PSoC 6 MCU Device Firmware Upgrade (DFU) by Copy

About this document

Scope and purpose

These examples demonstrate device firmware update (DFU) with PSoC® 6 MCU with preserving the primary application while downloading an update. The application is downloaded into the temporary storage region in internal flash, verified, and copied into the primary region for execution.

Requirements

Tool: **PSoC Creator™** 4.4; **Peripheral Driver Library** (PDL) 3.1.3

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

Associated Parts: All **PSoC 6 MCU** parts

Related Hardware: **CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit** with KitProg3 installed

Table of contents

1 Overview 2
2 Hardware and Software Setup 3
2.1 Hardware Setup 3
2.2 Software Setup 3
3 Operation 4
3.1 Build the Projects 4
3.2 Test the Projects 4
4 Debugging 6
5 Design and Implementation 7
5.1 Design Firmware 8
5.2 Memory Layout 9
5.3 Upgrade by Copy Operation 11
5.4 Design Considerations 12
5.4.1 Dual CPU 12
5.4.2 Software Reset 12
5.4.3 Memory Map Configurations 12
5.5 Components and Settings 13
5.6 Design-Wide Resources 13
6 Reusing This Example 14
7 References 15
Revision history 17

1 Overview

This example demonstrates DFU by copy operation:

- Communicating with a host, and downloading an application to temporary region on internal flash
- Validating the application after the download and transfer it to primary (bootable) region after successful validation
- Recover from interrupted DFU session and boot to last known good image.
- Validating an application, and then transferring control to that application.

There are two PSoC Creator project types, generally called "App0" and "App1". I^2C is the default communication channel used. The projects have the following features:

- App0 does the DFU operation; it downloads App1 to the temporary region on the internal flash and installs App1 into the primary region after successful validation.
- Each application configures a different color for the tricolor LED to make it easy to identify the running app.
- Pressing a kit button causes the project that is currently running to transfer control to the other project.

2 Hardware and Software Setup

2.1 Hardware Setup

This example uses the kit's default configuration. Refer to the kit guide to ensure that the kit is configured correctly. The KitProg3 system on the kit acts as a programmer for direct programming a USB-I^{2}C bridge. For more information, see the **KitProg3 User Guide**.

Note: *The PSoC 6 BLE Pioneer Kit (CY8CKIT-062-BLE) ships with KitProg2 installed. Before using this code example, make sure that the board is upgraded to KitProg3. The tool and instruction are available in this **GitHub** repository.*

2.2 Software Setup

User-defined configurations are available in *dfu_user.h*. Use the setup as-is to realize the use case.

3 Operation

3.1 Build the Projects

Note: *These instructions implement the I²C-based DFU system. For UART- and SPI-based DFU, see* **CE213903**.

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

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

Note: *If you are using MDK compiler, update the linker command line for both CM0 and CM4 in both App0 and App1 with "--diag_suppress 6314" command. There are few sections in the linker script that are reserved for future usage. This command helps in warning-free build while retaining those unused sections.*

3.2 Test the Projects

1. Connect the kit board to your PC using the provided USB cable through the KitProg3 USB connector.
2. Program the App0 project into the kit. For more information on device programming, see PSoC Creator Help.

   Confirm that the red LED on the kit blinks once every two seconds. This indicates that App0 is running. Additionally, App0 logs certain information on the console via USB-UART bridge at 115200 baud, 8N1.

3. Run PSoC Creator Bootloader Host Program (BHP). In PSoC Creator, select **Tools** > **Bootloader Host...**.

   Configure the BHP for the KitProg3 USB-I²C bridge connection. See **Figure 1** for I2C address and bit rate settings. For more information on using BHP, see BHP Help.

4. Using BHP, download App1 (*<project>/ PSoC6DfuBlinkyApp1.cydsn/CortexM4/<Compiler>/<Build Config> /PSoC6DfuBlinkyApp1.cyacd2*). Wait for the download to complete. During the download, observe the RGB LED blinks red at a rapid rate.
5. After download is complete, confirm that the RGB LED starts blinking blue, indicating that App1 is running.

Operation

6. Press the SW2 kit button for at least a second, and release it. Observe that the kit LED blinks red, indicating that control has been transferred to the bootloader.
7. Repeat the two previous steps to test the process of reinstalling App1.

Note: *Make sure that App0 is running before attempting to use BHP. App1 is not designed to do a DFU operation.*

Note: *In addition to a 'Program' option, BHP has a 'Verify' option and an 'Erase All' option. Be cautious while issuing 'Erase All' command. It will erase entire flash excluding App0 region.The 'Verify' command verifies the content of temporary region only. Design doesn't support verifying the primary region contents.*

4 Debugging

You can debug the example to step through the code. PSoC 6 MCU has two CPUs: an Arm Cortex®-M4 (CM4) and a Cortex-M0+ (CM0+). PSoC Creator supports debugging a single CPU (either CM4 or CM0+) at a time; in this example, most of the tasks are done by CM4, as **Table 2** shows. To debug App1, you may need to use the **Debug** > **Attach to Running Target...** option. For more information on debugging using PSoC Creator, see PSoC Creator Help.

5 Design and Implementation

This example has two applications, called "App0" and "App1". Each application is a separate PSoC Creator project; both projects are in the same PSoC Creator workspace. Each application has the following features:

- App0 does the DFU operation; it downloads and installs App1.
- App0 blinks controls RGB LED on **PSoC 6 BLE Pioneer Kit** and blinks red once every two seconds. App1 blinks it blue twice every second, both using firmware delays. The unique color of LED makes it easy to see which project is running.
- Both projects monitor the kit button SW2. If the button is pressed for more than a second and then released, the currently running application transfers control to the other application. The RGB LED color changes as described previously.
- **Figure 1** and **Figure 2** show the PSoC Creator project schematic for both App0 and App1. App0 has the host communication Component; App1 does not.

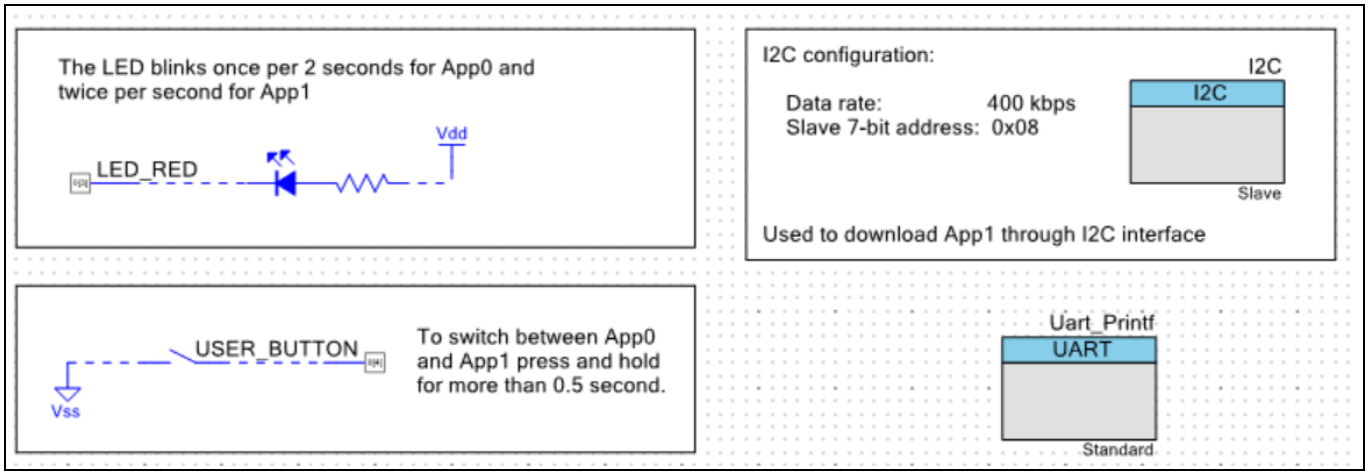

Figure 1 PSoC Creator Schematic for App0

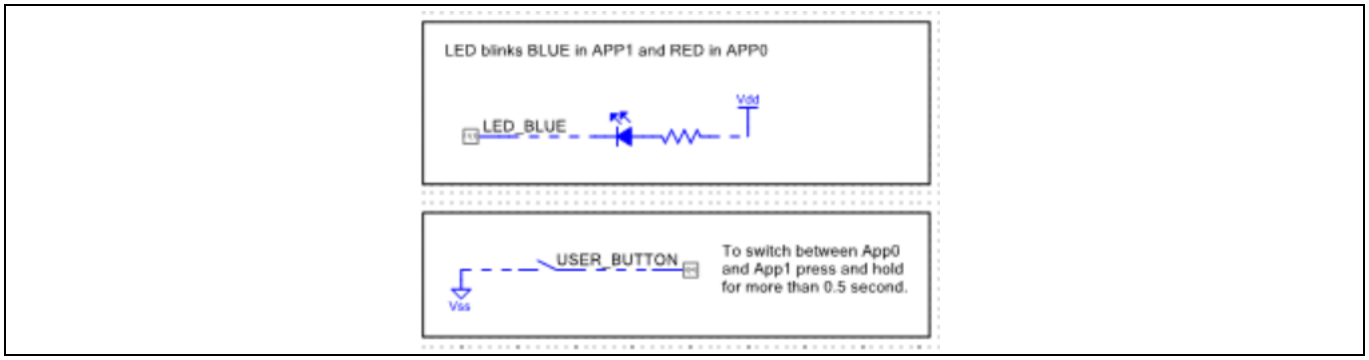

Figure 2 PSoC Creator Schematic for App1

5.1 Design Firmware

The firmware portion of the design is implemented in the files listed in **Table 1**. 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 CPU core. PSoC 6 MCU has two CPUs: CM4 and CM0+. See **Table 2** for specific tasks for each core.
user_interface.c, h	Contains the user button status check routine and LED blink control.
console_print.c	This function provides the implementation of helper functions required for the console print. This implementation is available for CM4 only in this code example.
cy_dfu.h, .c	The DFU software development kit (SDK) files.
cy_dfu_bwc_macro.h	Contains macros for backward compatibility to facilitate porting of legacy bootloader projects.
dfu_user.h	Contains user-editable configuration macros that control the operation and enabled 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 memory.
transport_xxx.h, .c	Contains transport functions for the host communications Component being used. These functions are typically called by the transport functions in dfu_user.c.
dfu_cm4.ld, dfu_cm0p.ld	Custom GCC linker scripts. 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. In each project, these files are customized to realize the use case.
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 .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. In each project, these files are customized to realize the use case.
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. In each project, these files are customized to realize the use case.
post_build_core1.bat	Batch file to create the downloadable application image (.cyacd2 file) for App1.

5.2 Memory Layout

Figure 3 shows the typical memory usage for each CPU core in each project. This layout is for PSoC 6 MCU devices with 1 MB flash and 288 KB SRAM.

App0 always starts at the beginning of device user 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 followed by the temporary region. Each app has defined flash areas for each CPU core: core0 (CM0+) and core1 (CM4). The temporary storage region starts at 0x10060000, immediately after the primary region (app1). This region is used by the bootloader to write the new application image during the DFU process.

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.

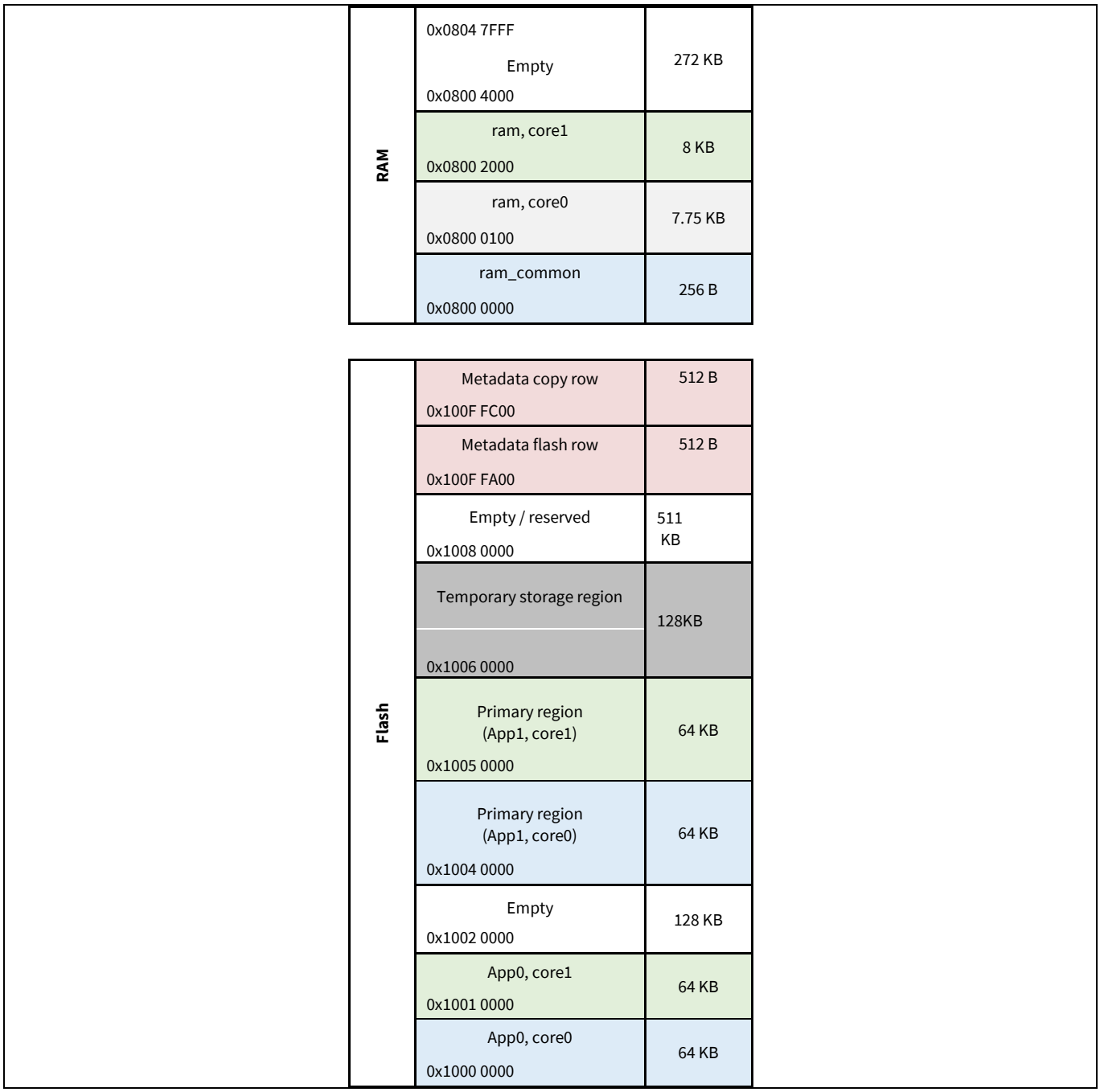

Figure 3 Memory Layout of Applications

5.3 Upgrade by Copy Operation

Figure 4 shows host-to-device communication over the dedicated communication channel (I^2C) and how the upgrade is performed. The Bootloader Host program running on the host (typically a PC) sends the application (app1.cyacd2) to the device. Bootloader (App0) receives the application in chunks and writes it to secondary region (temporary storage region). After the application has been downloaded completely, App0 verifies the downloaded image and installs it to primary region and erases the secondary region. If the downloaded image is corrupted or invalid, the bootloader will restart the DFU process.

App0 provides options to enter the DFU mode in following ways:

- On power on, starts the DFU mode, if both Primary and secondary slots are invalid
- Pressing the user button for at least a second during power on
- Pressing the user button for at least a second while running App1

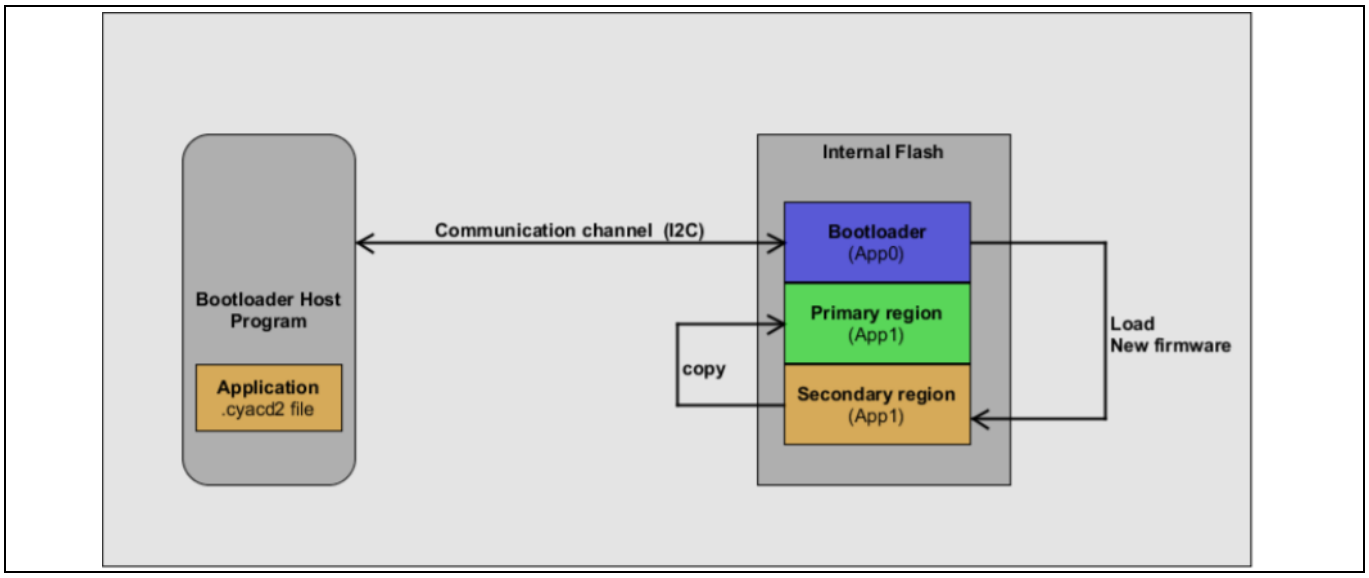

Figure 4 Upgrade by Copy Operation

The bootloader is designed to handle power failure or any possible interruption to DFU session. On every power on, it validates the secondary slot, and performs the necessary updates if a valid image exists in the secondary slot, and boots to the newly installed image. If no valid update is found in the secondary slot, it will boot to app1 directly if it is valid.

Note: *You can always skip installing the newly downloaded image by power cycling the device and holding the user button during power on. The device will directly start the DFU process and allow downloading a newer update.*

5.4 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.*

5.4.1 Dual CPU

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. This can easily be changed so that either core can run any of the tasks, including DFU.

Table 2 CPU Tasks in Each Application

Application	Cortex-M0+	Cortex-M4
App0	Executes first at device reset. The reset handler controls application transfer. Note that if application transfer does occur, it occurs before the CM4 CPU is turned on. Turns ON CM4. Does nothing else.	Blinks an LED once per two seconds. Downloads, verifies and installs App1. Monitors the button. After the DFU operation, or when the button pressed, initiates transfer of control to App1, with software reset.
App1	Executes first, then turns ON CM4. Does nothing else.	Blinks an LED twice per second. Monitors the button. When button is pressed, initiates transfer of control to App0, with software reset.

5.4.2 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.

It is possible to 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**.

5.4.3 Memory Map Configurations

This code example requires to define the size of the temporary region equal to that of the primary region (app1). The temporary region is used for downloading the application during the device-to-host communication using the DFU process in order to ensure that the device can safely boot to the primary application in case of interruptions, power failures, or image corruption in the DFU process.

Note: *This code example assumes that the temporary storage region and primary region (App1) are contiguous.*

5.5 Components and Settings

Table 3 lists the PSoC Creator Components used in this example, the hardware resources used by each, and parameter settings that are changed from the default values.

Table 3 PSoC Creator Components

Component	Instance Name	Purpose	Non-default Settings
UART	UART	Host communication	115200 baud, 8N1, TX Only.
I2C	I2C	Host communication	Data rate 400 kbps; Use TX FIFO; Use RX FIFO
Pin	LED	Drive an LED	No HW connection; External terminal; Initial drive state High (1); Max frequency 1 MHz
Pin	SW2	Read button state	No HW connection; External terminal; Drive mode Resistive Pull Up; Max frequency 1 MHz

5.6 Design-Wide Resources

Figure 5 shows the pin assignments for the PSoC 6 BLE Pioneer Kit, for the I2C, UART, LED, and user button.

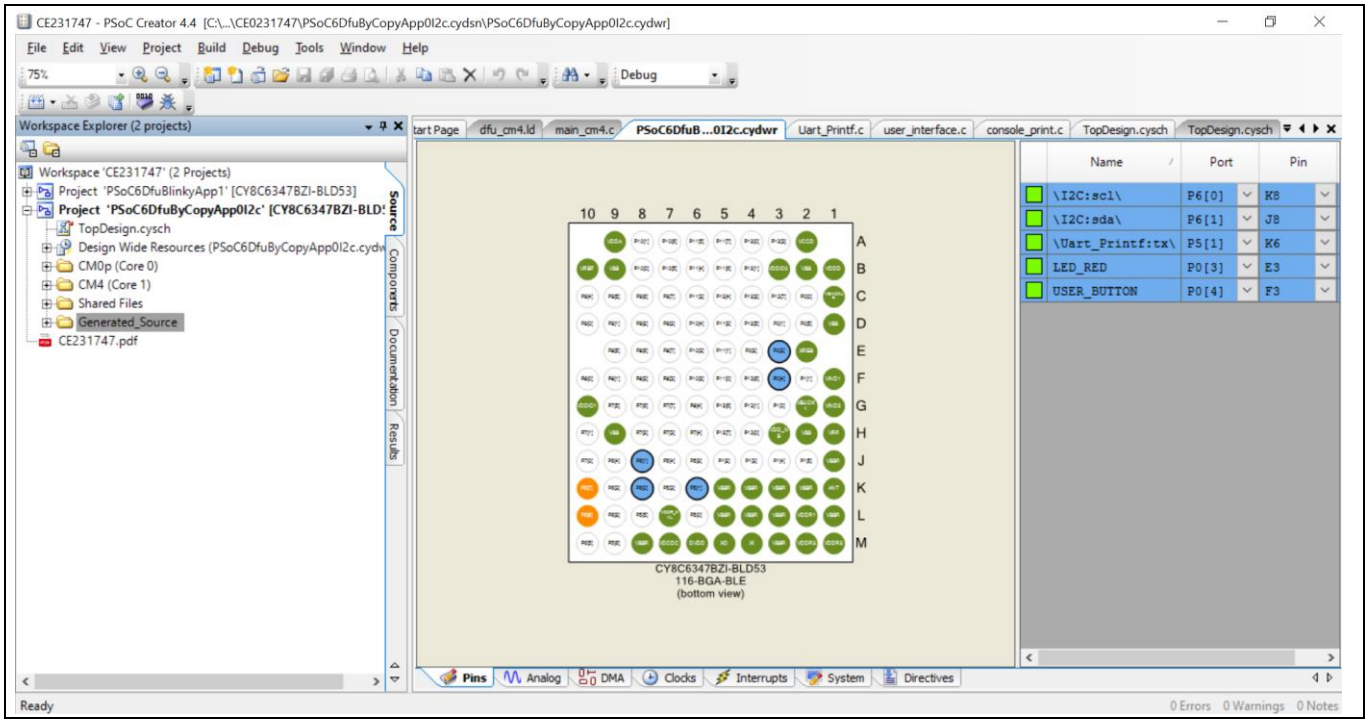

Figure 5 Pin Assignments for the PSoC 6 BLE Pioneer Kit for I²C DFU

6 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.

The **Operation** instructions implement I²C-based DFU. To implement UART or SPI-based DFU, first select **Build** > **Generate Application**, then edit *dfu_user.c* as follows:

- Replace the `#include`: "`transport_i2c.h`" with either "`transport_uart.h`" or "`transport_spi.h`"
- Replace five instances of "`I2C-I2c`" with either "`UART_uart`" or "`SPI_Spi`".

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

Note: *This code example enables UART Tx for logging. Either disable this component or select alternate pins for console UART in design wide resources.*

Filed applications might require custom memory map. Update the linker files as necessary for the application.

For single-CPU 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*. CM0+ performs will perform all the tasks; port all the code from *main_cm0p.c* to main.c.

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.

7 References

For the PSoC 6 MCU devices, see **KBA223067** in the psoc6 community for a comprehensive list of PSoC 6 MCU resources.

PSoC 6 DFU-Related Application Notes

[1] **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

[2] **AN221774** – Getting Started with PSoC 6 MCU: Describes PSoC 6 MCU devices and how to build your first ModusToolbox or PSoC Creator project

[3] **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 DFU-Related Code Examples

[4] **CE216767** – PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity Bootloader: Describes a BLE DFU system for PSoC 6 MCU

[5] **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

[6] **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

[7] **CE221984** – PSoC 6 MCU Dual-Application I2C DFUL: Similar to the basic I^2C DFU; manages two downloaded applications instead of one

[8] **CE222802** – PSoC 6 MCU Encrypted DFU: Similar to the basic UART DFU; the application is digitally signed and encrypted

PSoC Creator Component Datasheets

[9] **UART** – Supports the serial communication block in UART mode

[10] **I2C** – Supports the serial communication block in I^2C mode

[11] **SPI** – Supports the serial communication block in SPI mode

[12] **Pins** – Supports connection of hardware resources to physical pins

Device Documentation

[13] **PSoC 6 MCU: PSoC 63 with BLE Datasheet**

[14] **PSoC 6 MCU: PSoC 63 with BLE Architecture Technical Reference Manual**

Development Kits

[15] **CY8CKIT-062-BLE** – PSoC 6 BLE Pioneer Kit

[16] **CY8CKIT-062-WiFi-BT** – PSoC 6 WiFi-BT Pioneer Kit

[17] **CY8CPROTO-063-BLE** – PSoC 6 BLE Prototyping Kit

[18] **CY8CPROTO-062-4343W** – PSoC 6 Wi-Fi Prototyping Kit

References

Tool Documentation

[19] **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.

[20] **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.

[21] **Peripheral Driver Library** (PDL) – Installed by PSoC Creator 4.4. Look in the *<PDL install folder>/doc* directory for the User Guide and the API Reference

Revision history

Document version	Date of release	Description of changes
**	2020-12-09	Initial release

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2020-12-09
Published by
Infineon Technologies AG
81726 Munich, Germany

© 2020 Infineon Technologies AG. All Rights Reserved.

Do you have a question about this document?
Go to www.cypress.com/support

Document reference
002-31747 Rev. **

IMPORTANT NOTICE
The information contained in this application note is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this application note must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this application note.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office (**www.infineon.com**).

WARNINGS
Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.