CE216767 – PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity Bootloader

Objective

This example demonstrates simple over the air (OTA) bootloading with a PSoC® 6 MCU with Bluetooth Low Energy (BLE) connectivity. This includes downloading an application from a host, installing it in device flash, and then transferring control to that application. The downloaded application demonstrates several basic Bluetooth services. It is based on CE215121 BLE HID Keyboard, with small changes to support the Bootloader SDK.

Overview

This example demonstrates several basic bootloading operations:

- Downloading an application from a host, using the PSoC Creator™ BLE Component for host communication.
- Installing the downloaded application into flash memory.
- Validating an application, and then transferring control to that application.

Requirements

Tool: PSoC Creator 4.2; Peripheral Driver Library (PDL) 3.0.1

Programming Language: C (Arm® GCC 5.4.1 and Arm MDK 5.22)

Associated Parts: All PSoC 63 MCU BLE parts

Related Hardware: CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit

Hardware Setup

Set the VDD Select Switch (SW5) of the CY8CKIT-062-BLE kit to 3.3 V to fully use the RGB LED.

For BLE communications, the BLE USB dongle (CY 5677) provided with the CY8CKIT-062-BLE kit is required.

For debug, connect the kit's USB port to your computer. The KitProg system on the kit acts as both a programmer, and as a USB-UART bridge. For more information, see *KitProg2 User Guide*.

Software Setup

Install the latest CySmart tool in your computer to use the BLE USB dongle.

Operation

The bootloader can either download and install an application or transfer control to a previously downloaded application. The following sections explain how to download an application and how to switch from that application back to the bootloader.

Downloading a Bootloadable Application

1. Plug the CY8CKIT-062-BLE kit board into your computer's USB port.
2. Build the projects App0 and App1.

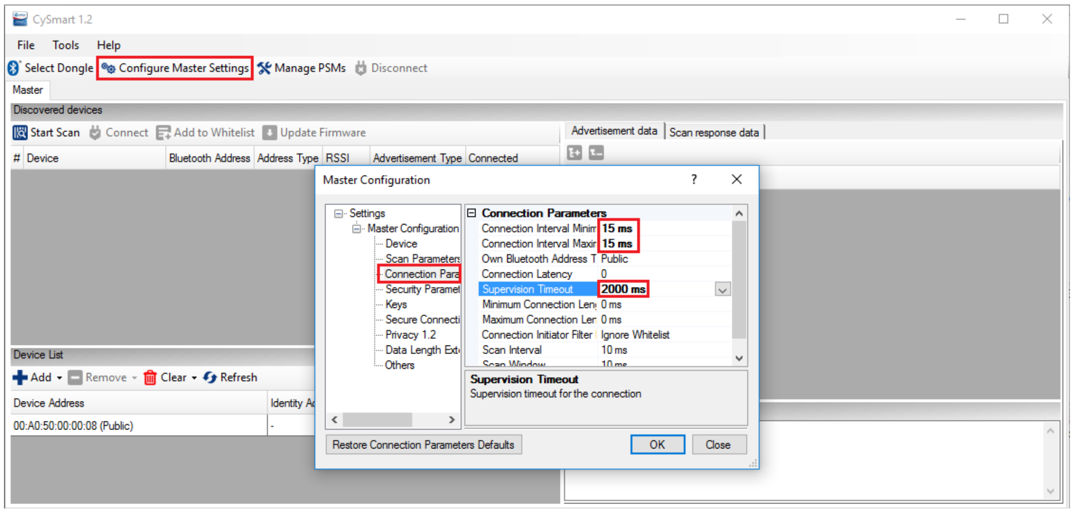
Note: In some cases, you may be prompted to replace files from your project with files from the PDL. These files are templates. Do not replace the customized files for the project. Click **Cancel**.

3. Program App0 into the PSoC 6 MCU device. Programming either core results in both cores being programmed. For more information on device programming, see PSoC Creator Help.
4. Confirm that the kit's LED blinks white once every 2 seconds, indicating that App0 is running.

5. Press and hold the user button (SW2) for at least 0.5 seconds. Confirm that when you release the button, the white LED continues to blink. App0 is the only application installed.
6. Connect the BLE USB dongle (CY5677) provided with the CY8CKIT-062-BLE kit to your computer.
7. Run the CySmart tool on your computer and connect to the BLE USB Dongle.
8. Click **Configure Master Settings**.
9. Go to **Connection Parameters**. Change the **Connection Interval Minimum**, **Connection Interval Maximum**, and **Supervision Timeout** to 15, 15, and 2000 ms, respectively, as Figure 1 shows. Click **OK**.

Note: Using a lower connection interval speeds up the application transmission at an increased risk of losing the connection.

Figure 1. CySmart Connection Parameters



#	Device	Bluetooth Address	Address Type	RSSI	Advertisement Type	Connected

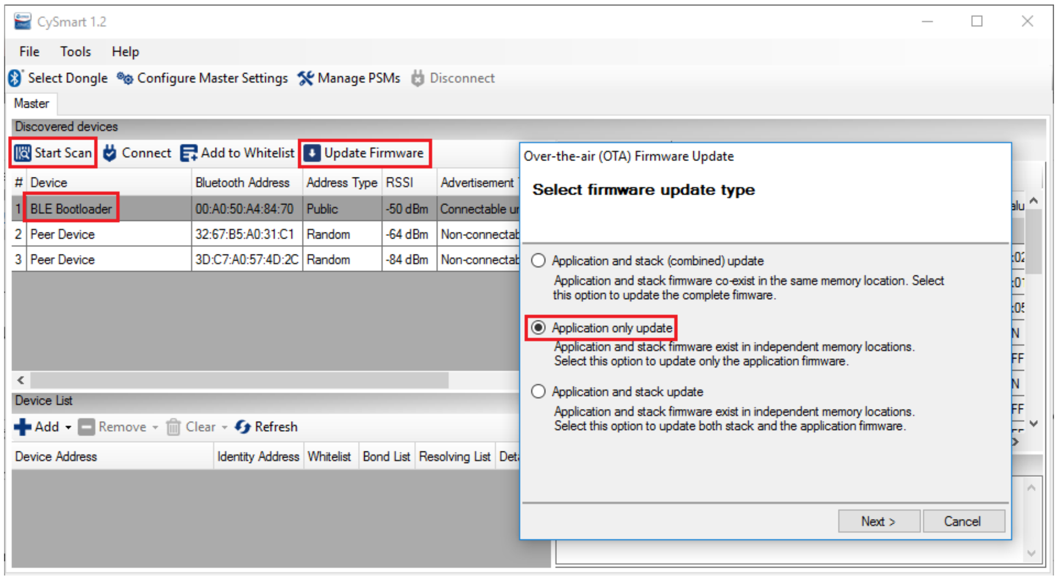
Connection Parameters	
Connection Interval Minim	15 ms
Connection Interval Maxi	15 ms
Own Bluetooth Address T	Public
Connection Latency	0
Supervision Timeout	2000 ms
Minimum Connection Len	0 ms
Maximum Connection Len	0 ms
Connection Initiator Filter	Ignore Whitelist
Scan Interval	10 ms
Scan Window	10 ms

Supervision Timeout
Supervision timeout for the connection

Device Address	Identity A
00:A0:50:00:00:08 (Public)	-

10. Press the user button on the kit if the PSoC 6 MCU is hibernating (indicated by a steady red LED).
11. In CySmart, click **Start Scan** to start scanning for the bootloader device. When the "Bootloader BLE" device is listed, select it.
12. Click **Stop Scan** to stop scanning.
13. Click **Update Firmware** and choose the **Application only update** option, as Figure 2 shows. Click **Next.**

Figure 2. CySmart Firmware Update



#	Device	Bluetooth Address	Address Type	RSSI	Advertisement
1	BLE Bootloader	00:A0:50:A4:84:70	Public	-50 dBm	Connectable u
2	Peer Device	32:67:B5:A0:31:C1	Random	-64 dBm	Non-connectab
3	Peer Device	3D:C7:A0:57:4D:2C	Random	-84 dBm	Non-connectab

Device Address	Identity Address	Whitelist	Bond List	Resolving List	Det

Over-the-air (OTA) Firmware Update

Select firmware update type

( ) Application and stack (combined) update
Application and stack firmware co-exist in the same memory location. Select this option to update the complete firmware.

(●) Application only update
Application and stack firmware exist in independent memory locations. Select this option to update only the application firmware.

( ) Application and stack update
Application and stack firmware exist in independent memory locations. Select this option to update both stack and the application firmware.

14. Browse and select the new bootloadable file (*Bootloader_BLE_App1.cyacd2)* located in the project folder *Bootloader_BLE_App1 > CortexM4 > [compiler name] > Debug*. This file is generated when App1 is built.
15. Click **Update**.
16. Wait for the device firmware to be updated. While the firmware is downloaded, the white LED blinks twice every two seconds.
17. Confirm that the LED is blinking green, indicating that App1 is running.
18. Switch to the bootloader using any of the methods shown below and repeat steps 11 – 17 to test the process of reinstalling App1.

The application has been installed into the internal flash memory, and automatically starts whenever the device is powered ON or reset. It is possible to go back to the bootloader, as the next section explains.

Switching to the Bootloader

The bootloader is not overwritten when downloading an application. The bootloader can still be used to download applications. The following steps show the methods of switching from the bootloadable application to the bootloader.

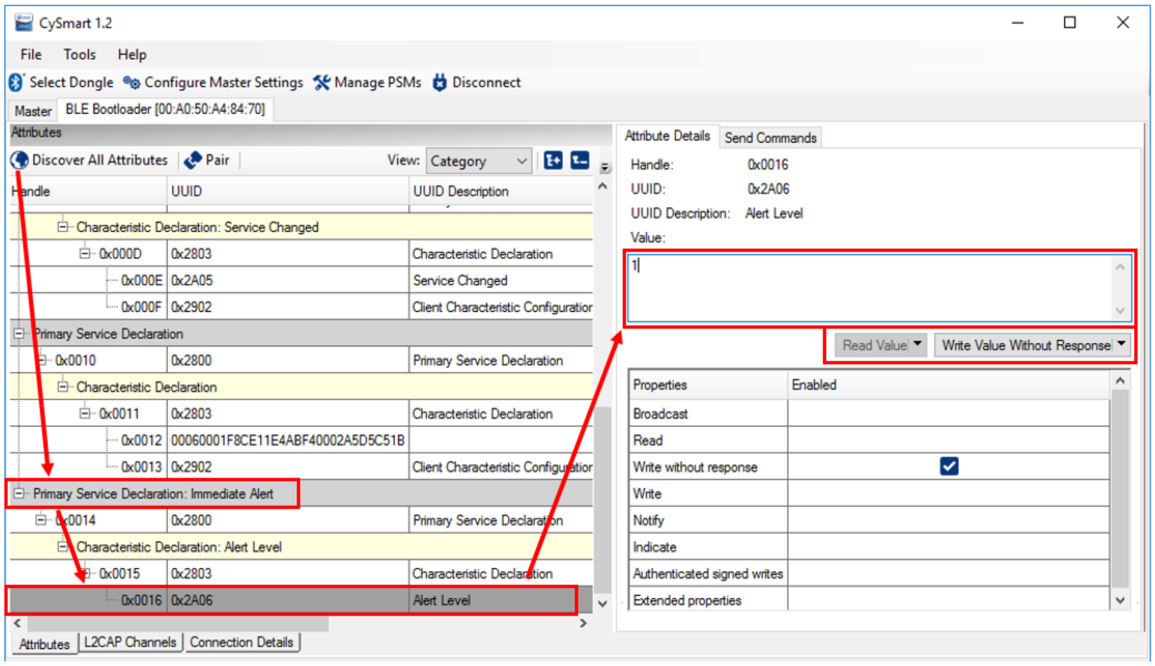
Switching Using IAS

1. Confirm that App1 is running. In CySmart, click **Start Scan**. When the “BLE Keyboard” device is listed, select it.
2. Click **Connect** to connect to the device.
3. Click **Pair** to pair with the device. The Pair button may be hidden if the window size is small.

Note: If pairing fails, disconnect from the device and clear the device list in the CySmart tool. Go back to step 1.

4. Click **No** when prompted to add the device to the resolving list.
5. Click **Discover All Attributes**.
6. Navigate to the **Immediate Alert** service at the bottom and click **Alert Level**. Enter **1** in the **Value** textbox and click **Write Value Without Response**, as Figure 3 shows.

Figure 3. Switching to the Bootloader Using IAS



Handle	UUID	UUID Description
Characteristic Declaration: Service Changed		
0x000D	0x2803	Characteristic Declaration
0x000E	0x2A05	Service Changed
0x000F	0x2902	Client Characteristic Configuration
Primary Service Declaration		
0x0010	0x2800	Primary Service Declaration
Characteristic Declaration		
0x0011	0x2803	Characteristic Declaration
0x0012	00060001F8CE11E4ABF40002A5D5C51B	
0x0013	0x2902	Client Characteristic Configuration
Primary Service Declaration: Immediate Alert		
0x0014	0x2800	Primary Service Declaration
Characteristic Declaration: Alert Level		
0x0015	0x2803	Characteristic Declaration
0x0016	0x2A06	Alert Level

Handle: 0x0016
UUID: 0x2A06
UUID Description: Alert Level
Value:
1

Properties	Enabled
Broadcast	
Read	
Write without response	✓
Write	
Notify	
Indicate	
Authenticated signed writes	
Extended properties	

The connection is terminated.

7. Confirm that the LED is blinking white once every two seconds, indicating that **App0** is running.

Note: This method can also be used to switch from App0 to App1, by selecting the “BLE Bootloader” device instead in step 1.

Switching Using the User Button and a Hardware Reset

Before testing this method, App1 must be running.

1. Confirm that App1 is running.
2. Press the reset and user button at the same time.
3. Release the reset button.
4. Wait until the LED starts blinking white before releasing the user button. App0 is now running.

After downloading an application, both the bootloader and the application reside in flash memory. Whenever the device is powered ON, the bootloader checks whether a valid application exists and seamlessly switches to it, appearing as if it were the only application in the PSoC 6 MCU. You can switch between applications using the Bluetooth IAS, or using the buttons as explained above. An update or an entirely new application can be downloaded to the device while in the bootloader.

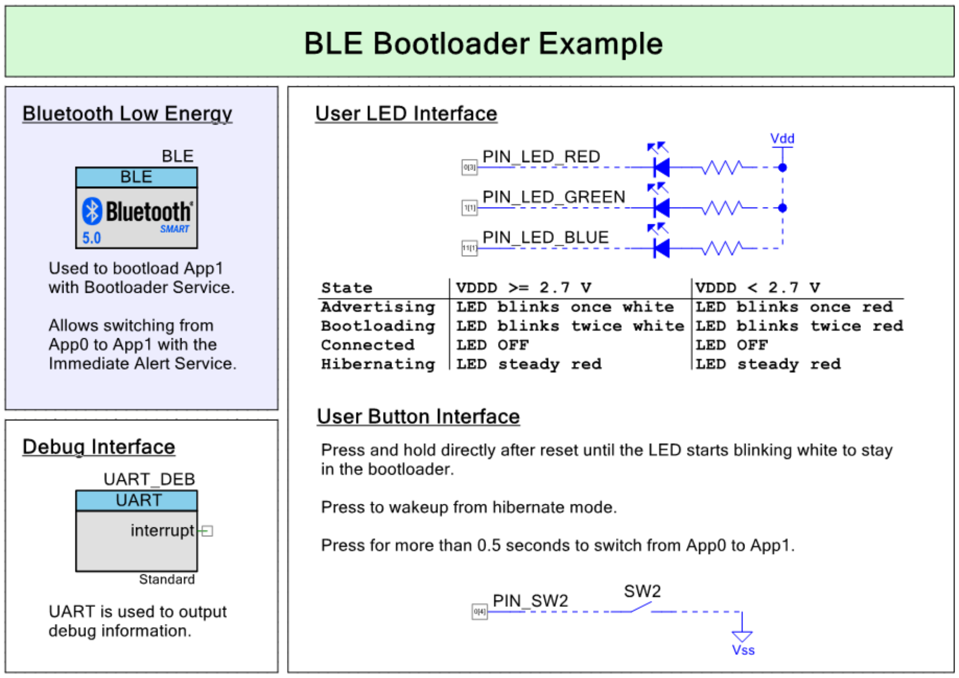
Design and Implementation

This example has two applications “App0” and “App1”. Each application is a separate PSoC Creator project with the following features:

- App0 is the bootloader application; it downloads and installs the bootloadable application (App1).
- Press the user button for more than 0.5 seconds within the bootloader to switch to App1. Switch only occurs if there is a valid App1 in the flash memory.
- To stay in the bootloader application when a valid bootloadable is present, press the user button and the reset button at the same time. Afterwards, release the reset button and hold the user button until the LED starts blinking white.
- To switch between applications without pressing the reset or user button, use the BLE Immediate Alert Service (IAS).
- After 300 seconds of inactivity within App0, the bootloader switches to App1; if there is no valid App1, the bootloader hibernates.
- App1 demonstrates several Bluetooth services. It is a lightly modified version of CE215121 BLE HID Keyboard with Bootloader SDK support added on top of it.

Figure 4 and Figure 5 show the PSoC Creator project schematic for App0 and App1, respectively. For more information on App1, see CE215121.

Figure 4. App0 Schematic



Used to bootload App1 with Bootloader Service.

Allows switching from App0 to App1 with the Immediate Alert Service.

UART is used to output debug information.

State	VDDD >= 2.7 V	VDDD < 2.7 V
Advertising	LED blinks once white	LED blinks once red
Bootloading	LED blinks twice white	LED blinks twice red
Connected	LED OFF	LED OFF
Hibernating	LED steady red	LED steady red

Press and hold directly after reset until the LED starts blinking white to stay in the bootloader.

Press to wakeup from hibernate mode.

Press for more than 0.5 seconds to switch from App0 to App1.

Figure 5. App1 Schematic

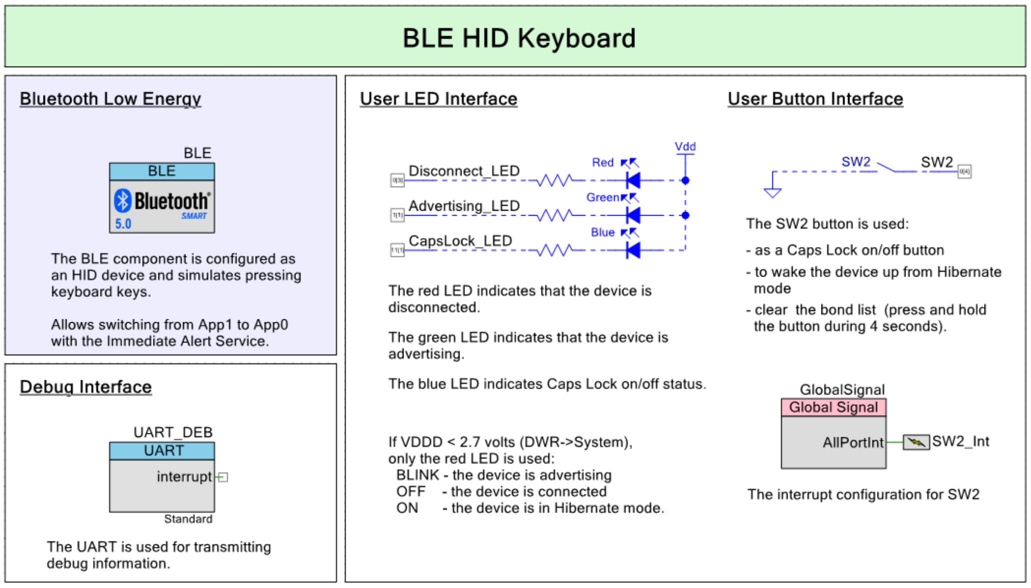
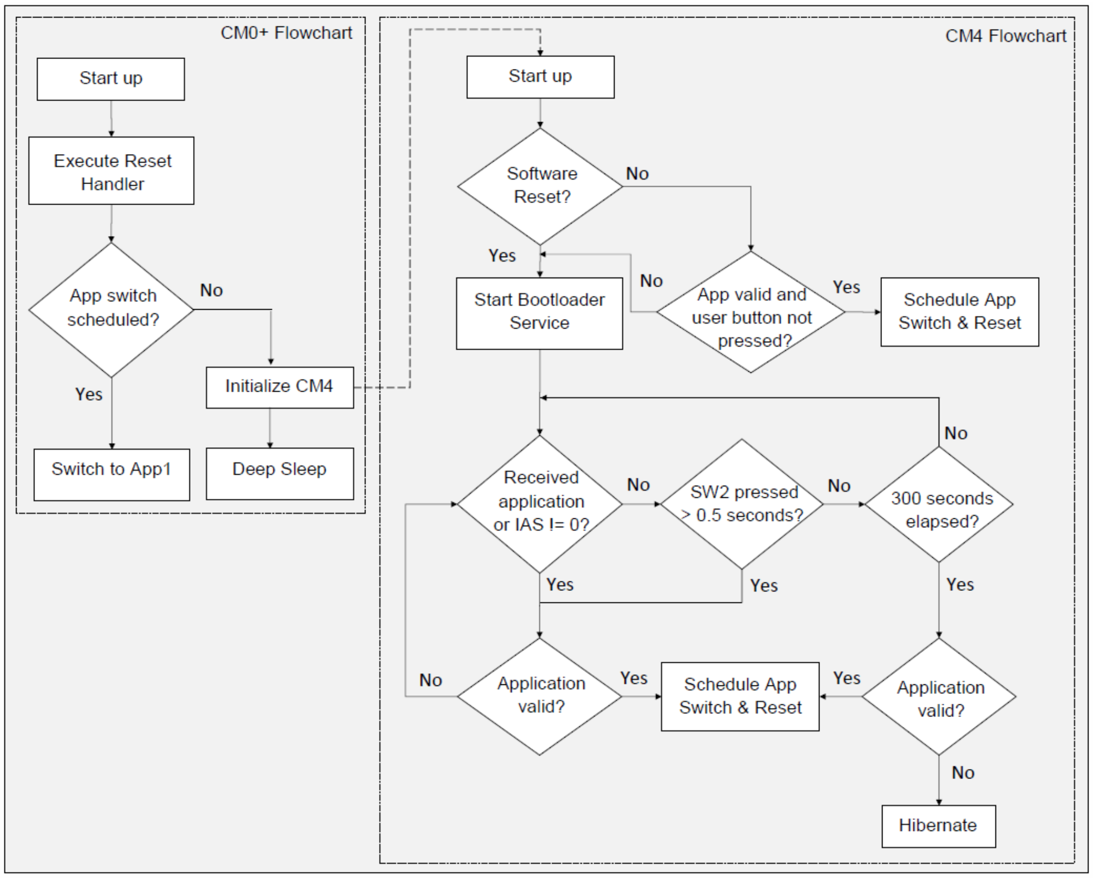

BLE HID Keyboard
Bluetooth Low Energy
BLE
BLE
Bluetooth SMART
5.0
The BLE component is configured as an HID device and simulates pressing keyboard keys.
Allows switching from App1 to App0 with the Immediate Alert Service.
Debug Interface
UART_DEB
UART
interrupt
Standard
The UART is used for transmitting debug information.
User LED Interface
Vdd
Red
Green
Blue
Disconnect_LED
Advertising_LED
CapsLock_LED
The red LED indicates that the device is disconnected.
The green LED indicates that the device is advertising.
The blue LED indicates Caps Lock on/off status.
If VDDD < 2.7 volts (DWR->System), only the red LED is used:
BLINK - the device is advertising
OFF - the device is connected
ON - the device is in Hibernate mode.
User Button Interface
SW2
SW2
The SW2 button is used:
- as a Caps Lock on/off button
- to wake the device up from Hibernate mode
- clear the bond list (press and hold the button during 4 seconds).
GlobalSignal
Global Signal
AllPortInt
SW2_Int
The interrupt configuration for SW2

Figure 6 shows the firmware flow of App0.

Figure 6. App0 Firmware Flowchart



Design Firmware

The firmware portion of the design is implemented in the files listed in Table 1. Many of these files require custom settings in both the file and the related PSoC Creator projects. For more information on customizing PSoC Creator projects for the Bootloader Source Development Kit (SDK), see the PSoC 6 MCU Bootloader SDK Guide.

Several `#define` statements are provided in the *bootload_user.h* file to customize the bootload operation and enable features of the Bootloader SDK. The default settings can be used for most designs.

Table 1. Design Firmware Files

File	Description
main_cm4.c main_cm0p.c	Contains the main() function for each core. Some PSoC 63 MCUs have two cores, an Arm Cortex®-M4 and a Cortex-M0+. See Table 2 for specific tasks for each core.
ias.c / .h	Immediate Alert Service (IAS) files. Used to implement IAS for communication between BLE client and server. When the device receives a non-zero value with the IAS, it switches applications.
debug.c / .h	UART `printf` implementation and LED status notification.
cy_bootload.c / .h	The bootloader software development kit (SDK) files.
bootload_user.h	Contains `#define` user-editable statements that control the operation and enable features in the SDK.
bootload_user.c	Contains user functions required by the SDK: ▪ Five functions that control communications with the bootloader host. These are also called *transport functions.* ▪ Two functions (`Cy_Bootload_ReadData` and `Cy_Bootload_WriteData`) that control access to the internal or external memory.
transport_ble.c / .h	Contains bootloader transport functions for the BLE Component. These functions are typically called by the transport functions in *bootload_user.c.*
bootload_common.ld	GCC linker script. It describes the memory layout and the locations in memory for each CPU core in each application. It is included in other GCC linker script files. It must be common to all application projects that are loaded into the same device.
bootload_cm0p.ld, bootload_cm4.ld	Custom GCC linker scripts. In each application, use these files as project linker scripts instead of the default PDL linker script files. These files place the code and data sections for each of the cores as well as the bootloader and other regions. These files include the memory layout described in *bootload_common.ld.*
bootload_mdk_common.h *bootload_mdk_symbols.c*	These files are used by the MDK linker scripts (*bootload_cm0.scat* and *bootload_cm4.scat*) and define memory region limits.
bootload_cm0p.scat, *bootload_cm4.scat*	MDK linker scripts. In each application, use these files as project linker scripts instead of the default PDL linker script files. These files place the code and data sections for each of the cores as well as the bootloader and other regions.
bootload_common.icf	This file is included by IAR Embedded Workbench linker scripts. It defines memory regions and shared ELF file symbols.
bootload_cm0p.icf, *bootload_cm4.icf*	IAR linker scripts. In each application, use these files as project linker scripts instead of the default PDL linker script files. These files place the code and data sections for each of the cores as well as the bootloader and other regions.
post_build_core1.bat	Batch file to create the downloadable application.

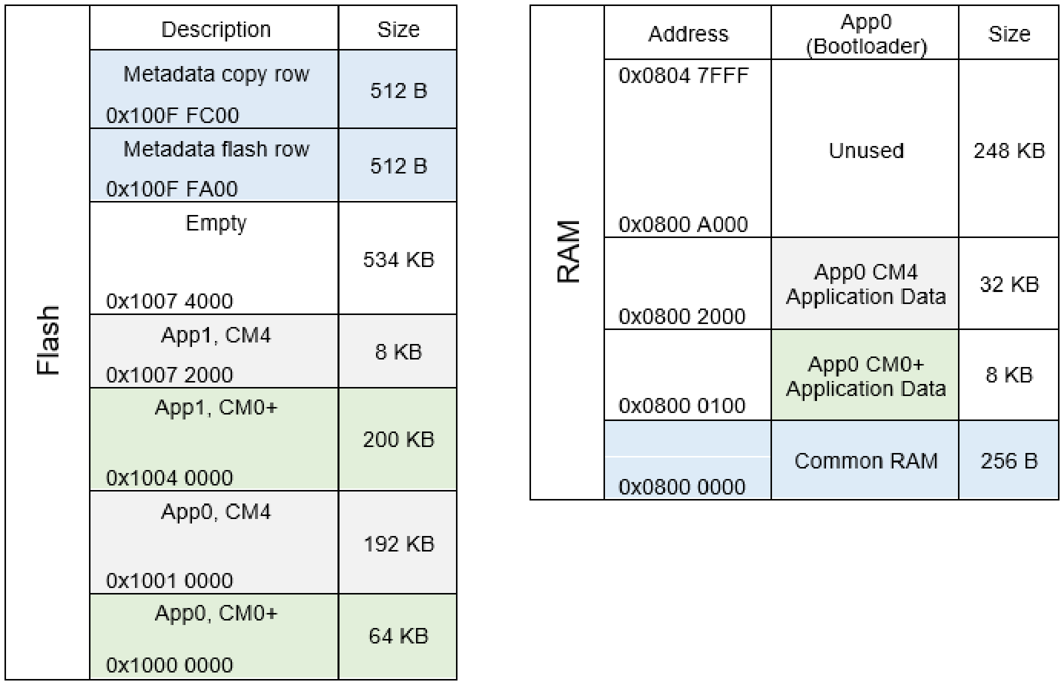
Memory Layout

Figure 7 shows typical memory usage for each core in each application. This layout is done for PSoC 6 MCU devices with 1 MB flash and 288 KB SRAM. The BLE bootloader uses around 170 KB of flash to accommodate the BLE API. Note that in App0, the BLE stack is executed by the Cortex-M4 (CM4) CPU, and in App1 the BLE stack is executed by the Cortex-M0+ (CM0+) CPU. For more information on the memory layout, see the PSoC 6 MCU Bootloader SDK Guide.

Note: App0 always starts at the beginning of the device user flash at address 0x1000 0000. For more information on the device memory map, see the device datasheet.

To change the memory layout or usage, update the linker script files shown in Table 1.

Figure 7. Memory Layout of Applications



Flash	Description	Size
	Metadata copy row 0x100F FC00	512 B
	Metadata flash row 0x100F FA00	512 B
	Empty 0x1007 4000	534 KB
	App1, CM4 0x1007 2000	8 KB
	App1, CM0+ 0x1004 0000	200 KB
	App0, CM4 0x1001 0000	192 KB
	App0, CM0+ 0x1000 0000	64 KB

RAM	Address	App0 (Bootloader)	Size
	0x0804 7FFF 0x0800 A000	Unused	248 KB
	0x0800 2000	App0 CM4 Application Data	32 KB
	0x0800 0100	App0 CM0+ Application Data	8 KB
	0x0800 0000	Common RAM	256 B

Design Considerations

Dual Core

Some PSoC 63 MCU BLE parts are dual core devices with an Arm Cortex-M4 and a Cortex-M0+. An application can include code for one or both cores. The cores in each application behave as shown in Table 2. This can be easily changed so that either core can run any of the tasks, including bootloading.

Table 2. Core Tasks per Application

Application	Cortex-M0+	Cortex-M4
App0 (Bootloader)	Executes first at device reset. Reset handler controls actual application transfer. Activates Cortex-M4 core.	Bootloads App1. Supervises the App switching button. After bootload or when the button is pressed, initiates transfer to App1 through a software reset. If IAS alert level is greater than zero, switches to App1 through a software reset.
App1 (Bootloadable)	Demonstrates BLE services, as documented in CE215121, BLE HID Keyboard. If IAS alert level is greater than zero, switches to App0 through a software reset.	Does nothing.

Software Reset

When transferring control from one application to another, the recommended method is through a device software reset. This enables each application to initialize device hardware blocks and signal routing from a known state.

You can freeze the state of I/O pins so that they are maintained through a software reset. Defined portions of SRAM are also maintained through a software reset. For more information, see the PSoC 6 MCU: PSoC 63 with BLE Architecture Technical Reference Manual.

Bootloader Modifications

The default bootloader linker scripts allocate an equal amount of flash and RAM to each CPU core in each application. In this code example, the allocations are modified because of the size of the BLE stack, and to decrease the size of App1 to reduce bootloading time. The modifications were done to linker script files for both App0 and App1.

LED Status

App0 indicates the current bootloading state on the kit's RGB LED. Table 3 shows the behavior of the LED depending on the bootloader state.

The LED blinks white once every 2 seconds indicating that the device is advertising. If no Bluetooth activity is detected for 300 seconds, the bootloader switches to App1. If App1 is not valid, the LED turns red and the device hibernates.

Table 3. LED Status for App0

Color	State	Description
White	Blinks once every 2 seconds	Bootloader is advertising
White	Blinks twice every 2 seconds	An application is being received
None	OFF	Bootloader is connected but not receiving an application
Red	Steady	Hibernating

Note: If the specified V_{DDD} within **Design Wide Resources** > **System** is less than 2.7 V, only the red LED is used.

App1 indicates the BLE keyboard status on the kit's RGB LED, as Table 4 shows. The LED blinks green indicating that the device is advertising. After the BLE keyboard connects to a device, the LED indicates the status of the Caps Lock. If the device remains disconnected for 150 seconds, the LED turns red and the device hibernates.

Table 4. LED Status for App1

Color	State	Description
Green	Blinking	BLE Keyboard is advertising
None	OFF	BLE Keyboard is connected
Blue	Steady	Caps Lock is ON
Red	Steady	Hibernating

Note: If the specified V_{DDD} within the **Design Wide Resources** > **System** is less than 2.7 V, only the red LED is used. Caps Lock status is not shown.

Components and Settings

Table 5 lists the PSoC Creator Components used in this example for App0, how they are used in the design, and the non-default settings required so they function as intended. For information on the Components used in App1, see CE215121.

Table 5: PSoC Creator Components

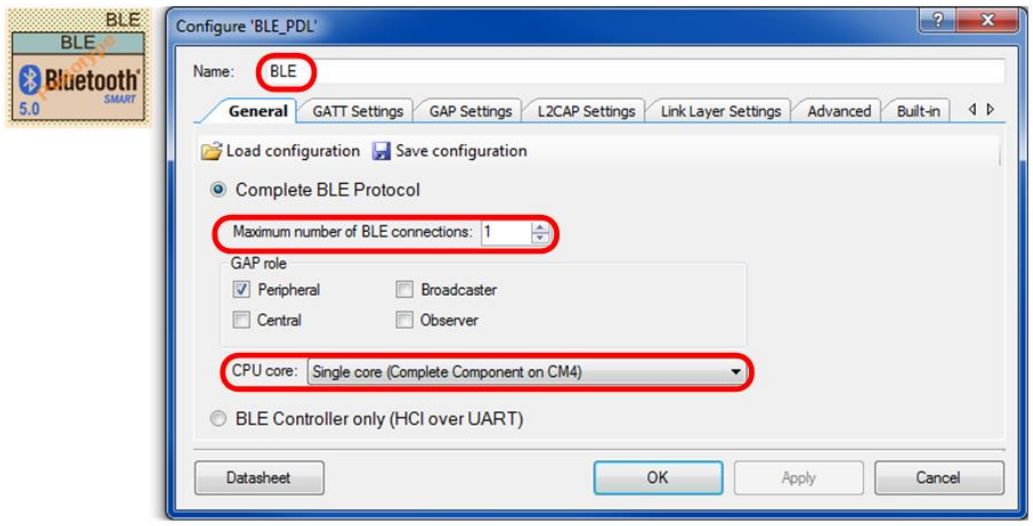
Component	Instance Name	Purpose	Non-default Settings
Bluetooth Low Energy	BLE	Provides communication between the PSoC 6 MCU device and the Bluetooth host for bootloading and app switching.	See BLE Component Configuration.
UART	UART_DEB	Outputs Bluetooth related debug information.	Interrupt mode external.
Pin	PIN_LED_RED PIN_LED_GREEN PIN_LED_BLUE PIN_SW2	LED Status notification.	HW Connection: Unchecked. External Terminal: Checked. LED Pins Drive Mode: High Impedance Digital. SW2 Pin Initial Drive Mode: Resistive Pull-Up.

For information on the hardware resources used by a Component, see the Component datasheet.

BLE Component Configuration

- General tab (see Figure 8):
  - Maximum number of BLE Connections: 1
  - CPU core: Single core (Complete Component on CM4)

Figure 8. BLE Component, General Tab Configuration



Name: BLE

(●) Complete BLE Protocol

Maximum number of BLE connections: 1

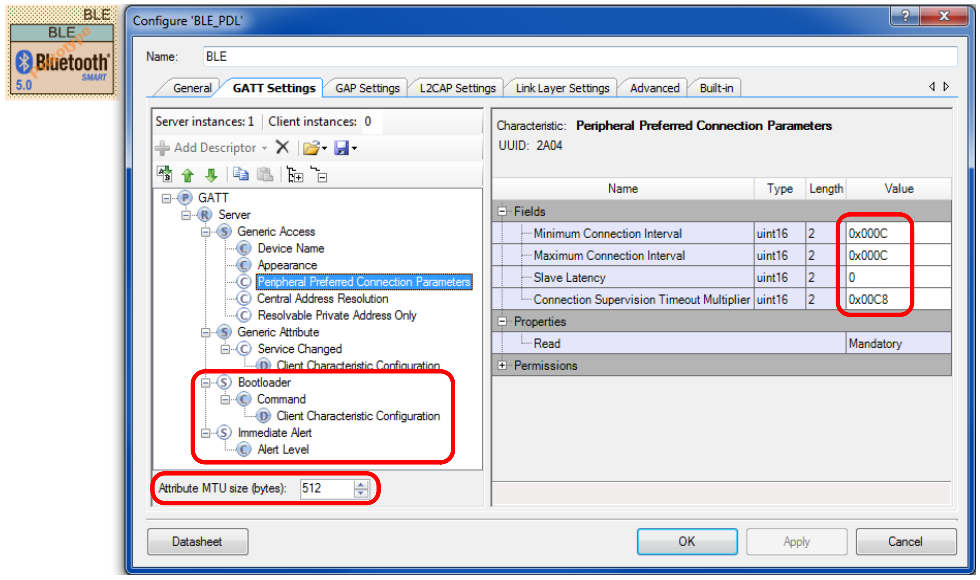
GAP role
[✓] Peripheral [ ] Broadcaster
[ ] Central [ ] Observer

CPU core: Single core (Complete Component on CM4)

( ) BLE Controller only (HCI over UART)

- GATT Settings tab (see Figure 9):
  - Generic Access, Peripheral Preferred Connection Parameters:
  - Minimum Connection Interval: 0x000C
  - Maximum Connection Interval: 0x000C
  - Connection Supervision Timeout Multiplier: 0x00C8
    The above intervals are selected to minimize bootloading time.
  - Bootloader service for BLE bootloading
  - Immediate Alert service for app switching
  - Attribute MTU size (bytes): 512

Figure 9. BLE Component, GATT Settings Tab Configuration



Configure 'BLE_PDL'

Name: BLE

Server instances: 1 | Client instances: 0

- GATT
  - Server
    - Generic Access
      - Device Name
      - Appearance
      - Peripheral Preferred Connection Parameters
      - Central Address Resolution
      - Resolvable Private Address Only
    - Generic Attribute
      - Service Changed
        - Client Characteristic Configuration
    - Bootloader
      - Command
        - Client Characteristic Configuration
    - Immediate Alert
      - Alert Level

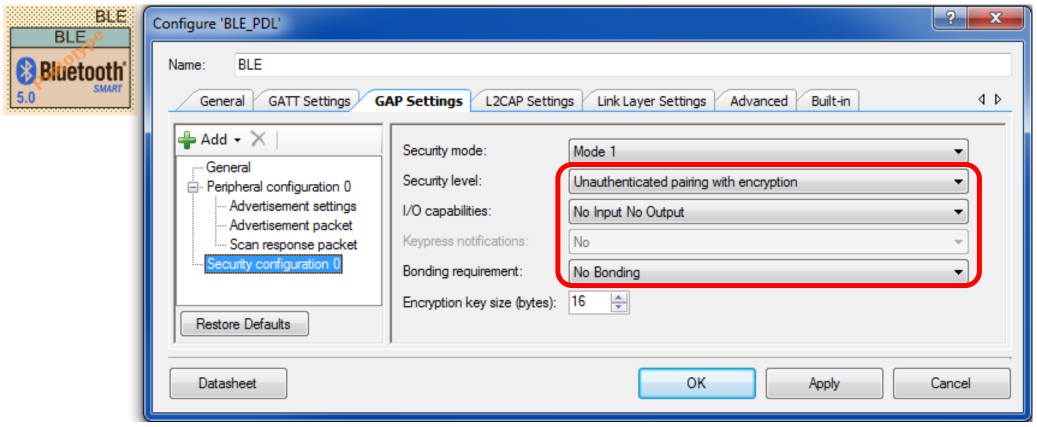
Attribute MTU size (bytes): 512

Characteristic: **Peripheral Preferred Connection Parameters**
UUID: 2A04

Name	Type	Length	Value
Fields			
Minimum Connection Interval	uint16	2	0x000C
Maximum Connection Interval	uint16	2	0x000C
Slave Latency	uint16	2	0
Connection Supervision Timeout Multiplier	uint16	2	0x00C8
Properties			
Read			Mandatory
Permissions			

- GAP Settings tab:
  - Device Name: “BLE Bootloader”
  - Peripheral Configuration 0, Advertisement packet: Local Name checked and set to Complete
  - Security configuration 0 (see Figure 10), Security level: Unauthenticated pairing with encryption
  - Security configuration 0, I/O capabilities: No Input No Output
  - Security configuration 0, Bonding requirement: No Bonding

Figure 10. BLE Component, GAP Settings Tab Configuration



Name: BLE

Security mode: Mode 1

Security level: Unauthenticated pairing with encryption

I/O capabilities: No Input No Output

Keypress notifications: No

Bonding requirement: No Bonding

Encryption key size (bytes): 16

- Link Layer Settings tab:
  - Link layer max TX and RX payload size (bytes): 251

Reusing This Example

This example is designed for the CY8CKIT-062-BLE Pioneer kit. To port the design to a different PSoC 6 MCU device and/or kit, change the target device using the Device Selector and update the pin assignments in the Design Wide Resources Pins settings as needed.

For single-core PSoC 63 MCU BLE devices, port the code from *main_cm4.c* to *main.c,* and copy the Cy_OnResetUser function from *main_cm0p.c* to *main.c.*

Adding BLE Bootloader Support to another Project

This section describes how to add the BLE OTA bootloading capability to an existing project. For more detailed information on how to create, configure, and use a bootloader project, see the PSoC 6 MCU Bootloader SDK Guide.

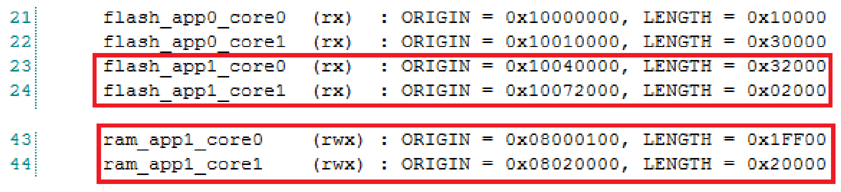
Note: The UART and LED Components in this project are not required for the bootloader. To disable them, open the *debug.h* file and modify the **Conditional Compilation Parameters**. Afterwards, remove the Components from the *TopDesign.cysch* schematic.

1. Add the BLE Bootloader to your workspace.

   1.1. Copy App0 project to your workspace.

   1.2. Edit the common linker script located within the **Shared Files** folder of App0 according to the flash and RAM requirements of your application. Figure 11 shows the fields that require modification for the GCC common linker script (*bootload_common.ld*), for MDK/IAR compilers, see Appendix A. If the values are not known, use the provided values.

Figure 11. Modification of the GCC Common Linker Script



```
    flash_app0_core0  (rx)  : ORIGIN = 0x10000000, LENGTH = 0x10000
    flash_app0_core1  (rx)  : ORIGIN = 0x10010000, LENGTH = 0x30000
    flash_app1_core0  (rx)  : ORIGIN = 0x10040000, LENGTH = 0x32000
    flash_app1_core1  (rx)  : ORIGIN = 0x10072000, LENGTH = 0x02000

    ram_app1_core0    (rwx) : ORIGIN = 0x08000100, LENGTH = 0x1FF00
    ram_app1_core1    (rwx) : ORIGIN = 0x08020000, LENGTH = 0x20000
```

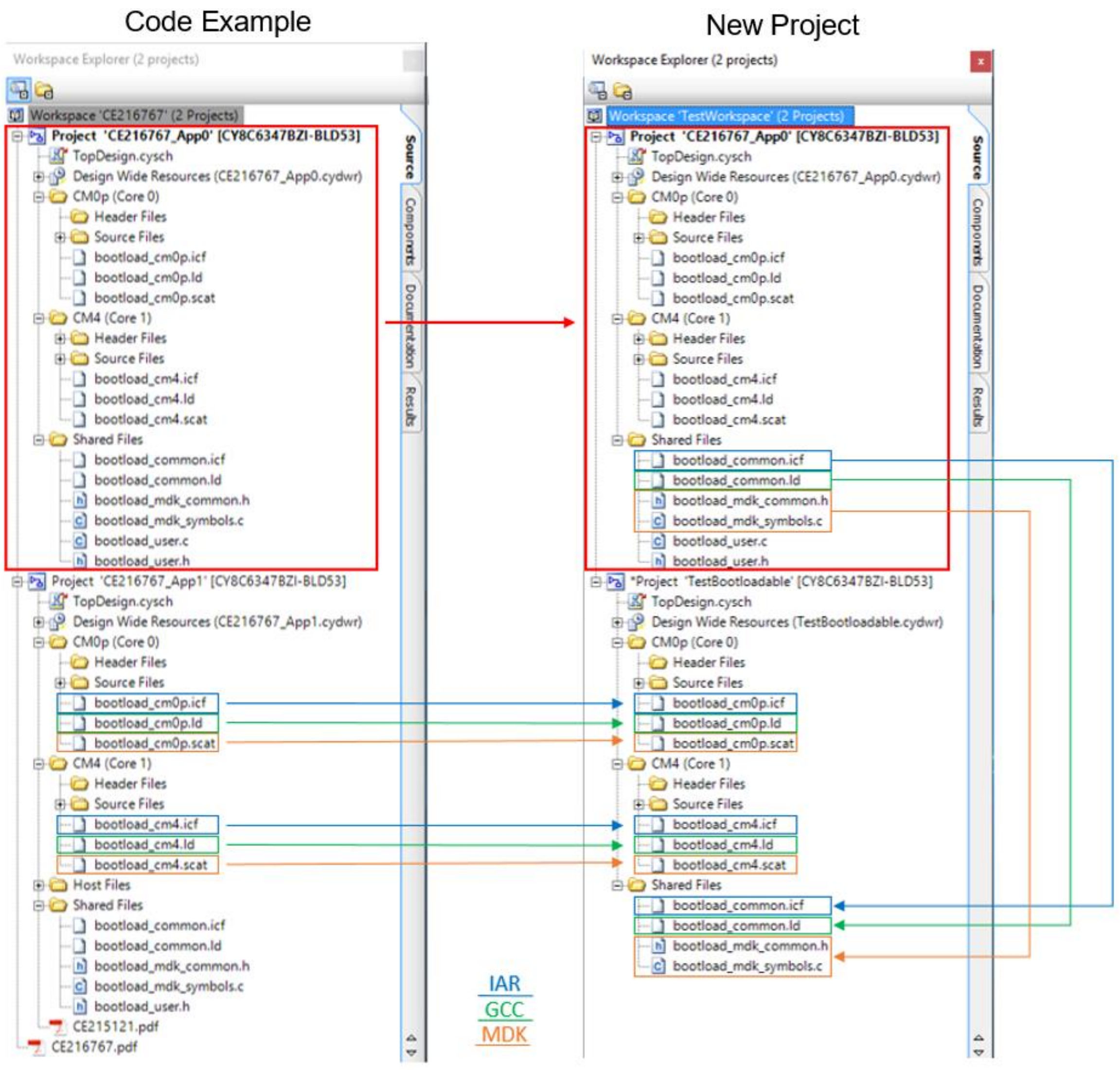
2. Configure your project as an installable application.

   2.1. Copy the core specific linker scripts according to your compiler, from App1 of this code example into your CM0p and CM4 project folders, respectively, as Figure 12 shows.

   2.2. Create a *Shared Files* folder within your project, if one does not exist already.

   2.3. Copy the previously edited common linker script within App0 into the *Shared Files* folder of your project. Figure 12 shows the folder structure and the files that need to be copied.

Figure 12. Bootloadable Project Structure



   2.4. Add the following line to your *main_cm4.c* file, after the `#include` statements:

```
CY_SECTION(".cy_app_signature") __USED const uint32_t cy_bootload_appSignature[1] = 0u;
```

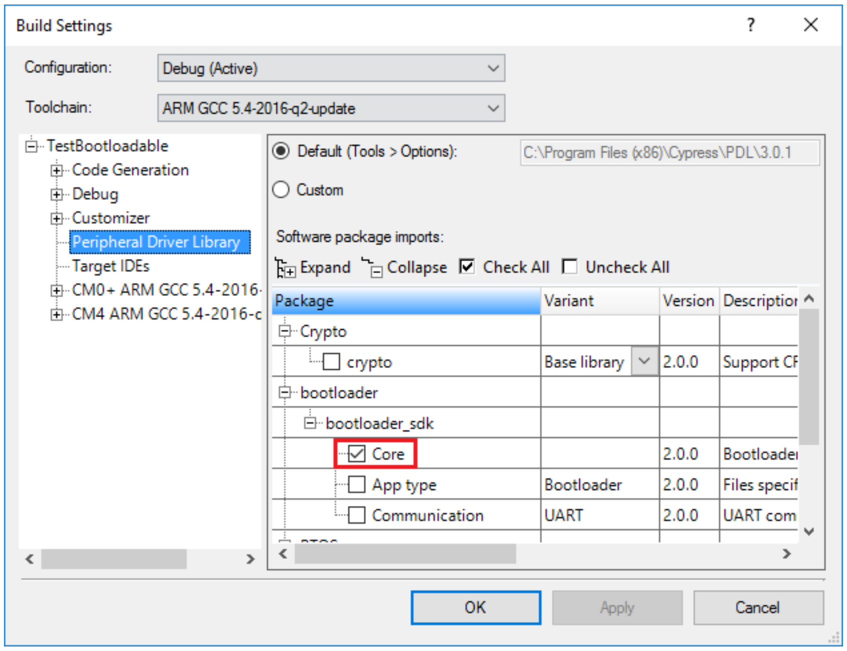
   2.5. If your project requires the CM4 core, change the line "`Cy_SysEnableCM4(CY_CORTEX_M4_APPL_ADDR);`" in your *main_cm0p.c* file to "`Cy_SysEnableCM4( (uint32_t) (&__cy_app_core1_start_addr) );`"

   2.6. Navigate to the location of this code example and open the *Bootloader_BLE_App1.cydsn* folder. Copy the *post_build_core1.bat* file and paste it into your bootloadable project folder.

   2.7. Right-click your project in the Workspace Explorer window, and select **Build Settings**.

2.8. Navigate to the **Peripheral Driver Library** and check the **Core** option, as Figure 13 shows.

Figure 13. Incorporating Bootloader SDK into the Bootloadable



Build Settings

Configuration: Debug (Active)

Toolchain: ARM GCC 5.4-2016-q2-update

- TestBootloadable
  - Code Generation
  - Debug
  - Customizer
  - Peripheral Driver Library
  - Target IDEs
  - CM0+ ARM GCC 5.4-2016-
  - CM4 ARM GCC 5.4-2016-c

● Default (Tools > Options): C:\Program Files (x86)\Cypress\PDL\3.0.1

○ Custom

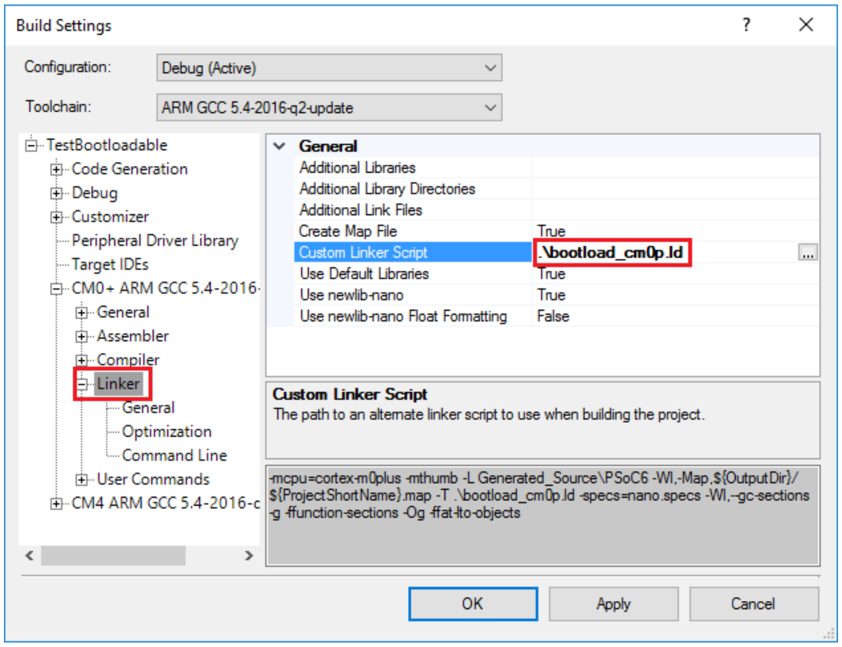
Software package imports:

Expand | Collapse | ☑ Check All | ☐ Uncheck All

Package	Variant	Version	Descriptior
Crypto			
☐ crypto	Base library	2.0.0	Support CF
bootloader			
bootloader_sdk			
☑ Core		2.0.0	Bootloader
☐ App type	Bootloader	2.0.0	Files specif
☐ Communication	UART	2.0.0	UART com

2.9. Set the custom linker scripts within the CM0+ and CM4 toolchain linker options. Select the cm0p linker script for the CM0+ core as shown in Figure 14 (figure shows the GCC linker script). Select the cm4 linker script for the CM4 core (not shown in figure). Use relative paths.

Figure 14. Project Build Settings for Custom Linker Scripts



Build Settings

Configuration: Debug (Active)

Toolchain: ARM GCC 5.4-2016-q2-update

- TestBootloadable
  - Code Generation
  - Debug
  - Customizer
  - Peripheral Driver Library
  - Target IDEs
  - CM0+ ARM GCC 5.4-2016-
    - General
    - Assembler
    - Compiler
    - Linker
      - General
      - Optimization
      - Command Line
    - User Commands
  - CM4 ARM GCC 5.4-2016-c

General	
Additional Libraries	
Additional Library Directories	
Additional Link Files	
Create Map File	True
Custom Linker Script	.\bootload_cm0p.ld
Use Default Libraries	True
Use newlib-nano	True
Use newlib-nano Float Formatting	False

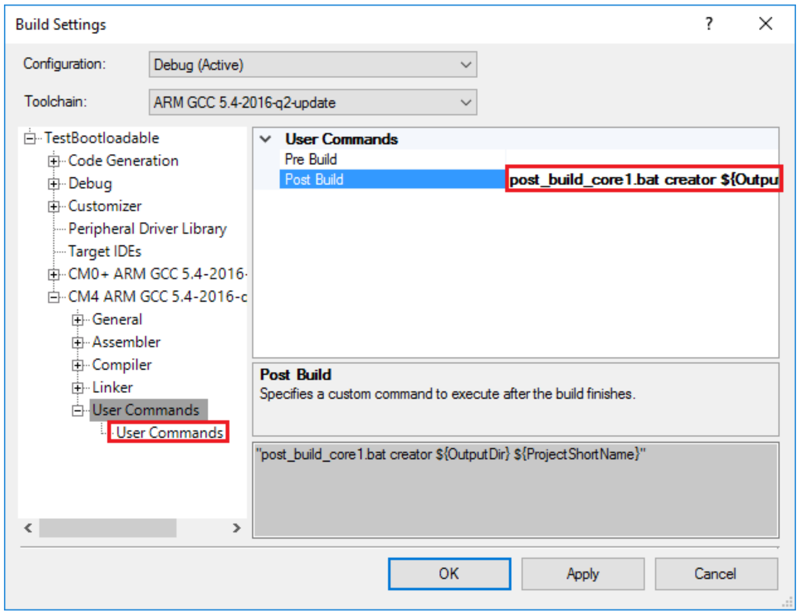
Custom Linker Script
The path to an alternate linker script to use when building the project.

```
-mcpu=cortex-m0plus -mthumb -L Generated_Source\PSoC6 -Wl,-Map,${OutputDir}/
${ProjectShortName}.map -T .\bootload_cm0p.ld -specs=nano.specs -Wl,--gc-sections
-g -ffunction-sections -Og -ffat-lto-objects
```

2.10. Navigate to the **User Commands** page within the CM4 toolchain options. Set the **Post Build** to the following line, as Figure 15 shows:

```
post_build_core1.bat creator ${OutputDir} ${ProjectShortName}
```

Figure 15. Post Build Command



post_build_core1.bat creator ${Outpu

Post Build
Specifies a custom command to execute after the build finishes.

"post_build_core1.bat creator ${OutputDir} ${ProjectShortName}"

2.11. Click **OK**.

2.12. Select **Build** > **Generate Application** to add the Bootloader SDK files to your project. Do not replace your linker scripts with the ones provided by the PDL.

2.13. Build the App0 project. The device can now be programmed with the bootloader.

2.14. Build your project. If a memory overflow occurs within a memory region, modify the common linker script to accommodate more flash/RAM memory. Changes must be made equally within the bootloader and bootloadable project.

Your application is now correctly configured as an installable application; it can be downloaded to the bootloader following the steps described in the Operation section of this document.

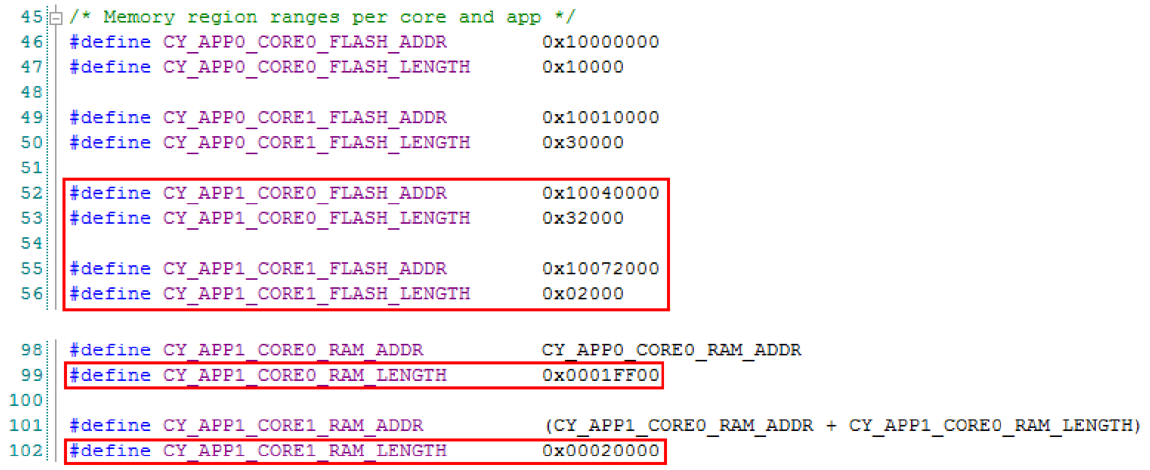
Related Documents

Application Notes	
AN210781 – Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity	Describes PSoC 6 MCU with BLE Connectivity devices and how to build your first PSoC Creator project.
AN213924 – PSoC 6 MCU Bootloader Software Development Kit (SDK) Guide	Provides information on how to use the Bootloader SDK, as well as information on bootloading in general.
PSoC 6 MCU Bootloader-Related Code Examples	
CE213903 Basic Bootloaders	Single application (bootloader app0 and application app1), using UART, I^{2}C, or SPI.
PSoC Creator Component Datasheets	
BLE	Provides information on Bluetooth Low Energy (BLE) settings and API.
UART	Provides information on UART settings and API.
Device Documentation	
PSoC 6 MCU: PSoC 63 with BLE Datasheets	PSoC 6 MCU: PSoC 63 with BLE Architecture Technical Reference Manual.
Development Kit Documentation	
PSoC 6 MCU Kits	

Appendix A: MDK and IAR Common Files

Figure 16 shows the fields that require modification in the *bootload_mdk_common.h* file.

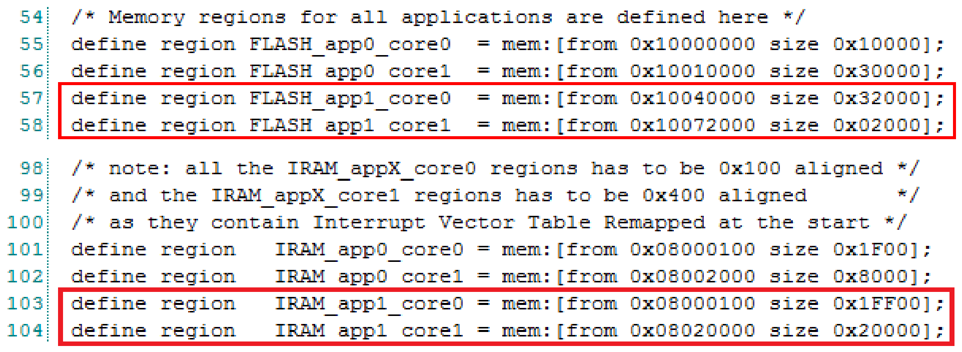
Figure 16. Modification of the MDK Common Header File



```
/* Memory region ranges per core and app */
#define CY_APP0_CORE0_FLASH_ADDR        0x10000000
#define CY_APP0_CORE0_FLASH_LENGTH      0x10000

#define CY_APP0_CORE1_FLASH_ADDR        0x10010000
#define CY_APP0_CORE1_FLASH_LENGTH      0x30000

#define CY_APP1_CORE0_FLASH_ADDR        0x10040000
#define CY_APP1_CORE0_FLASH_LENGTH      0x32000

#define CY_APP1_CORE1_FLASH_ADDR        0x10072000
#define CY_APP1_CORE1_FLASH_LENGTH      0x02000

#define CY_APP1_CORE0_RAM_ADDR          CY_APP0_CORE0_RAM_ADDR
#define CY_APP1_CORE0_RAM_LENGTH        0x0001FF00

#define CY_APP1_CORE1_RAM_ADDR          (CY_APP1_CORE0_RAM_ADDR + CY_APP1_CORE0_RAM_LENGTH)
#define CY_APP1_CORE1_RAM_LENGTH        0x00020000
```

Figure 17 shows the fields that require modification in the *bootload_common.icf* file.

Figure 17. Modification of the IAR Common Linker Script



```
/* Memory regions for all applications are defined here */
define region FLASH_app0_core0  = mem:[from 0x10000000 size 0x10000];
define region FLASH_app0_core1  = mem:[from 0x10010000 size 0x30000];
define region FLASH_app1_core0  = mem:[from 0x10040000 size 0x32000];
define region FLASH_app1_core1  = mem:[from 0x10072000 size 0x02000];

/* note: all the IRAM_appX_core0 regions has to be 0x100 aligned */
/* and the IRAM_appX_core1 regions has to be 0x400 aligned       */
/* as they contain Interrupt Vector Table Remapped at the start */
define region   IRAM_app0_core0 = mem:[from 0x08000100 size 0x1F00];
define region   IRAM_app0_core1 = mem:[from 0x08002000 size 0x8000];
define region   IRAM_app1_core0 = mem:[from 0x08000100 size 0x1FF00];
define region   IRAM_app1_core1 = mem:[from 0x08020000 size 0x20000];
```

Document History

Document Title: CE216767 – PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity Bootloader

Document Number: 002-16767

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	5880074	CFMM	10/20/2017	New code example
*A	5967782	CFMM	12/21/2017	Updated projects for PSoC Creator 4.2 ES100. Changed the name of the projects. Modified BLE settings for faster transmission. Changed LED behavior.
*B	6061562	MKEA	02/09/2018	Updated projects for Bootloader SDK 2.10. No document change.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

Arm® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6 MCU

Cypress Developer Community

Community Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709