Measuring *Bluetooth*® Low Energy Power Consumption

By Sandeep Kamath

Keywords

- *Bluetooth Low Energy*
- *BLE*
- *Power Consumption*
- *Battery Life*
- *CC2540*
- *CC2540DK-MINI*

1 Introduction

The *Bluetooth*® low energy (BLE) standard was developed with long battery life in mind, allowing for devices that can last anywhere from several months to several years while operating on a single coin-cell battery. This application note describes the setup and procedures to measure power consumption on the CC2540 device operating as a GAP "Peripheral" in a BLE connection.

It is assumed the reader of this application note has knowledge about the BLE standard and the CC2540. The example is based on the Texas Instruments CC2540 running version 1.0 of the BLE stack. Please refer to [1] and the [2] for further information.

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.

The current consumption measurements are presented, and battery life time is calculated for an example application. An accompanying Excel sheet is provided so that users can estimate their battery life based on their own custom usage scenario.

Note that the results presented in this document are intended as a guideline only. A variety of factors will influence the battery life calculation and final measurements. Measurements should be performed on customer hardware, in a controlled environment, and under the target application scenario.

Table of Contents

2 Abbreviations

BLE	*Bluetooth* low energy
DC	Direct current
DK	Development kit
DMM	Digital Multimeter
GAP	Generic Access Profile
GPIO	General-Purpose Input / Output
HAL	Hardware Abstraction Layer
I/O	Input / Output
MCU	Microcontroller unit
OSAL	Operating system abstraction layer
PC	Personal computer
PDU	Packet data unit
PM2	Power mode 2
PM3	Power mode 3
RF	Radio frequency
Rx	Receive
Tx	Transmit

3 Understanding Power Metrics in *Bluetooth* Low Energy

It is not possible to compare the power consumption of a BLE device to another using a single metric. For example, sometimes a device gets rated by its “peak current”. While the peak current plays a part in the total power consumption, a device running the BLE stack will only be consuming current at the peak level while it is transmitting. Please refer to [3] for further information on understanding the peak currents effect on batteries. Even in very high throughput systems, a BLE device is transmitting only for a small percentage of the total time that the device is connected.

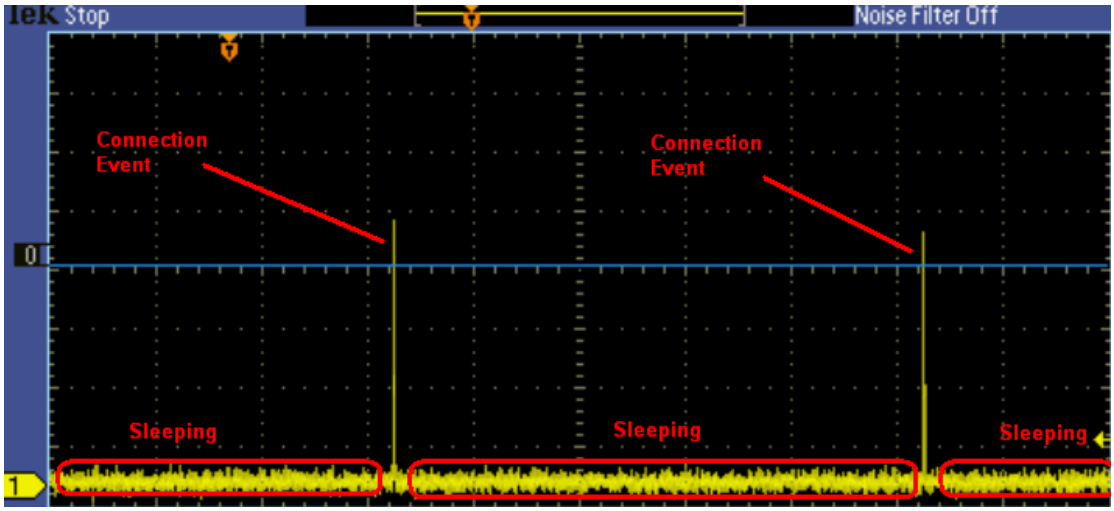

Figure 1- Current Consumption versus Time during a BLE Connection

In addition to transmitting, a BLE device will most likely go through several other states, such as receiving, sleeping, waking-up from sleep, etc… Even if a device’s current consumption in each different state is known, this is still not enough information to determine the total power consumed by the device. The different layers of the BLE stack all require certain amounts of processing in order to remain connected and comply with the protocol’s specifications. The MCU takes time to perform this processing, and during this time current is consumed by the device. In addition, the device might take some time when switching between states. All of this must be taken into account in order to get an accurate measurement of the total current consumed.

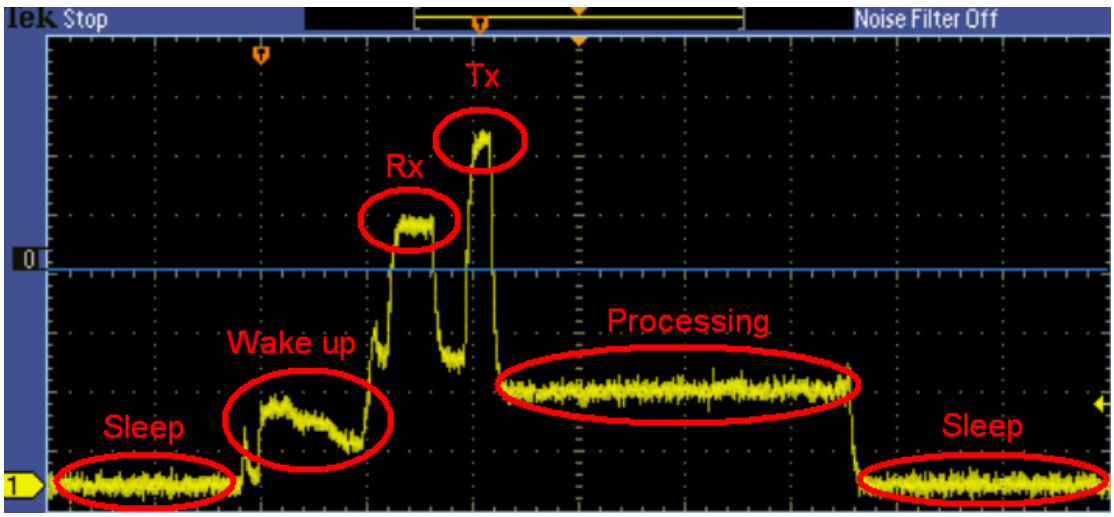

Figure 2- Current Consumption versus Time during a single Connection Event

In a typical application, a device running the BLE stack will spend most of the time in a sleeping state between connection events. When the CC2540 goes into “Power mode 2” (PM2) between connection events, the internal digital voltage regulator is turned off, along with the 16 MHz RCOSC and 32 MHz crystal oscillator. The 32 kHz sleep timer remains active while the RAM and registers are retained. The only way that the device will wake up is if an I/O interrupt or sleep timer interrupt occurs.

The primary metric that takes all of these other time and current measurements into account is the “average current”. It is this value that can be used to determine the battery life of a device running the BLE stack. Note that a single “average current” value cannot be given for a device in its datasheet or in the device’s specifications, as the average current is highly dependent on the connection parameters used. Anytime an “average current” specification is given, it is very important to understand the exact use-case during which the measurement was made.

For a complete system-on-a-chip such as the CC2540, it is important to understand that the MCU is typically not only running the BLE protocol stack, but it is also running profiles and an application. The application not only uses the MCU on the device, but it may also be using peripherals on the chip, such as an ADC or op-amp. In addition, other devices on the circuit board, aside from the device running the BLE protocol stack, may be drawing current as well. This document will focus on strictly measuring current consumed as a result of the BLE protocol stack; however it is important to be aware of other sources of current consumption, as they will affect the battery life.

4 Test Setup

This section describes the general setup required for performing power testing.

4.1 System Overview

In order to properly measure average current consumption, the current must be measured with respect to time. Therefore, a basic multimeter is not sufficient, and an oscilloscope is required. The simplest way to measure current with an oscilloscope is to use a current probe and directly monitor the current going into the system. If you do not have a current probe available, an easy alternative is to use a small resistor in line with the power supply input to the system. You can then use a standard oscilloscope voltage probe to measure the voltage across the resistor, and effectively measure the current by dividing the voltage by the resistance. A good resistor value to use is 10 Ω, as this value is small enough that it shouldn’t affect the existing circuitry, and large enough to provide a voltage that can be measured with decent precision. In addition, using a value of 10 Ω makes the calculations very easy.

When performing measurements, it is best to use a regulated DC power supply as opposed to an actual battery. This eliminates variables that might be caused by a defective or low battery. Figure 3 below shows the full setup.

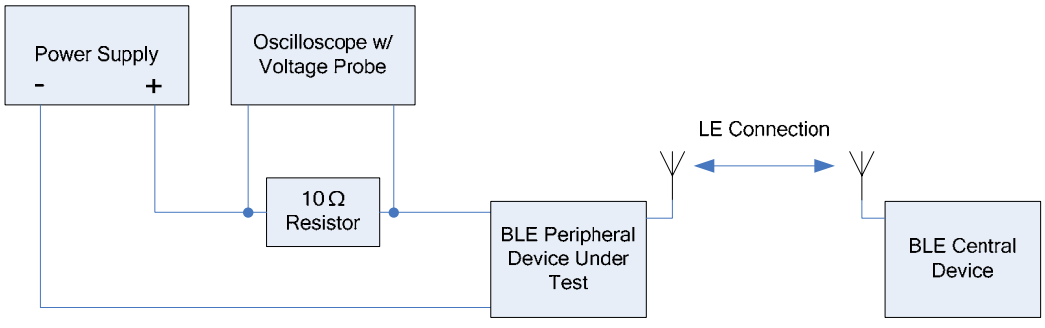

Figure 3- Test Setup using Oscilloscope with Voltage Probe

4.2 Hardware Modifications

In the CC2540DK-MINI kit, the peripheral device is the “keyfob” board. A few simple hardware modifications are required to implement the setup in Figure 3.

1. Solder a wire on the keyfob PCB to ground. An easy location to do this is at pin 1 of the DEBUG connector on the board, as shown in Figure 4.

2. Solder a wire on the keyfob PCB to VDD. An easy location to do this is at the left side (with the board oriented so that the text is read properly) of the unpopulated resistor R1, as shown in Figure 3.

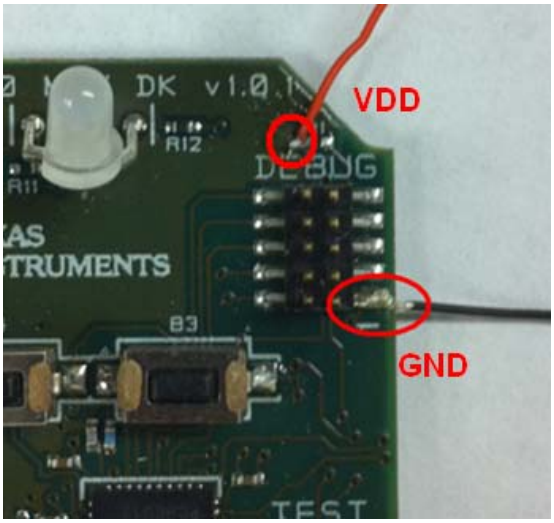

Figure 4- Locations to Solder Wires on Keyfob

3. Solder a 10 Ω resistor to the wire connected to VDD, as shown in Figure 5.

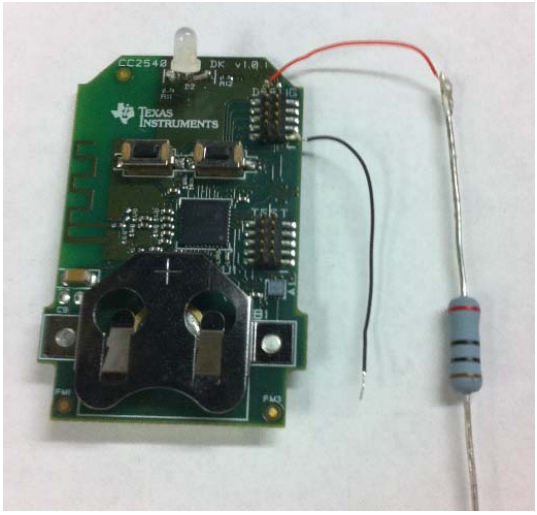

Figure 5- 10 Ω Resistor Soldered in Line with VDD

4. The keyfob board contains a 47uF capacitor (C7) to smooth out the current going into the CC2540 and reduce peaks. For the purposes of power testing, it is best to

remove the capacitor from the board in order to get cleaner and more accurate current measurements. Use a soldering iron to remove C7, as shown in Figure 6.

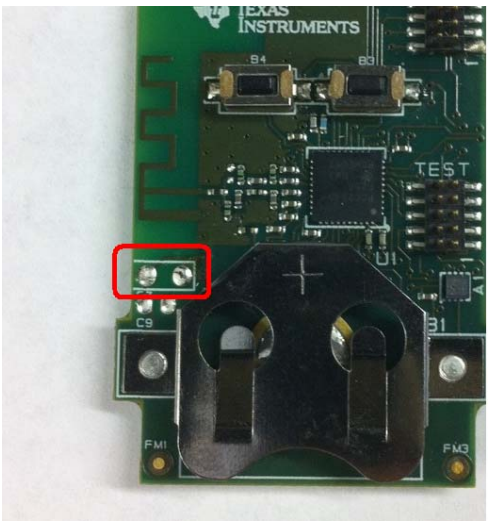

Figure 6- Capacitor C7 Removed

You should now be able connect the DC power supply and oscilloscope voltage probe to the board, with the hardware configured properly for current measurements.

4.3 Embedded Software Modifications

Before testing can begin, the keyfob software must be properly configured. To properly measure current consumption resulting from the BLE stack, additional application processing must be turned off. The SimpleBLEPeripheral project, included with the BLE stack, is simple in that the device immediately starts advertising upon power-up, and will accept any connection request from a master device. A couple of modifications to the software can help to reduce unnecessary power consumption.

4.3.1 Remove Periodic Event

As long as no buttons the keyfob are pressed, the only regular application processing that occurs should be the periodic event, which is set to occur once every five seconds. This reading may throw off the power measurements, and therefore must be removed.

To eliminate the periodic event from the application, simply comment out the following line of source code from the SimpleBLEPeripheral_ProcessEvent function in the file simpleBLEPeripheral.c:

```
// osal_start_timerEx( simpleBLEPeripheral_TaskID, PERIODIC_EVT, PERIODIC_EVT_PERIOD );
```

By commenting out this line, the OSAL timer for the first periodic event will never get set. By preventing the first timer from being set, all subsequent OSAL timers are set after the initial event. This will stop the application from performing unnecessary processing. Once you have implemented this change, rebuild the project and download to the keyfob.

4.3.2 Configure General-Purpose Input / Output (GPIO) Pins

If the GPIO pins are not configured properly, unnecessary current draw may be occur. Ideally, each GPIO pin will be unconnected, and therefore no leakage of current will occur. In the case of the keyfob, as with most boards, many of the GPIO pins are connected to peripheral devices, such as the LEDs, the buzzer, the accelerometer, and the buttons. To maximally reduce the current, all GPIO pins must be set to outputs at a low level. The easiest way to do

this is to add the following lines of code to the very end of the SimpleBLEPeripheral_Init function:

```
P0SEL = 0; // Configure Port 0 as GPIO
P1SEL = 0; // Configure Port 1 as GPIO
P2SEL = 0; // Configure Port 2 as GPIO

P0DIR = 0xFC; // Port 0 pins P0.0 and P0.1 as input (buttons),
              // all others (P0.2-P0.7) as output
P1DIR = 0xFF; // All port 1 pins (P1.0-P1.7) as output
P2DIR = 0x1F; // All port 2 pins (P2.0-P2.4) as output

P0 = 0x03; // All pins on port 0 to low except for P0.0 and P0.1 (buttons)
P1 = 0;   // All pins on port 1 to low
P2 = 0;   // All pins on port 2 to low
```

This will put all of the GPIO pins in a power-optimized state after all of the HAL and board initialization processes complete.

4.4 Central Device and BTool Setup

You will also need to have a central device in order to form a connection with the keyfob. The simplest way to do this is to use the USB Dongle in the CC2540DK-MINI kit running the standard HostTestRelease application, with BTool used to control the dongle.

With dongle connected to the PC and the keyfob powered by the DC power supply, open up BTool. Press the right button on the keyfob to make it discoverable, and then press the “Scan” button in BTool to verify that the dongle and the keyfob are able to talk to each other. The advertisement and scan response data from the keyfob should show up in the BTool log window. You are now ready to form a connection between the devices.

Before forming the connection, the proper connection parameters should be used. This will be dependent on the application that is being considered. The supervision timeout setting should not affect the power measurements; however be sure to use a legal value as per the *Bluetooth* 4.0 specification. For our first example, you will use a connection interval of 1 second, with zero slave latency. Therefore, use the values as shown in Figure 7. Be sure to hit the “Set” button after entering in the values.

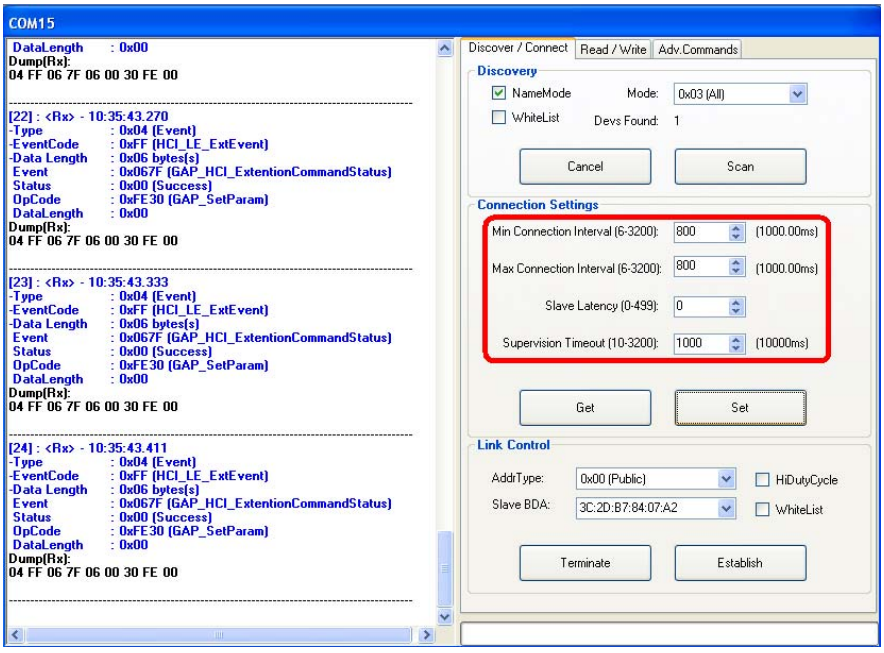

Figure 7- BTool set up for 1 Second Connection Interval and Zero Slave Latency

With the connection parameters set as needed, click the “Establish” button to connect to the keyfob.

5 Measurement and Analysis

If all of the instructions in section 4 were followed properly, the oscilloscope should be set up to measure the voltage across the resistor with respect to time.

5.1 Capturing a Waveform during a Connection Event

In Figure 8, the voltage waveform is captured using a rising-edge trigger, set to trigger each time that the CC2540 wakes up from power mode 2 for a connection event.

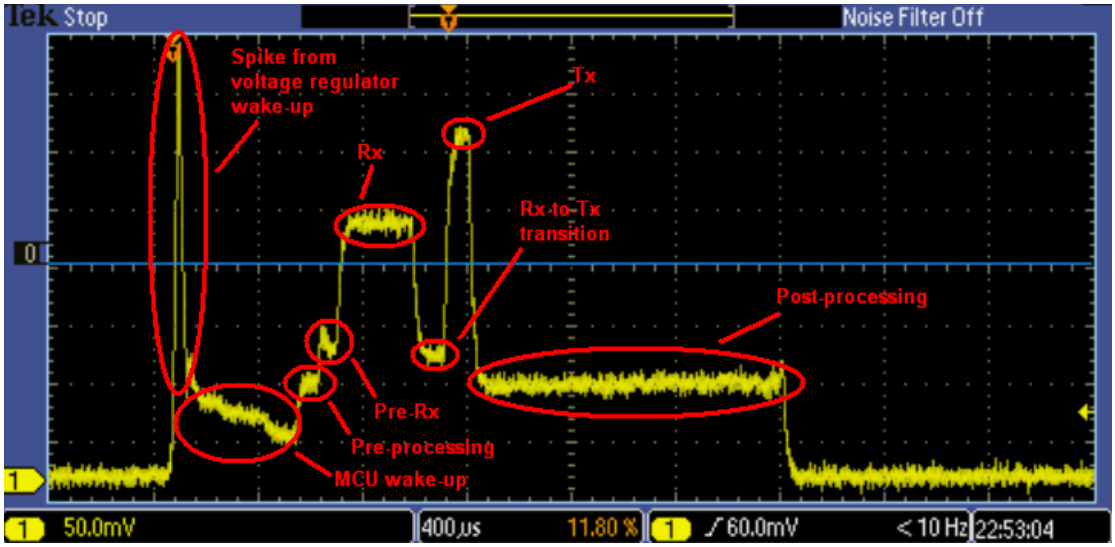

Figure 8- Single-Triggered Capture of Waveform

Because the voltage is measured across a 10 Ω resistor, the current level can easily be calculated by dividing the voltage by 10. One of the first things that you may notice when looking at the capture is a large spike in current at the moment when the MCU wakes up from sleep. This spike is caused by the digital voltage regulator inside the CC2540 re-powering up. The regulator contains capacitors that must be re-charged, and thus quickly draw current when the device wakes up. This spike normally would not appear with capacitor C7 still populated on the board; and therefore while testing power consumption this spike can be ignored.

In addition to the voltage spike, you will notice that the current draw changes as the CC2540 goes through several different states as a part of the connection event:

MCU wake-up – upon waking up, the current level drops slightly

Pre-processing – the BLE protocol stack prepares the radio for sending and receiving data

Pre-Rx – the CC2540 radio turns on in preparation of Rx and Tx

Rx – the radio receiver listens for a packet from the master

Rx-to-Tx transition – the receiver stops, and the radio prepares to transmit a packet to the master

Tx – the radio transmits a packet to the master

Post-processing – the BLE protocol stack processes the received packet and sets up the sleep timer in preparation for the next connection event, before returning back to sleep.

A similar waveform can be captured during every connection event; however the amount of time for each state will vary depending on the circumstances (the size of the PDUs being transmitted / received, the amount of processing time required by the stack, etc.). In addition, if more than one packet is transmitted or received, there will be additional Rx, Tx, and transition states.

While the capture in Figure 8 appears like it can be used to start performing measurements, it does not tell us the entire story. In Figure 9 below, the voltage waveform is captured again using a rising-edge trigger, set to trigger each time that the CC2540 wakes for a connection event. This time, the oscilloscope's "persistence" feature is used, in which many captures are overlaid on top of one another.

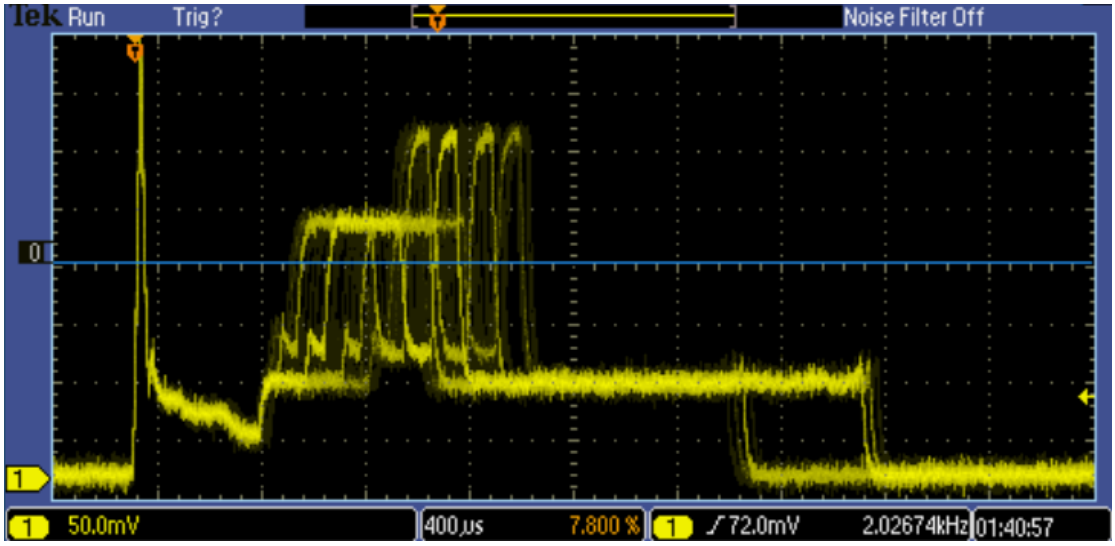

Figure 9- Oscilloscope Capture with Persistence On

From the Figure 9, you can deduce a few interesting facts about the current consumption during connection events. The first point is that the total processing time for each connection event is not always exactly the same. This means that a single scope capture is not sufficient to perform power measurements.

The next point to notice is that even though there is variance in the processing time from event-to-event, the total processing time is not completely random, but rather falls into "slots". Figure 10 below shows a closer view of these slots in which the device completes processing.

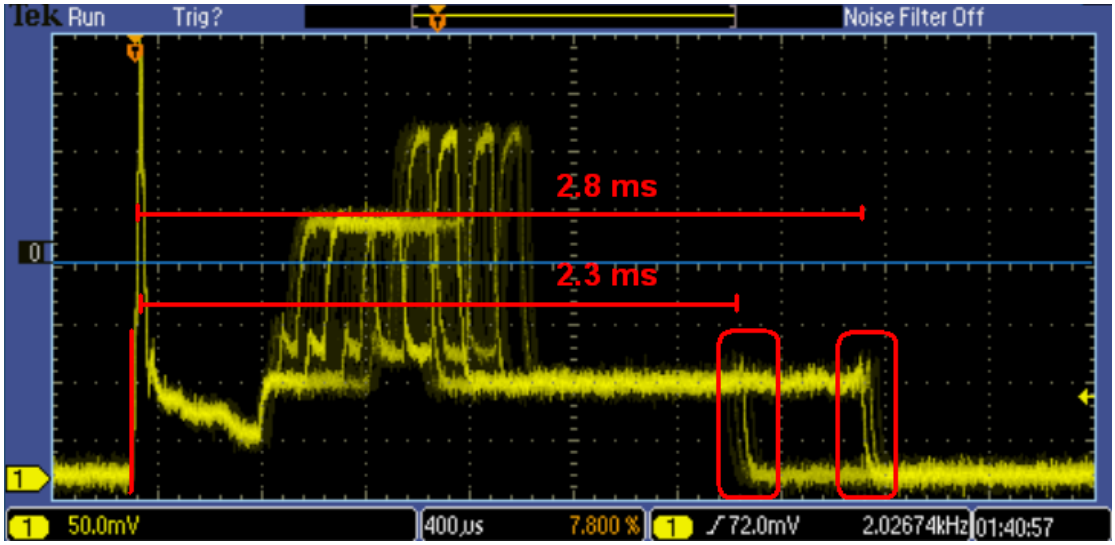

Figure 10- Processing Time "Slots"

Events that fall into the first slot take less time to process, and therefore less power is consumed. In this case, you can see using the oscilloscope's cursors that the CC2540 is awake for approximately 2.3 ms. Events that fall into the second slot take more time to process, and therefore more power is consumed. In this case, the CC2540 is awake for approximately 2.8 ms. In order to get an accurate calculation of the power being consumed, you must take into account the fact that the processing time for connection events is not always the same. A little bit of statistical analysis will be required to get accurate values.

Another point of note from Figure 9 is that the amount of time for receiving and transmitting data during the connection event appears to vary. This in fact is not the case, though it may appear so from the image. The amount of time taken to receive and transmit data is very stable from event to event. This can be seen by changing from a rising-edge trigger to a positive-pulse trigger, and triggering based on the Rx pulse. Set the criteria such that the scope triggers when the positive pulse width is greater than 75 µs. This will ensure that the Rx pulse will cause the triggering, and not the voltage regulator spike. Also be sure to raise the trigger level to a value where it will catch the Rx pulse, and not the pre-processing. A good value to use is 170 mV (which corresponds to 17.0 mA).

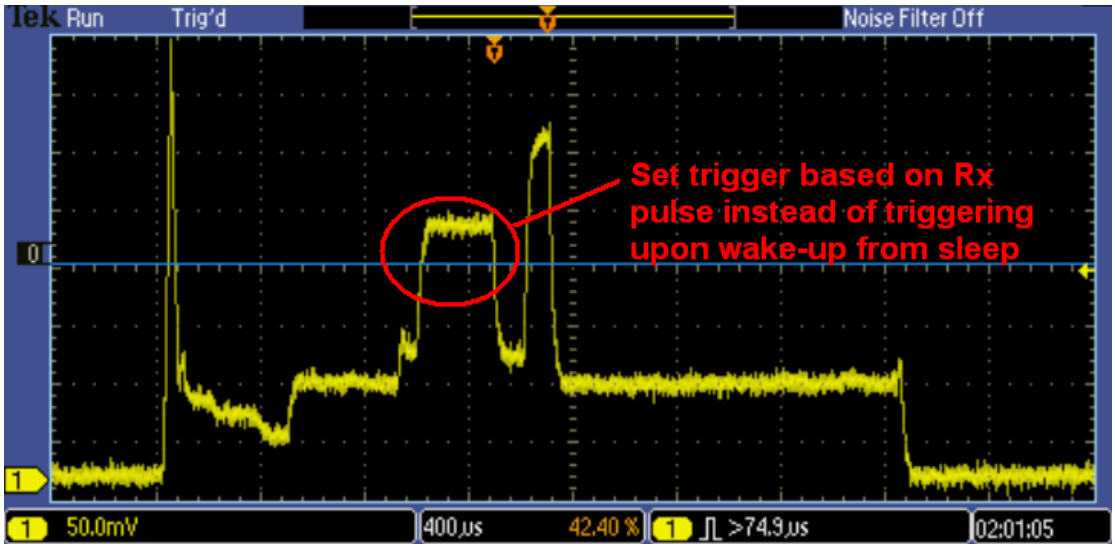

Figure 11- Trigger on Start of Rx

By triggering at this point and watching several connection events over time, it can be seen that the pulses during Rx and Tx are very stable in both their level (voltage) and their width (time).

The only exception to this may be that on occasion, a long Rx pulse may appear as in Figure 12 below. These long pulses mean that for that connection event, the slave device was not able to receive the packet from the master.

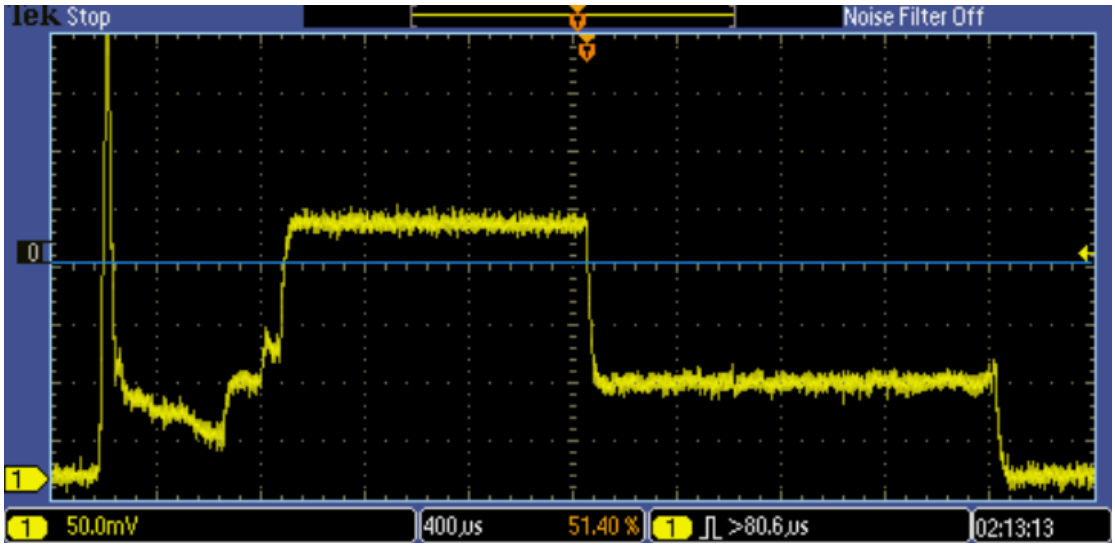

Figure 12- Long Rx Pulse due to Missed Connection Event

You will not see any Tx pulse in this case, because the slave will not respond if it does not receive from the master. These missed events should only occur for a small percentage of packets. If you are seeing many of these, it might mean that there is significant RF interference, or the slave and keyfob devices are far apart from each other.

5.2 Determining Variance in the Length of Connection Events

As mentioned in the previous section, the CC2540 is either awake for approximately 2.3 ms or 2.8 ms during each connection event. To improve the power consumption estimate, you must determine the percentage of the time that each of these cases occurs. Some oscilloscopes contain embedded software for statistical analysis, while others have PC applications that can be used to interface with a scope and perform analysis. If these tools are available this task can be done fairly easily; however even if these options are not available there is a way to estimate this percentage. By simply randomly triggering (be sure to use the rising-edge trigger upon wake-up from sleep) connection events and keeping a count of events in each slot, a good estimate can be made.

In the case of our example, the connection interval is long enough (one second) that you can just leave the trigger running on auto, and keep a tally of each event that falls into each slot. The more events that are watched, the more accurate the calculation will be. By doing this type of analysis, you will find that the 2.3 ms long wake-up occurs approximately 27% of the time, while the 2.8 ms long wake-up occurs 73% of the time.

5.3 Performing Measurements for Connection Events

You are now ready to perform the actual measurements. In the previous section, it was determined that separate measurements are required for the two cases of 2.3 ms long wake-up and 2.8 ms long wake-up. We will first measure using a 2.3 ms long wake-up. This capture can be made either by using special triggering functions on the oscilloscope, or

simply by repeatedly performing single-trigger captures on the scope until a 2.3 ms wake-up waveform is caught (recall that more than one in four events has a 2.3 ms long wake-up, so it should not take too many attempts to get such a capture).

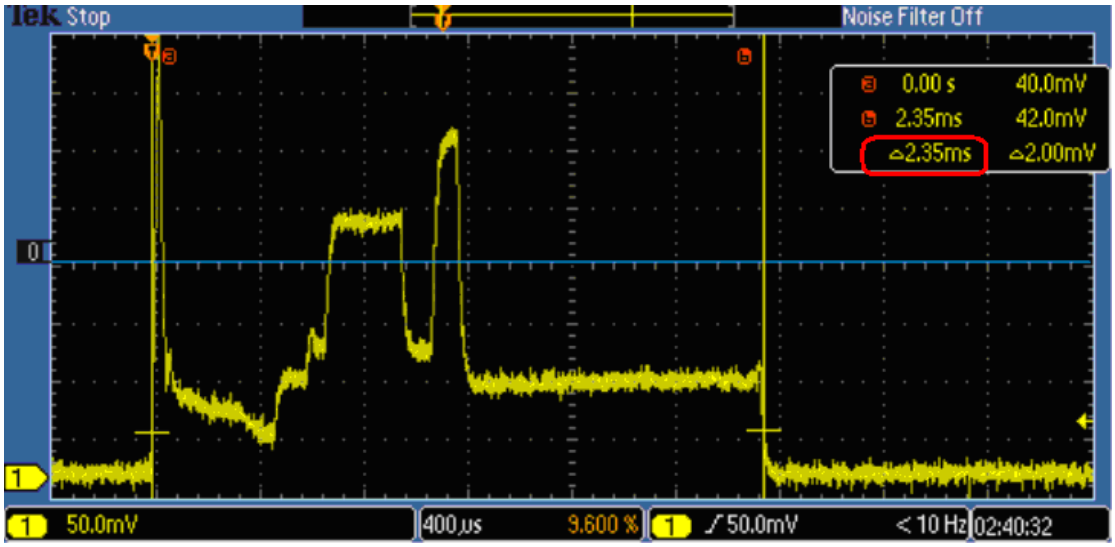

Figure 13- Typical Connection Event with CC2540 Awake for 2.3 ms

To perform measurements, the sections of the waveform must be divided up, with current and timing measured for each state. Figure 14 shows this division of sections for each state.

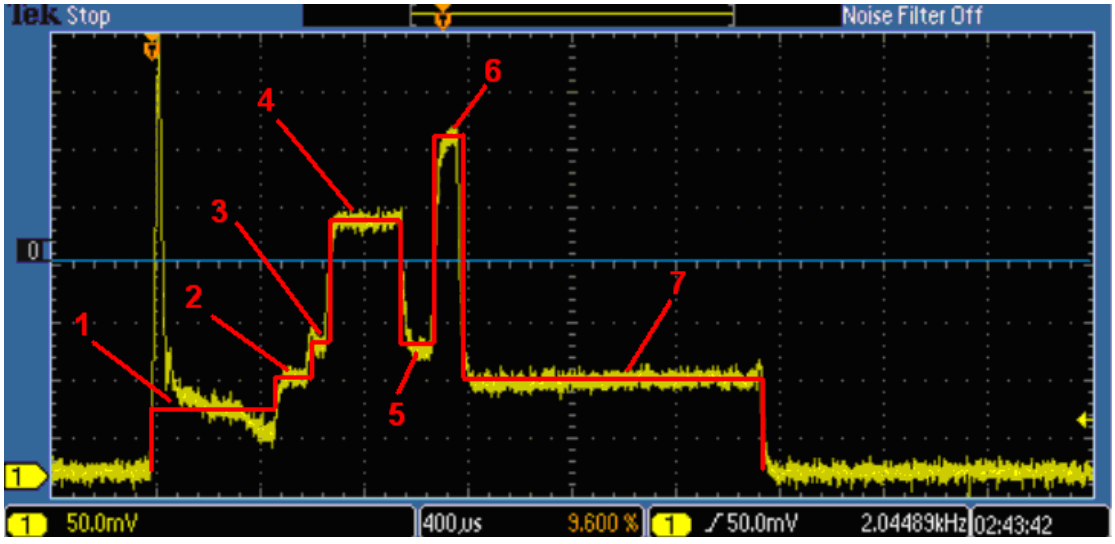

Figure 14- Current Waveform Split into Sections

The oscilloscope cursors can be used to get accurate timing and current measurements for each state. For some states, such as state 1 in Figure 14, the current draw is not steady. For very accurate measurements, the section could be split up into smaller sections; however using the divisions shown and guessing an average current value should provide fairly accurate estimates. Table 1 below shows the measurements from Figure 14:

	Time (µs)	Current (mA)
State 1 (wake-up)	496	6.1
State 2 (pre-processing)	80	8.1
State 3 (pre-Rx)	80	12.3
State 4 (Rx)	288	22.3
State 5 (Rx-to-Tx)	120	11.1
State 6 (Tx)	104	29.3
State 7 (post-processing)	1180	8.1

Table 1- Measurements from Capture in Figure 14

Note that the exact timings of the pre-processing and post-processing may differ; however the sum of the two values will always be the same. Since the current draw is the same during pre-processing and post-processing, these differences will not affect the average current.

Next, similar measurements need to be made using a capture in which the CC2540 is awake for 2.8 ms. Once you have the capture, use the same process as before to take measurements. Doing so will result in the following values:

	Time (µs)	Current (mA)
State 1 (wake-up)	504	6.1
State 2 (pre-processing)	376	8.1
State 3 (pre-Rx)	80	12.3
State 4 (Rx)	288	22.3
State 5 (Rx-to-Tx)	120	11.1
State 6 (Tx)	104	29.3
State 7 (post-processing)	1390	8.1

Table 2- Measurements from Capture with 2.8 ms Awake Time

5.4 Performing Measurements of Sleep Current

In addition to the active current, a very important metric for calculating the battery life is the sleep current. This is important for battery life, because in most use-cases the CC2540 will spend the majority of the time in PM2 while connected, waiting for the next connection event.

The easiest way to measure the PM2 current is to use an ammeter or a digital multimeter (DMM), with the setup shown in Figure 15.

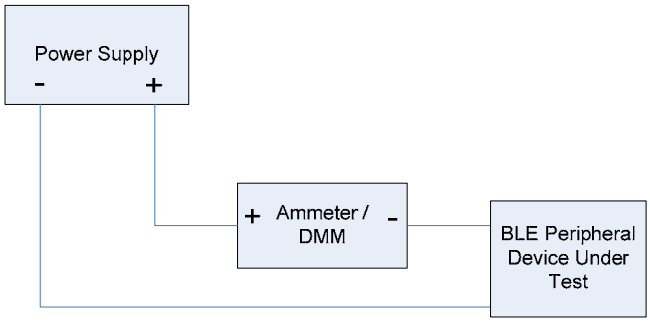

Figure 15- Test Setup using Ammeter for Sleep Current Measurement

It is important not only that your test equipment is capable of making measurements in the µA-range, but it also must be capable of handling current in the mA-range as well while the device powers-up and initializes with the MCU active. Certain ammeters will have separate modes for different ranges of current, and will limit the maximum current when in a low-current mode. With some ammeters / DMMs, it is possible to switch between modes while keeping the circuit connected. If this is the case, as it is with the Fluke 87V Multimeter (shown in Figure 18), the best way to perform measurements is to apply power from the DC power supply with the DMM in mA-mode, and then when the device is in sleep mode switch to µA-mode. It is important that the DMM is switched back to mA-mode before any event occurs that causes higher current to be drawn, as this may overload the DMM and cause unexpected results.

With the hardware modifications described in section 4.2 and the software modifications described in section 4.3, the current measured going into the keyfob will still be around 13.5 µA higher than it should be. This additional current is actually drawn by the peripheral parts on the board, such as the accelerometer, the buzzer, the LEDs, and the buttons. Even though the CC2540 GPIO pins were all set to an optimized state, these parts will still draw residual current from the supply. While this small amount of additional current is negligible when performing active current measurements, it makes a large difference in the total battery life when the sleep current is taken into account.

The simplest way to get a true measurement of the current drawn strictly from the CC2540 is to remove these parts from the board using a soldering iron. If you do not want to remove these components, a CC2540EM board (to be available in the near future as a part of the CC2540DK development kit) can be used instead to perform these measurements.

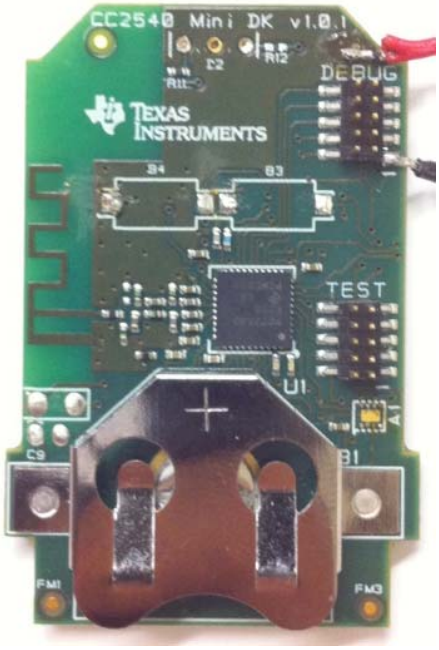

Figure 16- Keyfob Board with Peripherals Removed

With the software modifications described in section 4.3, you should be able to power-up the keyfob, with the device going into PM3 (“Power Mode 3”) immediately after initialization, which is the lowest power mode supported by the CC2540. The reason that the CC2540 goes into PM3 is because the device does not have any timers active, and is simply waiting for an interrupt to wake it up. The PM3 current draw can be measured to be approximately 0.4 µA.

While the PM2 current draw is higher than the PM3 current draw, it is necessary to measure the PM2 current, since the CC2540 goes into PM2 between events while in a connection. Therefore, you will need to establish a connection. Normally, the right button on the keyfob must be pressed in order to make the device discoverable and connectable, but since the buttons have been removed from the board you will have to slightly modify the software to make the device immediately begin advertising upon power-up. This can be done by modify the SimpleBLEPeripheral_Init function to have the variable initial_advertising_enable set to TRUE instead of FALSE, as such:

```
uint8 initial_advertising_enable = TRUE;
```

Be sure to rebuild the code and download it to the keyfob after making the change. Once this has been done, the keyfob should begin advertising upon power-up. In order to easily measure the sleep current, a long effective connection interval should be used. In this example, we will use a 4 second connection interval (the maximum allowed), along with a slave latency setting of 1. This can be set up in BTool using the settings shown in Figure 17.

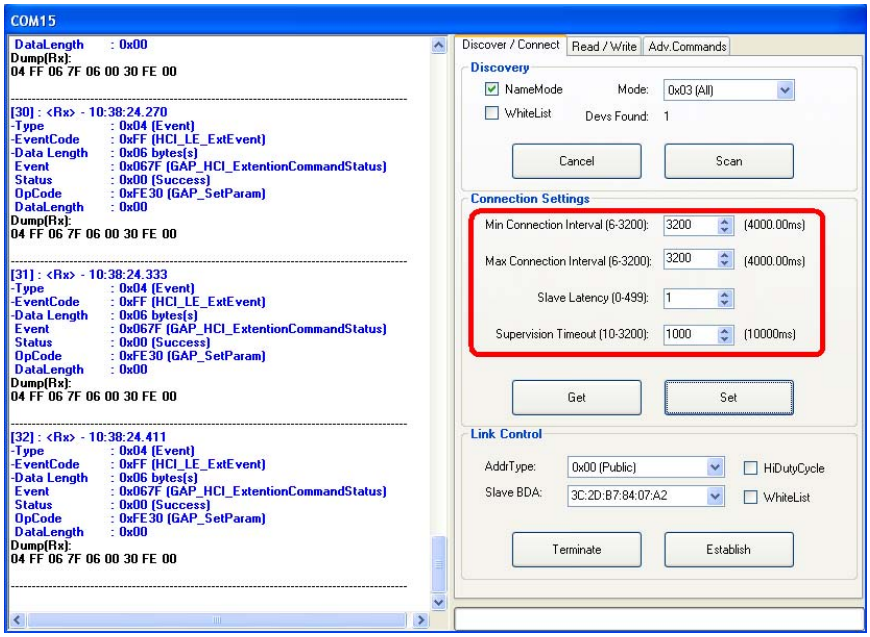

Figure 17- BTool set up for 8 Second Effective Connection Interval

With these settings, the effective connection interval is 8 seconds, meaning that the CC2540 will be in PM2 for 8 seconds between each connection event. Click the “Establish” button to form the connection between the devices.

After the first connection event occurs, the device will go to PM2. At this point, you should see a measured current of 1.0 µA, as shown in Figure 18. As mentioned before, you may need to switch from mA-mode to µA-mode on the ammeter / DMM in order to get a proper measurement. You may also need to switch back to mA-mode within eight seconds, to allow higher current draw during the next connection event.

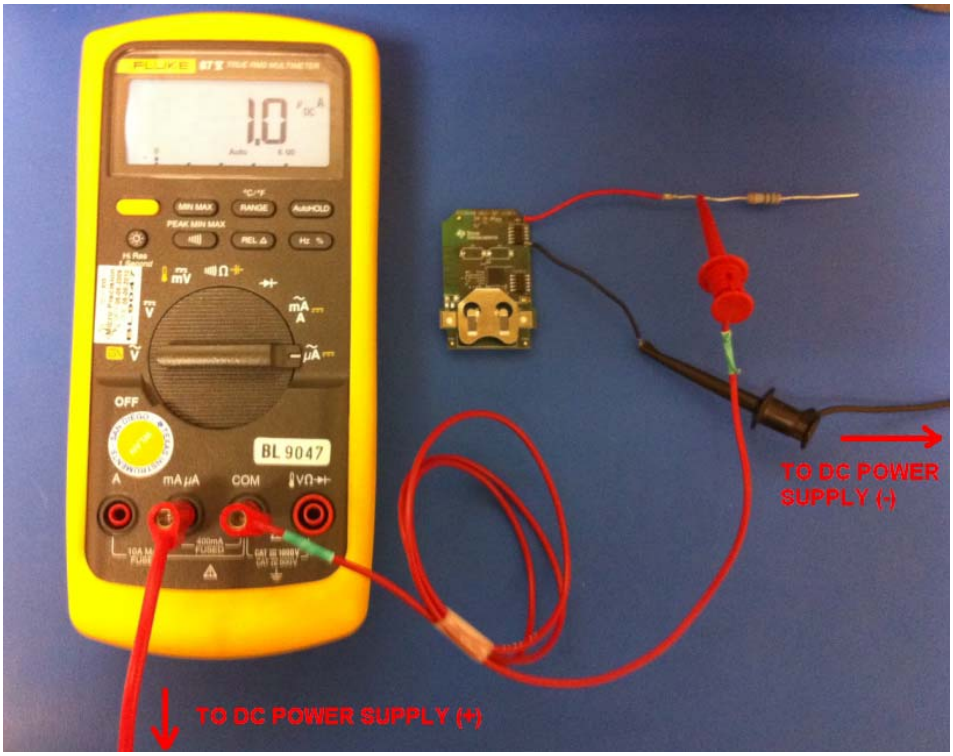

Figure 18- 1.0 µA PM2 Sleep Current Measured on Keyfob

5.5 Formulas and Calculations

Before taking into account the current consumed while the device is sleeping, you can first calculate the average current draw during the connection event. For this, you can use the following formula:

Average current during connection event =
[(State 1 time)*(State 1 current) + (State 2 time)*(State 2 current) + …] / (Total awake time)

It does not matter whether units of µs or ms are used in the time calculation, as long as they are used consistently for each state and for the total. In the case of the 2.3 ms long wake-up, you can values from Table 1 in the formula:

[(496 µs)*(6.1 mA) + (80 µs)*(8.1 mA) + (80 µs)*(12.3 mA) + (288 µs)*(22.3 mA) + (120 µs)*(11.1 mA) + (104 µs)*(29.3 mA) + (1180 µs)*(8.1 mA)] / (2348 µs) = 10.655 mA

The average current consumption during a single connection event with a 2.3 ms wake-up is calculated to be approximately 10.655 mA.

By repeating the calculation using the values from Table 2, we can get the average current consumption for the connection event in the case of 2.8 ms long wake-up:

[(504 µs)*(6.1 mA) + (376 µs)*(8.1 mA) + (80 µs)*(12.3 mA) + (288 µs)*(22.3 mA) + (120 µs)*(11.1 mA) + (104 µs)*(29.3 mA) + (1390 µs)*(8.1 mA)] / (2862 µs) = 10.190 mA

The average current consumption during a single connection event with a 2.8 ms wake-up is calculated to be approximately 10.190 mA.

The next step is to calculate average current for the entire connection interval, which takes into account the time during which the device is sleeping. For this, you can use the following formula:

Average current while connected =
[(Connection Interval – Total awake time)*(Average sleep current) +
(Total awake time)*(Average current during connection event)] / (Connection Interval)

Once again, be sure to use consistent time units throughout the formula. You must also once again perform this calculation twice; once for the 2.3 ms long wake-up, and once for the 2.8 ms long wake-up. Using the values from the previous calculations and the sleep current measurements, you can calculate the average current for the 2.3 ms long wake-up case:

[(1000 ms – 2.348 ms)*(0.001 mA) + (2.348 ms)*(10.655 mA)] / (1000 ms) = 0.026 mA

Repeating the calculation for the 2.8 ms long wake-up case:

[(1000 ms – 2.862 ms)*(0.001 mA) + (2.862 ms)*(10.190 mA)] / (1000 ms) = 0.030 mA

The final step to getting the average current while connected is to take the weighted average of the two averages that were previously calculated. Previously it was determined that the 2.3 ms long wake-up occurred 27% of the time, while the 2.8 ms wake-up occurred 73% of the time. The weighted average is then calculated as follows:

(0.026 mA)*(0.27) + (0.030 mA)*(0.73) = 0.029 mA

The average current consumption while the device is in a connected state is approximately 0.029 mA (28 uA). You can now use this value to calculate the amount of time that you can expect the battery last while running continuously in a connection. The total hours of battery life can be calculated using the following simple formula:

Expected battery life running continuously in a connected state =
(Battery capacity) / (Average current while connected)

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, you can calculate the expected battery life:

(230 mAh) / (0.029 mA) = 7931 hours

The battery can be expected to last for 7931 hours, or approximately 330 days, while running continuously in a connected state with a 1 second connection interval and zero slave latency.

5.6 Using Excel Spreadsheet for Calculations

An Excel spreadsheet is provided along with this application note that can be used to perform simple calculations. The first “EXAMPLE” worksheet in the spreadsheet features the measurements from section 5.3 and 5.4, with the average current draw of 0.029 mA and 330 days of battery life calculated automatically using the measured data.

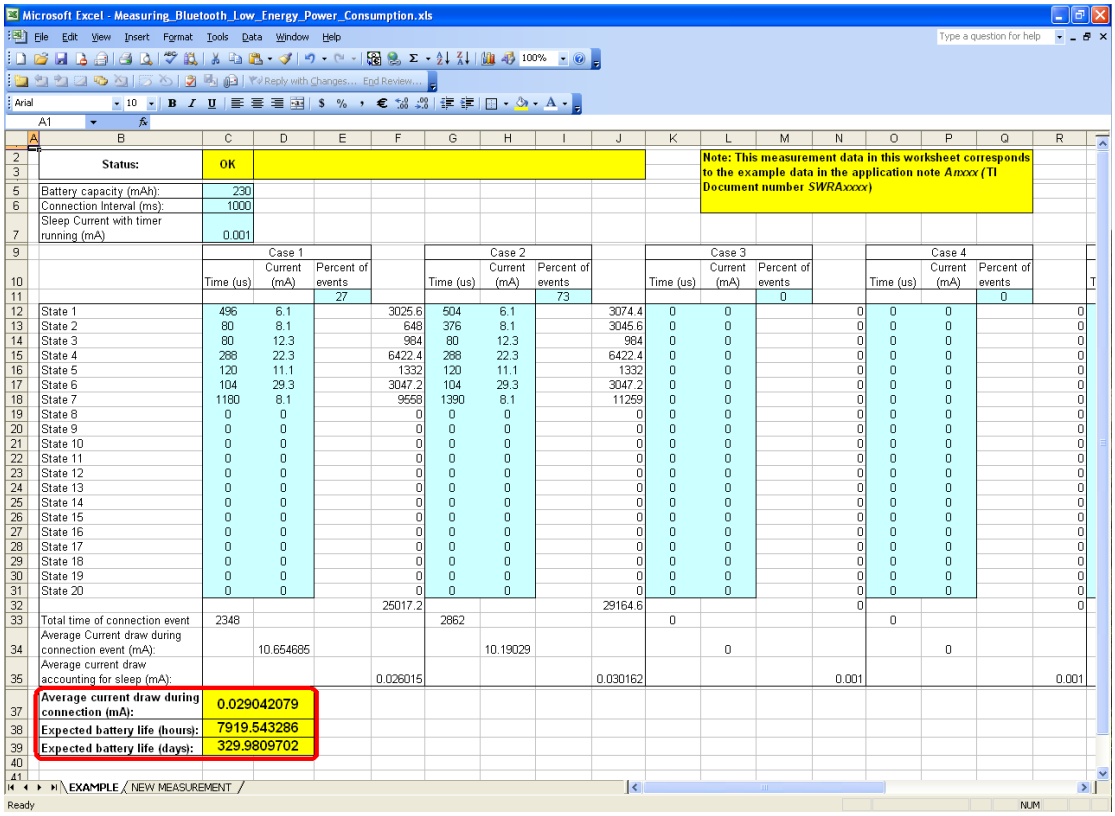

Figure 19- Results from Example Calculated Automatically Using Excel Spreadsheet

The second worksheet, labeled “NEW MEASUREMENT”, is blank and allows you to make current calculations based on your own use-case. All you need to do is set up your connection using the desired connection interval and battery capacity, and perform timing and current measurements similar to those done in the example. The timing, current, connection interval, and battery capacity sections highlighted in light blue are for you fill in. The spreadsheet will then perform the calculations based on those values. The average current, expected battery life in hours, and expected battery life in days will be calculated and displayed in the yellow cells.

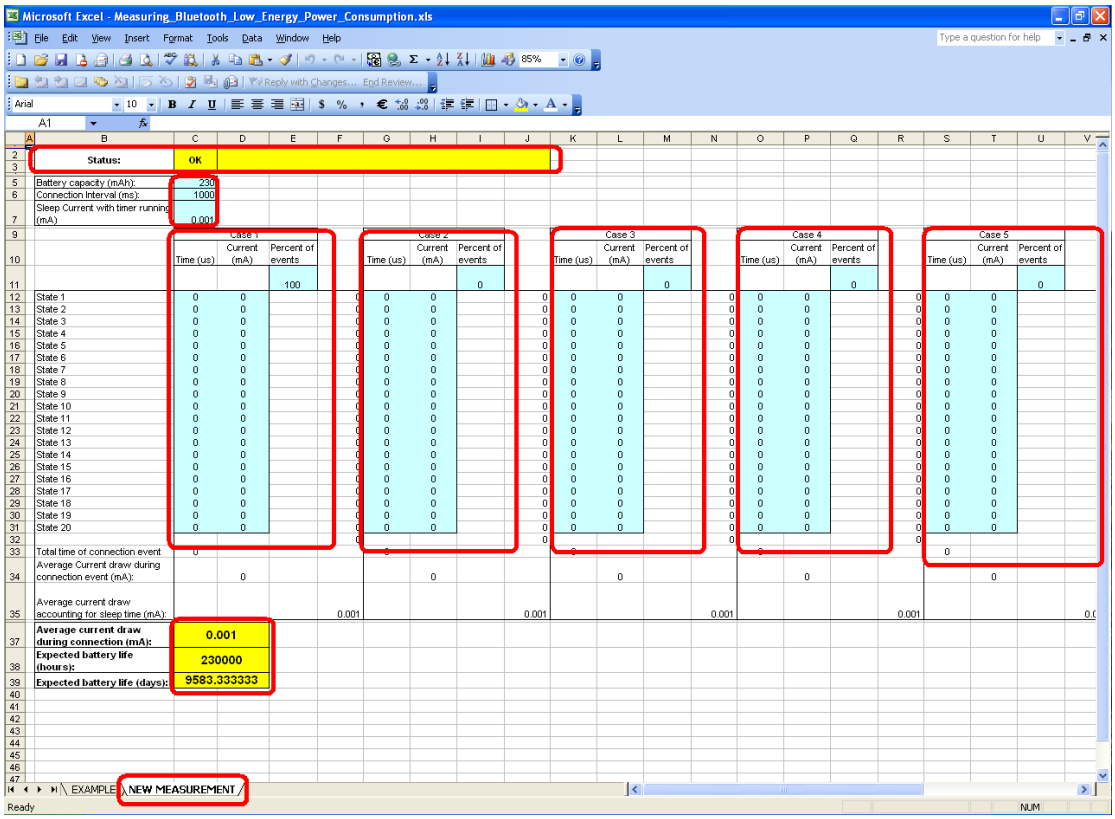

Figure 20- Excel Spreadsheet "NEW MEASUREMENT" Worksheet

Note that even though the example contained seven sections where current and timing was measured, some use-cases might require more sections to be measured. In the previous example no application data was being sent or received over the air; only empty PDUs. In some case, such as in Figure 21 below, one or more packets might be transmitted or received during a connection event. This would require additional sections to be measured. The spreadsheet contains lines for up to 20 different states for each connection event.

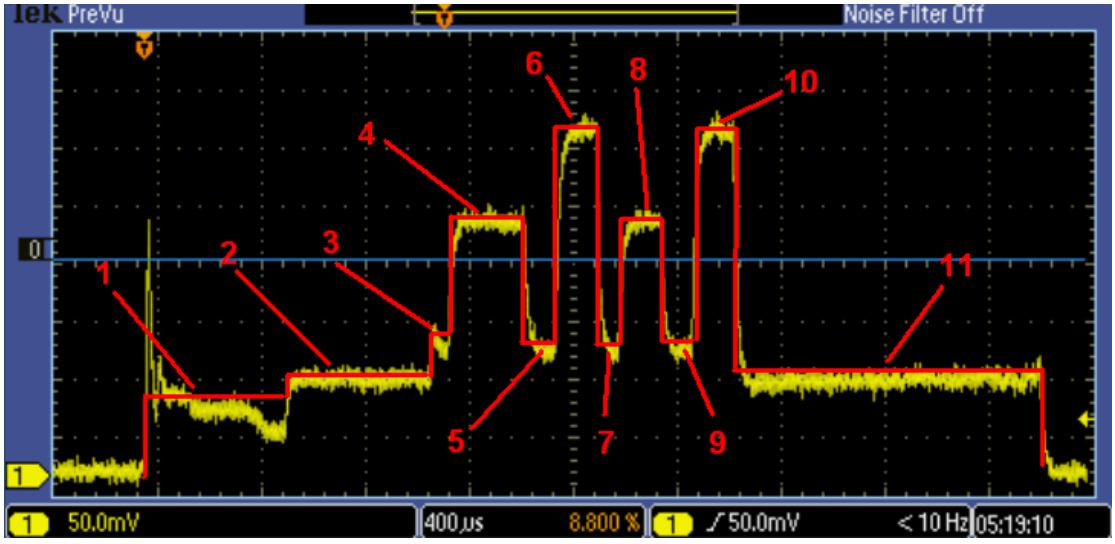

Figure 21- Example of Capture with Multiple Packets Transmitted

In the example from section 5.3 two cases were used, as two different slots of wake-up timing were accounted for. The spreadsheet allows for up to eight different measurements, and lets you set the percentage of events for each use-case. This feature permits you to perform calculations based on more complex use-cases than the one in the example in section 5.3.

For example, the device might transmit a packet once in every five events. The current drawn during the connection event in which the data is transmitted will be higher than that for empty packets, as the processing time will be greater and the Tx pulse will be longer. You can use the oscilloscope to capture an event where the data is transmitted, and perform measurements in the same manner as the example. The data from those measurements can then be entered into the spreadsheet, with the “Percent of events” set to 20 (or if there is variance in the processing time, this can be split into two or more events with the percentage totaling 20). This way, the calculation of battery life will account for multiple situations. Note that if the percentage values do not total up to 100, the calculations will not work and “ERROR!” will be displayed in place of the results.

References

[1] **CC2540 Bluetooth Low Energy Developer's Guide (SWRU271)**

[2] **CC2540DK-MINI User Guide (SWRU270)**

[3] **White paper: Coin Cells And Peak Current Draw (SWRA349)**

General Information

5.7 Document History

Revision	Date	Description/Changes
SWRA347	2010.10.15	Initial release.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DLP® Products	www.dlp.com	Communications and Telecom	www.ti.com/communications
DSP	dsp.ti.com	Computers and Peripherals	www.ti.com/computers
Clocks and Timers	www.ti.com/clocks	Consumer Electronics	www.ti.com/consumer-apps
Interface	interface.ti.com	Energy	www.ti.com/energy
Logic	logic.ti.com	Industrial	www.ti.com/industrial
Power Mgmt	power.ti.com	Medical	www.ti.com/medical
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
RFID	www.ti-rfid.com	Space, Avionics & Defense	www.ti.com/space-avionics-defense
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Video and Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless-apps

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