TI Designs

Commissioning Sensors in a Sub-1 GHz Network Over Bluetooth® low energy Reference Design

TEXAS INSTRUMENTS

Description

This TI Design demonstrates how to easily commission a Sub-1 GHz sensor node over Bluetooth® low energy (BLE), which enables quick connectivity with a smartphone or tablet device. The design is powered by the CC1350 SimpleLink™ dual-band wireless microcontroller (MCU), which is the sensor node being commissioned. There is an added option to emulate a sensor network concentrator using a SimpleLink dual-band CC1350 or Sub-1 GHz CC1310 for a complete user experience.

This reference design provides demonstration software to be used in conjunction with the TI *Bluetooth* low energy BLE-Stack 2.2.1 software development kit (SDK) and the TI-RTOS RF driver, along with a list of additional hardware and software to quickly begin IoT product development.

The goal of this TI Design is to reduce the complexity and cost of the commissioning process by allowing the user to commission over BLE, which improves the user-experience. This design also demonstrates how to reduce the overall connection time by a factor of 100 or more by removing the need to re-flash the network credentials to the device.

Resources

TIDC-01001	Design Folder
CC1350	Product Folder
CC1310	Product Folder

TI E2E™ Community

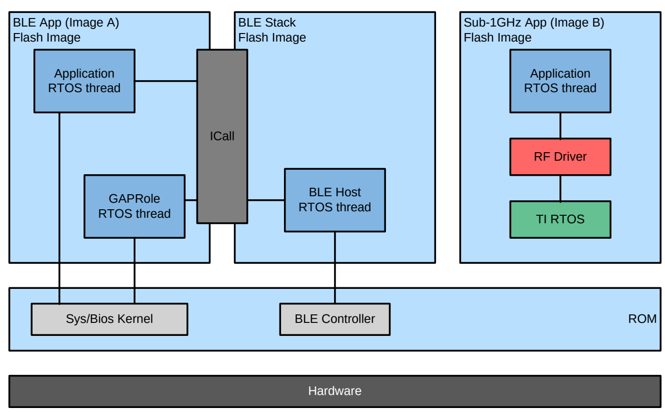
ASK Our E2E Experts

Features

- Enables Simple Commissioning of a Sensor in a Sub-1 GHz Network Over BLE Using a Smartphone or Tablet Device
- Switches Quickly (Within 20 ms) From a BLE to a Sub-1 GHz Network Connection
- Supported by TI SimpleLink Dual-Band CC1350 LaunchPad™ Development Kits Acting as Sensor Device
- Leverages the Latest SimpleLink CC13xx TI-RTOS RF Driver Examples and Latest TI BLE-Stack, Providing a Scalable Software Development Platform

Applications

- Door and Window Sensor Networks
- Motion Detector
- Garage Door Opener
- Electronic Smart Lock



An IMPORTANT NOTICE at the end of this TI reference design addresses authorized use, intellectual property matters and other important disclaimers and information.

1 System Overview

1.1 System Description

This TI Design provides a reference for easily commissioning a sensor node over BLE for use in a Sub-1 GHz network. In this design, the CC1350 LaunchPad acts as the sensor node to be commissioned. The user can connect to the BLE peripheral (using a smartphone or tablet device) and write a 16-byte characteristic value. This value represents the Sub-1 GHz network credentials necessary for commissioning the node. Once written, the device immediately reboots into Sub-1 GHz mode and continuously transmits packets containing the credentials. For a complete user experience, the user can use a CC1310/50 LaunchPad or development kit to emulate the Sub-1 GHz network concentrator. With the software provided, the packets will be received by the concentrator, and the network credentials can be displayed through a UART terminal or on an LCD BoosterPack™ plug-in module.

The software of this design is built on the CC1350 dual image concept example found in the TI BLE-Stack. It expands the capability of this example by adding a connectable BLE peripheral to the existing BLE application image and a simple implementation of the RF Driver to the second application image to enable Sub-1 GHz packet transmission. Software is included to add display functionality to CC1310/50 RF Packet RX example in TI-RTOS so that the user can receive the packets being sent over Sub-1 GHz.

The accessibility and ease of BLE combined with the low-power, long-range capability of Sub-1 GHz makes this TI Design beneficial for any distributed sensing network that requires commissioning. This reference design also provides a framework for easily switching from connected BLE to Sub-1 GHz operation on the CC1350.

1.2 Block Diagram

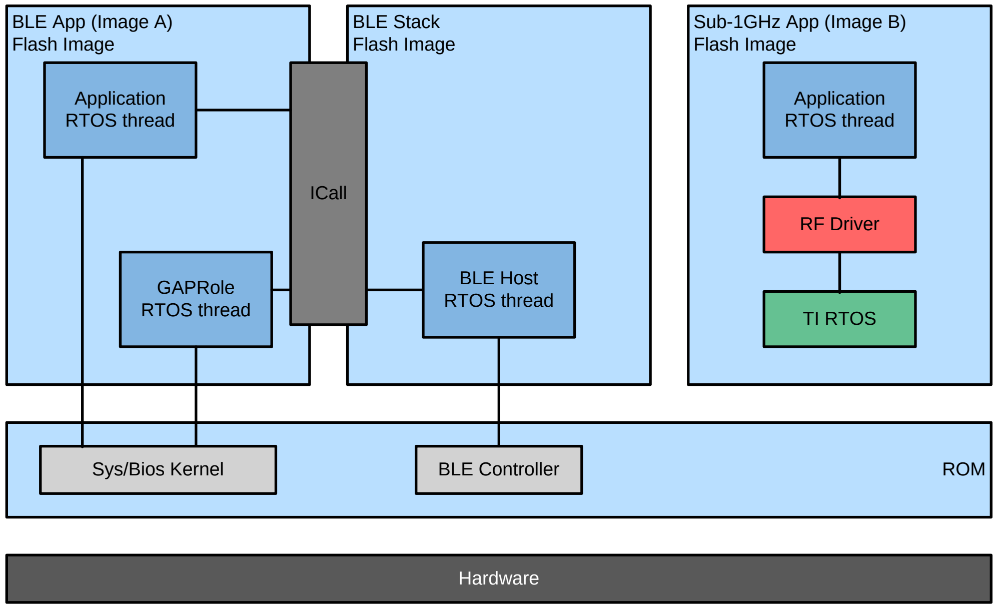


Figure 1. Top-Level Software Architecture

The CC1350 BLE plus Sub-1 GHz software environment consists of the following parts:

- A real-time operating system (RTOS)
- A BLE application image
- A BLE stack image
- A Sub-1 GHz application image

The TI-RTOS is a real-time, pre-emptive, multithreaded operating system that runs the software solution with task synchronization.

For BLE operation, both the application and BLE protocol stack exist as separate tasks within the RTOS. The BLE protocol stack has the highest priority. A messaging framework called indirect call (ICall) is used for thread-safe synchronization between the application and the stack. Figure 1 shows the architecture.

The stack image includes the lower layers of the BLE protocol stack [(1)], which consists of the controller and the host. Most of the BLE protocol stack code is provided as a library.

The BLE application image includes the profiles, application code, drivers, and the ICall module.

The Sub-1 GHz application image includes the application code, drivers, and Sub-1 GHz radio settings.

(1) For more information on the BLE protocol stack, see the *CC2640 and CC2650 SimpleLink Bluetooth low energy Software Stack 2.2.1 Developer's Guide*[1].

1.2.1 Functional Diagram

Figure 2 shows a flow chart describing how the Sub-1 GHz sensor node can be commissioned using a BLE connection.

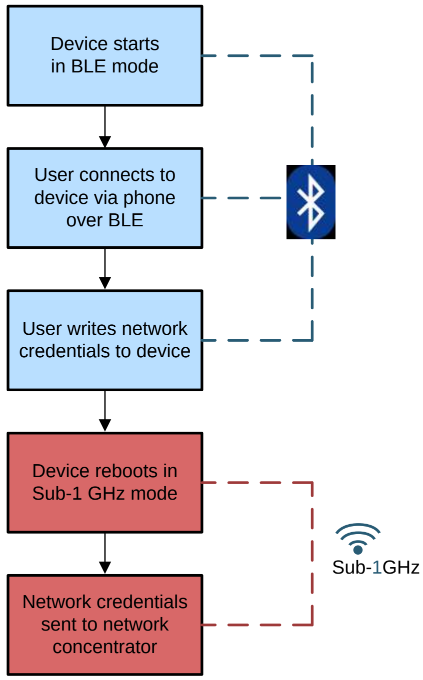


Figure 2. Operational Flow Chart

1.3 Highlighted Products

1.3.1 SimpleLink™ Ultra-Low-Power CC1350 and CC1310

The CC1350 is a member of the CC26xx and CC13xx family of cost-effective, ultra-low-power 2.4-GHz and Sub-1 GHz RF devices. Very-low active RF and MCU current consumption, in addition to flexible low-power modes, provide excellent battery lifetime and allow long-range operation on small coin-cell batteries and in energy-harvesting applications.

The CC1350 is the first device in the CC13xx and CC26xx family of cost-effective, ultra-low-power wireless MCUs capable of handling both Sub-1 GHz and 2.4-GHz RF frequencies. The CC1350 device combines a flexible, very-low-power RF transceiver with a powerful 48-MHz ARM® Cortex®-M3 MCU in a platform supporting multiple physical layers and RF standards. A dedicated radio controller (Cortex-M0) handles low-level RF protocol commands that are stored in ROM or RAM, thus, ensuring ultra-low power and flexibility to handle both Sub-1 GHz protocols and 2.4-GHz protocols (for example, BLE). This enables the combination of a Sub-1 GHz communication solution that offers the best possible RF range together with a Bluetooth low energy smartphone connection that enables great user experience through a phone application. The Sub-1 GHz only device in this family is the CC1310.

Figure 3 shows the block diagram of the SimpleLink CC1350 wireless MCU.

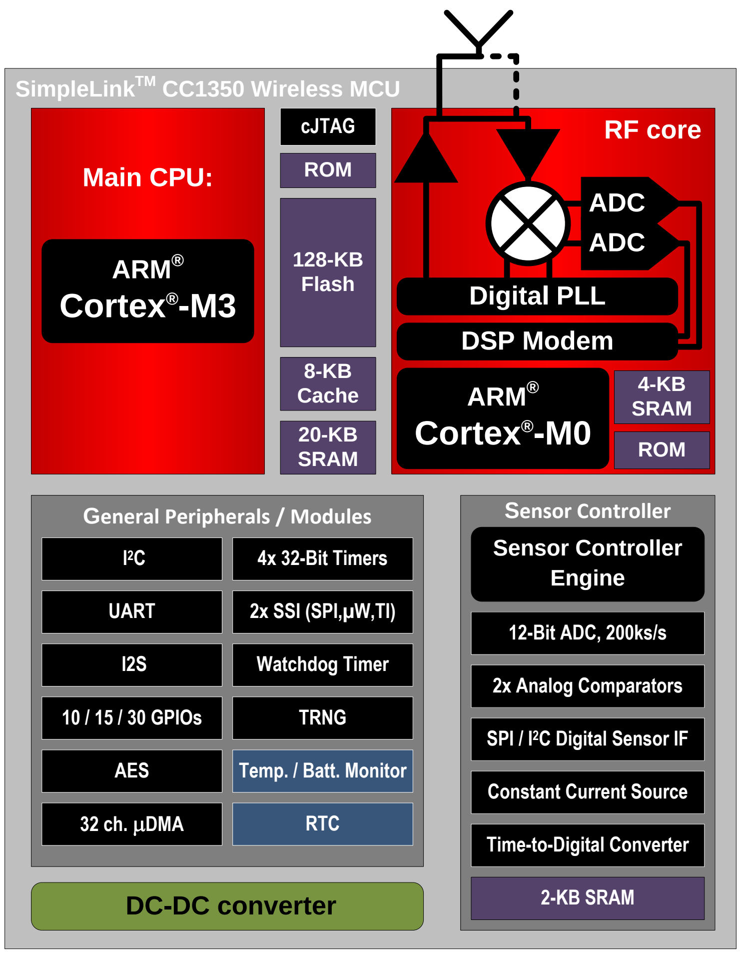


Figure 3. CC1350 Block Diagram

The CC1350 device is a highly-integrated, true single-chip solution incorporating a complete RF system and an on-chip DC-DC converter.

Sensors can be handled in a very-low-power manner by a dedicated autonomous ultra-low-power MCU that can be configured to handle analog and digital sensors; thus, the main MCU (Cortex-M3) can maximize sleep time.

1.3.2 TI BLE-STACK 2.21

TI's BLE-Stack 2.21 SDK is a full-featured *Bluetooth* 4.2 certified stack that includes all necessary software and sample applications to quickly get started with the development of single-mode BLE applications.

1.3.3 TI-RTOS for CC13xx and CC26xx

TI-RTOS is a scalable, one-stop embedded tools ecosystem for TI devices. TI-RTOS scales from a real-time multitasking kernel (SYS/BIOS) to a complete RTOS solution including additional middleware components and device drivers. By providing essential system software components that are pre-tested and pre-integrated, TI-RTOS enables the user to focus on creating the application

1.3.4 SmartRF™ Studio

SmartRF™ Studio is a Windows® application that can be used to evaluate and configure low-power RF devices from Texas Instruments. The application will help designers of RF systems to easily evaluate the radio at an early stage in the design process. SmartRF Studio is especially useful for generation of configuration register values and for practical testing and debugging of the RF system.

2 Getting Started Hardware and Software

2.1 Required Hardware

- A CC1350 LaunchPad to emulate the sensor node to be commissioned and to run the dual-mode application
- A CC1310/50 LaunchPad and an optional LCD BoosterPack to emulate a Sub-1 GHz network concentrator and display the packet sent from the node
- A smartphone or tablet device to commission the sensor node

2.2 Required Software

- Code Composer Studio™ (6.2 or later) with TI ARM Compiler 5.2.6 or later
- BLE-STACK 2.21 (Includes TI-RTOS 2.20)
- SmartRF Flash Programmer 2
- (Optional) SmartRF Studio
- Python® 2.7
- A BLE development application for phone or tablet (such as LightBlue™ Explorer by Punch Through™ Design, LLC for iOS or BLE Scanner by Bluepixel Technologies, LLP for Android®)
- (Optional) A Serial Terminal Program (such as Tera Term or CoolTerm)

2.3 Included Software

- dual_image_concept_img_a.c – the BLE application source file, which contains the application task
- dual_image_concept_img_b.c – the Sub-1 GHz application source file, which contains the application task
- sub1packetService.c/h – a BLE generic attribute profile (GATT) service (2) used for entering the Sub-1 GHz network credentials
- smartrf_settings.c/h – the Sub-1 GHz radio settings (3)

(2) The sub1packetService source files were generated using the Example Service Generator in the Custom Profile lab of *SimpleLink Academy*[2]. More information on GATT services and profiles can be found in the *CC2640 and CC2650 SimpleLink Bluetooth low energy Software Stack 2.2.1 Developer's Guide*[1].

(3) These files can be generated in SmartRF Studio using custom user settings.

- rfPacketRx.c – a Sub-1 GHz receiver application, which adds display functionality to the existing RF Packet RX example in TI-RTOS

2.4 Building the Project in CCS

2.4.1 Configure CCS

Configure CCS as described in Section 2.6.3.1 of the *CC2640 and CC2650 SimpleLink Bluetooth low energy Software Stack 2.2.1 Developer's Guide*[1]. TI ARM Compiler version 5.2.6, 5.2.7, or 5.2.8 must be installed when using the BLE SDK.

2.4.2 Add the Project Files

1. Copy the included source files dual_image_concept_img_a.c and dual_image_concept_img_b.c to $BLE_INSTALL$\src\examples\dual_image_concept\cc1350\app [4]. It may be useful to make copies of the existing source files in this directory, as they will be replaced.
2. Copy the included folders sub1packetService and smartrf_settings to $BLE_INSTALL$\examples\cc1350lp\dual_image_concept\ccs\app.

2.4.3 Import the Existing Project

1. Create a workspace. Ensure that the CCS workspace path does not contain a whitespace.
2. Open the CCS IDE from the Start Menu.
3. Select Project → Import CCS Project.
4. Select $BLE_INSTALL$\examples\cc1350lp\dual_image_concept\ccs.
5. Click the box next to the application project and the stack project to select them.
6. Click Finish to import (Figure 4).

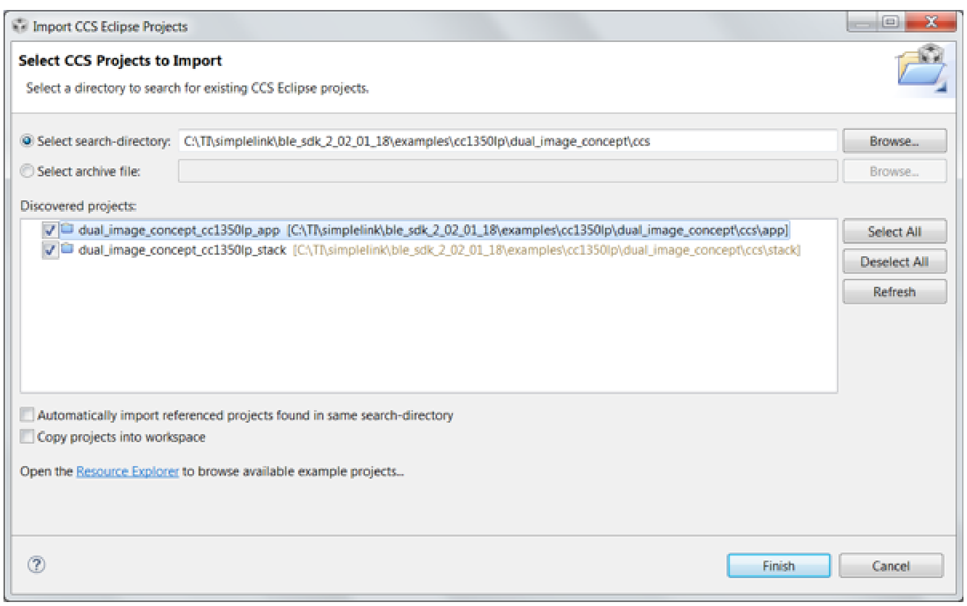


Figure 4. Importing Application and Stack Projects

7. Select Project → Import CCS Project.
8. Repeat steps 5 through 7 for the boot image manager (BIM) project, which is located in $BLE_INSTALL$\examples\util\bim\cc1350lp\ccs.

(4) $BLE_INSTALL$ is the directory where the BLE SDK is installed. The default directory for BLE SDK 2.2.1 is C:\TI\simplelink\ble_sdk_2_02_01_18.

2.4.4 Set Up the Build Environment

1. Right-click on the application project in the Project Explorer, and select Properties.
2. In the left pane, select General.
3. Under the Main tab, in Advanced settings, make sure that the compiler version is set to either TI v5.2.6, v5.2.7, or v5.2.8.
4. Under the RTSC tab, in Products and Repositories, select only the version of TI-RTOS that is included in the BLE SDK installation.
5. In the Platform drop-down menu, select ti.platforms.simplelink:CC1350F128
6. Select OK.
7. Steps 1 through 6 should be completed for both configurations, FlashROM_ImgA and FlashOnly_ImgB.
8. Open the project Properties window again, and select Build → ARM Compiler → Include Options.
9. Select FlashROM_ImgA as the Configuration.
10. Add the directory ${PROJECT_LOC}/sub1packetService to the #include search path (Figure 5).

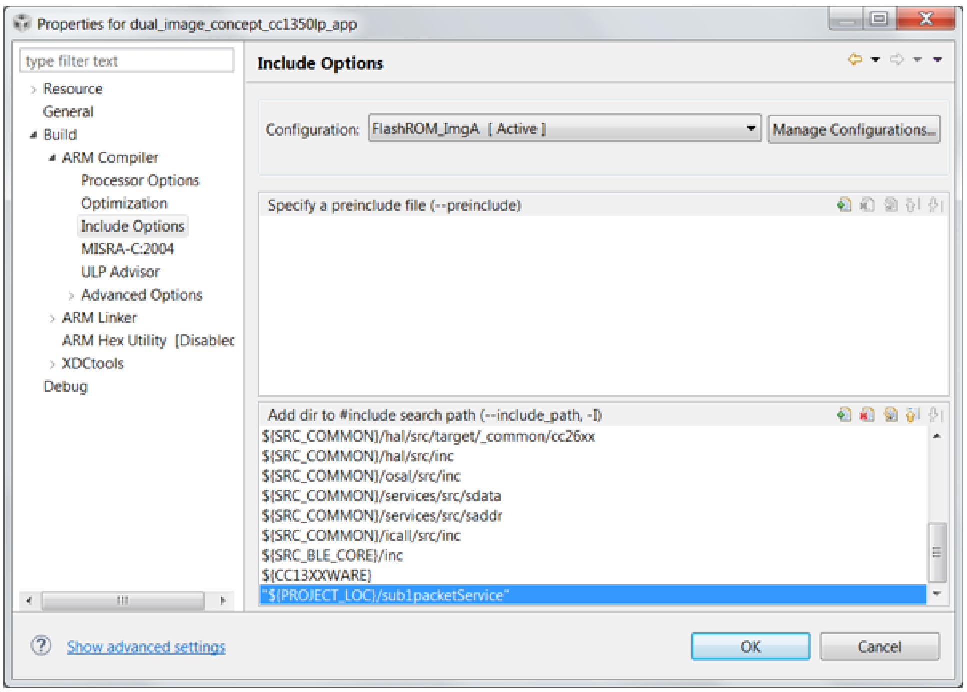


Figure 5. Including a Directory in Search Path

11. Select FlashOnly_ImgB as the Configuration.
12. Add the directory ${PROJECT_LOC}/smartrf_settings to the #include search path.
13. Select OK.

2.4.5 Build the Project

1. Right-click on the BIM project and select Build Project.
2. Build the stack project.
3. Delete the output folders FlashROM_ImgA and FlashOnly_ImgB in the *app* directory, if available (this can be done from CCS).
4. Delete the *src* directory, if available, located in $BLE_INSTALL$\examples\cc1350lp\dual_image_concept\ccs\config.
5. Set the Build Configuration of the application project to FlashROM_ImgA, and build the project (Figure 6).

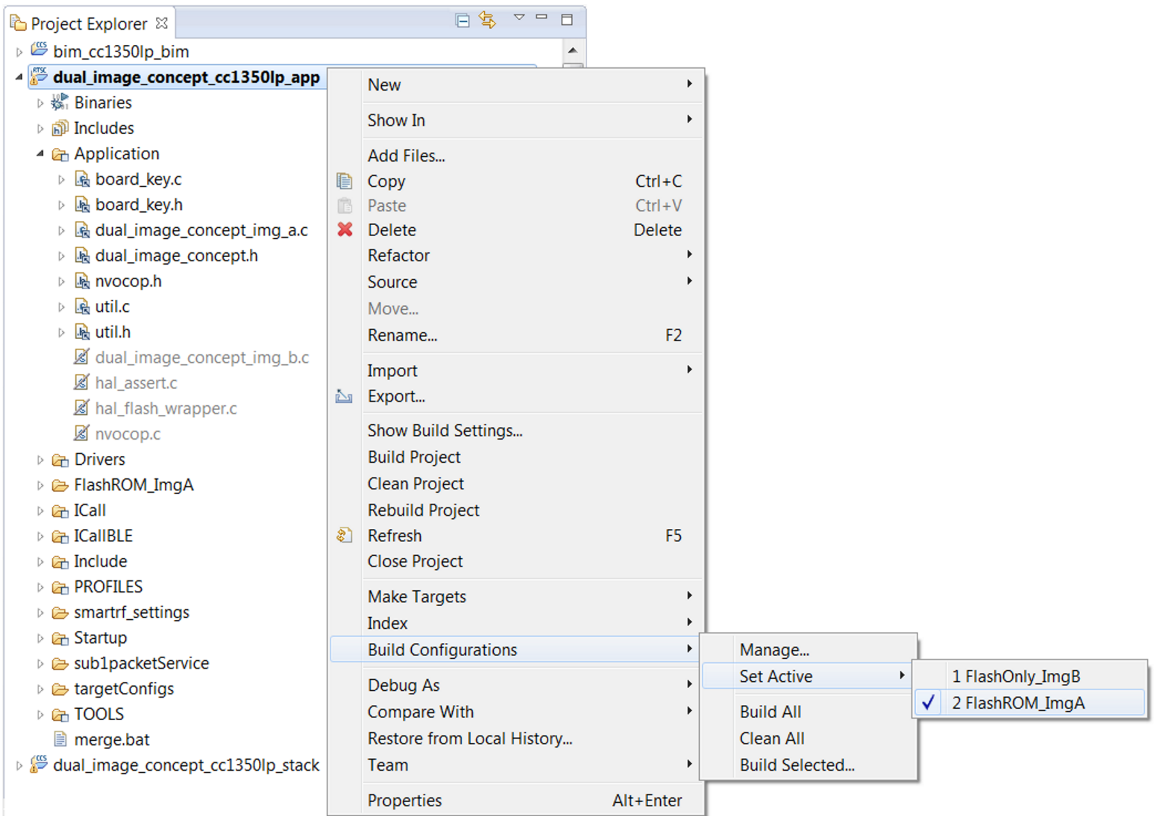


Figure 6. Setting Project Build Configurations

6. Repeat step 4.
7. Set the Build Configuration of the application project to FlashOnly_ImgB, and build the project.
8. Run merge.bat, located in $BLE_INSTALL$\ble_sdk_2_02_01_18\examples\cc1350lp\dual_image_concept\ccs\app. This should be done outside of CCS (requires Python 2.7).

NOTE: merge.bat merges the BIM, stack, and application images into dual_image_concept_cc1350lp_unified.hex, located in $BLE_INSTALL$\ble_sdk_2_02_01_18\examples\cc1350lp\dual_image_concept\ccs\app\FlashOnly_ImgB

NOTE: Steps 3 through 8 must be completed for every new application build.

2.4.6 Program the LaunchPad™

1. Open SmartRF Flash Programmer 2.
2. Select the CC1350 in list of connected devices.

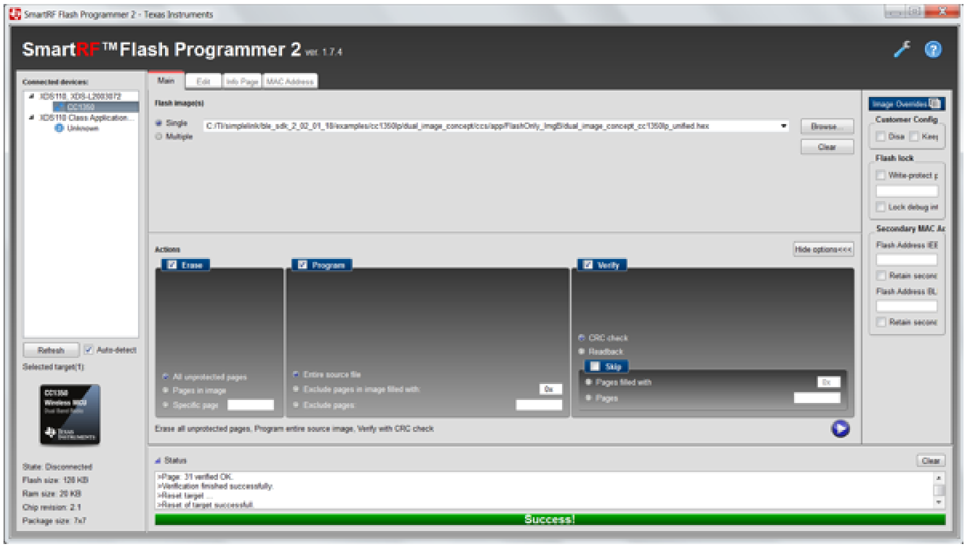


Figure 7. SmartRF Flash Programmer 2

3. Click on the wrench icon in the upper right corner and select CC26xx/CC13xx Forced Mass Erase (Figure 8).

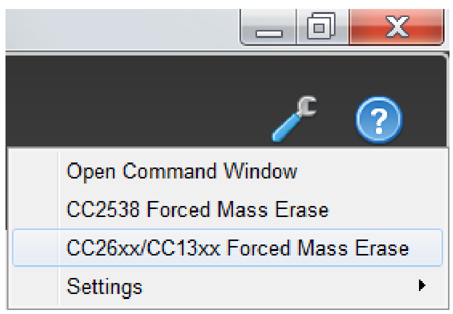


Figure 8. Erasing Flash on CC1350

4. Select the CC1350 again and browse for the unified hex file that was created with merge.bat.
5. Select the actions Erase, Program, and Verify, and click  to program the device.
6. Press the reset button on the LaunchPad. This reset must be done every time the device is power-cycled.
7. The red LED will be on while the CC1350 is in BLE mode.

2.5 Running the Demonstration

1. Open the BLE development application on a smartphone or tablet and enable *Bluetooth* on the device.
2. Search for nearby devices, find the *CC1350 Dual Mode TX* peripheral, and connect to it.

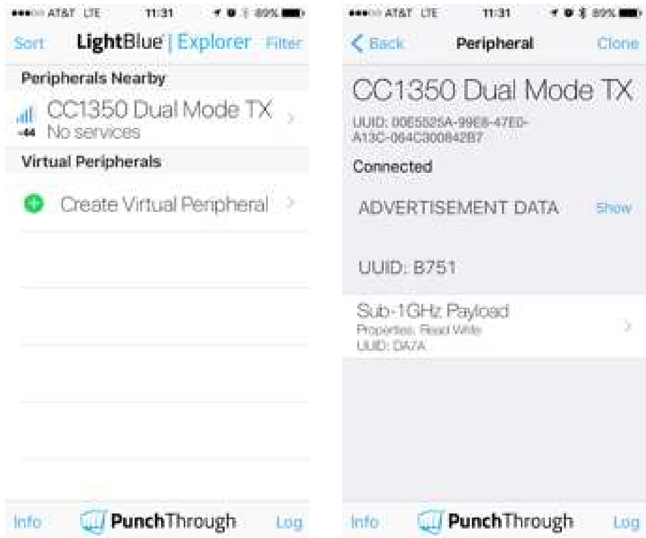


Figure 9. Connecting to BLE Peripheral

3. Select the Sub-1 GHz Payload Service to display the modifiable characteristic.

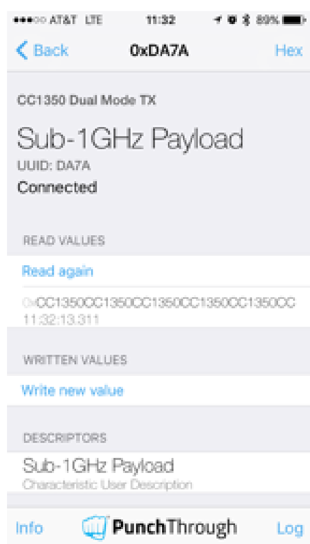


Figure 10. Selecting Characteristic

4. Write a 16-byte (or less) hexadecimal value to the characteristic. This value will be the Sub-1 GHz network credentials.

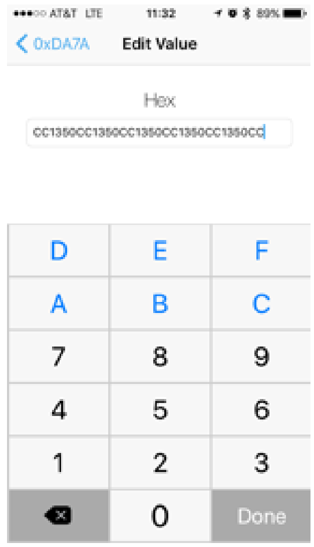


Figure 11. Writing Characteristic Value

5. Once the value is written, the device will immediately disconnect from BLE and begin transmitting packets containing the previously written credentials over Sub-1 GHz.
6. The green LED will toggle when a Sub-1 GHz packet is transmitted.
7. To reboot the device in BLE mode and write another value, press Button 2 (right) on the LaunchPad.

2.5.1 Receiving the Sub-1 GHz Packets

Use a CC1310 or CC1350 LaunchPad or development kit to receive and display the Sub-1 GHz packets containing the network credentials.

2.5.1.1 Using SmartRF™ Studio

1. Open SmartRF Studio.
2. Select the receiving device in the list of connected devices.
3. Using the default configuration (50 kbps, 2-GFSK, 25kHz Deviation), select Packet RX, set the expected packet count to Infinite, and press Start.
4. The received packets will be displayed with the sequence number, the payload, and the RSSI.

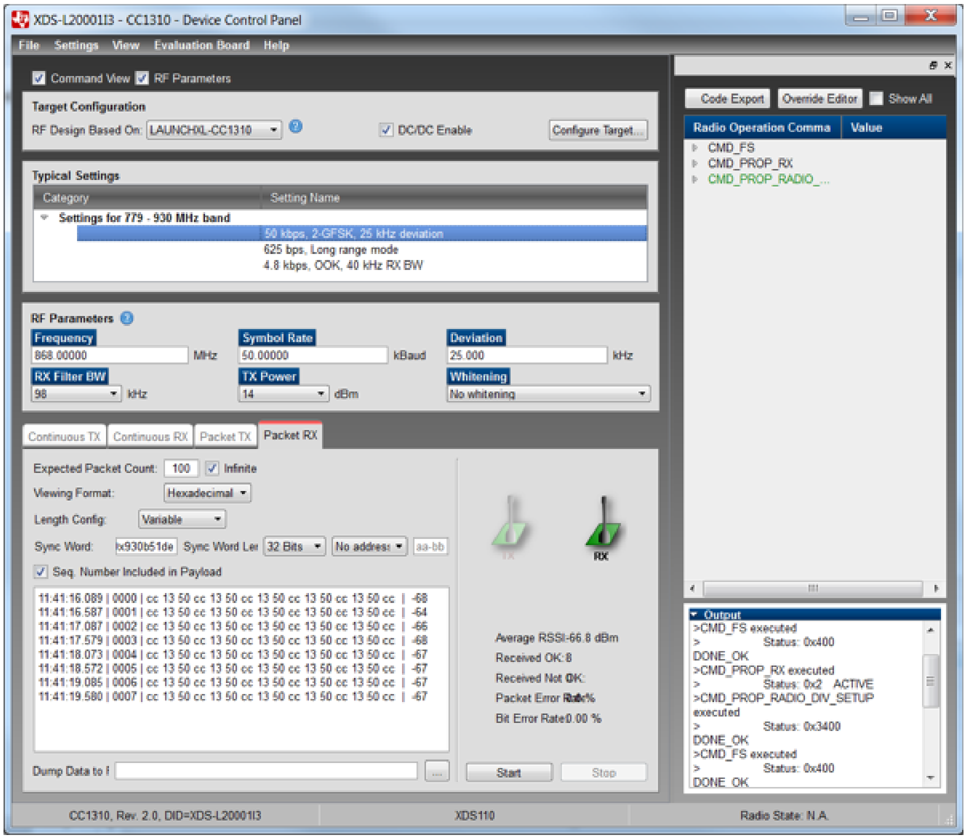


Figure 12. Using SmartRF™ Studio to Receive Packets

2.5.1.2 Using the RF Packet RX Example

1. In CCS, select View → Resource Explorer to display example projects.
2. In the Resource Explorer, select TI-RTOS for CC13XX and CC26XX → SimpleLink Wireless MCU → CC1310F128/CC1350F128 → CC1310 / CC1350 Launchpad / Development Kit → Driver Examples → TI Driver Examples → RF Examples → RF Packet RX.

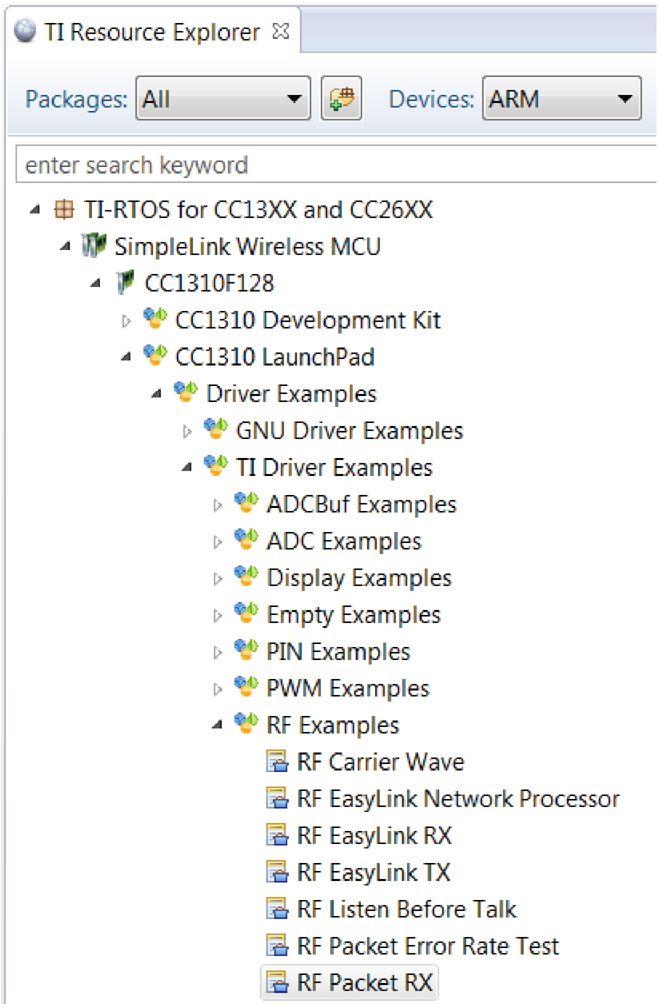


Figure 13. Importing RF Packet RX From Resource Explorer

3. Select Import the example project into CCS.
4. Replace rfPacketRx.c with the rfPacketRx.c included in this reference design. The included file adds display capability to display the sequence number, the payload, and the RSSI.
5. Build and download the project to the board.
6. Open a terminal program and connect to the COM port associated with the device (will be named User Application UART for the LaunchPad). Use a baud rate of 115200.

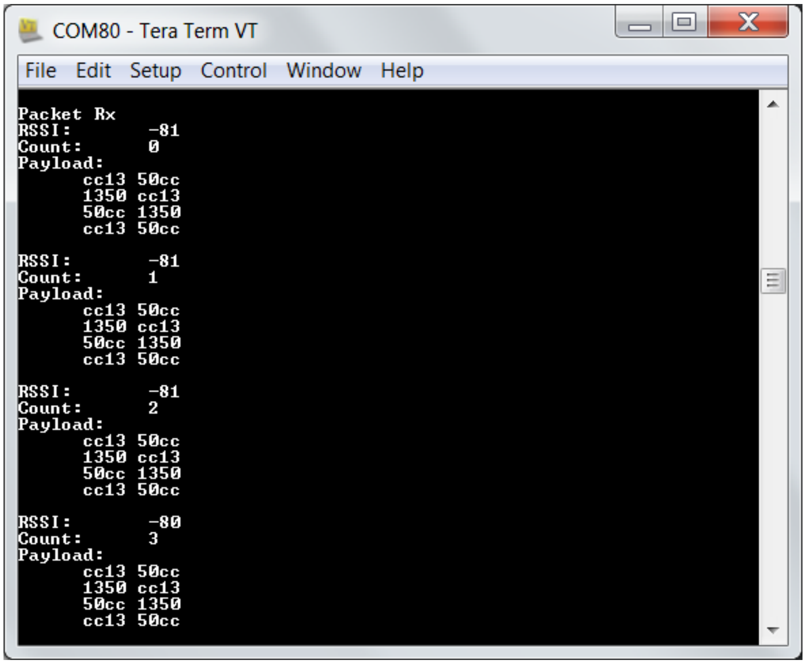


Figure 14. Displaying the Packet on UART Terminal

7. (Optional) The packet information can also be displayed using the LCD BoosterPack. Simply change the macro Display_Type from Display_Type_UART to Display_Type_LCD before building, and attach the BoosterPack.

3 Software Application Architecture

The purpose of this section is to describe the software of the dual image application from a systems perspective.

3.1 BIM and Application Main

The application begins with the BIM, where an image ID is read from nonvolatile memory. This ID tells the BIM whether to load Image A (BLE) or Image B (Sub-1 GHz). At the end of the BIM main, the program jumps to the beginning of the specified image and runs through that image's application main.

In the application main, the pins are initialized, the tasks are created, and the operating system is initiated.

Figure 15 shows the main functions of the BIM and the application.

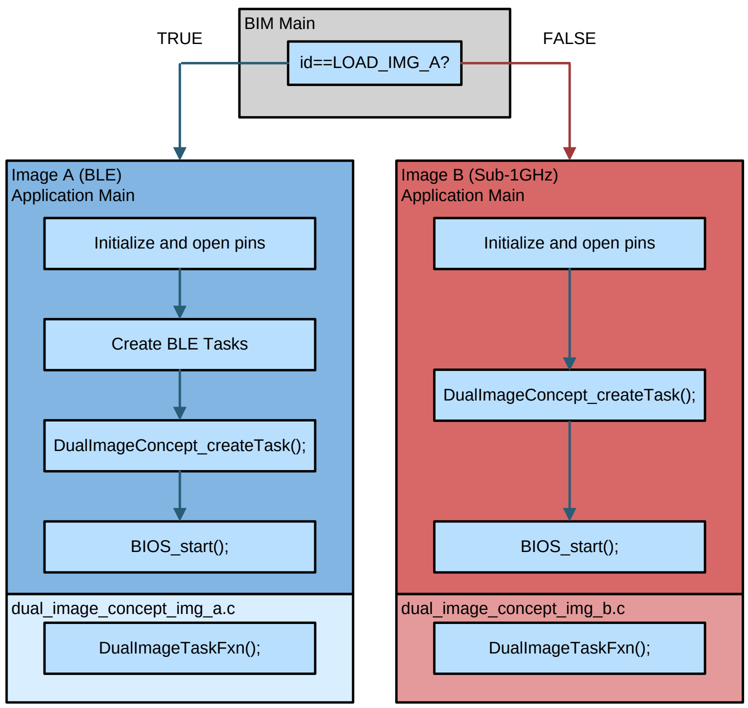


Figure 15. BIM and Application Main Functions

3.2 BLE Application Task (Image A)

The BLE application task establishes a connectable BLE link and processes messages from the Sub1Packet Service and from key presses. Figure 16 shows the BLE application's task functions. Additional services and profiles can be added to extend the functionality of the BLE application(2).

After the BLE initialization function, the task function enters an infinite loop, where it remains blocked and waits until a semaphore is posted by the ICall module. When the semaphore is posted, the task resumes and checks both the service message queue and the application message queue.

When a user writes a value to the sub1packetService characteristic, the value is stored in a specific area of RAM, so that it can be accessed globally. Once this value is written, a message containing the event DIC_KEY_CHANGE_EVT and status DIC_SWITCH_IMAGE_BUTTON is posted to the appMsg queue (this message will also be posted when the user presses the right key). This is done in the write attribute callback function within sub1packetService.c.

When the message is processed, LOAD_IMG_B is written to the image ID, and the device is reset to the BIM.

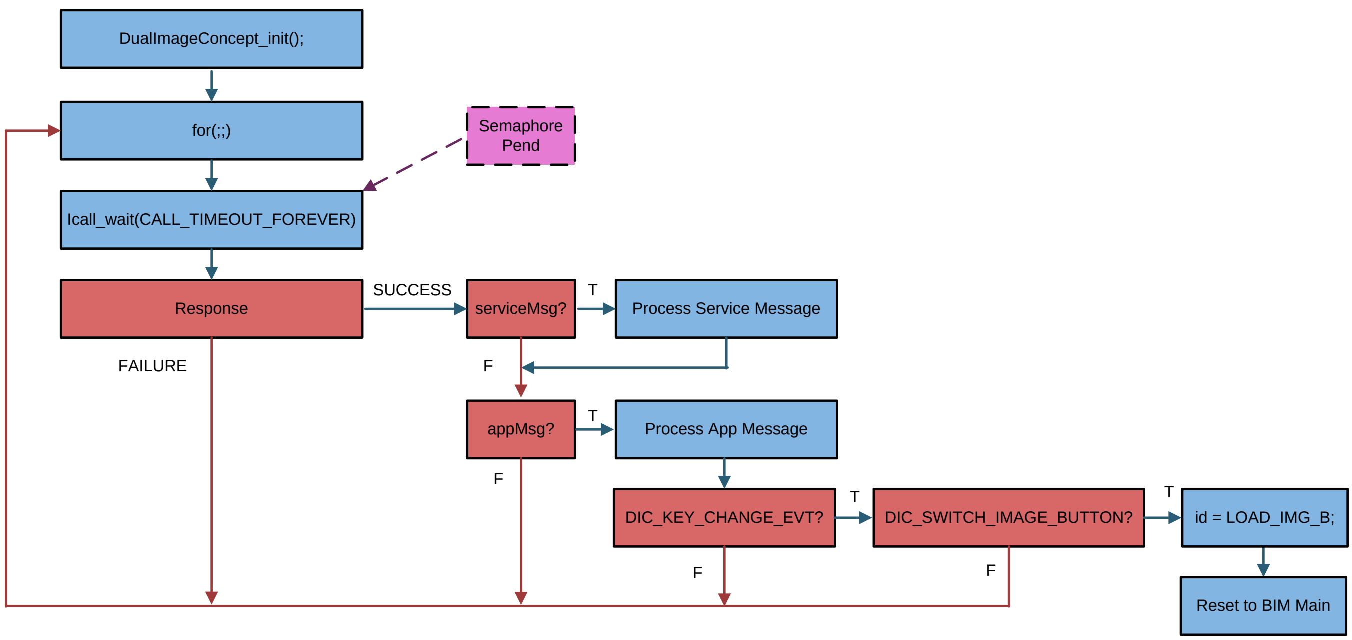


Figure 16. BLE Application Task Function

3.3 Sub-1 GHz Application Task (Image B)

The Sub-1 GHz application task makes use of the RF Driver to operate the radio using the SmartRF settings. Figure 17 shows the Sub-1 GHz application's task functions.

After the application initialization function, the RF Driver is used to open the radio handle and set the frequency, and the task enters an infinite loop. In the infinite loop, the Sub-1 GHz packet is created to contain an incrementing sequence number and the value stored in RAM from the BLE application. This packet is then transmitted every 500 ms, and the application message queue is checked after each transmission.

When the user presses the right key, an interrupt triggers, and a message containing the event DIC_KEY_CHANGE_EVT and status DIC_SWITCH_IMAGE_BUTTON is posted to the message queue. When the message is processed, LOAD_IMG_A is written to the image ID, and the device is reset to the BIM.

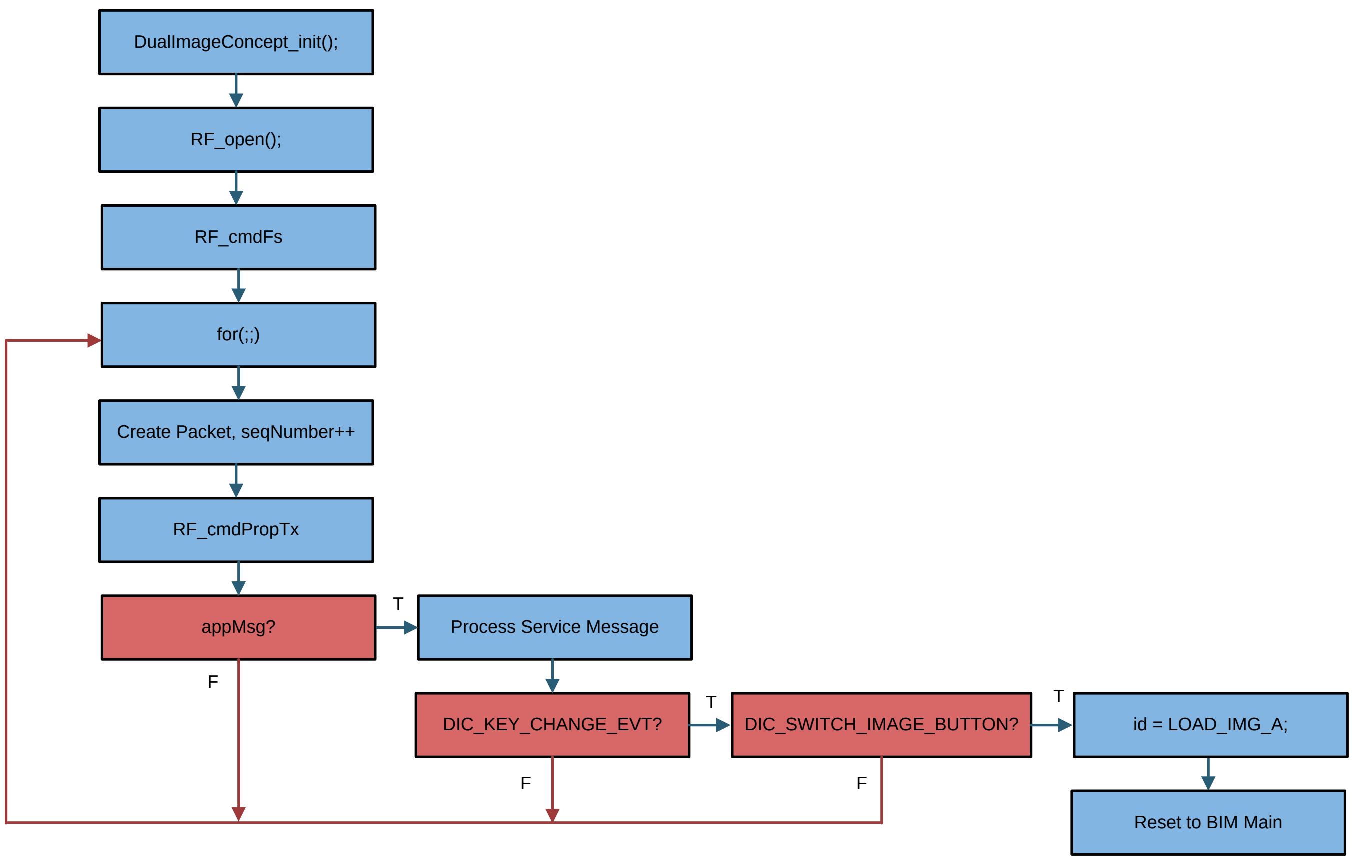


Figure 17. Sub-1 GHz Application Task Function

4 Testing and Results

4.1 Image Switching Time

After the development of this design, testing was done to determine the switching time between the BLE connection and the Sub-1 GHz transmission, and vice-versa. To perform this test, a GPIO is turned on once the device begins advertising in BLE mode, and is turned off when the Sub-1 GHz application image is loaded. A second GPIO is turned on once the first Sub-1 GHz transmission is completed, and is turned off when the BLE application image is loaded.

Using the method described, we effectively measured the switching time between a BLE connection to a completed Sub-1 GHz transmission to be about 12.26 ms (Figure 18). The switching time between the Sub-1 GHz image and BLE advertisement is about 18.47 ms (Figure 19).

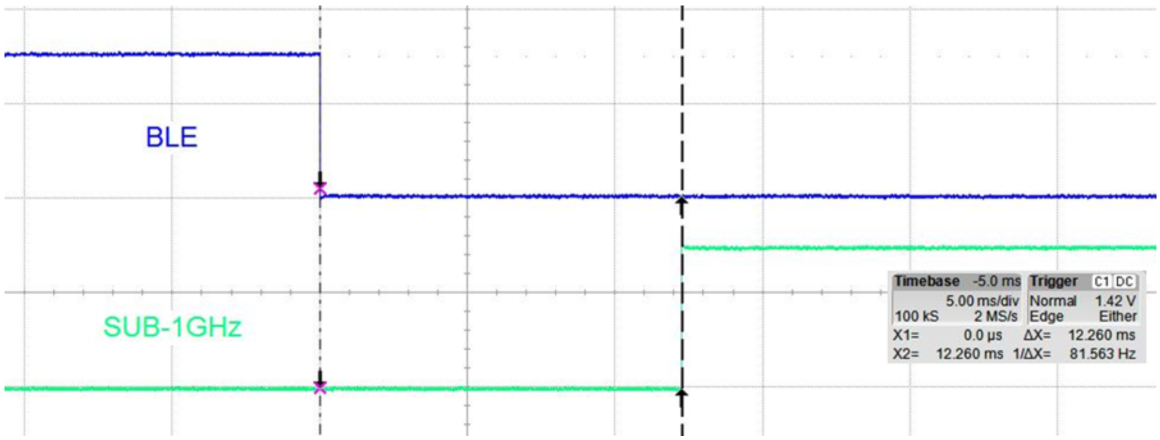


Figure 18. Switching Time From BLE Image to Sub-1 GHz Image

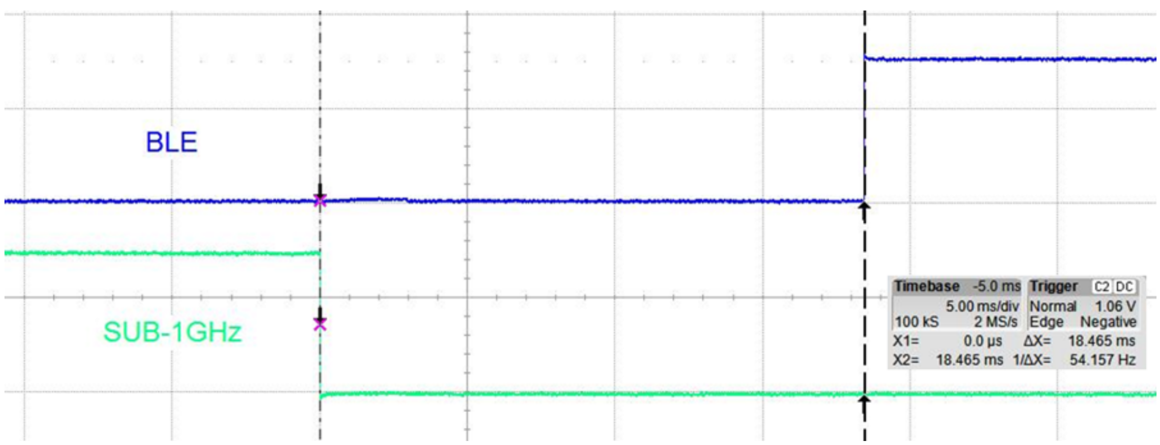


Figure 19. Switching Time From Sub-1 GHz Image to BLE Image

4.2 Flash Memory Allocation

Upon the final software build, the memory maps of each image were assessed to determine the amount of flash used. Figure 20 shows the flash memory map at the system level.

Flash Area	Start Address	Memory Used
Interrupt Vectors	0x00000	0x00c8
BLE Application (Image A)	0x000d0	0x7596
Sub-1GHz Application (Image B)	0x08000	0x716a
BLE Stack Image Code Space	0x10000	0xda40
Simple NV (SNV) Area	0x1e000	0x1000
BIM Area	0x1f000	0x031c

Figure 20. Flash Memory Map

The various sections of flash and their associate linker files are as follows.

- BLE Application (Image A): code space for the BLE application project. This image is configured in the linker configuration file of the application: cc13xx_dual_img.cmd.
- Sub-1 GHz Application (Image B): code space for the Sub-1 GHz application project. This image is configured in the linker configuration file of the application: cc13xx_dual_img.cmd.
- BLE Stack Image: code space for the stack project. This image is configured in the linker configuration file of the stack: cc26xx_stack.cmd.
- Simple NV (SNV): area used for nonvolatile memory storage by the GAP Bond Manager and also available for use by the application. See Section 3.10.3 of the *CC2640 and CC2650 SimpleLink Bluetooth low energy Software Stack 2.2.1 Developer's Guide*[1] for configuring SNV. When configured, the SNV flash storage area is part of the stack image.
- BIM Area: the last sector of flash used to store the BIM project and the customer-specific chip configuration (CCFG) parameters. This image is configured in the linker configuration file of the BIM: cc13xx_bim_dual_img.cmd.

5 Design Files

5.1 Schematics

To download the schematics, see the design files at TIDC-01001.

5.2 Bill of Materials

To download the bill of materials (BOM), see the design files at TIDC-01001.

5.3 PCB Layout Recommendations

5.3.1 Layout Prints

To download the layer plots, see the design files at TIDC-01001.

5.4 Cadence Project

To download the Cadence project files, see the design files at TIDC-01001.

5.5 Gerber Files

To download the Gerber files, see the design files at TIDC-01001.

5.6 Assembly Drawings

To download the assembly drawings, see the design files at TIDC-01001.

6 Software Files

To download the software files, see the design files at TIDC-01001.

7 Related Documentation

1. Texas Instruments, *CC2640 and CC2650 SimpleLink™ Bluetooth® low energy Software Stack 2.2.1*, Developer's Guide (SWRU393)
2. Texas Instruments, *SimpleLink Academy v1.11* (http://software-dl.ti.com/lprf/simplelink_academy/overview.html)
3. Texas Instruments, *TI Sub-1 GHz Wiki* (http://processors.wiki.ti.com/index.php/Category:Sub-1 GHz)
4. Texas Instruments, *TI BLE Wiki* (http://processors.wiki.ti.com/index.php/Category:BluetoothLE)

7.1 Trademarks

All trademarks are the property of their respective owners.

8 About the Author

SKYLER SCHMIDT is an Applications Engineer at Texas Instruments, where he is responsible for supporting customer designs using low power wireless systems. Skyler earned his Bachelor of Science in Electrical Engineering from the University of Michigan in Ann Arbor, MI.