

Objective

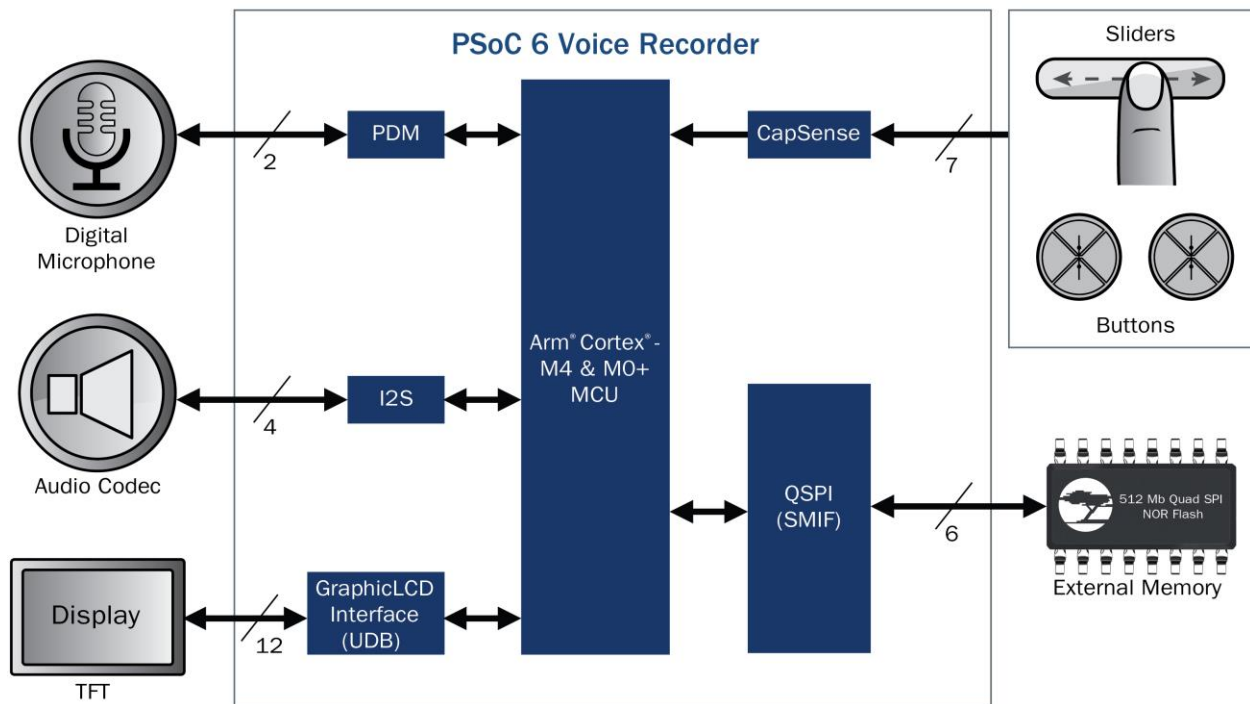
This example demonstrates the use of PSoC® 6 MCU to implement a voice recorder.

Overview

This project shows how PSoC 6 MCU can be used to record audio data, store it and play it back. It uses a digital microphone with the PDM/PCM hardware block. All the audio data captured by the microphone is stored to an external flash memory. After the recording is completed, you can play the audio data over I²S, which interfaces with an audio codec. You can record/stop/play/pause/resume with CapSense® buttons. You can control the audio volume with the CapSense slider. The TFT LCD displays the current state of the voice recorder, the volume, and the time of the record/play.

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

Figure 1. Block Diagram



Requirements

Tool: PSoC Creator™ 4.2; Peripheral Driver Library (PDL) 3.0.1

Programming Language: C (Arm® GCC 5.4.1)

Associated Parts: All PSoC 6 MCU parts

Related Hardware: CY8CKIT-062-WiFi-BT, CY8CKIT-028-TFT

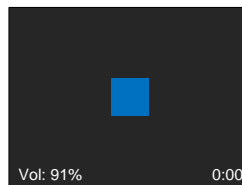
Hardware Setup

This example requires the CY8CKIT-028-TFT shield to be connected to CY8CKIT-062-WiFi-BT PSoC 6 Pioneer Kit. Keep SW5 and SW7 in their default positions. Refer to the Kit Guide for more information. You also need a headphone or speaker connected to the audio jack on the CY8CKIT-028-TFT shield. The SW1 position should match the type of headphone/speaker used – OMTP or AHJ.

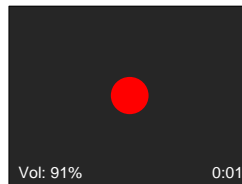
Operation

1. Connect the CY8CKIT-028-TFT shield to the Pioneer Kit.
2. Connect a headphone or speaker to the audio jack on the CY8CKIT-028-TFT.
3. Connect the Pioneer Kit to your PC using the provided USB cable through the USB connector (J10).
4. Build the project and program it into the PSoC 6 MCU device. Choose **Debug > Program**. When building the project, DO NOT replace the **FreeRTOSConfig.h** file. For more information on device programming, see PSoC Creator Help. Flash for both CPUs is programmed in a single program operation.

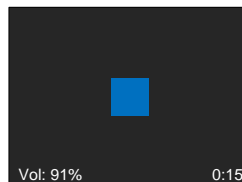
After pressing the RST button, the following screen appears on the TFT LCD display.



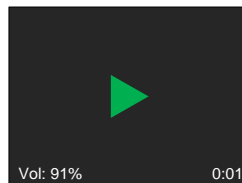
5. Press the left CapSense button (BTN0) on CY8CKIT-062 to start recording. The following screen appears on the TFT LCD.



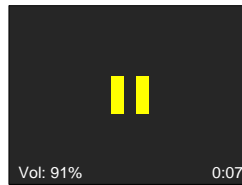
6. Play a sound, or speak over the microphone (PDM MIC) on CY8CKIT-028-TFT while recording.
7. Press the left CapSense button again to stop recording. The following screen appears on the LCD.



8. Press the right CapSense button (BTN1) to play the recording. The following screen appears on the LCD.



9. Listen to the recorded audio using the headphone or speaker. You can pause/resume any time by pressing the right CapSense button again. If paused, the following screen appears on the LCD:

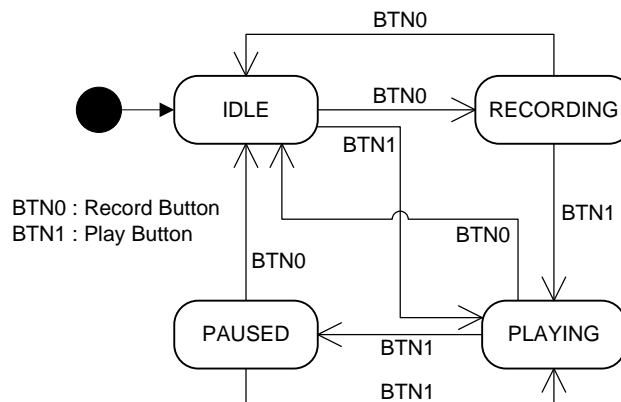


10. To change the volume, slide your finger on the CapSense slider to the right (increase volume) or to the left (decrease volume).

Design and Implementation

The CY8CKIT-028-TFT shield contains the audio codec [AK4954A](#), an audio jack, a digital microphone, and a TFT LCD. This allows you to record audio using the microphone and play it with the audio codec. The display is used to report the current state of the voice recorder – IDLE, RECORDING, PLAYING, or PAUSED. The Pioneer kit contains two CapSense buttons and a slider. The buttons trigger the actions supported by the voice recorder. Figure 2 shows how the transitions between states occur.

Figure 2. States and Transitions

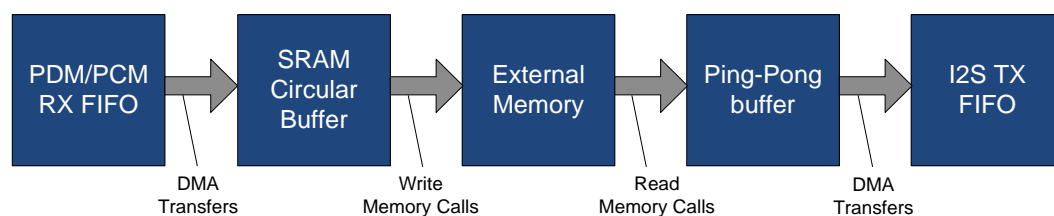


Each state is linked to a different icon displayed in the LCD. The Operation section shows the screenshots of each state. The screen also shows the current volume in percent, and the time duration of the record/play. The GraphLCDIntf Component drives the LCD. It is configured to have an 8-bit wide bus. The example uses an open-source graphics library for embedded system called [µGUI](#). The firmware uses this library to draw text and figures on the LCD display.

When recording, the PDM/PCM hardware block captures the data from the digital microphone. All the data written to its internal FIFO is transferred (using DMAs) to a circular buffer placed in the SRAM. The DMA is configured to generate interrupts when a certain amount of data is transferred. On each interrupt, the application transfers data from the circular buffer to the external memory over QuadSPI (SMIF). This memory is placed on Pioneer Kit and uses NOR flash technology ([S25FL512S](#)).

When playing, the I²S hardware block streams the recorded data. The application reads the data from the external memory and places it in a ping-pong buffer. While writing in the ping buffer, a DMA controller transfers the data from the pong buffer to the I²S TX FIFO. [Figure 3](#) shows the overall transfers performed by the application.

Figure 3. Overall Transfers



When in the IDLE or PAUSED states, no transfers are performed.

To minimize writing in the same sector of the external memory multiple times, a wearing level mechanism is implemented. On initialization, the application scans the external memory to locate the last sector written by the application. Every time a new record is started, the application erases and writes new data in the next sector available. This mechanism reduces the number of times the same sector is erased/written. The same sector will only be erased/written when all other sectors are used.

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. RecorderTask: handles recording and playing. It controls the transfers between the FIFOs, SRAM, and external memory.
2. TouchTask: handles CapSense touches on the buttons and slider.
3. EventsTask: handles any events that occur, such as touches from CapSense or recording/playing events.
4. GraphicsTask: handles updates and draws on the LCD.

Other RTOS elements used for synchronization and communication are:

1. Event Queue: used to notify EventsTask when specific events occur. The RecorderTask and TouchTask are senders.
2. GUI Queue: used to notify GraphicsTask to update or draw something on the screen. RecorderTask and EventsTask are senders.
3. SMIF Semaphore: used to lock the SMIF interface for accessing the external memory.
4. DMA Event Group Bits: used to notify RecorderTask that a DMA interrupt occurred.

Components and Settings

Table 1 lists the PSoC Creator Components used in this example, how they are used in the design, and the non-default settings required so they function as intended.

Table 1: PSoC Creator Components

Component	Instance Name	Purpose	Non-default Settings
PDM to PCM	PDM_PCM	Interface the digital microphone	See Figure 4
SMIF	SMIF_1	Interface the external flash memory	SMIF Datalines[0:1] checked SMIF Datalines[2:3] checked SMIF SPI Slave Select 0 checked Generate code from cy_smif.cysmif file checked All other parameters unchecked See Figure 5 for SMIF Configuration Tool (Launch by clicking with right mouse button over the SMIF component)
DMA	DMA_Record	Transfer data from PDM/PCM RX FIFO to SRAM	Trigger Input/Output are checked See Figure 6
DMA	DMA_PlayLeft	Transfer data from SRAM to I2S TX FIFO (left channel)	Trigger Input/Output are checked See Figure 7
DMA	DMA_PlayRight	Transfer data from SRAM to I2S TX FIFO (right channel)	Trigger Input/Output are checked See Figure 8
I2S	I2S	Interface the audio codec	See Figure 9
SCB (I2C Master)	Codecl2CM	Configure the audio codec	Mode: Master
CapSense	CapSense	Scan capacitive buttons and a slider	See Figure 10 to Figure 13
GraphicLCDIntf	GraphicLCDIntf	Interface the TFT LCD Display	
Interrupt	DMA_PDM_IRQ	Track the recording transfers	
Interrupt	DMA_I2S_IRQ	Track the playing transfers	
Clock	Clk_HF4	Master clock for the audio codec	

Clock	Clock_Graphics	Clock for GraphicLCDIntf	
Digital Input Pin	PDM_DATA	PDM Data input	
Digital Output Pin	PDM_CLK	PDM Clock output	
Digital Output Pin	TX_SDO	I2S TX Data output	
Digital Output Pin	TX_SCK	I2S TX Clock output	
Digital Output Pin	TX_WS	I2S TX Word Select output	
Digital Output Pin	Pin_d_c	Data/Command signal for the LCD	
Digital Output Pin	Pin_ncs	Active-LOW chip select for the LCD	
Digital Output Pin	Pin_nwr	Active-LOW write control signal for LCD	
Digital Output Pin	Pin_nrd	Active-LOW read control signal for LCD	
Digital Output Pin	Intf_nreset	Active-LOW reset signal for the LCD	HW Connection is Unchecked
Bidirectional Pin	Pin_LSB	8-bit pin bus for the LCD	Number of pins: 8
Digital Output Pin	RED_LED	Assert when an error is detected	Initial drive state: High HW Connection is Unchecked

For information on the hardware resources used by a Component, see the Component datasheet.

Figure 4 through Figure 10 highlight the non-default settings for each Component in this example.

Figure 4. PDM/PCM Configuration Window

Configure 'PDM_PCM'
 ? x

Name: PDM_PCM

Basic Built-in

Channels		
Channel Recording Swap	<input type="checkbox"/>	$f(x)$
Left Channel Gain	0dB	$f(x)$
Right Channel Gain	+10.5dB	$f(x)$
Stereo / Mono Mode Select	Mono R	$f(x)$
Filter		
Disable High Pass Filter	<input type="checkbox"/>	$f(x)$
High Pass Filter Gain	8	$f(x)$
Interrupts		
RX FIFO is Not Empty Interrupt	<input type="checkbox"/>	$f(x)$
RX FIFO Overflow Interrupts	<input type="checkbox"/>	$f(x)$
RX FIFO Trigger Interrupts	<input type="checkbox"/>	$f(x)$
RX FIFO Underflow Interrupts	<input type="checkbox"/>	$f(x)$
Output Data		
Output Data Sign Extension	<input type="checkbox"/>	$f(x)$
Output Data Word Length, in Bits	16	$f(x)$
Output FIFO		
DMA Trigger Enable	<input checked="" type="checkbox"/>	$f(x)$
Output FIFO Trigger Level	64	$f(x)$
Soft Mute		
Enable Soft Mute	<input type="checkbox"/>	$f(x)$
Select Soft Mute Fine Gain	0.26dB	$f(x)$
Soft Mute Cycles	96	$f(x)$
Timing		
1st Clock Divisor	1/4	$f(x)$
2nd Clock Divisor	1/4	$f(x)$
3rd Clock Divisor	1	$f(x)$
Number of PDM_CLK Periods	0	$f(x)$
Sinc Decimation Rate	32	$f(x)$

Datasheet OK Apply Cancel

Figure 5. SMIF Tool Configuration

SMIF Configuration Tool: C:\Users\rlos\Perforce\rlos_RLOSCMSLAP_2560\software\products\Code Examples\CE222221\CE222221_V...

File Run Options Help

PSOC 6

Slave slot	Memory part number	Data select	Memory mapped	Pair with slot	Start address	Size	End address	Write enable	Config data in flash	Encrypt
0	S25FL512S	Quad SPI-Data[0:3]	<input checked="" type="checkbox"/>	None	0x18000000	0x10000	0x1800FF...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
1	Not used	SPI-MOSI:MISO Data[0:1]	<input type="checkbox"/>	None	0x18010000	0x10000	0x1801FF...	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2	Not used	SPI-MOSI:MISO Data[0:1]	<input type="checkbox"/>	None	0x18020000	0x10000	0x1802FF...	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3	Not used	SPI-MOSI:MISO Data[0:1]	<input type="checkbox"/>	None	0x18030000	0x10000	0x1803FF...	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Location: C:\Program Files (x86)\Cypress\PDL\3.0.1\tools\win\smif_config\memory\S25FL512

User part number: S25FL512S Erase time: 520 ms

Status register busy mask: 0x01 Chip erase time: 134 s

Status register quad enable mask: 0x02 Program time (us): 340

Size of memory: 0x04000000 Description: 64Mbytes 3V serial Flash memory

Program page size: 0x00000200

Erase block size (bytes): 0x00040000

Number of address bytes for SMIF transactions: 0x03

Description	Number	Command width	Address width	Mode	Mode width	Dummy cycles	Data width
Read command format	0xEB	Single	Quad	0x01	Quad	4	Quad
Write enable command format	0x06	Single	Single	NA	Single	NA	Single
Write disable command format	0x04	Single	Single	NA	Single	NA	Single
Erase command format	0xD8	Single	Single	NA	Single	NA	Single
Chip erase command format	0x60	Single	Single	NA	Single	NA	Single
Program command format	0x38	Single	Single	NA	Quad	NA	Quad
Read status register command (containing QE bit)	0x35	Single	Single	NA	Single	NA	Single
Read status register command (containing WIP bit)	0x05	Single	Single	NA	Single	NA	Single
Write status register command (containing QE bit)	0x01	Single	Single	NA	Single	NA	Single

Figure 6. DMA_Record Configuration Window

Configure 'DMA_Record'

Name: DMA_Record

Channel Descriptors Built-in

...PDM_to_SRAM

Descriptor	
Descriptor Name	PDM_to_SRAM
Trigger output	Trigger on every element transfer completion
Interrupt	Trigger on every X loop transfer completion
Chain to descriptor	PDM_to_SRAM
Channel state on completion	Enable
Input trigger options	
Trigger input type	One transfer per trigger
Trigger deactivation and retriggering	Retrigger after 16 Clk Slow cycles
Transfer setting	
Data element size	Halfword (2 bytes)
Source and destination transfer width	Word to Halfword
X loop transfer	
Number of data elements to transfer	256
Source increment every cycle by	0
Destination increment every cycle by	1
Y loop transfer	
Number of X-loops to execute	32
Source increment every cycle by	0
Destination increment every cycle by	256

Data element size
This parameter sets up the data element size parameter in the descriptor.

Datasheet OK Apply Cancel

Figure 7. DMA_PlayLeft Configuration Window

Configure 'DMA_PlayLeft'

Name: DMA_PlayLeft

Channel Descriptors Built-in

...SRAM_to_I2S

Descriptor	
Descriptor Name	SRAM_to_I2S
Trigger output	Trigger on every element transfer completion
Interrupt	Trigger on completion of entire descriptor chain
Chain to descriptor	SRAM_to_I2S
Channel state on completion	Enable
Input trigger options	
Trigger input type	One transfer per trigger
Trigger deactivation and retriggering	Retrigger after 16 Clk Slow cycles
Transfer setting	
Data element size	Halfword (2 bytes)
Source and destination transfer width	Halfword to Word
X loop transfer	
Number of data elements to transfer	256
Source increment every cycle by	1
Destination increment every cycle by	0
Y loop transfer	
Number of X-loops to execute	2
Source increment every cycle by	256
Destination increment every cycle by	0

Data element size
This parameter sets up the data element size parameter in the descriptor.

Datasheet OK Apply Cancel

Figure 8. DMA_PlayRight Configuration Window

Configure 'DMA_PlayRight'

Name: DMA_PlayRight

Channel Descriptors Built-in

SRAM_to_I2S

Descriptor	
Descriptor Name	SRAM_to_I2S
Trigger output	Trigger on every element transfer completion
Interrupt	Trigger on every X loop transfer completion
Chain to descriptor	SRAM_to_I2S
Channel state on completion	Enable
Input trigger options	
Trigger input type	One transfer per trigger
Trigger deactivation and retriggering	Retrigger after 16 Clk_Slow cycles
Transfer setting	
Data element size	Halfword (2 bytes)
Source and destination transfer width	Halfword to Word
X loop transfer	
Number of data elements to transfer	256
Source increment every cycle by	1
Destination increment every cycle by	0
Y loop transfer	
Number of X-loops to execute	2
Source increment every cycle by	256
Destination increment every cycle by	0

Data element size
This parameter sets up the data element size parameter in the descriptor.

Datasheet OK Apply Cancel

Figure 9. I2S Configuration Window

Configure 'I2S_PDL_v2_0'

Name: I2S

Basic Built-in

☐ Clock from terminal Input clock frequency (kHz): 16384

Clock divider: 8

☒ **TX**

Mode: Master

Bit rate (kbps): 256

Alignment: I2S mode

Channels: 2

Channel length: 16

Word length: 16

Frame rate (kpsps): 8

Overhead value: 1

WS pulse width: 1 channel length

SDO latching time: Normal

Output clock polarity: Normal

FIFO trigger level: 253

☒ DMA trigger

Interrupts:

☐ Trigger

☐ Not full

☐ Empty

☐ Overflow

☐ Underflow

☐ Watchdog: FFFFFFFF

☐ **RX**

Mode: Master

Bit rate (kbps): 256

Alignment: I2S mode

Channels: 2

Channel length: 16

Word length: 16

Frame rate (kpsps): 8

Sign extension: 0

WS pulse width: 1 channel length

SDI latching time: Normal

Output clock polarity: Normal

FIFO trigger level: 0

☐ DMA trigger

Interrupts:

☐ Trigger

☐ Not empty

☐ Full

☐ Overflow

☐ Underflow

☐ Watchdog: FFFFFFFF

Datasheet OK Apply Cancel

Figure 10. CapSense Configuration Window - Basic

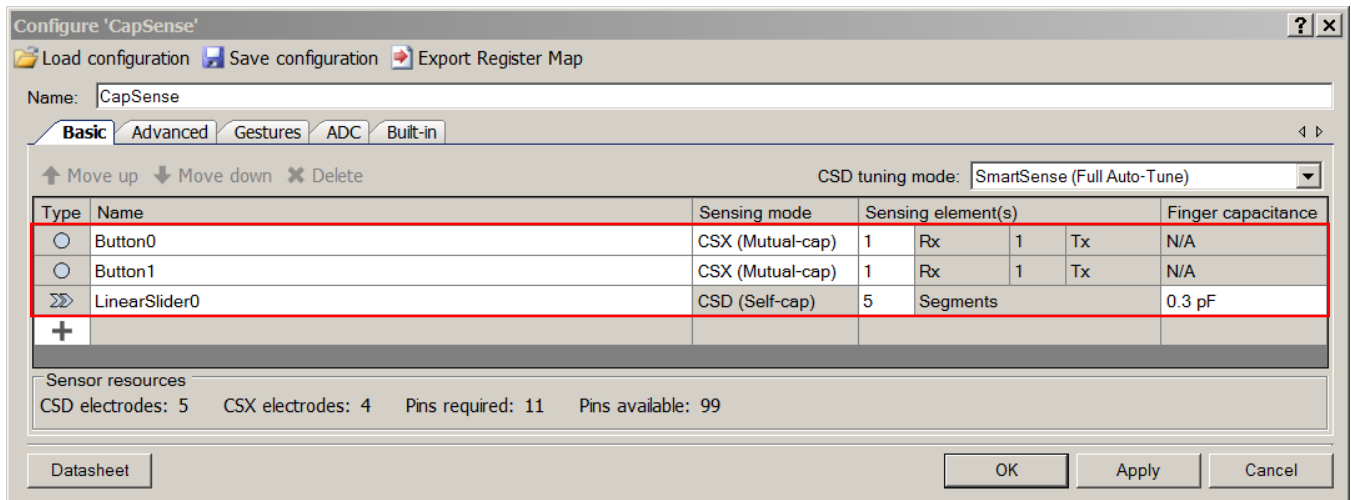


Figure 11. CapSense Configuration Window – Advanced / General

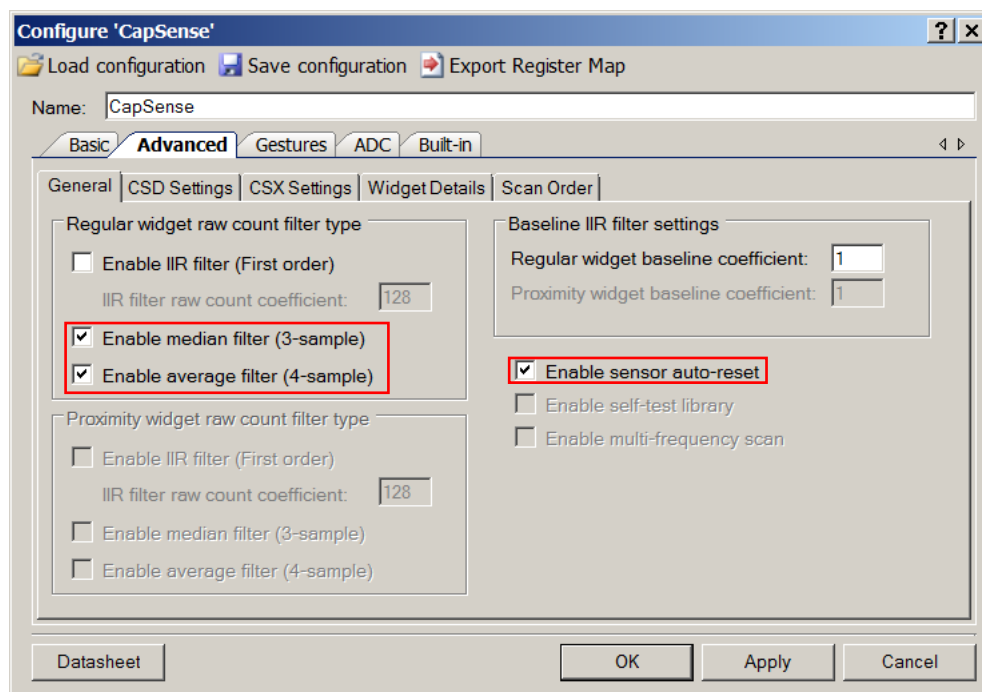


Figure 12. CapSense Configuration Windows - Advanced / CSD and CSX Settings

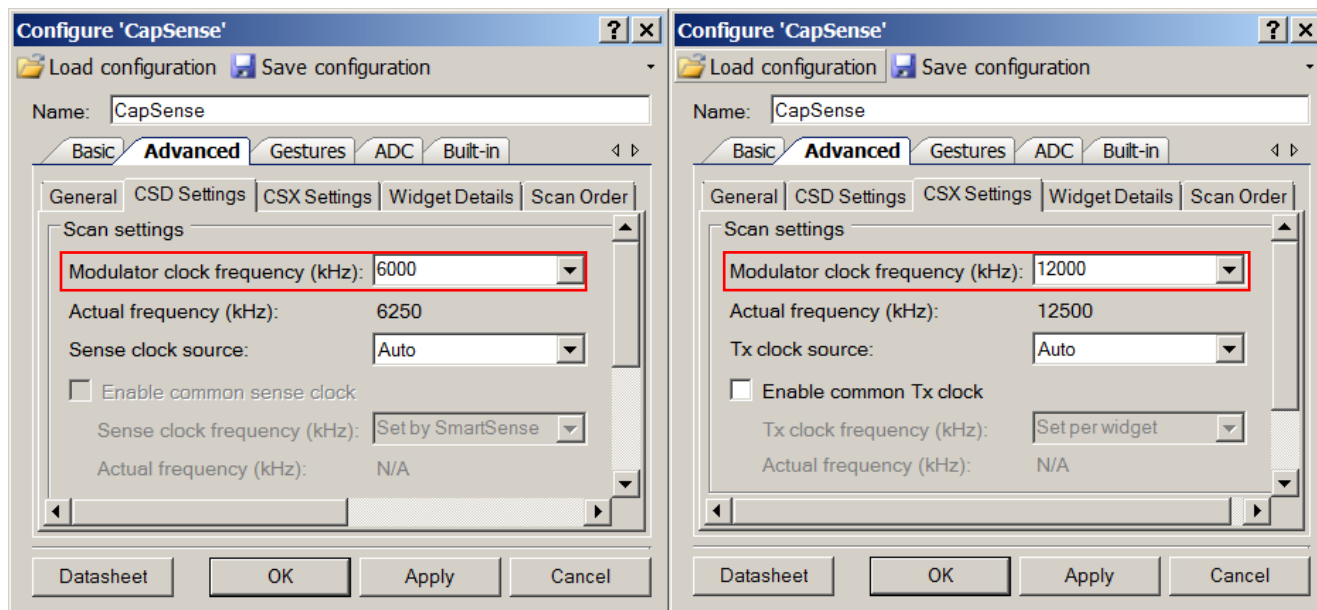


Figure 13. CapSense Configuration Window – Advanced / Widget details / Buttons

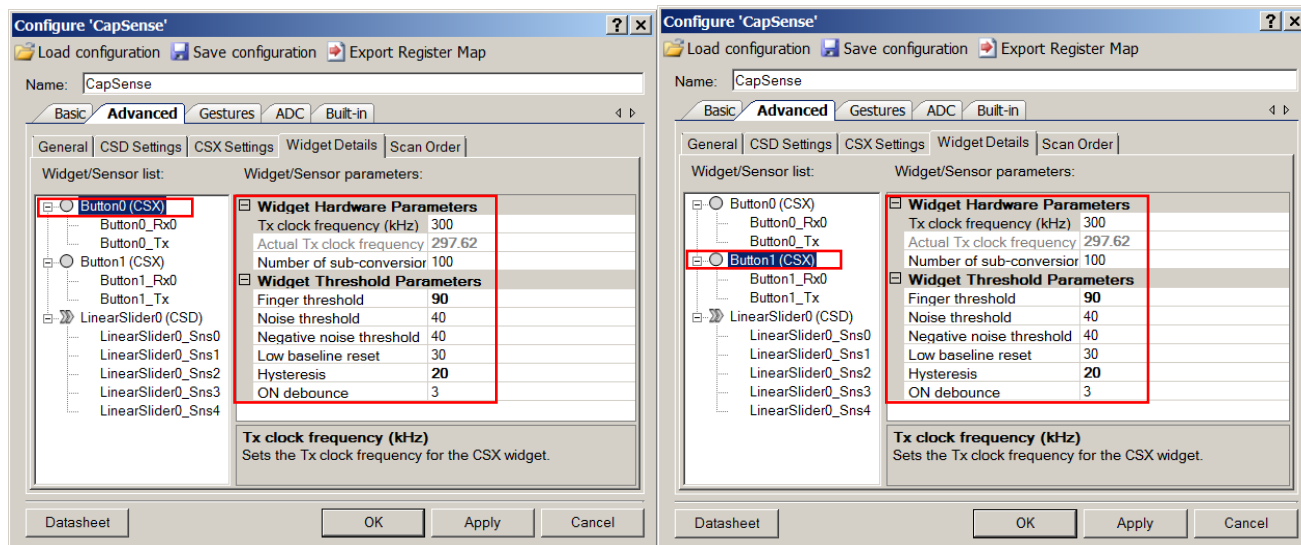


Figure 14 through Figure 16 shows the changes made in the clock configuration.

Figure 14. Source Clocks Configuration

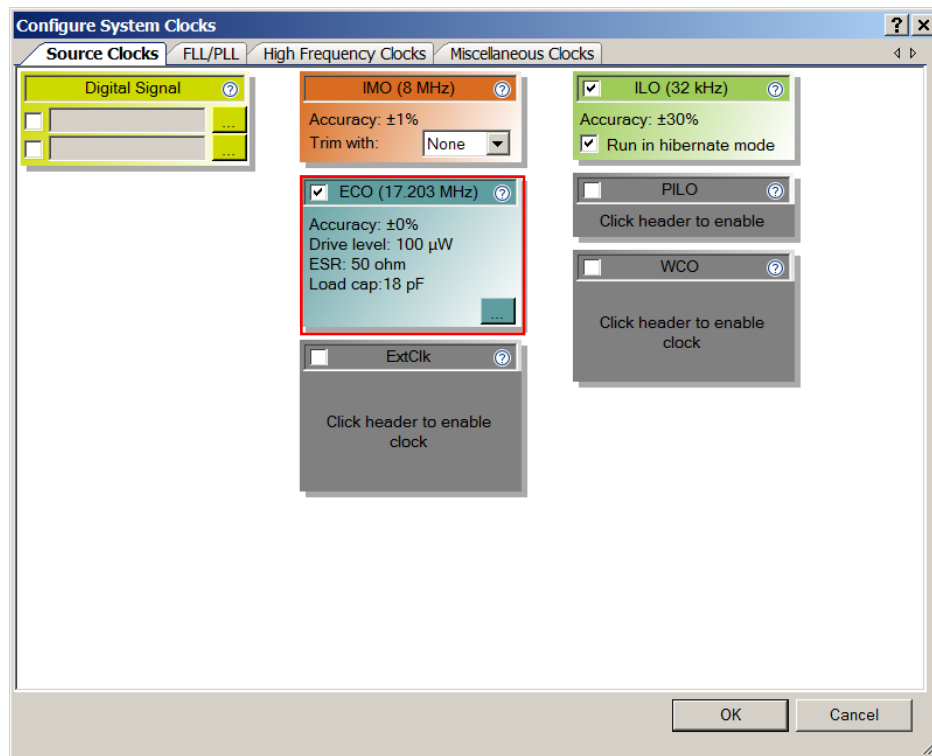


Figure 15. FLL/PLL Configuration

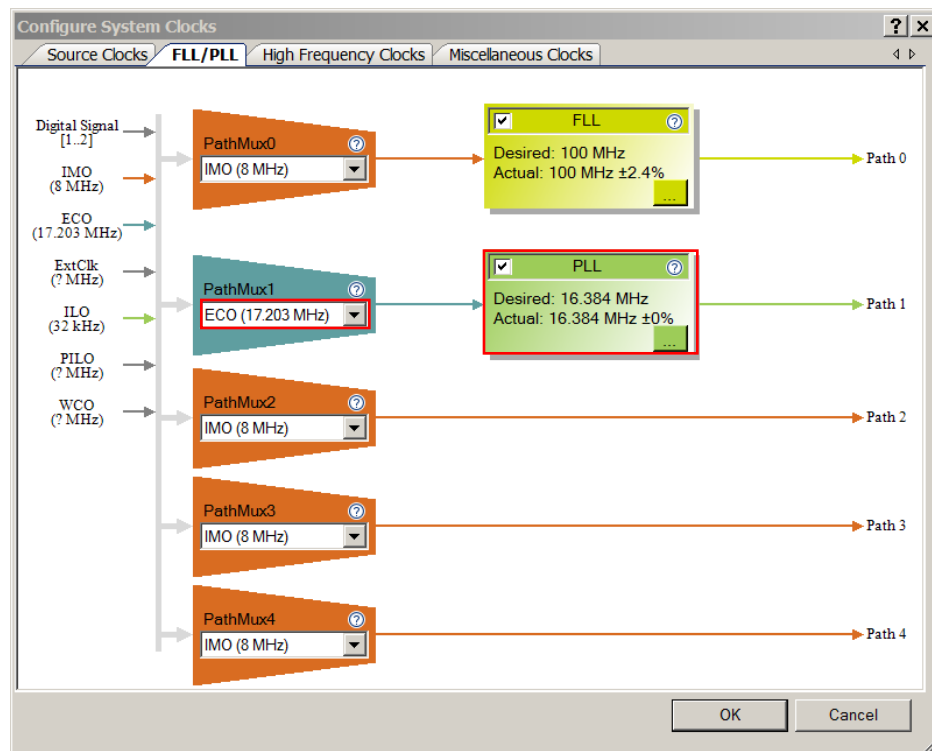
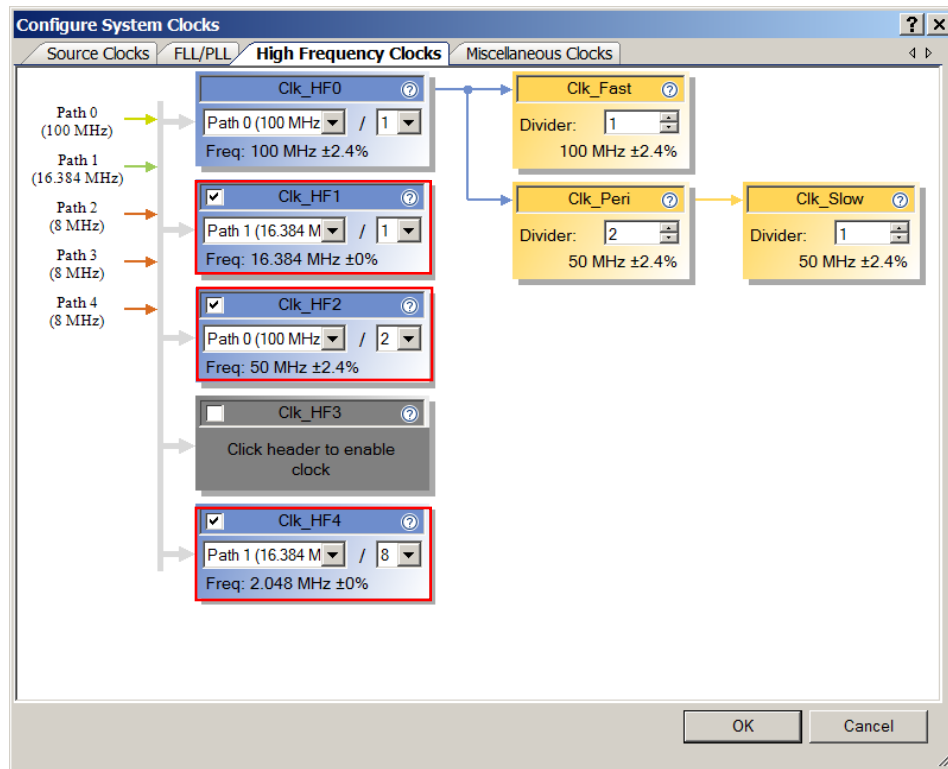


Figure 16. High Frequency Clocks Configuration



Reusing This Example

This example is designed for the CY8CKIT-062-WiFi-BT PSoC 6 Pioneer kit. To port the design to a different PSoC 6 MCU device and/or kit, change the target device using Device Selector and update the pin assignments in the Design Wide Resources Pins settings as needed.

In some cases, a resource used by a code example (for example, an IP block) is not supported on another device. In such cases, 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.

This example also uses an external 17.2032 ECO placed on CY8CKIT-062-WiFi-BT PSoC 6 Pioneer kit. If you are using another kit that does not contain such ECO, edit the clock configuration to source the PLL from another clock.

Related Documents

Application Notes	
AN210781 – Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity	Describes the PSoC 6 BLE, and how to build this code example
AN217666 – PSoC 6 MCU Interrupts	Describes how to use interrupts in PSoC 6 MCU
PSoC Creator Component Datasheets	
Inter-IC Sound Bus (I2S) Component	Sends digital audio streaming data to external I ² S devices
Intra-Integrated Circuit (I2C) Component	Supports I ² C slave, master, and master-slave operation configurations.
PDM to PCM Decoder Component	Converts a PDM signal to PCM.
Direct Memory Access (DMA) Component	Transfers data to and from memory and registers.
Capacitive Sensing (CapSense) Component	Scan capacitive buttons, sliders, touch pad, proximity sensors
Serial Memory Interface (SMIF) Component	Interface external serial memories
Device Documentation	
PSoC 6 MCU: PSoC 63 with BLE Datasheet (PRELIMINARY)	
PSoC 6 MCU: PSoC 63 with BLE Architecture Technical Reference Manual	
PSoC 6 MCU: PSoC 63 with BLE Register Technical Reference Manual	
PSoC 6 MCU: PSoC 62 Datasheet	
PSoC 6 MCU: PSoC 62 Register Technical Reference Manual	
Development Kit (DVK) Documentation	
CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit	
CY8CKIT-062-WiFi-BT PSoC 6 Pioneer Kit	

Document History

Document Title: CE222221 – PSoC 6 MCU Voice Recorder

Document Number: 002-22221

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	6001270	RLOS	01/05/2018	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 Forums](#) | [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.