USBFS Bootloader and Bootloadable Code Example 3.0

Features

- Bootloading via USB communication interface
- Switching between Bootloader and Bootloadable applications

General Description

This example project demonstrates the basic operation of the Bootloader and Bootloadable components when the communication interface is a USB.

Development Kit Configuration

This example project is designed to run on the CY8CKIT-046 kit from Cypress Semiconductor. A description of the kit, along with more code examples and ordering information, can be found at http://www.cypress.com/go/cy8ckit-046.

The project requires configuration settings changes to run on other kits from Cypress Semiconductor. Table 1 is the list of the supported kits. To switch from CY8CKIT-046 to any other kit, change the project's device with the help of Device Selector called from the project's context menu.

Table 1. Development Kits vs Parts

Development Kit	Device
CY8CKIT-046	CY8C4248BZI-L489
CY8CKIT-030	CY8C3866AXI-040
CY8CKIT-050	CY8C5868AXI_LP035
CY8CKIT-001	CY8C3866AXI-040/ CY8C5868AXI_LP035

The pins assignment for the supported kits is in Table 2 and Table 3.

Table 2. Pins Assignment of USBFS_Bootloader Project

Pin Name	Development Kit			
	CY8CKIT-046	CY8CKIT-030	CY8CKIT-050	CY8CKIT-001
\USBFS:Dm\	P13[1]	P15[7]	P15[7]	P15[7]
\USBFS:Dp\	P13[0]	P15[6]	P15[6]	P15[6]
LED_REB	P5[2]	–	–	–
LED_GREEN	P5[3]	–	–	–
LED_BLUE	P5[4]	–	–	–
LED4	–	P6[3]	P6[3]	P6[3]

Table 3. Pins Assignment of USBFS_Bootloadable Project

Pin Name	Development Kit			
	CY8CKIT-046	CY8CKIT-030	CY8CKIT-050	CY8CKIT-050
LED_REB	P5[2]	–	–	–
LED_GREEN	P5[3]	–	–	–
LED_BLUE	P5[4]	–	–	–
LED3	–	P6[2]	P6[2]	P6[2]

To handle hardware differences between the supported kits, separate TopDesigns and control files are added to the project. The control files are responsible for the pins assignment depending on the selected device. Manual placement of the pins overrides the control file directives, therefore the pins in the Design Wide Resource (DWR) file should be unlocked. All these files can be found in the **Components** tab of the workspace explorer.

Bootloader Project Description

The example project consists of the following components: USBFS, Bootloader, and pins.

The USBFS component is used to establish communication with the PC via a USB executing the role of the communication component for the bootloader. It means that the bootloader component is provided with an interface that allows managing command transfers from the PC to the device and status transfers from the device to the PC.

The bootloader component is used to update the device flash memory with a new application. The component is responsible for the update process and uses the communication component to get a new application with the help of a command and status protocol. The component is also responsible for writing the new application image in the flash memory.

The pin components are used to control the LED which indicates that the bootloader is running. For PSoC4, the RGB LED is red and for PSoC 3/PSoC 5LP, LED4 is turned on.

Bootloader Project Flash Protection

The bootloadable project does not allow re-writing a bootloader project during the application image update. But, the flash rows consumed by the bootloader should be protected to avoid being corrupted by the application. The rows consumed by the bootloader example project are not protected because the bootloader size changes depend on the compiler and optimization options. Incorrect flash protection of bootloader can cause failures during the application image update because the rows consumed by the application can become protected due to bootloader project size change. Therefore, it is important to check the number of protected rows consumed by the bootloader after rebuild.

The number of rows consumed by the bootloader is calculated as follows: the bootloader project flash used in bytes divided by the flash row size in bytes, then the calculated value is rounded up to the nearest whole number. The bootloader project flash used is equal to the flash used in the Output window after the bootloader project is built. The flash row size differs depending on the

selected device (hint in DWR file in the **Flash Security** tab telling the number of bytes in the row).

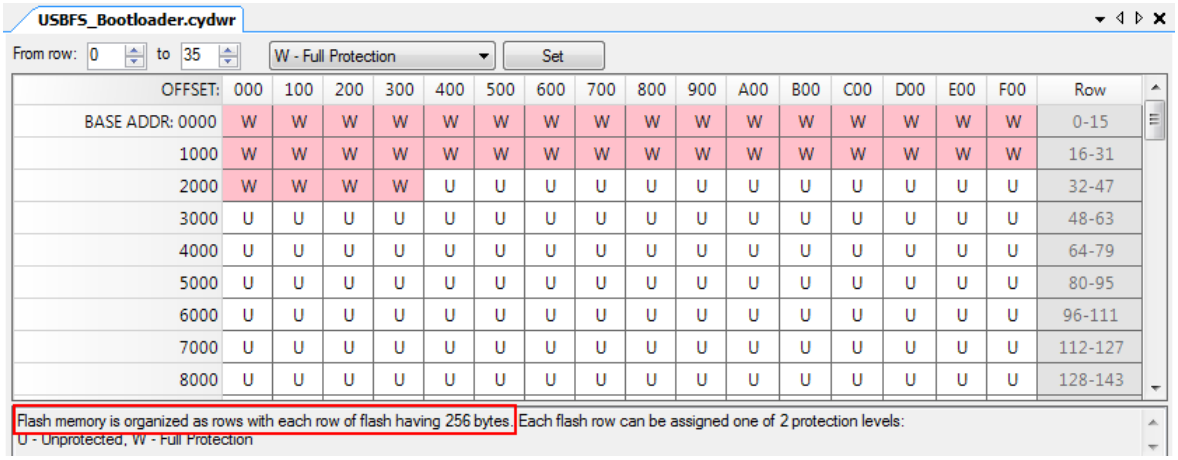
The example below for PSoC 4200L shows that the number of rows consumed by the bootloader is equal to 9216 bytes / 256 bytes = 36 rows.

Figure 1. Bootloader Project Output Window

```
arm-none-eabi-gcc.exe -mcpu=cortex-m0 -mthumb -Wno-main -I. -IGenerated_Source\PSoC4 -Wa,-alh=.\CortexM0\ARM_GCC_493\R
CyAppFeatureCfg.Creator__EnableAllDevices is set.
arm-none-eabi-ar.exe -rs .\CortexM0\ARM_GCC_493\Release\USBFS_Bootloader.a .\CortexM0\ARM_GCC_493\Release\cyfitter_cfg
arm-none-eabi-ar.exe: creating .\CortexM0\ARM_GCC_493\Release\USBFS_Bootloader.a
arm-none-eabi-gcc.exe -Wl,--start-group -o .\CortexM0\ARM_GCC_493\Release\USBFS_Bootloader.elf .\CortexM0\ARM_GCC_493\
CyAppFeatureCfg.Creator__EnableAllDevices is set.
cyelftool.exe -P C:\examples\USBFS_Bootloader\USBFS_Bootloader.cydsn\CortexM0\ARM_GCC_493\Release\USBFS_Bootloader.elf
cyelftool.exe -S C:\examples\USBFS_Bootloader\USBFS_Bootloader.cydsn\CortexM0\ARM_GCC_493\Release\USBFS_Bootloader.elf
Flash used: 9216 of 262144 bytes (3,5%)
SRAM used: 2760 of 32768 bytes (8,4%). Stack: 2048 bytes. Heap: 128 bytes.
--------------- Build Succeeded: 11/17/2015 14:25:47 ---------------
```

When the number of rows consumed by the bootloader project is known, open the DWR file on the **Flash Security** tab and set protection to the appropriate rows. Note that the bootloader project starts from row 0 and consumes rows successively.

Figure 2. Bootloader Project DWR Flash Security Tab



For more information about the bootloader and bootloadable flash layout refer to the components datasheets.

Bootloadable Project Description

The example project consists of the following components: the Bootloadable and pins.

The Bootloadable component allows linking the USBFS_Bootloader project to the USBFS_Bootloadable project and creating a bootloadable application image. Also, it specifies additional parameters for the bootloadable project.

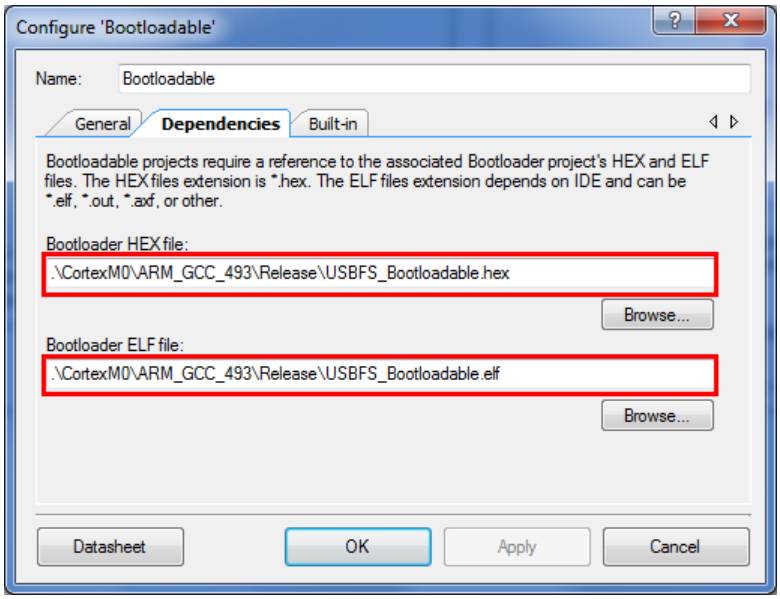
The pins component is used to control the LED which indicates that the application is running. For PSoC4, the RGB LED is green and for PSoC 3/PSoC 5LP, LED3 is turned on.

Example Project Execution Flow

To execute this code example, follow the procedure:

1. Use two projects to execute this code example: the USBFS_Bootloader and USBFS_Bootloadable. Note that both projects should be added into single workspace.
2. First, build the USBFS_Bootloader project because it has to be linked with the bootloadable project.
3. Open the USBFS_Bootloadable project top design schematic. Then, open the bootloader component **Dependencies** tab and specify the path to the bootloader project HEX and ELF files as shown in Figure 3.

Figure 3. Bootloadable Component **Dependencies** Tab

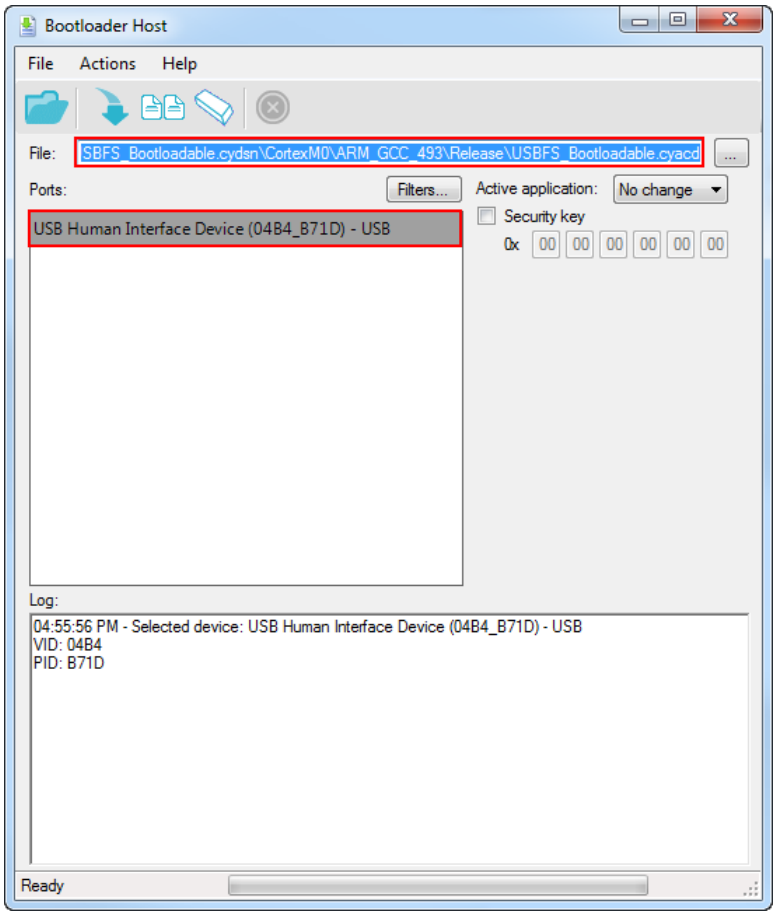


4. Build the USBFS_Bootloader project to get the bootloadable application image and hex files.
5. Connect the PSoC kit to the PC through a USB connector for programming.
6. Program the USBFS_Bootloader into the device. The USB cable can be disconnected from the programming connector at this point.
7. Connect the kit to the PC through a USB connector for communication. After the USB HID device is enumerated, it appears in the Bootloader Host tool. The LED indicates that the bootloader is running: for PSoC4 RGB, the LED is red and for PSoC 3/PSoC 5LP, LED4 is turned on.

   Note that the bootloader project does not contain an application and waits forever (instead of 10ms configured in GUI) for downloading.

8. Open the Bootloader Host tool by navigating to **Tools** > **Bootloader Host** in PSoC Creator. Observe that the HID USB device with VID 0x4B4 and PID 0xB71D appears in the Ports list.

Figure 4. Bootloader Host Tool HID Device Enumerated



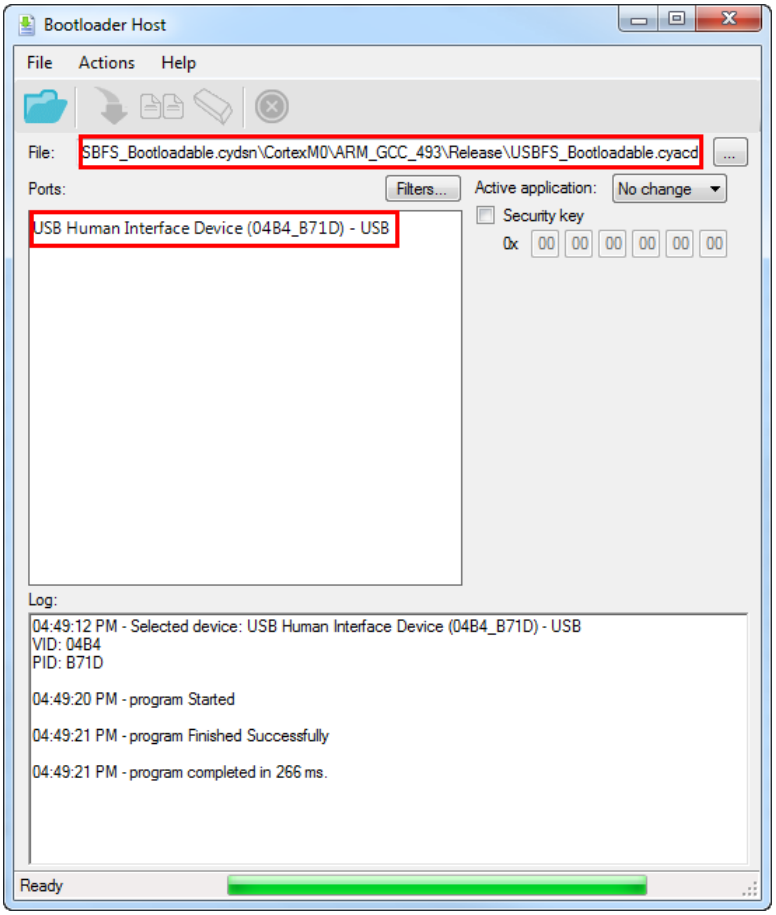
9. Press the **File** button and choose the bootloadable application image USBFS_Bootloadable.cyacd. It is available in the project folder USBFS_Bootloadable.cydsn:
   - For PSoC 3: \DP8051_Keil_951\(Debug or Release)
   - For PSoC 5LP: \CortexM3\ARM_GCC_493\(Debug or Release)
   - For PSoC 4200L: \CortexM0\ARM_GCC_493\(Debug or Release)
10. To start the application update, click the **Program** button.
11. After the bootloadable application image is downloaded successfully, a software reset occurs, and the device starts executing a new application. The LED indicates that the application is running: for PSoC4 RGB, the LED is green and for PSoC 3/PSoC 5LP, LED3 is turned on.

    Note that the bootloadable application does not contain a USBFS device and after downloading it disappears from the Bootloader Host Ports list. The device has to be reset to start the bootloader project. The device waits for the application update for 10 seconds, then jumps to the bootloadable application.

Expected Results

Following the instructions above, you can download a new application into the device and get the Bootloader Host tool output after successful downloading.

Figure 5. Bootloader Host Tool Successful Application Image Upload



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

Phone : 408-943-2600
Fax : 408-943-4730
Website : www.cypress.com