Advanced BLE System Design Lab 1: Over-The-Air Firmware Upgrade

Description

This lab teaches you how to add Over-The-Air (OTA) firmware upgrade functionality to your application. With this OTA implementation you will be able to upgrade your Application as well as the BLE stack in the field.

This lab starts with the Find Me code example in PSoC Creator, and then adds OTA capability.

Objectives

1. Understand the architecture of an Upgradable Stack Bootloader
2. Learn how to add OTA Firmware Upgrade capability to your project in PSoC Creator
3. Learn how to use the CySmart software and the BLE-USB bridge as a host device to upgrade firmware

Requirements	Details
Hardware	BLE Pioneer Kit (CY8CKIT-042-BLE) BLE 256KB Module (CY8CKIT-143A: Having device CY8C4248LQI-BL583) CySmart BLE 4.2 USB Dongle (CY5677)
Software	PSoC Creator 3.3 CP2 or later (www.cypress.com/creator)
	CySmart 1.2 (or newer)

Pre-Reading

The following section briefly describes the important terms related to PSoC Creator projects required to implement OTA upgrade functionality.

Bootloader

The Bootloader is the portion of the firmware that knows how to update the flash memory and is responsible for doing so.

Bootloadable

The Bootloadable is the portion of the firmware that contains the application that is received by the Bootloader and is updated in the flash memory of the target device. In other words, the Bootloader updates the Bootloadable.

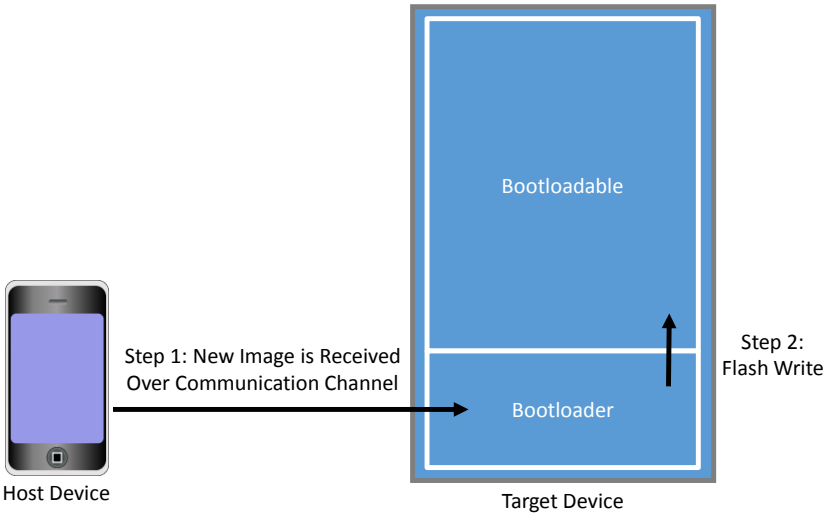
Bootloading

Bootloading is the process of transferring the Bootloadable image from a host (for example a PC or a Smartphone) to a target device. It is also referred to as a bootload operation, bootload, firmware upgrade, or device firmware upgrade (DFU).

Host

The Host is an external device, such as mobile phone or laptop that sends the firmware image to the Bootloader, over a communication channel. These communication channels can be wired (such as UART, I2C, SPI, SWD, etc.), or wireless (such as BLE). Refer to Figure 1.

Figure 1. The Bootloading Process



Stack Application

The Stack Application project holds the BLE Stack and a custom service for the Bootloader. It can download a new image either for the User Application or for the Stack Application itself.

User Application

The User Application project is the user project (such as a Find Me application) to be run on the device. It reuses the BLE stack from the Stack Application project. This User Application project is also referred to as the user project, application project, application image or simply user application in this lab manual.

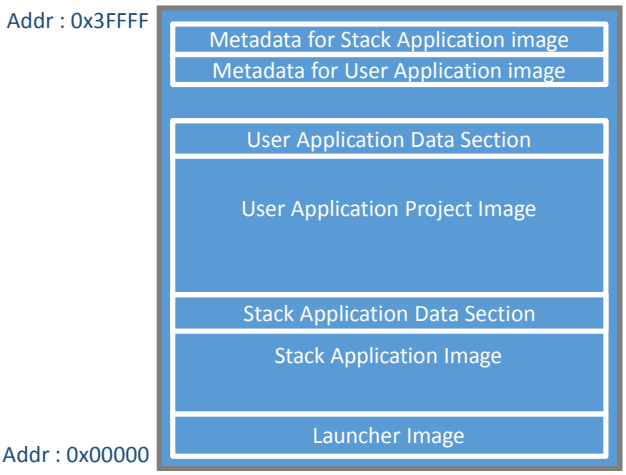
Launcher

The Launcher project or Launcher image is responsible for launching either the Stack Application or the User Application. Additionally, if the Stack Application has downloaded a new stack image, the Launcher replaces the current stack image with the new stack image. The Launcher is basically a Bootloader without a communication component.

Architecture for implementing OTA upgrade feature

This lab implements the Bootloading process using BLE as the communication component. This architecture has three major parts – Launcher project, Stack Application project and User Application project, as shown in Figure 2. The architecture is designed to support firmware upgrade of both the Stack Application and the User Application.

Figure 2. Detailed Flash Memory Organization used in Upgradable Stack OTA Architecture



At power up, the device starts executing code from the Launcher. The Launcher then starts either the Stack Application image or the User Application image, depending on which application is active, and on the validation of the images. The Launcher can also copy the latest Stack Application image (previously downloaded over the BLE link and stored at the User Application Image location in a flash) to the Stack Application image location, thereby upgrading the BLE Stack. This architecture is called an Upgradable Stack OTA architecture.

All three project images are created using separate PSoC Creator projects. These projects are then stitched together to create a final flash image. This image is used to program the device for the first time. After that depending on the user requirement, either just the User Application image, or both the Stack and User Application images can be upgraded over the air.

The device flash also has a 64-byte block called the metadata section. The Bootloader and the Bootloadable components use the metadata section to store/retrieve project image specific settings, such as the size of the application image, application ID, application version, copy flag, etc. This metadata is located at the end of the flash as shown in Figure 2. A detailed description of the various fields of the metadata section and their functions are provided in the *Metadata Memory Map* section of the *Bootloader and Bootloadable* component datasheet.

Let us look at the three important concepts, namely, Code Sharing, RAM sharing, and Checksum Exclusion, required for implementing this OTA upgrade architecture using PSoC Creator.

Code Sharing:

In this architecture, the Stack Application project contains the BLE component. It is used by the Bootloader component in the same project to download a new image from the host. The User Application project does not have a BLE stack of its own, but reuses it from the Stack Application project. This is possible through a PSoC Creator feature called Code Sharing. This feature allows the Stack Application project to share its BLE component with the User Application project.

RAM Sharing:

The BLE Stack residing in the Stack Application project requires some RAM that must be allocated statically. Since User Application project does not have a Stack of its own, we need to allocate the same memory manually. So, in the User Application project, custom linker scripts are used to provide some reserved RAM space for the BLE Stack operation. Note that this results in a reduction of the amount of RAM available for the User Application project.

These settings are already present in the linker scripts provided with this lab and do not need any changes for any application that uses BLE devices with 256KB of flash memory.

Checksum Exclusion:

During the Bootloading process, a Bootloader checks for the integrity of the application code residing in the flash by calculating its checksum and verifying it against the originally stored checksum. However, some configurations of the BLE component require saving some data (called Bonding data) to a flash memory space, which is a part of the application's flash memory space. These sections are shown in Figure 2, as 'Stack Application data section' and 'User Application data section'.

The bonding data changes whenever PSoC BLE device is paired to other devices. When bonding data changes, the stored checksum of the application image is no longer valid. In such cases, the bootloader would halt application loading as soon as it detects a mismatch in the calculated and existing checksum values.

To avoid this situation, we need to exclude data sections for the Stack Application project and the User Application project from checksum calculations. This is done using linker script commands. These commands are included in the provided linker scripts and do not need any user modifications for any application that uses BLE device with 256KB of flash memory.

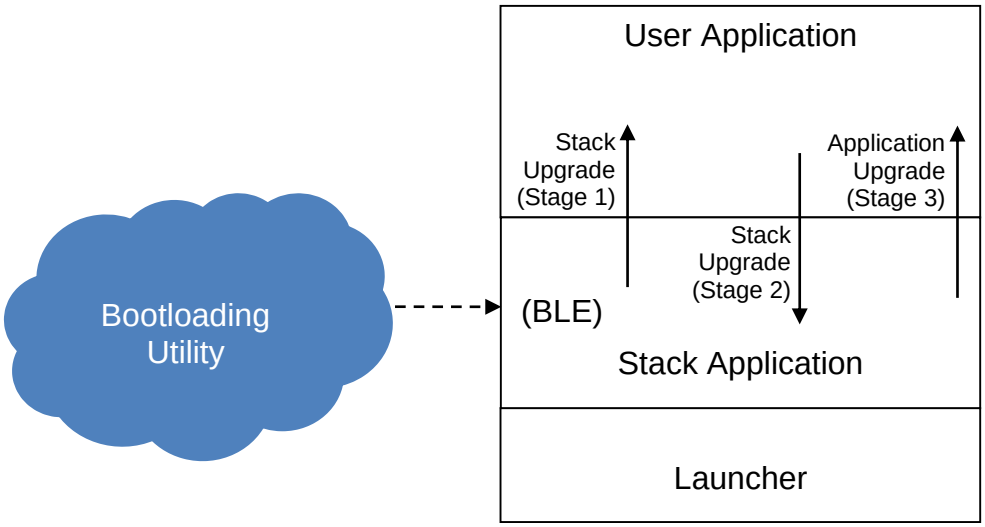
How OTA upgrade works

This architecture provides two options for OTA firmware upgrades. These options are available in the CySmart PC tool and the mobile app.

1) **Application Upgrade** – When this option is used, only the User Application image is upgraded.

   The Application upgrade happens in a single stage. To enter the OTA upgrade mode, the firmware passes control to the Stack Application project image. The Bootloader in the Stack Application receives the new User Application image and writes it to the corresponding region of the flash (see Figure 3).

Figure 3. Upgradable Stack OTA Bootload Process



2) **Stack Upgrade** – When this option is used both project images, Stack Application image and User Application image, are upgraded. In this case, upgrade happens in three stages:

   Stage 1 – The firmware passes control to the Stack Application, which receives the new Stack Application image and writes it to the User Application image region in the flash memory, overwriting the existing User Application image.

   Stage 2 – After the download is complete, a software reset is initiated by the Stack Application and the control passes to the Launcher. It detects the new Stack Application image located in the User Application region, and copies it to the Stack Application region. At this point, the User Application image is invalid. Therefore, User Application upgrade has to be performed as well.

   Stage 3 – Now, the User Application image is updated in the same way as explained in the previous option.

Note*:* The Upgradable Stack Bootloader option can be used only with Cypress's 256 KB BLE parts.

Initial Kit Setup

The BLE Pioneer Kit connects to the PC over a USB interface. The kit enumerates as a composite device and three separate devices appear under the Device Manager window in the Windows operating system. Follow these steps to get started:

1. If you have not already installed the BLE Pioneer Kit Software, do that first.
2. Before power-on, verify that the PSoC 4 BLE module CY8CKIT-143A (red color) is plugged into the baseboard on your kit.
3. Plug in your BLE Pioneer Kit to your PC using the provided USB cable. You will see the Windows driver-enumeration process begin.
4. Wait for the driver installation to complete as shown in Figure 4 and Figure 5. Click on **Skip obtaining driver software from Windows Update** to speed up the process, especially if you do not have an Internet connection. The required drivers are already installed on your computer with the kit software and, therefore, do not need to be downloaded via the Windows Update.
5. Plug in the CySmart BLE dongle and follow the instruction given in Step 4.

Figure 4. BLE Pioneer Kit Driver Installation in Progress

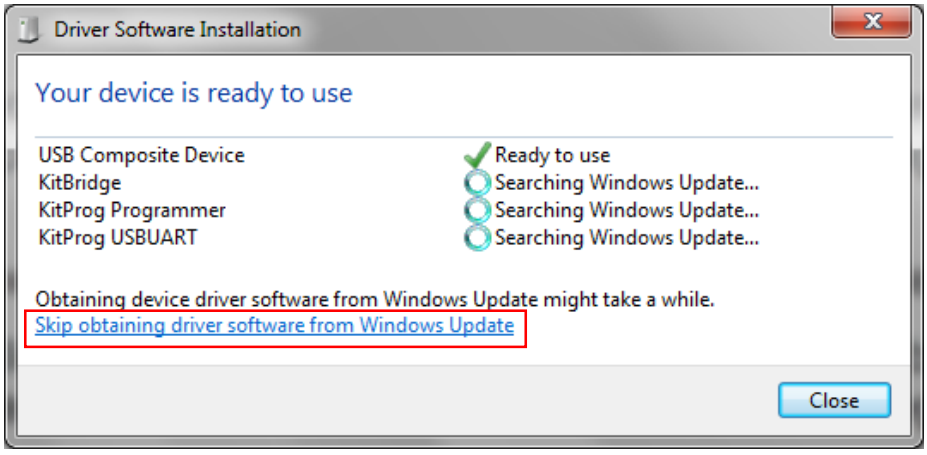
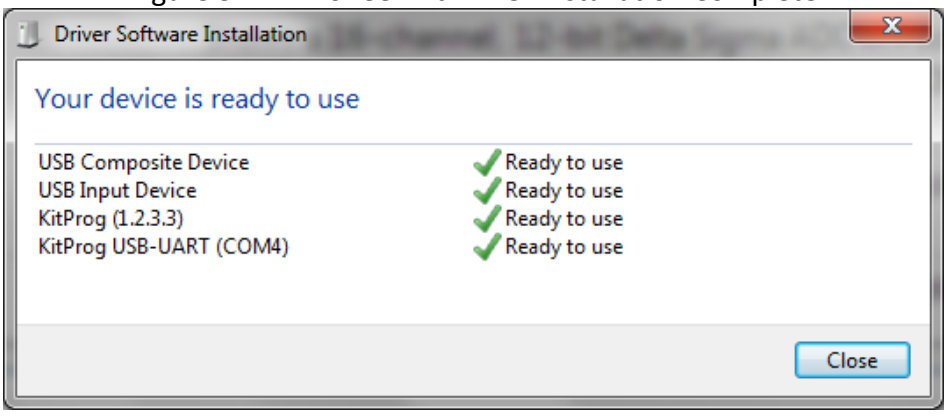


Figure 5. BLE Pioneer Kit Driver Installation Complete



Theory

The Find Me Project

This example uses Find Me Profile of the BLE component in the PSoC Creator. The Figure 6 shows the block diagram the project. The Pioneer kit acts as a Find Me Target. It utilizes the Find Me Profile with one instance of Immediate Alert Service to display the alerts. A Find Me Target operates with other devices (such as a mobile phone), which implement the Find Me Locator Profile. More details about the BLE Find Me Profile can be found on Bluetooth developer portal (Find Me Profile details).

The example project consists of the following Creator components:

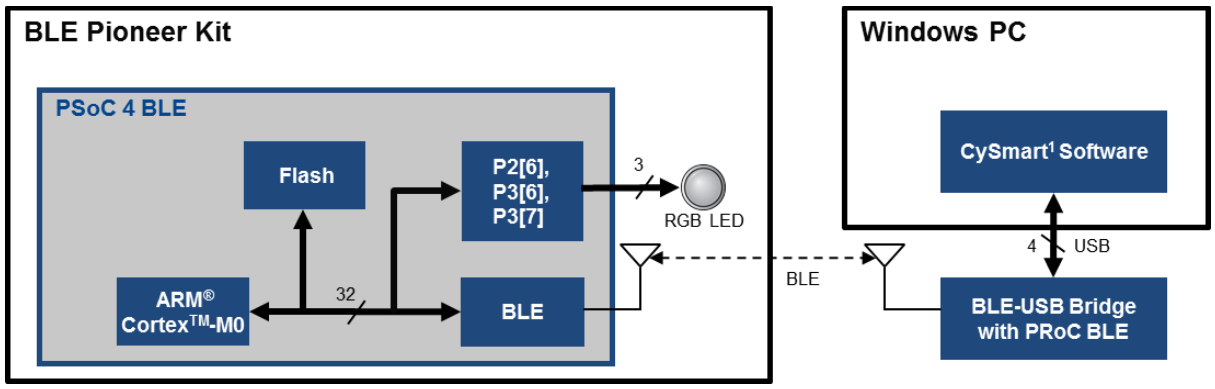
1. BLE
2. CY_PIN

This project uses the RGB LED on the kit to convey the status of operation as follows:

3. The green LED (port 3 pin 6) is used to indicate the advertising state of BLE.
4. The red LED (port 2 pin 6) is used to indicate the BLE disconnection state.
5. The blue LED (port 3 pin 7) is used to indicate alerts received.

This Find Me example project will be considered as the User Application project for this Lab. We will add OTA upgrade capability to this project.

Figure 6. Lab block diagram



Adding OTA Upgrade functionality to an existing project

Following steps are required to add OTA upgrade functionality to an existing project (in this case, the Find Me example project):

1) Add the **BLE_OTA_UpgradableStackExample_Launcher** example project to PSoC Creator Workspace.
2) Add the **BLE_OTA_UpgradableStackExample_Stack** example project to PSoC Creator Workspace.
    a. Specify the paths to the Launcher project's HEX and ELF files, in the Bootloadable component in this project.
3) Modify User Application project:
    a. Configure BLE component of the User Application project in *Profile Only* mode, and provide a reference to the BLE stack symbol file.
    b. Add an Input pin to receive user input from SW2 switch on the Pioneer kit. This switch is used to put the device into bootloader mode.
    c. Include necessary files in the project to support OTA functionality.

    These files contain APIs that are required for the OTA upgrade implementation. They contain bootloader RAM initialization functions, metadata initialization and verification functions, flash data write functions, values of various constants, and macro definitions. A list of the required files is given below:

    Header Files:

    - OTAMandatory.h (Contains declarations of the functions required for OTA)
    - main.h (Includes other header files)

    Source Files:

    - OTAMandatory.c (Contains definitions of the functions required for OTA)

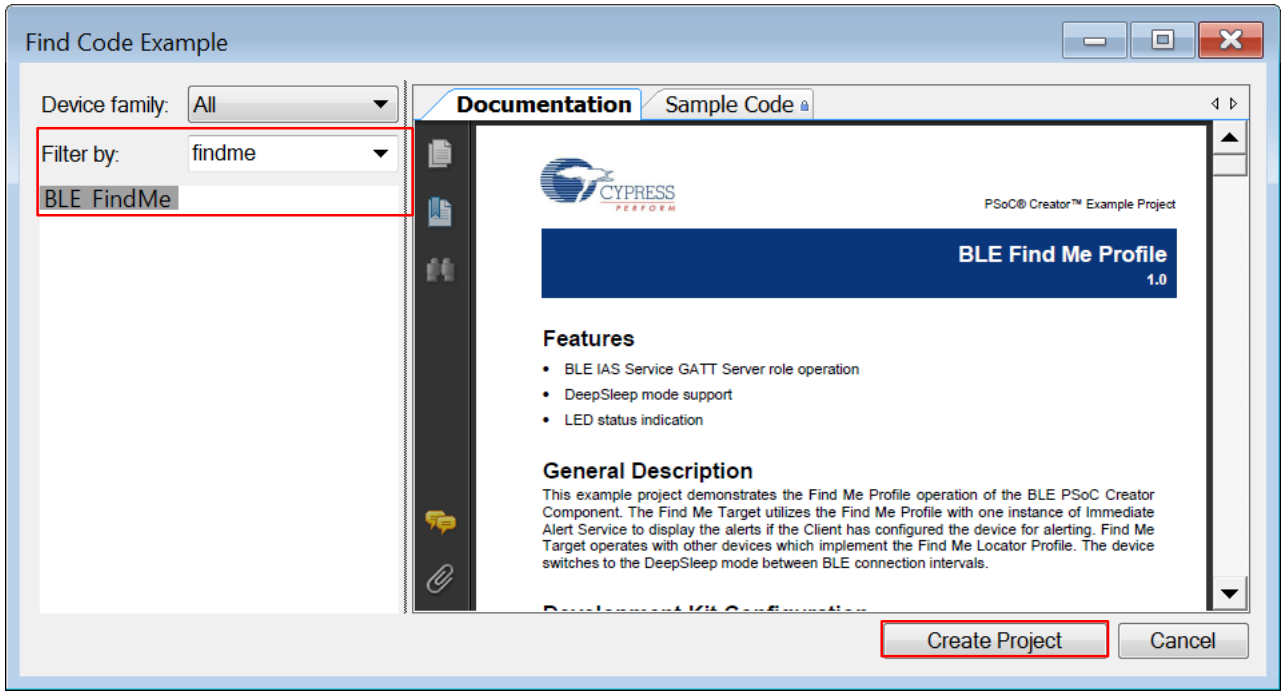
    d. Add a piece of code to switch to the Stack Application, after a key press by the user, to initiate firmware upgrade.
    e. Use a custom linker script provided with this lab to enable RAM sharing, and checksum exclusion. Note: These linker scripts do not need any modifications and can be used in any user project. Refer to the section on *Custom Linker Script* in the example project datasheet (BLE_OTA_UpgradableStackExample_Stack.pdf) for more details.

Following procedure shows how to carry out these steps to add OTA upgrade functionality to the BLE_FindMe example project. This being an advanced lab, certain trivial steps are not elaborated in the following procedure. You may look at Appendix A to understand the steps required for performing some basic operations (such as updating components, changing device part numbers, etc.) in the PSoC Creator IDE.

Procedure

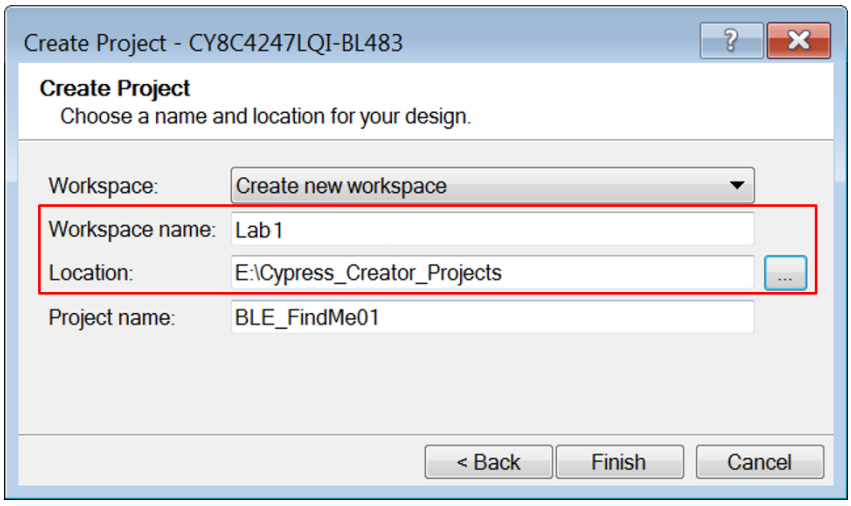
1. Open PSoC Creator. It is located in the **All Programs -> Cypress -> PSoC Creator 3.3** folder in the Windows start menu.
2. Create a new example project by using **File -> Code Example**. For this lab, we will use the **BLE_FindMe** code example. See Figure 7 below. Click **Create Project** to proceed.

Figure 7. Select BLE_FindMe From Code Examples



3. In the **Create Project** dialog box, provide a *Workspace Name* of Lab 1, and select a desired *Location* for the workspace. Click **Finish** to create the example project and associated workspace. See Figure 8 below.

Figure 8. Creating a Project



4. The code example uses a different PSoC than the one you will be using so you must change the device part number to **CY8C4248LQI-BL583**. Refer to the instructions provided in the Appendix section A.2 Selecting Another Device.

5. **Build** the **BLE_FindMe01** project. To do this, **Right Click** on the project name in the *Workspace Explorer* and click **Build BLE_FindMe01** option. The project should build without any errors.

   At this point, we have created the User Application project. In the next few steps, you will see how to add OTA upgrade functionality to this project.

6. Add the **BLE_OTA_UpgradableStackExample_Launcher** code example to the **Lab 1** workspace. Refer to the instruction provided in Appendix section A.1 Adding an Example Project to an Existing Workspace.

7. If the BLE_OTA_UpgradableStackExample_Launcher01 project is not the active project, set it as the active project. To do this, right-click the project in *Workspace Explorer* and select **Set As Active Project**.

8. Change the device part number to **CY8C4248LQI-BL583**.

9. **Build** the BLE_OTA_UpgradableStackExample_Launcher01 project. The project should build without any errors.

10. Now, add the **BLE_OTA_UpgradableStackExample_Stack** code example to the **Lab 1** workspace.

11. If the BLE_OTA_UpgradableStackExample_Stack01 project is not the active project, set it as the active project. To do this, right-click the project in *Workspace Explorer* and select **Set As Active Project**.

12. Change the device part number to **CY8C4248LQI-BL583**.

13. Specify the paths to the Launcher project's HEX and ELF files in the Stack project's bootloadable component.

    a. In the *Workspace Explorer* expand the BLE_OTA_UpgradableStackExample_Stack01 project and double click on **TopDesign.cysch**. In the design schematic window double-click on the **Bootloadable** component.

    b. Navigate to the **Dependencies** tab and link the **Bootloader HEX file** to the *BLE_OTA_UpgradableStackExample_Launcher01.hex* file as shown in Figure 9. This file is located at:

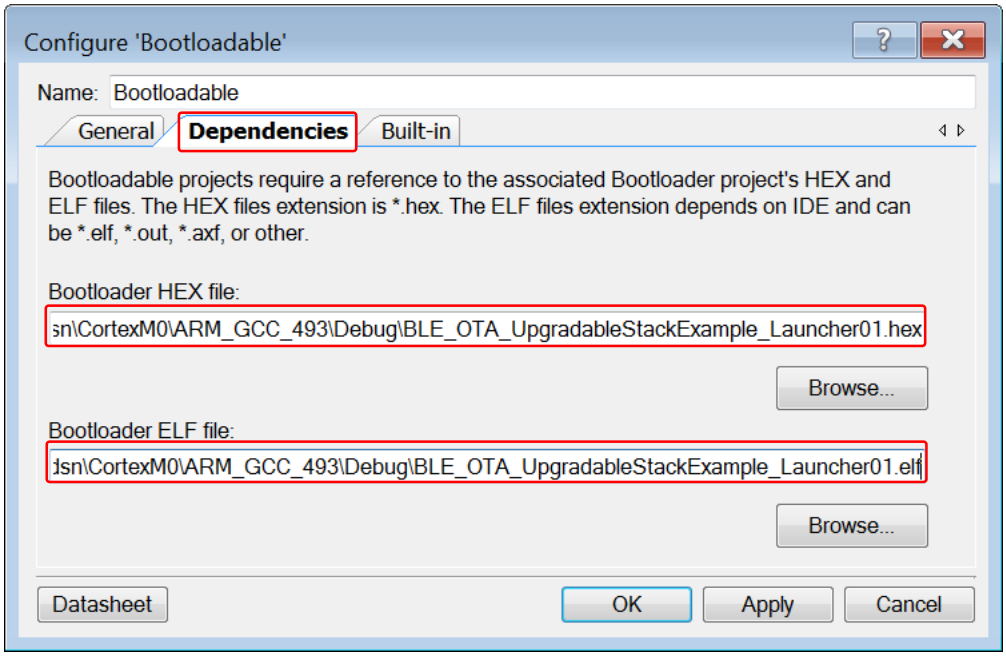
       ..*BLE_OTA_UpgradableStackExample_Launcher01.cydsn**CortexM0*\<*compiler−version*>\<*build configuration*>\.

       > Note: You can navigate to the path using the "Browse" button rather than typing in the entire path. The complete path will be shown but once you close the dialog, the full paths will be changed to the appropriate relative paths.

       Click **OK** to close the Bootloadable component configuration dialog.

       After you have selected the HEX file, the corresponding ELF file gets automatically selected for you.

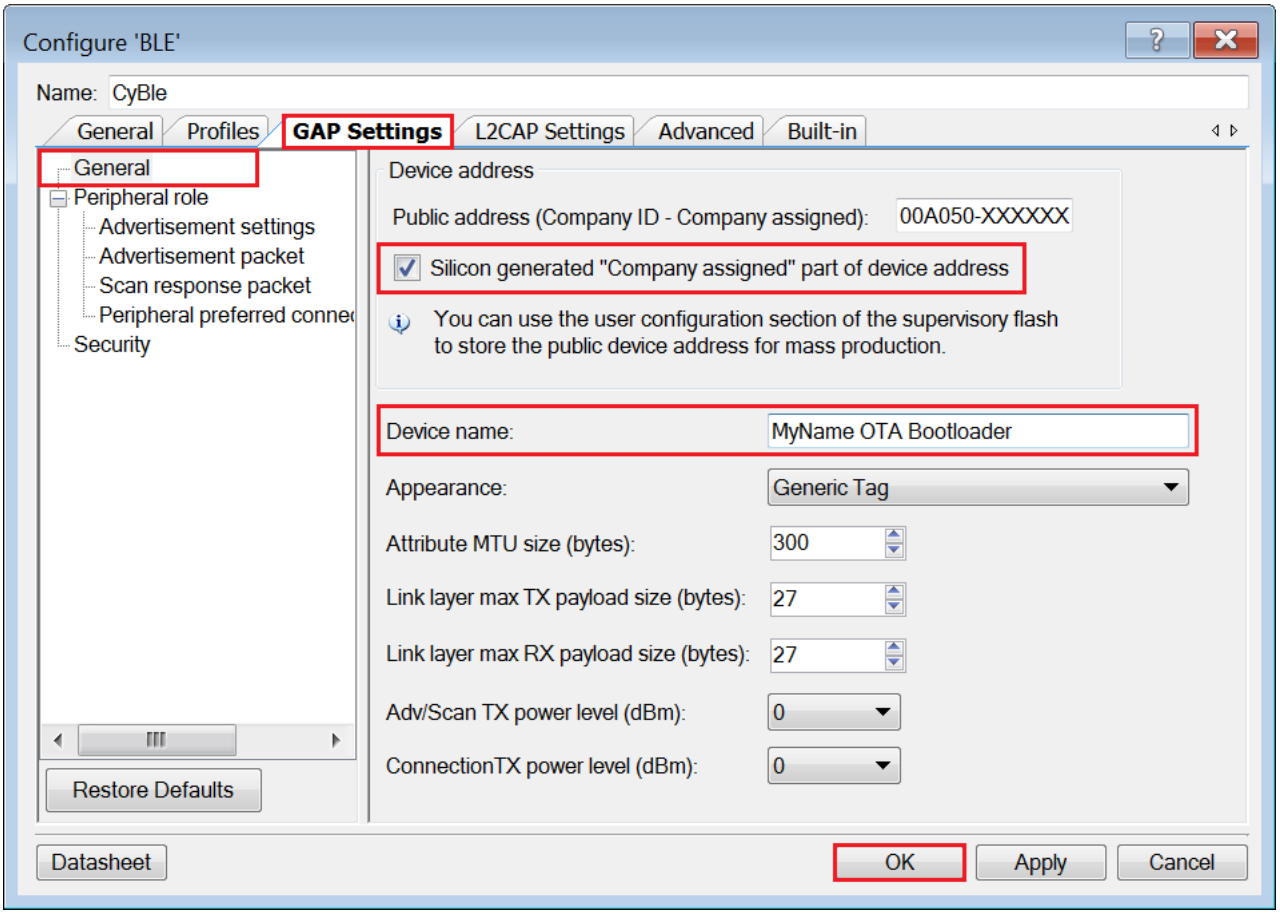
Figure 9. Bootloadable Component Configuration



14. Change the device name for the OTA bootloader to give a unique name for your kit.
    a. In the design schematic window double-click on the **BLE** component.
    b. Navigate to the **GAP settings** tab; and click on the **General** section.
    c. Select **Silicon generated "Company assigned" part of device address**. This is required to give unique Bluetooth device address to each device.
    d. Modify the **Device name** by adding your name as a prefix to *OTA Bootloader* string as shown in Figure 10.

       Click **OK**.

Figure 10. Modify the Device Name of the OTA Bootloader Service



15. **Build** the BLE_OTA_UpgradableStackExample_Stack01 project. The project should build without any errors.

    Now, we need to add two components to the **BLE_FindMe01** project to implement the OTA upgrade functionality.

    - Input Pin (For SW2: A user switch to enter bootloading mode)
    - Bootloadable (Required to make the BLE_FindMe01 project bootloadable)

16. From the *Workspace Explorer*, right click on the BLE_FindMe01 project and set it as the active project. Expand the project and **double-click** on **TopDesign.cysch**. This opens the schematic window.

17. Adding Input Pin:

    a. From the **Component Catalog** box search for *input pin,* select **Digital Input Pin**.

    b. Drag and drop this pin on to the schematic. See Figure 11.

    c. In the design schematic window double-click on the **Pin_1**. This opens the **Configure 'cy_pins'** dialog box. As shown in Figure 12, type the pin **Name** as **SW2**, set the *Drive* mode to **Resistive pull**

up, uncheck **HW connection**, and click **OK.** Note: This name change is required because some portion of the OTA upgrade code refers to this component by this name.

Figure 11. Adding Input Pin

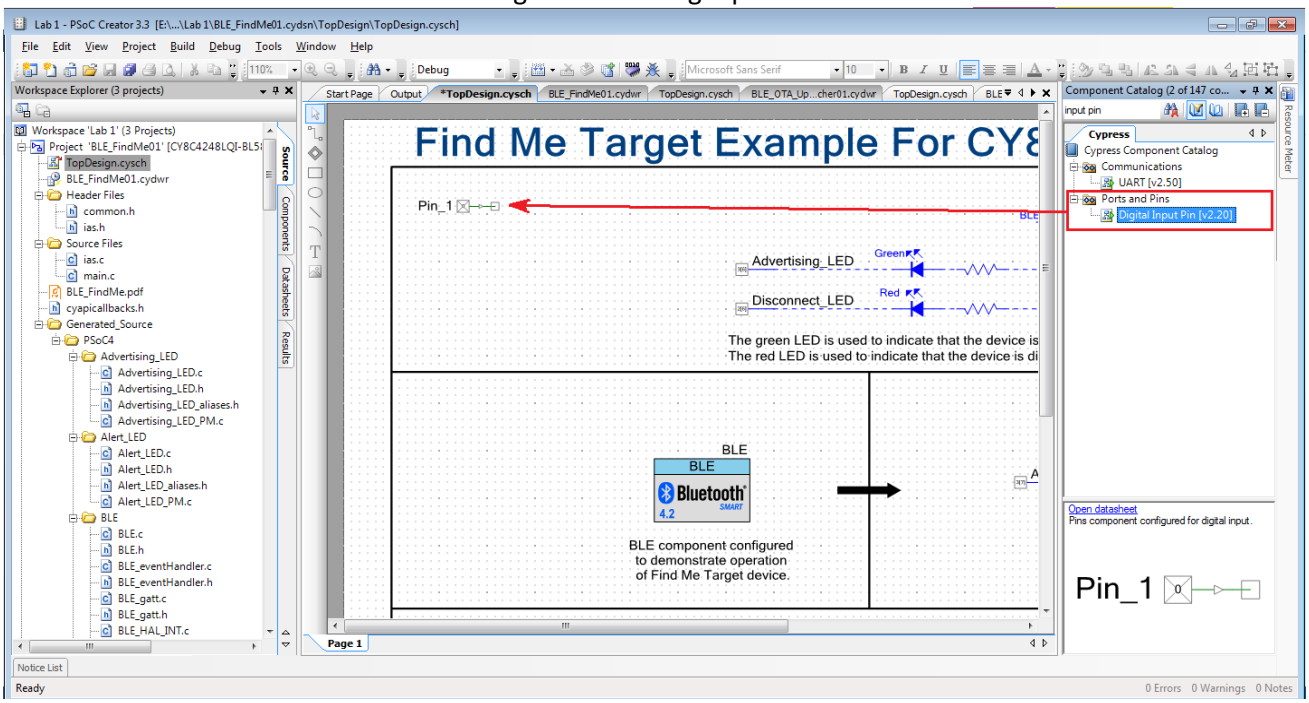
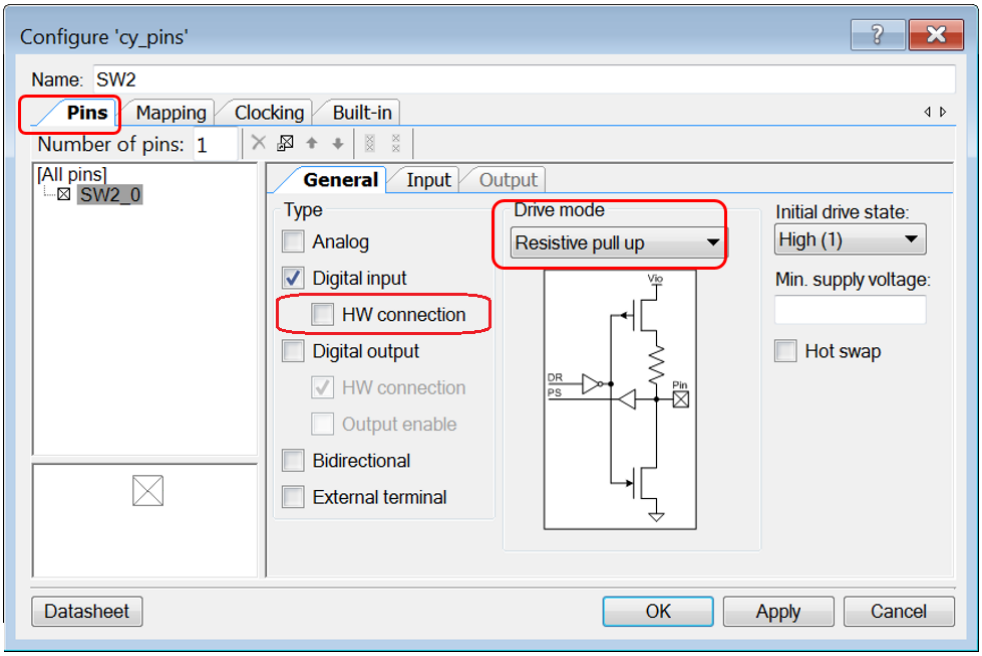


Figure 12. Configuring Pin Settings



18. Adding Bootloadable:

    a. From the **Component Catalog** box search for *Bootloadable*, and select a **Bootloadable** component.

    b. Drag and drop this component on to the schematic.

    c. In the design schematic window double-click on the **Bootloadable_1**. This opens the **Configure 'Bootloadable'** dialog box.

    d. In this, type the component **Name** as **Bootloadable**. Note: This name change is required because some portion of the OTA upgrade code refers to the component by this name.

    e. Navigate to the **Dependencies** tab and link **Bootloader HEX file** to the *BLE_OTA_UpgradableStackExample_Stack01.hex* file, as shown in Figure 13. This file is located at: *..\BLE_OTA_UpgradableStackExample_Stack01.cydsn\CortexM0\<compiler-version>\<build configuration>*

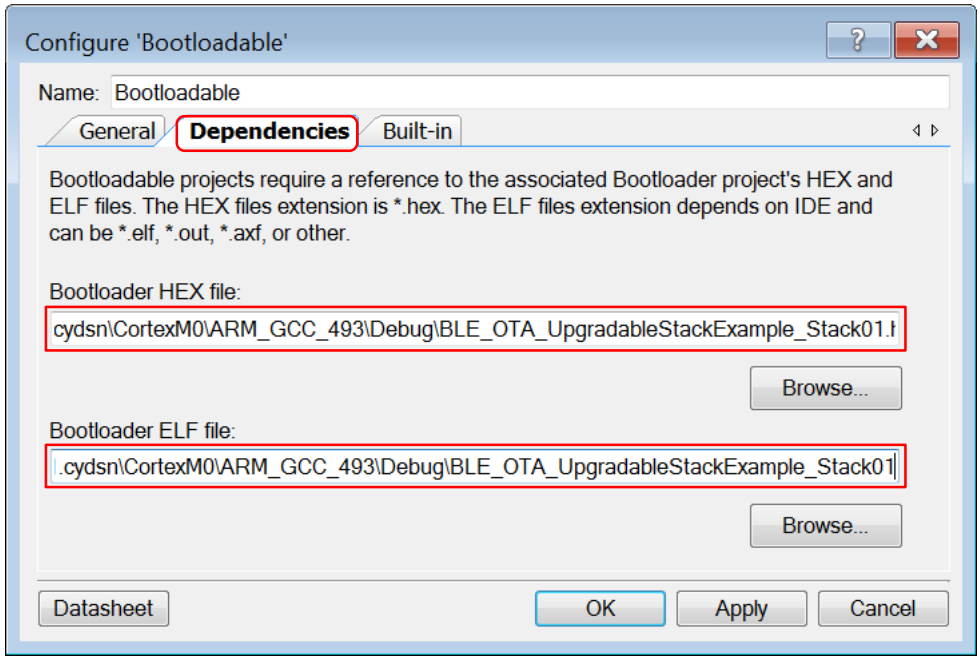
        Note: You can navigate to the path using the "Browse" button rather than typing in the entire path. The complete path will be shown but once you close the dialog, the full paths will be changed to the appropriate relative paths.

        Note that the Bootloadable for the application project points to the bootloader hex file in the stack project while the Bootloadable in the stack project pointed to the bootloader hex file in the launcher project. This is required for proper bootloading operations.

    Click **OK** to close the Bootloadable component configuration dialog.

    After you have selected the HEX file, the corresponding ELF file gets automatically selected for you.

Figure 13. Bootloadable Component Configuration



19. Specify the path to the ***CyBle.cycsa*** file of the BLE_OTA_UpgradableStackExample_Stack01 project in the **BLE** component of **BLE_FindMe01** project.

    a. In the design schematic window double-click on the **BLE** component.

    b. Navigate to the **General** tab; in the **Over-The-Air bootloading with code sharing** section, select the **Profile only** option.

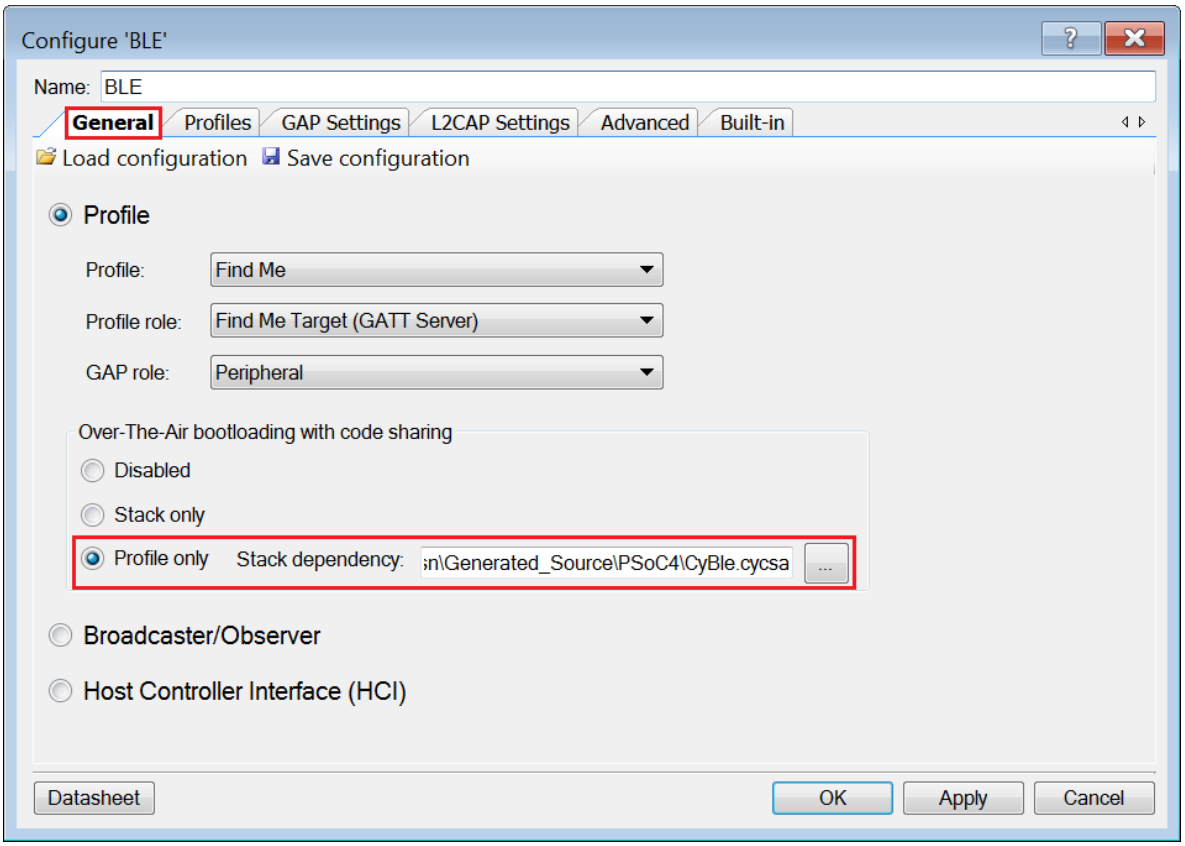
    Note: The 'Profile only' option makes the component include only profile-specific code (excluding Stack code) in the project. When this option is selected, we also need to provide a Stack dependency file from the Stack Application project. In the case of the 'Stack only' option, the component includes the Stack code, and the code required for the Bootloader service, and excludes all other profiles.

    c. Select the *CyBle.cycsa* file in the **Stack dependency** field as shown in Figure 14. This file is located in *..\BLE_OTA_UpgradableStackExample_Stack01.cydsn\Generated_Source\PSoC4* directory under the project directory.

    Note: You can navigate to the path using the "Browse" button rather than typing in the entire path. The complete path will be shown but once you close the dialog, the full paths will be changed to the appropriate relative path.

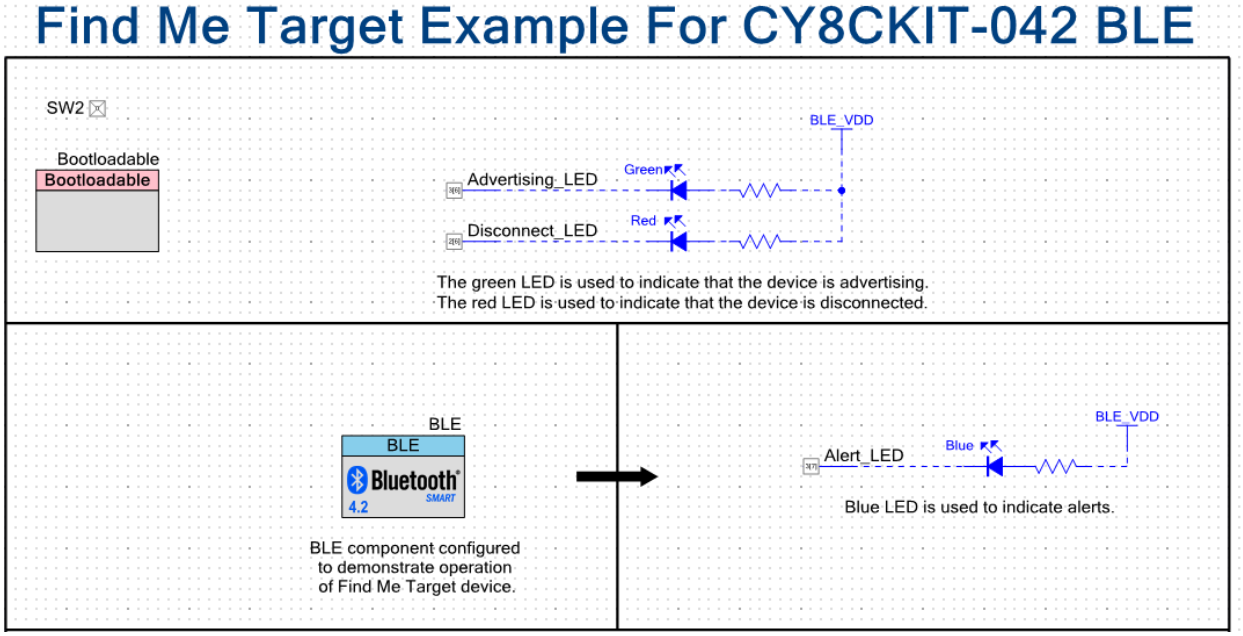
    Click **OK**.

Figure 14. BLE Component Configuration Showing Profile Only Option Selection



20. At the end of this, the schematic of the **BLE_FindMe01** project should look as shown in Figure 15.

Figure 15. After Adding OTA Related Components



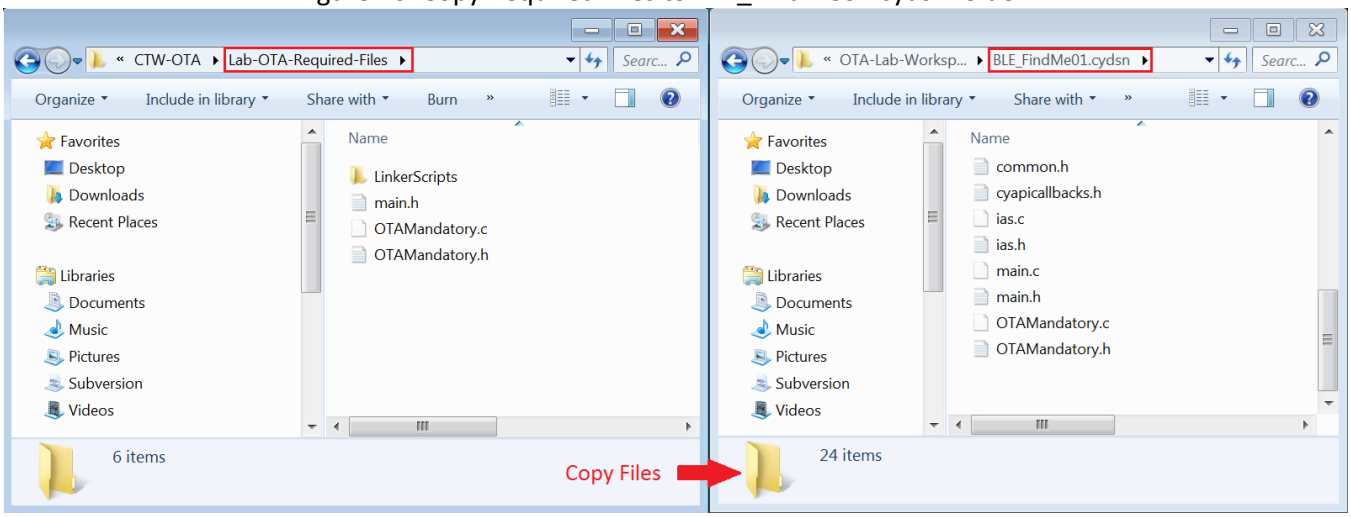
Now, double-click on **BLE_FindMe01.cydwr** and go to the Pins tab. Connect the SW2 to P2[7] port as shown in Table 1.

Table 1. Pin Mapping for **BLE_FindMe01**

	Name	Port	Pin	Lock
	Advertising_LED	P3[6]	53	☑
	Alert_LED	P3[7]	54	☑
	Disconnect_LED	P2[6]	43	☑
	SW2	P2[7]	44	☑

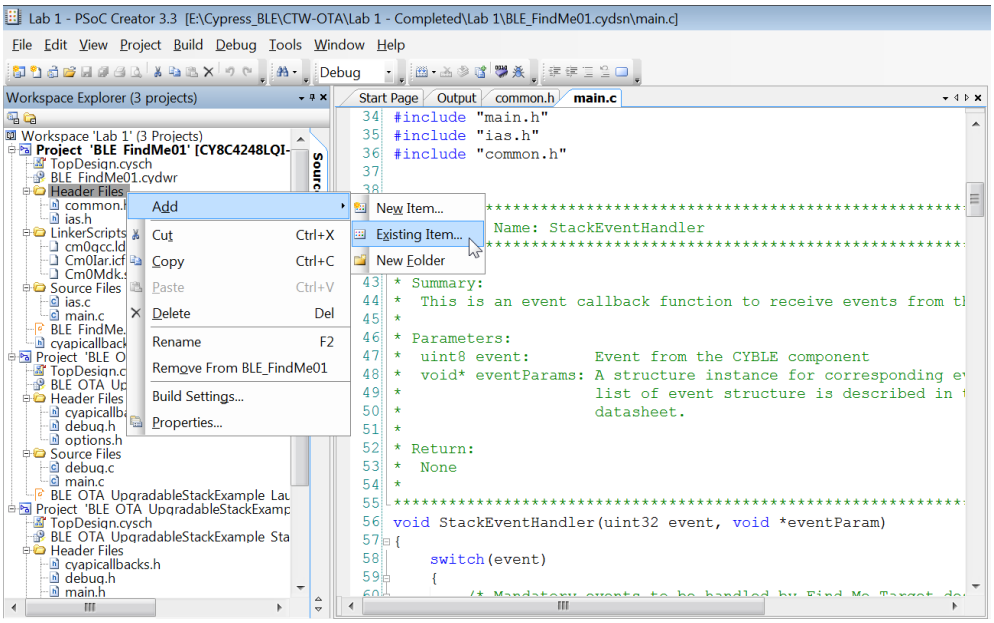
21. **Extract** the archive **Lab 1 - OTA-required-files.zip**, and **Copy** all files and folders to the **BLE_FindMe01.cydsn** project directory using **Windows Explorer** as shown in Figure 16.

Figure 16. Copy Required Files to BLE_FindMe01.cydsn folder



22. Add these newly copied files to the **BLE_FindMe01** project workspace. These files implement a part of the OTA upgrade functionality.
    a. To **add header** files, right-click the **Header Files** folder in *Workspace Explorer* and click **Add -> Existing Item** (See Figure 17). Browse and select the following files and click **Open**.
        - OTAMandatory.h (Contains declarations of the functions required for OTA)
        - main.h (Includes other header files)

Figure 17. Add Existing Files to The BLE_FindMe01 Project.



b. To **add source** files, right-click the **Source Files** folder in *Workspace Explorer* and click **Add -> Existing Item**. Browse and select the following file and click **Open**.

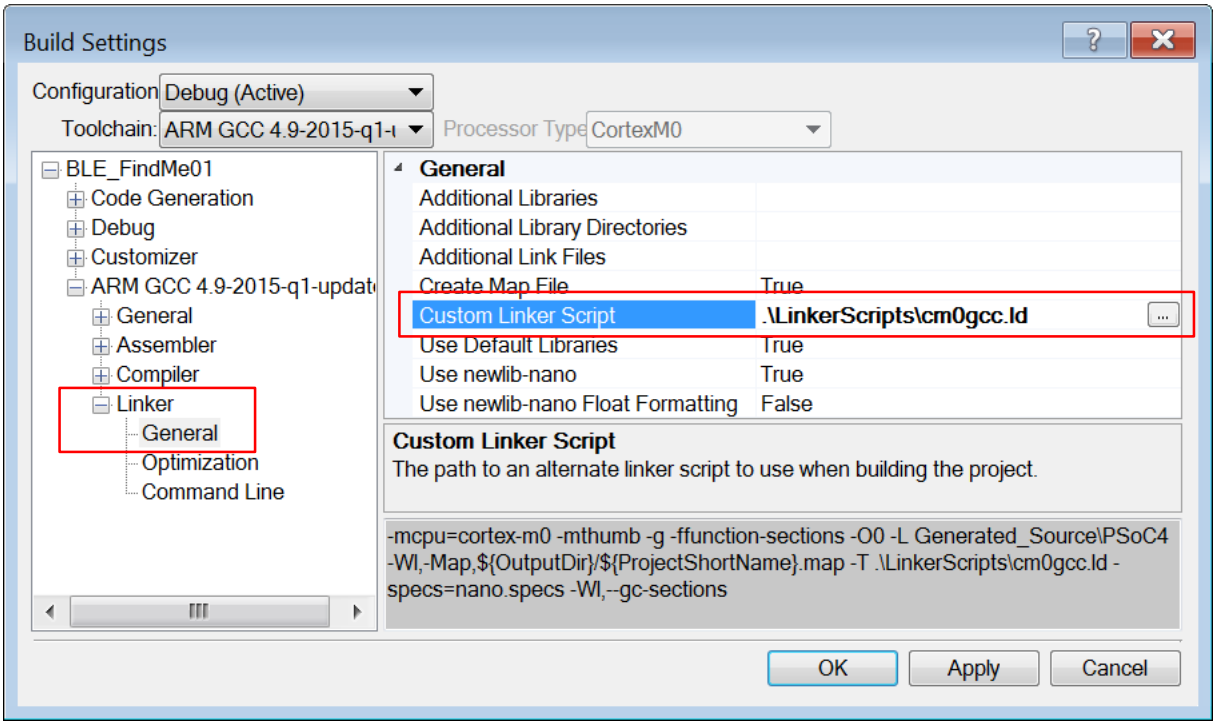
- OTAMandatory.c (Contains definitions of the functions required for OTA)

23. Create a new folder named "LinkerScripts" under the **BLE_FindMe01** project in the *Workspace Explorer*. To do this, in the *Workspace Explorer*, right click on the project **BLE_FindMe01** and select **Add -> New Folder**. Provide the name as "LinkerScripts".
24. Add all files in the **LinkerScripts** directory here. To do this, right-click the **LinkerScripts** folder in *Workspace Explorer,* click **Add -> Existing Item** and Browse to the LinkerScripts folder. Select file type as "All Files (*.*)", select all the files, and click **Open**.
25. Make sure that the settings listed in Table 2 are applied to the build settings for BLE_FindMe01. These changes tell the linker to use the new custom linker script. To change build settings, click **Project -> BuildSettings** and then select **BLE_FindMe01 -> ARM GCC 4.9-2015-q1-update -> Linker -> General** on the tree view as shown in Figure 18.

Table 2. Build setting changes

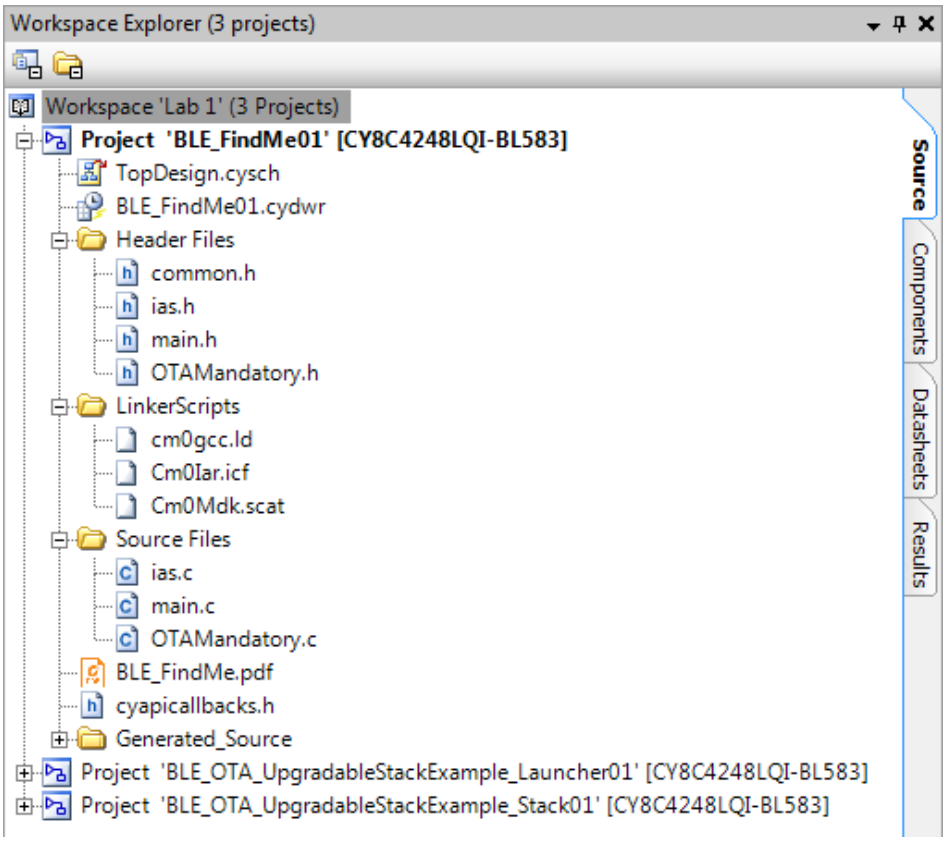
Field	Value
Custom Linker Script	.\LinkerScripts\cm0gcc.ld

Figure 18. Build Settings Dialog Showing Linker Settings



At this point your *Workspace Explorer* should look like as shown in Figure 19.

Figure 19. Complete Workspace Explorer View for This Lab



26. Add OTA upgrade specific code to the project:

    Additional code must be added to main.c of the BLE_FindMe01 project to enable the bootloader or OTA functionality.

    a. Add code shown in the box Code 1 in the file **main.c** after `#include <project.h>` line.

    Code 1. Include 'main.h' in 'main.c' File

```
#include "main.h"
```

    b. Add the code given in box Code 2 after `CyGlobalIntEnable;` in the `main()` function. The `AfterImageUpdate()` function checks if the Project Image has been updated and is running for the first time. If it is running for the first time and the Bonding requirement option is set to Bonding in the BLE component, it verifies the bonding data and erases it if it is not valid. It also sets up the update detection flag in the unused metadata area.

Code 2. SRAM Initialization and Bonding Data Verification Code

```
/* For GCC compiler use separate API to initialize BLE Stack SRAM.
 * This is needed for code sharing.
 */
#if !defined(__ARMCC_VERSION)
    InitializeBootloaderSRAM();
#endif


/* Checks if Self Project Image is updated and Runs for the First time
*/
AfterImageUpdate();
```

Your **main.c** source code should look like as shown in Figure 20.

Figure 20. Modified source code after adding the code provided in Code 2 box

```
int main()
{
    CYBLE_API_RESULT_T apiResult;

    CyGlobalIntEnable;

    /* For GCC compiler use separate API to initialize BLE Stack SRAM.
     * This is needed for code sharing.
     */
    #if !defined(__ARMCC_VERSION)
        InitializeBootloaderSRAM();
    #endif

    /* Checks if Self Project Image is updated and Runs for the First time */
    AfterImageUpdate();

    apiResult = CyBle_Start(StackEventHandler);
```

c. Add the code given in box Code 3, at the beginning of the `for(;;){}` loop in the `main()` function. This portion of code puts User Application into OTA upgrade mode after pressing SW2 for 0.5 seconds.

Code 3. Code Required to Switch to Stack Application for the OTA Upgrade

```
/* If key press event was detected - debounce it and switch to
bootloader emulator mode */
if (SW2_Read() == 0u)
{
    CyDelay(500u);
    if (SW2_Read() == 0u)
    {
        CyDelay(500u);
        while (SW2_Read() == 0u)
        {
            /* Wait for button to be released */
        }

        //Switch to the Stack project, which enables OTA service
        Bootloadable_SetActiveApplication(0);
        Bootloadable_Load();
        CySoftwareReset();
    }
}
```

Your **main.c** source code should look like as shown in Figure 21.

Figure 21. Modified source code after adding the code provided in Code 3 box

```
for(;;)
{
    /* If key press event was detected - debounce it and switch
    to bootloader emulator mode */
    if (SW2_Read() == 0u)
    {
        CyDelay(500u);
        if (SW2_Read() == 0u)
        {
            CyDelay(500u);
            while (SW2_Read() == 0u)
                {
                    /* Wait for button to be released */
                }

            //Switch to the Stack project, which enables OTA service
            Bootloadable_SetActiveApplication(0);
            Bootloadable_Load();
            CySoftwareReset();
        }
    }

    static uint8 toggleTimeout = 0;
    CYBLE_BLESS_STATE_T blessState;
    uint8 intrStatus;
```

27. Build the **BLE_FindMe01** project. The project should build without any errors.

Testing

1. Plug the BLE Dongle (included with the BLE Pioneer Kit) in your computer's USB port.
2. Connect USB cable to the Pioneer kit and plug the USB cable in your computer's USB port.
3. Make sure **BLE_FindMe01** project is the active project. To do this, right-click the project in *Workspace Explorer* and select **Set As Active Project**.
4. Select **BLE_FindMe01** project from the *Workspace Explorer*, and click **Debug> Program** or press **Ctrl+F5**. This programs the entire project image (Launcher + Stack + User-application) to the device.
5. Once programming finishes, the RGB LED on the kit cycles through the various colors and towards the end steadily lights up in **Green**.

   The RGB LED on the Pioneer kit exhibits different states of the OTA functionality. For ease of understanding, the LEDs and their usage by the various projects is summarized in Table 3 below.

Table 3. Use of LEDs in Various Projects

Project	LED color	LED Behavior
Launcher	Blue	Turns on and remains on while the Launcher image is executing
Stack Application	Red	Turns on and remains on while the Stack image is executing
User Application (Find Me)	Green	Turns on to indicate that the device is advertising. It remains on as long as device is in advertising mode.
	Red	Turns on after advertising timeout expires and device enters into power saving mode, or after the device is disconnected from the client.
	Blue	Turns on to indicate received Alert from the client.

In order to get a clear picture of the booting process you need to observe the color of the LEDs. In case you have missed it, repeat **Step 5** to reprogram the device and observe the following points:

As soon as device programming finishes:

a. **Blue** LED turns on momentarily. This indicates that the Launcher image is active.

b. Default application to launch is the Stack project image. As soon as Stack project image starts executing, the **Red** LED turns on momentarily. The Stack project image checks the application project image, validates its checksum, sets it as the default image to launch, and initiates a software reset.

c. After software reset, the Launcher starts, and **Blue** LED turns on momentarily. Now, the Launcher starts the BLE_FindMe01 project image. The Find me example starts advertising and turns on the **Green** LED to indicate advertising state. After the advertising timeout is expired (i.e. after 30 seconds), advertising stops, the **RED** LED turns ON, and the device enters into hibernate mode.

If you manually reset the device by pressing the RESET button on Pioneer kit, code execution will proceed in the following manner:

a. **The Blue** LED turns on momentarily. This indicates that the Launcher image is active.

b. The Launcher starts the User Application image. The Find me example starts advertising and turns the **Green** LED ON to indicate the advertising state. After an advertising timeout occurs, advertising stops, the **RED** LED turns ON, and device enters into hibernate mode.

Upgrading the User Application Over-The-Air:

6. Now, you need to modify the BLE_FindMe01 project's code and update it using OTA upgrade.

   Add the code shown in box Code 4 to blink RGB LED in a WHITE color at the beginning of the BLE_FindMe01 application. This serves as a simple visual check to demonstrate that the User Application has been updated. Add the code provided in box Code 4, just before `CyGlobalIntEnable;` in **main.c** file.

Code 4. Code Required to Flash RGB LED in WHITE Color

```
/* Blink all three LEDs in RGB LED (to produce WHITE color).
 * This  serves as a simple  visual check to demonstrate that
 * the User Application has been updated.
 */
Advertising_LED_Write(LED_ON);
Disconnect_LED_Write(LED_ON);
Alert_LED_Write(LED_ON);
CyDelay(1000u);
Advertising_LED_Write(LED_OFF);
Disconnect_LED_Write(LED_OFF);
Alert_LED_Write(LED_OFF);
CyDelay(1000u);
```

Your main.c source code should look like as shown in Figure 22.

Figure 22: Modified source code after adding the code provided in Code 4 box

```
int main()
{
    CYBLE_API_RESULT_T apiResult;

    /* Blink all three LEDs in RGB LED (to produce WHITE color).
     * This  serves as a simple  visual check to demonstrate that
     * the User Application has been updated.
     */
    Advertising_LED_Write(LED_ON);
    Disconnect_LED_Write(LED_ON);
    Alert_LED_Write(LED_ON);
    CyDelay(1000u);
    Advertising_LED_Write(LED_OFF);
    Disconnect_LED_Write(LED_OFF);
    Alert_LED_Write(LED_OFF);
    CyDelay(1000u);


    CyGlobalIntEnable;
```

7. Build **BLE_FindMe01** project. It should build without any errors. Do NOT program the project onto the board – we will instead update it using OTA bootloading.
8. Connect the **CySmart Dongle** to the Laptop/PC and start the **CySmart** software tool.
9. From the **Select BLE Dongle Target** dialog box, select **CySmart BLE 4.2 USB Dongle** and click on **Connect**. See Figure 23.

   Note: Instead of **CySmart BLE 4.2 USB Dongle** under **Supported targets**, if you see **Unsupported Device** under **Unsupported targets** as shown in Figure 24, follow the procedure given in Appendix section A.3 Updating BLE Dongle Firmware to upgrade the Dongle firmware to the firmware compatible with CySmart 1.2.

Figure 23. Connecting to BLE Dongle

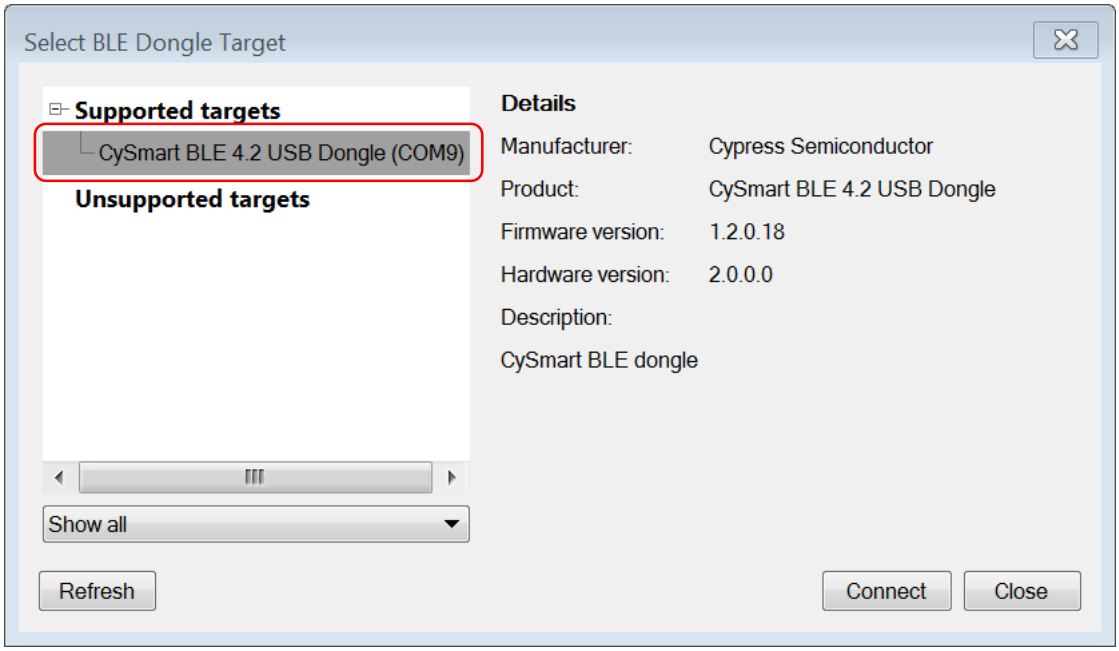
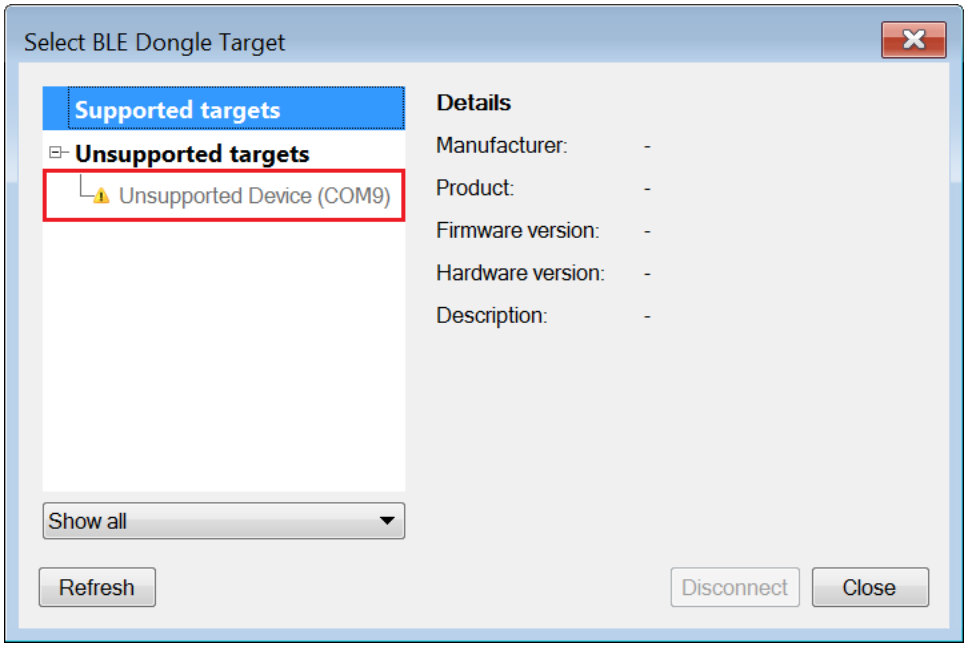


Figure 24. Unsupported Device Issue

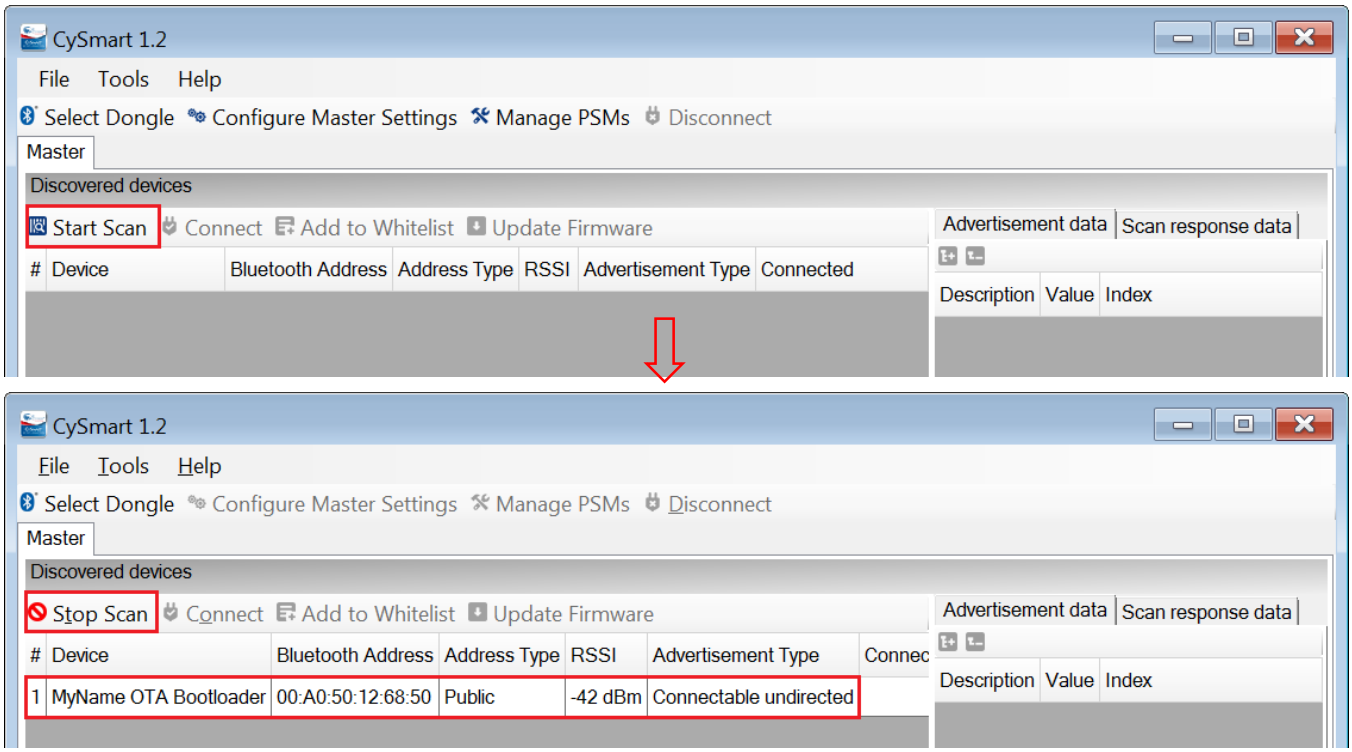


10. Reset the PSoC device by pressing the **Reset** switch on the Pioneer Kit board. When you see that the **Green** LED is ON (i.e. the Find Me project image is active and advertising), press switch **SW2** on the pioneer kit for **at least one second** and release it. The **Red** LED is turned ON to indicate that the device has switched to the Stack Application image and it is now in OTA bootloader mode.

Note: the BLE_FindMe01 example puts the device into hibernate mode after the advertisement timeout occurs. In hibernate mode, the device does not respond to any user input. To ensure the Find Me example is active and running, you need to reset the device and wait for Green LED to turn ON.

11. In CySmart tool, press the button **Start Scan**. The **OTA bootloader** becomes visible in the available devices list as shown in Figure 25.

Figure 25. CySmart Tool Snapshot



12. Click **Stop Scan**, select **OTA bootloader** from the list of devices, and click on the **Update Firmware** Button.

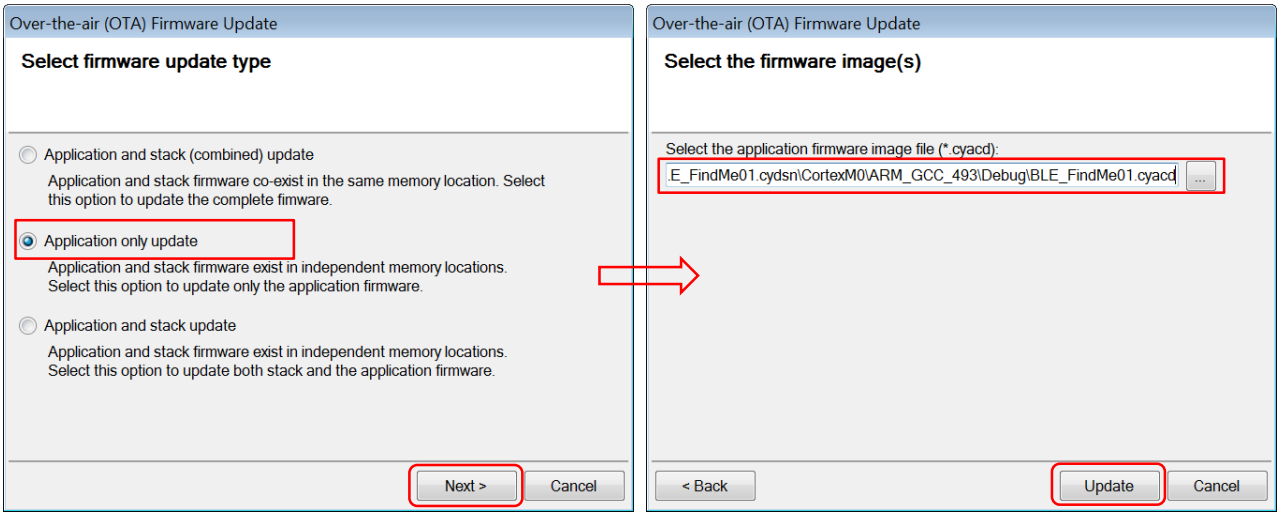
    Note: the Stack Application stops advertising the OTA bootloader service after the advertisement timeout occurs, and switches back to the User Application (Find Me application starts and Green LED turns ON). If this happens, go back to Step 10 and start over.

13. The **Over-The-Air (OTA) Firmware Upgrade** dialog box appears. Select **Application only** update from the available options, and click **Next**. Refer to Figure 26.

14. Navigate to the newly built Application firmware file (.*BLE_FindMe01.cydsn**CortexM0**ARM_GCC_493**Debug**BLE_FindMe01.cyacd*) and select **Open**, and then click the **Update** button.

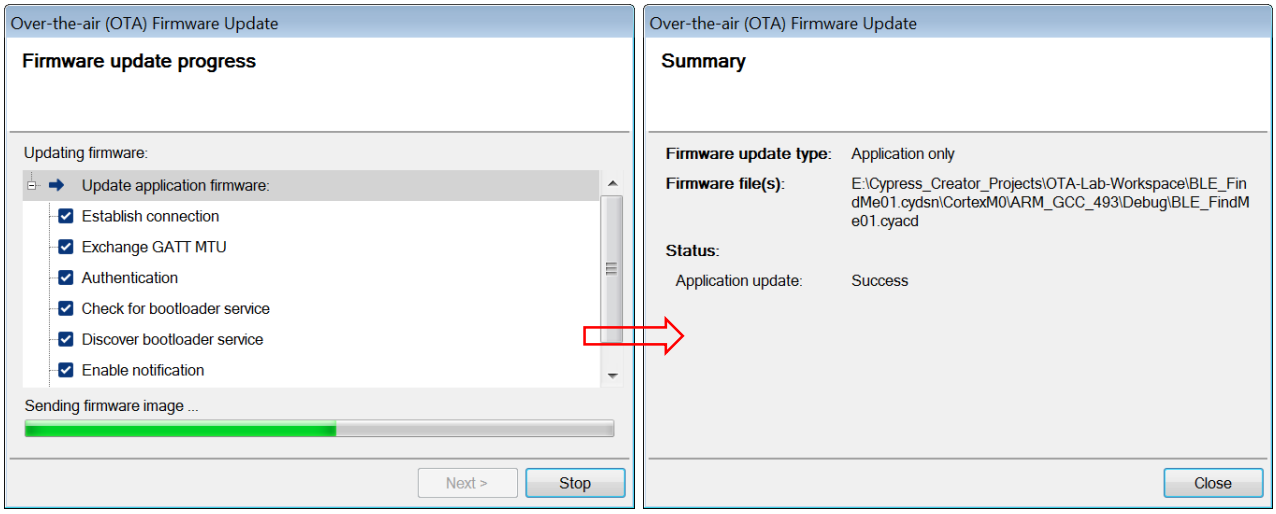
    Note: Ensure that the **Red** LED is still glowing on the Pioneer kit, indicating that Bootloader is active. OTA bootloader times out after about 30 seconds, and switches back to User Application if valid application image is present in the flash. If this happens, before pressing the **Update** button, repeat Step 10.

Figure 26. Selecting Application's .cyacd File for OTA Upgrade



The process of firmware transfer begins. As shown in Figure 27, CySmart executes various steps and uploads firmware to the device. You can see the debug messages getting printed in the **Log** window of the CySmart tool.

Figure 27. Firmware Update Process



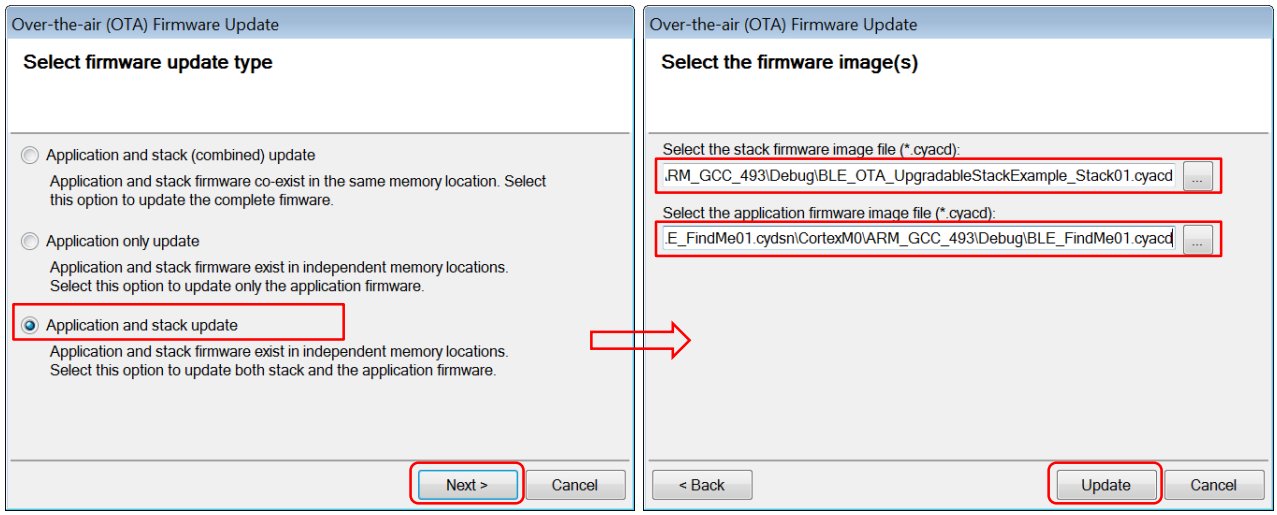
15. After the update, the device resets and starts the new firmware.

    The new BLE_FindMe01 firmware causes the Tri-color LED to blink once in the **White** color before starting advertisement.

Upgrading the Stack Application image, and User Application image:

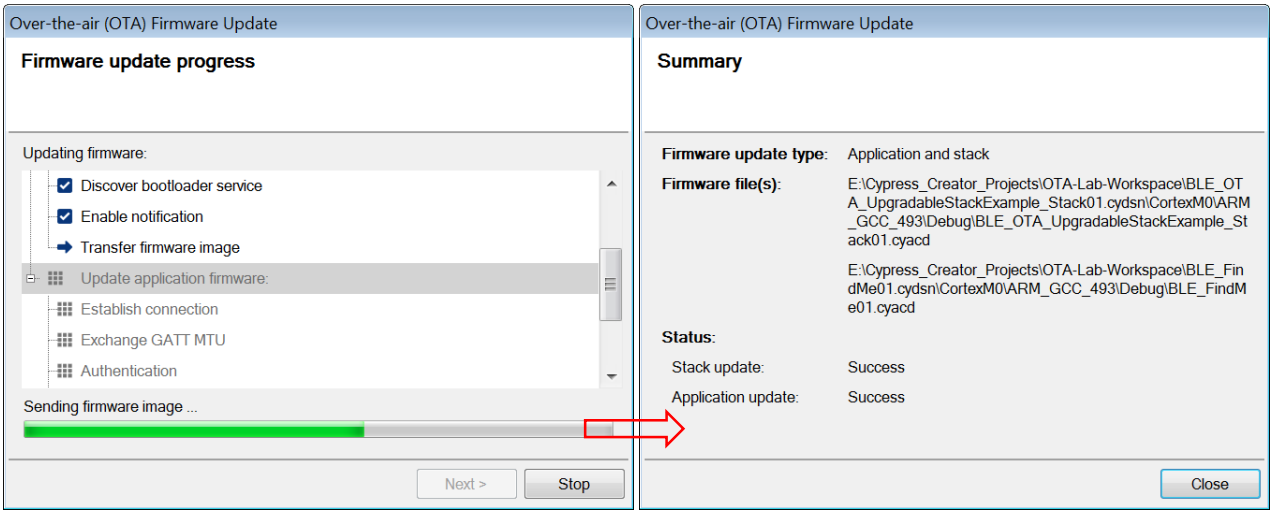
16. The process of upgrading the Stack Application is similar to the User Application upgrade. First, carry out Step 10, Step 11, and Step 12.
17. In **Over-The-Air (OTA) Firmware Upgrade** dialog box, select the **Application and stack update** option from the available options, and click **Next.** Refer to Figure 28.
18. Navigate to the newly built Application firmware image file (.*BLE_FindMe01.cydsn\CortexM0\ARM_GCC_493\Debug\BLE_FindMe01.cyacd*) and Stack firmware (.*BLE_OTA_UpgradableStackExample_Stack01.cydsn\CortexM0\ARM_GCC_493\Debug\BLE_OTA_UpgradableStackExample_Stack01.cyacd*) image file, and then click the **Update** button.

Figure 28. Selecting Application and Firmware .cyacd files for OTA upgrade



The process of firmware transfer begins. As shown in Figure 29, CySmart executes various steps and uploads the stack image to the device followed by the application image. You can see the debug messages getting printed in the **Log** window of the CySmart tool.

Figure 29. Stack Application and User Application Upgrade Process



Congratulations, you have successfully implemented OTA upgrade functionality in the Find Me application example!!

Additional Exercise

Cypress also provides a CySmart mobile app for Android and iOS. This application can be used to perform OTA upgrades from Android or iOS based mobile phones. The CySmart iOS and Android apps are available for download on the App Store and Google Play store respectively. More information and a user guide for this App can be found here - http://www.cypress.com/documentation/software-and-drivers/cysmart-mobile-app.

As an additional exercise, download this app on your mobile phone and perform OTA upgrades using this App. Detailed information about how to perform OTA upgrades using this app is provided in the App User Guide.

Additional Training

PSoC On-Demand Training: To learn more about PSoC Creator and its features watch the PSoC Creator 101 videos available here - http://www.cypress.com/training.

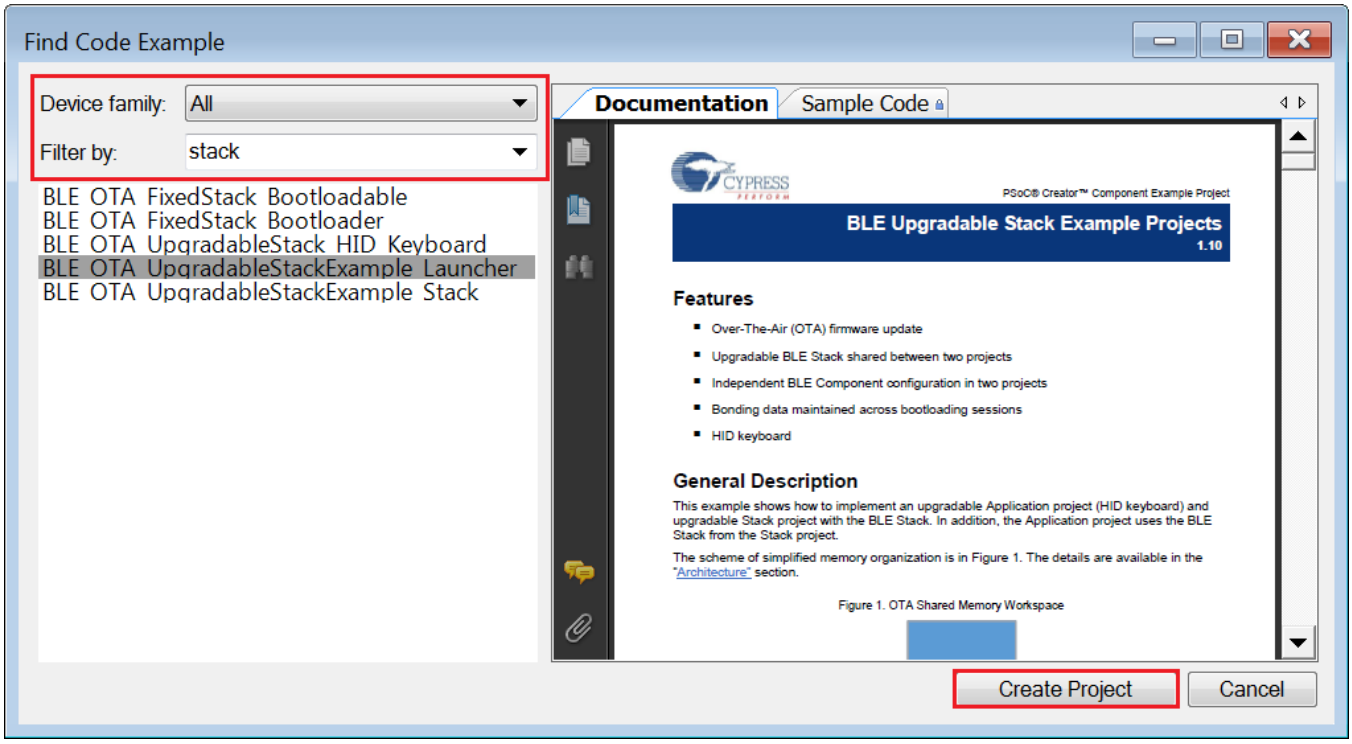
To learn more about Cypress's BLE products, BLE architecture and the BLE Component attend an Introduction to BLE System Design workshop near you. You can register for workshops here - http://www.cypress.com/resource-types/workshops/hands-training-workshop-introduction-ble-system-design

Appendix A: Using PSoC Creator

A.1 Adding an Example Project to an Existing Workspace

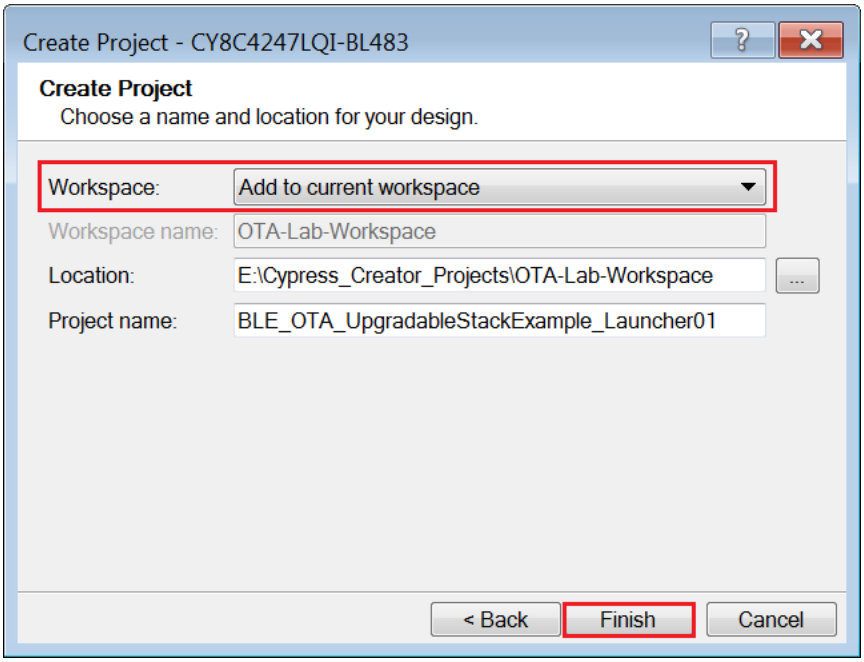
1. Click **File -> Code Example** to open the **Find Example Project** dialog as shown in Figure 30.

Figure 30. Find Example Project dialog



2. Apply the necessary settings in **Device Family** to narrow your search as shown in Figure 30. You can also type in the Project Name in the **Filter by** field to search for a project.
3. Select the **Add to current workspace** option for **Workspace**. Leave the **Location** unchanged (it is the same as that of the current workspace) and click **Finish**. Refer to Figure 31.

Figure 31. Create Project dialog

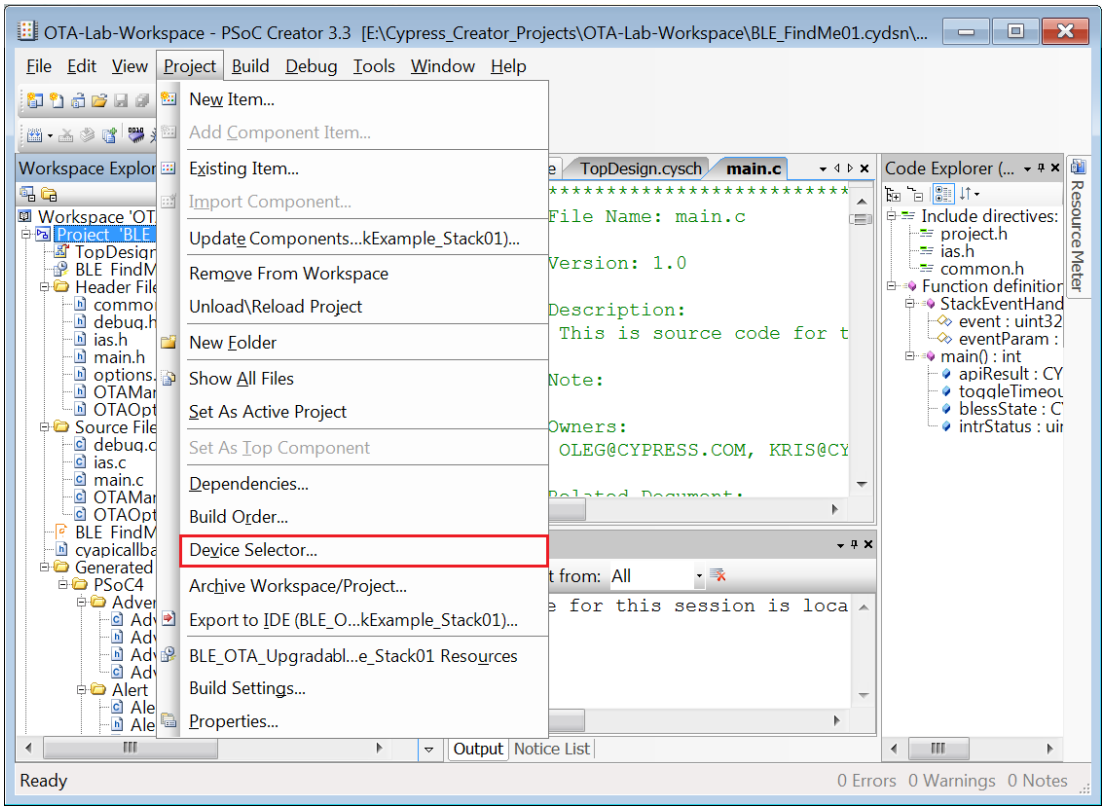


On completing these steps, the selected example project is added to the workspace already opened in that instance of PSoC Creator. A datasheet (<project_name>.pdf) is present under the newly added project. You can go through this document to get details about the example project.

A.2 Selecting Another Device

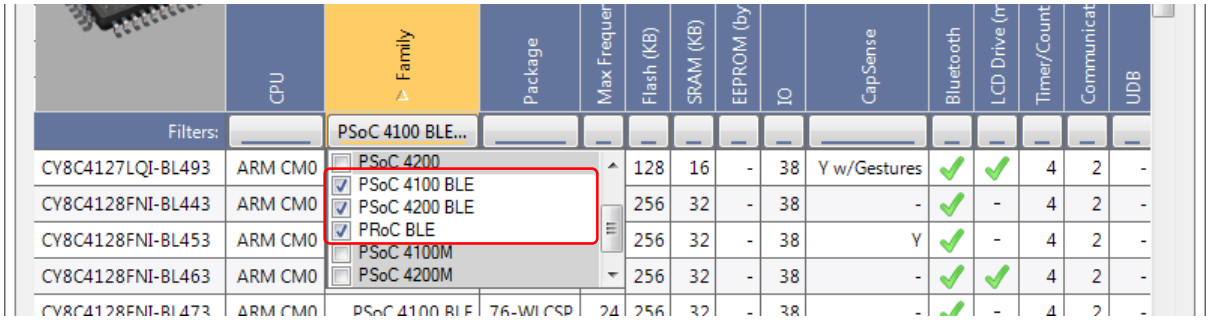
1. In PSoC Creator, select the desired project from **Workspace Explorer** and click **Project -> Device Selector** (see Figure 32).

Figure 32. Launching Device Selector



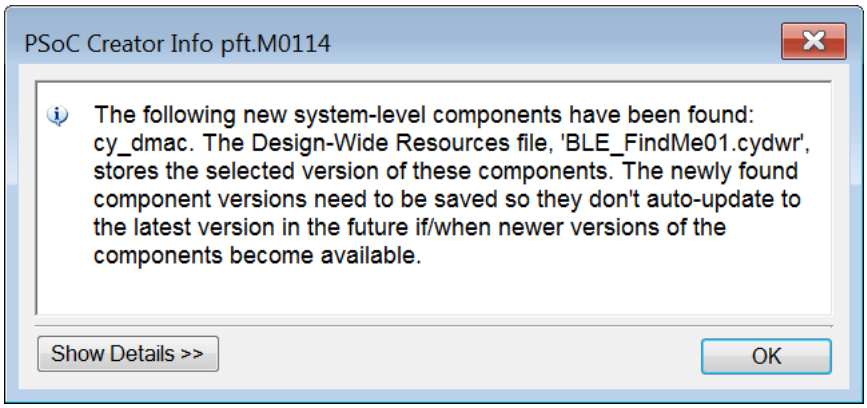
2. Enable 'PSoC 4200 BLE', 'PSoC 4100 BLE' and 'PRoC BLE' filter items under **Family** filter category as shown in Figure 33.

Figure 33. Select Device Family



3. Select the appropriate BLE device for your application from the list.
4. Click **OK** to close the **Device Selector**.
5. You may receive a popup box as shown in Figure 34. Click **OK**.

Figure 34. Dialog box after changing the device part number.



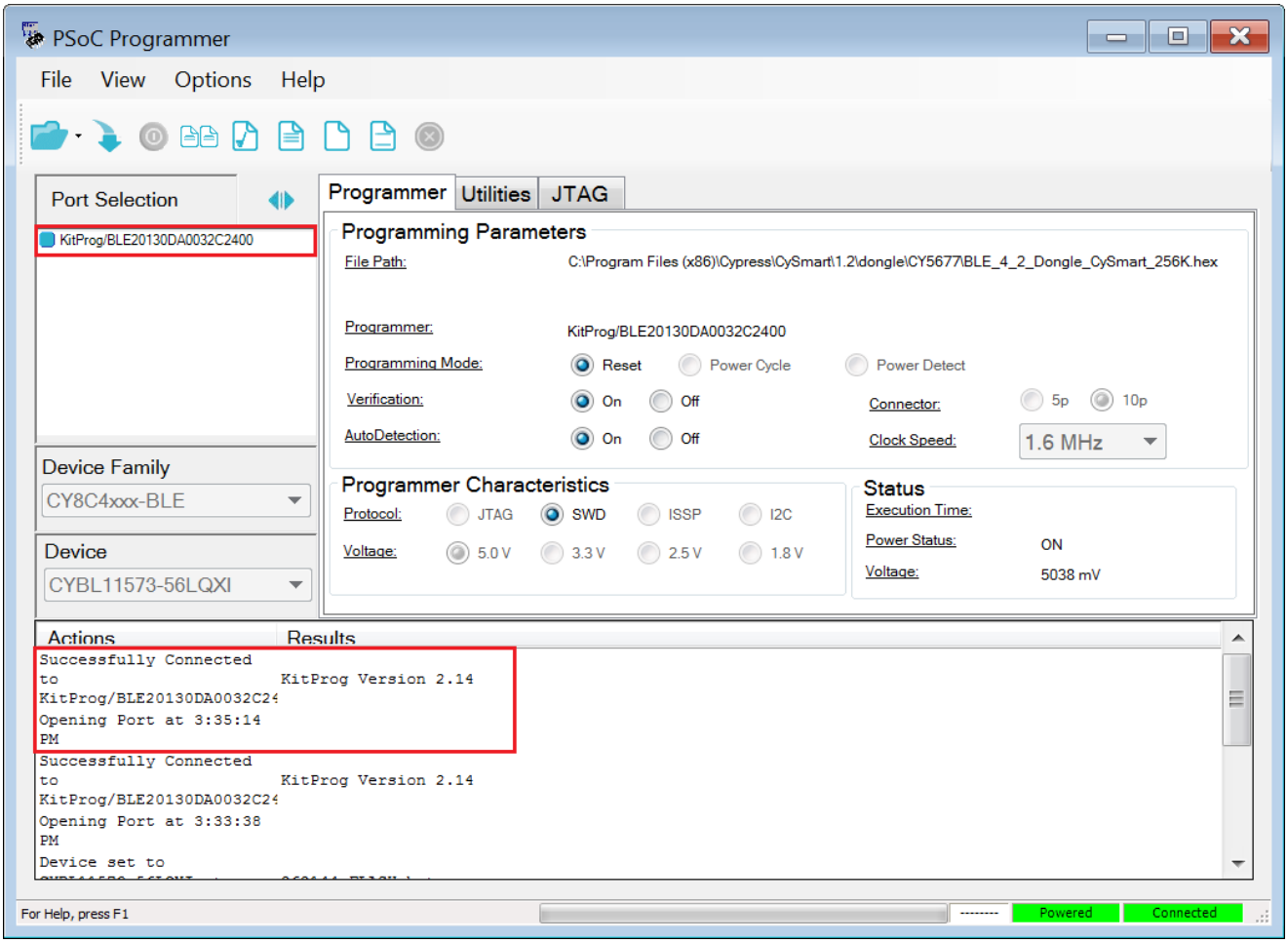
6. Click **File -> Save all** to save changes in the project.

A.3 Updating BLE Dongle Firmware

When you try to connect CySmart 1.2 tool to your BLE Dongle, you may see an **Unsupported Device** under **Unsupported targets** list as shown in Figure 24. This happens when BLE Dongle is loaded with firmware which is not supported by CySmart 1.2 version installed on your computer. You can upgrade the Dongle firmware to a suitable firmware version using the following procedure:

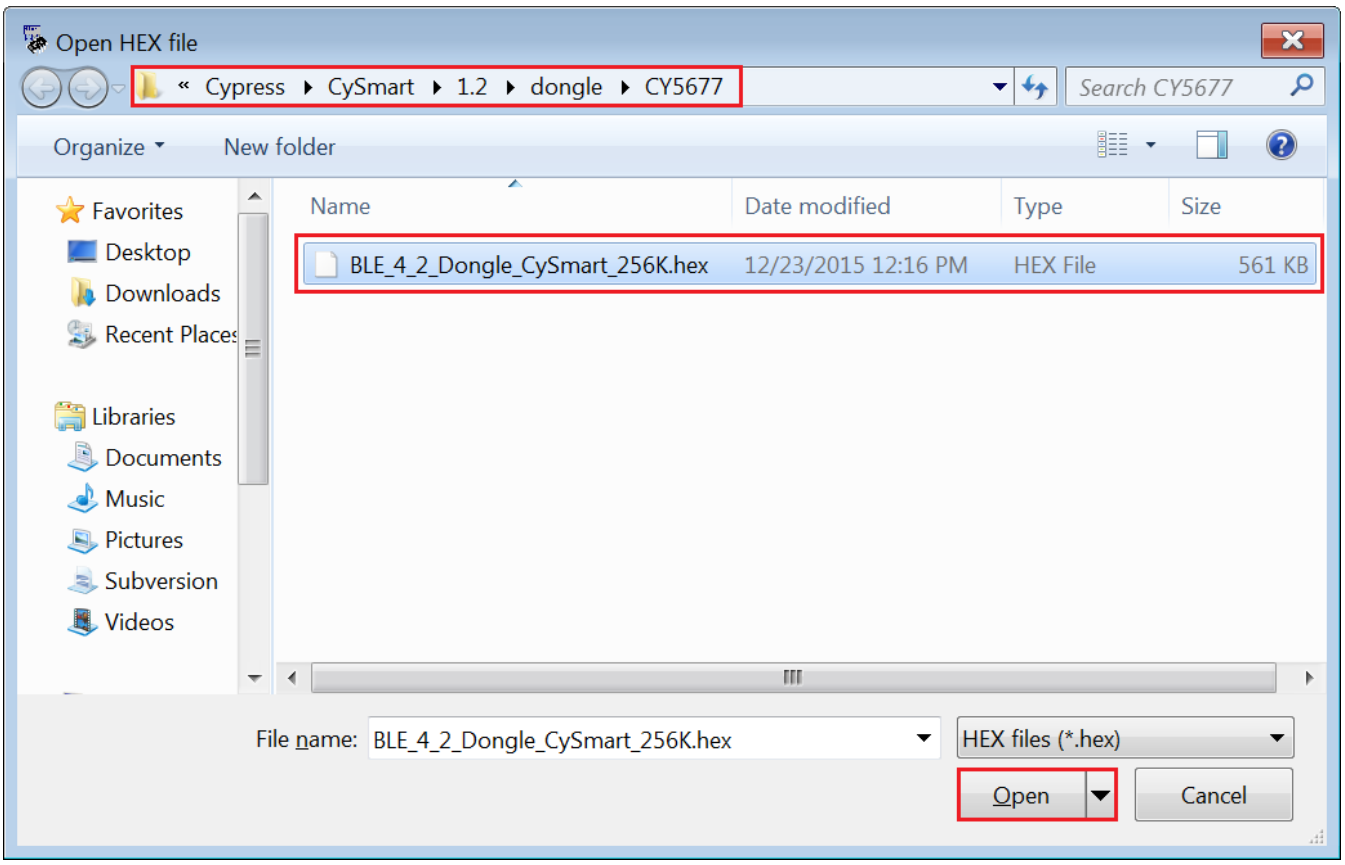
1. Open PSoC Programmer tool. It is located in the **All Programs -> Cypress -> PSoC Programmer 3.24.0** folder in the Windows start menu. Make sure the Dongle is plugged-in to a USB port of your computer.
2. PSoC Programmer will detect the presence of the Dongle and show **KitProg/BLE20130DA0032C2400** in the Port Selection list as shown in Figure 35. Note: the serial number for each Dongle is different, so you may see a different numbered device in the list.
3. Click on the Dongle name to connect to the KitProg available on the Dongle. PSoC Programmer shows a message 'Successfully Connected to KitProg …' in a log window as shown in Figure 35.

Figure 35. Connecting to the KitProg on a Dongle using PSoC Programmer



4. Click on **File -> File Load,** navigate to the *C:\Program Files (x86)\Cypress\CySmart\1.2\dongle\CY5677* directory, select the **BLE_4_2_Dongle_CySmart_256K.hex** file, and Click **Open**. See Figure 36.

Figure 36. Selecting New Firmware File for the Dongle



5. Click **File -> Program**, to initiate programming. PSoC Programmer shows a 'Programming succeeded' message in the log window. See Figure 37.
6. After successful completion of the Dongle firmware upgrade, close PSoC Programmer.
7. In the CySmart tool window, click the **Refresh** button in **Select BLE Dongle Target** dialog box. Now, the BLE Dongle should be visible under the **Supported targets** list as the **CySmart BLE 4.2 USB Dongle**. See Figure 38.

Figure 37. Programming a New Firmware

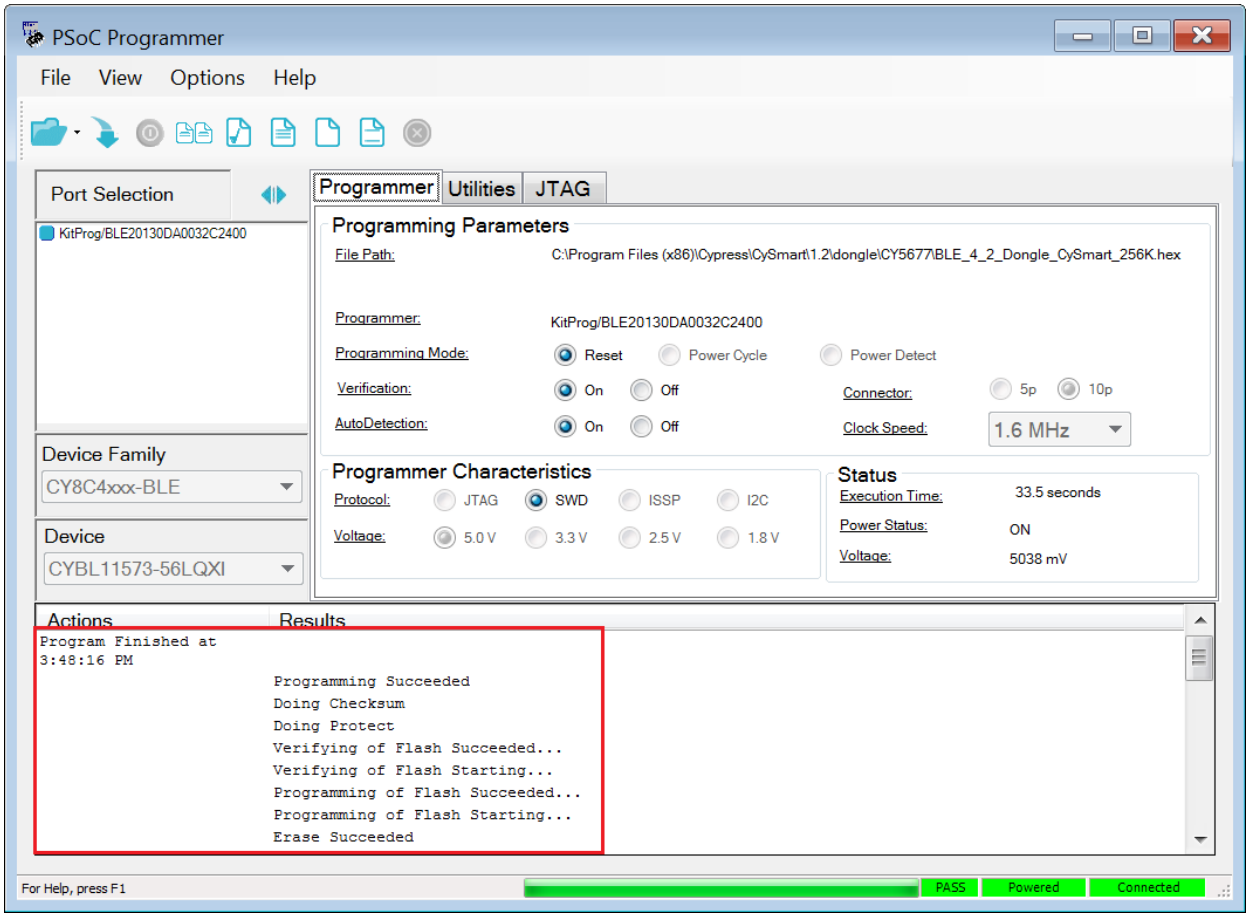
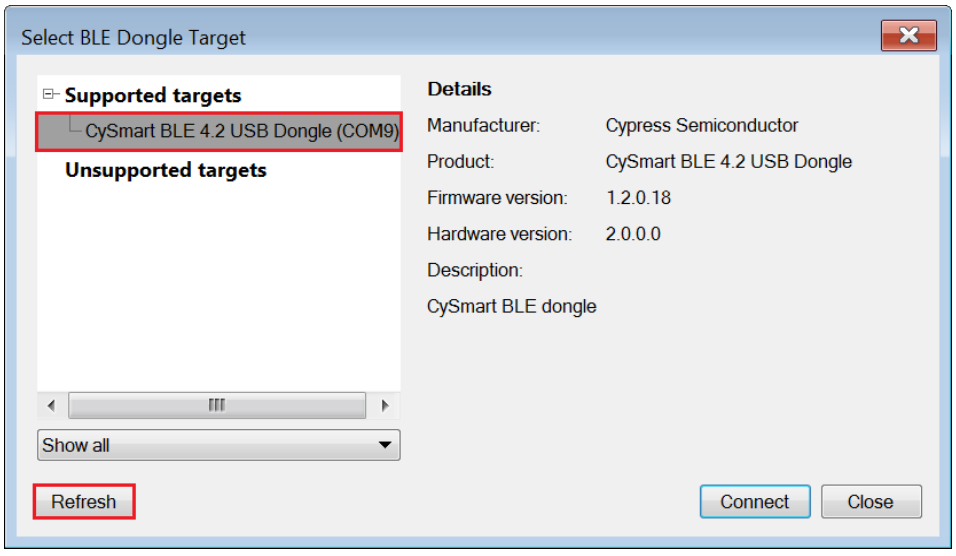


Figure 38. Connecting to BLE Dongle After Dongle Firmware Upgrade



Document Revision History (002-10983)

Revision	By	Description
**	OMKA	Initial Release