Bluetooth® Low Energy
CC2540/41 Mini Development Kit
User's Guide

Document Number: SWRU270C

Document Version: 1.2

Development Kit Part Number: CC2540DK-MINI, CC2541DK-MINI

TABLE OF CONTENTS

1. REFERENCES 3

1.1 PRINTED COPY INCLUDED IN THE BOX WITH CC2540DK-MINI 3
1.2 PRINTED COPY INCLUDED IN THE BOX WITH CC2541DK-MINI 3
1.3 INCLUDED WITH TEXAS INSTRUMENTS *BLUETOOTH* LOW ENERGY SOFTWARE INSTALLER 3
1.4 AVAILABLE FROM *BLUETOOTH* SPECIAL INTEREST GROUP (SIG) 3

2. INTRODUCTION 4

2.1 KIT CONTENTS OVERVIEW 4
2.2 SYSTEM REQUIREMENTS 5

3. GETTING STARTED 6

3.1 ASSOCIATE DRIVER WITH USB DONGLE 6
3.2 DETERMINING THE COM PORT 7

4. USING BTOOL 9

4.1 STARTING THE APPLICATION 9
4.2 CREATING A BLE CONNECTION BETWEEN USB DONGLE AND KEYFOB 10
4.2.1 Making the Keyfob Discoverable 10
4.2.2 Scanning for Devices 11
4.2.3 Selecting Connection Parameters 12
4.2.4 Establishing a Connection 12
4.3 KEYFOBDEMO PROFILES 14
4.3.1 Reading a Characteristic Value by UUID 14
4.3.2 Reading a Characteristic Value by Handle 14
4.3.3 Writing a Characteristic Value 15
4.3.4 Discovering a Characteristic by UUID 15
4.3.5 Reading Multiple Characteristic Values 15
4.3.6 Enable Notifications 15
4.4 USING THE PROXIMITY PROFILE 16
4.4.1 Activate Link Loss Service 16
4.4.2 Activate Immediate Alert 17
4.4.3 Read TX Power 17
4.5 USING THE BATTERY SERVICE 19
4.5.1 Read the Battery Level 19
4.5.2 Activate Battery Level Notification 19
4.6 USING THE ACCELEROMETER SERVICE 21
4.6.1 Enable Accelerometer 21
4.6.2 Enable Accelerometer Notifications 22
4.7 USING THE SIMPLE KEYS GATT PROFILE 23
4.7.1 Enable Simple Keys notifications 23
4.8 USING BLE SECURITY 24
4.8.1 Encrypting the Connection 24
4.8.2 Using Bonding and Long-Term Keys 25
4.9 ADDITIONAL SAMPLE APPLICATIONS 28

5. PROGRAMMING / DEBUGGING THE CC2540 OR CC2541 29

5.1 HARDWARE SETUP FOR KEYFOB 29
5.2 HARDWARE SETUP FOR USB DONGLE 31
5.3 USING SMARTRF FLASH PROGRAMMER SOFTWARE 33
5.3.1 Reading or Writing a Hex File to the CC2540/41 33
5.3.2 Reading or Writing the CC2540/41 Device Address 34

6. SMARTRF™ PACKET SNIFFER 35

7. GENERAL INFORMATION 36

7.1 DOCUMENT HISTORY 36

APPENDIX 37

1. References

The following references provide additional information on the CC2540, CC2541, the Texas Instruments *Bluetooth®* low energy (BLE) stack, and the BLE specification in general. (All path and file references in this document assume that the BLE development kit software has been installed to the default path C:\Texas Instruments\BLE-CC254x-1.3\)

1.1 Printed Copy Included in the Box with CC2540DK-MINI

[1] CC2540 Mini Development Kit Quick Start Guide (SWRU272)

1.2 Printed Copy Included in the Box with CC2541DK-MINI

[2] CC2541 Mini Development Kit Quick Start Guide (SWRU332)

1.3 Included with Texas Instruments *Bluetooth* Low Energy Software Installer

(The software installer is available for download at **www.ti.com/ble-stack**)

[3] Texas Instruments *Bluetooth®* Low Energy Software Developer's Guide (SWRU271)
C:\Texas Instruments\BLE-CC254x-1.3\Documents\TI_BLE_Software_Developer's_Guide.pdf

[4] TI BLE Vendor Specific HCI Reference Guide
C:\Texas Instruments\BLE-CC254x-1.3\Documents\TI_BLE_Vendor_Specific_HCI_Guide.pdf

[5] Texas Instruments BLE Sample Applications Guide (SWRU297)
C:\Texas Instruments\BLE-CC254x-1.3\Documents\TI_BLE_Sample_Applications_Guide.pdf

1.4 Available from *Bluetooth* Special Interest Group (SIG)

[6] *Specification of the Bluetooth System*, Covered Core Package version: 4.0 (30-June-2010)
https://www.bluetooth.org/technical/specifications/adopted.htm

2. Introduction

Thank you for purchasing a Texas Instruments (TI) *Bluetooth*® low energy (BLE) Mini Development Kit. The purpose of this document is to give an overview of the hardware and software included in the CC2540 Mini Development Kit (CC2540DK-MINI) and the CC2541 Mini Development Kit (CC2541DK-MINI).

The information in this guide will get you up and running with the kit; however for more detailed information on BLE technology and the TI BLE protocol stack, please consult the Texas Instruments *Bluetooth*® Low Energy Software Developer's Guide [3].

2.1 Kit Contents Overview

The kits contain the following hardware components including cables:

	CC2540 Keyfob	CC2541 Keyfob	CC2540 USB Dongle	CC Debugger
CC2540DK-MINI	•		•	•
CC2541DK-MINI		•	•	•

The **CC2540/41 Keyfob** is designed to act as a Peripheral Device (BLE Slave). Plastic casing for the keyfob is included which can be assembled by a Phillips screwdriver. The keyfob operates on a single CR2032 coin cell battery and includes a two-colored LED, a buzzer, an accelerometer and two buttons.

The main difference between the CC2540 and CC2541 is a peripheral hardware change; Where CC2540 has a USB interface, the CC2541 has an I2C interface. The CC2541 is also optimized for lower power consumption.

The keyfob uses SPI to interface to a 3 axis accelerometer. Note that the accelerometer sensors are different on the two keyfobs:

- CC2540 Keyfob v1.1 uses CMA3000d
- CC2541 Keyfob v1.4 uses BMA250

The CC2541 Keyfob has an LDO (TPS62730) to lower the voltage to 2.1 Volt for lower current consumption. It is possible to power the CC2541 Keyfob via the CC debugger by mounting a jumper on the header pin P1. The same can be done on the CC2540 Keyfob by mounting a 0 Ohm resistor on the R1 placeholder.

The **CC2540 USB Dongle** can be used to emulate any Bluetooth low energy behavior but is usually acting as a Central Device (BLE Master). It connects to a Windows PC's USB Port, and is pre-loaded with necessary software to run the PC application BTool.

The **CC Debugger** is used to flash the software onto both the USB Dongle as well as the keyfob. It can also be used for debugging software using IAR Embedded Workbench.

Figure 1 – Hardware Included with CC2540DK-MINI

The RF Boards in this kit are FCC and IC certified and tested to comply with ETSI/R&TTE over temperature from 0 to +35°C.

Caution! The kits include a non-rechargeable lithium battery. Always make sure the battery is removed from the CC2540/41 Keyfob when it is connected to an external power source (Do not apply voltage > 3.6V). Dispose the battery properly and keep out of the reach of children. If swallowed, contact a physician immediately.

Caution! The kits contain ESD sensitive components. Handle with care to prevent permanent damage.

2.2 System Requirements

To use the TI BLE software, a PC running Microsoft Windows (XP or later) is required, as well as Microsoft .NET Framework 3.5 Service Pack 1 (SP1) or greater.

In order to check whether your system has the appropriate .NET Framework, open up the Windows Control Panel, and select “Add or Remove Programs”. Amongst the list of currently installed programs, you should see “Microsoft .NET Framework 3.5 SP1”, as shown in Figure 2:

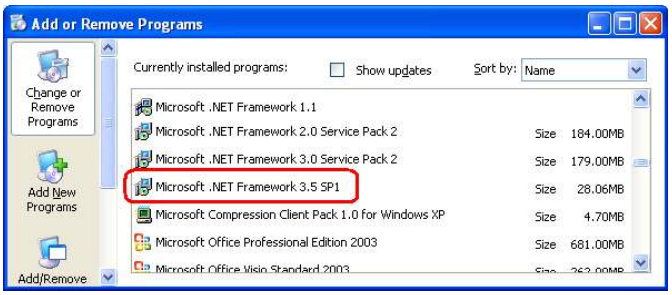

Figure 2 System Requirements, .NET Framework 3.5 SP1

If you do not see it in the list, you can download the framework from Microsoft.

From a hardware standpoint, the Windows PC must contain one free USB port. An additional free USB port is required in order to use the CC Debugger and the USB Dongle simultaneously.

IAR Embedded Workbench for 8051 development environment is required in order to make changes to the keyfob software. More information on IAR can be found in the Texas Instruments *Bluetooth*® Low Energy Software Developer’s Guide [1].

For the keyfob, a small **Philips screwdriver** (not included in the kit) is required if you want to enclose the keyfob in the plastic case, and a CR2032 coin cell battery (included in the kit) is required for power.

3. Getting Started

This section describes how to set up the software and get started with the Mini Development Kit. It is assumed that the CC2540/41 Keyfob comes pre-programmed out of the box. If not (as for older versions of the kit), please see Chapter 5 for details on how to program the keyfob with the latest firmware. It is indicated in the printed copy of the quick start guide that follows with the kit, if the keyfob and CC2540 USB Dongle has been pre-programmed. In addition, this section assumes that the latest version of the TI BLE software (v1.3 as of the release of this document) has been installed. The latest BLE software can be downloaded at www.ti.com/ble-stack.

3.1 Associate Driver with USB Dongle

After the software installation is complete, the USB Dongle driver must be associated with the device in order to use the demo application. To associate the USB Dongle driver, first you must connect the USB Dongle to the PC's USB port, or to a USB hub that connects to the PC.

The first time that the dongle is connected to the PC, a message will most probably pop-up, indicating that Windows does not recognize the device.

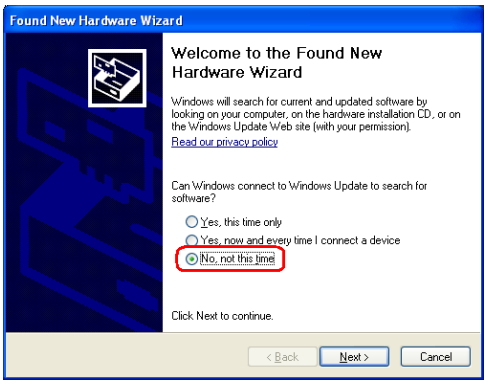

Figure 3 PC, Found New Hardware

When prompted whether to use Windows Update search for software, select "No, not this time" and press the "Next" button. On the next screen, select the option "Install from a list or specific location (Advanced)", and press the "Next" button:

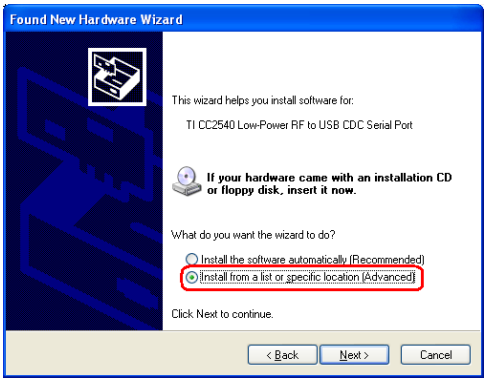

Figure 4 PC, Install Driver

On the next screen, click the checkbox labeled "Include this location in the search:", and click the "Browse" button. Select the following directory (assuming the default installation path was used):

C:\Texas Instruments\BLE-CC254x-1.3\Accessories\Drivers

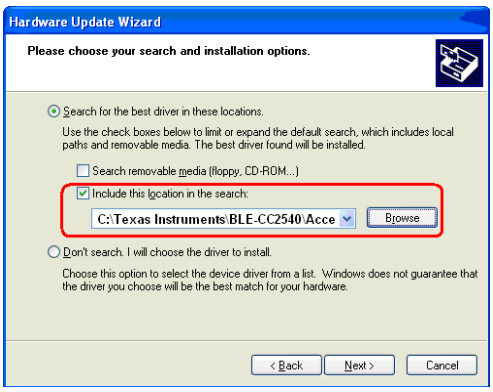

Figure 5 PC, Select Driver

Click the “Next” button. This should install the driver. It will take a few seconds for the file to load. If the installation was successful, you should see the screen to the below. Click the “Finish” button to complete the installation.

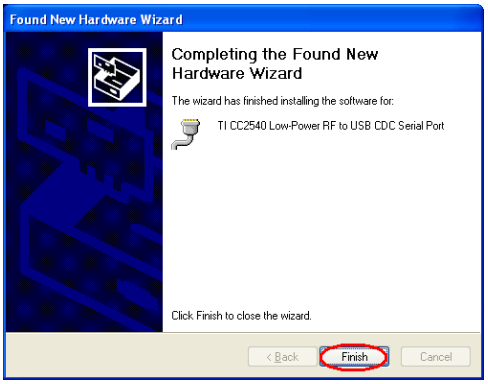

Figure 6 PC, CDC Driver Installation Complete

3.2 Determining the COM Port

Once the driver is installed, you need to determine which COM port Windows has assigned to the USB Dongle. After you have completed the USB Dongle driver association in section 3.1, right-click on the “Computer” icon on your Start and select “Properties”, as shown in Figure 6.

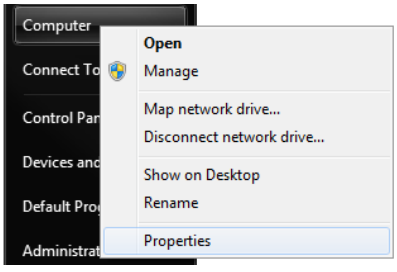

Figure 7 Win7 PC, Finding Computer Properties

The “System Properties” window should open up. Click “Device Manager as shown in Figure 8.

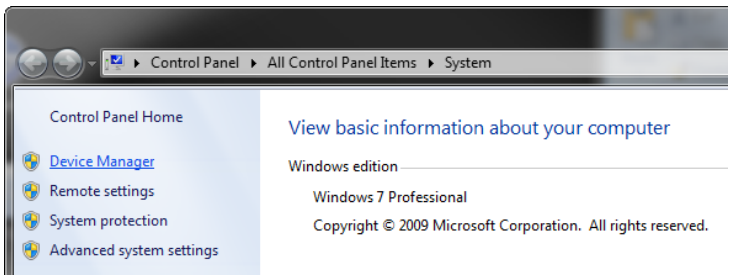

Figure 8 Win7 PC, Finding Device Manager

A list of all hardware devices should appear. Under the section “Ports (COM & LPT)”, the device “TI CC2540 Low-Power RF to USB CDC Serial Port” should appear. Next to the name should be the port number (for example, the CC2540USB Dongle uses COM8 in Figure 9).

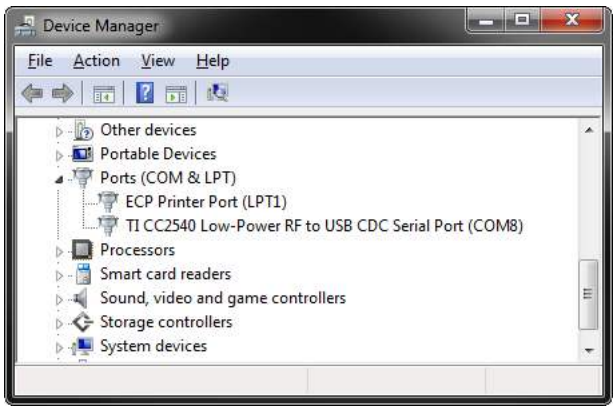

Figure 9 Win7 PC, Connected Ports List

Take note of this port number, as it will be needed in order to use BTool. You may close the device manager at this point.

4. Using BTool

BTool is a PC Application that allows a user to form a connection between two BLE devices. BTool works by communicating with the CC2540, acting as a network processor, by means of HCI vendor specific commands. The USB Dongle software (when running the HostTestRelease project) and driver create a virtual serial port over the USB interface. BTool, running on the PC, communicates with the USB Dongle through this virtual serial port.

More information on the network processor configuration and the HostTestRelease project can be found in the Texas Instruments *Bluetooth*® Low Energy Software Developer's Guide [3]. More information on the HCI interface, as well as details on the HCI vendor specific commands that are used by the CC2540/41, can be found in the TI BLE Vendor Specific HCI Reference Guide [4].

For this section, a PC running windows 7 has been used, but the procedures are essentially the same for other windows version, such as XP.

4.1 Starting the Application

To start the application go into your programs by choosing Start > Programs > Texas Instruments > BLE-CC254x-1.3 > BTool. On Start-up you should be able to set the Serial Port Settings. Set the "Port" value to the COM port earlier noted in Section 3.2. For the other settings, use the default values as shown in Figure 10. Press "OK" to connect to the CC2540 USB Dongle.

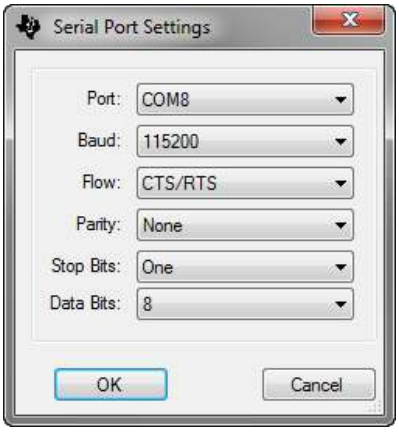

Figure 10 BTool, Serial Port settings

When connected you should see the screen presented in Figure 11. The screen indicates that you now have a serial port connection to the CC2540 USB Dongle. The screen is divided up into a few sections: the left sidebar contains information on the CC2540 USB Dongle status. The left side of the sub-window contains a log of all messages sent from the PC to the CC2540 USB Dongle and received by the PC from the CC2540 USB Dongle. The right side of the sub-window contains a GUI for control of the CC2540 USB Dongle.

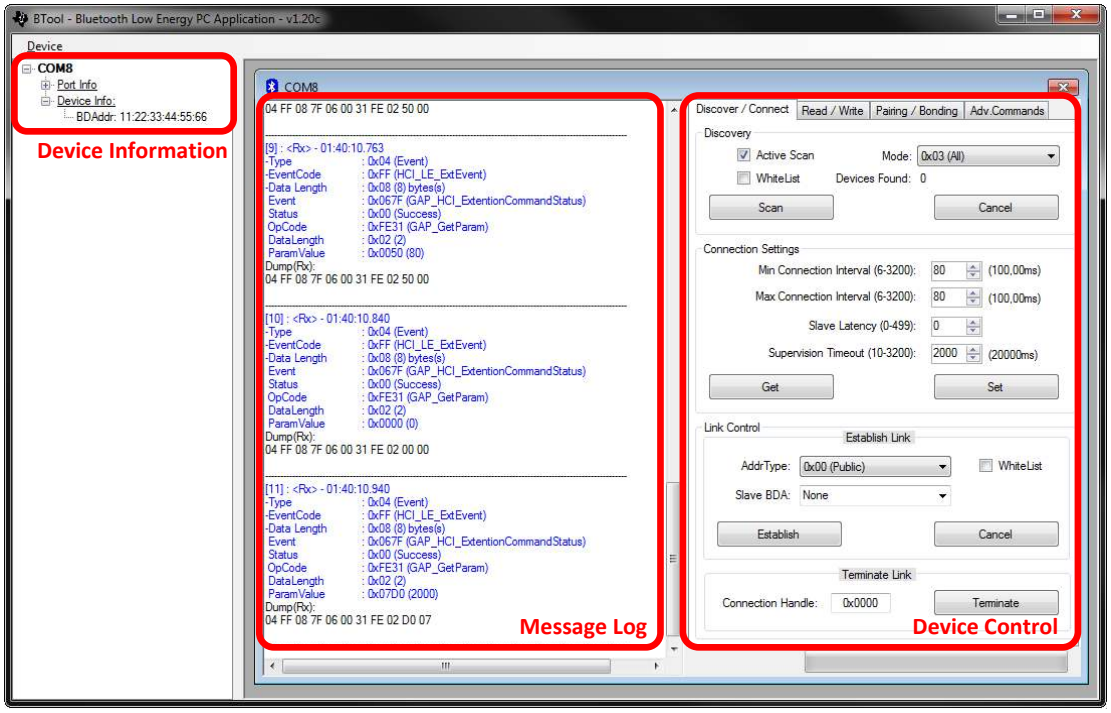

Figure 11 BTool, Overview

4.2 Creating a BLE Connection between USB Dongle and Keyfob

At this point the USB Dongle (central) is ready to discover other BLE devices that are advertising. The keyfob should be preloaded with the KeyFobDemo application. The full project and application source code files for KeyFobDemo is included in the BLE software development kit.

At this time you will want to insert the battery (or remove and re-insert the battery to reset the device) into the keyfob (peripheral). When you insert the CR 2032 battery, the LED will be lit green for one second.

4.2.1 Making the Keyfob Discoverable

When the keyfob powers up, it will not immediately go into a discoverable state. To enable advertising and make the keyfob discoverable, press the right-hand button on the keyfob once. This will turn advertisements on; making the device discoverable for 30 seconds (this value is defined in the *Specification of the Bluetooth System* [6]). After that time, the device will return to standby mode. To make the device discoverable again, simply press the button once again. During discoverable mode, the LED will flash red.

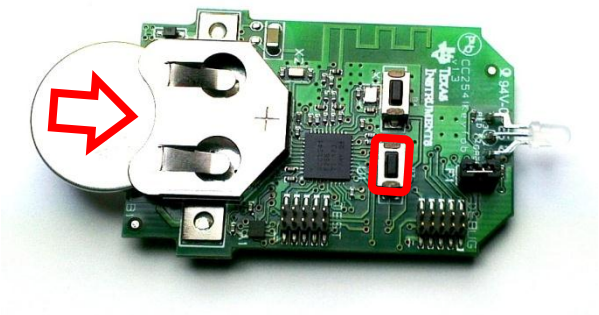

Figure 12 Press Right Button to Turn On Advertisements

4.2.2 Scanning for Devices

In BTool, Press the “Scan” button under the “Discover / Connect” tab, as shown in Figure 13.

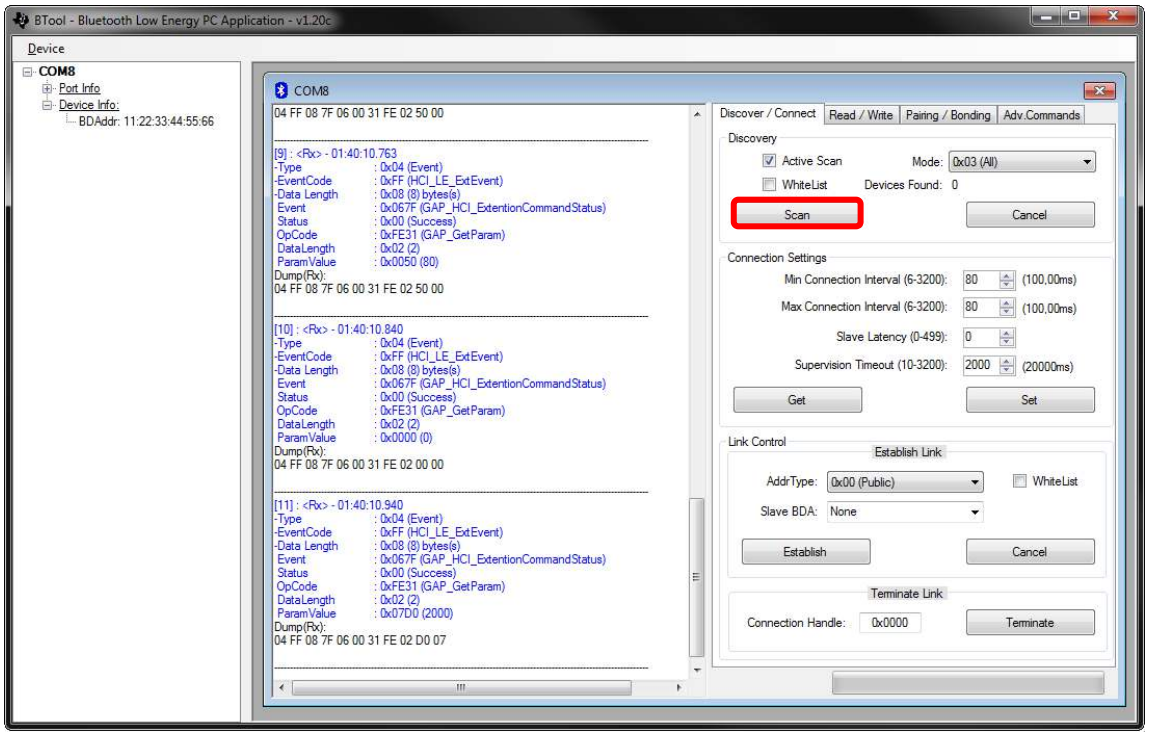

Figure 13 BTool, Scan for Devices

The USB Dongle will begin search for other BLE devices. As devices are found, the log on the left side of the screen will display the devices discovered. After 10 seconds, the device discovery process will complete, and the USB Dongle will stop scanning. A summary of all the scanned devices will be displayed in the log window. In the example in Figure 14, one peripheral device was discovered while scanning. If you do not want to wait through the full 10 seconds of scanning, the “Cancel” button can be pressed alternatively, which will stop the device discovery process. The address of any scanned devices will appear in the “Slave BDA” section of the “Link Control” section in the bottom right corner of the sub-window.

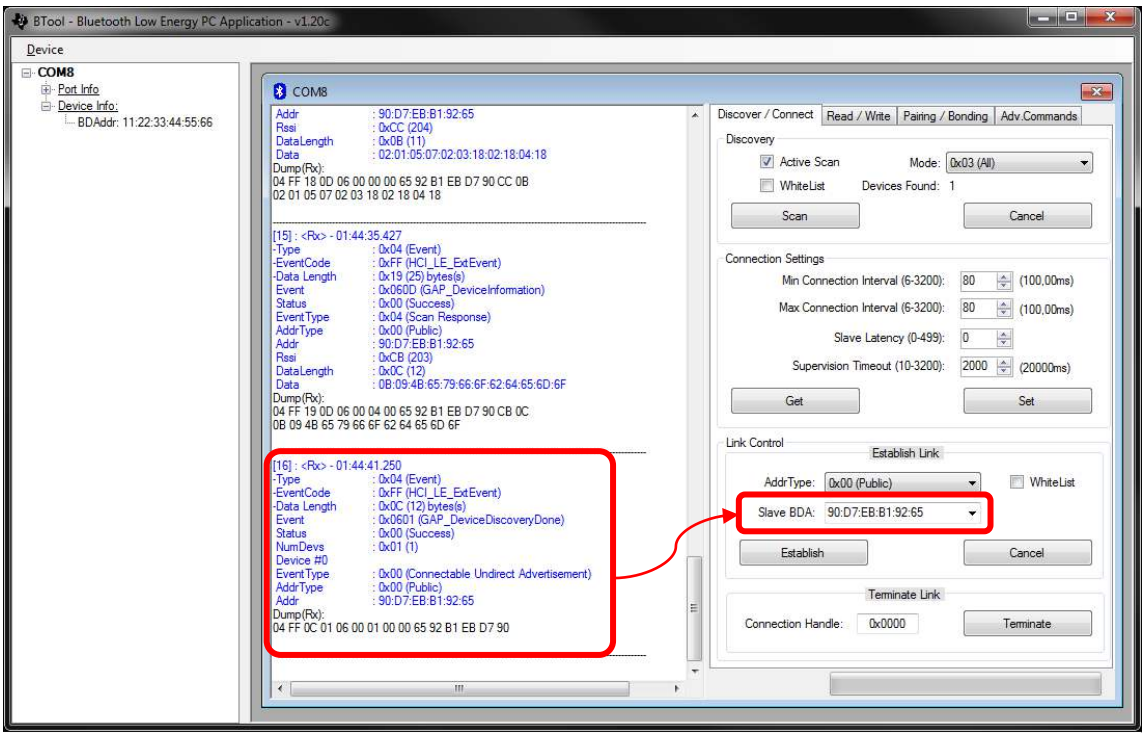

Figure 14 BTool, Slave Address

4.2.3 Selecting Connection Parameters

Before establishing a connection, you can set up the desired connection parameters. The default values of 100ms connection interval, 0 slave latency, and 20s supervision timeout should serve as a good starting point; however for different applications you may want to experiment with these values.

Once the desired values have been set, be sure to click the “Set” button; otherwise the settings will not be saved. Note that the connection parameters must be set before a connection is established; changing the values and clicking the “Set” button while a connection is active will not change the settings of an active connection. The connection must be terminated and re-established to use the new parameters. (The *Bluetooth* specification does support connection parameter updates while a connection is active; however this must be done using either an L2CAP connection parameter update request, or using a direct HCI command. More information can be found in the *Specification of the Bluetooth System* [6])

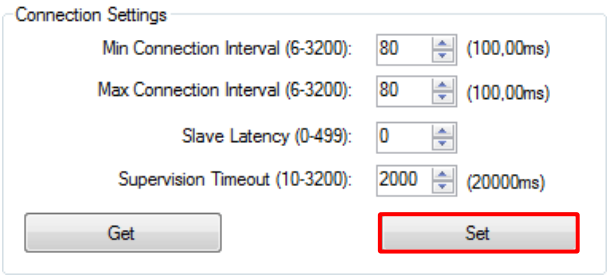

Figure 15 BTool, Connection Settings

4.2.4 Establishing a Connection

To establish a connection with the keyfob, select the address of the device to connect with, and click the “Establish” button as shown in Figure 16.

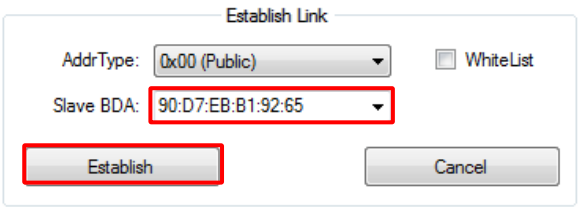

Figure 16 BTool, Establish Connection

If the keyfob is still in discoverable mode, a connection should be established (if more than 30 seconds have passed since the device was previously made discoverable, press the right button on the keyfob once again). Once a connection is established, the message window will return a "GAP_EstablishLink" event message with a "Status" value of "0x00 (Success)" as shown in Figure 17.

```
[19] : <Rx> - 01:45:37.225
-Type              : 0x04 (Event)
-EventCode         : 0xFF (HCI_LE_ExtEvent)
-Data Length       : 0x13 (19) bytes(s)
Event              : 0x0605 (GAP_EstablishLink)
Status             : 0x00 (Success)
DevAddrType        : 0x00 (Public)
DevAddr            : 90:D7:EB:B1:92:65
ConnHandle         : 0x0000 (0)
ConnInterval       : 0x0050 (80)
ConnLatency        : 0x0000 (0)
ConnTimeout        : 0x07D0 (2000)
ClockAccuracy      : 0x00 (0)
Dump(Rx):
04 FF 13 05 06 00 00 65 92 B1 EB D7 90 00 00 50
00 00 00 D0 07 00
```

Figure 17 BTool Log, Link Established

In BTool, you can see your connected peripheral device in the Device Information field, as shown in Figure 18.

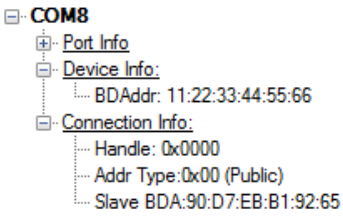

Figure 18 BTool, Device Information

4.3 KeyFobDemo Profiles

The KeyFobDemo software contains several GATT service profiles (More information on the KeyFobDemo can be found in the Texas Instruments BLE Sample Applications Guide [5]). GATT services contain data values known as “characteristic values”. All application data that is being sent or received in *Bluetooth* low energy must be contained within characteristic value. This section details a step-by-step process that demonstrates several processes for reading, writing, discovering, and notifying GATT characteristic values using BTool.

Profiles/Services included in the KeyFobDemo are:

- Proximity Profile
  - Link Loss Service
  - Immediate Alert Service
  - TX Power Level Service
- Battery Service
- Accelerometer Service
- Simple Keys Service

In a *Bluetooth* low energy system, upon connection, the Central Device (Client) performs a service discovery on the Peripheral device (server) to build up an attribute table. This attribute table will provide handles (internal addresses of the characteristics) which can be used by the Client to access the data located in the Server. The service discovery is typically an automated process that can be started with a single command. In BTool however, the automated service discovery is not implemented (although it’s still possible to perform it manually). To simplify the evaluation of the KeyFobDemo, the attribute table will be known so it is possible to use handles directly to read out data.

You will find the KeyFobDemo complete attribute table in the Appendix section, and can be used as a reference. Services are shown in yellow, characteristics are shown in blue, and characteristic values / descriptors are shown in grey. When working with the KeyFobDemo application, it might be useful to print out the table as a reference.

4.3.1 Reading a Characteristic Value by UUID

A characteristic value is essentially where the data is stored, which could be for example temperature data or battery level. It’s the stored data in a server, that a client wants to access. A characteristic is a discrete value that has the following three properties associated with it:

1. A handle (address)
2. A type (UUID)
3. A set of permissions

The simplest way to read its value is to use the “Read Characteristic by UUID” sub-procedure. To do this, you will first need to click the “Read / Write” tab in BTool. Select the option “Read Using Characteristic UUID” under the “Sub-Procedure” option in the “Characteristic Read” section at the top of the screen. Enter the UUID (note that the LSB is entered first, and the MSB is entered last) in the “Characteristic UUID” box, and click the “Read” button.

An attribute protocol *Read by Type Request* packet gets sent over the air from the dongle to the keyfob, and an attribute protocol *Read by Type Response* packet gets sent back from the keyfob to the dongle. The characteristic value is displayed in the “Value” box, and “Success” is displayed in the “Status” box. In addition, the message window will display information on the *Read by Type Response* packet that was received by the dongle. The message includes not only the characteristic’s data value, but also the handle of the characteristic value.

4.3.2 Reading a Characteristic Value by Handle

It is also possible to read the characteristic value if the handle is known. This is done by selecting the option “Read Characteristic Value / Descriptor” under the “Sub-Procedure” option in the “Characteristic Read” section. The handle is entered to the “Characteristic Value Handle” box and clicking the “Read” button which execute the read.

An attribute protocol *Read Request* packet gets sent over the air from the dongle to the keyfob, and an attribute protocol *Read Response* packet gets sent back from the keyfob to the dongle. The new value is displayed in the "Value" box, and "Success" is displayed in the "Status" box. This value should match the value that was written in the previous step.

4.3.3 Writing a Characteristic Value

If a characteristic has write permission, it is possible for a client to write a value to the server. This is done in the "Characteristic Write" section by entering the handle into the "Characteristic Value Handle" box and value in the "Value" section (the format can be set to either "Decimal" or "Hex"). The write operation is performed when the "Write Value" button is clicked.

An attribute protocol *Write Request* packet gets sent over the air from the dongle to the keyfob, and an attribute protocol *Write Response* packet gets sent back from the keyfob to the dongle. The status box will display "Success", indicating that the write was successful.

4.3.4 Discovering a Characteristic by UUID

By discovering a characteristic by UUID, not only the handle of the UUID will be obtained, but also the properties of the characteristic (handle and permissions). The operation is performed by selecting the option "Discover Characteristic by UUID" under the "Sub-Procedure" option in the "Characteristic Read" section at the top of the screen. The discovery is performed when the UUID is entered in the "Characteristic UUID" box, and the "Read" button is clicked.

A series of attribute protocol *Read by Type Request* packets get sent over the air from the dongle to the keyfob, and for each request an attribute protocol *Read by Type Response* packet gets sent back from the keyfob to the dongle. Essentially, the dongle is reading every attribute on the keyfob with a UUID of 0x2803 (this is the UUID for a characteristic declaration as defined in [6]), and checking the "Characteristic Value UUID" portion of each declaration to see if it matches the UUID type you've entered. The procedure is complete once every characteristic declaration has been read.

As per the *Bluetooth* specification, the first byte presents the properties of the characteristic. The second and third bytes present the handle of the characteristic value. The fourth and fifth bytes present the UUID of the characteristic.

4.3.5 Reading Multiple Characteristic Values

It is also possible to read multiple characteristic values with one request, as long as the handle of each value is known. To read several values from different characteristics, select the option "Read Multiple Characteristic Values" under the "Sub-Procedure" option in the "Characteristic Read" section at the top of the screen. Enter the handles with semicolon separation (for example "0x0022;0x0025") in the "Characteristic Value Handle" box, and click the "Read" button.

An attribute protocol *Read Multiple Request* packet gets sent over the air from the dongle to the keyfob, and an attribute protocol *Read Multiple Response* packet gets sent back from the keyfob to the dongle. The values of the two characteristics are displayed in the "Value" box, and "Success" is displayed in the "Status" box.

One important note about reading multiple characteristic values in a single request is that the response will not parse the separate values. This means that the size of each value being read must be fixed, and must be known by the client. In the example here, this is not an issue since there are only two bytes in the response; however care must be taken when using this command.

4.3.6 Enable Notifications

In BLE, it is possible for a GATT server device to "push" characteristic value data out to a client device, without being prompted with a read request. This process is called a "characteristic value notification". Notifications are useful in that they allow a device in a BLE connection to send out as much or as little data as required at any point in time. In addition, since no request from the client is required, the overhead is reduced and the data is transmitted more efficiently.

In order to enable notifications, the client device must write a value of 0x0001 to the client characteristic configuration descriptor for the particular characteristic. The handle for the client characteristic configuration descriptor immediately follows the characteristic value's handle. Therefore, a value of 0x0001 must be written to the "handle + 1".

4.4 Using the Proximity Profile

The Proximity profile is defined for a usage where an alert may be triggered on certain connection events, for example when the connection is, or about to be, dropped.

handle (hex)	handle (dec)	Type (hex)	Type (#DEFINE)	Value (default)	GATT Server Permissions	Notes
0x23	35	0x2800	GATT_PRIMARY_SERVICE_UUID	0x1803 (LINK_LOSS_SERVICE_UUID)	GATT_PERMIT_READ	Start of Link Loss Service
0x24	36	0x2803	GATT_CHARACTER_UUID	0A (read/write permissions) 25 00 (handle 0x0025) 06 2A (UUID 0x2A06)	GATT_PERMIT_READ	Alert Level characteristic declaration
0x25	37	0x2A06	PROXIMITY_ALERT_LEVEL_UUID	0 (PP_ALERT_LEVEL_NO)	GATT_PERMIT_READ | GATT_PERMIT_WRITE	Link Loss Alert Level characteristic value (see defined values in proxperiph.h)
0x26	38	0x2800	GATT_PRIMARY_SERVICE_UUID	0x1802 (IMMEDIATE_ALERT_SERVICE_UUID)	GATT_PERMIT_READ	Start of Immediate Alert Service
0x27	39	0x2803	GATT_CHARACTER_UUID	04 (write permission) 28 00 (handle 0x0028) 06 2A (UUID 0x2A06)	GATT_PERMIT_READ	Alert Level characteristic declaration
0x28	40	0x2A06	PROXIMITY_ALERT_LEVEL_UUID	0 (PP_ALERT_LEVEL_NO)	GATT_PERMIT_WRITE	Path Loss Alert Level characteristic value (see defined values in proxperiph.h)
0x29	41	0x2800	GATT_PRIMARY_SERVICE_UUID	0x1804(TX_PWR_LEVEL_SERVICE_UUID)	GATT_PERMIT_READ	Start of Tx Power Level Service
0x2A	42	0x2803	GATT_CHARACTER_UUID	12 (read permission) 2B 00 (handle 0x002B) 07 2A (UUID 0x2A07)	GATT_PERMIT_READ	Tx Power Level characteristic declaration
0x2B	43	0x2A07	PROXIMITY_TX_PWR_LEVEL_UUID	0	GATT_PERMIT_READ	Tx Power Level characteristic value (dBm)
0x2C	44	0x2902	GATT_CLIENT_CHAR_CFG_UUID	0x0000	GATT_PERMIT_READ | GATT_PERMIT_WRITE	Tx Power Level characteristic configuration

Figure 19 Proximity Profile, Attribute Table

The Proximity profile includes three services:

- Link Loss (Mandatory Link Loss Service (LLS) Specification v1.0)
- Immidiate Alert (Optional Immediate Alert Service (IAS) Specification v1.0)
- Tx Power (Optional Tx Power Service (TPS) Specification v1.0)

4.4.1 Activate Link Loss Service

The link loss service allows the proximity reporter to begin an alert in the event the connection drops. The link loss alert is set by writing a value to <PROXIMITY_ALERT_LEVEL_UUID> in the link loss service. The default alert value setting is “00”, which indicates “no alert.” To turn on the alert, write a 1-byte value of “01” (low alert) or “02” (high alert).

This is done in BTool by using the “Characteristic Write” operation as shown in Figure 20. The handle is obtained from the Proximity Profile attribute table in Figure 19. Note that this handle is unique for this solution and will probably not be the same on other solutions using Proximity Profile. Usually, the handle is obtained by performing a “Discover characteristic UUID” as described in Section 4.3.4.

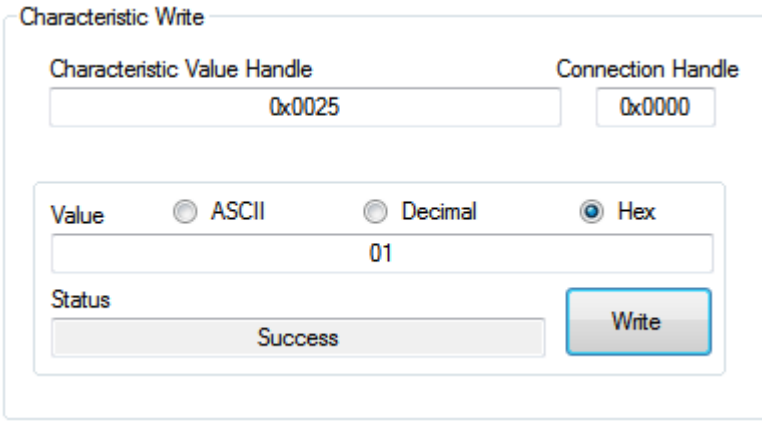

Figure 20 Proximity Profile, Link Loss Activation

An attribute protocol *Write Request* packet gets sent over the air from the dongle to the keyfob, and an attribute protocol *Write Response* packet gets sent back from the keyfob to the dongle. The status box will display “Success”, indicating that the write was successful.

By default, the link does not timeout until 20 seconds have gone by without receiving a packet. This “Supervision Timeout” value can be changed in the “Connection Settings” group in the “Discover/Connect” tab; however the timeout value must be set before the connection is established. After completing the write, move the keyfob device far enough away from the USB Dongle until the link drops. Alternatively, you can disconnect the USB Dongle from the PC, effectively dropping the connection. Once the timeout on the keyfob expires, the alarm will be triggered. If a low alert was set, the keyfob will make a low pitched beep. If a high alert was set, the keyfob will make a high pitched beep and the LED will blink. In either case, the keyfob will beep ten times and then stop. Alternatively to stop the beeping, either a new connection can be formed with the keyfob, or the left button can be pressed.

4.4.2 Activate Immediate Alert

The Immediate Alert Service allows the proximity reporter to immediately begin an alert. The immediate alert is set by writing a value to <PROXIMITY_ALERT_LEVEL_UUID> in the immediate alert service. The default alert value setting is “00”, which indicates “no alert.” To turn on the alert, write a 1-byte value of “01” (low alert) or “02” (high alert).

This is done in BTool by using the “Characteristic Write” operation as shown in Figure 21. The handle is obtained from the Proximity Profile attribute table in Figure 19. Note that this handle is unique for this solution and will probably not be the same on other solutions using Proximity Profile. Usually, the handle is obtained by performing a “Discover characteristic UUID” as described in Section 4.3.4.

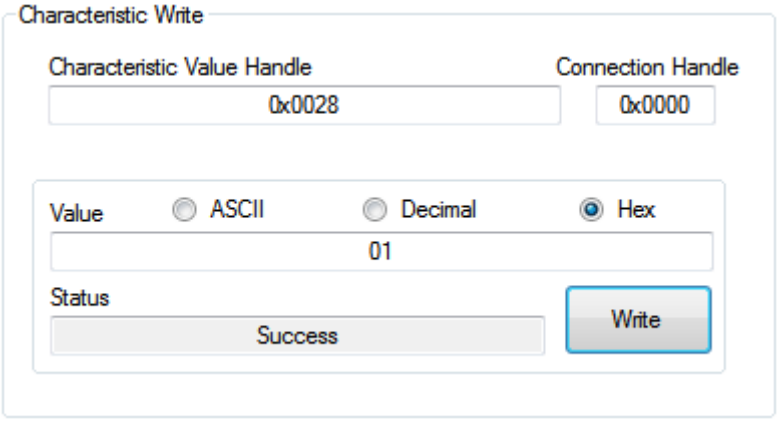

Figure 21 Proximity Profile, Immediate Alert

If a low alert was set, the keyfob will make a low pitched beep for 10 seconds. If a high alert was set, the keyfob will make a high pitched beep for 10 seconds and the LED will blink. In either case, the keyfob will beep ten times and then stop. Alternatively to stop the beeping, the left button can be pressed.

4.4.3 Read TX Power

The TX Power Service allows the proximity reporter to report its output power. The TX Power level is obtained by reading the characteristic value of <PROXIMITY_TX_PWR_LEVEL_UUID> in the TX Power Service. This can be done in BTool by two ways, “Read Characteristic Value” or “Read Using Characteristic UUID”. In Figure 22, the “Read Characteristic Value” is been used, because the handle is known.

An attribute protocol *Read by Type Request* packet gets sent over the air from the dongle to the keyfob, and an attribute protocol *Read by Type Response* packet gets sent back from the keyfob to the dongle. The value is displayed in the “Value” box, and “Success” is displayed in the “Status” box. In addition, the message window will display information on the *Read by Type Response* packet that was received by the dongle. The message includes not only the characteristic’s data value, but also the handle of the characteristic value (0x002B in this case).

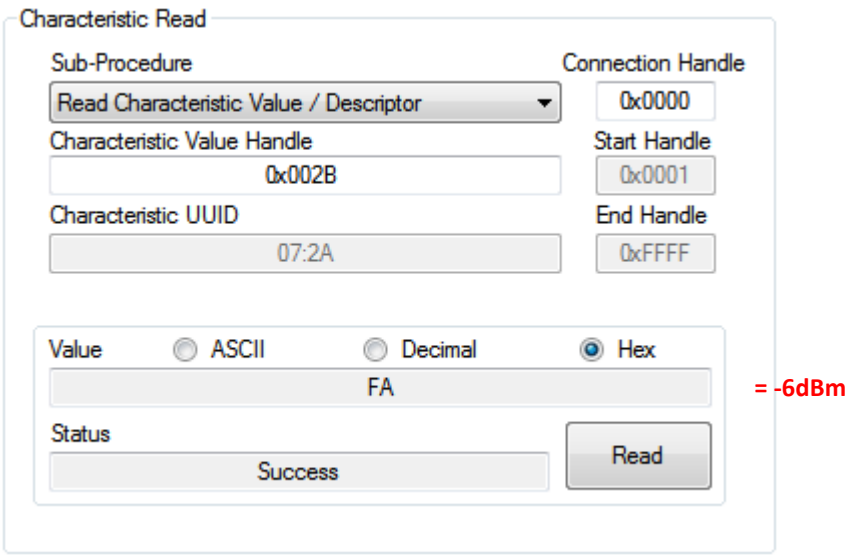

Figure 22 Proximity Profile, TX Power

The output power can be toggled (0dBm and -6dBm) by pressing the left button.

4.5 Using the Battery Service

The Battery Service exposes the Battery Level of the coin cell battery on the keyfob.

handle (hex)	handle (dec)	Type (hex)	Type (#DEFINE)	Value (default)	GATT Server Permissions	Notes
0x2D	45	0x2800	GATT_PRIMARY_SERVICE_UUID	0x180F(BATTERY_SERVICE_UUID)	GATT_PERMIT_READ	Start of Battery Service
0x2E	46	0x2803	GATT_CHARACTER_UUID	12 (read permission) 2F 00 (handle 0x002F) 19 2A (UUID 0x2A19)	GATT_PERMIT_READ	Battery level characteristic declaration
0x2F	47	0x2A19	BATTERY_LEVEL_UUID	100	GATT_PERMIT_READ	Battery level characteristic value (percentage)
0x30	48	0x2902	GATT_CLIENT_CHAR_CFG_UUID	00:00 (2 bytes)	GATT_PERMIT_READ | GATT_PERMIT_WRITE	Battery level characteristic configuration
0x31	49	0x2908	GATT_REPORT_REF_UUID	02:01 (2 bytes)	GATT_PERMIT_READ	HID Report Reference characteristic descriptor, Batter Level Input

Figure 23 Battery Service, Attribute Table

4.5.1 Read the Battery Level

They KeyFob used an ADC to read remaining battery level. The battery profile allows for a peer device (such as BTool) to read the percentage of battery remaining on the keyfob by reading the value of <BATTERY_LEVEL_UUID> in the Battery Service. This can be done in BTool by two ways, “Read Characteristic Value” or “Read Using Characteristic UUID”. In Figure 24, the “Read Using Characteristic UUID” is been used. Note that the value format is set to “Decimal” in this example, so roughly 61% battery left.

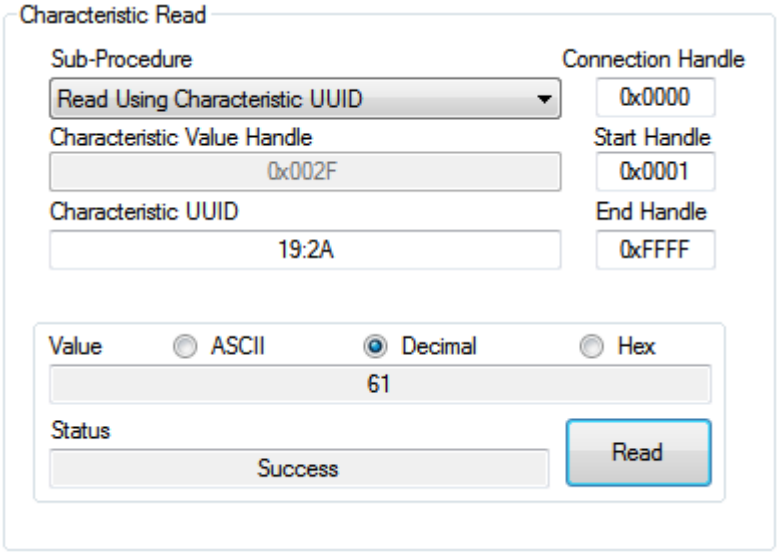

Figure 24 Battery Service, Read Battery Level

4.5.2 Activate Battery Level Notification

In order to enable notifications, the client device (BTool in this case) must write a value of 0x0001 to the client characteristic configuration descriptor for the particular characteristic. The handle for the client characteristic configuration descriptor immediately follows the characteristic value’s handle. Therefore, a value of 0x0001 must be written to handle 0x0030. Enter “0x0030” into the “Characteristic Value Handle” box in the “Characteristic Write” section, and enter “01:00” in the “Value” section (note that the LSB is entered first, and the MSB is entered last) as shown in Figure 25. Click the “Write Value” button. The status box will display “Success”, indicating that the write was successful.

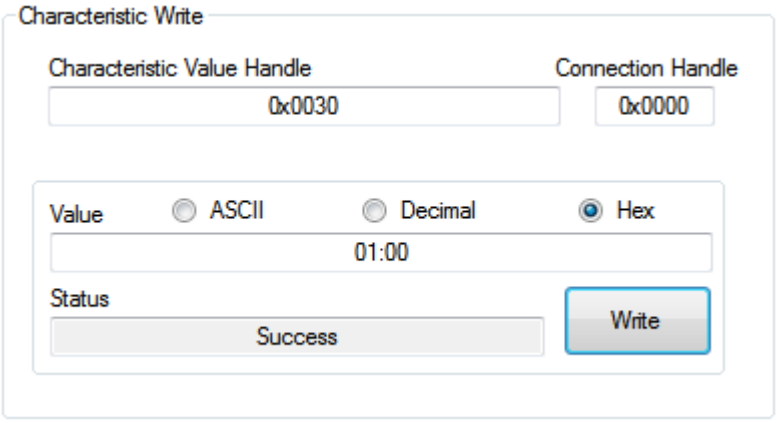

Figure 25 Battery Service, Enable Notification

Notification of the battery level-state is handled inside the battery service and if the battery level has dropped since the previous measurement, a notification is sent as seen in Figure 26.

```
------------------------------------------------------------------------------
[73] : <Rx> - 03:06:30.607
-Type              : 0x04 (Event)
-EventCode         : 0xFF (HCI_LE_ExtEvent)
-Data Length       : 0x07 (7) bytes(s)
 Event             : 0x050B (ATT_ReadRsp)
 Status            : 0x00 (Success)
 ConnHandle        : 0x0000 (0)
 PduLen            : 0x01 (1)
 Value             : 35                                        = 53 %
Dump(Rx):
04 FF 07 0B 05 00 00 00 01 35

------------------------------------------------------------------------------
[74] : <Rx> - 03:11:18.436
-Type              : 0x04 (Event)
-EventCode         : 0xFF (HCI_LE_ExtEvent)
-Data Length       : 0x09 (9) bytes(s)
 Event             : 0x051B (ATT_HandleValueNotification)
 Status            : 0x00 (Success)
 ConnHandle        : 0x0000 (0)
 PduLen            : 0x03 (3)
 Handle            : 0x002F (47)
 Value             : 34                                        = 52 %
Dump(Rx):
04 FF 09 1B 05 00 00 00 03 2F 00 34

------------------------------------------------------------------------------
```

Figure 26 BTool Log, Battery Service Notification

4.6 Using the Accelerometer Service

The keyfob uses SPI to interface to a 3 axis accelerometer on the keyfob. Note that accelerometer sensor can be of different vendors depending on the board revision. For example:

- CC2540 Keyfob v1.1 uses CMA3000d
- CC2541 Keyfob v1.4 uses BMA250

Note that the types (UUIDs) of the five characteristic values (0xFFA1, 0xFFA2, 0xFFA3, 0xFFA4, and 0xFFA5), as well as the primary service UUID value (0xFFA0), do not conform to any specifications in the *Bluetooth* SIG. They are simply used as a demonstration.

handle (hex)	handle (dec)	Type (hex)	Type (#DEFINE)	Value (default)	GATT Server Permissions	Notes
0x32	50	0x2800	GATT_SERVICE_UUID	0xFFA0 (ACCEL_SERVICE_UUID)	GATT_PERMIT_READ	Start of Accelerometer Service
0x33	51	0x2803	GATT_CHARACTER_UUID	0A (read/write permissions) 34 00 (handle 0x0034) A1 FF (UUID 0xFFA1)	GATT_PERMIT_READ	Accelerometer enable characteristic declaration
0x34	52	0xFFA1	ACCEL_ENABLER_UUID	FALSE	GATT_PERMIT_READ | GATT_PERMIT_WRITE	Accelerometer enable characteristic value (TRUE or FALSE)
0x35	53	0x2901	GATT_CHAR_USER_DESC_UUID	"Accel Enable" (14 Bytes)	GATT_PERMIT_READ	Accelerometer enable characteristic user description
0x36	54	0x2803	GATT_CHARACTER_UUID	02 (read permission) 37 00 (handle 0x0037) A2 FF (UUID 0xFFA2)	GATT_PERMIT_READ	Accelerometer range characteristic declaration
0x37	55	0xFFA2	ACCEL_RANGE_UUID	20 (ACCEL_RANGE_2G)	GATT_PERMIT_READ	Accelerometer range characteristic value (can be 2G or 8G)
0x38	56	0x2901	GATT_CHAR_USER_DESC_UUID	"Accel Range" (13 Bytes)	GATT_PERMIT_READ	Accelerometer range characteristic user description
0x39	57	0x2803	GATT_CHARACTER_UUID	10 (notify permission) 3A 00 (handle 0x003A) A3 FF (UUID 0xFFA3)	GATT_PERMIT_READ	Accelerometer X-coordinate characteristic declaration
0x3A	58	0xFFA3	ACCEL_X_UUID	00:00 (2 bytes)	(none)	Accelerometer X-coordinate characteristic value
0x3B	59	0x2902	GATT_CLIENT_CHAR_CFG_UUID	00:00 (2 bytes)	GATT_PERMIT_READ | GATT_PERMIT_WRITE	Accelerometer X-coordinate characteristic configuration
0x3C	60	0x2901	GATT_CHAR_USER_DESC_UUID	"Accel X-Coordinate" (20 Bytes)	GATT_PERMIT_READ	Accelerometer X-coordinate characteristic user description
0x3D	61	0x2803	GATT_CHARACTER_UUID	10 (notify permission) 3E 00 (handle 0x003E) A4 FF (UUID 0xFFA4)	GATT_PERMIT_READ	Accelerometer Y-coordinate characteristic declaration
0x3E	62	0xFFA4	ACCEL_Y_UUID	00:00 (2 bytes)	(none)	Accelerometer Y-coordinate characteristic value
0x3F	63	0x2902	GATT_CLIENT_CHAR_CFG_UUID	00:00 (2 bytes)	GATT_PERMIT_READ | GATT_PERMIT_WRITE	Accelerometer Y-coordinate characteristic configuration
0x40	64	0x2901	GATT_CHAR_USER_DESC_UUID	"Accel Y-Coordinate" (20 Bytes)	GATT_PERMIT_READ	Accelerometer Y-coordinate characteristic user description
0x41	65	0x2803	GATT_CHARACTER_UUID	10 (notify permission) 42 00 (handle 0x0042) A5 FF (UUID 0xFFA5)	GATT_PERMIT_READ	Accelerometer Z-coordinate characteristic declaration
0x42	66	0xFFA5	ACCEL_Z_UUID	00:00 (2 bytes)	(none)	Accelerometer Z-coordinate characteristic value
0x43	67	0x2902	GATT_CLIENT_CHAR_CFG_UUID	00:00 (2 bytes)	GATT_PERMIT_READ | GATT_PERMIT_WRITE	Accelerometer Z-coordinate characteristic configuration
0x44	68	0x2901	GATT_CHAR_USER_DESC_UUID	"Accel Z-Coordinate" (20 Bytes)	GATT_PERMIT_READ	Accelerometer Z-coordinate characteristic user description

Figure 27 Accelerometer Service, Attribute Table

4.6.1 Enable Accelerometer

The first characteristic of the Accelerometer service has both read and write permissions, and has a UUID of 0xFFA1. This characteristic is “Accelerometer Enable” and is used to start and stop the running of the accelerometer sensor. The accelerometer is enabled by writing a value of “01” to the characteristic.

This is done in BTool by using the “Characteristic Write” operation as shown in Figure 28. The handle is obtained from the Accelerometer Service attribute table in Figure 27.

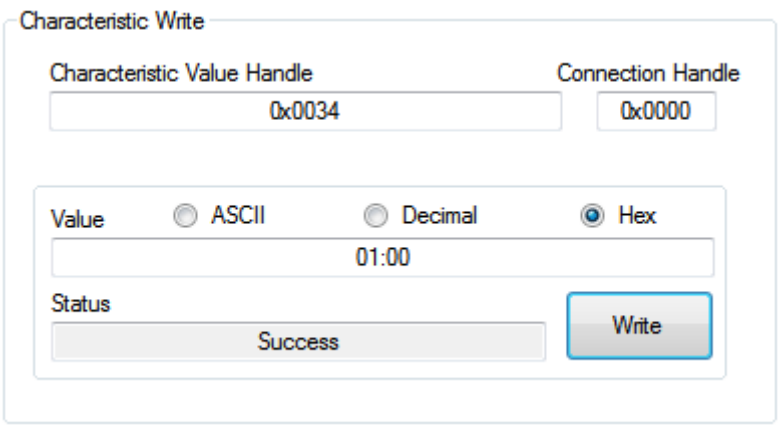

Figure 28 Accelerometer Service, Enable Accelerometer

4.6.2 Enable Accelerometer Notifications

Once the accelerometer is enabled, each axis can be configured to send notifications by writing “01 00” to the characteristic configuration for each axis < GATT_CLIENT_CHAR_CFG_UUID>. In addition, the values can be read directly by reading <ACCEL_X_UUID>, <ACCEL_Y_UUID> and <ACCEL_Z_UUID>.

This is done in BTool by using the “Characteristic Write” operation as shown in Figure 29. The handle is obtained from the Accelerometer Service attribute table in Figure 27.

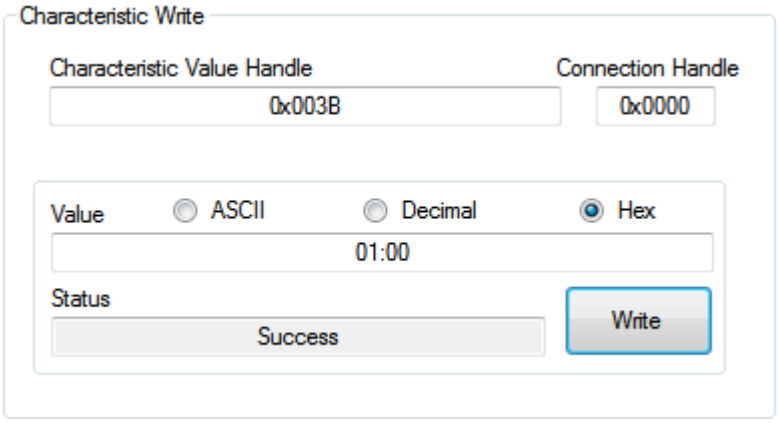

Figure 29 Accelerometer Service, Enable Notification on X-Axis

Moving the keyfob will result in notifications being sent and received by BTool, as shown in Figure 30.

```
[160] : <Rx> - 03:33:36.833
-Type              : 0x04 (Event)
-EventCode         : 0xFF (HCI_LE_ExtEvent)
-Data Length       : 0x09 (9) bytes(s)
 Event             : 0x051B (ATT_HandleValueNotification)
 Status            : 0x00 (Success)
 ConnHandle        : 0x0000 (0)
 PduLen            : 0x03 (3)
 Handle            : 0x003A (58)
 Value             : E6
Dump(Rx):
04 FF 09 1B 05 00 00 00 03 3A 00 E6
```

Figure 30 BTool Log, X-Axis Notification

It is possible to enable notifications for the other two characteristics to obtain acceleration information from x-, y- and z-axis.

4.7 Using the Simple Keys GATT Profile

The simple keys profile on the keyfob allows the device to send notifications of key presses and key releases to a central device.

handle (hex)	handle (dec)	Type (hex)	Type (#DEFINE)	Value (default)	GATT Server Permissions	Notes
0x45	69	0x2800	GATT_SERVICE_UUID	0xFFE0 (SK_KEYPRESSED_UUID)	GATT_PERMIT_READ	Start of Simple Keys Service
0x46	70	0x2803	GATT_CHARACTER_UUID	10 (notify permission) 47 00 (handle 0x0047) E1 FF (UUID 0xFFE1)	GATT_PERMIT_READ	Key Press State characteristic declaration
0x47	71	0xFFE1	SK_KEYPRESSED_UUID	0 (1 byte)	(none)	Key Press State characteristic value
0x48	72	0x2902	GATT_CLIENT_CHAR_CFG_UUID	00:00 (2 bytes)	GATT_PERMIT_READ | GATT_PERMIT_WRITE	Key Press State characteristic configuration
0x49	73	0x2901	GATT_CHAR_USER_DESC_UUID	"Key Press State" (16 bytes)	GATT_PERMIT_READ	Key Press State characteristic user description

Figure 31 Simple Keys Service, Attribute Table

It is important to note that the simple keys profile included with the BLE development kit does not conform to any standard profile specification available from the *Bluetooth* SIG. At the time of the release of the software, no official GATT service profile specifications have been approved by the *Bluetooth* SIG. Therefore the profile, including the GATT characteristic definition, the UUID values, and the functional behavior, was developed by Texas Instruments for use with the CC254*X*DK-MINI development kit.

As the *Bluetooth* SIG begins to approve specifications for different service profiles, Texas Instruments plans to release updates to the BLE software development kit with source code conforming to the specifications.

4.7.1 Enable Simple Keys notifications

The UUID of the simple keys data characteristic value is 0xFFE1. Using the "Discover Characteristic by UUID" command, it can be determined that the handle of the simple keys data is 0x0047 (which also can directly be read from the attribute table in Figure 31). The simple keys data is a "configurable" characteristic, in that the client device can configure the server to send notifications of the characteristic value. The handle immediately following the characteristic value is the client characteristic configuration descriptor.

The characteristic configuration of the simple keys data is the attribute at handle 0x0048. To turn on notifications, enter 0x0048 into the "Characteristic Value Handle" box in the "Characteristic Write" section, and enter "01:00" in the "Value" section. The format can be set to either "Hex" or "Decimal". Click the "Write" button to send the write request over the air. When the keyfob receives the request, it will turn on notifications of the simple keys data, and send a write response to indicate success.

With notifications enabled, an attribute protocol *Handle Value Notification* packet is sent from the keyfob to the dongle as you press or release either of the buttons on the keyfob. The notifications should show up in the log window. A value of "00" indicates that neither key is pressed. A value of "01" indicates that the left key is pressed. A value of "02" indicates that the right key is pressed. A value of "03" indicates that both keys are pressed.

```
---------------------------------------------------------------------------
[42] : <Rx> - 11:02:26.367
-Type           : 0x04 (Event)
-EventCode      : 0xFF (HCI_LE_ExtEvent)
-Data Length    : 0x09 (9) bytes(s)
 Event          : 0x051B (ATT_HandleValueNotification)
 Status         : 0x00 (Success)
 ConnHandle     : 0x0000 (0)
 PduLen         : 0x03 (3)
 Handle         : 0x0047 (71)
 Value          : 01
Dump(Rx):
04 FF 09 1B 05 00 00 00 03 47 00 01

---------------------------------------------------------------------------
```

Figure 32 BTool Log, Key Press Notification

4.8 Using BLE Security

BTool also includes the ability to make use of security features in BLE, including encryption, authentication, and bonding.

4.8.1 Encrypting the Connection

To encrypt the link, the pairing process must be initiated. Click on the "Pairing / Bonding" tab in BTool. In the "Initiate Pairing" section at the top of the screen, check the boxes labeled "Bonding Enabled" and "Authentication (MITM) Enabled", and click the button "Send Pairing Request" as shown in Figure 33. This will send the pairing request to the peripheral device.

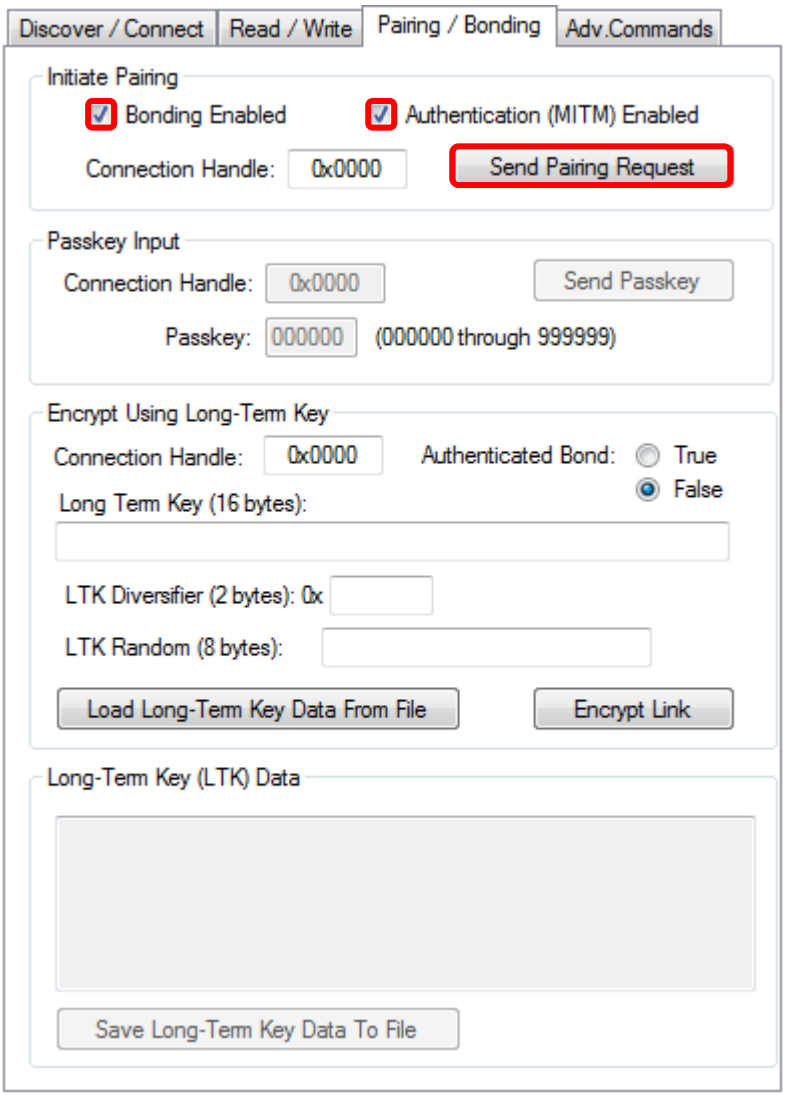

Figure 33 BTool, Sending Pairing Request

The peripheral will send a pairing response in return, which will require a six-digit pass code to be entered by the user in order to complete the process. Typically, this pass code is intended to be used by a peripheral device containing a display. By displaying the passkey on the peripheral device and requiring the user to enter it in on the central device's user interface, the link is authenticated, in that it has been verified that the connection has not been hijacked using a man-in-the-middle (MITM) attack.

In the case of the KeyFobDemo software, a fixed pass code "000000" is used, since the keyfob does not have a display (this value can be modified in the source code). In the box labeled "Passkey" in the "Passkey Input" section, enter the value "000000" and click the "Send Passkey" button as shown in Figure

34. Note that if you do not send the passkey within 30 seconds after receiving the pairing response message, the pairing process will fail, and you will need to re-send the pairing request.

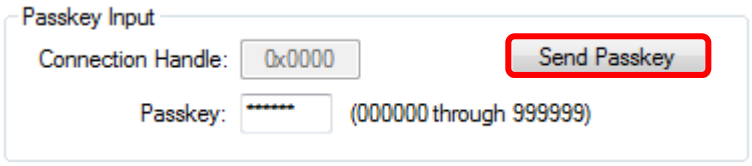

Figure 34 BTool, Sending Passkey

When pairing is successfully completed, you will see a "GAP_AuthenticationComplete" event in the log window, with a "Success" status. The BLE connection is now encrypted as shown in Figure 35.

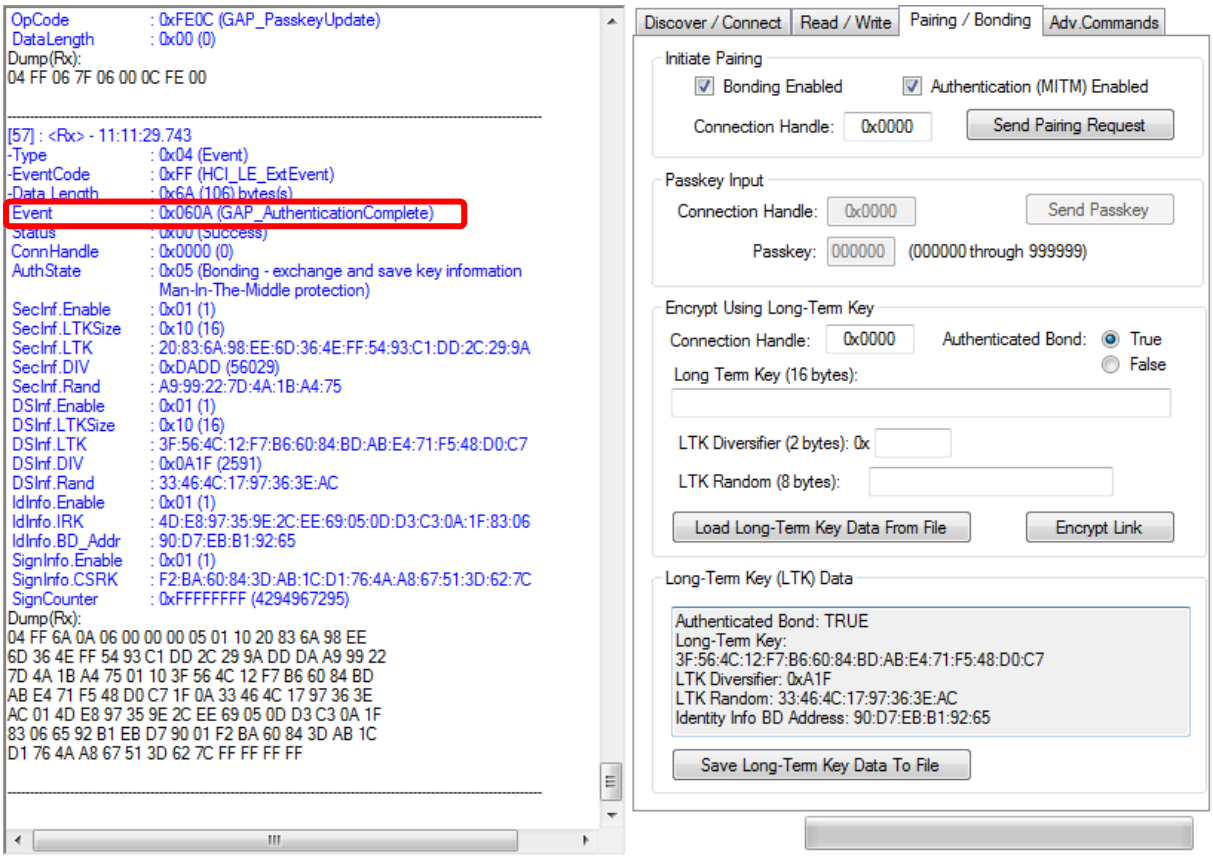

Figure 35 BTool, Encrypted Connection

4.8.2 Using Bonding and Long-Term Keys

Bonding is a feature in BLE that allows a device, after initial pairing with a peer, to remember specific information about that peer device. In particular, the long-term key data that is generated during the initial pairing process can be stored locally. If the connection is then terminated and the two devices later reconnect, this data can be used to quickly re-initiate encryption without needing to go through the full pairing process and/or use a passkey. In addition, if a client device had enabled notifications of any characteristics on the server device while the two devices were bonded, the server device will remember the setting and the client will not have to re-enable them.

After pairing has been completed with bonding enabled, the "Long-Term Key (LTK) Data" will be populated with some of the data from the "GAP_AuthenticationComplete" event that was generated during the encryption process. This data is required for re-initiating encryption upon reconnect. Click the "Save Long-Term Key Data to File" button to save this information to file. The data is saved as in a "comma separated value" (CSV) format as simple text, and can be store anywhere on disk. Be sure to note the location that the file is stored.

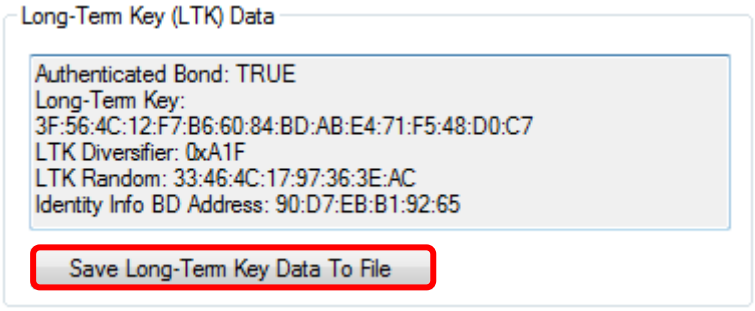

Figure 36 BTool, Save Long-Term Key

Within the keyfob, a similar process is going on, in that the KeyFobDemo software contains a bond manager that is storing the long-term key data that it had generated during encryption. Since the KeyFobDemo does not have a file system, it is simply storing the data in the nonvolatile memory of the CC2540/41. More information on the bond manager can be found in the Texas Instruments *Bluetooth*® Low Energy Software Developer's Guide [3].

With a bond now active, you can enable notifications of a characteristic value and have that setting remembered for later. Note that if notifications were enabled before going through the pairing process, then the setting will not be stored. Therefore, you will need to re-write the value "01:00" to a client characteristic configuration descriptor. For example, write "01:00" to handle 0x0048 to enable notifications of key presses, as was done in section 4.7.1. You should now be receiving notifications whenever the buttons are pressed or released. Because the devices are paired with bonding enabled, the bond manager in the KeyFobDemo software will store the client characteristic configuration descriptor data in nonvolatile memory.

To verify that bonding worked, you will need to disconnect and re-connect. Click on the "Discover / Connect" tab and click the "Terminate" button at the bottom of the screen to disconnect from the keyfob. The message window will show a "GAP_TerminateLink" event with "Success" status as shown in Figure 37. In addition, the connection information in the upper-left corner of the screen will disappear.

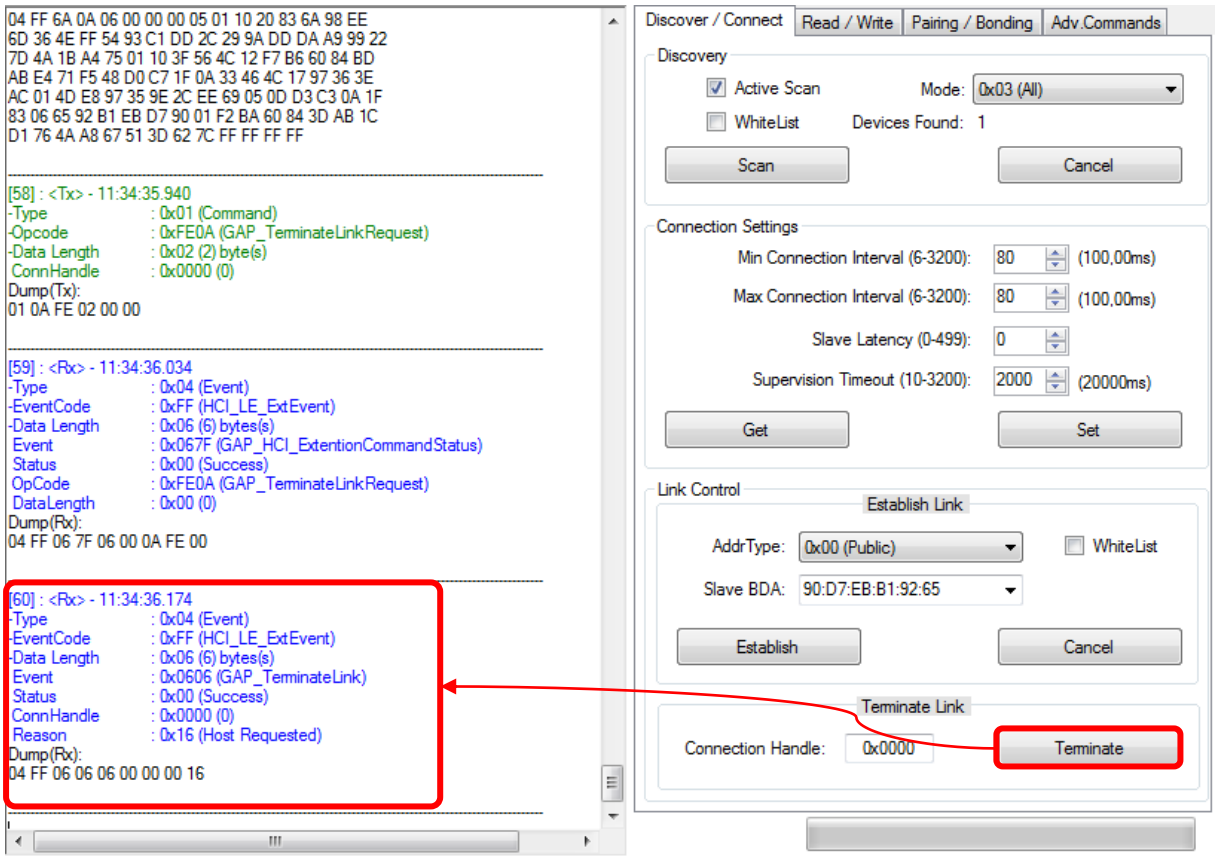

Figure 37 BTool, Terminate Link

At a later time, re-connect with the keyfob following the procedure in section 4.2. Once connected, you will notice that the simple keys notifications are no longer enabled. This is because the Simple Keys profile will always reset the value of the client characteristic configuration descriptor back to “00:00” if a connection is terminated or if the device resets.

To re-initiate encryption and re-enable notifications of key presses, return to the “Pairing / Bonding” tab. In the “Initiate Bond” section, click the “Load Long-Term Key Data From File” button, and select the file in which the data was previously stored. The data fields will get automatically populated from the data in the file. Click the “Initiate Bond” button to re-enable encryption as shown in Figure 38.

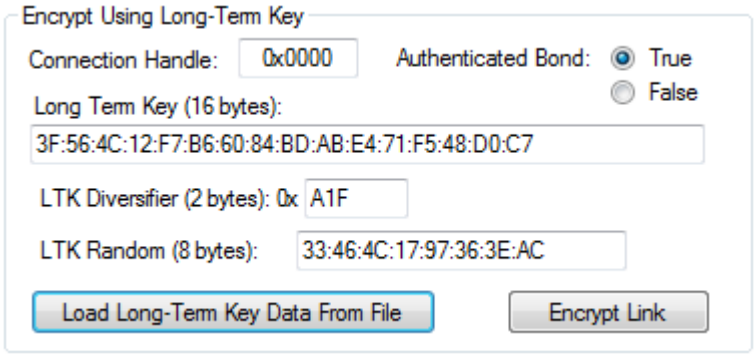

Figure 38 BTool, Re-Encrypt using Long-Term Keys

A “GAP_BondComplete” event with “Success” status will be displayed in the log window. This indicates that the link has been re-encrypted as shown in Figure 39. You will also now be able to receive notifications now when the buttons on the keyfob are pressed or released, as the client characteristic configuration descriptor value of the key press characteristic has been stored. Any changes to the client characteristic configuration descriptor value (i.e. turning off notifications) will be saved to nonvolatile memory and remembered for next time that encryption is initiated using the long-term key.

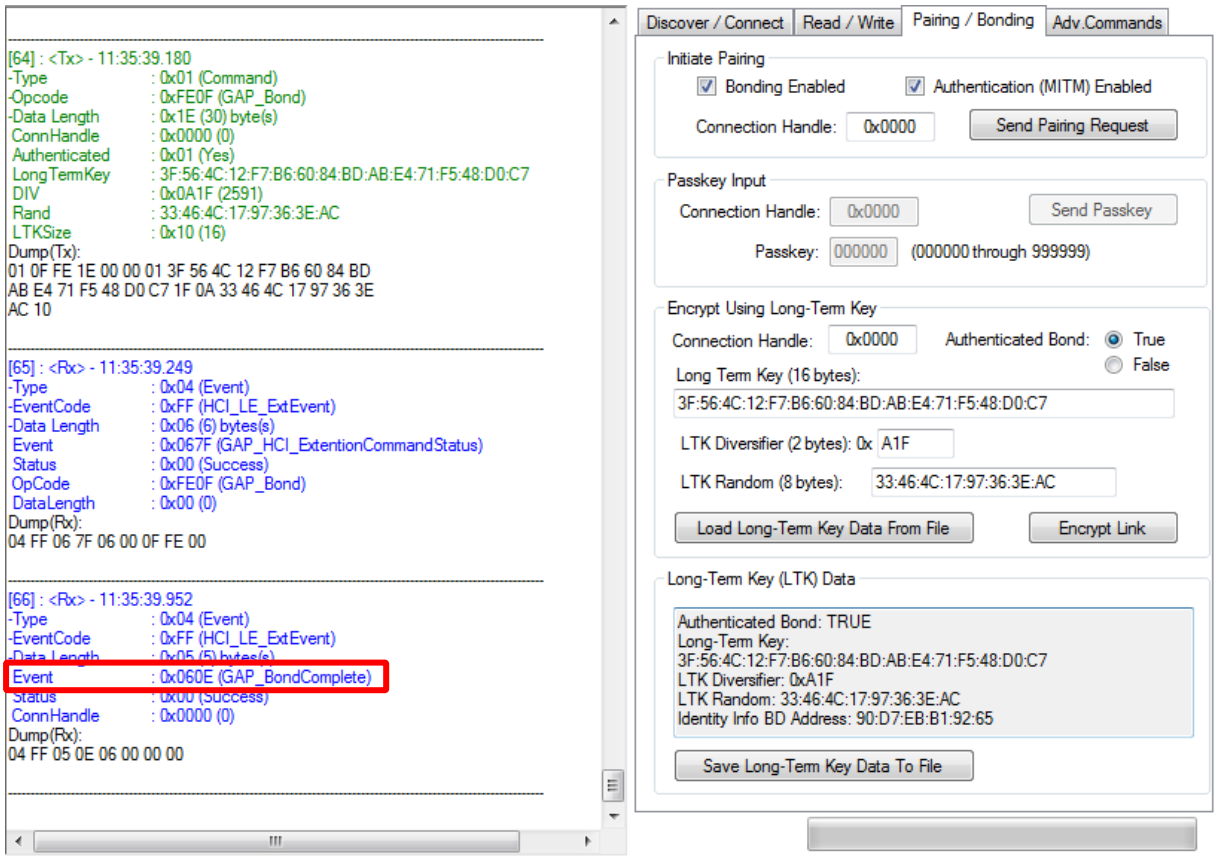

Figure 39 BTool, Bonding completed

4.9 Additional Sample Applications

In addition to the KeyfobDemo application, the BLE software development kit includes project and source code files for several additional applications and profiles, including:

- Blood Pressure Sensor- with simulated measurements
- Emulated Keyboard- press the two buttons on the keyfob to simulate keyboard presses
- Heart Rate Sensor- with simulated measurements
- Health Thermometer- with simulated measurements
- Glucose Sensor – with simulated measurements
- SimpleBLEPeripheral - with proprietary profile which implements all various types of permissions

More information on these projects can be found in the Texas Instruments BLE Sample Applications Guide [5].

5. Programming / Debugging the CC2540 or CC2541

The CC Debugger included with the CC254*X*DK-MINI kit allows for debugging using IAR Embedded Workbench for 8051, as well as for reading and writing hex files to the CC2540/41 flash memory using the SmartRF Flash Programmer software. SmartRF Flash Programmer also has the capability to change the IEEE address of the CC2540/41 device. The BLE software development kit includes hex files for both the USB Dongle as well as the keyfob. This section details the hardware setup when using the CC Debugger, as well as information on using SmartRF Flash Programmer. Information on using IAR Embedded Workbench for debugging can be found in the Texas Instruments *Bluetooth*® Low Energy Software Developer's Guide [3].

5.1 Hardware Setup for Keyfob

If the keyfob is viewed with the LED on top and the coin cell battery holder at the bottom, then the set of pins closer to the top are the ones that should be used for connecting to the debugger. Pin 1 is the pin on the lower right side as shown in Figure 40.

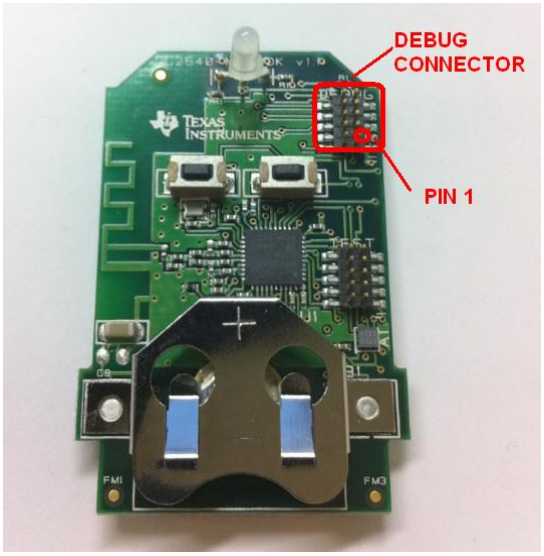

Figure 40 CC2540 Keyfob, Debug Connector

Connect the CC Debugger to the keyfob as shown below. Be sure that the ribbon cable is oriented properly, with the red stripe connected to pin 1 as shown in Figure 41.

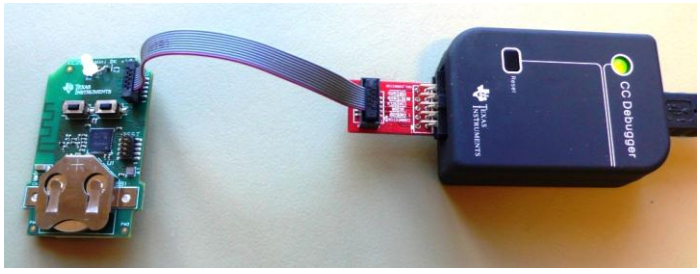

Figure 41 CC2540 Keyfob Connected to CC Debugger

Insert a coin cell battery in the keyfob to supply power to the target. **NB!** Note the orientation of the battery (+ up, - down). Next, connect the CC Debugger to the PC's USB port and then to the keyfob. Note that the CC debugger will by default not supply any power, but it will sense the voltage on the target (in this case the keyfob) for proper level shifting of the debug signals. The status indicator LED on the CC Debugger should turn on. If the LED is red, that means no CC2540/41 device was detected. If it is green, then a CC2540/41 device has been detected. If the keyfob is connected and the LED is red, try pressing the reset button on the CC Debugger. This resets the debugger and re-checks for a CC2540/41 device. If the LED still does not turn green, re-check that all cables are securely connected. Also verify that the CC Debugger has the latest firmware (see section 5.3).

Figure 42 CC Debugger Interface

Once the CC Debugger is set up with the status indicator LED showing green, you are ready to either read or write a hex file from the board, or to start debugging a project using IAR Embedded Workbench.

Power Savings Tip: Do not leave the CC Debugger connected to the keyfob for and extended period of time with the battery in the keyfob. This will cause a higher, constant current draw from the battery, and will significantly reduce the battery life.

If you intend to perform a lot of debugging and expect to leave the debugger connected to the keyfob for a long time, it is possible to supply power directly from the CC Debugger. In this case, the first thing you need to do is to remove the battery. This is important in order to avoid any charging current to the battery.

On the CC2540Keyfob, locate the pads for resistor R1, which are located immediately next to the debug header. Using a soldering iron, solder a small piece of wire across the two pads, shorting them together as shown in Figure 43.

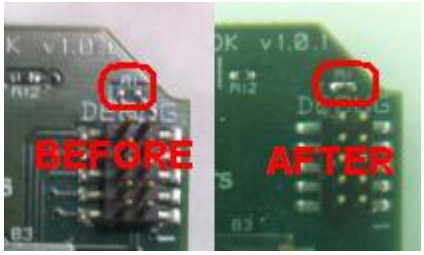

Figure 43 CC2540, Power Device Using CC Debugger

On the CC2541 Keyfob, short-circuit the two pins on the P1 connector, next to the LED, with the small jumper included in the kit.

WARNING! This kit includes a non-rechargeable lithium battery. To minimize risk of personal injury and/or property damage due to potential of explosion/rupture of battery due to charging the coin cell, always make sure battery is completely removed from the CC2541 Keyfob before trying to power it from the CC Debugger. As with any lithium battery, proper disposal should always be done and keep out of the reach of children at all times.

5.2 Hardware Setup for USB Dongle

The setup process for flashing the USB Dongle is very similar to the process when flashing the keyfob. First, plug the USB Dongle into a PC USB port (or a USB hub), as shown in Figure 44.

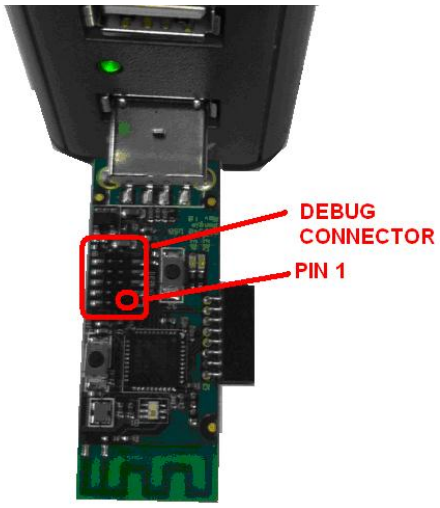

Figure 44 CC2540 USB Dongle

Connect the CC Debugger to the USB Dongle as shown below. Be sure that the ribbon cable is oriented properly, with the red stripe connected to pin 1 as shown in Figure 45.

Figure 45 CC2540 USB Dongle Connected to CC Debugger

Connect the CC Debugger to the PC USB port. The status indicator LED on the CC Debugger should turn on. If the LED is red, that means no CC2540 device was detected. If it is green, then a CC2540 device has been detected. If the USB Dongle is connected and the LED is red, try pressing the reset button on the CC Debugger. This resets the debugger and re-checks for a CC2540 device. If the LED still does not turn green, re-check that all cables are securely connected.

Figure 46 CC Debugger Interface

Once the CC Debugger status LED is showing green, as shown in Figure 46, you are ready to use IAR to debug or to read or write a hex file from/to the USB Dongle.

5.3 Using SmartRF Flash Programmer Software

Note: the instructions in the section apply to the latest version of SmartRF Flash Programmer (version 1.12.6), which is available at the following URL: http://www.ti.com/tool/flash-programmer

To start the application go into your programs by choosing Start > All Programs > Texas Instruments > SmartRF Flash Programmer > SmartRF Flash Programmer. The program start-up screen is shown in Figure 47.

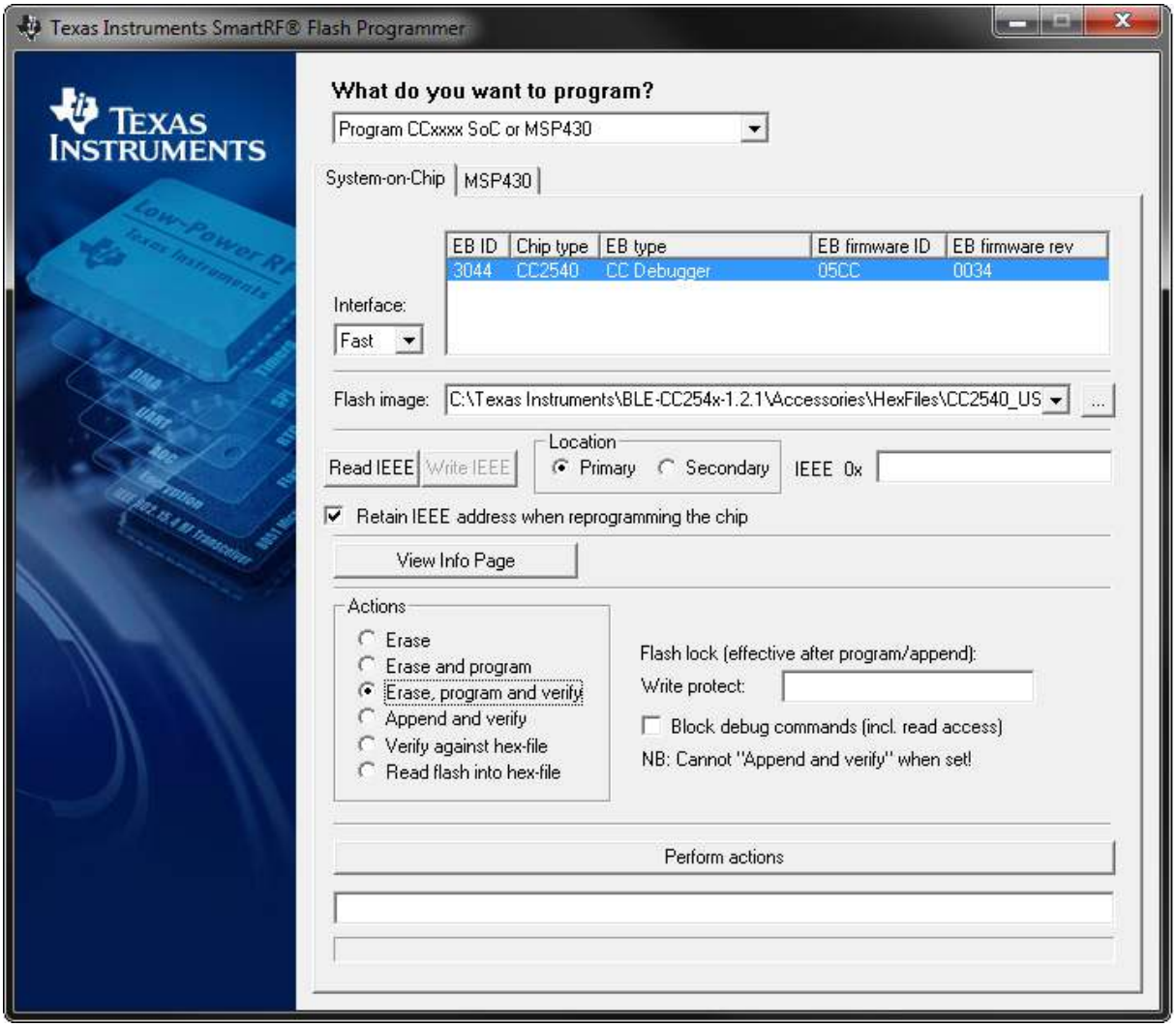

Figure 47 Flash Programmer

Note. If you get prompted to update the EB Firmware (CC Debugger), follow the presented instructions to update the CC Debugger.

5.3.1 Reading or Writing a Hex File to the CC2540/41

To read or write a hex file to the CC2540/41, select the "System-on-Chip" tab (default). The connected CC2540/41 should be detected and show up in the list of devices. Under "Flash image" select the desired hex file that you would like to write to the device. If you are reading from the CC2540/41, under "Flash image" enter the desired path and filename for the hex file. To write to the CC2540/41, under "Actions" select "Erase, program and verify". To read from the CC2540/41, under "Actions" select "Read flash into hex-file". To begin the read or write, click the button "Perform actions".

If the action completes successfully, you should see the progress bar at the bottom of the window fill up, and either one of the following two messages, depending on whether a write or a read was performed: "CC254*X* - ID*XXXX*: Erase, program and verify OK" or "CC254*X* - ID*XXXX*: Flash read OK".

5.3.2 Reading or Writing the CC2540/41 Device Address

Every CC2540/41 device comes pre-programmed with a unique 48-bit IEEE address. This is referred to as the device's "primary address", and cannot be changed. It is also possible to set a "secondary address" on a device, which will override the primary address upon power-up. Flash Programmer can be used to read the primary address, as well as to read or write the secondary address.

To read the primary address of a device connected to the CC Debugger, select "Primary" under the "Location" option, and click the "Read IEEE" button. The primary device address should appear in the box on the right as shown in Figure 48.

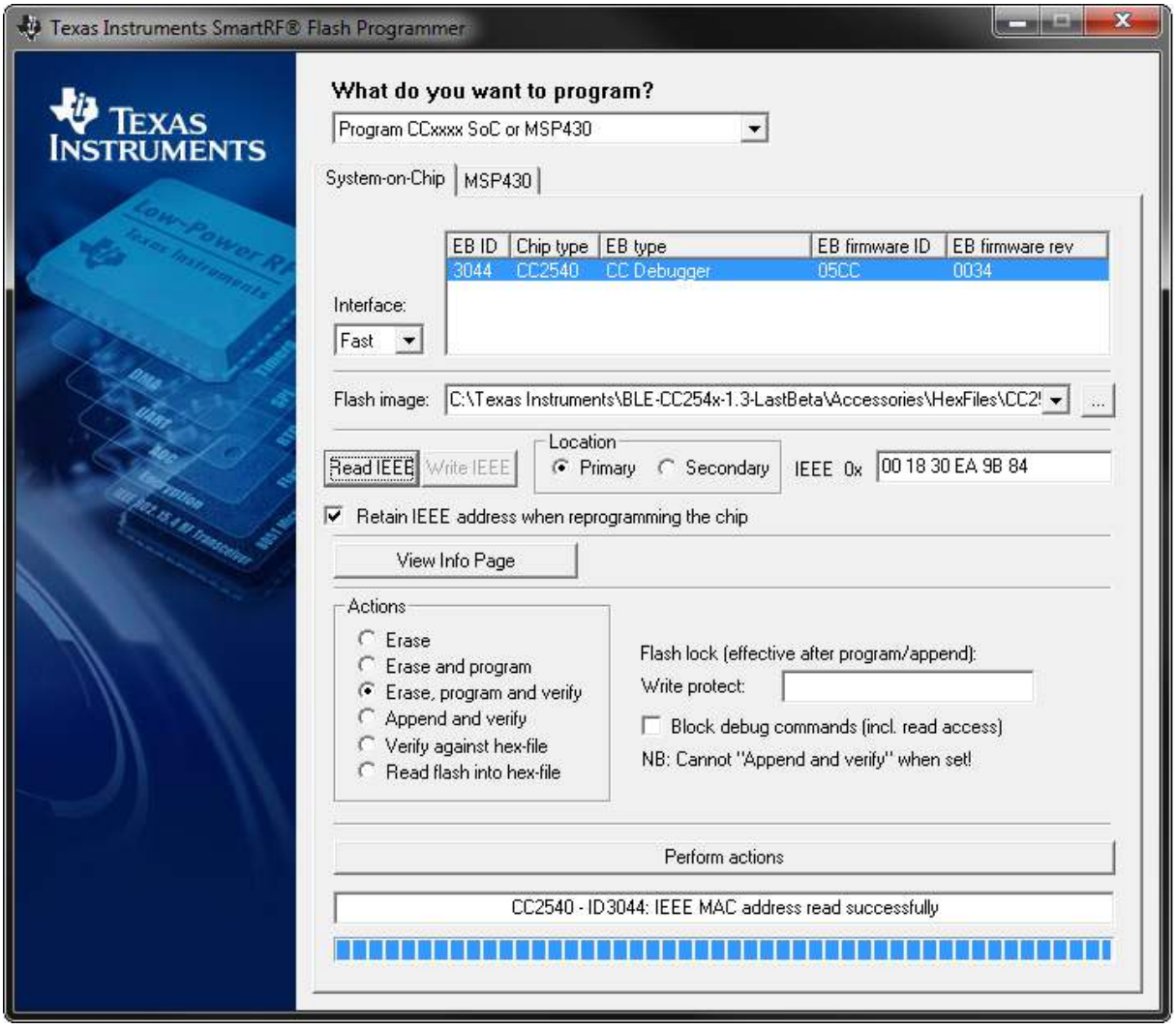

Figure 48 Flash Programmer, Read Primary address

To read the secondary address, select "Secondary" under the "Location" option, and click the "Read IEEE" button. The secondary device address should appear in the box on the right.

To set a new secondary address, select "Secondary" under the "Location" option, and enter the desired address in the box on the right. Click the "Write IEEE" button to perform the write. If the secondary device is set to "FF FF FF FF FF FF", the device will use the primary address. If the secondary device is set to anything else, the secondary address will be used.

6. SmartRF™ Packet Sniffer

The SmartRF™ Packet Sniffer is a PC software application used to display and store RF packets captured with a listening RF hardware node. Various RF protocols are supported, included *Bluetooth* low energy. The Packet Sniffer filters and decodes packets and displays them in a convenient way, with options for filtering and storage to a binary file format.

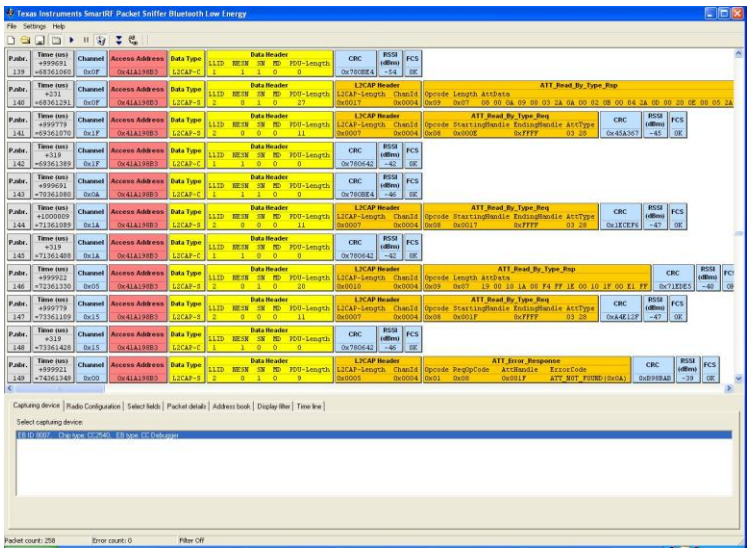

Figure 49 SmartRF Packet Sniffer

The CC2540 USB Dongle included with the CC2540/41 Mini Development Kit can be used as the listening hardware node, and can be useful when debugging *Bluetooth* low energy software applications. The SmartRF™ Packet Sniffer software can be downloaded at http://www.ti.com/tool/packet-sniffer.

7. General Information

7.1 Document History

Revision	Date	Description/Changes
SWRU270 (1.0)	2010-10-08	Initial release
SWRU270A (1.0.1)	2010-11-29	Added information about packet sniffer and KeyFobDemo application
SWRU270B (1.1)	2011-07-13	Updated with information from BLEv1.1 software release
SWRU270C (1.2)	2012-12-28	Updated with information on CC2541DK-MINI, now based on KeyFobDemo from BLEv1.3. Added warnings about coin cell battery.

Appendix

handle (hex)	handle (dec)	Type (hex)	Type (#DEFINE)	Value (default)	GATT Server Permissions	Notes
0x1	1	0x2800	GATT_PRIMARY_SERVICE_UUID	0x1800 (GAP_SERVICE_UUID)	GATT_PERMIT_READ	Start of GAP Service (Mandatory)
0x2	2	0x2803	GATT_CHARACTER_UUID	02 (read permissions) 03 00 (handle 0x0003) 00 2A (UUID 0x2A00)	GATT_PERMIT_READ	Device Name characteristic declaration
0x3	3	0x2A00	GAP_DEVICE_NAME_UUID	"CC2540DK-mini Keyfob"	GATT_PERMIT_READ	Device Name characteristic value
0x4	4	0x2803	GATT_CHARACTER_UUID	02 (read permissions) 05 00 (handle 0x0005) 01 2A (UUID 0x2A01)	GATT_PERMIT_READ	Appearance characteristic declaration
0x5	5	0x2A01	GAP_APPEARANCE_UUID	00 00	GATT_PERMIT_READ	Appearance characteristic value
0x6	6	0x2803	GATT_CHARACTER_UUID	0A (read/write permissions) 07 00 (handle 0x0007) 02 2A (UUID 0x2A02)	GATT_PERMIT_READ	Peripheral Privacy Flag characteristic declaration
0x7	7	0x2A02	GAP_PERI_PRIVACY_FLAG_UUID	0x00 (GAP_PRIVACY_DISABLED)	GATT_PERMIT_READ | GATT_PERMIT_WRITE	Peripheral Privacy Flag characteristic value
0x8	8	0x2803	GATT_CHARACTER_UUID	0A (read/write permissions) 09 00 (handle 0x0009) 03 2A (UUID 0x2A03)	GATT_PERMIT_READ	Reconnection address characteristic declaration
0x9	9	0x2A03	GAP_RECONNECT_ADDR_UUID	00 00 00 00 00 00	GATT_PERMIT_READ	Reconnection address characteristic value
0xA	10	0x2803	GATT_CHARACTER_UUID	02 (read permissions) 0B 00 (handle 0x000B) 04 2A (UUID 0x2A04)	GATT_PERMIT_READ	Peripheral Preferred Connection Parameters characteristic declaration
0xB	11	0x2A04	GAP_PERI_CONN_PARAM_UUID	50 00 (100ms preferred min connection interval) A0 00 (200ms preferred max connection interval) 00 00 (0 preferred slave latency) E8 03 (10000ms preferred supervision timeout)	GATT_PERMIT_READ	Peripheral Preferred Connection Parameters characteristic declaration
0xC	12	0x2800	GATT_PRIMARY_SERVICE_UUID	0x1801 (GATT_SERVICE_UUID)	GATT_PERMIT_READ	Start of GATT Service (Mandatory)
0xD	13	0x2803	GATT_CHARACTER_UUID	20 (indicate permissions) 0E 00 (handle 0x000E) 05 2A (UUID 0x2A05)	GATT_PERMIT_READ	Service Changed characteristic declaration
0xE	14	0x2A05	GATT_SERVICE_CHANGED_UUID	(null value)	(none)	Service Changed characteristic value
0xF	15	0x2902	GATT_CLIENT_CHAR_CFG_UUID	00 00	GATT_PERMIT_READ | GATT_PERMIT_WRITE	Service Changed characteristic value
0x10	16	0x2800	GATT_PRIMARY_SERVICE_UUID	0x180A (DEVINFO_SERV_UUID)	GATT_PERMIT_READ	Start of Device Information Service
0x11	17	0x2803	GATT_CHARACTER_UUID	02 (read permissions) 12 00 (handle 0x0012) 23 2A (UUID 0x2A23)	GATT_PERMIT_READ	System ID characteristic declaration
0x12	18	0x2A23	DEVINFO_SYSTEM_ID_UUID	xx xx xx 00 00 xx xx xx (xx's are IEEE address)	GATT_PERMIT_READ	System ID
0x13	19	0x2803	GATT_CHARACTER_UUID	02 (read permissions) 14 00 (handle 0x0014) 24 2A (UUID 0x2A24)	GATT_PERMIT_READ	Model Number String characteristic declaration
0x14	20	0x2A24	DEVINFO_MODEL_NUMBER_UUID	"Model Number"	GATT_PERMIT_READ	Model Number String
0x15	21	0x2803	GATT_CHARACTER_UUID	02 (read permissions) 16 00 (handle 0x0016) 25 2A (UUID 0x2A25)	GATT_PERMIT_READ	Serial Number String characteristic declaration
0x16	22	0x2A25	DEVINFO_SERIAL_NUMBER_UUID	"Serial Number"	GATT_PERMIT_READ	Serial Number String
0x17	23	0x2803	GATT_CHARACTER_UUID	02 (read permissions) 18 00 (handle 0x0018) 24 2A (UUID 0x2A24)	GATT_PERMIT_READ	Firmware Revision String characteristic declaration
0x18	24	0x2A26	DEVINFO_FIRMWARE_REV_UUID	"Firmware Revision"	GATT_PERMIT_READ	Firmware Revision String
0x19	25	0x2803	GATT_CHARACTER_UUID	02 (read permissions) 1A 00 (handle 0x001A) 24 2A (UUID 0x2A24)	GATT_PERMIT_READ	Hardware Revision String characteristic declaration
0x1A	26	0x2A27	DEVINFO_HARDWARE_REV_UUID	"Hardware Revision"	GATT_PERMIT_READ	Hardware Revision String
0x1B	27	0x2803	GATT_CHARACTER_UUID	02 (read permissions) 1C 00 (handle 0x001C) 24 2A (UUID 0x2A24)	GATT_PERMIT_READ	Software Revision String characteristic declaration
0x1C	28	0x2A28	DEVINFO_SOFTWARE_REV_UUID	"Software Revision"	GATT_PERMIT_READ	Software Revision String
0x1D	29	0x2803	GATT_CHARACTER_UUID	02 (read permissions) 1E 00 (handle 0x001E) 24 2A (UUID 0x2A24)	GATT_PERMIT_READ	Manufacturer Name String characteristic declaration
0x1E	30	0x2A29	DEVINFO_MANUFACTURER_NAME_UUID	"Manufacturer Name"	GATT_PERMIT_READ	Manufacturer Name String
0x1F	31	0x2803	GATT_CHARACTER_UUID	02 (read permissions) 20 00 (handle 0x0020) 24 2A (UUID 0x2A24)	GATT_PERMIT_READ	IEEE 11073-20601 Regulatory Certification Data List characteristic declaration
0x20	32	0x2A2A	DEVINFO_11073_CERT_DATA_UUID	FE 00 65 78 70 65 72 69 6D 65 6E 74 61 6C	GATT_PERMIT_READ	IEEE 11073-20601 Regulatory Certification Data List
0x21	33	0x2803	GATT_CHARACTER_UUID	02 (read permissions) 22 00 (handle 0x0022) 50 2A (UUID 0x2A50)	GATT_PERMIT_READ	PnP ID characteristic declaration
0x22	34	0x2A50	DEVINFO_PNP_ID_UUID	01 0D 00 00 00 10 01	GATT_PERMIT_READ	PnP ID

0x23	35	0x2800	GATT_PRIMARY_SERVICE_UUID	0x1803 (LINK_LOSS_SERVICE_UUID)	GATT_PERMIT_READ	Start of Link Loss Service
0x24	36	0x2803	GATT_CHARACTER_UUID	0A (read/write permissions) 25 00 (handle 0x0025) 06 2A (UUID 0x2A06)	GATT_PERMIT_READ	Alert Level characteristic declaration
0x25	37	0x2A06	PROXIMITY_ALERT_LEVEL_UUID	0 (PP_ALERT_LEVEL_NO)	GATT_PERMIT_READ | GATT_PERMIT_WRITE	Link Loss Alert Level characteristic value (see defined values in proxperiph.h)
0x26	38	0x2800	GATT_PRIMARY_SERVICE_UUID	0x1802 (IMMEDIATE_ALERT_SERVICE_UUID)	GATT_PERMIT_READ	Start of Immediate Alert Service
0x27	39	0x2803	GATT_CHARACTER_UUID	04 (write permission) 28 00 (handle 0x0028) 06 2A (UUID 0x2A06)	GATT_PERMIT_READ	Alert Level characteristic declaration
0x28	40	0x2A06	PROXIMITY_ALERT_LEVEL_UUID	0 (PP_ALERT_LEVEL_NO)	GATT_PERMIT_WRITE	Path Loss Alert Level characteristic value (see defined values in proxperiph.h)
0x29	41	0x2800	GATT_PRIMARY_SERVICE_UUID	0x1804(TX_PWR_LEVEL_SERVICE_UUID)	GATT_PERMIT_READ	Start of Tx Power Level Service
0x2A	42	0x2803	GATT_CHARACTER_UUID	12 (read permission) 2B 00 (handle 0x002B) 07 2A (UUID 0x2A07)	GATT_PERMIT_READ	Tx Power Level characteristic declaration
0x2B	43	0x2A07	PROXIMITY_TX_PWR_LEVEL_UUID	0	GATT_PERMIT_READ	Tx Power Level characteristic value (dBm)
0x2C	44	0x2902	GATT_CLIENT_CHAR_CFG_UUID	0x0000	GATT_PERMIT_READ | GATT_PERMIT_WRITE	Tx Power Level characteristic configuration
0x2D	45	0x2800	GATT_PRIMARY_SERVICE_UUID	0x180F(BATTERY_SERVICE_UUID)	GATT_PERMIT_READ	Start of Battery Service
0x2E	46	0x2803	GATT_CHARACTER_UUID	12 (read permission) 2F 00 (handle 0x002F) 19 2A (UUID 0x2A19)	GATT_PERMIT_READ	Battery level characteristic declaration
0x2F	47	0x2A19	BATTERY_LEVEL_UUID	100	GATT_PERMIT_READ	Battery level characteristic value (percentage)
0x30	48	0x2902	GATT_CLIENT_CHAR_CFG_UUID	00:00 (2 bytes)	GATT_PERMIT_READ | GATT_PERMIT_WRITE	Battery level characteristic configuration
0x31	49	0x2908	GATT_REPORT_REF_UUID	02:01 (2 bytes)	GATT_PERMIT_READ	HID Report Reference characteristic descriptor, Batter Level Input
0x32	50	0x2800	GATT_SERVICE_UUID	0xFFA0 (ACCEL_SERVICE_UUID)	GATT_PERMIT_READ	Start of Accelerometer Service
0x33	51	0x2803	GATT_CHARACTER_UUID	0A (read/write permissions) 34 00 (handle 0x0034) A1 FF (UUID 0xFFA1)	GATT_PERMIT_READ	Accelerometer enable characteristic declaration
0x34	52	0xFFA1	ACCEL_ENABLER_UUID	FALSE	GATT_PERMIT_READ | GATT_PERMIT_WRITE	Accelerometer enable characteristic value (TRUE or FALSE)
0x35	53	0x2901	GATT_CHAR_USER_DESC_UUID	"Accel Enable" (14 Bytes)	GATT_PERMIT_READ	Accelerometer enable characteristic user description
0x36	54	0x2803	GATT_CHARACTER_UUID	02 (read permission) 37 00 (handle 0x0037) A2 FF (UUID 0xFFA2)	GATT_PERMIT_READ	Accelerometer range characteristic declaration
0x37	55	0xFFA2	ACCEL_RANGE_UUID	20 (ACCEL_RANGE_2G)	GATT_PERMIT_READ	Accelerometer range characteristic value (can be 2G or 8G)
0x38	56	0x2901	GATT_CHAR_USER_DESC_UUID	"Accel Range" (13 Bytes)	GATT_PERMIT_READ	Accelerometer range characteristic user description
0x39	57	0x2803	GATT_CHARACTER_UUID	10 (notify permission) 3A 00 (handle 0x003A) A3 FF (UUID 0xFFA3)	GATT_PERMIT_READ	Accelerometer X-coordinate characteristic declaration
0x3A	58	0xFFA3	ACCEL_X_UUID	00:00 (2 bytes)	(none)	Accelerometer X-coordinate characteristic value
0x3B	59	0x2902	GATT_CLIENT_CHAR_CFG_UUID	00:00 (2 bytes)	GATT_PERMIT_READ | GATT_PERMIT_WRITE	Accelerometer X-coordinate characteristic configuration
0x3C	60	0x2901	GATT_CHAR_USER_DESC_UUID	"Accel X-Coordinate" (20 Bytes)	GATT_PERMIT_READ	Accelerometer X-coordinate characteristic user description
0x3D	61	0x2803	GATT_CHARACTER_UUID	10 (notify permission) 3E 00 (handle 0x003E) A4 FF (UUID 0xFFA4)	GATT_PERMIT_READ	Accelerometer Y-coordinate characteristic declaration
0x3E	62	0xFFA4	ACCEL_Y_UUID	00:00 (2 bytes)	(none)	Accelerometer Y-coordinate characteristic value
0x3F	63	0x2902	GATT_CLIENT_CHAR_CFG_UUID	00:00 (2 bytes)	GATT_PERMIT_READ | GATT_PERMIT_WRITE	Accelerometer Y-coordinate characteristic configuration
0x40	64	0x2901	GATT_CHAR_USER_DESC_UUID	"Accel Y-Coordinate" (20 Bytes)	GATT_PERMIT_READ	Accelerometer Y-coordinate characteristic user description
0x41	65	0x2803	GATT_CHARACTER_UUID	10 (notify permission) 42 00 (handle 0x0042) A5 FF (UUID 0xFFA5)	GATT_PERMIT_READ	Accelerometer Z-coordinate characteristic declaration
0x42	66	0xFFA5	ACCEL_Z_UUID	00:00 (2 bytes)	(none)	Accelerometer Z-coordinate characteristic value
0x43	67	0x2902	GATT_CLIENT_CHAR_CFG_UUID	00:00 (2 bytes)	GATT_PERMIT_READ | GATT_PERMIT_WRITE	Accelerometer Z-coordinate characteristic configuration
0x44	68	0x2901	GATT_CHAR_USER_DESC_UUID	"Accel Z-Coordinate" (20 Bytes)	GATT_PERMIT_READ	Accelerometer Z-coordinate characteristic user description
0x45	69	0x2800	GATT_SERVICE_UUID	0xFFE0 (SK_KEYPRESSED_UUID)	GATT_PERMIT_READ	Start of Simple Keys Service
0x46	70	0x2803	GATT_CHARACTER_UUID	10 (notify permission) 47 00 (handle 0x0047) E1 FF (UUID 0xFFE1)	GATT_PERMIT_READ	Key Press State characteristic declaration
0x47	71	0xFFE1	SK_KEYPRESSED_UUID	0 (1 byte)	(none)	Key Press State characteristic value
0x48	72	0x2902	GATT_CLIENT_CHAR_CFG_UUID	00:00 (2 bytes)	GATT_PERMIT_READ | GATT_PERMIT_WRITE	Key Press State characteristic configuration
0x49	73	0x2901	GATT_CHAR_USER_DESC_UUID	"Key Press State" (16 bytes)	GATT_PERMIT_READ	Key Press State characteristic user description

EVALUATION BOARD/KIT/MODULE (EVM) ADDITIONAL TERMS

Texas Instruments (TI) provides the enclosed Evaluation Board/Kit/Module (EVM) under the following conditions:

The user assumes all responsibility and liability for proper and safe handling of the goods. Further, the user indemnifies TI from all claims arising from the handling or use of the goods.

Should this evaluation board/kit not meet the specifications indicated in the User's Guide, the board/kit may be returned within 30 days from the date of delivery for a full refund. THE FOREGOING LIMITED WARRANTY IS THE EXCLUSIVE WARRANTY MADE BY SELLER TO BUYER AND IS IN LIEU OF ALL OTHER WARRANTIES, EXPRESSED, IMPLIED, OR STATUTORY, INCLUDING ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. EXCEPT TO THE EXTENT OF THE INDEMNITY SET FORTH ABOVE, NEITHER PARTY SHALL BE LIABLE TO THE OTHER FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES.

Please read the User's Guide and, specifically, the Warnings and Restrictions notice in the User's Guide prior to handling the product. This notice contains important safety information about temperatures and voltages. For additional information on TI's environmental and/or safety programs, please visit www.ti.com/esh or contact TI.

No license is granted under any patent right or other intellectual property right of TI covering or relating to any machine, process, or combination in which such TI products or services might be or are used. TI currently deals with a variety of customers for products, and therefore our arrangement with the user is not exclusive. TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein.

REGULATORY COMPLIANCE INFORMATION

As noted in the EVM User's Guide and/or EVM itself, this EVM and/or accompanying hardware may or may not be subject to the Federal Communications Commission (FCC) and Industry Canada (IC) rules.

For EVMs **not** subject to the above rules, this evaluation board/kit/module is intended for use for ENGINEERING DEVELOPMENT, DEMONSTRATION OR EVALUATION PURPOSES ONLY and is not considered by TI to be a finished end product fit for general consumer use. It generates, uses, and can radiate radio frequency energy and has not been tested for compliance with the limits of computing devices pursuant to part 15 of FCC or ICES-003 rules, which are designed to provide reasonable protection against radio frequency interference. Operation of the equipment may cause interference with radio communications, in which case the user at his own expense will be required to take whatever measures may be required to correct this interference.

General Statement for EVMs including a radio

User Power/Frequency Use Obligations: This radio is intended for development/professional use only in legally allocated frequency and power limits. Any use of radio frequencies and/or power availability of this EVM and its development application(s) must comply with local laws governing radio spectrum allocation and power limits for this evaluation module. It is the user's sole responsibility to only operate this radio in legally acceptable frequency space and within legally mandated power limitations. Any exceptions to this are strictly prohibited and unauthorized by Texas Instruments unless user has obtained appropriate experimental/development licenses from local regulatory authorities, which is responsibility of user including its acceptable authorization.

For EVMs annotated as FCC – FEDERAL COMMUNICATIONS COMMISSION Part 15 Compliant

Caution

This device complies with part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) This device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

FCC Interference Statement for Class A EVM devices

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

FCC Interference Statement for Class B EVM devices

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

For EVMs annotated as IC – INDUSTRY CANADA Compliant

This Class A or B digital apparatus complies with Canadian ICES-003.

Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

Concerning EVMs including radio transmitters

This device complies with Industry Canada licence-exempt RSS standard(s). Operation is subject to the following two conditions: (1) this device may not cause interference, and (2) this device must accept any interference, including interference that may cause undesired operation of the device.

Concerning EVMs including detachable antennas

Under Industry Canada regulations, this radio transmitter may only operate using an antenna of a type and maximum (or lesser) gain approved for the transmitter by Industry Canada. To reduce potential radio interference to other users, the antenna type and its gain should be so chosen that the equivalent isotropically radiated power (e.i.r.p.) is not more than that necessary for successful communication.

This radio transmitter has been approved by Industry Canada to operate with the antenna types listed in the user guide with the maximum permissible gain and required antenna impedance for each antenna type indicated. Antenna types not included in this list, having a gain greater than the maximum gain indicated for that type, are strictly prohibited for use with this device.

Cet appareil numérique de la classe A ou B est conforme à la norme NMB-003 du Canada.

Les changements ou les modifications pas expressément approuvés par la partie responsable de la conformité ont pu vider l'autorité de l'utilisateur pour actionner l'équipement.

Concernant les EVMs avec appareils radio

Le présent appareil est conforme aux CNR d'Industrie Canada applicables aux appareils radio exempts de licence. L'exploitation est autorisée aux deux conditions suivantes : (1) l'appareil ne doit pas produire de brouillage, et (2) l'utilisateur de l'appareil doit accepter tout brouillage radioélectrique subi, même si le brouillage est susceptible d'en compromettre le fonctionnement.

Concernant les EVMs avec antennes détachables

Conformément à la réglementation d'Industrie Canada, le présent émetteur radio peut fonctionner avec une antenne d'un type et d'un gain maximal (ou inférieur) approuvé pour l'émetteur par Industrie Canada. Dans le but de réduire les risques de brouillage radioélectrique à l'intention des autres utilisateurs, il faut choisir le type d'antenne et son gain de sorte que la puissance isotrope rayonnée équivalente (p.i.r.e.) ne dépasse pas l'intensité nécessaire à l'établissement d'une communication satisfaisante.

Le présent émetteur radio a été approuvé par Industrie Canada pour fonctionner avec les types d'antenne énumérés dans le manuel d'usage et ayant un gain admissible maximal et l'impédance requise pour chaque type d'antenne. Les types d'antenne non inclus dans cette liste, ou dont le gain est supérieur au gain maximal indiqué, sont strictement interdits pour l'exploitation de l'émetteur.

【Important Notice for Users of this Product in Japan】

This development kit is NOT certified as Confirming to Technical Regulations of Radio Law of Japan

If you use this product in Japan, you are required by Radio Law of Japan to follow the instructions below with respect to this product:

1. Use this product in a shielded room or any other test facility as defined in the notification #173 issued by Ministry of Internal Affairs and Communications on March 28, 2006, based on Sub-section 1.1 of Article 6 of the Ministry's Rule for Enforcement of Radio Law of Japan,
2. Use this product only after you obtained the license of Test Radio Station as provided in Radio Law of Japan with respect to this product, or
3. Use of this product only after you obtained the Technical Regulations Conformity Certification as provided in Radio Law of Japan with respect to this product. Also, please do not transfer this product, unless you give the same notice above to the transferee. Please note that if you could not follow the instructions above, you will be subject to penalties of Radio Law of Japan.

Texas Instruments Japan Limited
(address) 24-1, Nishi-Shinjuku 6 chome, Shinjuku-ku, Tokyo, Japan

http://www.tij.co.jp

【ご使用にあたっての注】

本開発キットは技術基準適合証明を受けておりません。

本製品のご使用に際しては、電波法遵守のため、以下のいずれかの措置を取っていただく必要がありますのでご注意ください。

1. 電波法施行規則第6条第1項第1号に基づく平成18年3月28日総務省告示第173号で定められた電波暗室等の試験設備でご使用いただく。
2. 実験局の免許を取得後ご使用いただく。
3. 技術基準適合証明を取得後ご使用いただく。

なお、本製品は、上記の「ご使用にあたっての注意」を譲渡先、移転先に通知しない限り、譲渡、移転できないものとします。

上記を遵守頂けない場合は、電波法の罰則が適用される可能性があることをご留意ください。

日本テキサス・インスツルメンツ株式会社
東京都新宿区西新宿６丁目２４番１号
西新宿三井ビル
http://www.tij.co.jp

EVALUATION BOARD/KIT/MODULE (EVM)
WARNINGS, RESTRICTIONS AND DISCLAIMERS

For Feasibility Evaluation Only, in Laboratory/Development Environments. Unless otherwise indicated, this EVM is not a finished electrical equipment and not intended for consumer use. It is intended solely for use for preliminary feasibility evaluation in laboratory/development environments by technically qualified electronics experts who are familiar with the dangers and application risks associated with handling electrical mechanical components, systems and subsystems. It should not be used as all or part of a finished end product.

Your Sole Responsibility and Risk. You acknowledge, represent and agree that:

1. You have unique knowledge concerning Federal, State and local regulatory requirements (including but not limited to Food and Drug Administration regulations, if applicable) which relate to your products and which relate to your use (and/or that of your employees, affiliates, contractors or designees) of the EVM for evaluation, testing and other purposes.
2. You have full and exclusive responsibility to assure the safety and compliance of your products with all such laws and other applicable regulatory requirements, and also to assure the safety of any activities to be conducted by you and/or your employees, affiliates, contractors or designees, using the EVM. Further, you are responsible to assure that any interfaces (electronic and/or mechanical) between the EVM and any human body are designed with suitable isolation and means to safely limit accessible leakage currents to minimize the risk of electrical shock hazard.
3. You will employ reasonable safeguards to ensure that your use of the EVM will not result in any property damage, injury or death, even if the EVM should fail to perform as described or expected.
4. You will take care of proper disposal and recycling of the EVM's electronic components and packing materials.

Certain Instructions. It is important to operate this EVM within TI's recommended specifications and environmental considerations per the user guidelines. Exceeding the specified EVM ratings (including but not limited to input and output voltage, current, power, and environmental ranges) may cause property damage, personal injury or death. If there are questions concerning these ratings please contact a TI field representative prior to connecting interface electronics including input power and intended loads. Any loads applied outside of the specified output range may result in unintended and/or inaccurate operation and/or possible permanent damage to the EVM and/or interface electronics. Please consult the EVM User's Guide prior to connecting any load to the EVM output. If there is uncertainty as to the load specification, please contact a TI field representative. During normal operation, some circuit components may have case temperatures greater than 60°C as long as the input and output are maintained at a normal ambient operating temperature. These components include but are not limited to linear regulators, switching transistors, pass transistors, and current sense resistors which can be identified using the EVM schematic located in the EVM User's Guide. When placing measurement probes near these devices during normal operation, please be aware that these devices may be very warm to the touch. As with all electronic evaluation tools, only qualified personnel knowledgeable in electronic measurement and diagnostics normally found in development environments should use these EVMs.

Agreement to Defend, Indemnify and Hold Harmless. You agree to defend, indemnify and hold TI, its licensors and their representatives harmless from and against any and all claims, damages, losses, expenses, costs and liabilities (collectively, "Claims") arising out of or in connection with any use of the EVM that is not in accordance with the terms of the agreement. This obligation shall apply whether Claims arise under law of tort or contract or any other legal theory, and even if the EVM fails to perform as described or expected.

Safety-Critical or Life-Critical Applications. If you intend to evaluate the components for possible 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, such as devices which are classified as FDA Class III or similar classification, then you must specifically notify TI of such intent and enter into a separate Assurance and Indemnity Agreement.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2012, Texas Instruments Incorporated

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

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

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license 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 significant portions 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. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

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

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2013, Texas Instruments Incorporated