USBFS Bulk Wraparound Example Project

3.0

Features

- Vendor-Specific USB Full-Speed device
- Communication with USB device via BULK IN and OUT endpoints
- 16-bit APIs (only for PSoC 4200L)

General Description

This example project demonstrates how to establish communication between the PC and Vendor-Specific USB device. The device has two endpoints: BULK IN and BULK OUT. The OUT endpoint allows the host to write data into the device and the IN endpoint allows the host to read data from the device. The data received in the OUT endpoint is looped back to the IN endpoint.

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.

Table 2. Pins Assignment

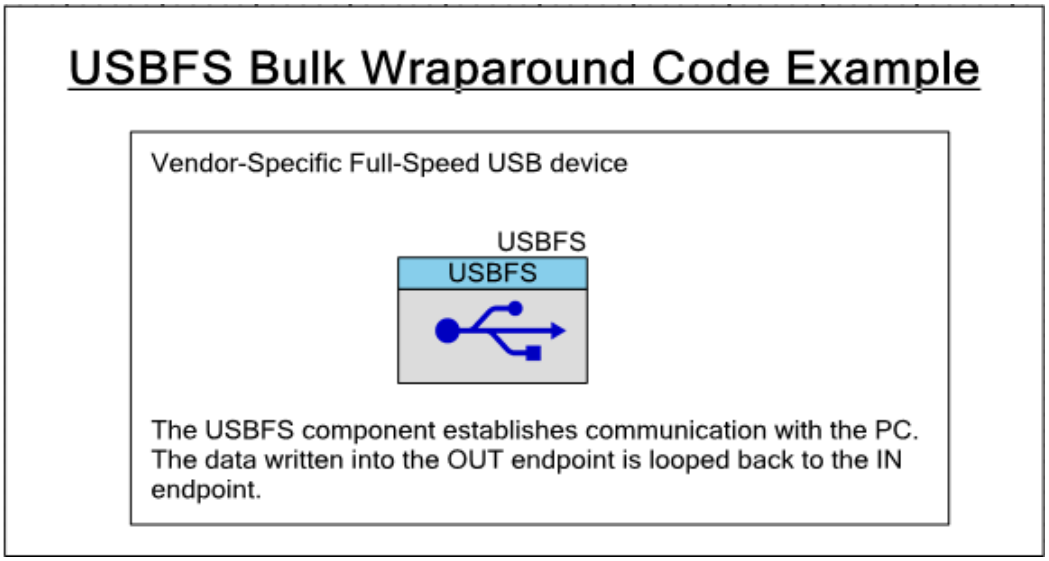
Pin Name	Development Kit			
	CY8CKIT-046	CY8CKIT-030	CY8CKIT-050	CY8CKIT-001
\USBFS:Dm\	P13[1]	P15[7]	P15[7]	P15[7]
\USBFS:Dp\	P13[0]	P15[6]	P15[6]	P15[6]

To handle hardware differences between the supported kits, separate TopDesigns and control files are added to the project. The control files monitor/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.

Project Configuration

The example project consists of the single component USBFS used to establish communication with a PC. The project schematic is shown in Figure 1.

Figure 1. Example Project Design Schematic



The USB device's VID is 0x4B4 (this is Cypress Semiconductor vendor ID) and PID is 0x8051 (arbitrarily chosen). The appropriate string descriptors contain the company's name ("Cypress Semiconductor") and the example project name ("Generic USB Bulk Wraparound Transfer"). The device is powered via a USB, so the device configuration is bus-powered. The device has two BULK endpoints: EP1 IN and EP2 OUT. The OUT endpoint allows the host to write data into the device and the IN endpoint allows the host to read data from the device. Each endpoint maximum packet is 64 bytes, therefore the up to 64 bytes can be transferred.

The important option of the USBFS component is the endpoint buffer management, which defines the way how the endpoint buffers are serviced. This option is set to DMA with Manual Buffer Management (Static Allocation). It means that data transfer between the endpoint memory to the SRAM buffer is executed via DMA and a request is needed to start transferring.

PSoC 4200L provides additional 16-bit endpoint memory access whereas PSoC 3 and PSoC 5LP allow only 8-bit access. The component GUI provides option **Generate 16-bit endpoint access APIs** to enable this capability. These APIs allow decreasing the execution time of loading and reading by factor 2. But it is required that the endpoint buffer and SRAM buffer be aliened for 16-bit access. The component monitors the endpoint memory buffer alignment and

the example project monitors the SRAM buffer alignment and the APIs replacement when the option is enabled (Default).

The USBFS component main configuration Tabs are shown below

Figure 2. Descriptor Root

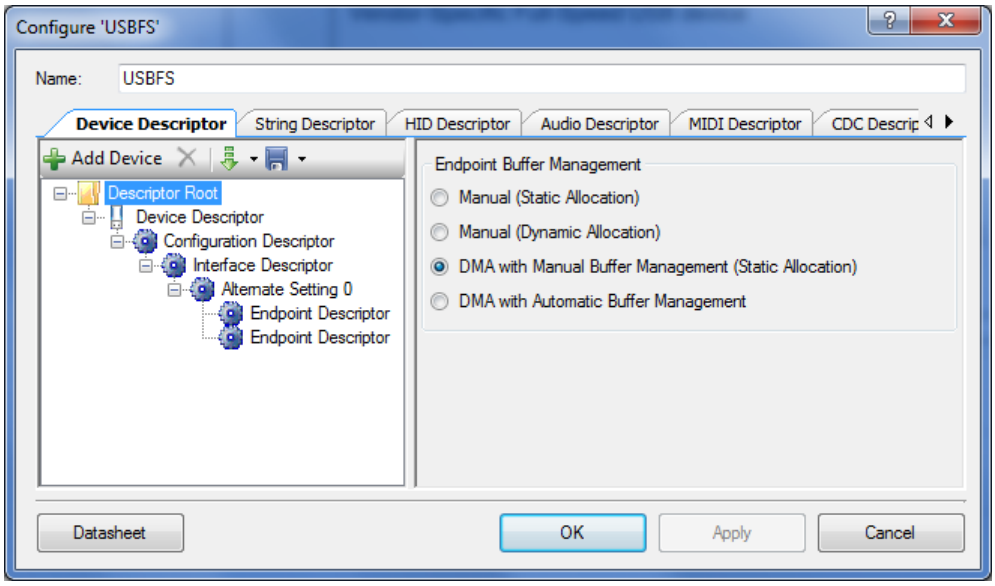


Figure 3. Device Descriptor

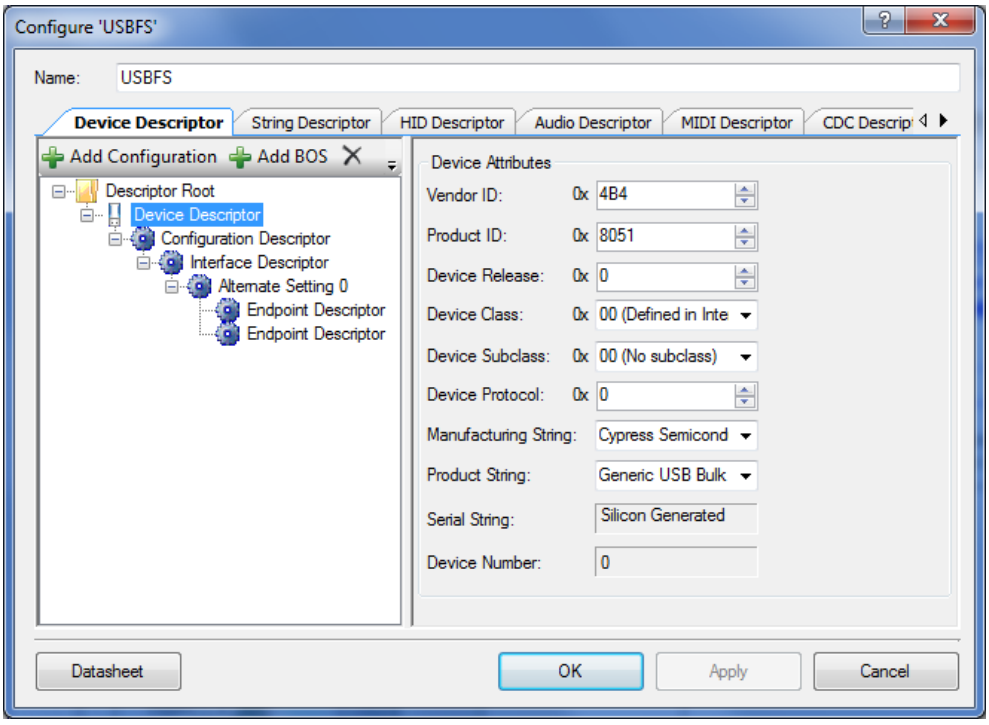


Figure 4. USBFS Configuration Descriptor

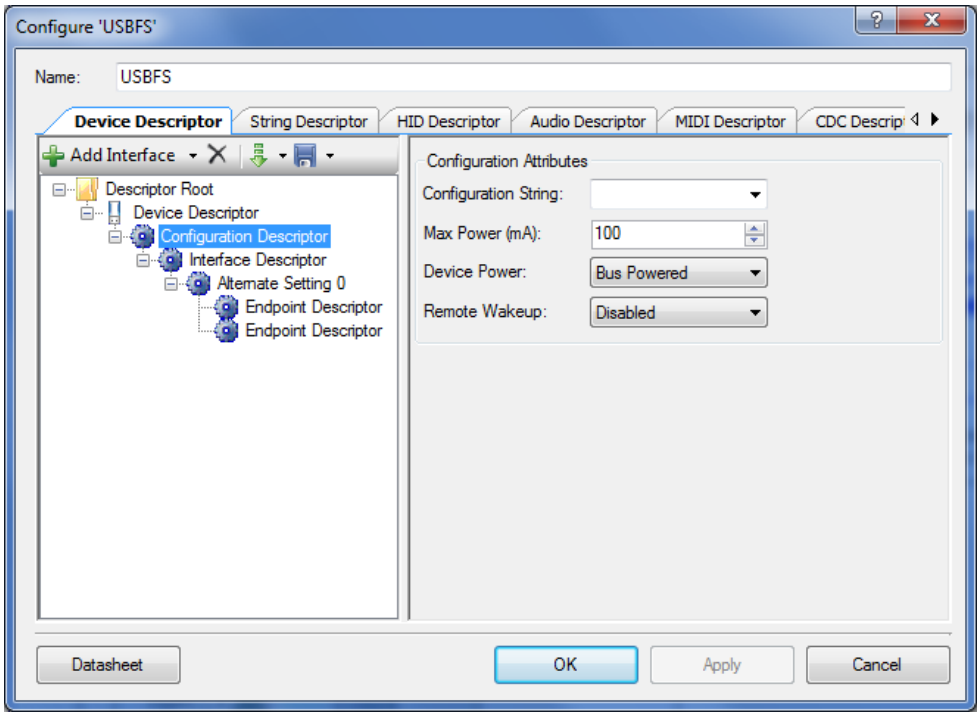


Figure 5. USBFS Endpoint 1 Descriptor

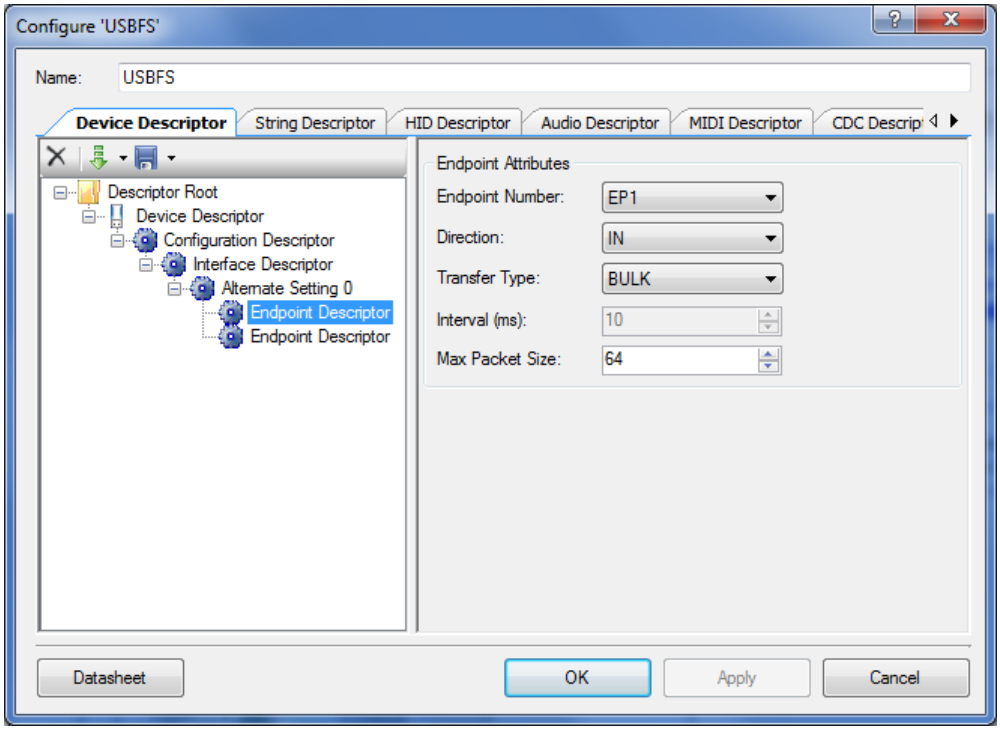
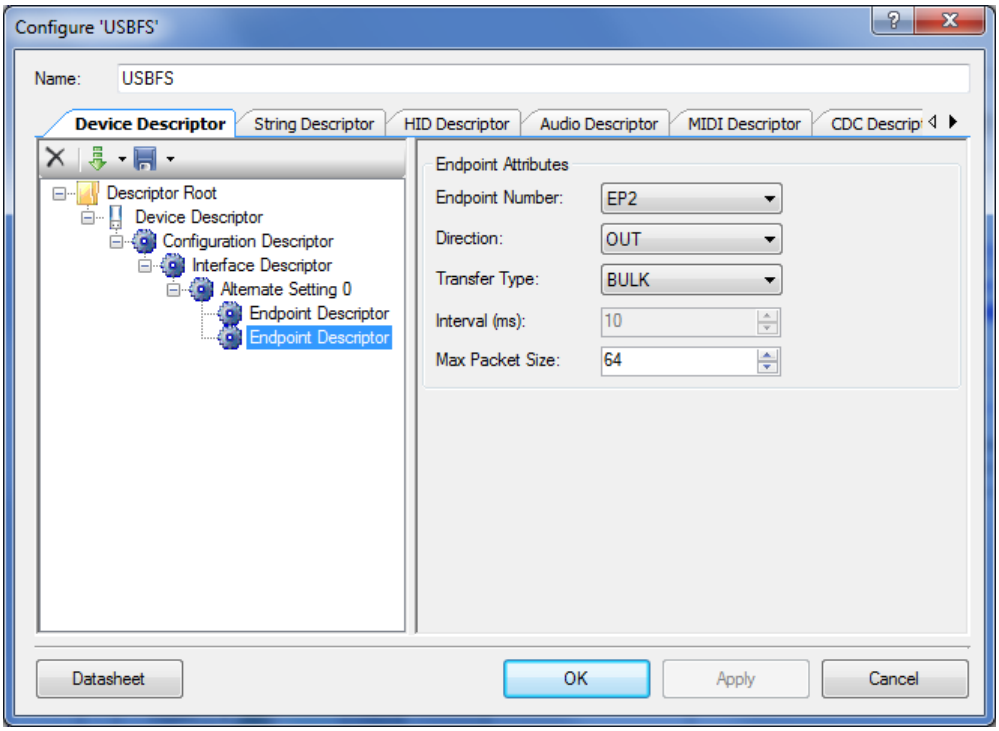


Figure 6. USBFS Endpoint 2 Descriptor



Project Description

In the main firmware routine, the USBFS component is configured for operation and started. The global interrupts are enabled because they are mandatory for the component operation. The component waits for completion of the enumeration process. This process starts by the host when the USB cable is connected to the PC.

After the Vendor-Specific USB Full-Speed device is enumerated, the device waits for data transfer. The OUT endpoint state is polling to identify whether data has been written. When data is available, it is read from the endpoint memory buffer into the SRAM buffer. The IN endpoint state is polling for a place to put the data, then data is copied from the SRAM buffer into the endpoint buffer. The host now is able to read the written data back.

Note that the load and read endpoint APIs only start the DMA transfers, therefore the code waits until data transfer is completed.

Example Project Execution Flow

To execute the USBFS component code example follow the procedure:

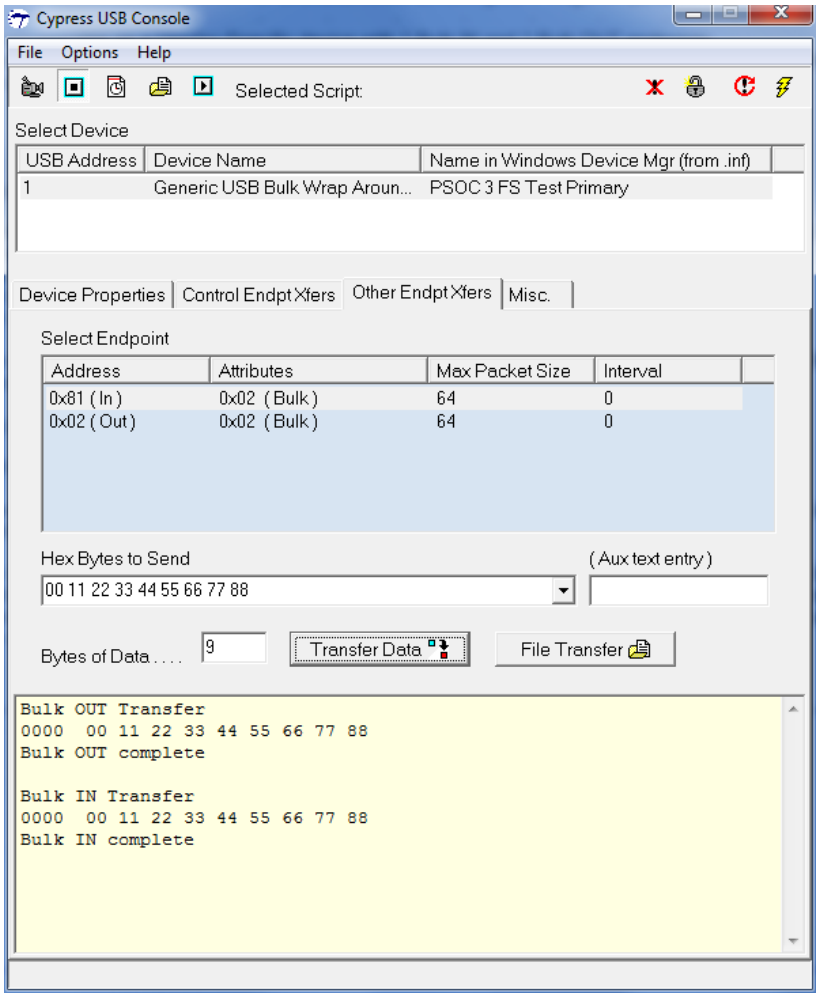
1. Connect the PSoC kit to the PC through a USB connector for programming.
2. Build the USBFS_Bulk_Wraparound example project and program it into the device. The USB cable can be disconnected from the programming connector at this point.

3. Install SuiteUSB 3.4 (follow link for download and installation instructions) to get the USB driver and Cypress USB Console (CyConsole) application required in the following steps.
4. Add the device's **VendorID** (0x04B4) and **ProductID** (0x8051) to the **CYUSB.INF** file. After installation of the Cypress Suite USB installer, the driver file is located in the Driver subdirectory of the install directory (Default is C:\Program Files\Cypress\Cypress Suite USB 3.4.7\Driver\bin). The detailed instructions are in the **CyUSB.pdf** file shipped with the suite.
5. Connect the kit to the PC through a USB connector for communication.

   Note The USBFS device does not have signed driver. To make the device work for Windows 7, reboot PC as normally pressing F8 repeatedly while the boot process is running. When the boot menu appears, select **Disable Driver Signature Enforcement.** This option is temporary – on the next reboot, driver signing will be activated again.
6. To communicate with the USB device, run the Cypress USB Console application. Write up to 64 bytes into the OUT endpoint, then read back the same data from the IN endpoint. The Figure 7 shows the output of CyConsole after data loops back.

   Note The host fails to read the IN endpoint if data has not been loaded into its buffer. This can happen if the OUT endpoint has not been written before reading data from the IN endpoint.

Figure 7. Cypress USB Console Output



Expected Results

If you follow the instructions above, the Vendor-Specific USB Full-Speed device is enumerated by the PC with BULK IN and OUT endpoints and then you can communicate with the USB device: receive data from the PC via the OUT endpoint and to loop that data back using the IN endpoint using Cypress USB Console application.