CE216767 – PSoC 6 MCU Bluetooth Low Energy (BLE) Device Firmware Update (DFU)

Objective

This example demonstrates device firmware update (DFU), also known as "bootloading", over the air (OTA) with 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 some basic Bluetooth services. It is based on CE215121 BLE HID Keyboard.

Requirements

Tool: PSoC Creator™ 4.2; Peripheral Driver Library (PDL) 3.1.0

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, with KitProg2 installed

Note: The PSoC 6 BLE Pioneer kit is shipped with KitProg2, and PSoC Creator only works with KitProg2. If your kit was upgraded to KitProg3 for use with ModusToolbox™, revert the kit to KitProg2 before using this code example. See ModusToolbox **Help** > **ModusToolbox IDE Documentation** > **User Guide**; section **PSoC 6 MCU KitProg Firmware Loader**.

Overview

This example demonstrates several basic DFU operations:

- Communicating with a host via BLE, and downloading an application
- Installing the downloaded application into flash memory
- Validating an application, and then transferring control to that application

There are two PSoC Creator project types, generally called "App0" and "App1". The projects have the following features:

- App0 performs the DFU operation; it downloads and installs App1 into device flash.
- Each project blinks a kit LED at a different rate or color making it easy to see which one is currently running.
- Transfer control between applications using the BLE Immediate Alert Service (IAS).

Other communication channels such as UART, I^{2}C, and SPI are demonstrated in other code examples; see Related Documents.

Hardware Setup

This example uses the kit's default configuration. See the kit guide to ensure the kit is configured correctly. For more information, see the KitProg2 User Guide.

Software Setup

To customize the DFU operation and enable DFU software development kit (SDK) features, update the `#define` statements as needed in the file *dfu_user.h*. The default settings can be used for most designs.

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

Operation

Build the Projects

Note: If you are using a version of the PDL that is different from that specified in the Requirements, PSoC Creator may have cleared the PDL software package import selections. Check the project **Build Settings** > **Peripheral Driver Library** > **DFU**. For more information, see PSoC Creator Help or AN213924, *PSoC 6 MCU Device Firmware Update Software Development Kit Guide*.

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**.

1. Using PSoC Creator, build the App0 project. For more information on building projects, see PSoC Creator Help.

   Note: For the MDK compiler, there is a known issue documented in PDL 3.1.0 Release Notes. To handle this issue, first select **Build** > **Generate Application**. Then in the *dfu_mdk_common.h* generated file, comment out the line containing "`__asm void cy_DFU_mdkAsmDummy(void);`". Finally, complete the project build by selecting **Build** > **Build <project name>**.

2. Build the App1 project. First, select **Build** > **Generate Application**. The installed linker script files are by default set up for application #0 (App0). For the App1 project, edit the files to change the application number:

   - For the GCC compiler, the following example shows edits for App1 in *dfu_cm0p.ld* and *dfu_cm4.ld*:

```
/*
* DFU SDK specific: aliases regions, so the rest of code does not use
* application specific memory region names
*/
REGION_ALIAS("flash_core0", flash_app1_core0);
REGION_ALIAS("flash",       flash_app1_core1);
REGION_ALIAS("ram",           ram_app1_core1);

/* DFU SDK specific: sets app Id */
__cy_app_id = 1;
```

   - For the MDK compiler, the following example shows edits for App1 in *dfu_cm0p.scat* and *dfu_cm4.scat*:

```
; Flash
#define FLASH_START                CY_APP1_CORE0_FLASH_ADDR
#define FLASH_SIZE                 CY_APP1_CORE0_FLASH_LENGTH

; Emulated EEPROM Flash area
#define EM_EEPROM_START            CY_APP1_CORE0_EM_EEPROM_ADDR
#define EM_EEPROM_SIZE             CY_APP1_CORE0_EM_EEPROM_LENGTH

; External memory
#define XIP_START                  CY_APP1_CORE0_SMIF_ADDR
#define XIP_SIZE                   CY_APP1_CORE0_SMIF_LENGTH

; RAM
#define RAM_START                  CY_APP1_CORE0_RAM_ADDR
#define RAM_SIZE                   CY_APP1_CORE0_RAM_LENGTH
```

And edits for App1 in *dfu_mdk_symbols.c*:

```
__cy_app_core1_start_addr   EQU __cpp(CY_APP1_CORE1_FLASH_ADDR)

/* Application number (ID) */
__cy_app_id                 EQU 1

/* CyMCUElfTool uses these to generate an application signature */
__cy_app_verify_start     EQU __cpp(CY_APP1_CORE0_FLASH_ADDR)
__cy_app_verify_length    EQU __cpp(CY_APP1_CORE0_FLASH_LENGTH +
                                    CY_APP1_CORE1_FLASH_LENGTH -
                                    __CY_BOOT_SIGNATURE_SIZE)
```

Then, complete the project build by selecting **Build** > **Build <project name>**.

App0 can either perform a DFU operation or transfer control to a previously downloaded application. The following sections explain how to download an application, and how to transfer control between the two applications.

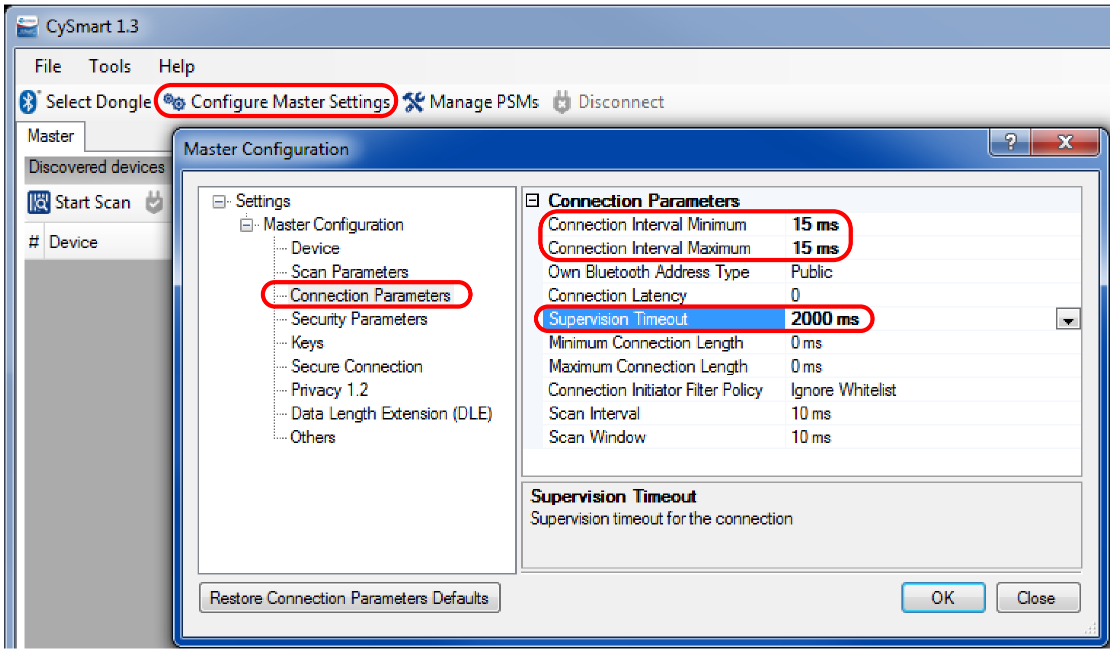
Test the Projects: Install App1

1. Connect the kit board to your PC using the provided USB cable.
2. Program the App0 project into the kit. For more information on device programming, see PSoC Creator Help.

   Confirm that the kit LED blinks white once every two seconds. This indicates that App0 is running.
3. Press and hold the kit button for at least half a second, and then release it. Confirm that nothing happens because App0 is the only application installed.
4. Connect the BLE USB dongle (CY5677) provided with the CY8CKIT-062-BLE kit to your computer.
5. Run the CySmart tool on your computer and connect to the BLE USB Dongle.
6. Click **Configure Master Settings**. 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



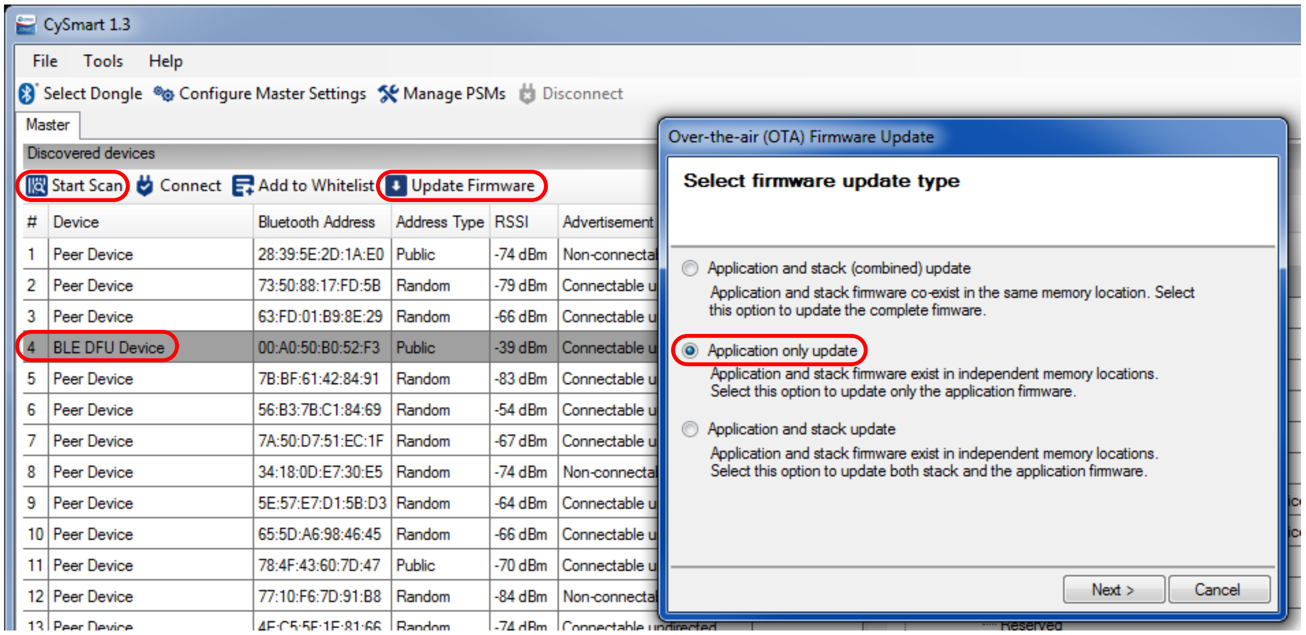
Connection Parameters	
Connection Interval Minimum	15 ms
Connection Interval Maximum	15 ms
Own Bluetooth Address Type	Public
Connection Latency	0
Supervision Timeout	2000 ms
Minimum Connection Length	0 ms
Maximum Connection Length	0 ms
Connection Initiator Filter Policy	Ignore Whitelist
Scan Interval	10 ms
Scan Window	10 ms

Supervision Timeout
Supervision timeout for the connection

7. Press the user button on the kit if the PSoC 6 MCU is hibernating (indicated by a steady red LED).

8. In CySmart, click **Start Scan** to start scanning for the DFU device, as Figure 2 shows. When "BLE DFU Device" is listed, click **Stop Scan**. When scanning stops, select that device. Click **Update Firmware** and choose the **Application only update** option. Click **Next**.

Figure 2. CySmart Firmware Update



#	Device	Bluetooth Address	Address Type	RSSI	Advertisement
1	Peer Device	28:39:5E:2D:1A:E0	Public	-74 dBm	Non-connectal
2	Peer Device	73:50:88:17:FD:5B	Random	-79 dBm	Connectable u
3	Peer Device	63:FD:01:B9:8E:29	Random	-66 dBm	Connectable u
4	BLE DFU Device	00:A0:50:B0:52:F3	Public	-39 dBm	Connectable u
5	Peer Device	7B:BF:61:42:84:91	Random	-83 dBm	Connectable u
6	Peer Device	56:B3:7B:C1:84:69	Random	-54 dBm	Connectable u
7	Peer Device	7A:50:D7:51:EC:1F	Random	-67 dBm	Connectable u
8	Peer Device	34:18:0D:E7:30:E5	Random	-74 dBm	Non-connectal
9	Peer Device	5E:57:E7:D1:5B:D3	Random	-64 dBm	Connectable u
10	Peer Device	65:5D:A6:98:46:45	Random	-66 dBm	Connectable u
11	Peer Device	78:4F:43:60:7D:47	Public	-70 dBm	Connectable u
12	Peer Device	77:10:F6:7D:91:B8	Random	-84 dBm	Non-connectal
13	Peer Device	4F:C5:5F:1F:81:66	Random	-74 dBm	Connectable undirected

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 fimware.

[✓ 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.

9. Browse and select the App1 file (*PSoC6DfuBleApp1.cyacd2)* located in the project folder *PSoC6DfuBleApp1 > CortexM4 > [compiler name] > Debug*. This file is generated when App1 is built. Click **Update**.
10. Wait for the device firmware to be updated. While the firmware is downloading, the kit LED blinks white twice every two seconds.
11. After download is complete, confirm that the LED is blinking green, indicating that App1 is running.
12. Switch to App0 using any of the methods shown below. Repeat steps 8 – 11 to test the process of reinstalling App1.

Test the Projects: Transfer Applications

App1 has been installed into the internal flash memory, and automatically starts whenever the device is powered ON or reset. The following steps show multiple ways to transfer between the applications.

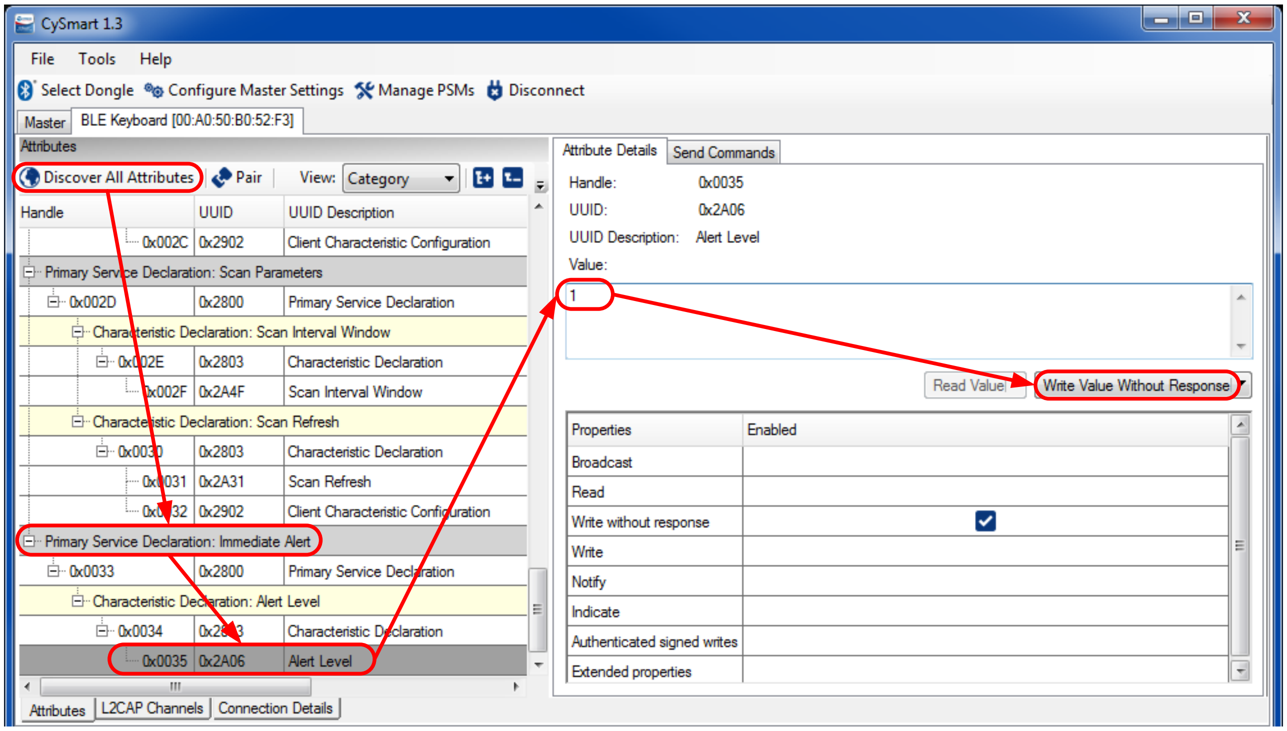
Transfer Using IAS

1. In CySmart, click **Start Scan**. When listed, select "BLE DFU Device" or "BLE Keyboard" for App0 and App1 respectively.
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. For App1, click **No** when prompted to add the device to the resolving list. For App0, just click **OK**.

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

4. Click **Discover All Attributes**. 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. Transferring Control Between Applications Using IAS


CySmart 1.3
File Tools Help
Select Dongle Configure Master Settings Manage PSMs Disconnect
Master BLE Keyboard [00:A0:50:B0:52:F3]
Attributes
Discover All Attributes Pair View: Category
Attribute Details Send Commands
Read Value Write Value Without Response
Attributes L2CAP Channels Connection Details

Handle	UUID	UUID Description
0x002C	0x2902	Client Characteristic Configuration
Primary Service Declaration: Scan Parameters		
0x002D	0x2800	Primary Service Declaration
Characteristic Declaration: Scan Interval Window		
0x002E	0x2803	Characteristic Declaration
0x002F	0x2A4F	Scan Interval Window
Characteristic Declaration: Scan Refresh		
0x0030	0x2803	Characteristic Declaration
0x0031	0x2A31	Scan Refresh
0x0032	0x2902	Client Characteristic Configuration
Primary Service Declaration: Immediate Alert		
0x0033	0x2800	Primary Service Declaration
Characteristic Declaration: Alert Level		
0x0034	0x2803	Characteristic Declaration
0x0035	0x2A06	Alert Level

Handle: 0x0035
UUID: 0x2A06
UUID Description: Alert Level
Value: 1

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

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

This method can also be used to transfer control from App0 to App1. In step 1, select “BLE DFU Device”.

Transfer Using the Kit Buttons

This method only transfers control from App0 to App1.

1. Confirm that App1 is running.
2. Press the kit reset and user buttons – SW1 and SW2 – at the same time.
3. Release the reset button (SW1).
4. Wait until the LED starts blinking white before releasing the user button. App0 is now running.

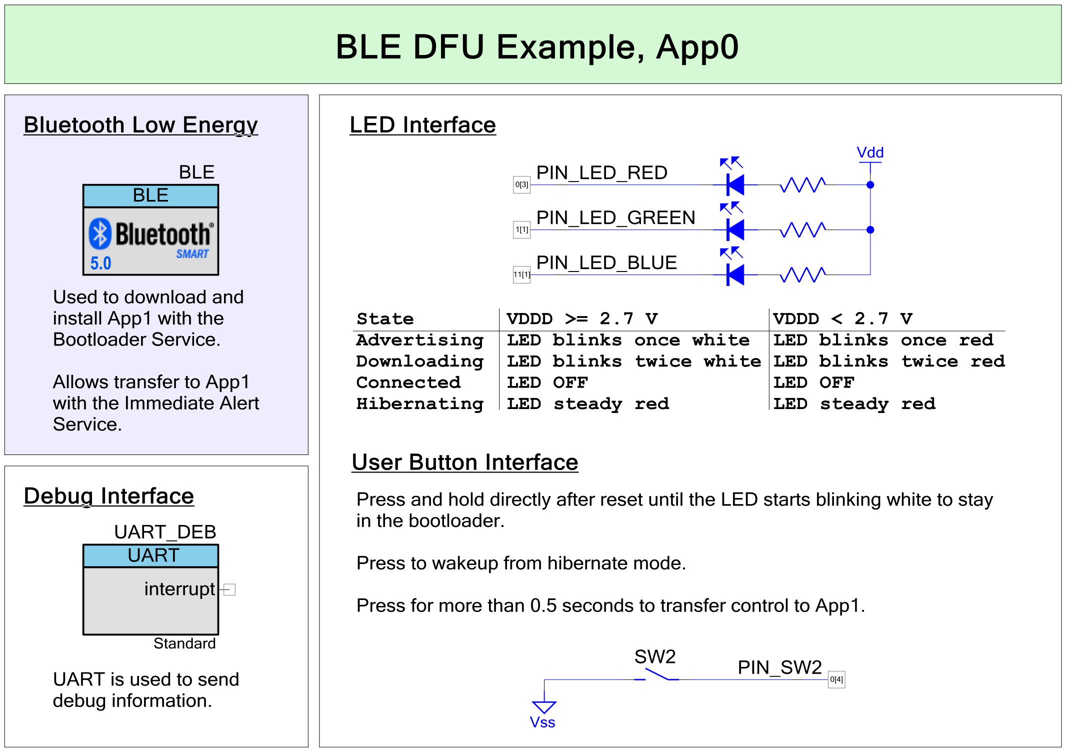
Design and Implementation

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

- App0 performs the DFU operation; it installs (or re-installs) the application App1.
- After downloading App1, both applications reside in flash memory; see Memory Layout. Whenever the device is powered ON or reset, App0 runs first. App0 checks whether a valid App1 exists, and if so transfers control to it. This makes it appear as if App1 is the only application in flash. For more information, see Design Firmware.
- You can transfer control between applications using the Bluetooth IAS, or using the kit buttons.
- After 300 seconds of inactivity within App0, it transfers control to App1. If there is no valid App1, App0 hibernates.
- App1 demonstrates several Bluetooth services. It is a lightly modified version of CE215121 BLE HID Keyboard with DFU SDK support added.

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



BLE DFU Example, App0

Bluetooth Low Energy

Used to download and install App1 with the Bootloader Service.

Allows transfer to App1 with the Immediate Alert Service.

Debug Interface

UART is used to send debug information.

LED Interface

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

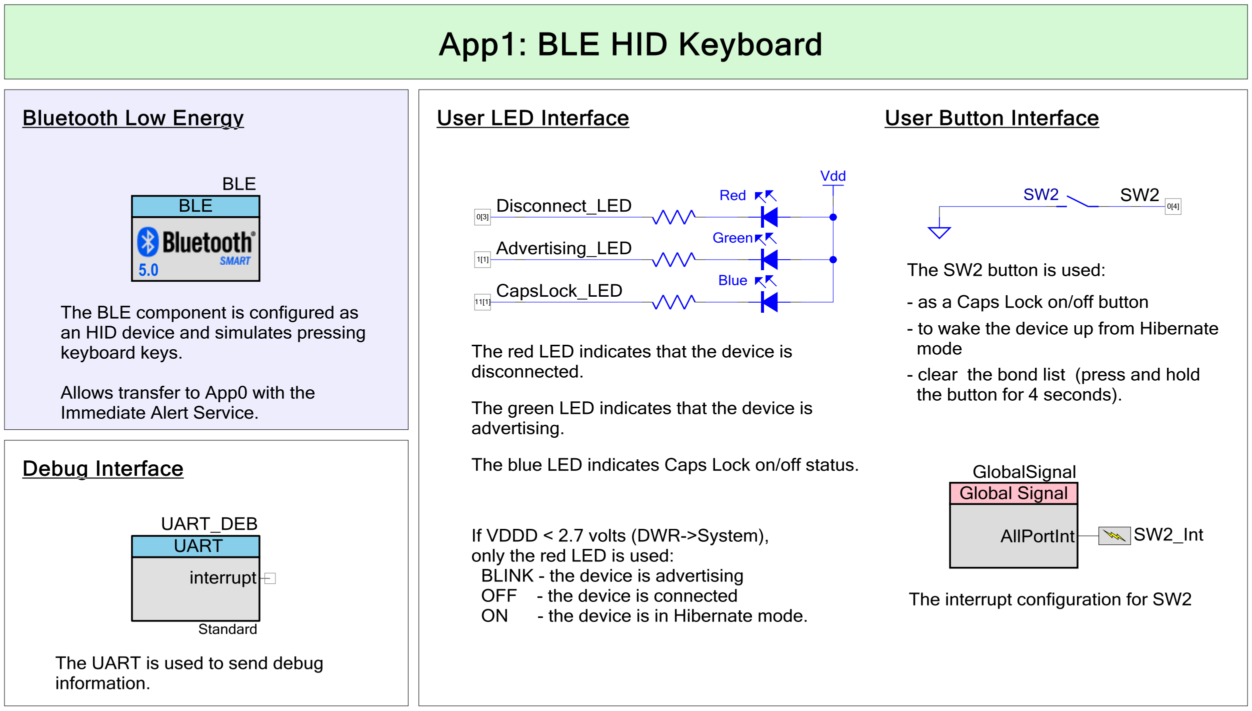
User Button Interface

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 transfer control to App1.

Figure 5. App1 Schematic



App1: BLE HID Keyboard

Bluetooth Low Energy

The BLE component is configured as an HID device and simulates pressing keyboard keys.

Allows transfer to App0 with the Immediate Alert Service.

Debug Interface

The UART is used to send debug information.

User LED Interface

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

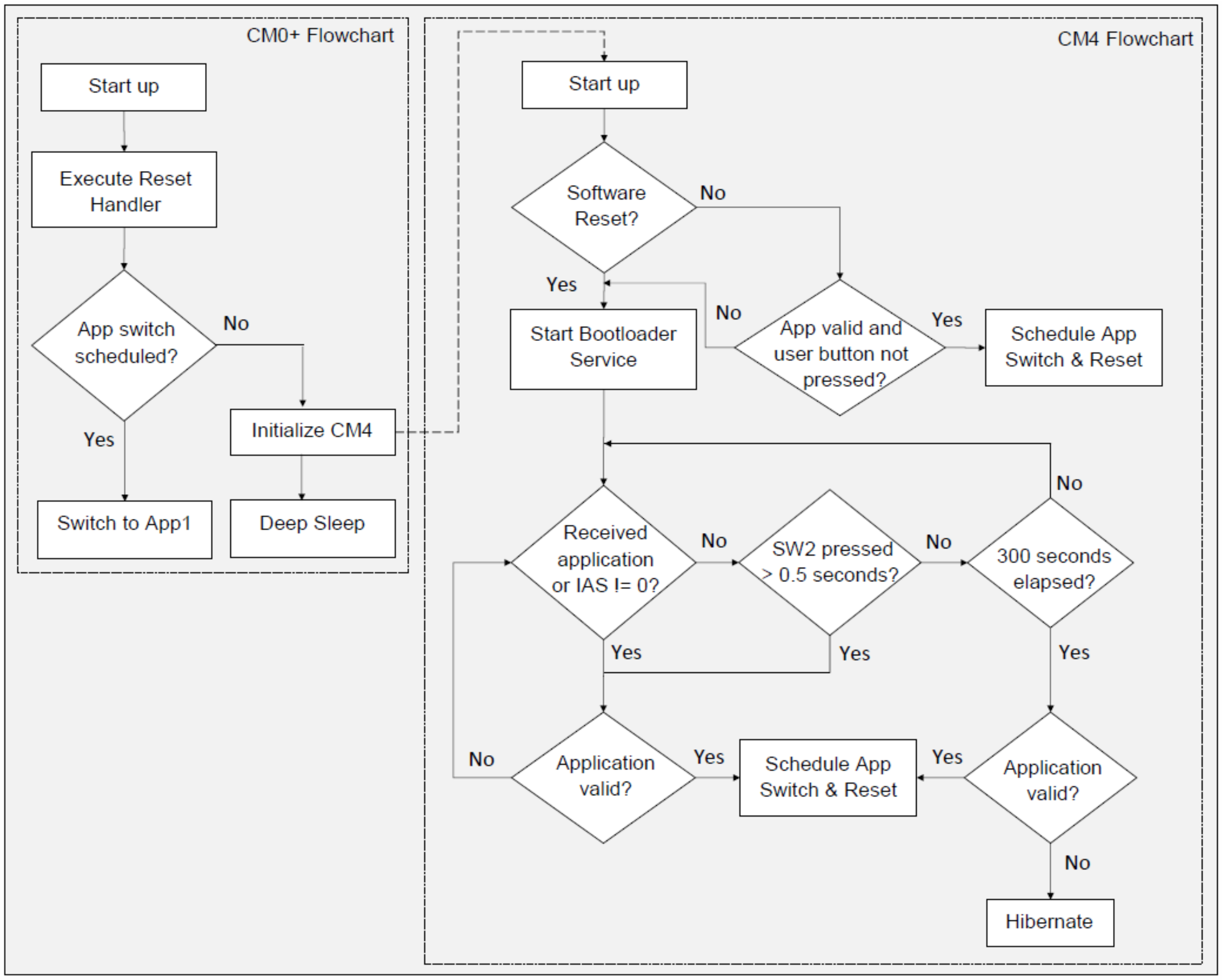
The SW2 button is used:

- as a Caps Lock on/off button
- to wake the device up from Hibernate mode
- clear the bond list (press and hold the button for 4 seconds).

The interrupt configuration for SW2

Figure 6 shows the firmware flow of App0. Both CPUs in PSoC 6 MCU are used; for more information see Dual CPUs.

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 DFU projects, see AN213924, *PSoC 6 MCU Device Firmware Update Software Development Kit Guide*.

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 (CM4) and a Cortex-M0+ (CM0+). See Table 2 for specific tasks for each core.
ias.c / .h	BLE IAS files. Used to implement IAS for communication between BLE client and server. When the IAS receives a non-zero value, it transfers control to the other application.
debug.c / .h	UART `printf` implementation and LED status notification.
cy_dfu.h, .c	The DFU SDK files.
cy_dfu_bwc_macro.h	Contains macros for backward compatibility to facilitate porting of legacy bootloader projects.
dfu_user.h	Contains `#define` user-editable statements that control the operation and enable features in the SDK.
dfu_user.c	Contains user functions required by the SDK: ▪ Five functions that control communications with the DFU host. These are also called transport functions. ▪ Two functions – `ReadData()` and `WriteData()` – that control access to internal or external memory
transport_ble.h, .c	Contains DFU transport functions for the BLE Component. These functions are typically called by the transport functions in *dfu_user.c.*
dfu_cm4.ld, dfu_cm0p.ld	Custom GCC linker scripts. In each project, these files replace the auto-generated linker script files. These files locate the code and data sections for each of the CPU cores as well as other sections. These files include a “common” section that must be the same in both files.
dfu_mdk_common.h dfu_mdk_symbols.c	Similar in function to the GCC and IAR common linker scripts, for MDK. The MDK linker does not support includes in the *.scat* files, so these files exist to create the necessary defines. These files are user-editable – they control the memory layout and the locations in memory for each application, and the code and data for each CPU core in each application. These files are common to all applications.
dfu_cm4.scat, dfu_cm0p.scat	Custom MDK linker scripts. In each application, these files replace the auto-generated linker script files. These files locate the code and data sections for each of the CPU cores as well as the DFU code and other regions.
dfu_cm4.icf, dfu_cm0p.icf	Custom IAR linker scripts. In each application, these files replace the auto-generated linker script files. These files locate the code and data sections for each of the CPU cores as well as the DFU code and other regions. These files include a “common” section that must be the same in both files.
post_build_core1.bat	Batch file to create the downloadable application image (*.cyacd2* file) for App1.

Memory Layout

Figure 7 shows the typical memory usage for each CPU core in each project. This layout is done for PSoC 6 MCU devices with 1 MB flash and 288 KB SRAM.

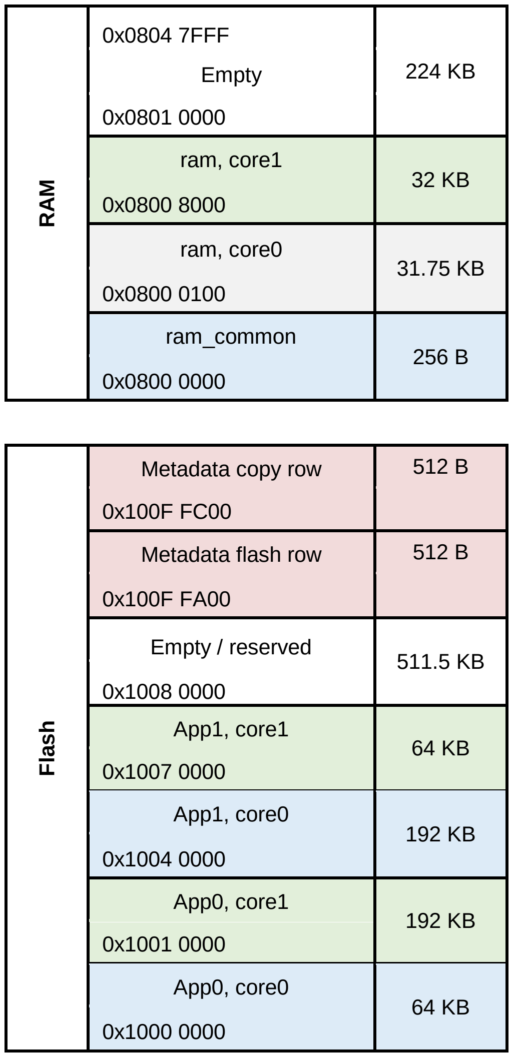
Approximately 170 KB of flash is required for the BLE API. Note that in App0, the BLE stack is executed by the Cortex-M4 (CM4) CPU, and in App1 the BLE stack is executed by the Cortex-M0+ (CM0+) CPU. For more information on the memory layout, see AN213924, *PSoC 6 MCU Device Firmware Update Software Development Kit Guide*.

App0 always starts at the beginning of device flash at address 0x1000 0000. For more information on the device memory map, see the device datasheet. App1 starts at the next 256 KB boundary. Each app has defined flash areas for each CPU core: core0 (CM0+) and core1 (CM4).

The RAM is shared by App0 and App1, with a common area used by both projects. Each app has defined RAM areas for each CPU core: core0 (CM0+) and core1 (CM4).

To change the memory layout or usage, update the linker script files shown in Table 1. The linker scripts can also be modified to define dedicated regions of memory for each application.

Figure 7. Memory Layout of Applications



Design Considerations

Note: App0 and App1 projects must be built with the same toolchain (GCC or MDK); application transfer may fail otherwise. Check the **Build Settings** for each project.

Dual CPU

PSoC 6 MCU has two CPU cores: CM4 and CM0+. An application can include code for one or both CPUs. For more information, see AN215656, *PSoC 6 MCU Dual-CPU System Design*.

In these projects, CPUs in each application do as Table 2 shows. For details, see Appendix A, Code Theory of Operation. This can easily be changed so that either core can run any of the tasks, including DFU.

Table 2. Core Tasks per Application

Application	Cortex-M0+	Cortex-M4
App0	Executes first at device reset. Reset handler controls application transfer. Note that if application transfer does occur, it occurs before the CM4 is turned ON. Turns ON CM4 core. Does nothing else.	Blinks the LED once per two seconds. Downloads and installs App1. Monitors the button. After the DFU operation, or when the IAS alert level is greater than zero, initiates transfer to App1, through a software reset.
App1	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.

Linker Script Modifications

The default 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 DFU time. The modifications were done to linker script files for both App0 and App1.

LED Status

App0 indicates the current DFU state on the kit's RGB LED, as Table 3 shows:

Table 3. LED Status for App0

Color	State	Description
White	Blinks once every 2 seconds	Device is advertising
None	OFF	Device is connected, but not receiving an application
White	Blinks twice every 2 seconds	An application is being received
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.

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.

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

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

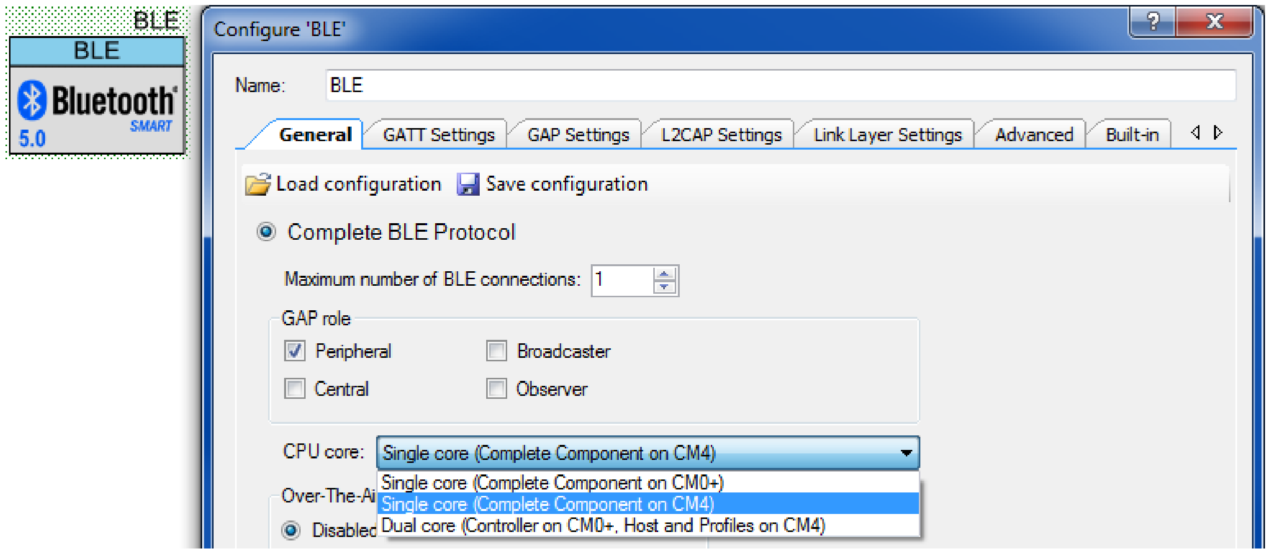
Component	Instance Name	Purpose	Non-default Settings
Bluetooth Low Energy	BLE	Provides communication between the PSoC 6 MCU device and the BLE host.	See BLE Component Configuration.
UART	UART_DEB	Sends BLE debug information.	Interrupt mode external.
Pin	PIN_LED_RED PIN_LED_GREEN PIN_LED_BLUE PIN_SW2	Status notification on LEDs Button monitoring	HW Connection: Unchecked. External Terminal: Checked. LED Pins Drive Mode: High Impedance Digital. SW2 Pin 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): The CPU core is set to **CPU core: Single core (Complete Component on CM4)** for App0, **CPU core: Single core (Complete Component on CM0+)** for App1.

Figure 8. BLE Component, General Tab Configuration



Name: BLE

Complete BLE Protocol

Maximum number of BLE connections: 1

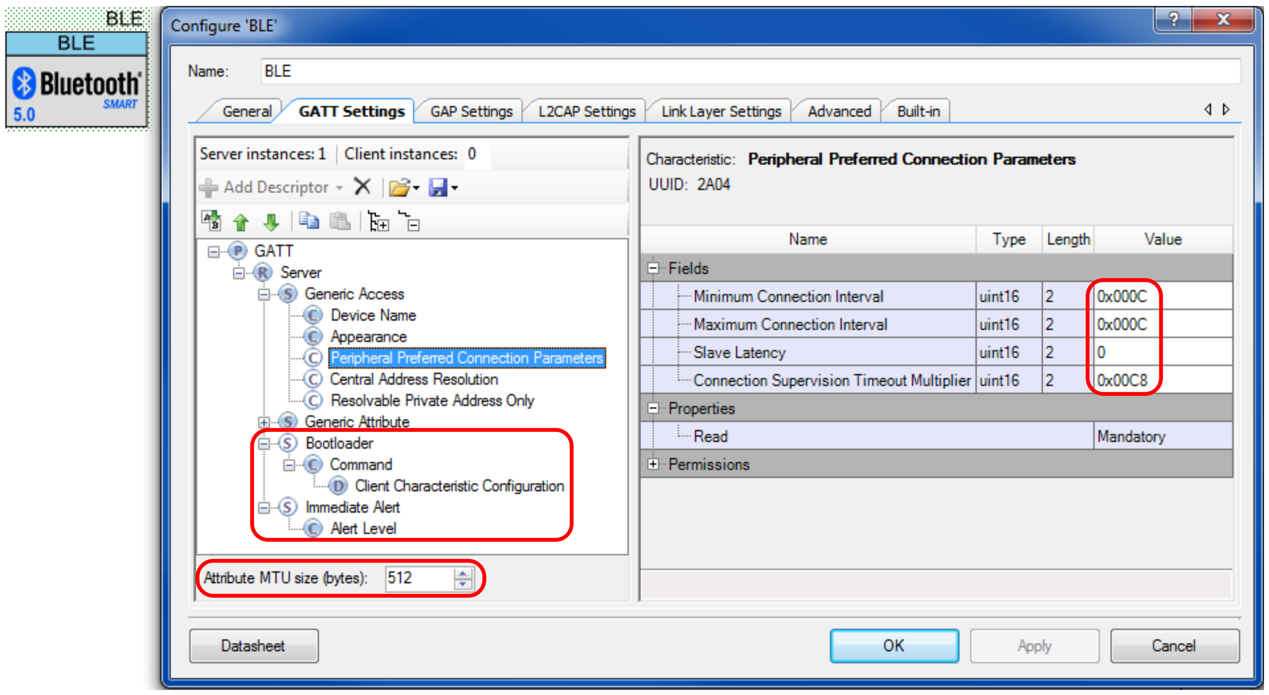
GAP role
[✓] Peripheral
[ ] Broadcaster
[ ] Central
[ ] Observer

CPU core: Single core (Complete Component on CM4)
- Single core (Complete Component on CM0+)
- Single core (Complete Component on CM4)
- Dual core (Controller on CM0+, Host and Profiles on CM4)

- **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 DFU time.
  - Add **Bootloader** service
  - Add **Immediate Alert** service
  - **Attribute MTU size (bytes)**: 512

Figure 9. BLE Component, GATT Settings Tab Configuration



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
    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 DFU Device”
  - Peripheral Configuration 0, Advertisement packet: Local Name checked and set to Complete
  - Security configuration 0, Security level: Unauthenticated pairing with encryption
  - Security configuration 0, I/O capabilities: No Input No Output
  - Security configuration 0, Bonding requirement: No Bonding
- **Link Layer Settings** tab:
  - Link layer max TX and RX payload size (bytes): 251

Reusing This Example

This example is designed for the kit indicated in Related Hardware. To port the design to a different PSoC 6 MCU device, kit, or both, 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 6 MCU devices, port the code from *main_cm4.c* to *main.c,* and copy the `Cy_OnResetUser` function from *main_cm0p.c* to *main.c*. For App1, note in Table 2 that the CM0+ performs all the tasks; port all the code from *main_cm0p* to *main.c*.

You can add BLE OTA DFU capability to an existing project. For detailed information on how to create, configure, and use a DFU project, see AN213924, *PSoC 6 MCU Device Firmware Update Software Development Kit Guide*.

In some cases, a resource used by a code example is not supported on another device. In that case, the example will not work. If you build the code targeted at such a device, you will get errors. See the device datasheet for information on which resources a device supports.

Related Documents

PSoC 6 DFU-Related Application Notes	
AN213924 – PSoC 6 MCU Device Firmware Update Software Development Kit Guide	Provides comprehensive information on how to use the Device Firmware Update (DFU) Software Development Kit (SDK)
Other Application Notes	
AN221774 – Getting Started with PSoC 6 MCU	Describes PSoC 6 MCU devices and how to build your first ModusToolbox or PSoC Creator project
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.
PSoC 6 MCU DFU-Related Code Examples	
CE213903 – PSoC 6 MCU Basic DFU	Single-application DFU using UART, I^{2}C, or SPI
CE220959 – PSoC 6 MCU BLE DFU with External Memory	Similar to the BLE DFU; the downloaded application is temporarily saved in external memory and then copied to its final destination
CE220960 – PSoC 6 MCU BLE DFU with Upgradeable Stack	Similar to the BLE DFU; the BLE stack can be updated in addition to the application
CE221984 – PSoC 6 MCU Dual-Application I2C DFU	Similar to the basic I^{2}C DFU; manages two downloaded applications instead of one
CE222802 – PSoC 6 MCU Encrypted DFU	Similar to the basic UART DFU; the application is digitally signed and encrypted
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 Kits	
CY8CKIT-062-BLE	PSoC 6 BLE Pioneer Kit
CY8CPROTO-063-BLE	PSoC 6 BLE Prototyping Kit

Tool Documentation	
PSoC Creator	PSoC Creator enables concurrent hardware and firmware editing, compiling and debugging of PSoC devices. Applications are created using schematic capture and over 150 pre-verified, production-ready peripheral Components. Look in the downloads tab for Quick Start and User Guides.
ModusToolbox	ModusToolbox simplifies development for IoT designers. It delivers easy-to-use tools and a familiar microcontroller (MCU) integrated development environment (IDE) for Windows, macOS, and Linux.
Peripheral Driver Library (PDL)	Installed by PSoC Creator 4.2. Look in the <PDL install folder>/doc for the User Guide and the API Reference

Cypress Resources

Cypress provides a wealth of data at www.cypress.com to help you to select the right device, and quickly and effectively integrate the device into your design.

For the PSoC 6 MCU devices, see KBA223067 in the Cypress community for a comprehensive list of PSoC 6 MCU resources.

Appendix A: Code Theory of Operation

This section describes in detail how the code example source code implements the functions listed in Table 2 on page 10.

File: *main_cm0p.c*:

Function `main()`:

Wakes up from device Hibernate mode by calling functions in the power management (`SysPm`) API.

Calls `Cy_SysEnableCM4((uint32_t)(&__cy_app_core1_start_addr))`

`__cy_app_core1_start_addr` is defined in *dfu_cm0p.ld*.

Then, goes into device Deep Sleep mode, by calling a function in the `SysPm` API.

Function `Cy_OnResetUser()`:

Called by the startup reset handler. Calls `Cy_DFU_OnResetApp0()`, which is defined in *cy_dfu.c*. This is the mechanism by which control is transferred to another application after device software reset.

File: *main_cm4.c*:

Has `#defines` for BLE parameters.

Function `main()`:

Has local variables:

```
const uint32_t paramsTimeout = 20u; /* timeout, in milliseconds */
cy_stc_dfu_params_t dfuParams; /* configures DFU */
cy_en_dfu_status_t status; /* Status codes from DFU SDK API */
bool buttonReleased = false; /* user button management */
uint32_t state; /* NONE, UPDATING, FINISHED, or FAILED */

cy_en_ble_api_result_t apiResult; /* BLE parameters */
cy_stc_ble_stack_lib_version_t stackVersion;

uint32_t count = 0; /* counts seconds */
uint32_t ledTimer = 0; /* for pulsing the LED */

CY_ALIGN(4) static uint8_t buffer[CY_DFU_SIZEOF_DATA_BUFFER]; /* flash row data */
CY_ALIGN(4) static uint8_t packet[CY_DFU_SIZEOF_CMD_BUFFER]; /* host packet */
```

Initializes debug UART and LED.

Initializes `dfuParams` with timeout, and two buffer addresses.

Calls `Cy_DFU_Init()` (in *cy_dfu.c*), which sets the `state` to NONE.

Calls `HandleMetadata()`, which is part of the code example, not the SDK. It updates metadata (MD) and MD copy rows of flash, or initializes the MD row.

Calls `CopyRow()`, which is part of the code example, not the SDK. Reads a source row and writes it to a destination row. Does a compare before writing, to avoid an unnecessary row write.

If the reset reason (`Cy_SysLib_GetResetReason()`, *cy_syslib.c*) was NOT a software reset (SRES), and the kit user button has NOT been pressed for two seconds (`IsButtonPressed()`, in this file):

Validates App1 (`Cy_DFU_ValidateApp(1u)`, *cy_dfu.c*). If OK, clears the reset reason and transfers control to App1 (`Cy_DFU_ExecuteApp(1u)`, *cy_dfu.c*). This function does an SRES and does not return.

Initializes host communication channel (`Cy_DFU_TransportStart()`, *dfu_user.c*), which in turn calls `CyBLE_CyBtldrCommStart()`, *transport_ble.c*.

Initializes BLE Immediate Alert Service (`IasInit()`, *ias.c*, which is part of this project). This function registers the `IasEventHandler()` function, also in *ias.c*, for BLE IAS events.

Sends a BLE stack version message to the debug UART.

Main loop:

Calls `Cy_BLE_ProcessEvents()`. This may result in a call to `AppCallBack()`.

Calls `Cy_DFU_Continue()` (*cy_dfu.c*), which, depending on the state, may read one command packet from the host, process the command, and write one response packet to the host. This function has calls to functions in *dfu_user.c. This function m*ay set the state to UPDATING or FINISHED.

If FINISHED, validates App1 and, if success, stops host communication (`Cy_DFU_TransportStop()`, *dfu_user.c*) and transfers control to App1 (SRES; no return). If validation fails, then resets host communication, turns off LEDs, and restarts DFU by calling `Cy_DFU_Init()`. User error handling can be placed here.

Else if FAILED, does the same as above as well as sending an error message to the debug UART.

Else if still UPDATING, checks for 5-second timeout. If so, resets host communication and restarts DFU.

If 300-second timeout and state is NONE, transfers control to App1 if it is valid, otherwise `Cy_SysLib_Halt()`, with the kit red LED ON.

Manages the LEDs per Table 3 on page 10.

Checks the button and the IAS alert level, and transfers control to App1 accordingly.

Function `AppCallBack()`:

Handles the following BLE events for BLE communication starts, device connected and disconnected, advertisement state change, and authorization and encryption.

If an advertising stopped event occurs, it is due to either download complete or advertising timeout. In this event host communication is stopped. Control is transferred to App1 if it is valid. Otherwise the device enters hibernate mode with the corresponding LED display.

File: *transport_ble.c*:

This file implements DFU host communication over BLE.

Function `CyBLE_CyBtldrCommStart()`:

Registers the `AppCallBack()` function in *main_cm4.c*, for general BLE GAP and GATT events, and the `DFUCallBack()` function in *transport_ble.c*, for BLE Bootloader Service (BTS) events.

Function `CyBLE_CyBtldrCommStop()`:

Initiates a GAP disconnect, and waits for the disconnect event. Then stops the BLE Component.

Function `CyBLE_CyBtldrCommWrite()`:

Sends a BTSS notification to send data to the DFU host.

Function `CyBLE_CyBtldrCommRead()`:

Processes BLE events while waiting for the required number of bytes to be received from the host. Processing BLE events results in a call to the callback function `DFUCallBack()`.

Function `DFUCallBack()`:

Handles BTSS write events by reading data from the DFU host.

File: *ias..h, c*:

This file implements the BLE Immediate Alert Service (IAS), to transfer control to the downloaded application.

Function `IasInit()`:

Registers the `IasEventHandler()` function for BLE IAS events.

Function `IasEventHandler()`:

Handles IAS write events by updating the global variable ‘alertLevel', which is read in main_cm4.c main loop.

File: *dfu_user.c*:

This file implements transport function redirects to *transport_ble.c*.

It also provides functions to override the DFU SDK weak functions `Cy_DFU_WriteData()` and `Cy_DFU_ReadData()`. These functions implement flash updates and flash reads respectively.

Document History

Document Title: CE216767 – PSoC 6 MCU Bluetooth Low Energy (BLE) Device Firmware Update (DFU)

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.
*C	6472121	MKEA	02/15/2019	Updated projects for PDL 3.1.0. Changed "bootloader" to "DFU".

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