Application Note

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 C26xx 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.

Table of Contents

List of Figures

List of Tables

Trademarks

EnergyTrace™, Code Composer Studio™, SimpleLink™, LaunchPad™, and SmartRF™ are trademarks of Texas Instruments.
IAR Embedded Workbench™ is a trademark of IAR Systems AB.
Arm® and Cortex® are registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere.
All 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 1-1. Acronyms Used in This Document

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

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 2-1 shows what a recharge pulse looks like.

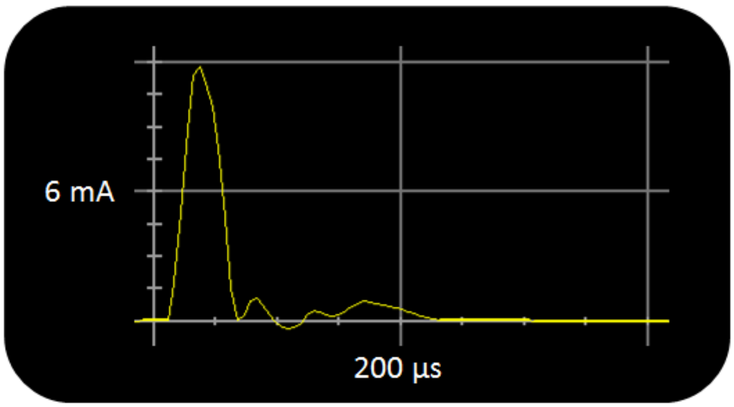


Figure 2-1. VDDR Recharge

While the CC13x2 and CC26x2 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 2-2.

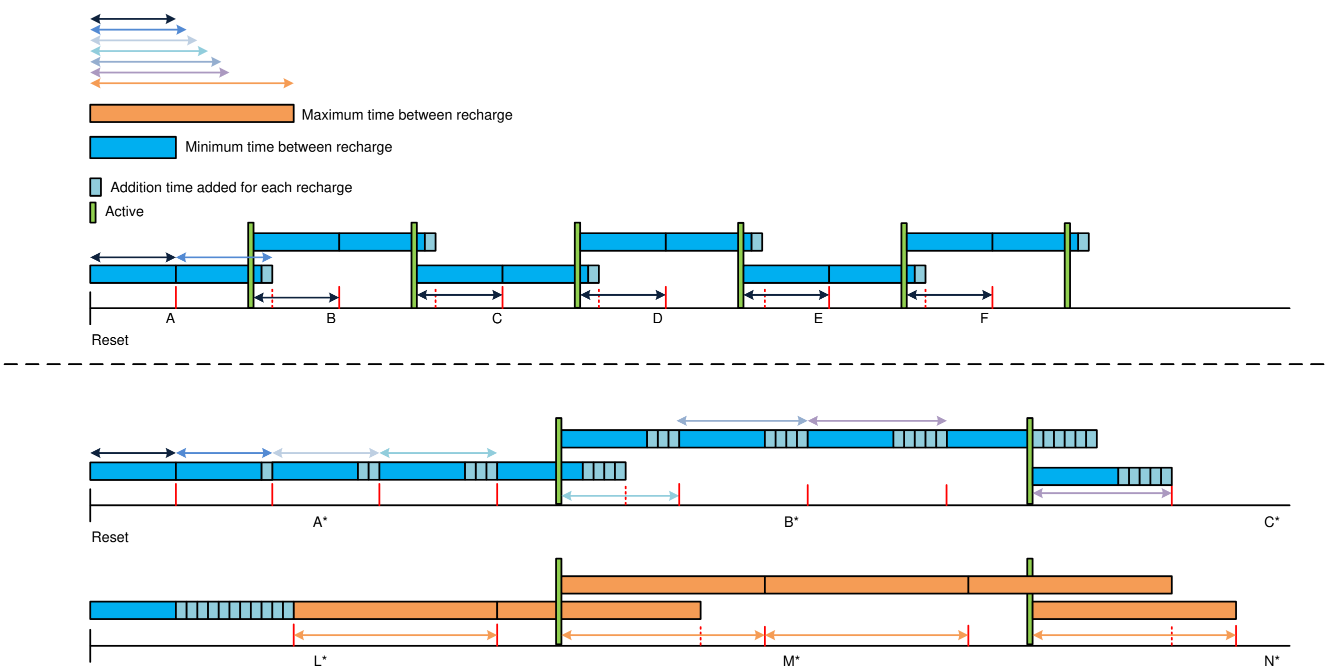


Figure 2-2. Change of Recharge Interval Based on Standby Interval and Time From Reset

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 re-charge 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 2-3, 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 2-4) and the current consumption is below 60 nA. Recharge pulses will be observed if increasing the advertisement interval sufficiently.

Figure 2-3. Measuring Standby Current During Advertisement (CC26x0)

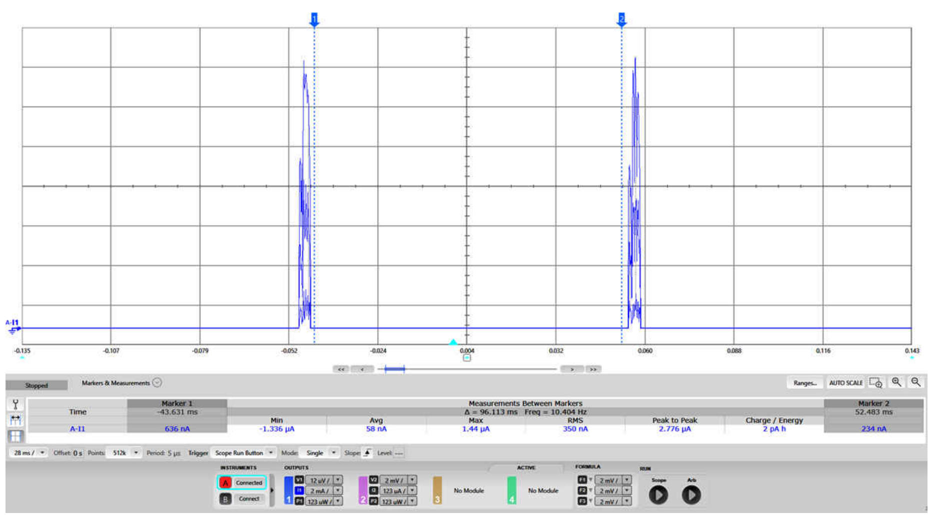


Figure 2-4. Measuring Standby Current During Advertisement (CC26x2)

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 µA 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 2-5). Measuring Standby current on the CC26x2 with the same connection interval results in a Standby current of 0.90 µA and there is only one recharge pulse during Standby (see Figure 2-6).

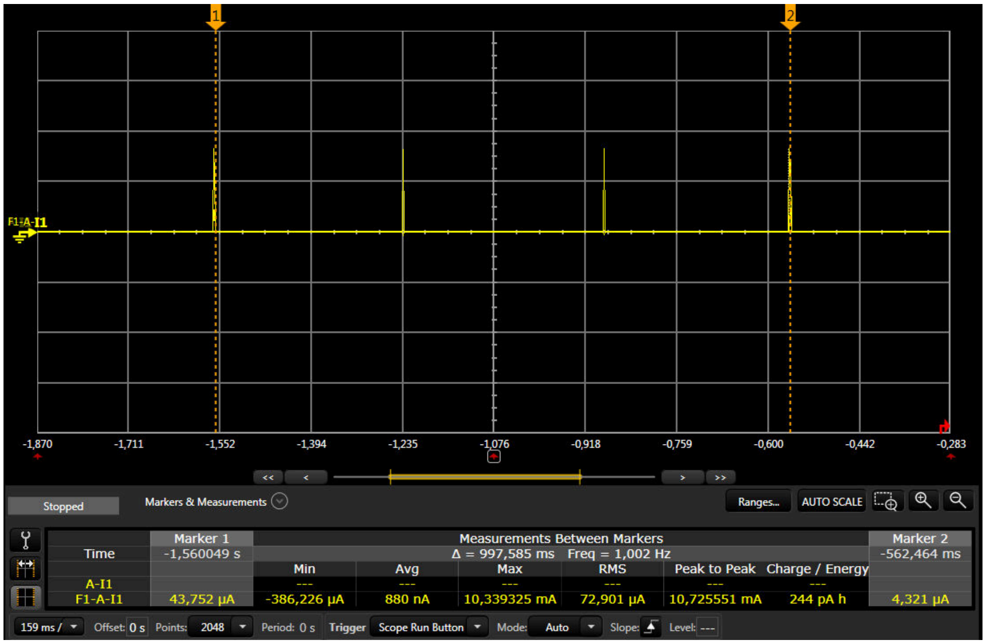


Figure 2-5. Measuring Standby Current During Connection (CC26x0)

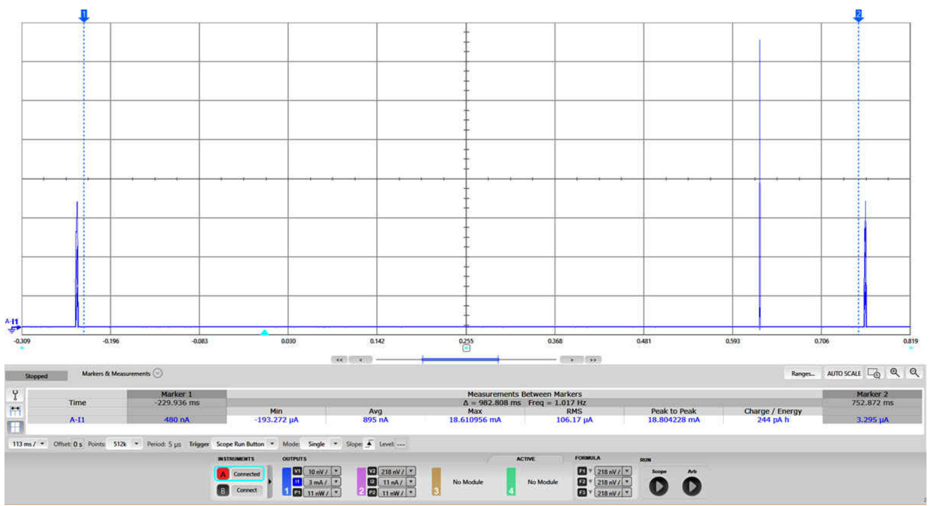


Figure 2-6. Measuring Standby Current During Connection (CC26x2)

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 estimated 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 3-1 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]).

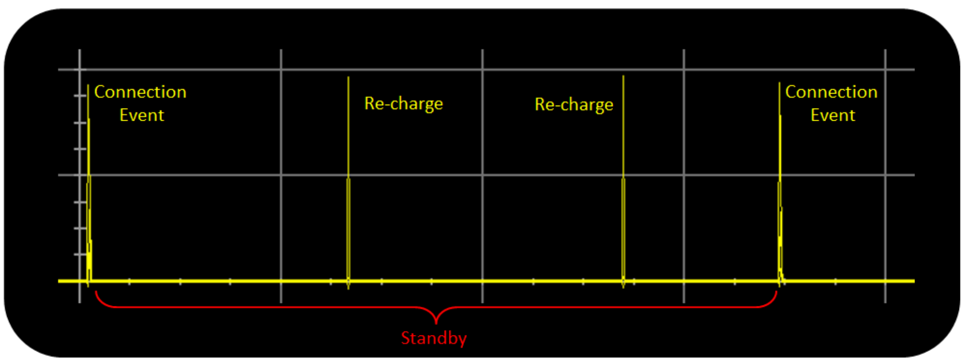


Figure 3-1. Current Consumption vs. Time During a Bluetooth Low Energy Connection

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 4-1. 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 4-1. CC2652R Supply Voltage

Supply Voltage	Internal DCDC	Minimum	Maximum
1.8 V System (External Regulator Mode)	No	1.7 V	1.95 V
3.3 V System	Optional	1.80 V	3.80 V

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].

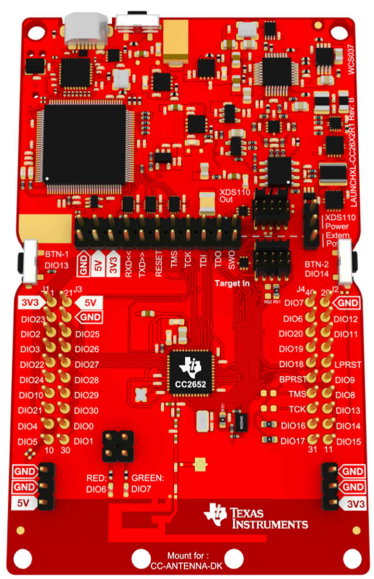


Figure 5-1. Device Under Test

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 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 5-2. 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]).

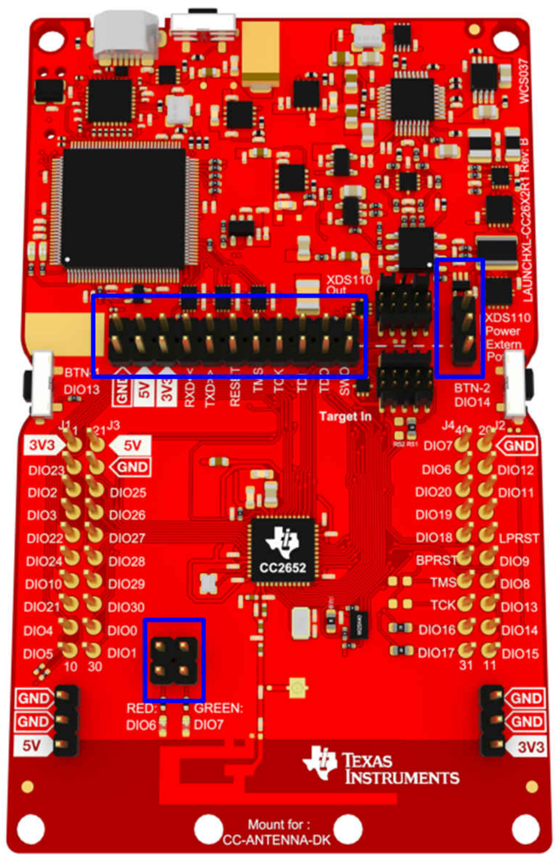


Figure 5-2. CC2652R LaunchPad Jumper Removal

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 5-3).

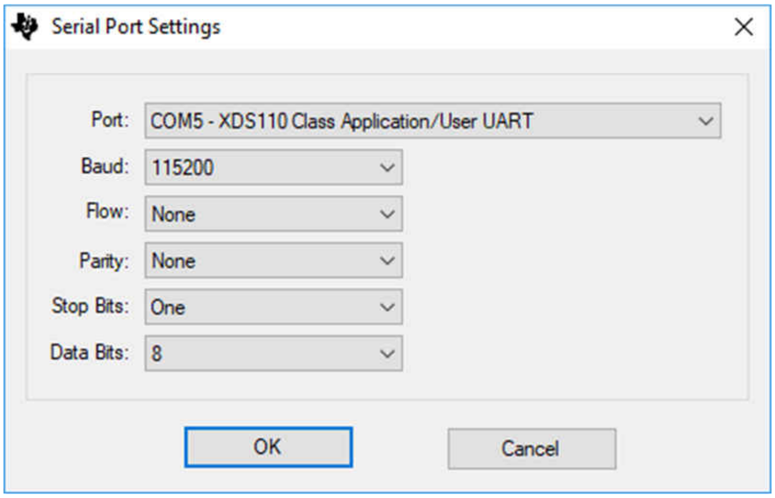


Figure 5-3. BTool Serial Port Settings

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 5-4. 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.

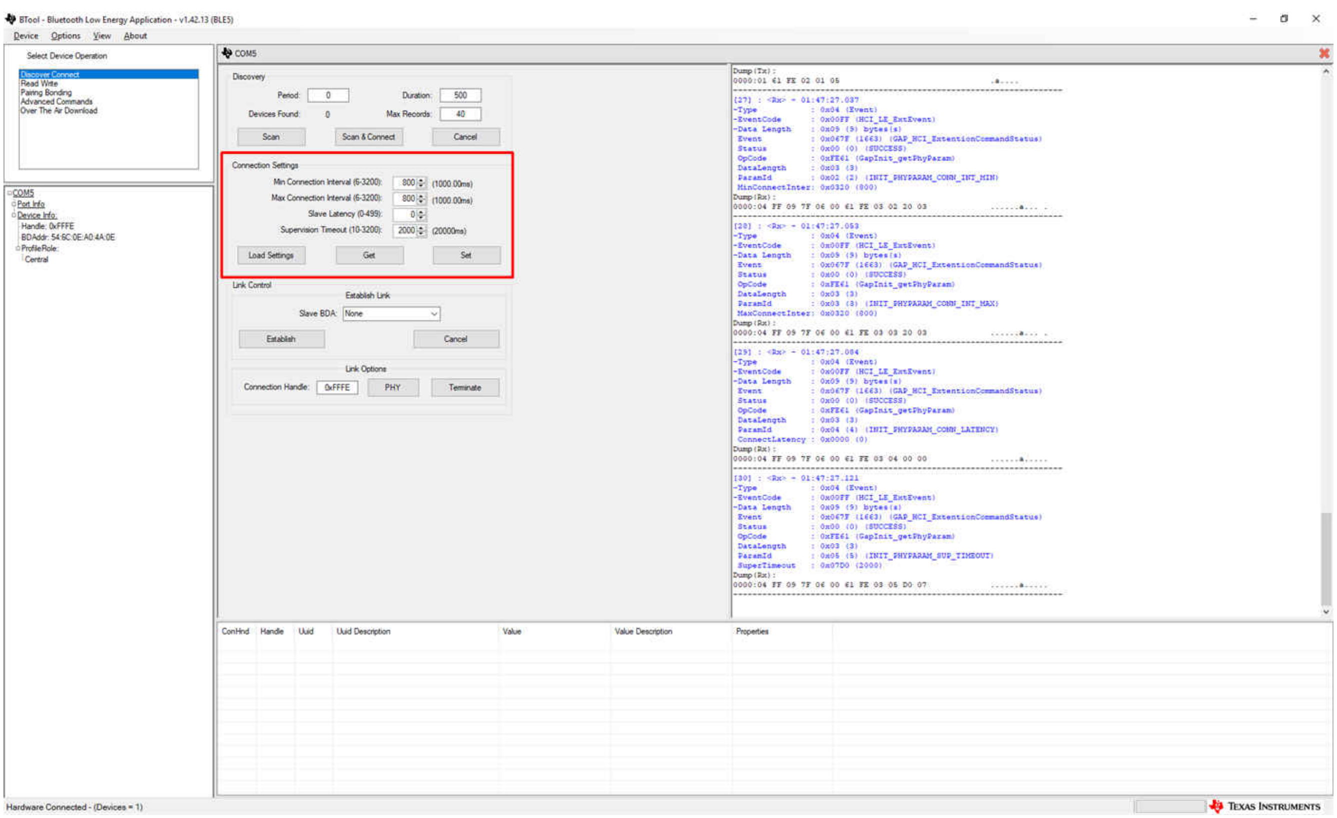


Figure 5-4. BTool Connection Settings

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 5-5.

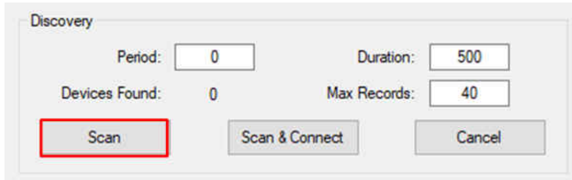


Figure 5-5. BTool Scan

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 5-6.

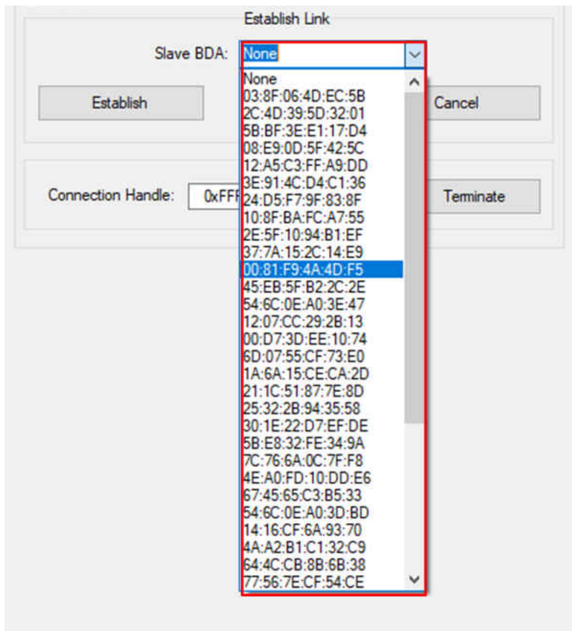


Figure 5-6. BTool Scan Results

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 5-7.

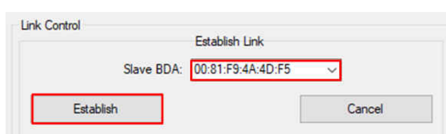


Figure 5-7. BTool Establish Link

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 5-8.

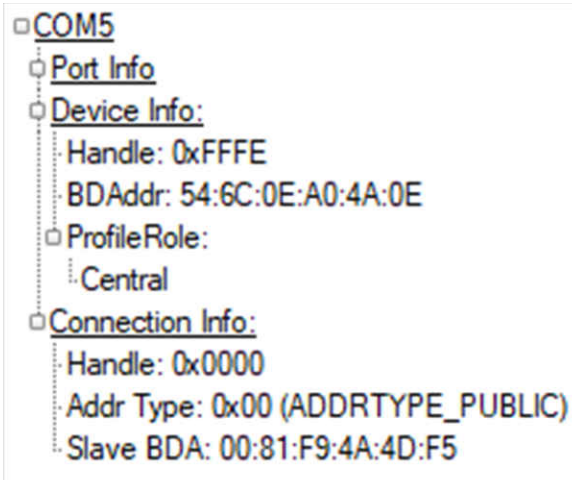


Figure 5-8. BTool Connected Device

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 6-1). The internal module is a N6781A, a 2-quadrant source and measure unit for battery drain analysis.

Figure 6-1. 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 6-2. 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.

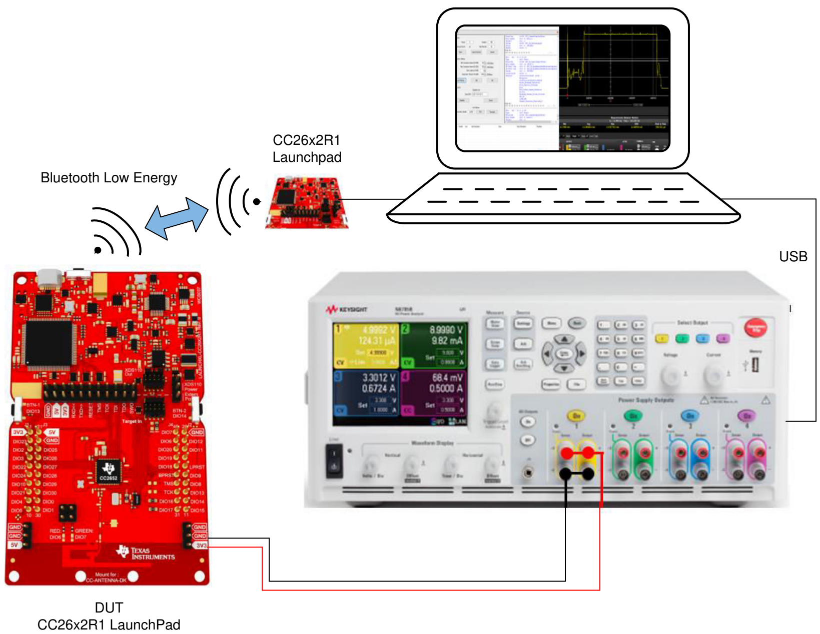


Figure 6-2. DUT Test Setup

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 6-3.

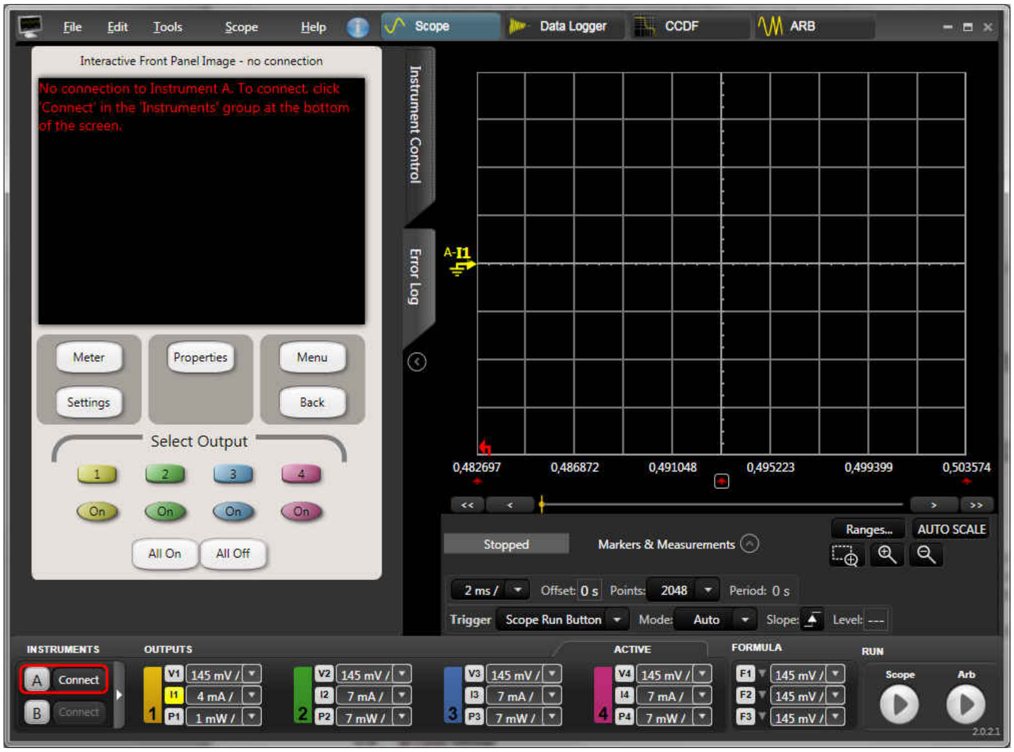


Figure 6-3. Agilent 14585A Control and Analysis Software, Start-Up

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 6-4.

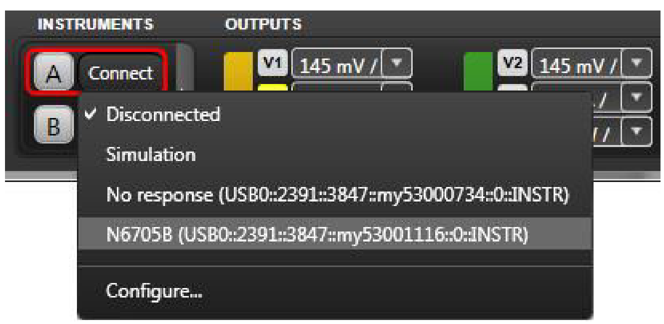


Figure 6-4. Agilent 14585A Control and Analysis Software, Connect

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 6-5.

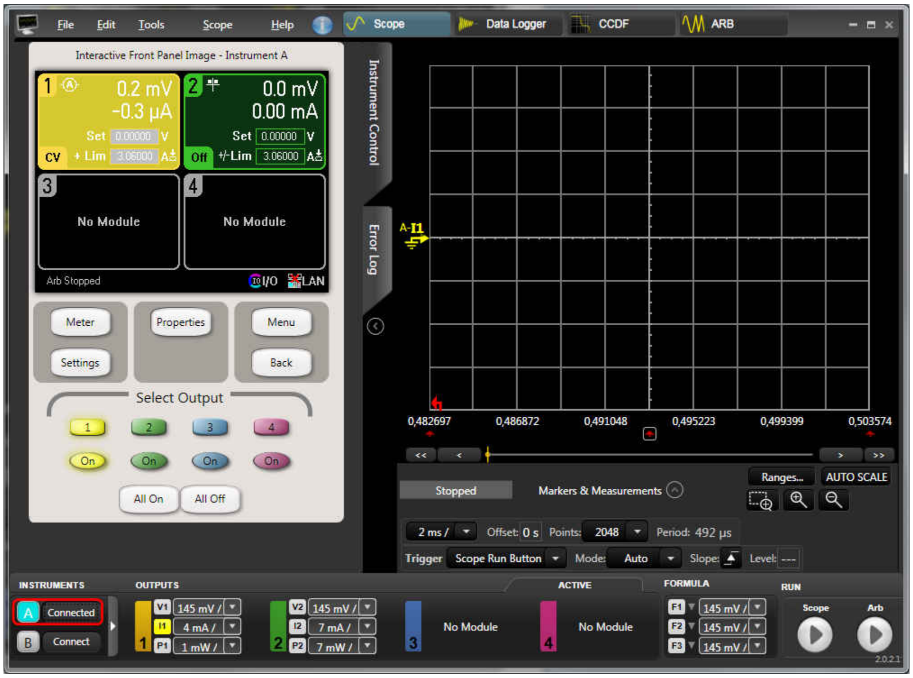


Figure 6-5. Agilent 14585A Control and Analysis Software, Connected

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.

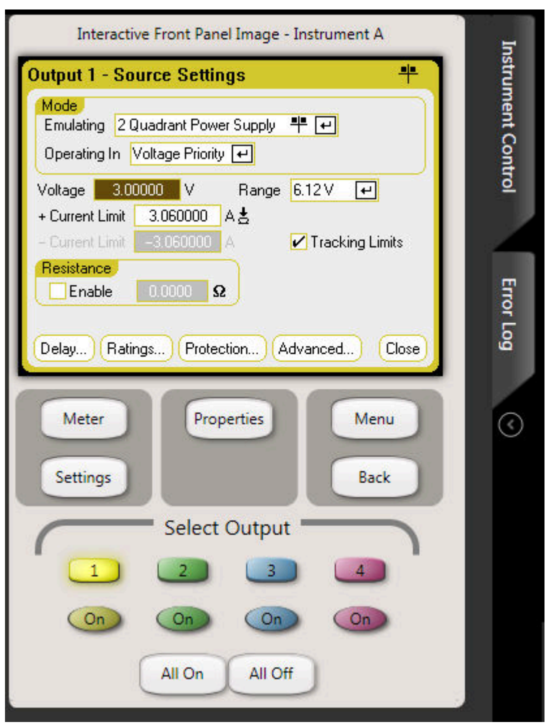


Figure 6-6. Agilent 14585A Control and Analysis Software, Source Settings

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 6-7.

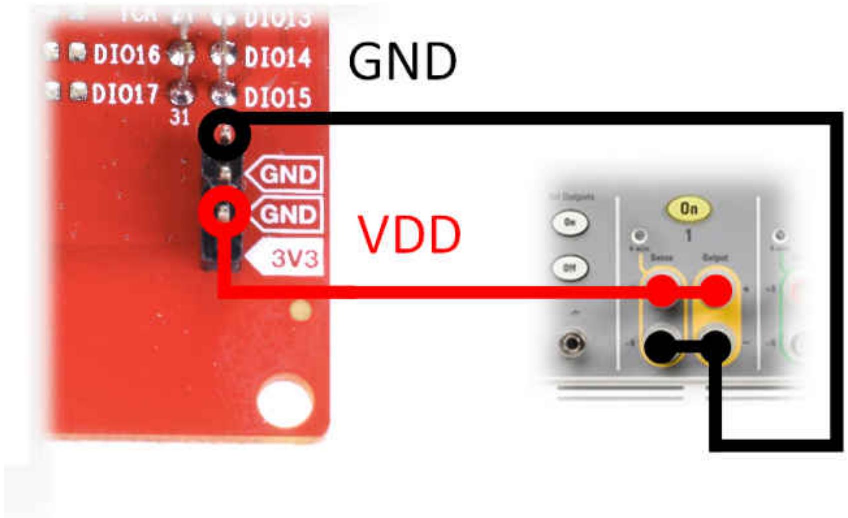


Figure 6-7. Connecting to the CC26x2r1 LaunchPad

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.

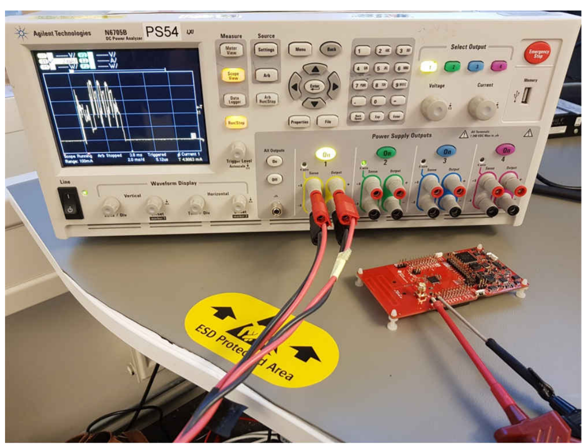


Figure 6-8. Connected DUT to Agilent 14585A

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 6-9.

Figure 6-9. Agilent 14585A Control and Analysis Software, Scope

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 6-10).

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

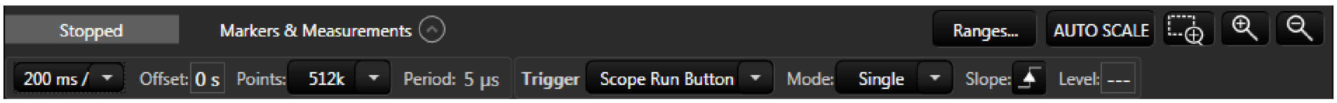


Figure 6-10. Agilent 14585A Control and Analysis Software, Scope Setup

Next, make sure that the "Ranges…" is setup to Auto, as shown in Figure 6-11.

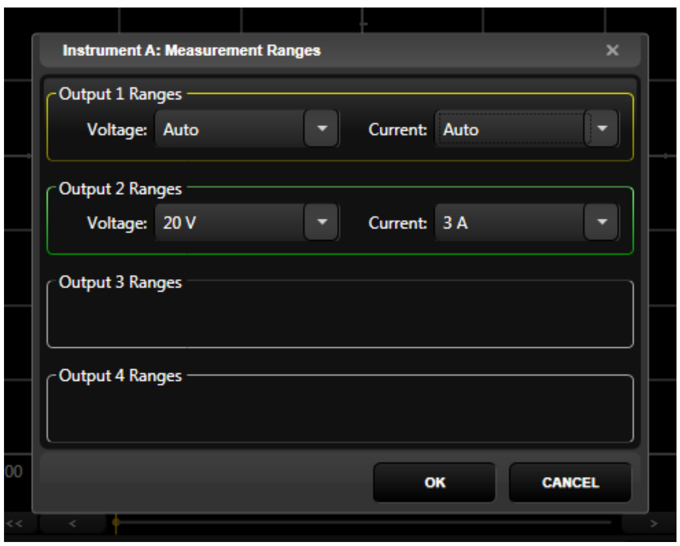


Figure 6-11. Agilent 14585A Control and Analysis Software, Instrument Range

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 6-12).

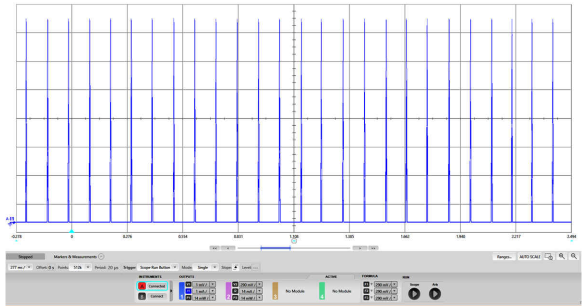


Figure 6-12. Agilent 14585A Control and Analysis Software, Advertisement Capture (CC26x2)

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 6-13 and summarized in Table 6-1.

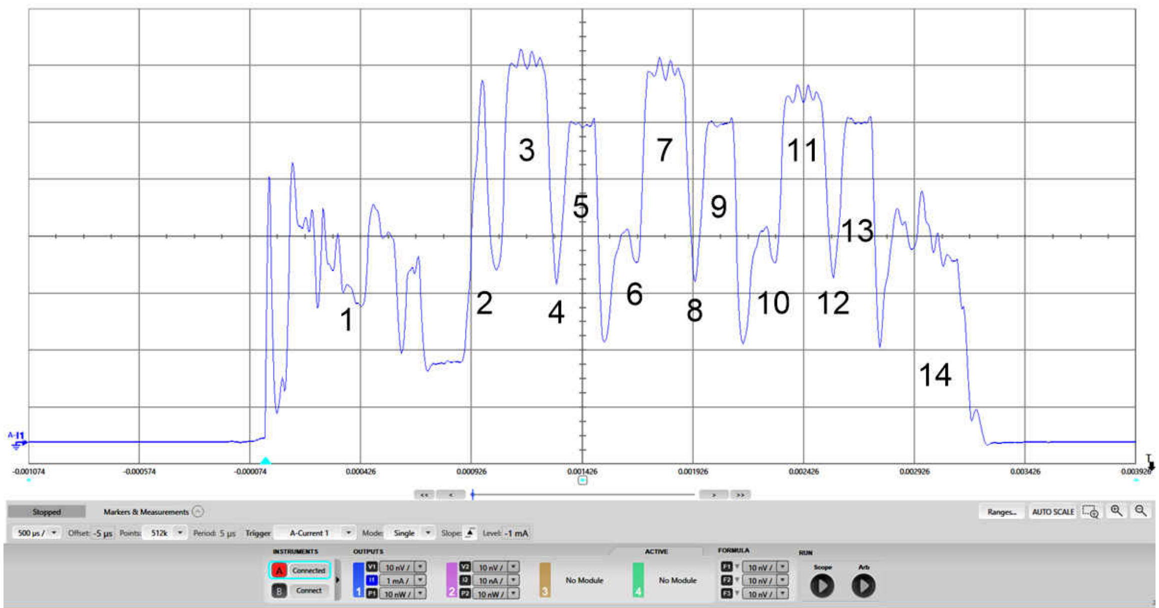


Figure 6-13. Connectable Advertising Event, Capture

Table 6-1. Advertising Event, State Analysis

Number	State	Comments
1	Pre-processing	RTOS wake-up, radio setup, XTAL guard time
2	Radio preparation	Radio is turned on and in transition to TX
3	TX	The radio transmits an advertisement packets with 3 bytes data on Channel 37. Time is dependent on amount of transmitted data
4	TX to RX transition	TX to RX transition
5	RX	Time depends on advertising interval and Sleep Crystal Accuracy (SCA)
6	RX to TX transition	RX to TX transition
7	TX	The radio transmits an advertisement packets with 3 bytes data on Channel 38. Time is dependent on amount of transmitted data
8	TX to RX transition	TX to RX transition
9	RX	Time depends on advertising interval and SCA
10	RX to TX transition	RX to TX transition
11	TX	The radio transmits an advertisement packets with 3 bytes data on Channel 39. Time is dependent on amount of transmitted data
12	TX to RX transition	TX to RX transition
13	RX	Time depends on advertising interval and SCA
14	Post-processing and going to Standby	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

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.

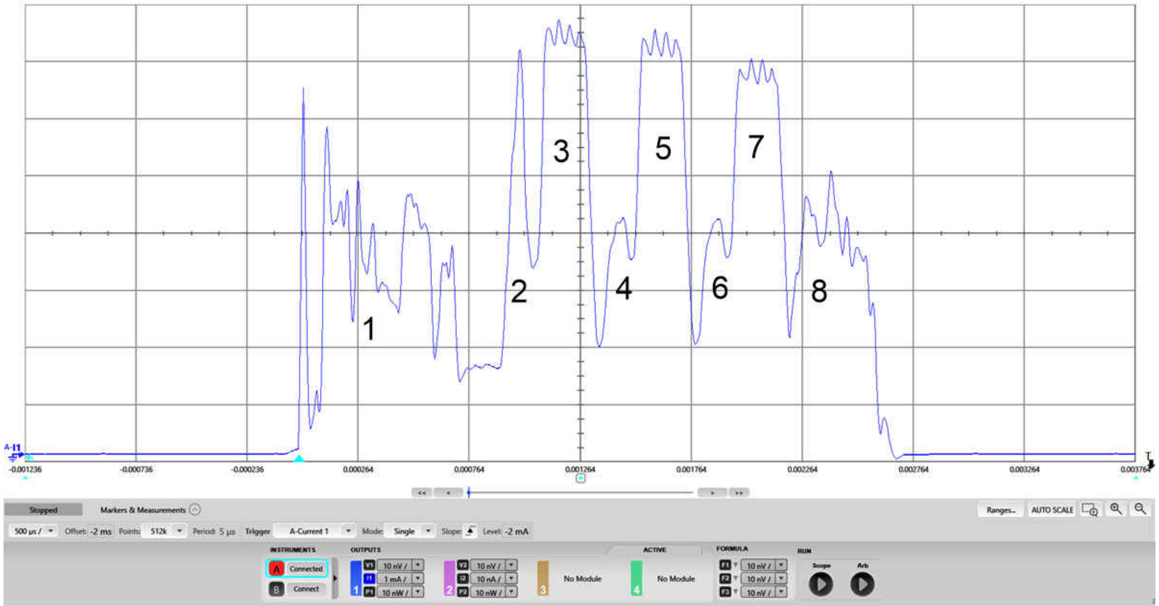


Figure 6-14. Beacon Event, Capture

Table 6-2. Beacon Event, State Analysis

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

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 2-6), you should identify two Connection events. Place Marker 1 right after a Connection event, as shown in Figure 6-15, and the second marker after the following Connection event. The average current is in this case approximately 10.2 µA, as shown in Figure 6-16.

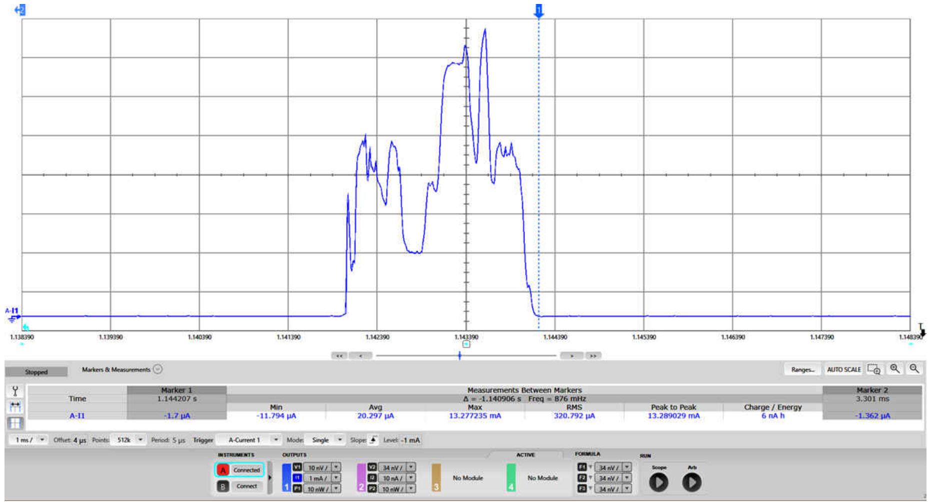


Figure 6-15. Connection Event, Marker #1 Placement

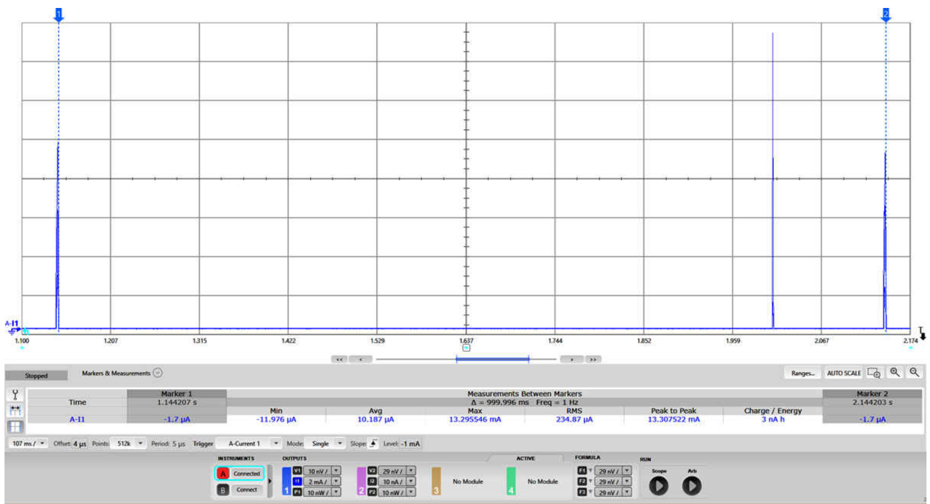


Figure 6-16. Average Current Consumption After Establishing a Connection

The Connection event can also be analyzed (like what was done with the Connectable Advertising event in Figure 6-13) by selecting "Markers & Measurements". This is shown in Figure 6-17 and summarized in Table 6-3.

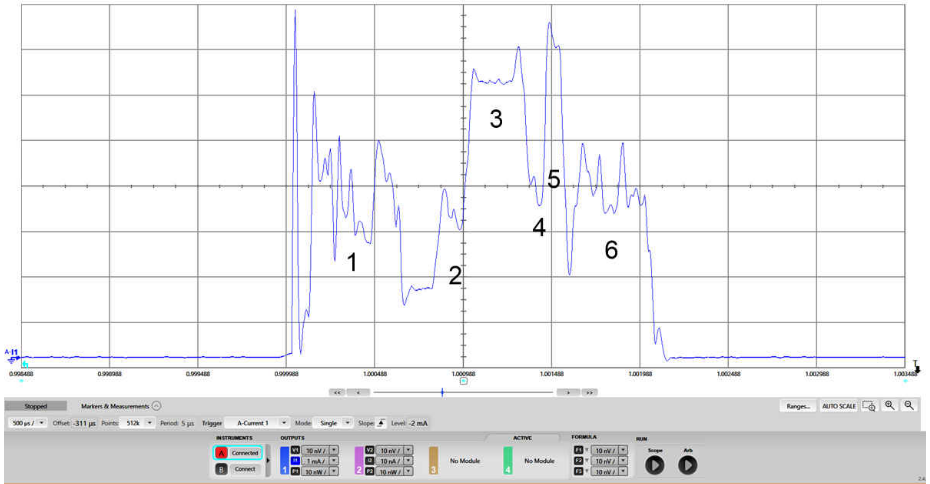


Figure 6-17. Current Consumption versus Time During a Single Connection Event

Table 6-3. Connection Event, State Analysis

Number	State	Comment
1	Pre-processing	RTOS wake-up, radio setup, XTAL guard time
2	Radio preparation	Radio is turned on and in transition to RX
3	Recieve (RX)	The radio receiver listens for a packet from the master. Time depends on connection interval and SCA.
4	RX to TX transition	RX to TX transition
5	Transmit (TX)	The radio transmits a packet to the master on one of the 37 channels. Time is dependent on the amount of transmitted data
6	Post-processing and going to Standby	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

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.

The calculations integrated into the spreadsheet are fairly simple. The average current draw during an event is calculated by the formula shown in Equation 1.

$$I_{Avg.E} = \frac{\Sigma_0^n \left(\left(t_{State\ n} \right)^* \left(I_{State\ n} \right) \right)}{t_{Total}} [A] \tag{1}$$

For the connection event measured in Section 6.3.2, the average current during event is calculated with the formula shown in Equation 2.

$$I_{Avg.E} = \frac{83\,\mu s * 3.13\,mA + \ldots + 284\,\mu s * 0.69\,mA}{2425\,\mu s} = 3.619\,mA \tag{2}$$

The next step is to calculate the average current for the entire connection interval, which takes into account the time during which the device is sleeping. For this, the formula in Equation 3 is used.

$$I_{Avg.} = \frac{\left(t_{Conn.Int} - t_{Awake} \right)^* I_{Standby} + \left(t_{Awake} * I_{Avg.E} \right)}{t_{Conn.Int}} [A] \tag{3}$$

For the connection event measured in Section 6.3.2, the average current while connected is calculated with the formula shown in Equation 4.

$$I_{Avg.} = \frac{(1s - 2.425ms) * 0.965uA + (2.425ms * 3.619mA)}{1s} = 9.74\,\mu A \tag{4}$$

The average current while connected is used to calculate the amount of time to expect that the battery last while running continuously in a connection. The total hours of battery life can be calculated using the simple formula shown in Equation 5.

$$Battery\ Life = \frac{Battery\ Capacity}{I_{Avg.}} \tag{5}$$

If you assume that the battery capacity is 230 mAh (a common capacity value for a CR2032 coin cell battery) and use the average current calculated from before, it is now possible to calculate the expected battery life with the formula shown in Equation 6.

$$Battery\ Life = \frac{230\,mAh}{9.74\,\mu A} \approx 23613\,hrs \approx 984\,days \tag{6}$$

The battery can be expected to last for 23613 hours, or approximately 984 days, while running continuously in a connected state with a 1 second connection interval.

This is presented in the spreadsheet along with a similar calculation for when a device is in advertising mode. By modifying the interval settings, good estimations can be achieved for certain use cases.

Bluetooth Smart Power consumption - *Beta*

Hardware	SmartRF06EB + CC2650EM	
Software	SimpleBLEPeripheral (pre-RTM)	
Supply Voltage	3V	
Battery capacity [mAh]	230	1

Activity 1: Advertising

Standby Current [uA]	1,49	
Advertising Interval [ms]	100	2

State	Time [µs]	Current [mA]	
Re-Charge	83	1,6	132,8
Wake Up & Pre-Processing	898	3,39	3044,22
Radio Preparation	220	3,77	829,4
Transmit (TX)	202	7,19	1452,38
TX to RX transition	111	4,43	491,73
Recieve (RX)	184	6,34	1166,56
RX to TX transition	370	3,40	1258
Transmit (TX)	202	7,09	1432,18
TX to RX transition	111	4,54	503,94
Recieve (RX)	184	6,39	1175,76
RX to TX transition	370	3,36	1243,2
Transmit (TX)	202	6,87	1387,74
TX to RX transition	111	4,45	493,95
Recieve (RX)	184	6,42	1181,28
Post-Processing	409	3,5	1431,5
Going to Standby	230	0,85	195,5
			17420,14
Total time of connection event	4071		
Average Current draw during advertising event [mA]		4,27908	
Average current draw accounting for sleep [mA]			0,175630742

Avg. current draw during advertisement:	**0,176**	mA
Expected battery life:	**1310**	Hours
Expected battery life:	**55**	Days

Activity 2: Connected as slave

Standby Current [uA]	0,965	
Connection Interval [ms]	1000	3

State	Time [µs]	Current [mA]	
Re-charge	83	3,13	259,79
Wake Up & Pre-processing	931	3,31	3081,61
Preparation for Recieve	261	3,56	929,16
Recieve (RX)	236	6,43	1517,48
RX to TX transition	141	5,27	743,07
Transmit (TX)	95	7,1	674,5
Post-Processing (RF)	394	3,49	1375,06
Device goes to Standby	284	0,696	197,664
			8778,334
Total time of connection event	2425		
Average Current draw during connection event [mA]		3,619931546	
Average current draw accounting for sleep [mA]			0,009741

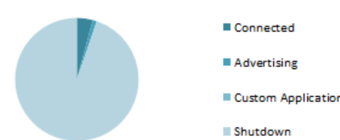
Average current draw during connection:	**0,010**	mA
Expected battery life:	**23612**	Hours
Expected battery life:	**984**	Days

Combined activities

use case calculator - per day

Mode	Hours	Minutes	Avg. Current	Unit	%	
Connected	1	0	0,176	mA	4 %	
Advertising	0	15	0,010	mA	1 %	4
Custom Application	0	0	0	mA	0 %	
Shutdown	23	22,75	0,0001	mA	95 %	
Average current			0,00751	mA		
			30 609	Hours		
Battery Life			**1 275**	Days		5
			3,49	Years		

Vizualisation of the modes:



7 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 LaunchPad [20].

The example can be found and downloaded using Resource Explorer (see Figure 7-1).

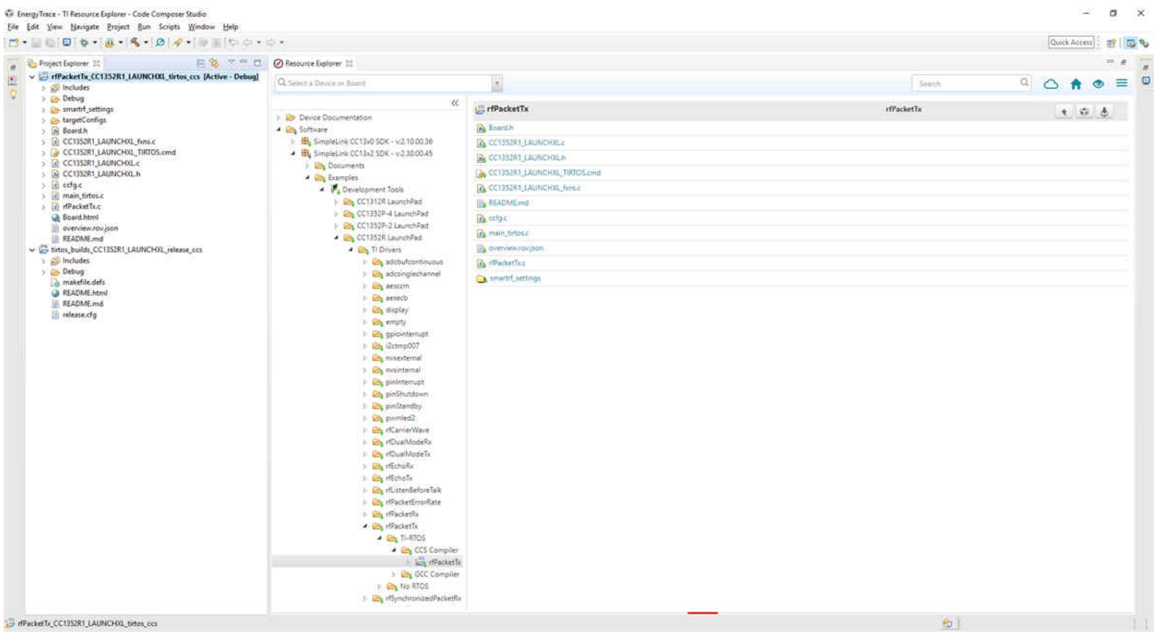


Figure 7-1. rfPacketTx in Resource Explorer

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 7-2). 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).

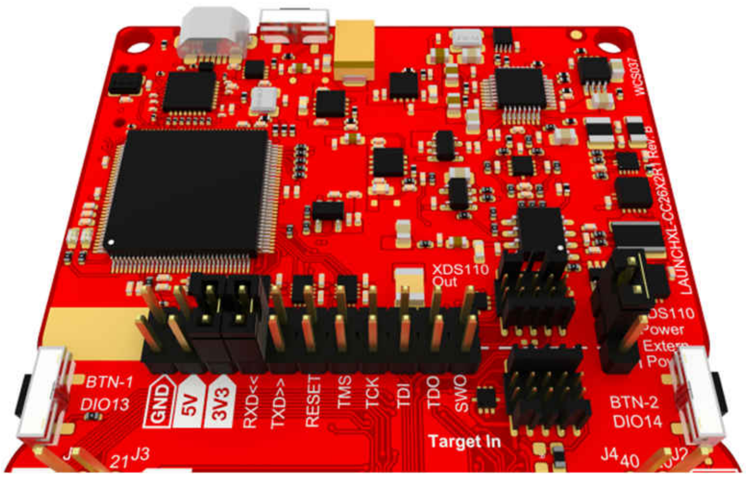


Figure 7-2. Jumper Settings

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 7-3).

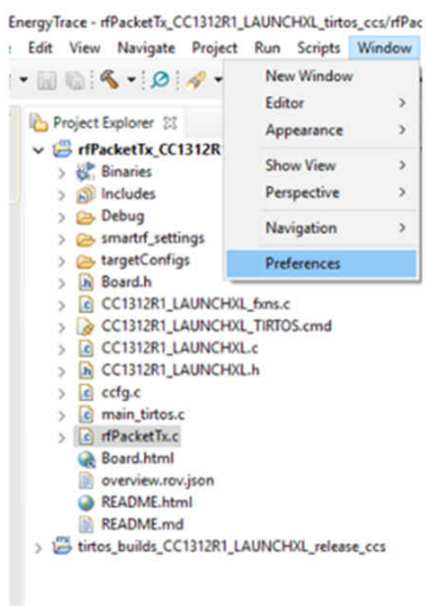


Figure 7-3. Preferences

4. Navigate to the `EnergyTrace Technology` window and configure it as shown in Figure 7-4.

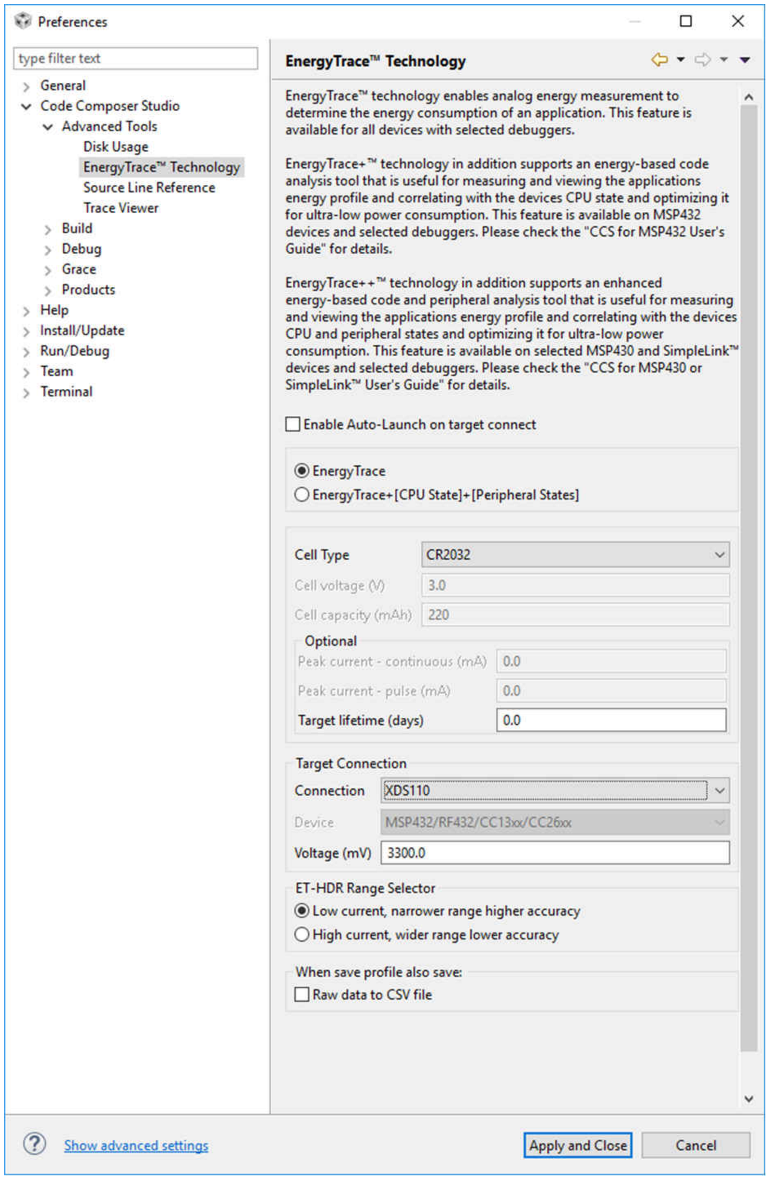


Figure 7-4. EnergyTrace Technology Configuration

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 7-5).

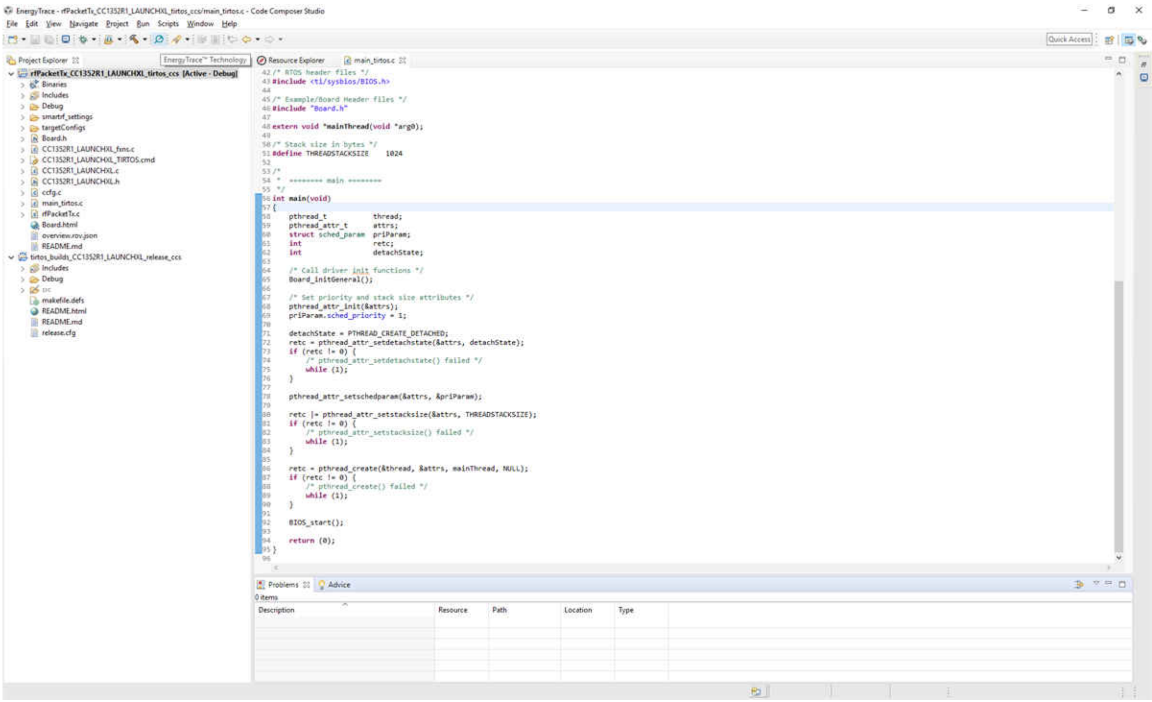


Figure 7-5. EnergyTrace Button

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 7-6).

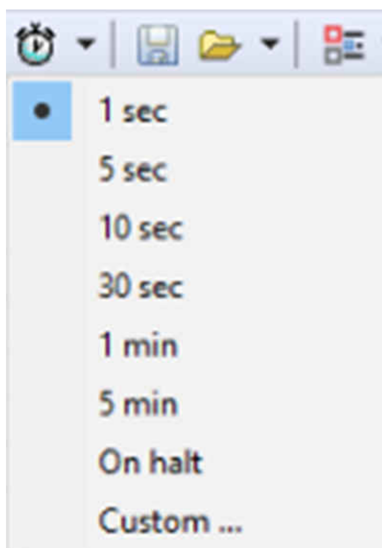


Figure 7-6. Set Measurement Duration

7. Click the green play button to start capturing data (see Figure 7-7). The red LED on the XDS110 debugger should be turned on, and will be so for the duration of the EnergyTrace capture.

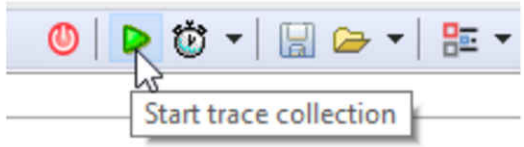


Figure 7-7. Start Trace Collection

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

.

Figure 7-8 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.

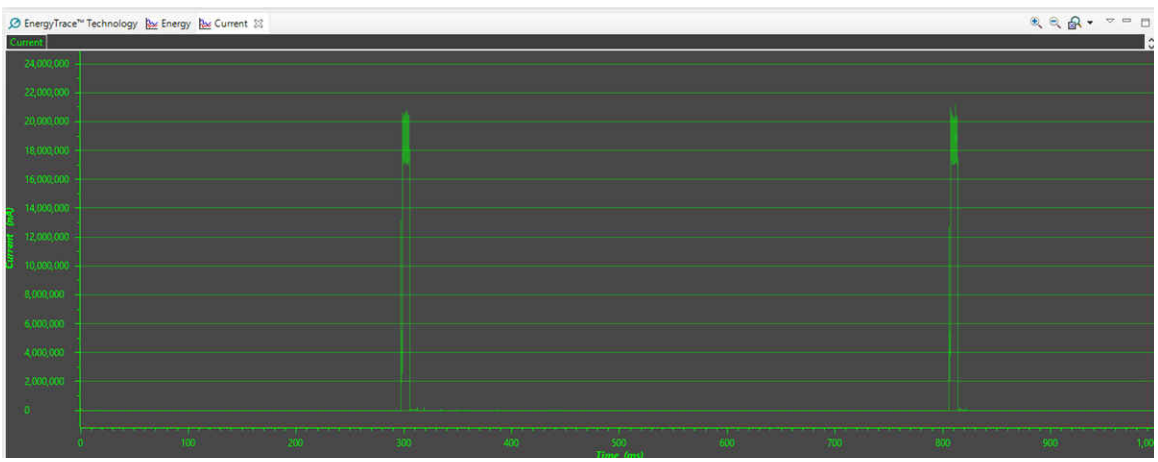


Figure 7-8. Current Profile for rfPacketTx (without modifications)

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 7-9).

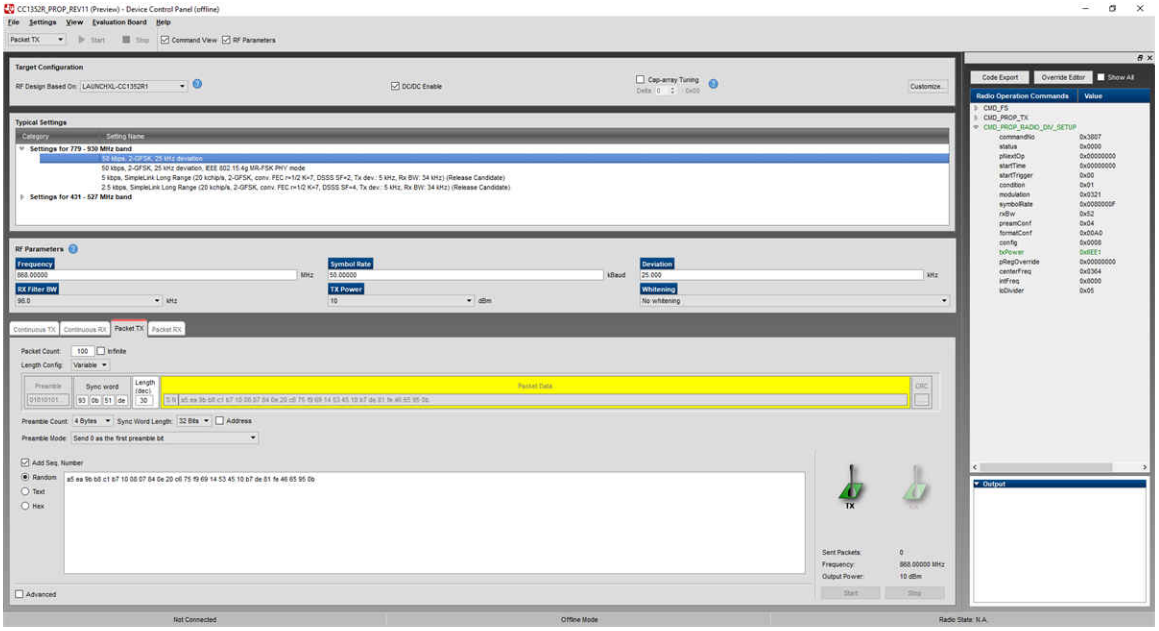


Figure 7-9. Using SmartRF Studio to Find New Settings

After changing the output power by modifying the smartrf_setting.c file, as shown in Figure 7-10, the measurements were repeated.

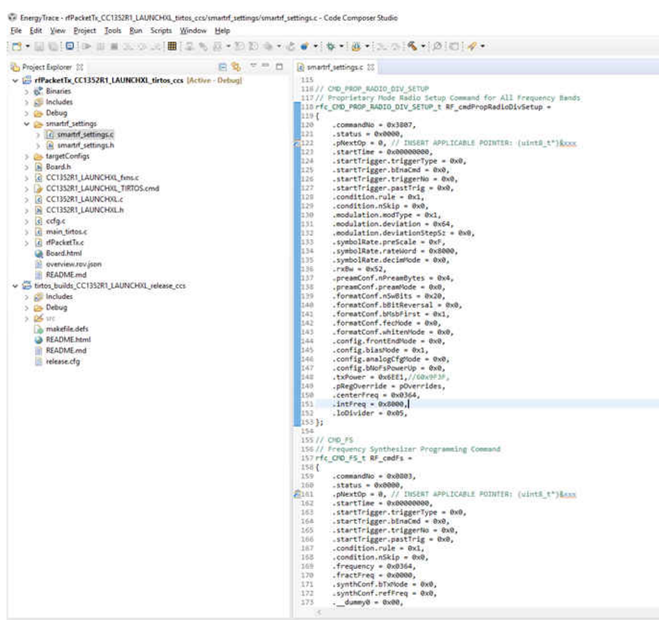


Figure 7-10. Modifying the smartrf_settings.c File

Figure 7-11 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 7-12 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.

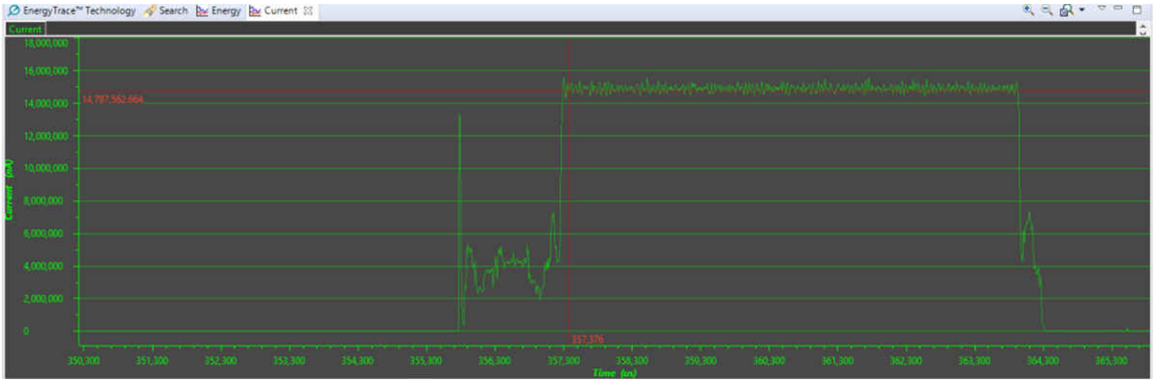


Figure 7-11. Current Profile of TX (EnergyTrace)

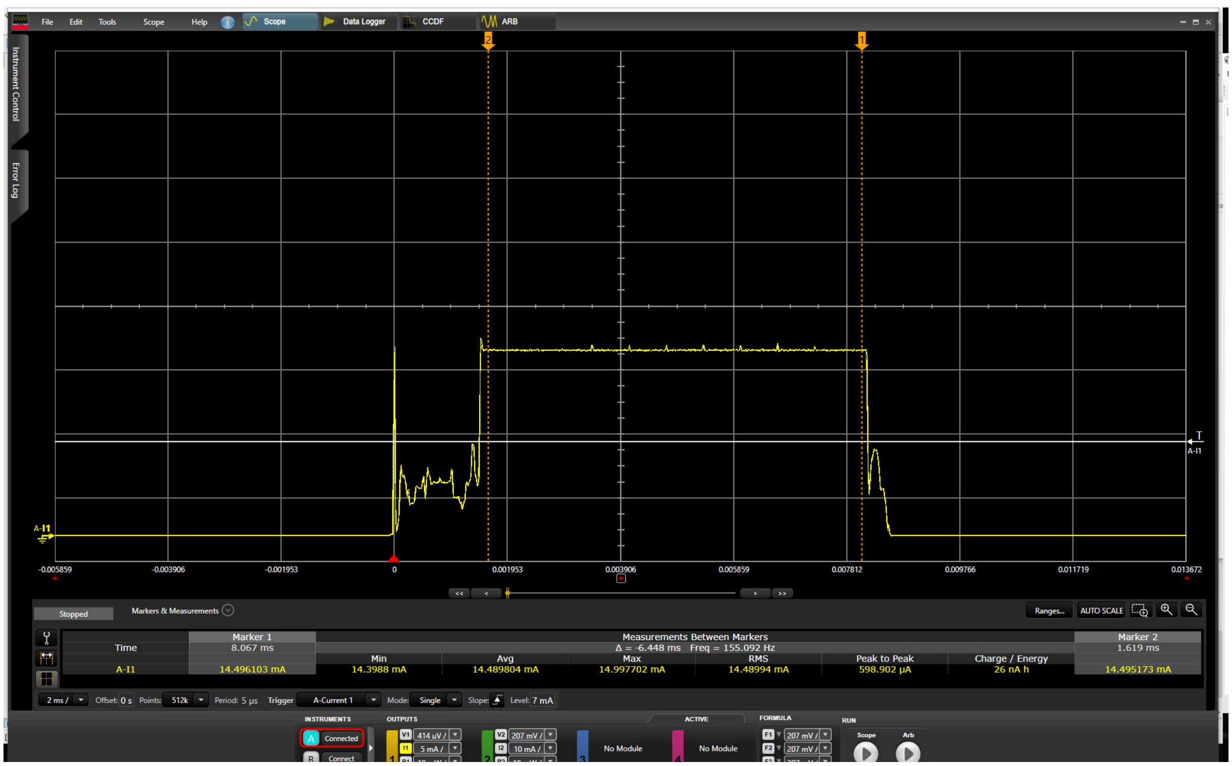


Figure 7-12. Current Profile of TX (DC Power Analyzer)

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 7-13).

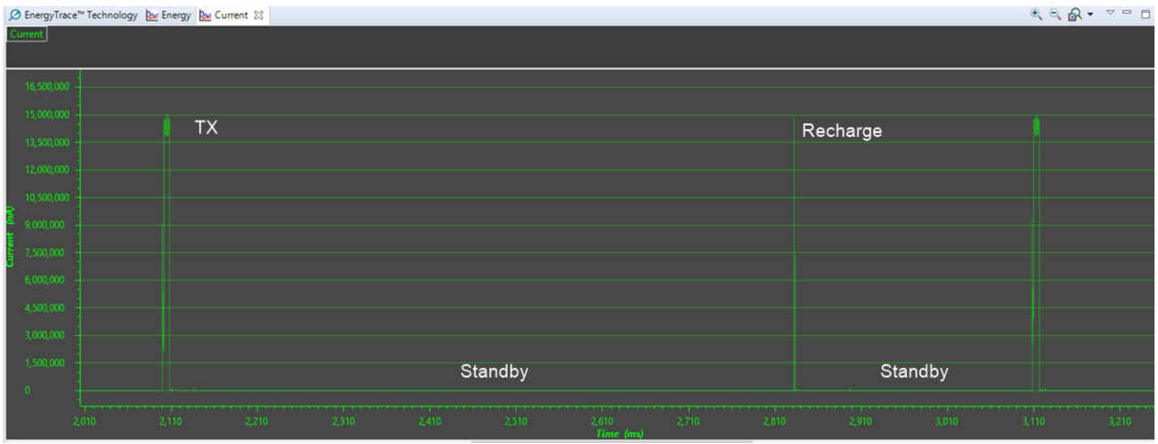


Figure 7-13. Recharge Pulses

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 7-14).

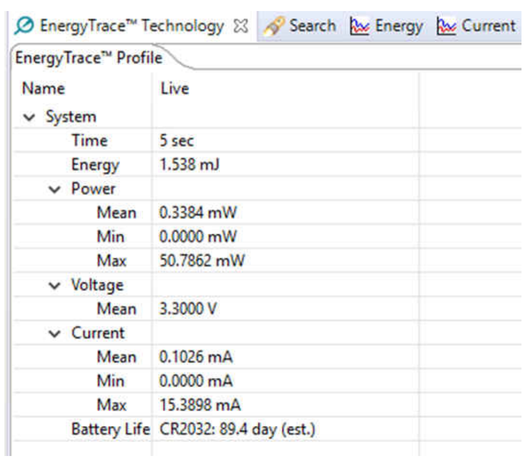


Figure 7-14. Average Current Consumption When Device Enters Standby

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:

```
#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 7-15 shows what the current profile looks like, with an average current consumption of 1 mA (see Figure 7-16).

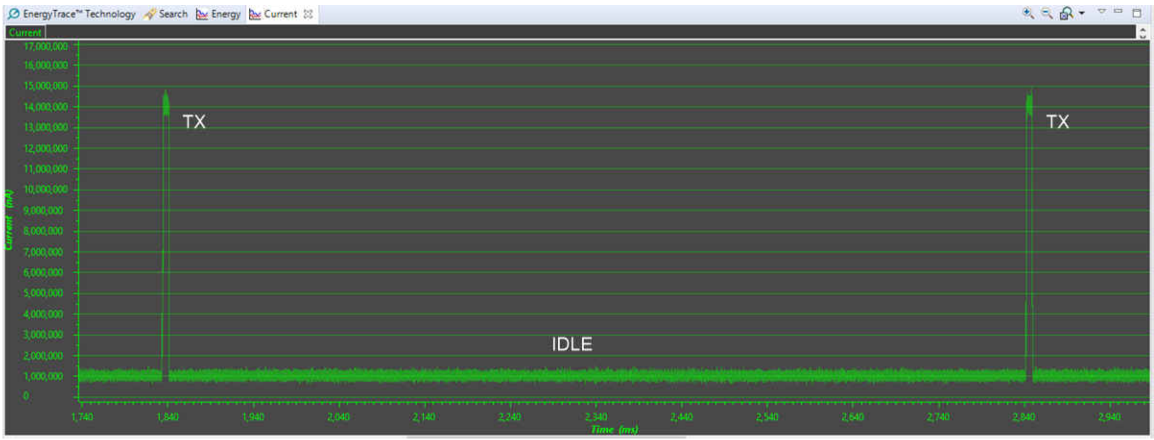
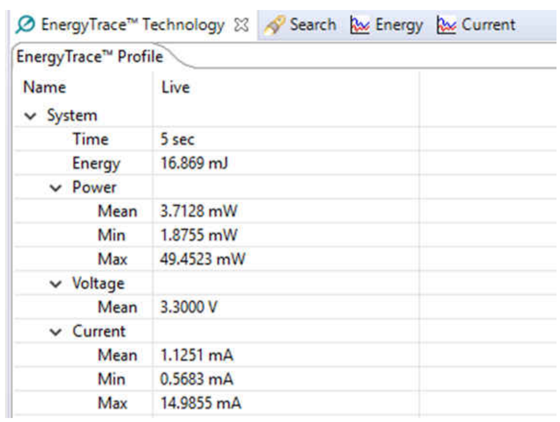


Figure 7-15. IDLE State Between Packets



EnergyTrace™ Technology | Search | Energy | Current

EnergyTrace™ Profile

Name	Live
System	
Time	5 sec
Energy	16.869 mJ
Power	
Mean	3.7128 mW
Min	1.8755 mW
Max	49.4523 mW
Voltage	
Mean	3.3000 V
Current	
Mean	1.1251 mA
Min	0.5683 mA
Max	14.9855 mA

Figure 7-16. Average Current Consumption When Device Does not Enter Standby

8 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*
3. Texas Instruments: *CC13x2, CC26x2 SimpleLink™ Wireless MCU Technical Reference Manual*
4. Texas Instruments: *CC1310 SimpleLink™ Ultra-Low-Power Sub-1 GHz Wireless MCU Data Manual*
5. Texas Instruments: *CC1312R SimpleLink™ High-Performance Sub-1 GHz Wireless MCU Data Manual*
6. Texas Instruments: *CC1350 SimpleLink™ Ultra-Low-Power Dual-Band Wireless MCU Data Manual*
7. Texas Instruments: *CC1352R SimpleLink™ High-Performance Dual-Band Wireless MCU Data Manual*
8. Texas Instruments: *CC1352P SimpleLink™ High-Performance Dual-Band Wireless MCU With Integrated Power Amplifier Data Manual*
9. Texas Instruments: *CC2640 SimpleLink™ Bluetooth® 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/
15. BLE-Stack™: www.ti.com/ble-stack
16. IAR Embedded Workbench™ for Arm: http://www.iar.com/Products/IAR-Embedded-Workbench/ARM/
17. SimpleLink™ CC2650 wireless MCU LaunchPad™ Development Kit (http://www.ti.com/tool/launchXL-cc2650)
18. SimpleLink™ Bluetooth® low energy CC2640R2F wireless MCU LaunchPad™ development kit (http://www.ti.com/tool/launchxl-cc2640r2)
19. SimpleLink™ Multi-Standard CC26x2R Wireless MCU LaunchPad™ Development Kit (http://www.ti.com/tool/launchxl-cc26x2r1)
20. SimpleLink™ Multi-Band CC1352R Wireless MCU LaunchPad™ Development Kit (http://www.ti.com/tool/launchxl-cc1352r1)
21. SimpleLink™ Multi-Band CC1352P Wireless MCU LaunchPad™ Development Kit (http://www.ti.com/tool/launchxl-cc1352p)
22. CCS Integrated Development Environment: (http://www.ti.com/tool/ccstudio)
23. BLE5-Stack User's Guide: (http://dev.ti.com/tirex/#/?link=Software%2FSimpleLink%20CC26X2%20SDK%2FDocuments%2FBLE5-Stack%2FBLE5-Stack%20User's%20Guide)
24. Bluetooth Power Calculator Tool: (http://www.ti.com/tool/bt-power-calc)
25. SmartRF™ Studio: (http://www.ti.com/tool/smartrftm-studio)
26. CC13x2 Proprietary RF User’s Guide: (http://dev.ti.com/tirex/#/?link=Software%2FSimpleLink%20CC13x2%20SDK%2FDocuments%2FProprietary%20RF%2FProprietary%20RF%20User's%20Guide)

9 Revision History

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