ..................................................... 8
5 Power Measurement Setup – Preparing the DUT ......................................................... 9
6 Measuring Power Consumption With a DC Power Analyzer ....................................... 15
7 EnergyTrace ............................................................................................................... 27
8 References .................................................................................................................. 36

List of Figures

1 VDDR Recharge ............................................................................................................. 4
2 Change of Recharge Interval Based on Standby Interval and Time From Reset ............. 4
3 Measuring Standby Current During Advertisement (CC26x0) ........................................ 5
4 Measuring Standby Current During Advertisement (CC26x2) ........................................ 5
5 Measuring Standby Current During Connection (CC26x0) .......................................... 6
6 Measuring Standby Current During Connection (CC26x2) .......................................... 6
7 Current Consumption vs. Time during a Bluetooth Low Energy Connection .................. 7
8 Device Under Test .................................................................................................... 9
9 CC2652R LaunchPad Jumper Removal .................................................................... 11
10 BTool Serial Port Settings ........................................................................................ 12
11 BTool Connection Settings ...................................................................................... 13
12 BTool Scan .............................................................................................................. 13
13 BTool Scan Results ................................................................................................. 14
14 BTool Establish Link ............................................................................................... 14
15 BTool Connected Device ......................................................................................... 15
16 Agilent N6705B DC Power Analyzer ........................................................................ 15
17 DUT Test Setup ....................................................................................................... 16
18 Agilent 14585A Control and Analysis Software, Start-Up .......................................... 17
19 Agilent 14585A Control and Analysis Software, Connect ......................................... 17
20 Agilent 14585A Control and Analysis Software, Context ......................................... 18
21 Agilent 14585A Control and Analysis Software, Source Settings ................................ 19
22 Connecting to the CC26x2r1 LaunchPad .................................................................. 20
23 Connected DUT to Agilent 14585A ......................................................................... 20
24 Agilent 14585A Control and Analysis Software, Scope ........................................... 21
List of Tables

1. Acronyms Used in This Document ................................................................. 3
2. CC2652R Supply Voltage .............................................................................. 8
3. Advertising Event, State Analysis ................................................................. 23
4. Beacon Event, State Analysis ...................................................................... 24
5. Connection Event, State Analysis ................................................................. 26

Trademarks

EnergyTrace, Code Composer Studio, SimpleLink, LaunchPad are trademarks of Texas Instruments. Arm, Cortex are registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. IAR Embedded Workbench is a trademark of IAR Systems AB. All other trademarks are the property of their respective owners.
1 Introduction

The first part of this application report focuses on Bluetooth Low Energy and shows how a DC Power Analyzer can be used to measure power consumption on a “Peripheral” device. The second half of this application report introduces you to the EnergyTrace technology, an energy-based code analysis tool that measures and displays the application’s energy profile. EnergyTrace technology is available as part of TI's Code Composer Studio™ IDE and the required HW is available on all CC13x2/CC26x2 LaunchPad Development kits.

Power consumption measurements are presented and battery life time is calculated for an example application. An accompanying Power Calculation Tool is provided so that you can estimate your battery life based on your own custom usage scenario.

Note that the results presented in this document are intended as guidelines and measurement results presented in this application report may not be up to date with the latest software optimizations. A variety of factors will influence the battery life of a Bluetooth Low Energy product. Measurements should be performed on hardware in a controlled environment and under the target application scenario.

It is assumed the reader of this document has some knowledge of the Bluetooth Low Energy standard, as well as the Texas Instruments SimpleLink™ Bluetooth Low Energy wireless MCUs with the Software Development Kit BLE-Stack. In addition, it is assumed that the reader has some knowledge of basic electrical engineering concepts and understands how to use laboratory test equipment such as an oscilloscope and DC power supply.

1.1 Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC</td>
<td>Analog to Digital Converter</td>
</tr>
<tr>
<td>BLE</td>
<td>Bluetooth Low Energy</td>
</tr>
<tr>
<td>CCS</td>
<td>Code Composer Studio</td>
</tr>
<tr>
<td>CM3</td>
<td>Cortex-M3</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>CSV</td>
<td>Comma-Separated Values</td>
</tr>
<tr>
<td>DC</td>
<td>Direct Current</td>
</tr>
<tr>
<td>DK</td>
<td>Development Kit</td>
</tr>
<tr>
<td>DUT</td>
<td>Device under Test</td>
</tr>
<tr>
<td>GAP</td>
<td>Generic Access Profile</td>
</tr>
<tr>
<td>GPIO</td>
<td>General-Purpose Input/Output</td>
</tr>
<tr>
<td>MCU</td>
<td>Micro Controller Unit</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
</tr>
<tr>
<td>RF</td>
<td>Radio Frequency</td>
</tr>
<tr>
<td>RTC</td>
<td>Real Time Clock</td>
</tr>
<tr>
<td>RTOS</td>
<td>Real Time Operating System</td>
</tr>
<tr>
<td>RX</td>
<td>Receive</td>
</tr>
<tr>
<td>SCA</td>
<td>Sleep Crystal Accuracy</td>
</tr>
<tr>
<td>SPI</td>
<td>Serial Peripheral Interface</td>
</tr>
<tr>
<td>TX</td>
<td>Transmit</td>
</tr>
</tbody>
</table>
2 Standby

Before we start looking into how we can measure current consumption, it is important to understand the Standby mode of the CC13xx and CC26xx devices. Standby is the lowest power mode where the CC13xx and CC26xx devices still have functionality other than maintaining I/O output pins. Standby is normally the power mode used between radio events if no other parts of the system are active. Current consumption in Standby mode consists of two parts: a recharge current pulse, used to charge up the VDDR capacitor, and the current consumption between the recharges. The latter is around 70 nA, almost too small to measure. It is the average power consumption during Standby including recharge that is defined as the Standby current, approximately 1 μA, as stated in the data manual for the given device (see [4] through [13]). Figure 1 shows what a recharge pulse looks like.

![Figure 1. VDDR Recharge](image)

While the CC13xx and CC26xx have a built-in comparator that gives the optimal recharge interval at any time (and at any temperature), the recharge pulses are dynamically adapted based on the required time in Standby for the CC13x0 and CC26x0 devices. For the latter devices, the recharge interval will also depend on when in time the measurements are done with respect to the last reset of the DUT. This is illustrated in Figure 2.

![Figure 2. Change of Recharge Interval Based on Standby Interval and Time From Reset](image)
In the first case, the Standby intervals (A, B, C, …) are short and there is only one recharge pulse between each wakeup. In this case, the Standby current will be higher than the 1 µA stated in the data manuals.

In the second case, the Standby intervals are longer (A*, B*, C*), and there is room for several recharge pulses within one Standby interval. In this case, the recharge interval will get a little bit longer for every recharge pulse. When starting a new Standby interval (B*), you will not get back to the minimum recharge interval, but start where you ended up in the previous Standby period (A*) (with some margins). Because of this, you will end up with the max recharge interval after a while (M*) (if your Standby intervals are long enough) and your Standby current will get down to 1 µA. In Figure 3, a CC26x0 is advertising with a 100 ms interval and there is one recharge in between the advertising events; in this case, the resulting Standby current is 1.57 µA. The Standby current will not go lower than this when advertising with a 100 ms connection interval. For CC26x2, you will not have any recharge pulses for this setup (see Figure 4) and the current consumption is below 60 nA. Recharge pulses will be observed if increasing the advertisement interval sufficiently.

Figure 3. Measuring Standby Current During Advertisement (CC26x0)

Figure 4. Measuring Standby Current During Advertisement (CC26x2)
When a connection has been established as described in Section 6.3.2, similar measurements can be done (still using CC26x0), resulting in a Standby current of 0.88 \( \mu \text{A} \) due to the long connection interval (1 s). In this case the recharge interval has increased and there are only 2 recharge pulses during the Standby period (see Figure 5). Measuring Standby current on the CC26x2 with the same connection interval results in a Standby current of 0.90 \( \mu \text{A} \) and there is only one recharge pulse during Standby (see Figure 6).
3 Understanding Bluetooth Low Energy Power Metrics

A Bluetooth Low Energy device achieves low power consumption by keeping radio activity short and allowing the device to reside in Standby or Power Down mode most of the operating time.

The operation of a Bluetooth Low Energy device is typically static in the sense that it's staying in a certain mode for a certain amount of time, for example, when advertising or maintaining a connection. These modes are based on re-occurring events that can easily be used to estimate average power consumption. Each of these modes can be quantified into states for future estimations based on added data throughput or reduced latency (through higher connection interval, as an example).

The primary metric is the average current for the advertising and connected mode. It is these values that can be used to determine the battery life of a Bluetooth Low Energy device.

For a wireless MCU it is important to understand that the device is typically not only running the Bluetooth Low Energy protocol stack, but also profile services and an application. The application may also be using peripherals on the chip, such as serial peripheral interface (SPI) or analog-to-digital converter (ADC). In addition, other devices on the circuit board, aside from the device running the Bluetooth Low Energy protocol stack, may be drawing current as well.

There are three main components of a Bluetooth Low Energy application that together sum up the average power consumption: Standby, Protocol events and Application events. Depending on the use case of the Bluetooth Low Energy device, these components will consume different amounts of power.

Figure 7 shows the current profile for a connected Bluetooth Low Energy device. The device spends most of the time in Standby, where the average current consumption is around 1 µA (see [4] through [13]).

![Figure 7. Current Consumption vs. Time During a Bluetooth Low Energy Connection](image)

From Standby, the device only wakes up based on either external interrupts or scheduled events/interrupts from the RTC. Standby also includes the recharge, which is described in Section 2.

The Protocol event is where communication over the Bluetooth Low Energy protocol occurs. For a Bluetooth Low Energy device, these events can be either Advertising events or Connection events. There are multiple roles featured that allow a Bluetooth Low Energy device to enter Observer role and scan as well but they are not covered in this application report.

The Application event is the application-specific implementation, for example, a periodic event, serial communication, or running algorithms based on sensor inputs. Depending on the amount of activity, the application event can increase power consumption significantly, hence, always aim to optimize processing usage. The Application events typically occur between protocol events, which mean that a longer advertising or connection interval gives longer time slots for processing.
4 SimpleLink Bluetooth Low Energy Wireless MCUs

There are several Bluetooth Low Energy Solutions provided by Texas Instruments. These cover a wide range of solutions, from simple broadcaster only to advanced multiple-role Real Time Operating System (RTOS) featured solutions. An overview of TI’s Bluetooth Low energy offering can be found here: http://www.ti.com/wireless-connectivity/simplelink-solutions/bluetooth-low-energy/products.html.

In the first part of this application report, measurements are done on the CC2652R [13], but everything discussed here regarding how to measure the current consumption is also valid for the other CC26xx and CC13xx devices.

The CC2652R is a Multi-Standard Wireless MCU providing a complete solution on a single chip. The application processor is an Arm® Cortex®-M4F and it is used for running the Bluetooth Low Energy Profiles along with any user defined functionality.

The RF core ensures that all timing regarding the Bluetooth Low Energy protocol is being configured and handled properly. An Arm Cortex-M0 is dedicated for the radio operations and runs the Bluetooth Low Energy Radio Firmware from its own dedicated ROM.

The CC2652R can be powered by two supply ranges, as presented in Table 2. To enable the 1.8 V system, both hardware and software modifications are required, which is documented in the CC13x2, CC26x2 SimpleLink™ Wireless MCU Technical Reference Manual [3]. For the CC13x0/CC26x0 device, see the CC13x0, CC26x0 SimpleLink™ Wireless MCU Technical Reference Manual [2].

<table>
<thead>
<tr>
<th>Supply Voltage</th>
<th>Internal DCDC</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.8 V System (External Regulator Mode)</td>
<td>No</td>
<td>1.7 V</td>
<td>1.95 V</td>
</tr>
<tr>
<td>3.3 V System</td>
<td>Optional</td>
<td>1.80 V</td>
<td>3.80 V</td>
</tr>
</tbody>
</table>

For more information about Supply Voltage, see the device-specific user’s manual (Section 8).
5 Power Measurement Setup – Preparing the DUT

Before measurement and analysis can be performed, the device under test (DUT) must be prepared both from a hardware and software perspective. A peer device can also be configured in order to establish a connection. In this application report, a device running the example project HostTest [15] is used to establish the connection. This project can be run on any BLE-enabled development board and we have used a CC26x2R LaunchPad™ [19].

![Figure 8. Device Under Test](image)

5.1 Requirements

To measure average power consumption for Bluetooth Low Energy on CC26xx, the following hardware from Texas Instruments can be used:

- CC26x2R LaunchPad
- CC2650/CC2640R2/CC1352R/CC1352P LaunchPad [17]/[18]/[20]/[21]
- A device running the HostTest project - Optional

The above mentioned hardware can be purchased at the TI Store [14].

In terms of software resources, the following are required:

- BLE-Stack [15]
- IAR EWARM [16]
  or
- CCS Integrated Development Environment [22]

Make sure you are using the IAR and CCS version recommended for your version of the SDK/BLE-Stack. For more details, see the device-specific software release notes.
5.2 Embedded Software

The BLE-stack is either a stand-alone deliverable or provided as part of the Software Development Kit (SDK), depending on which device you use. The software package includes the full Bluetooth Low Energy protocol stack along with sample applications. The protocol stack is provided as a pre-qualified library component and the complete system is operated by an RTOS that introduces a threaded environment with full power management. The power management is maintained by the Power Driver automatically and the application can constrain tasks or disallow certain power modes, if required. The current consumption for the different power modes can be found in the device-specific data manual (Section 8).

The generic sample application simple_peripheral that is included with the BLE-Stack is ideal to use in order to analyze power consumption for the sole Bluetooth Low Energy protocol running on a wireless MCU. Depending on what version of simple_peripheral you are using, the application does one or more of the following:

- Advertise with legacy advertisements on LE 1M PHY
- Advertise with extended advertisements on LE Coded PHY
- Start a clock that calls a periodic event
- Print messages on UART display

For the CC2650 LaunchPad, the external flash is not turned off by default, hence you will measure an extra current consumption of about 7 µA. The flash can be turned off by calling the ExtFlash_open(); followed by ExtFlash_close(); (functions found in ExtFlash.h/ExtFlash.c). To get a power measurement that is easy to interpret, you should disable all application behaviors described above except legacy advertising, and make sure the external flash on your board is shut down.

For more information including instructions on how to program the CC26x0, see the CC2640 and CC2650 SimpleLink™ Bluetooth® low energy Software Stack 2.2.1 Developer's Guide [1]. For CC13x2/CC26x2, see the BLE5-Stack User's Guide [23].
5.3 Hardware

5.3.1 CC26x2R LaunchPad

To get a clean current measurement, the jumpers on the LaunchPad should be removed. The CC2652R Launchpad with all jumpers removed is shown in Figure 9. Note that when the JTAG jumpers are removed, the programming and debug capabilities of the chip become unavailable. This is also applicable for the other CC26xx and CC13xx LaunchPads ([17] - [21]).

![Figure 9. CC2652R LaunchPad Jumper Removal](image-url)
5.4 **BTool (Optional)**

The BLE-Stack also includes BTool along with drivers and firmware. BTool can be used to emulate a Bluetooth Low Energy application from a PC environment. BTool is used to create a connection with the DUT. If the intention is to measure power consumption of the DUT when being in advertising or beacon mode, BTool is not required.

To connect to the DUT using BTool, a device running a Bluetooth Low Energy wireless network processor image named HostTest is required. This can be found with the other example projects in the BLE-Stack.

With a Launchpad running the HostTest application connected to the PC, open BTool and select the COM port used by the application (see Figure 10).

![Figure 10. BTool Serial Port Settings](image)

Press OK and there should be an initialization process that is observed in the log window.

Before forming the connection, the proper connection parameters should be used. This is dependent on the application that is being considered. The supervision timeout setting should not affect the power measurements. A connection interval of one second, with zero slave latency, is used in this document. Therefore, use the values as shown in Figure 11. Be sure to select the “Set” button after entering in the values. Setting up the connection parameters needs to be done before a connection is established.
With the connection parameters set as needed, setup is completed.

At this point, BTool is ready to discover the DUT. If you left the SimpleBLEPeripheral application running on your DUT, you should be ready to use BTool. As long as the device running SimpleBLEPeripheral is powered up and not connected to anything, it should be in discoverable (advertising) mode.

In the Discovery section, press the “Scan” button, as shown in Figure 12.

Figure 11. BTool Connection Settings

Figure 12. BTool Scan
BTool will begin searching for Bluetooth Low Energy devices. When the discovery process finishes, the address of any scanned devices will appear in the “Slave BDA” section, as shown in Figure 13.

![Figure 13. BTool Scan Results](image)

To establish a connection with the peripheral device, select the address of the device to connect with and click the “Establish” button, as shown in Figure 14.

![Figure 14. BTool Establish Link](image)
As long as the peripheral is powered-up and still in discoverable mode, a connection should be established immediately. Once a connection is established, the message window will return a “GAP_EstablishLink” event message with a “Status” value of “0x00 (Success)”. In BTool, you can see your connected peripheral device in the Device Information field, as shown in Figure 15.

Figure 15. BTool Connected Device

6 Measuring Power Consumption With a DC Power Analyzer

The most accurate way of measuring power consumption is to use a DC Power Analyzer (since the power consumption varies over time, a simple multimeter will not be sufficient). An oscilloscope can be used as well, as long as the sampling rate and bandwidth is good enough. For the purpose of this application report, an Agilent N6705B DC Power Analyzer is used (see Figure 16). The internal module is a N6781A, a 2-quadrant source and measure unit for battery drain analysis.

Figure 16. Agilent N6705B DC Power Analyzer
6.1 Test Setup

Make sure that the system is set up properly and review the steps described in Section 5. For reference, the full overview is illustrated in Figure 17. VDD is connected to the 3V3 pin on the LaunchPad.

When the DUT is correctly connected, the power supply is enabled by pressing the “On” button within the Agilent 14585A Control and Analysis Software. The power consumption measurements can be done by two separate functions: Scope or Data Logger. The Data Logger provides an average power consumption measurement over longer time, for example, minutes and hours, although the resolution is not as good as using Scope. This document focuses on doing measurements by using the Scope feature.

The Agilent N6705B powers the DUT as well as performs the current measurement.
6.1.1 Analysis Software Setup

All measurements and analysis can be done directly with the Agilent N6705B interface, but in this application report, a PC Tool called "14585A Control and Analysis Software for Advanced Power Supplies" (v2.0.2.1) is used to control the Agilent N6705B. The software is available from http://www.keysight.com (in 2014, Agilent electronics instruments division was acquired by Keysight Technologies).

When the PC Tool is started, no external equipment is connected, which is observed in the "Instrument Control Tab", as shown in Figure 18.

![Figure 18. Agilent 14585A Control and Analysis Software, Start-Up](image)

To connect the Agilent N6705B, make sure that it is connected via USB and that it is powered. Use the bottom left “Connect” button to select the connected hardware, as shown in Figure 19.

![Figure 19. Agilent 14585A Control and Analysis Software, Connect](image)
When the hardware has been successfully connected, it is fully controlled from the PC Tool, which is verified by the “Instrument Control" tab, as shown in Figure 20.

Figure 20. Agilent 14585A Control and Analysis Software, Connected
Note that the Output may be “On” per default (observed by the lit “on” button). If so, turn the Output off since the actual output parameters have not been configured yet. The next step is to configure the output. In the “Instrument Control” tab, click the “Settings” button to bring up the Source Settings for Output 1. Depending on the module within the Agilent N6705B, the options may be limited. Select “2 Quadrant Power Supply” and set the “Voltage” to 3.0 V.

![Agilent 14585A Control and Analysis Software, Source Settings](image)

Figure 21. Agilent 14585A Control and Analysis Software, Source Settings
Connect the instrument probes to the DUT. For the LaunchPads, the VDD line should be connected to the 3V3 pin. The GND can be connected to any GND pin. Connecting to the CC26x2R1 LaunchPad is shown in Figure 22.

![Figure 22. Connecting to the CC26x2r1 LaunchPad](image)

When the DUT is correctly connected, the power supply is enabled by pressing the “On” button within the Agilent 14585A Control and Analysis Software. The power consumption measurements can be done by two separate functions: Scope or Data Logger. The Data Logger provides an average power consumption measurement over longer time, for example, minutes and hours, although the resolution is not as good as using Scope. This document focuses on doing measurements by using the Scope feature.

![Figure 23. Connected DUT to Agilent 14585A](image)
6.2 Measurement Using Scope

When the instrument has been correctly setup and configured, make sure that Scope has been selected, as shown in Figure 24.

![Figure 24. Agilent 14585A Control and Analysis Software, Scope](image)

The scope mode allows that measurement be ran over a short amount of time. In order to maximize the amount of data, use the following measurement setup: (see Figure 25).

- Time/div: 200 ms/
- Points: 512k
- Trigger: Scope Run Button
- Mode: Single
- Slope: Rising Edge

![Figure 25. Agilent 14585A Control and Analysis Software, Scope Setup](image)

Next, make sure that the “Ranges…” is setup to Auto, as shown in Figure 26.

![Figure 26. Agilent 14585A Control and Analysis Software, Instrument Range](image)

The instrument should now be setup properly and the measurement can start. Click the Play button in the bottom right corner → allow the instrument to start the measurement.
6.3 Analysis

Depending on what the DUT is setup to do, the result will vary. If no interaction has been made with the DUT, it will be sending out periodic advertisements each 100 ms (see Figure 27).

![Figure 27. Agilent 14585A Control and Analysis Software, Advertisement Capture (CC26x2)](image)

The approximately 2.6 s measurement includes 26 advertising events. There are no recharge pulses in Standby, as the measurement is done on CC26x2 (see Section 2).

There is functionality to do detailed measurements of the acquired waveform. Select “Markers & Measurements” to enable the markers. There are two approaches of using the markers:

- Measure the average power consumption from a symmetric point of the measurement, (for example, from the start of an event to the point where the next event starts). This will give an approximation of the overall power consumption over time because of the reoccurring symmetry.
- Break down the events into states to be used for various use case studies and estimations. This is very useful in order to analyze the resulting power consumption when intervals are changed.

If the objective is to simply obtain a power consumption figure of the DUT, the first option is fast and reliable.

6.3.1 Advertising Event

An advertising event is where the (Bluetooth Low Energy) peripheral device broadcasts information in order to either share information or become connected to a (Bluetooth Low Energy Ready) Central device, such as a smart phone. The device wakes up and broadcasts packets on three separate channels and listens on each of these channels for Scan Requests or Connection Requests. Scan Requests is a way for a Central device to obtain more information about the device before connecting, because the advertising data is typically chosen to be very short to minimize power consumption. Based on advertising data or the scan response data, connection requests can be sent, which initiates a connection between the Peripheral and the Central.

With connectable advertising packet format, the base time of data transmitting is 144 µs, which contains a pilot tone plus 1 byte preamble, 4 bytes Access Address, 2 bytes PDU, 3 bytes CRC and 6 bytes AdvA in the payload. For every additional transmitted bit, 1 µs should be added to the TX time.
The Agilent 14585A Control and Analysis Software “Markers & Measurements” functions are used to quantify a single advertising event, which is visualized in Figure 28 and summarized in Table 3.

![Figure 28. Connectable Advertising Event, Capture](image)

**Table 3. Advertising Event, State Analysis**

<table>
<thead>
<tr>
<th>Number</th>
<th>State</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Pre-processing</td>
<td>RTOS wake-up, radio setup, XTAL guard time</td>
</tr>
<tr>
<td>2</td>
<td>Radio preparation</td>
<td>Radio is turned on and in transition to TX</td>
</tr>
<tr>
<td>3</td>
<td>TX</td>
<td>The radio transmits an advertisement packets with 3 bytes data on Channel 37. Time is dependent on amount of transmitted data</td>
</tr>
<tr>
<td>4</td>
<td>TX to RX transition</td>
<td>TX to RX transition</td>
</tr>
<tr>
<td>5</td>
<td>RX</td>
<td>Time depends on advertising interval and Sleep Crystal Accuracy (SCA)</td>
</tr>
<tr>
<td>6</td>
<td>RX to TX transition</td>
<td>RX to TX transition</td>
</tr>
<tr>
<td>7</td>
<td>TX</td>
<td>The radio transmits an advertisement packets with 3 bytes data on Channel 38. Time is dependent on amount of transmitted data</td>
</tr>
<tr>
<td>8</td>
<td>TX to RX transition</td>
<td>TX to RX transition</td>
</tr>
<tr>
<td>9</td>
<td>RX</td>
<td>Time depends on advertising interval and SCA</td>
</tr>
<tr>
<td>10</td>
<td>RX to TX transition</td>
<td>RX to TX transition</td>
</tr>
<tr>
<td>11</td>
<td>TX</td>
<td>The radio transmits an advertisement packets with 3 bytes data on Channel 39. Time is dependent on amount of transmitted data</td>
</tr>
<tr>
<td>12</td>
<td>TX to RX transition</td>
<td>TX to RX transition</td>
</tr>
<tr>
<td>13</td>
<td>RX</td>
<td>Time depends on advertising interval and SCA</td>
</tr>
<tr>
<td>14</td>
<td>Post-processing and going to Standby</td>
<td>Bluetooth Low Energy protocol stack processes the received packets and sets up the sleep timer in preparation for the next event, and then goes to Standby</td>
</tr>
</tbody>
</table>
This is also the event occurring when a device is in beacon mode. For a non-connectable beacon, there are no RX states during the advertising event, reducing the average current consumption.

![Figure 29. Beacon Event, Capture](image)

**Table 4. Beacon Event, State Analysis**

<table>
<thead>
<tr>
<th>Number</th>
<th>State</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Pre-processing</td>
<td>RTOS wake-up, radio setup, XTAL guard time</td>
</tr>
<tr>
<td>2</td>
<td>Radio preparation</td>
<td>Radio is turned on and in transition to TX</td>
</tr>
<tr>
<td>3</td>
<td>TX</td>
<td>The radio transmits an advertisement packets with 3 bytes data on Channel 37. Time is dependent on amount of transmitted data</td>
</tr>
<tr>
<td>4</td>
<td>TX-to-TX transition</td>
<td>TX to TX transition</td>
</tr>
<tr>
<td>5</td>
<td>TX</td>
<td>The radio transmits an advertisement packets with 3 bytes data on Channel 38. Time is dependent on amount of transmitted data</td>
</tr>
<tr>
<td>6</td>
<td>TX-to-TX transition</td>
<td>TX to TX transition</td>
</tr>
<tr>
<td>7</td>
<td>TX</td>
<td>The radio transmits an advertisement packets with 3 bytes data on Channel 39. Time is dependent on amount of transmitted data</td>
</tr>
<tr>
<td>8</td>
<td>Post-processing and going to Standby</td>
<td>Bluetooth Low Energy protocol stack sets up the sleep timer in preparation for the next event, and then goes to Standby</td>
</tr>
</tbody>
</table>

### 6.3.2 Connection Event

When a connection has been established between a Peripheral and a Central device, they communicate during connection events. The Central device operates as the master and the Peripheral device as the slave.

All communication between two connected devices occurs on these connection events. They are periodically with a configurable connection interval, ranging from 7.5 ms to 4 s.

Each event occurs on one of the 37 data channels and the master always initiates the event, with the slave listening. They can continue to communicate back and forth as much as they want during one connection event.

Connection events occur even if neither side has data to send. This ensures that the link is still valid. If a specified number of connection events occur without acknowledgment, the connection will be considered lost.
In order to measure the current consumption during a Connection event, the DUT must be connected to a peer device. By using BTool as described in Section 5.4, a connection with 1 second connection interval is established.

To measure the average current consumption after a connection has been established (Figure 6), you should identify two Connection events. Place Marker 1 right after a Connection event, as shown in Figure 30, and the second marker after the following Connection event. The average current is in this case approximately 10.2 µA, as shown in Figure 31.
The Connection event can also be analyzed (like what was done with the Connectable Advertising event in Figure 28) by selecting “Markers & Measurements”. This is shown in Figure 32 and summarized in Table 5.

![Figure 32. Current Consumption versus Time During a Single Connection Event](image)

### Table 5. Connection Event, State Analysis

<table>
<thead>
<tr>
<th>Number</th>
<th>State</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Pre-processing</td>
<td>RTOS wake-up, radio setup, XTAL guard time</td>
</tr>
<tr>
<td>2</td>
<td>Radio preparation</td>
<td>Radio is turned on and in transition to RX</td>
</tr>
<tr>
<td>3</td>
<td>Receive (RX)</td>
<td>The radio receiver listens for a packet from the master. Time depends on connection interval and SCA.</td>
</tr>
<tr>
<td>4</td>
<td>RX to TX transition</td>
<td>RX to TX transition</td>
</tr>
<tr>
<td>5</td>
<td>Transmit (TX)</td>
<td>The radio transmits a packet to the master on one of the 37 channels. Time is dependent on the amount of transmitted data</td>
</tr>
<tr>
<td>6</td>
<td>Post-processing and going to Standby</td>
<td>Bluetooth Low Energy protocol stack processes the received packets and sets up the sleep timer in preparation for the next event, and then goes to Standby</td>
</tr>
</tbody>
</table>

### 6.3.3 Power Consumption Calculator

The state analysis from advertising and connection states can be used to investigate how the battery life varies depending on connection interval. For that purpose, a Power Calculator Tool [24] is provided that can be used to perform calculations for your custom application.
EnergyTrace

EnergyTrace is available on all CC13x2 and CC26x2 LaunchPads. The tool can be used stand-alone, as a power profiling tool, or in EnergyTrace++ mode (4-pin JTAG required) within a debug session for state monitoring to help optimize the application for ultra-low-power consumption. This section focuses on the steps necessary to run EnergyTrace in stand-alone-mode in CCS. In this mode, the debugger is not active and the displayed current consumption is what to expect for the final application. Since the previous sections have focused on Bluetooth Low Energy, this section will use one of the proprietary examples to show how to use EnergyTrace. Please note that the CC13x2 Proprietary RF User’s Guide [26] has a section with info regarding EnergyTrace.

The rfPacketTx example (available for our CC13x2 devices) was used running on the CC1352R1 LunchPad [20].

The example can be found and downloaded using Resource Explorer (see Figure 33).

Figure 33. rfPacketTx in Resource Explorer
After building the example, it should be downloaded to the LaunchPad and the following steps should be done:

1. Remove all jumpers on the LaunchPad between the XDS debugger and the device, except for the XDS110 Power, the 3V3 and RXD jumpers (see Figure 34). It is important that the XDS110 jumper is mounted in the “XDS110 Power” position when the LaunchPad is powered up, otherwise the calibration of EnergyTrace will fail. The UART driver in the SDK configures the UART RX pin without internal pull-up. To avoid current leakage in the input buffer, the pin must always be firmly pulled to a logic level. This can be achieved by keeping the RXD jumper on (connecting the debugger output to the UART RXD input).

![Figure 34. Jumper Settings](image)

2. If the jumpers were not set correctly BEFORE powering the board, trigger a re-calibration of EnergyTrace by power cycling the LaunchPad (disconnect and re-connect the micro-USB cable).

3. EnergyTrace requires some configurations the first time it is being used within a CCS workspace. Go to the menu `Window` and select `Preferences` (see Figure 35).

![Figure 35. Preferences](image)
4. Navigate to the ‘EnergyTrace Technology’ window and configure it as shown in Figure 36.

**Figure 36. EnergyTrace Technology Configuration**

If post-processing of the acquired data is wanted, select the ‘Raw data to CSV file’ checkbox. If this checkbox is selected, you can, after EnergyTrace is finished capturing data, select the ‘Save current energy profile’ button, to save a .csv file. The default location for this file will be under your project workspace.
5. Click the EnergyTrace Button (see Figure 37).

![Figure 37. EnergyTrace Button](image.png)

A dialog with instructions on how to use the EnergyTrace Stand-alone Measurement Mode will pop-up. Click ‘Proceed’ to continue.

6. Select how long you want to capture data by clicking the ‘Set Measurement Duration’ button (see Figure 38).

![Figure 38. Set Measurement Duration](image.png)
7. Click the green play button to start capturing data (see Figure 39). The red LED on the XDS110 debugger should be turned on, and will be so for the duration of the EnergyTrace capture.

![Start Trace Collection](image)

**Figure 39. Start Trace Collection**

8. When EnergyTrace is finished capturing data, review the application’s power profile and have a closer look at the current graph.

*Figure 40* shows the current profile for the rfPacketTX example taken over 1 s. From the plot, it is easy to identify the packet interval of 500 ms and to verify that the device enters Standby in-between packets. If you want to zoom in on the current graph, you can use the magnifying glass symbol.

![Current Profile](image)

**Figure 40. Current Profile for rfPacketTx (without modifications)**
7.1 Modifying the rfPacketTX Example

Using SmartRF™ Studio [25], you can find the modifications that need to be done in smartrf_settings.c to change, for example, the output power to 10 dBm (see Figure 41).

![Figure 41](image1.png)

**Figure 41. Using SmartRF Studio to Find New Settings**

After changing the output power by modifying the smartrf_settings.c file, as shown in Figure 42, the measurements were repeated.

![Figure 42](image2.png)

**Figure 42. Modifying the smartrf_settings.c File**
Figure 43 shows the current profile for transmitting a packet achieved with using EnergyTrace. It shows a TX power consumption at +10 dBm (3.3 V) of 14.78 mA. Figure 44 shows the same profile, captured using the DC Power Analyzer. It shows a current consumption in TX of 14.4 mA. The data sheet numbers [7] for TX at +10 dBm is 14.3 mA.

Even if the numbers you get when using EnergyTrace is not as accurate as the ones obtained with the DC Power Analyzer, the numbers will be very useful when estimating current consumption for an application.

![Figure 43. Current Profile of TX (EnergyTrace)](image1)

![Figure 44. Current Profile of TX (DC Power Analyzer)](image2)

It is not possible to accurately measure the Standby current when using EnergyTrace, but you can still use it to verify that the device is in Standby. As described in Section 2, the CC13x2/CC26x2 have a built-in comparator that gives the optimal recharge interval at any time. In the previous measurements, no re-charge pulses have been seen.

If the packet interval, and hence the time in Standby, is doubled the recharge pulses should appear. In the rfPacketTX example, this can easily be done by including the POWER_MEASUREMENT define in rfPacketTx.c.
The packet interval is then changed from 500 ms to 5 s. The PACKET_INTERVAL also needs to change from 5 to 1.

```
/* Do power measurement */
#define POWER_MEASUREMENT

/* Packet TX Configuration */
#define PAYLOAD_LENGTH 30
#ifdef POWER_MEASUREMENT
    #define PACKET_INTERVAL 1 /* For power measurement set packet interval to 1 s */
#else
    #define PACKET_INTERVAL 500000 /* Set packet interval to 500000 us or 500 ms */
#endif
```

With these modifications, the re-charge pulses are easy to identify during the time in Standby (see Figure 45).

![Figure 45. Recharge Pulses](image)

For the average current consumption of an application, it is very important that the device always enters the lowest possible power mode. If, for some reason, there were things in the rfPacketTX examples that prevented the device from entering Standby in-between packets, the average current consumption would increase significantly. For the current profile shown in the previous code example, the average current consumption is 0.1 mA (see Figure 46).

![Figure 46. Average Current Consumption When Device Enters Standby](image)
To see how the current consumption will look like if the device is not entering Standby, a power constraint is set to disallow this power mode. This can be done with the following modification:

```c
#include <ti/drivers/Power.h>
#include <ti/drivers/power/PowerCC26X2.h>

void *mainThread(void *arg0)
{
    RF_Params rfParams;
    RF_Params_init(&rfParams);

    Power_setConstraint(PowerCC26XX_SB_DISALLOW); // Prevent the device from entering Standby
```

Figure 47 shows what the current profile looks like, with an average current consumption of 1 mA (see Figure 48).

![Figure 47. IDLE State Between Packets](image)

![Figure 48. Average Current Consumption When Device Does not Enter Standby](image)
References

1. Texas Instruments: *CC2640 and CC2650 SimpleLink™ Bluetooth® low energy Software Stack 2.2.1 Developer's Guide*
2. Texas Instruments: *CC13x0, CC26x0 SimpleLink™ Wireless MCU Technical Reference Manual*
5. Texas Instruments: *CC1312R SimpleLink™ High-Performance Sub-1 GHz Wireless MCU Data Manual*
7. Texas Instruments: *CC1352R SimpleLink™ High-Performance Dual-Band Wireless MCU Data Manual*
10. Texas Instruments: *CC2640R2F SimpleLink™ Bluetooth® low energy Wireless MCU Data Manual*
11. Texas Instruments: *CC2642R SimpleLink™ Bluetooth® 5 low energy Wireless MCU Data Manual*
12. Texas Instruments: *CC2650 SimpleLink™ Multistandard Wireless MCU Data Manual*
13. Texas Instruments: *CC2652R SimpleLink™ Multiprotocol 2.4-GHz Wireless MCU Data Manual*
14. TI Store: [https://store.ti.com/](https://store.ti.com/)
15. BLE-Stack™: [www.ti.com/ble-stack](http://www.ti.com/ble-stack)

Measuring CC13xx and CC26xx current consumption
## Revision History

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

### Changes from C Revision (January 2017) to D Revision

<table>
<thead>
<tr>
<th>Item</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>• The complete document has been updated to cover the CC13x2/CC26x2 devices and references to the old hardware has been removed.</td>
<td>1</td>
</tr>
<tr>
<td>• Added new Section 7.</td>
<td>27</td>
</tr>
</tbody>
</table>
IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), 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, 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 (www.ti.com/legal/termsofsale.html) 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.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2019, Texas Instruments Incorporated