Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

CE225786 – PSoC 6 MCU: USB Audio Recorder

Objective

This example demonstrates the use of PSoC® 6 MCU to implement an audio recorder using the USB Audio Device Class.

Requirements

Tool: ModusToolbox™ IDE 1.0

Programming Language: C

Associated Parts: All PSoC 6 MCU parts with USB

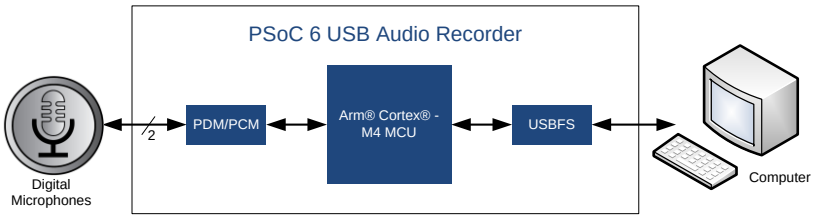
Related Hardware: PSoC 6 Wi-Fi-BT Pioneer Kit, PSoC 6 Wi-Fi-BT Proto Board

Overview

This code example shows how to stream audio data to a PC over USB using a PSoC 6 device. It uses digital microphones with the pulse-density modulation (PDM) to pulse-code modulation (PCM) converter hardware block. All the audio data captured by the microphones are streamed over USB Full-Speed (USBFS) using the Audio Device Class. An audio recorder software tool, such as Audacity, running on a PC initiates the streaming of audio data.

Figure 1 shows the high level-block diagram of this application.

Figure 1. Block Diagram



The PDM/PCM interface requires two wires – PDM Clock and PDM Data. Up to two digital PDM microphones can share the same PDM data line. One microphone samples in the falling edge and the other in the rising edge.

This code example uses FreeRTOS. Visit the FreeRTOS website for documentation and API references of FreeRTOS.

Hardware Setup

This example does not require any additional hardware to run.

Note: The PSoC 6 WiFi-BT Pioneer kit ships with KitProg2 installed. ModusToolbox only works with KitProg3. Before using this code example, make sure that the kit is upgraded to KitProg3. See ModusToolbox **Help** > **ModusToolbox IDE Documentation** > **User Guide,** Section **PSoC 6 MCU KitProg Firmware Loader**. If you do not upgrade, you will see an error like “unable to find CMSIS-DAP device” or “KitProg firmware is out of date”.

Operation

1. Connect the kit board to your PC using the provided USB cable through the USB connector.
2. Import the application projects into a new workspace or an existing one. See KBA225201.
3. Program the PSoC 6 MCU device. In the project explorer, select the **mainapp** project. In the Quick Panel, scroll to the **Launches** section and click **Program (KitProg3)** configuration.
4. Connect another USB cable (or reuse the same cable used to program the kit) to the USB Device connector [J28 for CY8CKIT-062-WiFi-BT and J10 for CY8CPROTO-062-4343W].
5. In the PC, verify that a new USB device was enumerated as a Microphone and named as “PSoC 6 USB Audio Recorder”.

6. Open an audio recorder software tool in the PC. If you do not have one, download the open-source software Audacity. Make sure to select the correct microphone in the software tool.
7. Start a recording session. Play a sound, or speak over the microphones in the kit.
8. Stop the recording session and play it to confirm that the audio was recorded correctly.

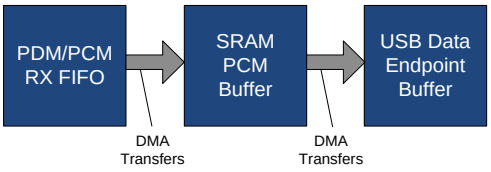
Debugging

You can debug the example to step through the code. Use a **Program+Debug** configuration. See KBA224621 in the Cypress community to learn how to start a debug session with ModusToolbox IDE.

Design and Implementation

The PDM-to-PCM hardware block can sample one or two PDM digital microphones. In this application, the hardware block is configured to sample stereo audio at 48 ksps with 16-bit resolution. The sample audio data is eventually transferred to the USB data endpoint buffer. Figure 2 shows the overall transfers performed by the application.

Figure 2. Overall Transfers



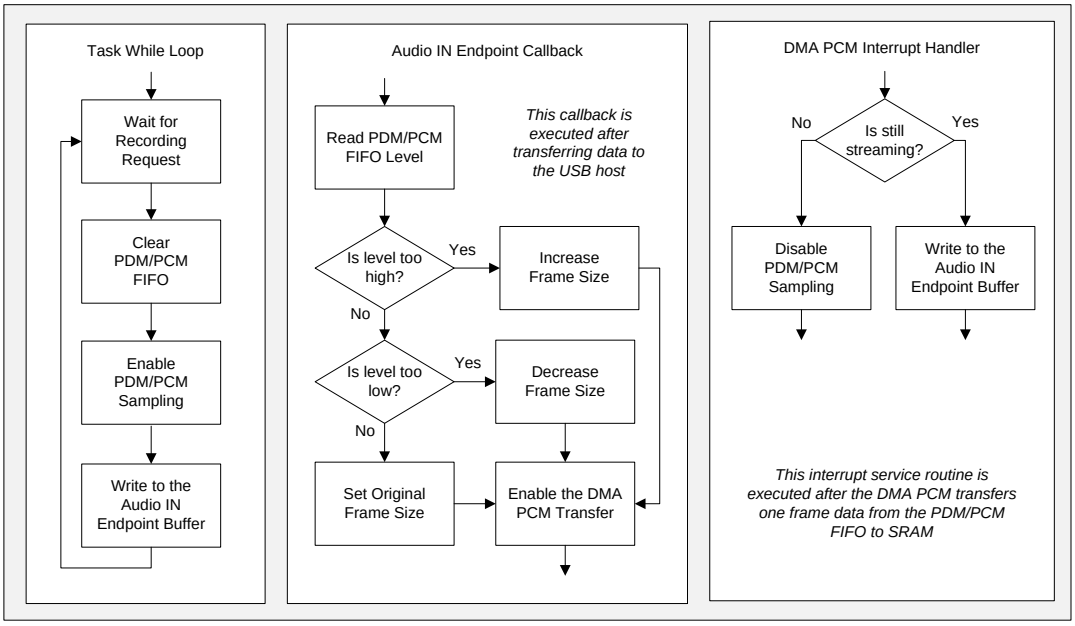
The first DMA transfer reads from the PDM/PCM RX FIFO and writes to a buffer in the SRAM. This buffer is populated with 16-bit interleaved data (left and right channels). The second DMA transfer reads from this buffer and writes to the USB Data Endpoint buffer. The second DMA transfer is handled automatically by the USBFS driver, while the first DMA transfer is part of the application layer.

The USB descriptor implements the Audio Device Class with three endpoints:

- **Audio Control Endpoint:** controls the access to the audio streams
- **Audio IN Endpoint:** sends data to the USB host
- **Audio OUT Endpoint:** receives data from the USB host (not used in this application)

Only the first two endpoints are used in this application; however, all the hooks in the firmware are placed to enable the Audio OUT Endpoint. Figure 3 shows the flowchart on how the PDM/PCM data is streamed over USB.

Figure 3. PDM/PCM to USB Flowchart



The audio in 'task while' loop waits for a recording request from the USB host. When it receives the request, it prepares the PDM/PCM block to sample a new frame. It initially writes a null frame to the Audio IN Endpoint buffer to initiate the streaming. The frame size depends on the sample rate (48 ksps), the number of channels (2x) and the time of USB transfers (1 ms). The overall equation is:

$$Frame\ size = Sample\ Rate \times Number\ of\ Channels \times Transfer\ Time$$

In this example, the frame size is equal to 48000 × 2 x 0.001 = 96 samples. Note that the Audio IN Endpoint Callback also implements a method to change the frame size depending on the FIFO level in the PDM/PCM block. This is important because there are clocking differences between the USB host bus and the PSoC 6 MCU audio subsystem.

The DMA PCM is configured to transfer an entire audio frame per trigger. It triggers when there is enough data in the PDM/PCM FIFO. Once it completes a transfer, it disables itself; it is enabled again only when the Audio IN Endpoint Callback is executed. On completion, the DMA PCM Interrupt Handler is executed, which might terminate the recording by disabling the PDM/PCM block or keep the streaming by writing more data to the Audio IN Endpoint buffer.

The firmware uses FreeRTOS to execute the processes required by this application. All tasks run in the Arm® Cortex®-M4 CPU. The following tasks are created:

1. **AudioInTask:** described in Figure 3 as the Task While Loop
2. **AudioAppTask:** initiates the application and handles certain USB events. In this application, no USB events are handled in this task.
3. **DebugTask:** handles messages printed to the terminal over UART (optional).

The example also uses the Event Group. It is used to notify tasks when USB events occur. Only the event that initiates a recording is used in this application.

The project consists of the following files:

- *main.c* (CM4) contains the main function, which is the entry point and execution of the firmware application. The main function sets up user tasks and then starts the RTOS scheduler. The CM0+ equivalent contains the main function that starts up CM4. CM0+ is not used to run any application code.
- *audio_app.c/h* files contain the task that initializes the audio-related blocks, such as the USB and the PDM/PCM blocks, and handles USB events in the task main loop.
- *audio_in.c/h* files contain the task that handle recording requests to the Audio IN endpoint. These files also implement the Audio IN Data Endpoint callback and the DMA interrupt service routine.

- *usb_comm.c/h* files contain all macros and functions related to the USBFS and USB Audio Device Class.
- *uart_debug.c/h* and *stdio_user.c/h* files are used for retargeting I/O functions such as printf via UART to a terminal emulation program.
- *FreeRTOSConfig.h* contains the FreeRTOS settings and configuration. Non-default settings are marked with inline comments. For details of FreeRTOS configuration options, see the FreeRTOS customization webpage.

Resources and Settings

Table 1 lists the ModusToolbox resources used in this example, and how they are used in the design. For pin usage and configuration, open the **Pins** tab of the *design.modus* file.

Table 1: ModusToolbox Resources

Resource	Alias	Purpose
PDM/PCM	KIT_PDM	Connects to the digital microphones.
UART (SCB)	KIT_UART	Debug interface.
USBFS	USBFS	Implements the Audio Device Class.
DMA DataWire	DMA_PCM	Transfers data from PDM/PCM FIFO to SRAM.

Figure 5 shows the configuration settings for KIT_PDM.

Figure 4. KIT_PDM Configuration

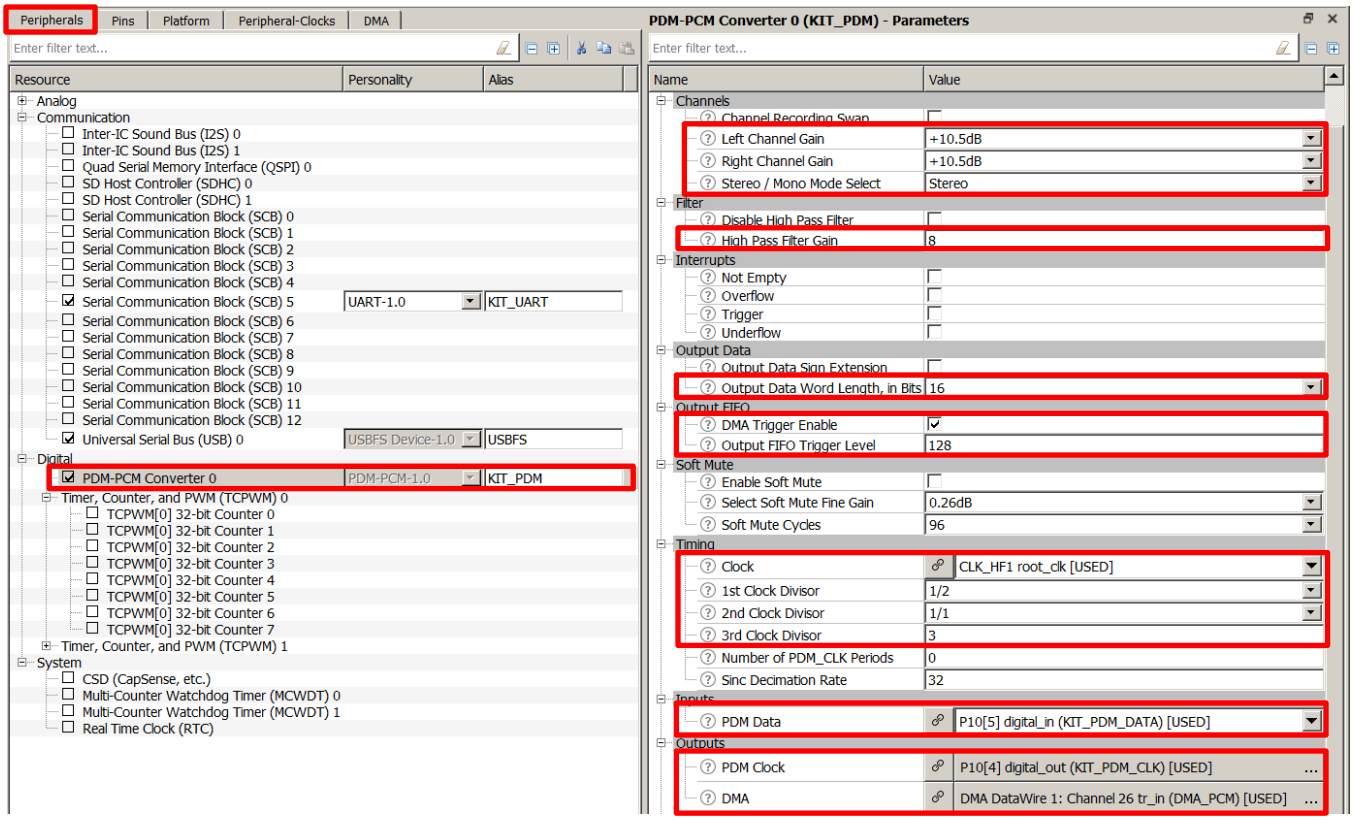
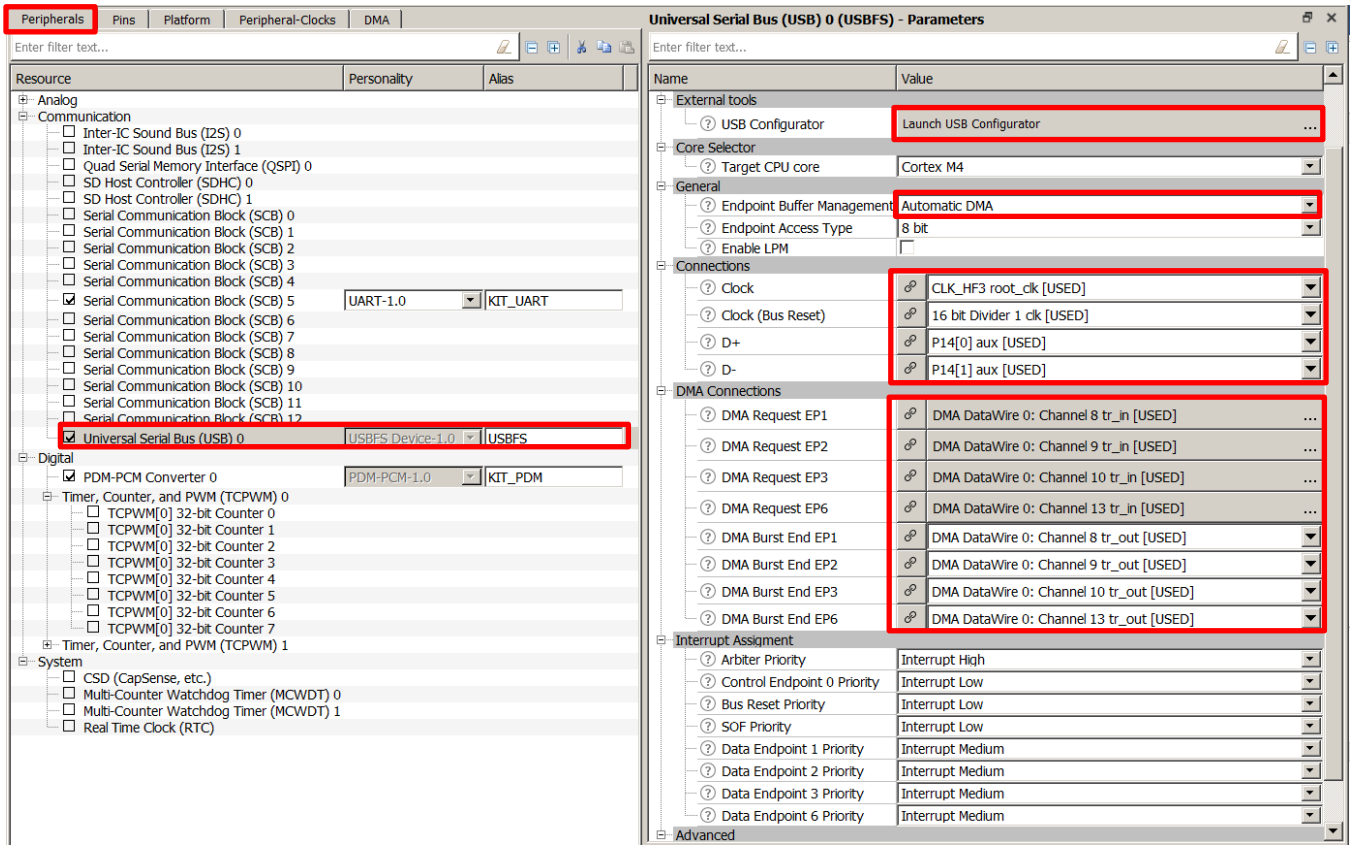


Figure 5 shows the configuration settings for the USBFS block. To visualize the USBFS descriptor, click in the **Launch USB Configuration** button from Figure 5. You can also refer to the *cycfg_usbdev.h* file in the Generated Source folder.

Figure 5. USBFS Configuration



Error! Not a valid bookmark self-reference. shows the configuration settings for DMA_PCM.

Figure 6. DMA_PCM Configuration

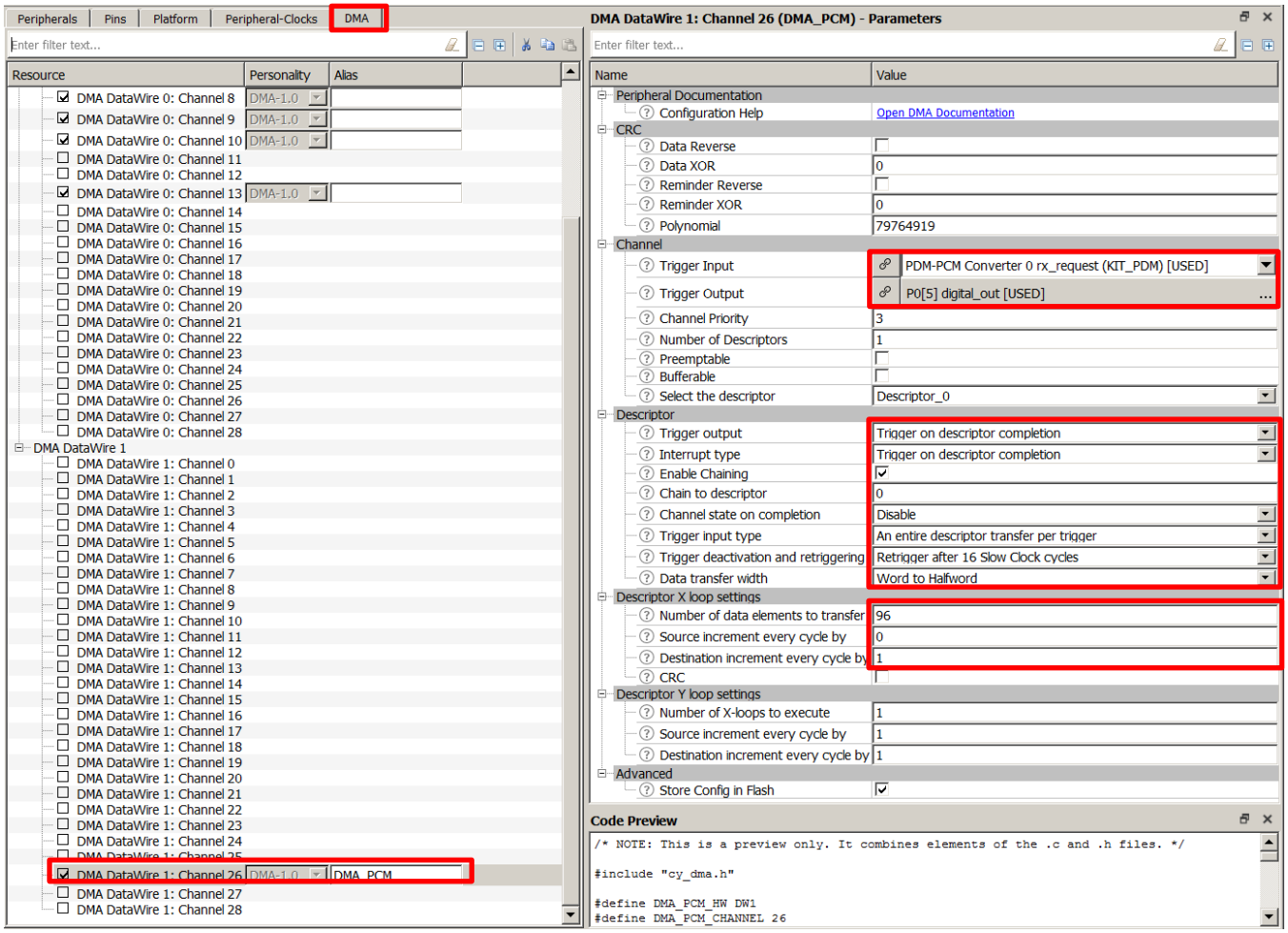


Figure 8 and Figure 8 show the platform system clock configuration. Note that the FLL and PLL are used in this application. The FLL is used to clock the CPUs and the USB block. The PLL is used to clock the audio subsystem.

Figure 7. FLL Configuration

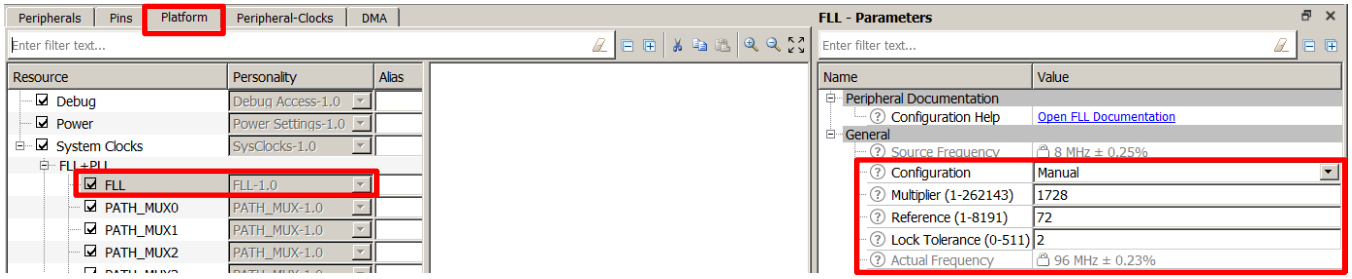


Figure 8. PLL Configuration

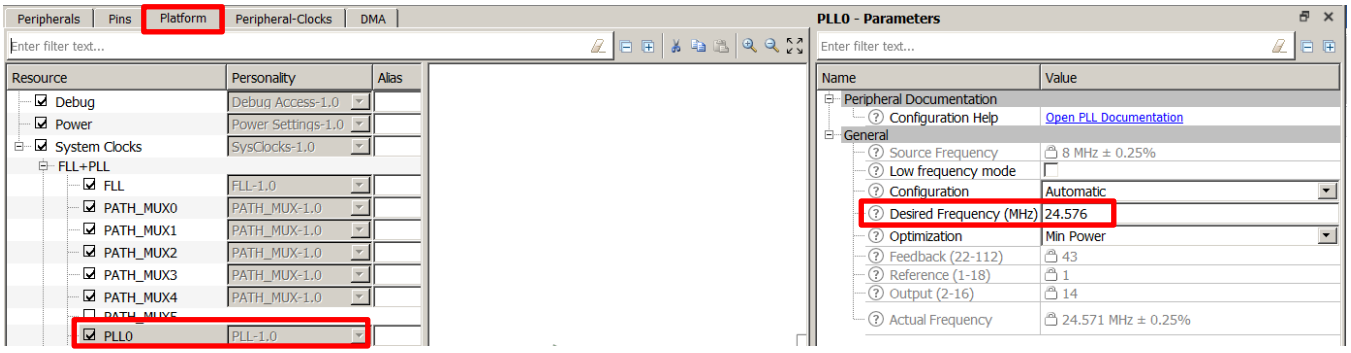
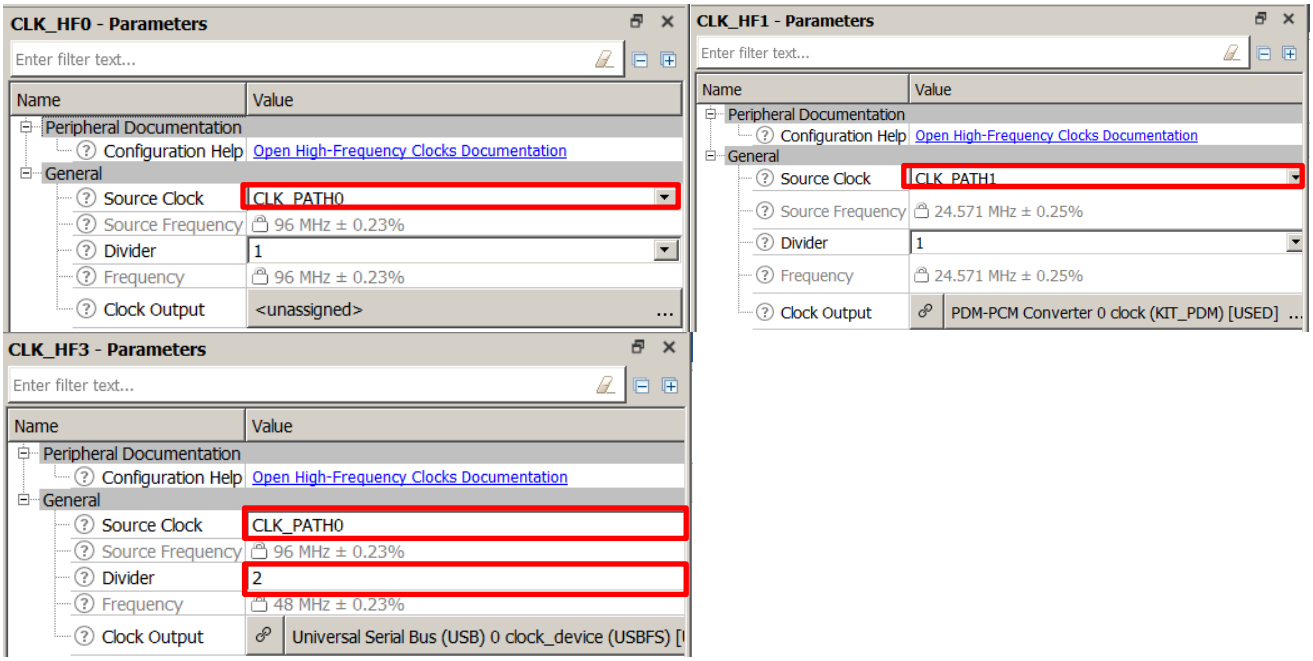


Figure 9 shows the high-frequency clock settings.

Figure 9. High-Frequency Clocks Configuration



KIT_UART uses the default configuration for UART.

The following middleware libraries are used in this project:

- FreeRTOS v10.0.1
- Retarget I/O
- USB Device

If you want to explore middleware, right-click the *CE225786_mainapp* folder in the project explorer window and select **ModusToolbox Middleware Selector**.

Reusing This Example

This example is designed for the CY8CPROTO-062-4343W kit. To port the design to a different PSoC 6 MCU device, right-click an application project and choose **Change ModusToolbox Device**. Depending on the device chosen, you might need to redefine the DMA configuration for the PDM/PCM DMA (DMA_PCM). Refer to Figure 6 to re-create the DMA_PCM configuration for the new device.

In some cases, a resource used by a code example (for example, an IP block) 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 what a particular device supports.

Related Documents

Application Notes	
AN221774 – Getting Started with PSoC 6 MCU	Describes the PSoC 6 MCU devices and how to build your first PSoC project
AN215656 – PSoC 6 MCU Dual-Core CPU System Design	Describes the dual-CPU architecture in the PSoC 6 MCU, and shows how to build a simple dual-CPU design
Code Examples	
Visit the Cypress GitHub site for a comprehensive collection of code examples using ModusToolbox IDE	
Device Documentation	
PSoC 6 MCU Datasheets	PSoC 6 MCU Technical Reference Manuals
Development Kit Documentation	
CY8CKIT-062-WiFi-BT PSoC 6 WiFi-BT Pioneer Kit	
CY8CPROTO-062-4343W PSoC 6 Wi-Fi BT Prototyping Kit	
Tool Documentation	
ModusToolbox IDE	ModusToolbox IDE 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.

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

Document History

Document Title: CE225786 – PSoC 6 MCU: USB Audio Recorder

Document Number: 002-25786

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	6410738	RLOS	12/03/2018	New code example

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 | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

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

© Cypress Semiconductor Corporation, 2018. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. No computing device can be absolutely secure. Therefore, despite security measures implemented in Cypress hardware or software products, Cypress does not assume any liability arising out of any security breach, such as unauthorized access to or use of a Cypress product. In addition, the products described in these materials may contain design defects or errors known as errata which may cause the product to deviate from published specifications. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.