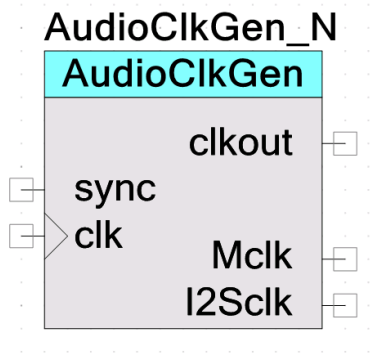
AudioClkGen
Version 0.83

Features



- Used with the system PLL and an input reference clock to generate audio sample rate clocks
- Fractional clock divider from input reference
- Generates audio master clocks at exact multiple of audio sample rate
- Internal synchronization locks output clock rate to reference
- Dynamically change generated audio clocks though API
- Support for I2S v2.4 component and PSoC Creator 2.1

General Description

The AudioClkGen block is an implementation of a frequency synthesis system that generates output clocks that are exactly synchronized to an input frequency, over the long term. For USB audio where this component is used the clock recovery scheme is used to exactly match the data converter master clock to the USB master clock.

Fractional divider outputs a clock divided down from an input reference and rate locked to a second input reference. Divide ratios are set dynamically through API calls.

Note that the AudioClkGen component is only supported in the CY8C34xxx, CY8C36xxx and CY8C38xxx part families.

When to use an AudioClkGen

AudioClkGen is used in digital audio reference design to generate a master clock at a multiple of the desired audio sample and long term rate locked to the USB host master clock. This functionality facilitates USB v2.0 audio class isochronous audio. This preliminary version of the component only supports audio applications.

1. Fixed Audio Sample rates supported in SetAudioRate() API call
    a. 32000 Hz, 44100 Hz, 48000 Hz and 96000 Hz
2. Fixed input frequency (24 MHz) for audio sample rate control in SetAudioRate() API call

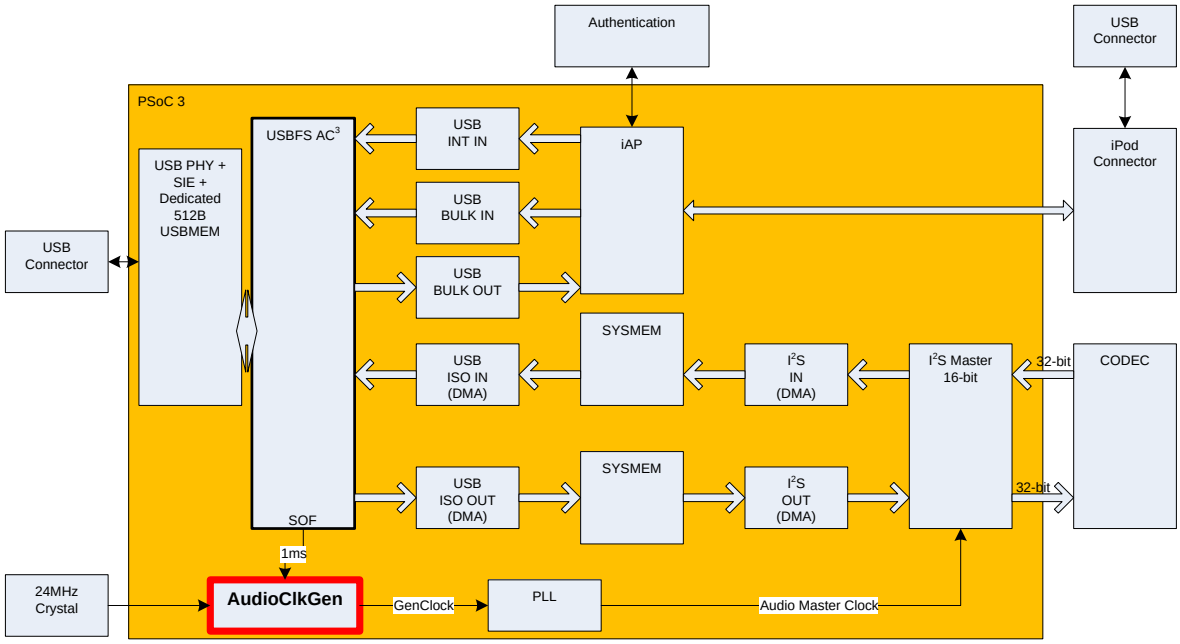


Figure 1 USB Digital Audio Block Diagram with AudioClkGen Component Highlighted.

For a fully functional digital audio solution additional components from the USB digital audio reference design kit are required:

1. USBFS Custom Audio Class Component.
2. iAP component for iAP interface (iPhone/iPod/iPad) support.

To use the AudioClkGen for isochronous USB digital audio with USB synchronization the sync input must be connected to the USBFS start of frame (SOF) output terminal as shown in Figure 2 below. If synchronization is not used the sync_sof pin will be hidden and only connections for the input reference and output clock to the PLL are required.

To connect to an external CODEC the audio master clock from AudioClkGen pin Mclk can be routed to a GPIO pin and the I2Sclk clock should drive the clock of the associated I2S (v2.4) component. These connections are also shown in Figure 2 below.

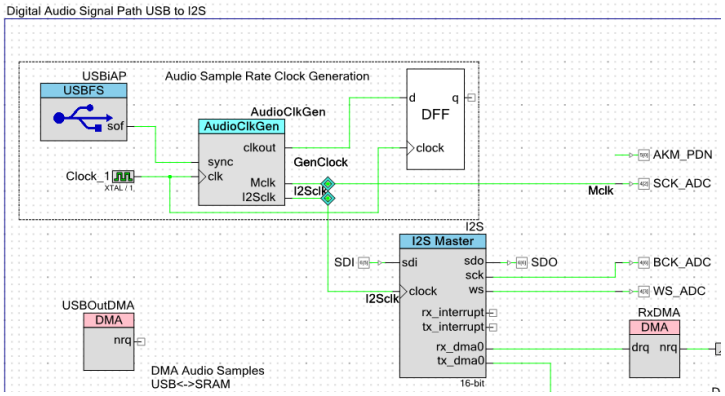


Figure 2 AudioClkGen schematic connections to clocks for I2S and to USBFS synchronize the audio clock with the USB start of frame (SOF) timing.

To route the AudioClkGen clock output through the digial signal interconnect (DSI) to the input of the PLL the output must be connected to another block so Creator can find the named signal. As shown above the simplist connection is to connect clkout to a single D flip-flop. To configure the PLL follow the steps listed in the section “PLL Configuration for Audio Sample Rate Generation” on page 8.

Input/Output Connections

This section describes the various input and output connections for the AudioClkGen. An asterisk (*) in the list of I/O’s states that the I/O may be hidden on the symbol under the conditions listed in the description of that I/O.

clk – Input

Reference clock input that is the input to the fractional divider. For digital audio this clock must be 24 MHz.

sync* – Input

Second reference clock input used for rate synchronization if this options is enabled. The output clock is rate locked to the frequency of this input synchronization signal. The sync input must be a lower frequency clock then clk input.

clkout – Output

Generated divider clock output is divided version of the input clock synchronized to the sync input. This clock is routed into the PLL through the digital signal interconnect (DSI) routes.

Mclk*– Output

Audio master clock at 256 x Fs where Fs is the current audio sample rate. This clock can use used to drive off chip CODECs or other digital blocks on PSoC 3.

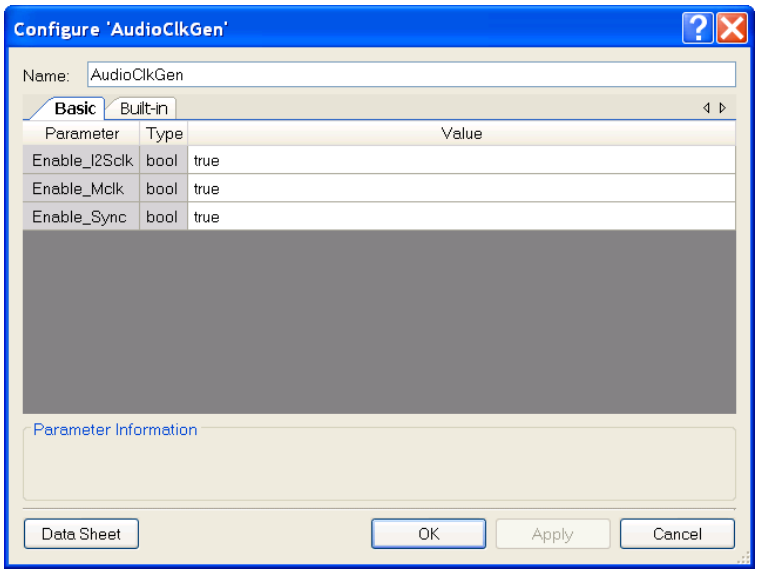
I2Sclk*– Output

I2S reference clock synchronized to the audio master clock. This clock is 64 x Fs where Fs is the current audio sample rate. This clock is synchronous with the Mclk and can use used to drive an I2S data interface.

Component Parameters

Drag a AudioClkGen onto your design and double-click it to open the Configure dialog. The preliminary version of the component only has a user configuration option to enable or disable clock synchronization. Select true for Enable_Sync to synchronize the generated audio clocks to a secondary reference source and false to disable this synchronization.

Figure 3 Configure AudioClkGen Dialog



Placement

The AudioClkGen component uses UDB resources and the DSI connection between the UDBs and system PLL. No specific placement is required.

Resources

The AudioClkGen uses the following device resources:

	Digital Blocks					API Memory (Bytes) (Opt 5, size)	
Resolution	Datapaths	Macro cells	Status Registers	Control Registers	Counter7	Flash	RAM
Audio Rate Support	6	17	0	0	0	2713	34

Application Programming Interface

Application Programming Interface (API) routines allow you to configure the component using software. The following table lists and describes the interface to each function. The subsequent sections cover each function in more detail.

By default, PSoC Creator assigns the instance name "AudioClkGen_1" to the first instance of a component in a given design. You can rename it to any unique value that follows the syntactic rules for identifiers. The instance name becomes the prefix of every global function name, variable, and constant symbol. For readability, the instance name used in the following table is "AudioClkGen".

Function	Description
`AudioClkGen_Start`	Initialize the component
`AudioClkGen_Stop`	Stop the component.
`AudioClkGen_SetAudioRate`	Set divider to support desired audio sample rate.

void AudioClkGen_Start()

Description: Initialize AudioClkGen Component. Initializes internal register values to support audio sample rates.

Parameters: None.

Return Value: void.

Side Effects: Sets internal values for prescaler divisor (low and high period), system scaling factor (N) and count value (C0).

void AudioClkGen_Stop()

Description: Stop AudioClkGen Component.

Parameters: None.

Return Value: void.

Side Effects: None

void AudioClkGen_SetAudioRate(uint8 rate)

Description: Set the divider to generate an output clock to support the desired audio sample rate. Requires a 24 MHz input to the clk terminal.

Parameters: uint8 rate.

Rate Setting	Notes
AudioClkGen_RATE_8KHZ	Generate an output clock to support 8 kHz audio sample rates.
AudioClkGen_RATE_11KHZ	Generate an output clock to support 11.025 kHz audio sample rates.
AudioClkGen_RATE_16KHZ	Generate an output clock to support 16 kHz audio sample rates.
AudioClkGen_RATE_22KHZ	Generate an output clock to support 22.05 kHz audio sample rates.
AudioClkGen_RATE_32KHZ	Generate an output clock to support 32 kHz audio sample rates.
AudioClkGen_RATE_44KHZ	Generate an output clock to support 44.1 kHz audio sample rates.
AudioClkGen_RATE_48KHZ	Generate an output clock to support 48 kHz audio sample rates.
AudioClkGen_RATE_96KHZ	Generate an output clock to support 96 kHz audio sample rates.

Return Value: void.

Side Effects: Changes the divider ratio to generate different output reference frequencies depending on the desired audio sample rate. Changes the output reference clock frequency and the generated

Mclk and I2Sclk signals derived from the PLL. This call also changes settings in the system PLL.

Sample Firmware Source Code

The following is a C language example demonstrating the basic functionality of the AudioClkGen component. This example assumes the component has been placed in a design with the name "AudioClkGen."

Note If you rename your component you must also edit the example code as appropriate to match the component name you specify.

```
#include <device.h>
#include "AudioClkGen.h"
void main()
{
   /* Set up the clocking */
   AudioClkGen_Start();

   /* Set the desired audio sample rate */
   rate = AudioCLkGen_RATE_48KHZ;
   AudioClkGen_SetAudioRate(rate);
}
```

PLL Configuration for Audio Sample Rate Generation

For digital audio support configuration of the system PLL is required. This is done through the CY design wide resources. Double-click on the <project>.cydwr file in the workspace explorer and select the Clocks tab. From the Clocks tab double click the PLL_OUT system clock to configure the system clocks to setup the PLL and start the CPU on the IMO.

In addition API calls to the AudioClkGen component can change the system PLL configuration and temporarily disrupt PLL operation and will impact any other blocks in the system running from PLL output clocks.

For digital audio configurations settings are required to enable the AudioClkGen component: Routing the DSI signal from the clkout of the AudioClkGen component to the PLL input and setting the system MASTER_CLK to run off the IMO at startup. These Route DSI signal from AudioClkGen to PLL input:

1) Name the signal at the clkout pin of the AudioClkGen component.
    a. Place a D flip-flop on the schematic and add a wire to connect the clkout pin to the D input. Add a clock for this DFF. (See Figure 2).
    b. Name the wire by right clicking on it and selecting “Edit Name and Width”. Click the checkbox for “Specify Full Name” and type in a signal name.

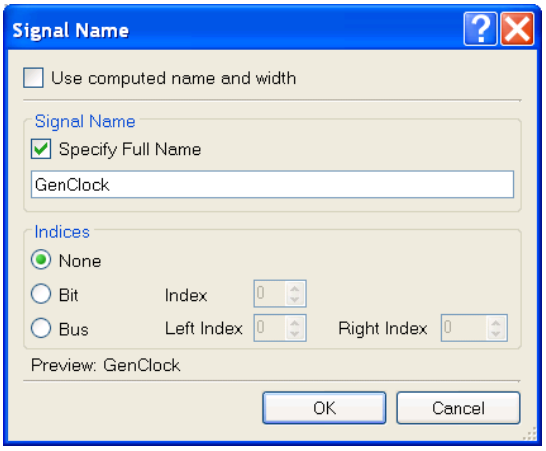


Figure 4 name clkout signal in order to route it through the DSI to the PLL input.

2) Route this signal to the PLL input.

   a. In the design wide resource (<project>.cydwr) Clocks tab double click on the PLL_OUT clock to bring up the "Configure System Clocks" dialog.

   b. Check the box for "Digital Signal" and click on the […] to select an input signal.

   c. Select the signal name connected to the AudioClkGen clkout from the previous step. Set the signal frequency to be 1.024 MHz. Click OK.

   d. Check the box for "PLL" and set the Input pull-down to "IMO". This is for startup, it will be changed to "Digital Signal" later by the SetAudioRate

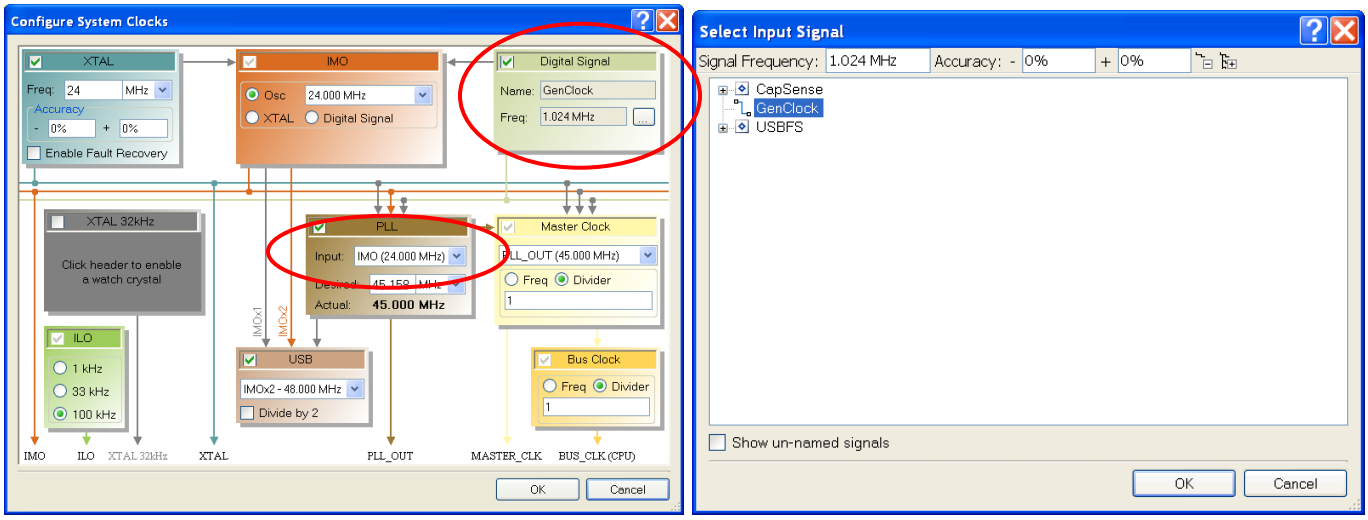


Figure 5 Setup the DSI to route to the PLL input.

Setup the system clock to run from the IMO at startup.

3) Check the "Master Clock" box and select IMO, XTAL or PLL from the pulldown.

   a. Setup any frequency or divider desired.

b. If PLL is selected, the PLL must be configured to startup with the IMO, not the digital signal. Do not use the PLL with “Digital Signal” option since this will not be properly configured at startup. Configure the Digital Signal as described above and the PLL can be switched to use this input after the AudioClkGen component has been initialized.

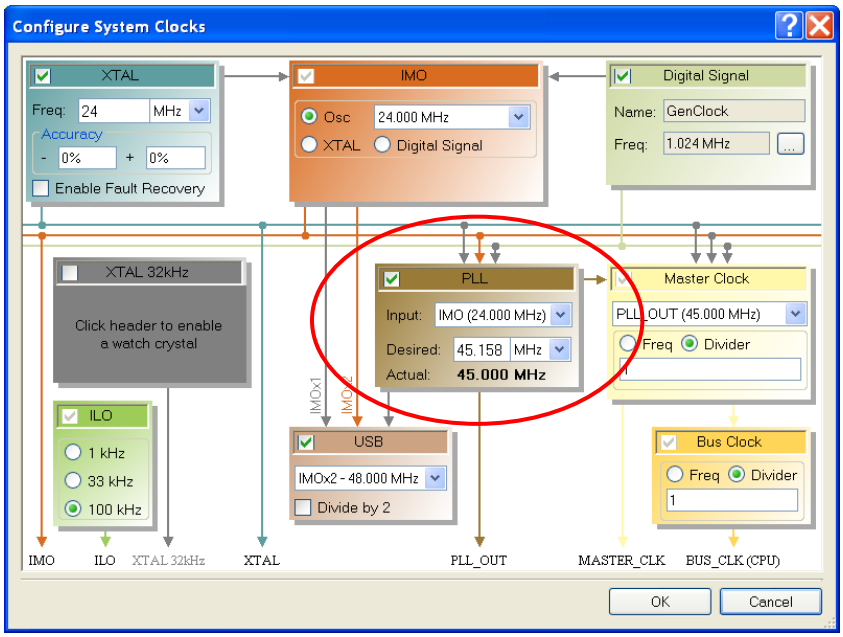


Figure 6 Creator “Configure System Clocks” window from Clocks tab of CY design wide resources.

Functional Description

The fractional divider uses an N/N+1 divider to generate a reference clock frequency at the desired frequency with the divider/prescaler controlled by a sigma-delta modulator. Simultaneously a second process is independently sampling the timing of the USB signal and using this signal to update the sigma-delta modulator counts to exactly synchronize the generated reference frequency to the USB reference. The crystal reference clock is generated with the internal PSoC 3 crystal oscillator, the sigma-delta fractional divider and clock synchronization is implemented in the configurable UDB resources.

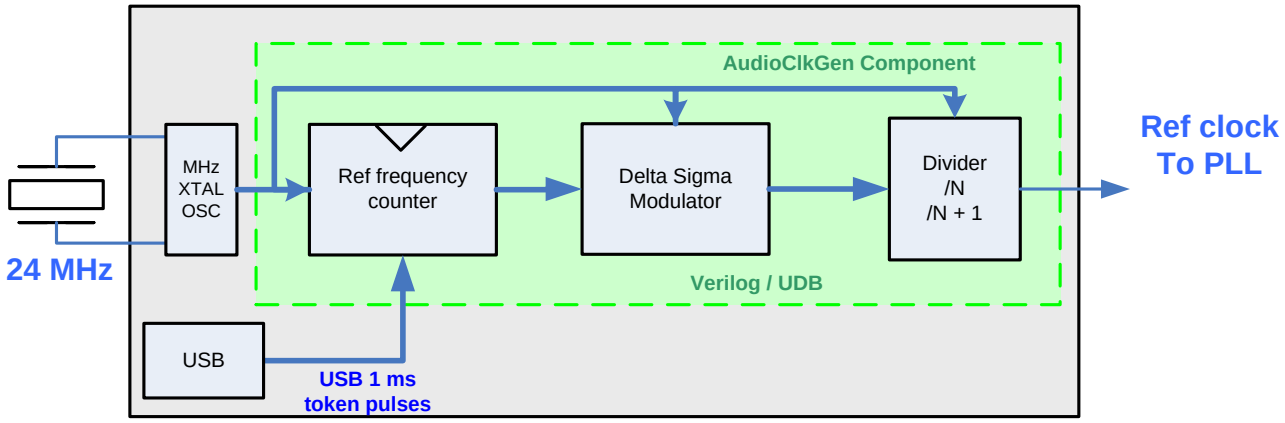


Figure 7 AudioClkGen block diagram shows operation of the fractional divider and synchronization with USB timing signal.

The reference frequency generated by the AudioClkGen component is multiplied up in the PSoC 3 system PLL to generate a master clock that is exactly synchronized to the USB host timing. This clock is used to deliver an audio master clock and drive the I^2S interface to the audio codec (Figure 8).

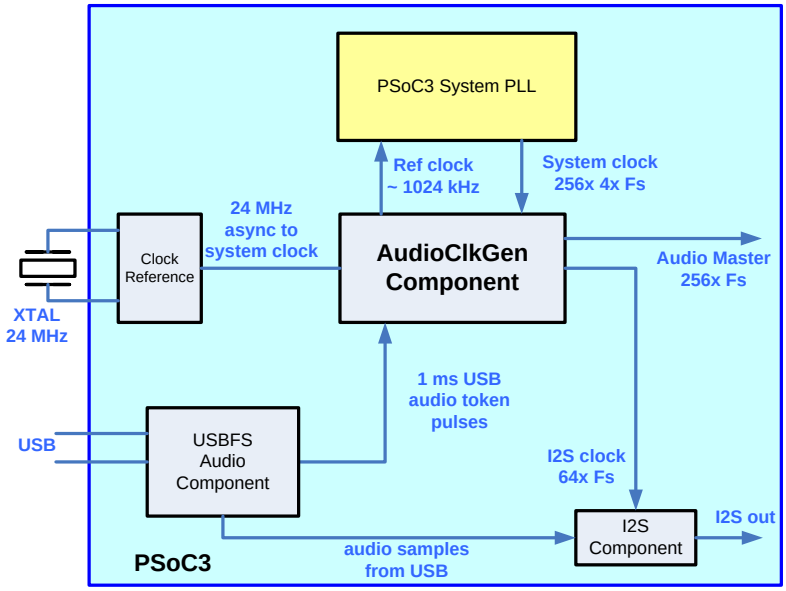


Figure 8 Audio clocking system uses the AudioClkGen component with the PSoC 3 PLL to generate an exactly synchronized audio sample clock to drive the I2S component.

DC and AC Electrical Characteristics

The following values are indicative of expected performance and based on initial characterization data.

AudioClkGen AC Specifications

Parameter	Description	Conditions	Min	Typ	Max	Units
	Operating frequency		12	24	50	MHz

Component Changes

This section lists the major changes in the component from the previous version.

Version	Description of Changes
0.1	Initial version with support limited to USB digital audio reference design.
0.80	Updated version to support I2S v2_0 and Creator Beta5.
0.81	Minor update
0.82	SetAudioRate return type changed
0.83	Added support for 96kHz sampling frequency

© Cypress Semiconductor Corporation, 2013. 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® Creator™, Programmable System-on-Chip™, and PSoC Express™ are trademarks and PSoC® is a registered trademark 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.