CE220959 – PSoC 6 MCU with BLE Bootloader Using External Memory

Objective

This example demonstrates over the air (OTA) bootloading with a PSoC® 6 MCU with Bluetooth Low Energy (BLE) connectivity using an external memory. The application is downloaded into the external memory, verified, and afterwards copied into the internal flash memory for execution.

Overview

This example demonstrates how to use the Serial Memory Interface (SMIF) Component with a Quad SPI external memory to store and validate the received application. After validation, the application is copied into the internal flash memory for execution. If an error occurs during the application transmission, the application located within the internal flash memory is not corrupted.

Requirements

Tool: PSoC Creator™ 4.2; Peripheral Driver Library (PDL) 3.0.1 with Bootloader SDK 2.10; CySmart 1.2.1.711

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 (CY5677) provided with the CY8CKIT-062-BLE kit is required.

Software Setup

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

Operation

The bootloader can either download an application into the external flash memory or transfer control to a previously downloaded application. The following sections explain how to download an application and how to switch back from a bootloadable application 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.

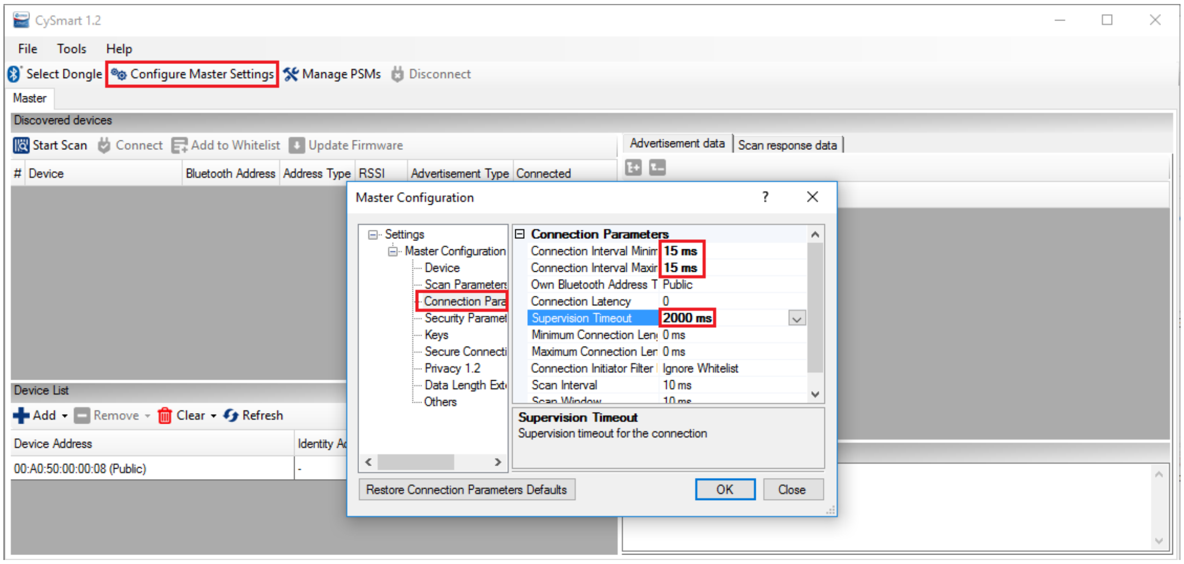
Note: Because there is no App1 on the device, the LED shows a warning sequence at start. For more information on the warning sequence, see Table 4.

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**.

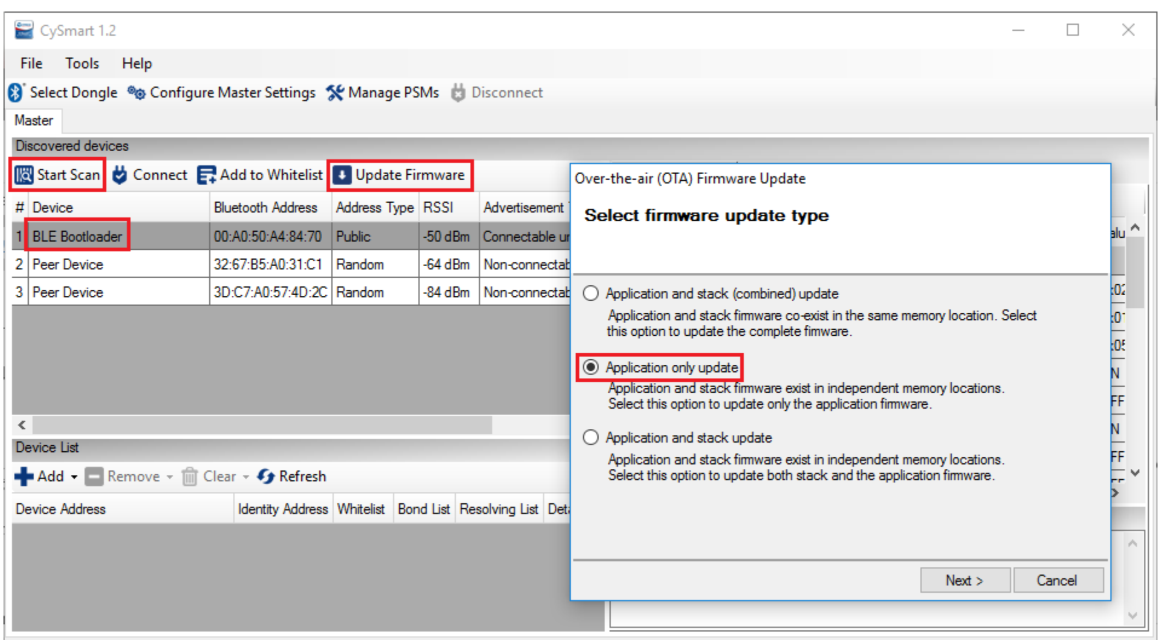
Note: Using a lower connection interval speeds up the application transmission at an increased risk of losing the connection.

Figure 1. CySmart Connection Parameters



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. Select the “Bootloader BLE” device when listed.
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



14. Browse and select the new bootloadable file (*Bootloader_BLE_External_Memory_App1.cyacd2)* located in the project folder *Bootloader_BLE_External_Memory_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 fast white, indicating that the validated App1 is being copied from external memory into the internal flash memory. This operation requires around 8 seconds.
18. Confirm that the LED is blinking green, indicating that App1 is running.
19. Switch to the bootloader using any of the methods shown below and repeat steps 11 – 18 to test the process of reinstalling App1.

The application has been copied to 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.

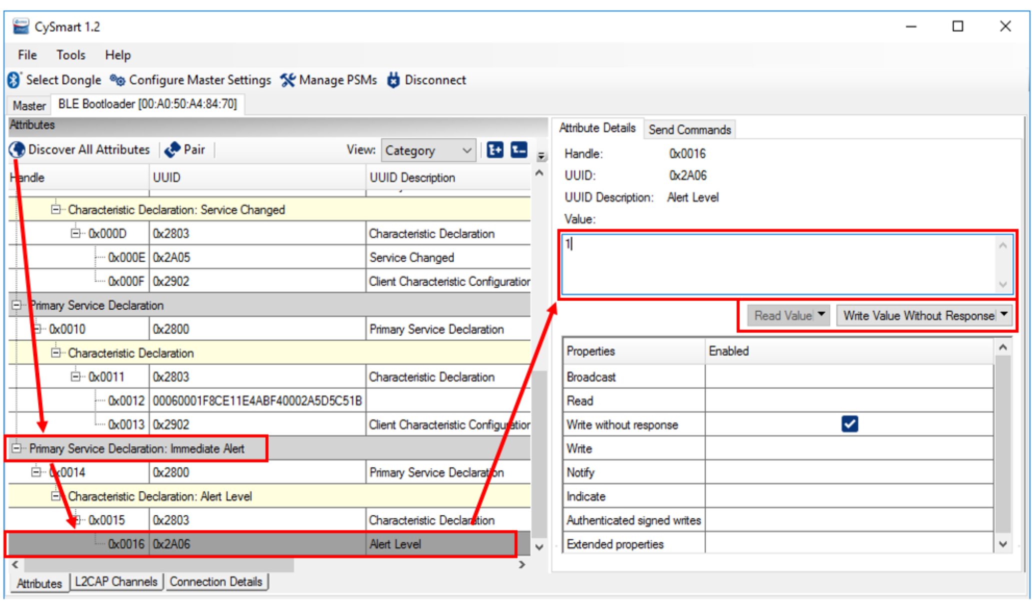
Switching Using IAS

1. Confirm that App1 is running. In CySmart, click **Start Scan**. Select the “BLE Keyboard” device when listed.
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



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

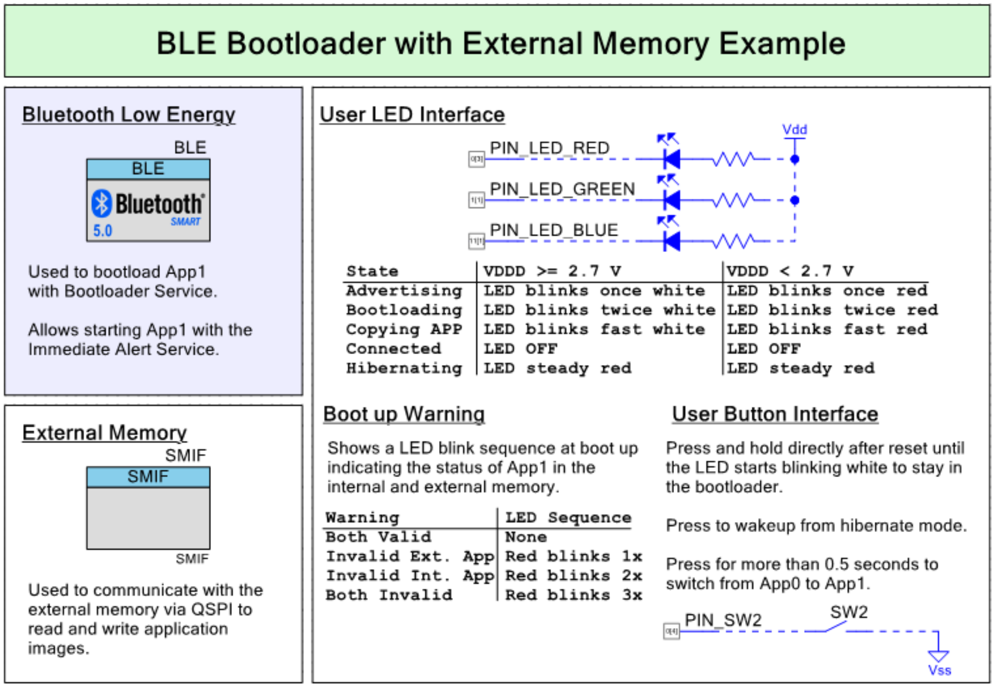
This example has two applications, called “App0” and “App1”. Each application is a separate PSoC Creator project with the following features:

- App0 is the bootloader application; it downloads and installs the bootloadable application (App1).
- The PSoC 63 MCU uses a SMIF Component to access the 64-MB Quad SPI memory available on the kit board. The application is stored and validated in the external memory and afterwards copied into the internal flash memory.
- A warning sequence at App0 startup indicates the status of App1 in the internal and external memory.
- Press the user button for more than 0.5 seconds within the bootloader to switch to App1. Switch occurs only if there is a valid App1 in the internal or external memory.[1]
- 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 in the internal memory, 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.

1 The switch operation may take longer if App1 is valid only in the external memory, because the application must be first copied into internal flash memory.

Figure 4 shows the PSoC Creator project schematic for App0.

Figure 4. App0 Schematic



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.

The LED boot up warning sequence can be turned off by changing the `SHOW_WARNING` definition in *main_cm4.c*.

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	LED status notification.
smif_mem.c / .h	Contains functions for initializing the SMIF Component, and read/writing to the external memory.
cy_smif_memconfig.c / .h	Contains the parameters and commands of the external flash memory on the kit board (S25FL512S).
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.

File	Description
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. ▪ `EraseExternalApp`. Erases the proper sectors of the external memory where the application is located. ▪ The redefined function `Cy_Bootload_ValidateApp` to add support for the 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 5 shows how the application is downloaded, stored, and copied. The Bluetooth host sends an application (App1) to the BLE bootloader (App0). The bootloader erases the application in the external memory when the bootloading operation starts. The write operations are redirected to the proper memory space within the external memory.

After the application has been completely downloaded, the bootloader verifies it in the external memory. If it is valid, the bootloader copies it into the proper memory space within the internal flash memory, overwriting the previous application.

This method allows the application stored within the internal flash memory to remain valid during the bootloading process. If a corrupted application is received, the application in the internal flash is not affected and can still be executed.

The downloaded application persists in the external memory after the copy operation. It may be used as backup or erased to use the memory for other purposes.

Figure 5. Application storing process.

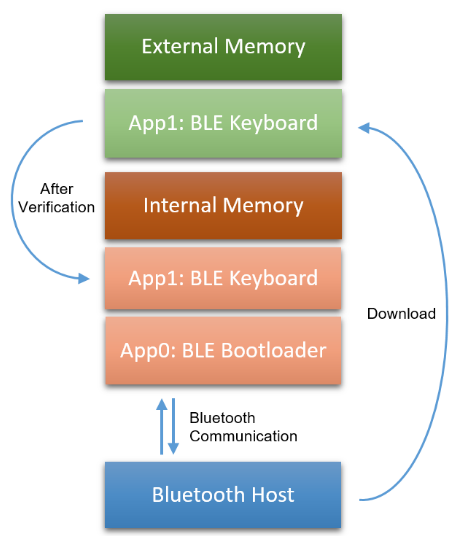


Figure 6 shows typical memory usage for each core in each application. This layout is done for PSoC 6 MCU devices with 1 MB flash memory. The BLE bootloader uses around 170 KB of flash memory to accommodate the BLE API. Note that in App0, the BLE stack is executed by the Cortex-M4 (CM4) CPU, and in App1 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.

The external memory on the kit board has a sector size of 256 KB. Because the erase operation is sector based, the entire sector of the external memory is erased every time a new application is received.

Note: App0 (BLE Bootloader) 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.

Figure 6. Memory Layout of Applications

Internal Flash	Description	Size
	Metadata copy row 0x100F FC00	512 B
	Metadata flash row 0x100F FA00	512 B
	Empty 0x1007 3800	560 KB
	App1, CM4 0x1007 2000	6 KB
	App1, CM0+ 0x1004 0000	200 KB
	App0, CM4 0x1001 0000	192 KB
	App0, CM0+ 0x1000 0000	64 KB

External Memory	Description	Size
	Empty 0x1808 0000	65,280 KB
	Unused 0x1803 3800	50 KB
	App1, CM4 0x1803 2000	6 KB
	App1, CM0+ 0x1800 0000	200 KB

To change the memory layout or usage, update the linker script files shown in Table 1 on page 5.

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.

The metadata row defined in *bootloader_user.c* was extended to include a virtual App2. The virtual App2 contains the metadata information of the received application. After the application has been completely received and copied into internal flash memory, the metadata information of App1 is updated with the virtual App2 metadata. The core specific linker scripts of App1 were declared with an AppID of 2, to use the virtual App2 metadata. For more information on the usage of the application metadata, see Appendix D of the PSoC 6 MCU Bootloader SDK Guide.

The default `Cy_Bootload_WriteData/ReadData` functions in *bootloader_user.c* were modified to add support for accessing the external memory. When writing to the first row of the external memory, `Cy_Bootload_WriteData` calls the EraseExternalApp function to erase the needed sectors for writing. The default `Cy_Bootload_ValidateApp` function in *cy_bootload.c* was redefined for validating applications in external memory.

To add support for external memory to your bootloader, copy the *bootloader_user.c* functions along with the *bootloader_user.h* file and SMIF files (*cy_smif_memconfig.c* /*.h* and *smif_mem.c* / *.h*) to your project. Update the AppID of the bootloadable application to 2. If a different memory interface, memory device or kit is used, see the Reusing This Example section.

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 in internal memory, 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
White	Blinking fast	An application is being copied from external memory into internal flash memory
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.

At startup, App0 shows the status of App1 in the internal and external memory on the LEDs as Table 4 shows. Note that this warning may be disabled with a `#define` statement.

Table 4. Boot up Warning Sequence

Warning	LED Sequence
Both Valid	None
Invalid App1 in external memory	Blinks once red
Invalid App1 in internal memory	Blinks twice red
Invalid App1 in internal and external memory	Blinks three times red

An application must be bootloaded if no valid App1 is present in either the internal or external memory.

If a valid App1 is only present in the external memory, it is not required to download it again. The application may be copied into the internal memory by pressing the user button for more than 0.5 seconds; the application starts afterwards. This situation may arise if a problem occurs during the copy operation (such as a power loss) or if the device is reprogrammed.

The application in external memory may become invalid if the download operation fails or if the user uses the external memory for other purposes. In both situations, the application in the internal flash memory remains valid. Keeping App1 on the external memory as a backup or erasing it to free memory is at the user's discretion.

App1 indicates the BLE keyboard status on the kit's RGB LED, as Table 5 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 5. 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 6 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 6. PSoC Creator Components

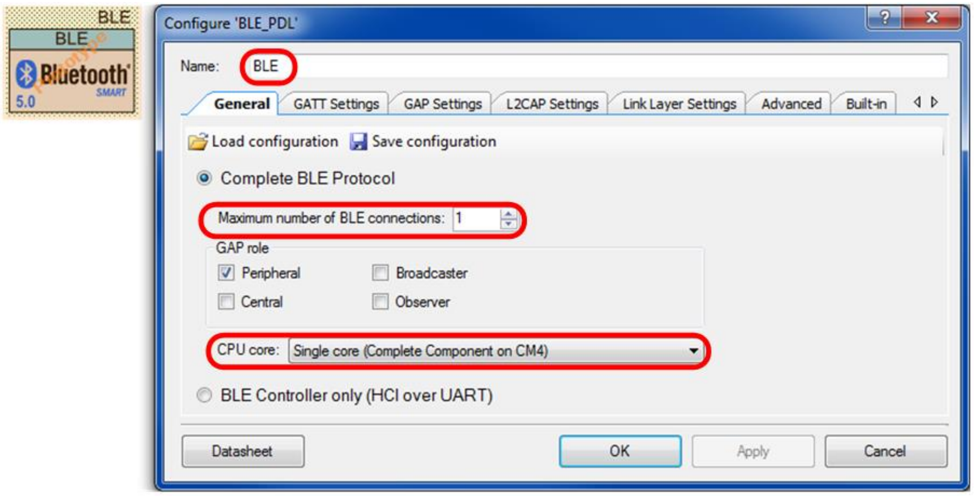
Component	Instance Name	Purpose	Non-default Settings
Bluetooth Low Energy	BLE	Provides communication between the PSoC 6 MCU device and the Bluetooth host for bootloading and app switching.	See BLE Component Configuration.
Serial Memory Interface (SMIF)	SMIF	Supports communication to the external QSPI memory to store the application.	SMIF dataline [2:3]: Checked. Generate code from cysmif file: Unchecked.
Pin	PIN_LED_RED PIN_LED_GREEN PIN_LED_BLUE PIN_SW2	LED Status notification and app switching.	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 7):
  - Maximum number of BLE Connections: 1
  - CPU core: Single core (Complete Component on CM4)

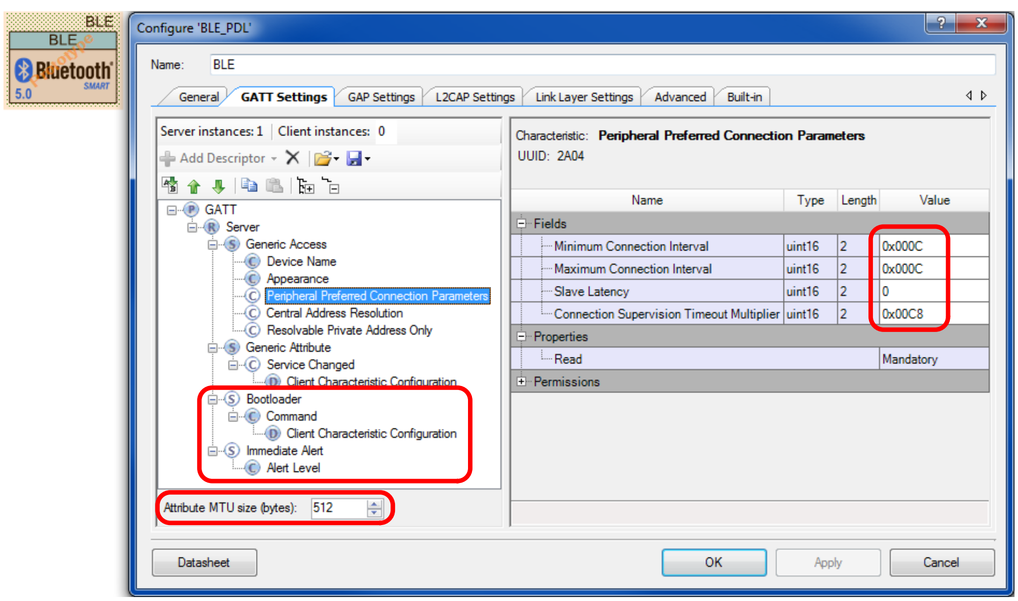
Figure 7. BLE Component, General Tab Configuration



- GATT Settings tab (see Figure 8):
    - 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

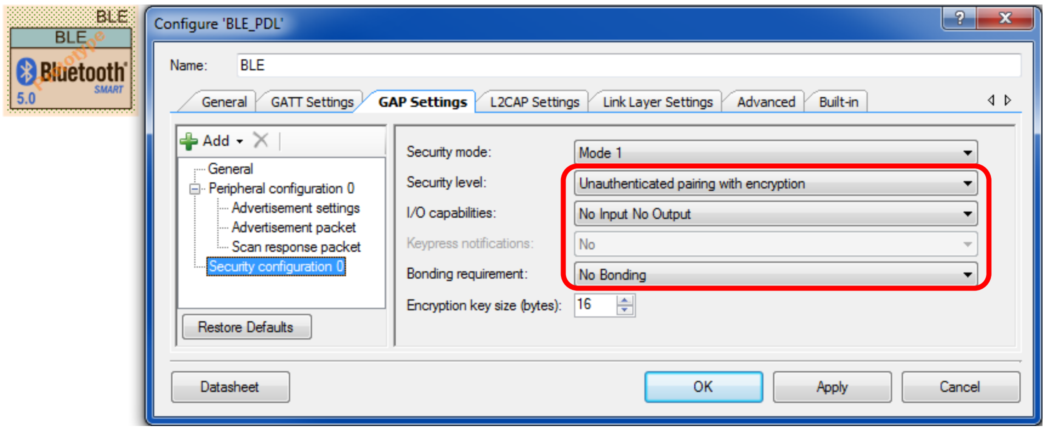
Figure 8. BLE Component, GATT Settings Tab Configuration



- GAP Settings tab:
    - Device Name: “BLE Bootloader”

- Peripheral Configuration 0, Advertisement packet: Local Name checked and set to Complete
- Security configuration 0 (see Figure 9), Security level: Unauthenticated pairing with encryption
- Security configuration 0, I/O capabilities: No Input No Output
- Security configuration 0, Bonding requirement: No Bonding

Figure 9. BLE Component, GAP Settings Tab Configuration



- 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 63 MCU BLE 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.

To use a different dual/quad/octal SPI external memory device, select the required datalines and check the **Generate code from cy_smif.cysmif file** option in the SMIF Component. Generate new memory configuration files (*cy_smif_memconfig.c* / *.h*) using the SMIF configuration tool. See the SMIF Component datasheet for more information. The SectorSize variable in the EraseExternalApp function of *bootload_user.c* must be updated. The functions in the *smif_mem.c* file may require modification depending on the addressing and initialization of the memory device.

For different memory interfaces, such as SPI and I2C, delete the SMIF component from the schematic and the files within the SMIF folder. Add the desired interface component to the schematic. The Read/WriteMemory, EraseExternalApp, and the `Cy_Bootload_ValidateApp` functions in the *bootloader_user.c* file must be updated to use the new memory interface.

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.*

The code to handle the external memory for bootloading purposes is located in the *bootloader_user.c / .h* files. This code can be copied into other bootloaders to add external memory functionality. For more information on the bootloader SDK files, see the PSoC 6 MCU Bootloader SDK Guide.

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^2C, or SPI.
CE216767 BLE Bootloader.	Single application (bootloader app0 and application app1), using BLE.
PSoC Creator Component Datasheets	
BLE	Provides information on Bluetooth Low Energy (BLE) settings and API.
SMIF	Provides information on the Serial Memory Interface settings and parameters.
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	

Document History

Document Title: CE220959 – PSoC 6 MCU with BLE Bootloader Using External Memory

Document Number: 002-20959

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	6002398	CFMM	02/23/2018	New code example

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