

# USBFS AUDIO PSoC3/5LP Code Example

## 3.0

## Features

- Simple USB audio streaming
- Transferring USB audio data to VDAC by DMA

## General Description

This USB example project implements a USB Audio Device that connects to the PC via the USB interface. The USB Audio Device does not require a special USB driver, because the USB Audio support is already built into Windows. The device appears in the system as a mono speaker with 8-bit resolution and a sample rate of 32 kHz.

## Development Kit Configuration

This example project is designed to run on the [CY8CKIT-030](#) or [CY8CKIT-050](#) or [CY8CKIT-001](#) kit from Cypress Semiconductor.

The project requires only the device change to run on the selected kit. Table 1 provides the relationship between the development kit and device. To change the device use Device Selector called from the project's context menu.

Table 1. Development Kits vs Parts

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

## Project Configuration

The example project consists of the component USBFS, VDAC8, DMA, interrupt, and LCD. The project schematic is in [Figure 1](#).

The USBFS component implements a limited USB Audio Device which appears in the system as a speaker. It means that audio can be streamed only from the PC to the device and not in the opposite direction. The USB device has VID = 0x4B4 (this is Cypress Semiconductor vendor ID) and PID = 0x2001 (arbitrarily chosen). The appropriate string descriptors are also provided with the company's name (Cypress Semiconductor) and appropriate audio device information. The device is powered through the USB connection, so the device configuration is bus-powered. The

device has a single OUT ISOC endpoint with the maximum packet of 192 bytes to receive an audio stream for the host.

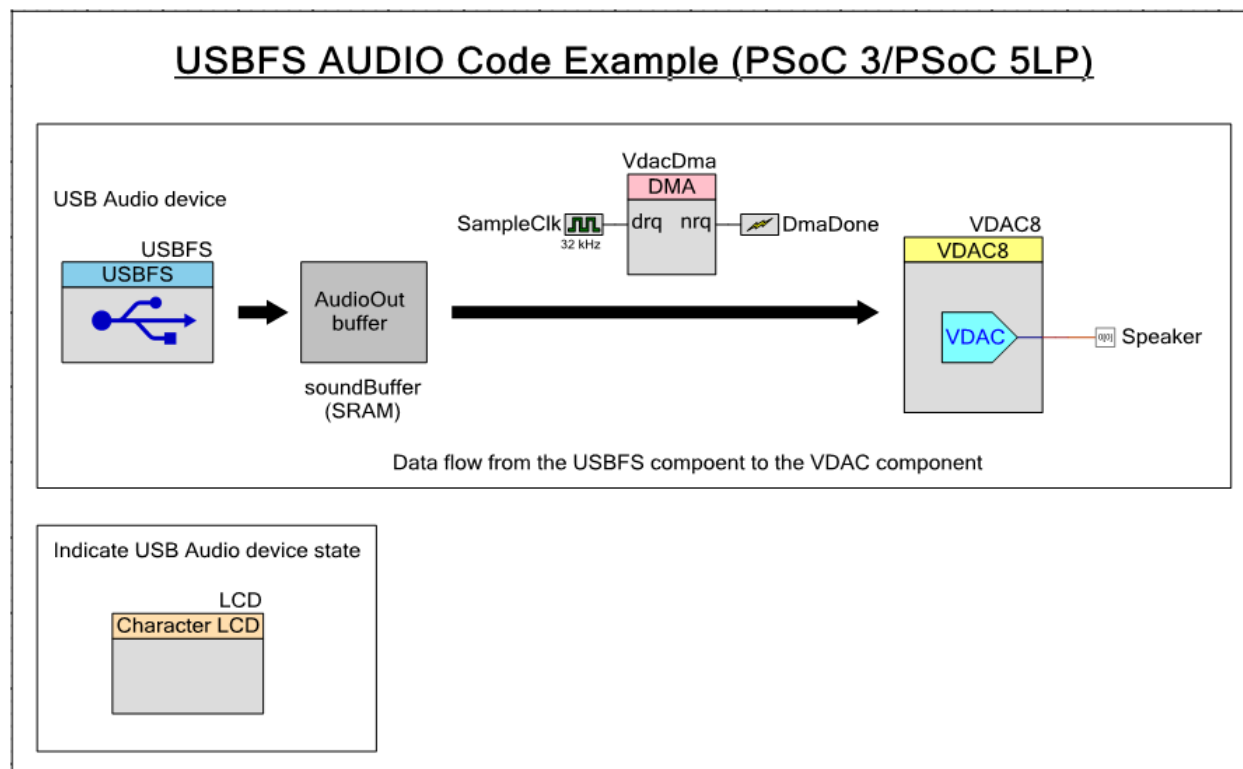
The important option of the USBFS component is the endpoint buffer management that defines the way how the endpoint buffers are serviced. This option is set to a DMA with Manual Buffer Management (Static Allocation). It means that data transfer between the endpoint memory and SRAM buffer is executed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

The DMA component is used to transfer audio data, received from the host, to the VDAC. The data bytes are supplied to the VDAC with a sample rate of 32 kHz. Therefore, the 32 kHz clock component is connected to the DMA request input. The DMA is configured to transfer 1 byte per request but the total transfer size is 32 bytes. The interrupt is connected to the DMA component to trigger when DMA transfer is done (each time when 32 bytes are transferred).

The VDAC8 component is an 8-bit voltage output Digital-to-Analog Converter (DAC) with the output range from 0 to 4.08 V (16 mV/bit). Its output is controlled by the data register written by DMA.

The LCD component is used to provide information about the data transfer direction in the 1<sup>st</sup> row (USB -> DMA -> VDAC) and whether audio is streamed to the device from the PC in the 2<sup>nd</sup> row (Audio On/Off).

Figure 1. Example Project Design Schematic



The important USBFS component configuration Tabs are shown below.

Figure 2. USBFS Descriptor Root

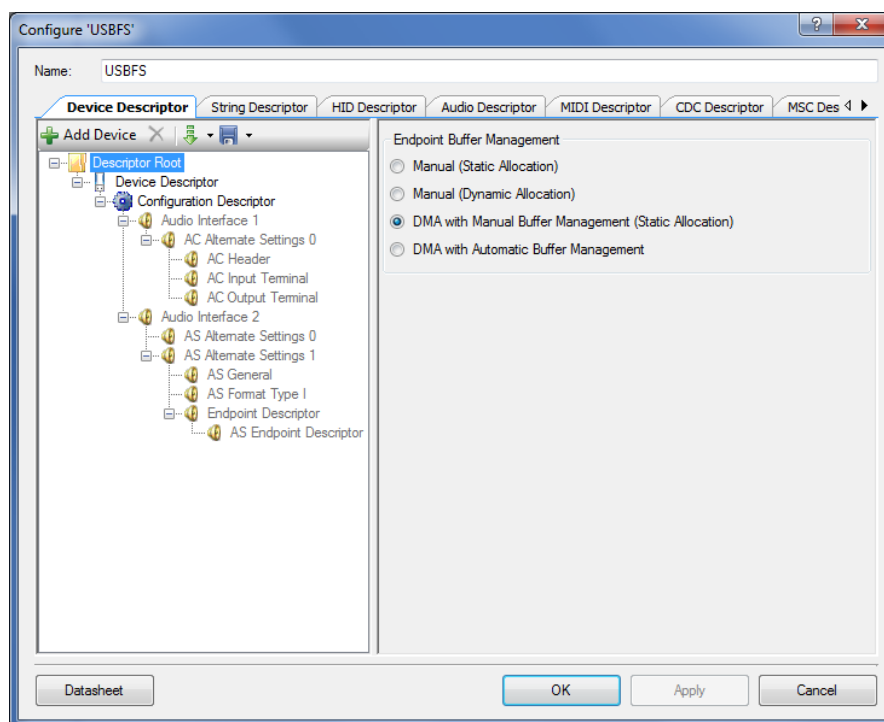


Figure 3. USBFS Audio Descriptor Alternate Settings 0

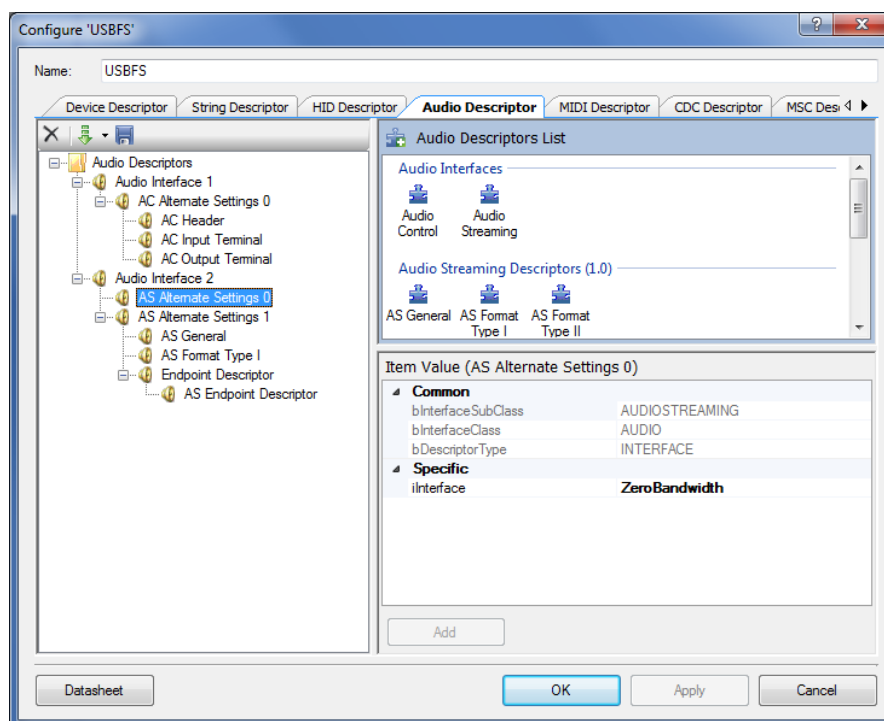


Figure 4. USBFS Audio Descriptor Alternate Settings 1

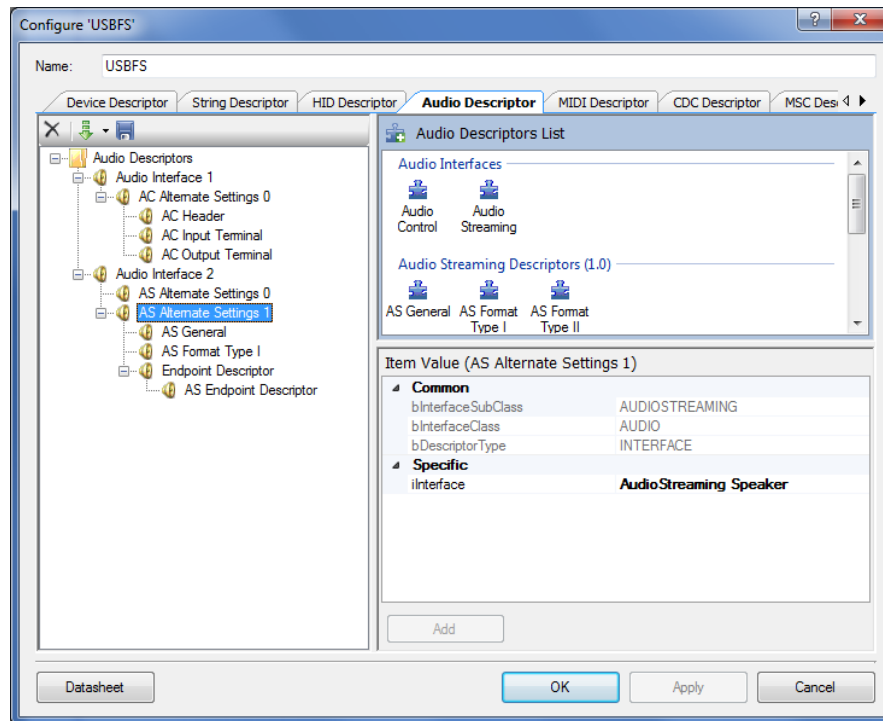


Figure 5. USBFS Audio Descriptor AS Format Type I

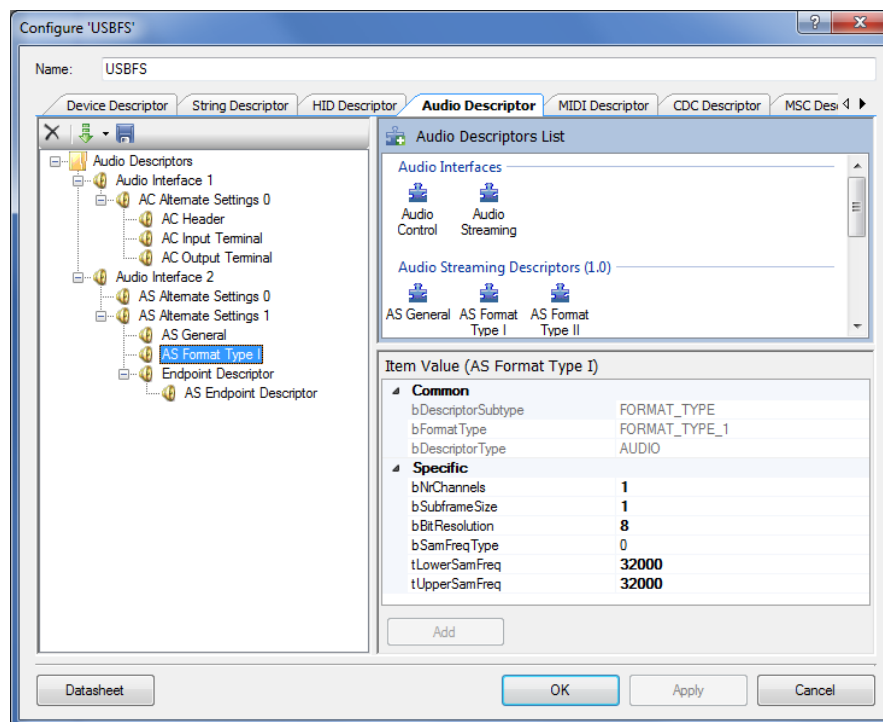
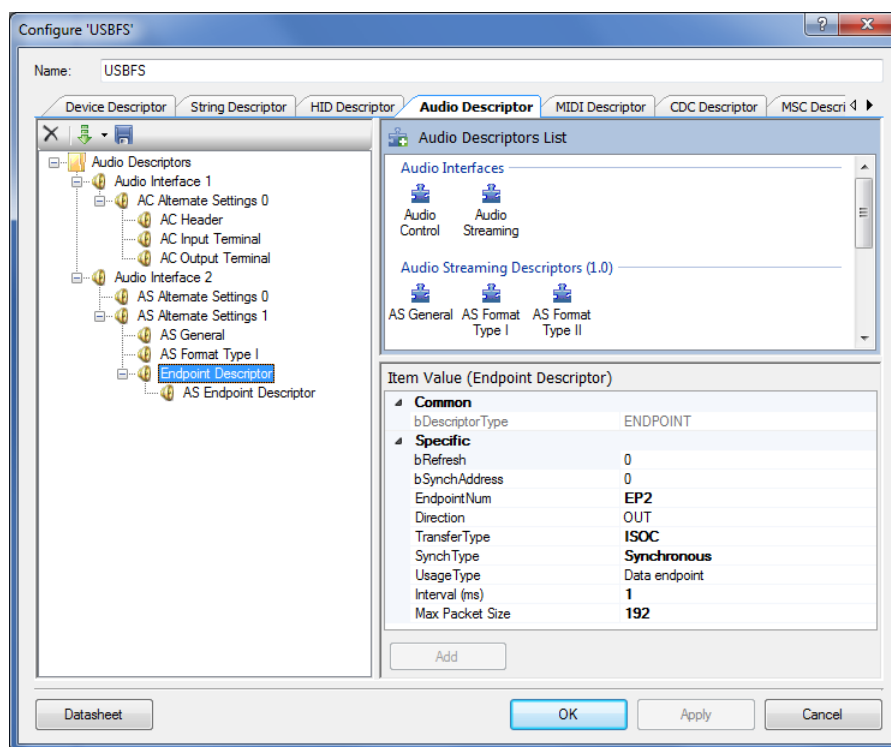


Figure 6. USBFS Audio Descriptor Endpoint



## Project Description

The initialization part of the main firmware routine consists of two major steps:

- Step 1 sets up the infrastructure to transfer data received via USB to VDAC by DMA. The 10 transfer descriptors are allocated to transfer 32 bytes from the data buffer to the VDAC data register. Each transfer descriptor transfers a chunk from the buffer (buffer size is  $32 * 10$  bytes): the 1<sup>st</sup> transfer descriptor transfers 1 – 32 buffer elements, the 2<sup>nd</sup> – 33 – 64 buffer elements and so on. The descriptors are successively chained and the last one is chained to the 1<sup>st</sup> to create a circle. The interrupt handler of the DMA done is registered to disable the DMA when underflow occurs. It triggers every time when 32 bytes have been transferred by DMA to VDAC.

**Note** PSoC does not support DMA direct transactions between the USBFS endpoints and other peripherals. All DMA transactions involving the USBFS endpoints (IN and OUT) must terminate or originate with the main system memory.

- Step 2 is configuration and start of the USBFS component operation. The global interrupts are enabled as they are mandatory. The code waits for completion of the enumeration process before entering a forever loop. This process starts by the host when the USB cable is connected to the PC.

After the USB Audio device was enumerated, the device waits for audio streaming. When audio streaming starts, the Interface 1 alternate settings are changed from 0 to 1 and the OUT

endpoint is enabled. 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. Then DMA starts transferring data to VDAC when the buffer is half-full.

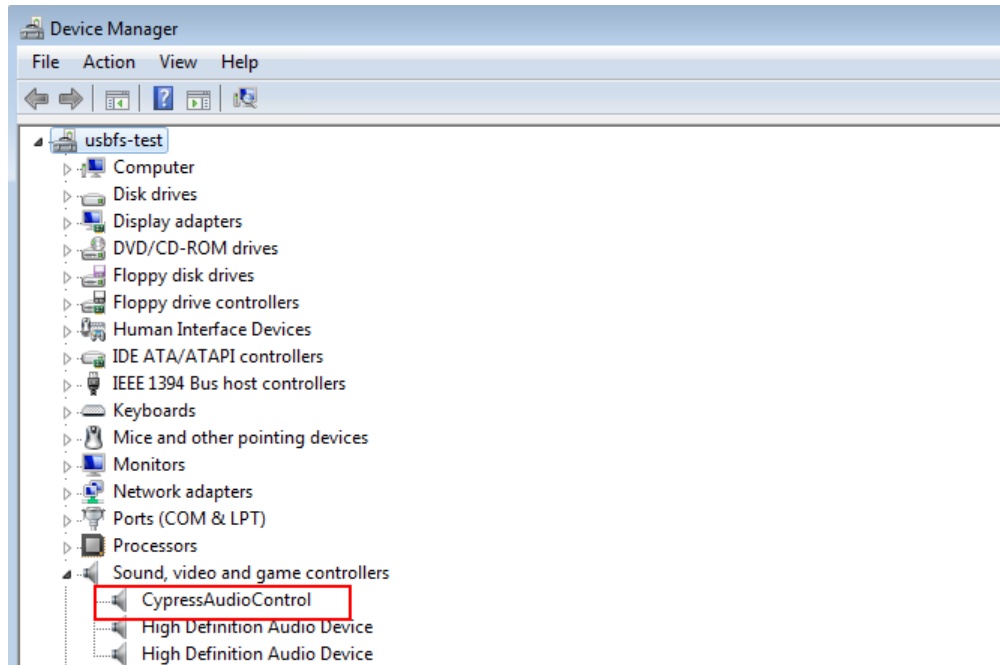
**Note** The internal 32 kHz clock of PSoC can be faster or slower than the PC. But none of synchronization schemes is implemented. Therefore, when underflow occurs, the DMA is stopped and enabled again after the buffer is half-full. When overflow occurs, the currently received data chunk (32 bytes) is skipped.

## Example Project Execution Flow

To execute the USBFS component code example, follow the procedure:

1. Connect the PSoC kit to the PC through the USB connector for programming.
2. Build the USBFS\_AUDIO\_PSoC3\_5LP example project and program into the device. The USB cable can be disconnected from the programming connector at this point.
3. Connect the speaker or oscilloscope to the Speaker pin to listen audio or observe the signal output.
4. Connect the kit to the PC through the USB connector for communication. The USB audio device does not require a driver and is automatically recognized by the system as CypressAudioControl.

Figure 7. CypressAudioControl Is Enumerated



5. Change the default speaker to the CypressAudioControl device. Follow the [link](#) to get instructions how to change the audio device in the Windows Media Player.

6. Play an audio file and listen or observe the speaker output. The example project contains audio file 1kHz\_sine\_wave.wav which generates a 1-kHz sine wave with 8-bit resolution and 32-kHz sample rate. It is recommended to use the audio file attached to this code example.

## Expected Results

If you follow the instructions above, the USB Audio device is enumerated by the PC and then you can play audio on the PC and observe it on the VDAC output.

© Cypress Semiconductor Corporation, 2015. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

PSoC® is a registered trademark, and PSoC Creator™ and Programmable System-on-Chip™ are trademarks of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are property of the respective corporations.

Any Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.