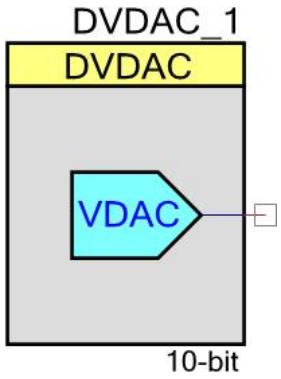
Dithered Voltage Digital to Analog Converter (DVDAC) 2.0



Features

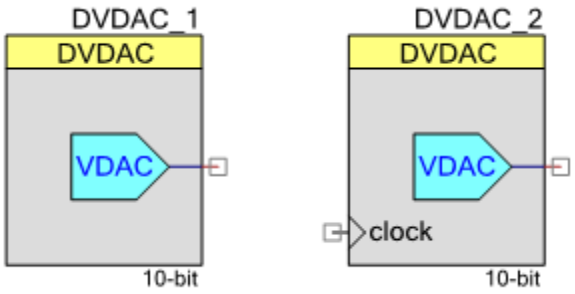
- Two voltage ranges, 1 and 4 volts
- Adjustable 9, 10, 11, or 12 bit resolution
- Dithered using DMA for zero CPU overhead
- Uses a single DAC block

General Description

The Dithered Voltage Digital to Analog Converter (DVDAC) component has a selectable resolution between 9 and 12 bits. Dithering is used to increase the resolution of its underlying 8-bit VDAC8. Only a small output capacitor is required to suppress the noise generated by dithering.

Input/Output Connections

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



Vout – Analog

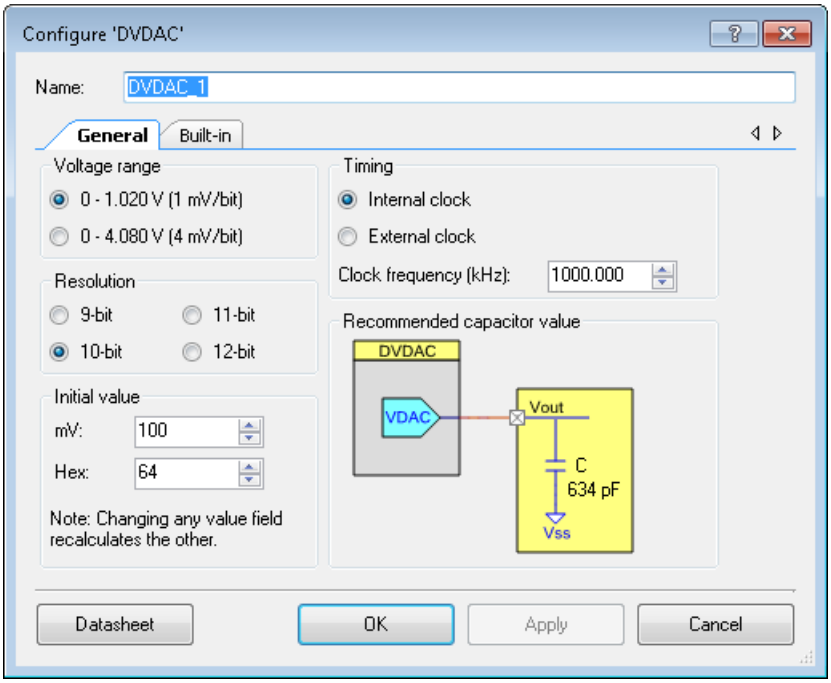
The Vout terminal is the connection to the DAC's voltage output. It may be routed to any analog-compatible pin on the PSoC. An external capacitor should be placed between this output and Vssa to filter the noise generated by the voltage dithering.

clock – Input *

External clock terminal. This terminal is present if the **External clock** option is selected. If the **Internal clock** option is selected, the clock is configured automatically within the component and the clock terminal is not shown. The clock input is a clock that is generated outside the component. This clock determines the DAC data register update rate.

Component Parameters

Drag a DVDAC component onto your design and double click it to open the Configure dialog.



The DVDAC component provides the following parameters.

Voltage Range

This parameter allows you to set one of the two voltage ranges as the default value:

Range	Input value range	Output Equation
1 Volt	0 to 1.020 Volts	Vout = (value/2^{bits}) * 1.024 Volts
4 Volts	0 to 4.080 Volts	Vout = (value/2^{bits}) * 4.096 Volts

Note The term “bits” is the number of bits of resolution for the DAC. The term “value” is the integer value that the DAC is currently set to.

Resolution

This parameter allows you to select one of four resolutions (9, 10, 11, or 12). Refer to the table under **Value** for the connection between **Resolution** and **Value** parameters.

Value

The **mV** field represents the initial value of the DVDAC output voltage in millivolts. The **Hex** field represents the DVDAC input data value in hexadecimal number format. The DVDAC_SetValue() function can override the default value during runtime.

Resolution	Value	Step Size, mV (1 volt range)	Step Size, mV (4 volt range)
9	0x0 to 0x1FE	2	8
10	0x0 to 0x3FC	1	4
11	0x0 to 0x7F8	0.5	2
12	0x0 to 0xFF0	0.25	1

Timing

The **Internal clock** and **External clock** options allow you to choose whether the component will be clocked by one of the following:

- Internal clock: source internal to the DVDAC component
- External clock: source external to the component

When the external clock is enabled, a clock input terminal is displayed on the DVDAC symbol.

The **Clock frequency (kHz)** field represents how fast the output voltage is updated with a dithered voltage value. At each clock a DMA request is made which results in the VDAC being written with the next value. The maximum frequency is limited by the 8-bit VDAC block update rate and is 1 MHz for the 1 volt range and 250 KHz for the 4 volt range. This frequency must be larger than BUS_CLK divided by 12 to ensure that the DMA request will be able to complete the transaction before the next request. If DMA is being used for other functions in the system, a higher margin between BUS_CLK and the DVDAC clock should be used. When using an internal clock the default frequency chosen will be the maximum supported for the voltage range. That value can be edited to a lower frequency.

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 “DVDAC_1” to the first instance of a component in a given design. It can be renamed 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 “DVDAC.”

Function	Description
DVDAC_Start()	Initializes the DVDAC with default customizer values.
DVDAC_Stop()	Disables the DVDAC and sets it to the lowest power state.
DVDAC_SetValue()	Sets the DVDACs output.
DVDAC_Sleep()	Stops and saves the user configuration.
DVDAC_WakeUp()	Restores and enables the user configuration.
DVDAC_Init()	Initializes or restores the default DVDAC configuration
DVDAC_Enable()	Enables the DVDAC.
DVDAC_SaveConfig()	Saves the nonretention DAC data register value.
DVDAC_RestoreConfig()	Restores the nonretention DAC data register value.

Global Variables

Variable	Description
DVDAC_initVar	Indicates whether the DVDAC has been initialized. The variable is initialized to 0 and set to 1 the first time DVDAC_Start() is called. This allows the component to restart without reinitialization after the first call to the DVDAC_Start() routine. If reinitialization of the component is required, then the DVDAC_Init() function can be called before the DVDAC_Start() or DVDAC_Enable() function.

void DVDAC_Start(void)

Description: Performs all of the required initialization for the component and enables power to the block. The first time the routine is executed, the component is initialized to the configured settings. When called to restart the DVDAC following a DVDAC_Stop() call, the current component parameter settings are retained.

Parameters: None

Return Value: None

Side Effects: If the DVDAC_initVar variable is already set, this function only calls the DVDAC_Enable() function.

void DVDAC_Stop(void)

Description: Stops the component and turns off the analog blocks in the DVDAC.

Parameters: None

Return Value: None

void DVDAC_SetValue(uint16 value)

Description: Sets the DVDACs output. The function populates the SRAM array based on the value and the resolution setting. That array is then transferred to the internal VDAC by DMA.

Parameters: (uint16) value: The maximum value will be dependent on the resolution selected:

- 0 – 1FE, for 9 bits
- 0 – 3FC, for 10 bits
- 0 – 7F8, for 11 bits
- 0 – FF0, for 12 bits

This value includes an integer portion and a fractional portion that varies depending on number of bits of resolution.

- 9-bits: 1 fractional bit
- 10-bits: 2 fractional bits
- 11-bits: 3 fractional bits
- 12-bits: 4 fractional bits

Return Value: None

void DVDAC_Sleep(void)

Description: This is the preferred API to prepare the component for sleep. The DVDAC_Sleep() API saves the current component state. Then it calls the DVDAC_Stop() function and calls DVDAC_SaveConfig() to save the hardware configuration.

Call the DVDAC_Sleep() function before calling the CyPmSleep() or the CyPmHibernate() function. Refer to the PSoC Creator *System Reference Guide* for more information about power management functions.

Parameters: None

Return Value: None

Side Effects: None

void DVDAC_Wakeup(void)

Description: This is the preferred API to restore the component to the state when DVDAC_Sleep() was called. The DVDAC_Wakeup() function calls the DVDAC_RestoreConfig() function to restore the configuration. If the component was enabled before the DVDAC_Sleep() function was called, the DVDAC_Wakeup() function will also re-enable the component.

Parameters: None

Return Value: None

Side Effects: Calling the DVDAC_Wakeup() function without first calling the DVDAC_Sleep() or DVDAC_SaveConfig() function may produce unexpected behavior.

void DVDAC_Init(void)

Description: Initializes or restores the component according to the customizer Configure dialog settings. It is not necessary to call DVDAC_Init() because the DVDAC_Start() API calls this function and is the preferred method to begin component operation.

Parameters: None

Return Value: None

void DVDAC_Enable(void)

Description: Activates the hardware and begins component operation. It is not necessary to call DVDAC_Enable() because the DVDAC_Start() API calls this function, which is the preferred method to begin component operation.

Parameters: None

Return Value: None

void DVDAC_SaveConfig(void)

Description: This function saves the component configuration and non-retention registers. This function is called by the DVDAC_Sleep() function.

Parameters: None

Return Value: None

void DVDAC_RestoreConfig(void)

Description: This function restores the component configuration and non-retention registers. This function is called by the DVDAC_Wakeup() function.

Parameters: None

Return Value: None

MISRA Compliance

This section describes the MISRA-C:2004 compliance and deviations for the component. There are two types of deviations defined:

- project deviations – deviations that are applicable for all PSoC Creator components
- specific deviations – deviations that are applicable only for this component

This section provides information on component-specific deviations. Project deviations are described in the MISRA Compliance section of the *System Reference Guide* along with information on the MISRA compliance verification environment.

The DVDAC component does not have any specific deviations.

This component has the following embedded components: Clock, VDAC8, and DMA. Refer to the corresponding components datasheet for information on their MISRA compliance and specific deviations.

Sample Firmware Source Code

PSoC Creator provides many example projects that include schematics and example code in the Find Example Project dialog. For component-specific examples, open the dialog from the Component Catalog or an instance of the component in a schematic. For general examples, open the dialog from the Start Page or **File** menu. As needed, use the **Filter Options** in the dialog to narrow the list of projects available to select.

Refer to the “Find Example Project” topic in the PSoC Creator Help for more information.

Functional Description

The theory of this component operation is that if you quickly write two or more different values to a DAC and filter the output, then the output is the average of the values written to the DAC. This assumes that values written to the DAC are periodic.

Natively, the 8-bit VDAC in the 1-volt range provides a resolution of 4 mV (1.024 V / 256 = 4 mV). If you want an output of 500 mV you can simply write 125 to the DAC. (125 * 4 mV = 500 mV).

However, if you require 501 mV, you have to settle for 500 or 504 mV. By dithering the output at a relatively high speed, the 501 mV output can be generated by averaging multiple output values. In this case a succession of four values can be periodically written to the VDAC.

In the 1-volt range, a value of 125 written to the VDAC produces 500 mV. A value of 126 produces an output voltage of 504 mV. If you average the numbers 500, 500, 500, and 504, you get 501. The following table shows an example of how the output is dithered. The same pattern may be used between any two 8-bit steps to increase the resolution.

Sample	Array1	Array2	Array3	Array4
1	125	125	125	125
2	125	125	125	126
3	125	125	126	126
4	125	126	126	126
Average	125.00	125.25	125.50	125.75
Average Voltage (mV)	500	501	502	503

Dithered VDAC Limitations

The dithered DAC cannot generate the full 2^{bits} unique output values. The last N codes, where $N = 2^{(bits - 8)}$, all generate the same output voltage. This is due to the fact that dithering requires two adjacent 8-bit DAC values to generate an average output signal.

Once the internal 8-bit DAC's output is 255 (0xFF), there is no adjacent higher value. When the voltage DAC is configured for the 1 volt range and the highest output value is written to the VDAC, the output is 1.024 * (255 / 256) = 1.020 volts. This is the highest voltage the 8-bit VDAC can generate.

Using the same equation for a true 10-bit VDAC, results in a slightly higher output, 1.024 * (1023 / 1024) = 1.023 volts. For the dithered VDAC we are using a single 8-bit VDAC to simulate a 10-bit VDAC, so the maximum voltage is still that of the 8-bit VDAC. This means that any VDAC value higher than that of 8-bit VDAC is invalid. In the case of a 10-bit dithered VDAC, the highest valid code is 1020, (1.024 * (1020 / 1024) = 1.020 volts.

The following table specifies the code limit for each resolution.

Resolution (bits)	Valid Range	Invalid Codes	Invalid Range
9	0 – 510	1	511
10	0 – 1020	3	1021 - 1023
11	0 – 2040	7	2041 - 2047
12	0 - 4080	15	4081 - 4095

Another more obvious limitation is the noise generated by the process of dithering. Since the output is the average of two adjacent values, the noise generated by dithering is small. In this case, the noise is 4 mV (1.024 / 256) for the 1-volt range and 16 mV (4.096 / 256) for the 4-volt range. The actual dither frequency varies with the resolution of the DAC. If a 1 MHz dither clock is used for the DMA and the period is set to 4 (10-bits) the actual dither frequency is about 250 kHz (1 MHz / 4).

Capacitor Value Calculation

A filter can be added to reduce the dither noise to an acceptable level. The filter is external to the device so any type of active or passive filter required to reduce the dithered output noise can be used.

To keep the external parts count low, a first order passive filter may be sufficient. A first order filter is simply a resistor and capacitor. Since the output resistance of the DAC is known, 4 kΩ for the 1-volt range and 16 kΩ for the 4-volt range, the resistor is included for free. This means that all that is needed is to add a capacitor on the output.

To calculate the capacitor value we first need to know just how much attenuation is required, and then determine the filter cutoff frequency. For each bit over 8-bits of resolution, the output needs to be attenuated by about 6 dB to attenuate the noise caused by the dither frequency. If we are making a 10-bit DAC, the dithered output would need to be attenuated by 12 dB, (6 db * (10 bits – 8 bits)). For an 11-bit DAC the attenuation will need to be 18 dB and so on.

The filter's cutoff frequency is relative to the dither frequency. The VDAC8 specification states that the maximum clock rate is 1 MHz for the 1-volt range and 256 kHz for the 4 volt range, but this output is divided by 2^{B-8} where "B" is bits of resolution. For example, if we want 10 bits of resolution in the 1 volt range, we divide the 1 MHz sample clock by 4 ($2^{(10-8)}$) or 1 MHz / 4 = 250 kHz.

The following table shows the attenuation required and dither frequency for each resolution and voltage range.

Resolution	9	10	11	12	Bits
Attenuation	6	12	18	24	dB
1-volt dither frequency	500	250	125	62.5	kHz
4-volt dither frequency	125	62.5	31.3	15.6	kHz

Using the following equation, we can find the filter cutoff frequency.

Equation 1. Filter Attenuation

$$Atten = 20 * \log\left(\frac{F_{dith}}{F_c}\right)$$

Where:

Atten is the amount of attenuation required for a given resolution. F_{dith} is the dither frequency and F_c is the filter cutoff frequency.

Solving for F_c:

Equation 2. Cutoff Frequency

$$F_c = \frac{F_{dith}}{10^{\frac{Atten}{20}}}$$

Now that we know the cutoff frequency, we can calculate the filter capacitor value.

Equation 3. RC Filter Cutoff Frequency

$$F_c = \frac{1}{2\pi RC}$$

Solving for C:

Equation 4. Required Capacitor

$$C = \frac{1}{2\pi RF_c}$$

Settling Time Calculation

The last thing to be concerned about is the DAC's settling time at the given resolution.

Equation 5. RC Filter Voltage Settling

$$V_S = V_{in} e^{\frac{-t}{RC}}$$

Where V_s is the settled voltage and V_{in} is the smallest step size of the 8-bit VDAC. If we make V_{in} one unit, then V_s is the faction of the smallest step that we need to settle to for the output to be accurate. Ideally, we want the output to be within one half the smallest step of the VDAC. The step size of the DAC in terms of the initial VDAC can be expressed as follows.

Equation 6. Half Step Voltage

$$V_s = 0.5 * \frac{1}{2^{B-8}}$$

Where B is the bits of resolution required. The “0.5” multiplier is because we want the error to be one half the step size. If we combine these two equations, we get;

Equation 7. Combined Settling Equation

$$0.5 * \frac{1}{2^{B-8}} = V_{in} e^{\frac{-t}{RC}}$$

Solving for time (settling time):

Equation 8. Voltage Settling Time

$$t = -\ln\left(\frac{0.5}{2^{B-8}}\right) * R * C$$

Resources

The DVDAC component uses the following device resources:

- VIDAC block,
- DMA channel,
- Digital Clock Divider (for Internal clock option).

API Memory Usage

The component memory usage varies significantly, depending on the compiler, device, number of APIs used and component configuration. The following table provides the memory usage for all APIs available in the given component configuration.

The measurements have been done with the associated compiler configured in Release mode with optimization set for Size. For a specific design the map file generated by the compiler can be analyzed to determine the memory usage.

Configuration	PSoC 3 (Keil_PK51)		PSoC 5LP (GCC)	
	Flash Bytes	SRAM Bytes	Flash Bytes	SRAM Bytes
Default	605	12	704	11

DC and AC Electrical Characteristics

Specifications are valid for –40 °C ≤ T_A ≤ 85 °C and T_J ≤ 100 °C except where noted. Specifications are valid for 1.71 V to 5.5 V, except where noted. Typical values are for T_A = 25 °C.

DC Characteristics

Parameter	Description	Conditions	Min	Typ	Max	Units
	Resolution		9	–	12	bits
INL	Integral nonlinearity	1-V scale 9 - Bits	–	TBD	TBD	LSB
		1-V scale 10 - Bits	–	TBD	TBD	LSB
		1-V scale 11 - Bits	–	TBD	TBD	LSB
		1-V scale 12 - Bits	–	TBD	TBD	LSB
INL	Integral nonlinearity	4-V scale 9 - Bits	–	TBD	TBD	LSB
		4-V scale 10 - Bits	–	TBD	TBD	LSB
		4-V scale 11 - Bits	–	TBD	TBD	LSB
		4-V scale 12 - Bits	–	TBD	TBD	LSB
DNL	Differential nonlinearity	1-V scale 9 - Bits	–	TBD	TBD	LSB
		1-V scale 10 - Bits	–	TBD	TBD	LSB
		1-V scale 11 - Bits	–	TBD	TBD	LSB
		1-V scale 12 - Bits	–	TBD	TBD	LSB
DNL	Differential nonlinearity	4-V scale 9 - Bits	–	TBD	TBD	LSB
		4-V scale 10 - Bits	–	TBD	TBD	LSB
		4-V scale 11 - Bits	–	TBD	TBD	LSB
		4-V scale 12 - Bits	–	TBD	TBD	LSB
R_{OUT}	Output resistance	1-V scale	–	4	–	kΩ
		4-V scale	–	16	–	kΩ
V_{OUT}	Output voltage range, maximum code value	1-V scale	–	1.02	–	V
		4-V scale, V_{DDA} = 5 V	–	4.08[1]	–	V
	Monotonicity		–	–	Yes	–
V_{OS}	Zero-scale error		–	TBD		LSB
Eg	Gain error	1-V scale	–	TBD	TBD	%

[1] For V_{DDA} voltage below 5V, the output complies to specifications for output voltages below (V_{DDA} – 1 V).

Parameter	Description	Conditions	Min	Typ	Max	Units
		4-V scale	–	TBD	TBD	%
TC_Eg	Temperature coefficient, gain error	1-V scale	–	–	TBD	%FSR/°C
		4-V scale	–	–	TBD	%FSR/°C
I_{DD}	Operating current		–	–	TBD	µA

AC Characteristics

Parameter	Description	Conditions	Min	Typ	Max	Units
F_{DAC}	Effective update rate	9 bits (1V Range)	–	–	500	ksps
		9 bits (4V Range)	–	–	125	ksps
		10 bits (1V Range)	–	–	250	ksps
		10 bits (4V Range)	–	–	62	ksps
		11 bits (1V Range)	–	–	125	ksps
		11 bits (4V Range)	–	–	31	ksps
		12 bits (1V Range)	–	–	62	ksps
		12 bits (4V Range)	–	–	15	ksps
TsettleP	Settling time to 0.1%, step 25% to 75%	1-V scale, C_{LOAD} = 15 pF	–	TBD		µs
		4-V scale, C_{LOAD} = 15 pF	–	TBD		µs
TsettleN	Settling time to 0.1%, step 75% to 25%	1-V scale, C_{LOAD} = 15 pF	–	TBD		µs
		4-V scale, C_{LOAD} = 15 pF	–	TBD		µs
Vn1V	Voltage noise	Range = 1 V, fast mode, Vdda = 5 V, 10 kHz	–	TBD	–	nV/sqrtHz

Component Errata

This section lists known problems with the component.

Cypress ID	Component Version	Problem	Workaround
191257	v2.0	This component was modified without a version number change in PSoC Creator 3.0 SP1. For further information, see Knowledge Base Article KBA94159 (www.cypress.com/go/kba94159).	No workaround is necessary. There is no impact to designs.

Component Changes

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

Version	Description of Changes	Reason for Changes / Impact
2.0.b	Minor datasheet edits.	
2.0.a	Edited datasheet to add Component Errata section.	Document that the component was changed, but there is no impact to designs.
2.0	First Cypress-supported version of the component.	

© Cypress Semiconductor Corporation, 2013-2016. 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. 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.