

Associated Part Families: CY8C38xx/CY8C55xx
Software Version: PSoC[®] Creator™
Related Hardware: CY8CKIT-001
Author: Praveen Sekar

Code Example Objective

This code example describes how to generate a sine wave of a specified frequency using the digital-to-analog converter (DAC) and DMA.

Overview

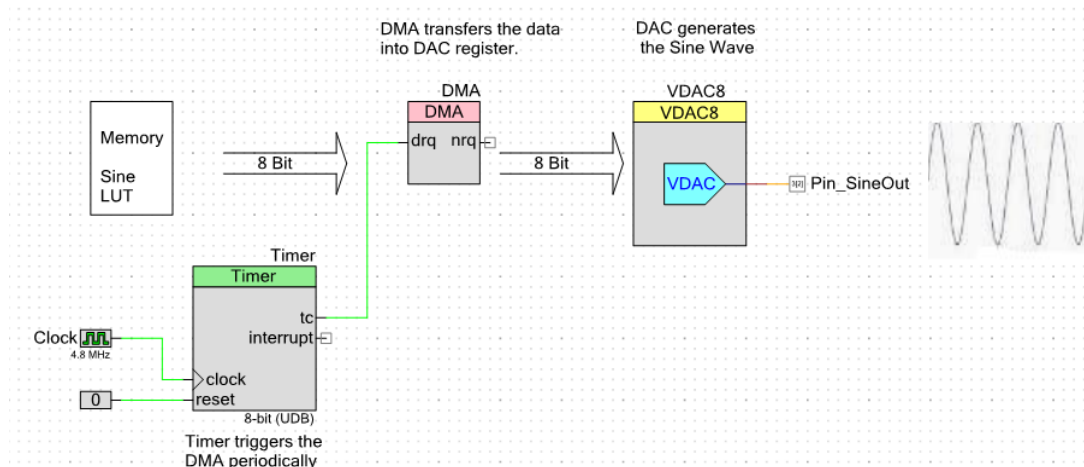
A DAC is updated sequentially with the values in a sine lookup table. If the points are considerably close, output waveform is smooth. The sine output can be made smooth by passing the DAC output through a low pass filter.

Component List

Instance Name	Component Name	Component Category	Comments
VDAC8	Voltage DAC (8-bit)	Analog → DAC	Gives out sine wave output.
Timer	Timer	Digital → Functions	Period is set to '2' to output a terminal count of 400 KHz.
DMA	Direct Memory Access (DMA)	System	drq is enabled and timer's terminal count acts as a trigger.
Pin_SineOut	Analog Pin	Ports and Pins	Configured as analog port with high impedance analog.
Clock	Clock	System	Configured to output 4.8 MHz.

Top Design

The following diagram shows the components and their routing.

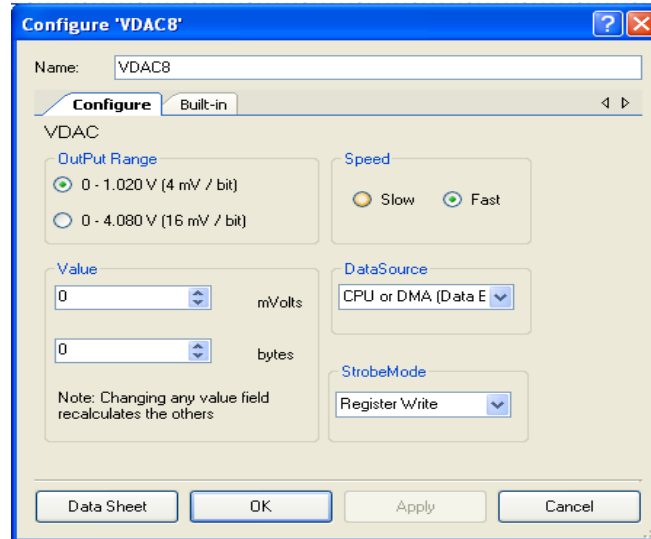


The following figure shows pin placement (as in .cydwr file):

Alias	Name	Pin	Lock
	Pin_SineOut	P3[2]	<input checked="" type="checkbox"/>

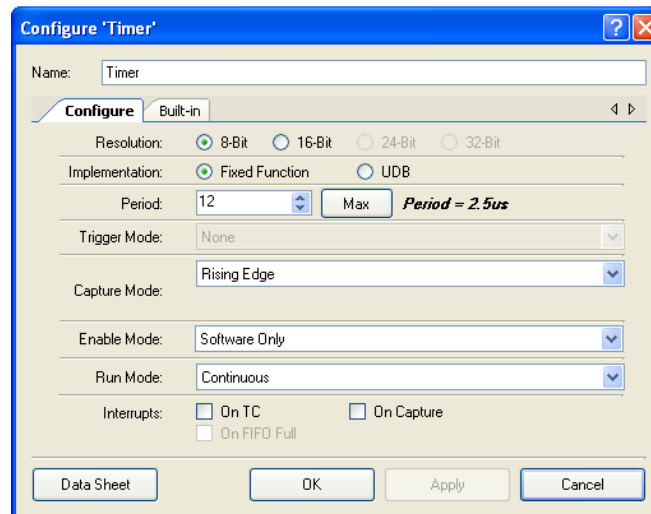
Component Configuration

DAC



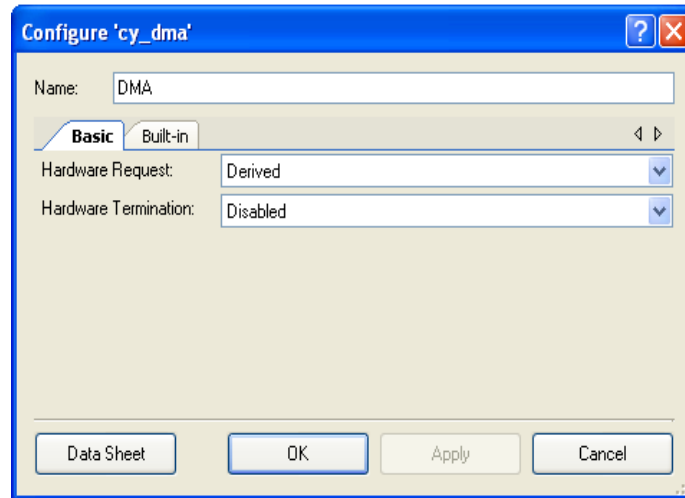
The output range of VDAC8 is set to 0 – 1.020 V with 4 mV / bit. DataSource is chosen as CPU or DMA (Data Bus).

Timer



With a clock frequency of 4.8 MHz, the Timer Period is set to 12 to obtain an update rate of 12 cycles or 2.5 μ s period. When 32 points lookup table is used, period of the sine wave obtained is 2.5 μ s x 32 = 80 μ s (12.5 KHz). The run mode of the Timer is set to Continuous. By varying the Timer Period, the update rate can be varied.

DMA



In DMA configuration, the Hardware Request type is set to Derived. Hardware Termination is disabled which implies that the DMA transfer can be terminated only by CPU request or when the DMA data transfer is complete.

DMA Wizard

The DMA Wizard is an easy to configure GUI that enables the user to make the DMA and Transaction Descriptor (TD) configuration selection. After the DMA and TD configuration is done, the wizard generates the APIs that should be copied into the code example. Before understanding the use of DMA wizard, let us familiarize ourselves with some important terminologies associated with DMA.

PHUB: This is a peripheral hub. It is a high performance bus used to access different peripherals and for data transfer. It has spokes to which peripherals are connected.

Spoke: Spokes are buses that radiate from PHUB. They are connected to one or more peripherals. The spoke data bus width is 16 or 32 bits.

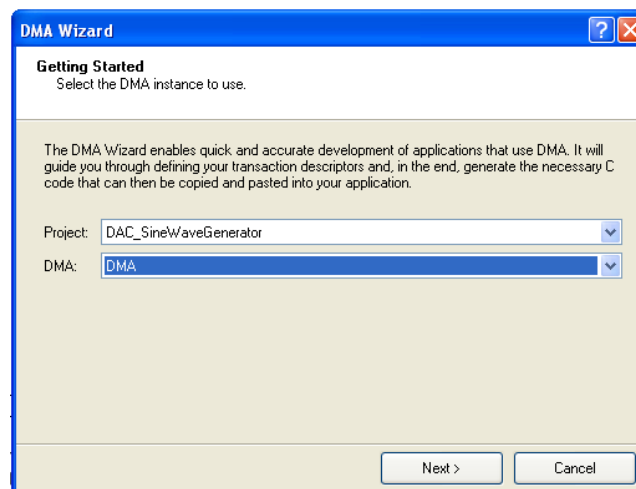
Channel: Channel resides in the DMA controller. Channels use the PHUB to do the data transfer. Channels fetch the transaction descriptors, access the PHUB spoke for the source and destination and performs the data transfer.

Transaction Descriptor (TD): The Transaction Descriptor contains all the information required for data transfer. The information includes source and destination address and the number of bytes to transfer.

Burst Count: This term is used to describe the data transfer done by the DMA. A large chunk of data is split into smaller chunks to avoid hogging the bus. The size of smaller chunks is defined by the burst count parameter.

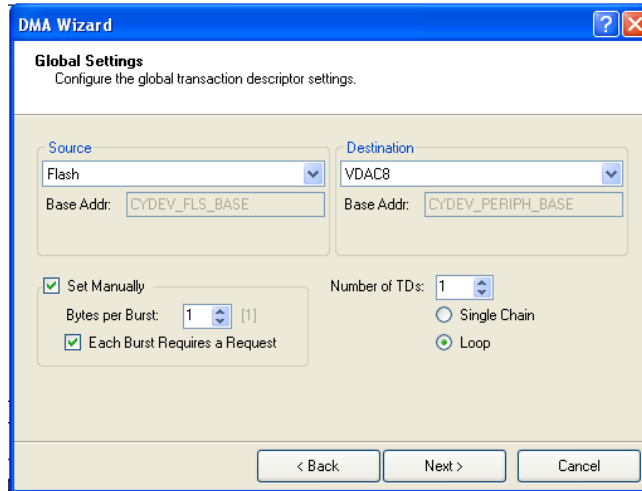
Transfer Count: Transfer count defines the number of bytes to be transferred by a TD. The DMA Controller (DMAC) uses the PHUB to do the data transfer. The PHUB has two masters- the CPU and DMAC.

Now the DMA Wizard can be started by clicking on **Tools → DMA Wizard**



The DMA Wizard can be used to configure each DMA channel and the TDs for each DMA channel separately. The first page in the Wizard shows the code example name and name of the DMA channels used in the code example. The wizard retrieves the name of code example automatically. It also retrieves the DMA Channel names from the Top Design automatically. The figure above shows the first page, **Getting Started**.

When the code example uses more than one DMA channel, all the DMA channel names are displayed in the list. The configuration should be done separately for each DMA channel. To go to the DMA channel setting page, click **Next** in the **Getting Started** page.

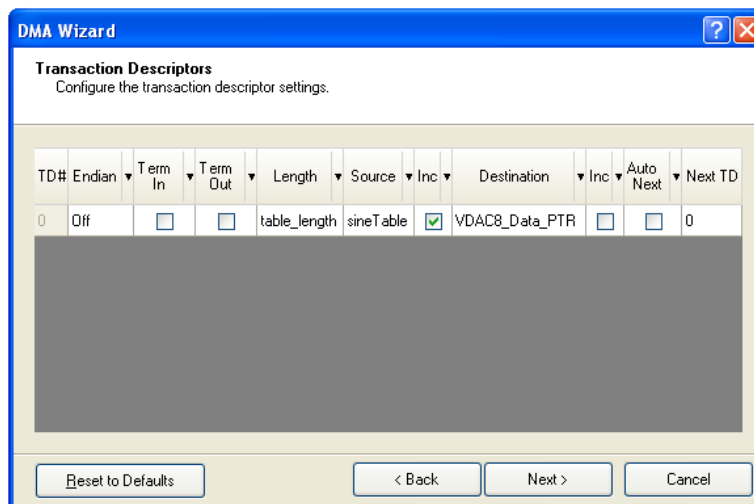


In the Global settings page, the Source is chosen as Flash where the sine lookup table is stored and the Destination is a VDAC8 register.

Bytes per Burst allows you to set the number of bytes to transfer in a single burst. The data transfer is configured for 1 byte per burst. Each burst is transferred at the terminal count of the Timer. Hence, 'Each Burst Requires a Request' is enabled.

The data transfer needs only one TD and the TD is repeated for continuous wave generation. Hence, the 'Loop' option is enabled, which loops back to the first TD.

After configuring the appropriate settings, click **Next** to go to the Transaction Descriptors page where you can configure the TDs for each DMA channel.



TD#: Describes the logical TD number. It is used in conjunction with next TD.

Endian: It enables 2 or 4 byte endian byte swapping.

Term In: Enables the terminating of this TD on a rising edge of the TERMIN signal.

Term Out: Enables the generation of TERMOUT signal when the TD completes.

Length: This indicates the length in bytes for this TD. In this code example, it is the length of the sine look up table.

Source: This indicates the source address for the DMA transfer. Here, the sine look up table is the source.

Inc (Source): Enables incrementing of the source address as the DMA progresses through the specified number of bytes. It is enabled so that the succeeding values in the sine look up table are automatically transferred.

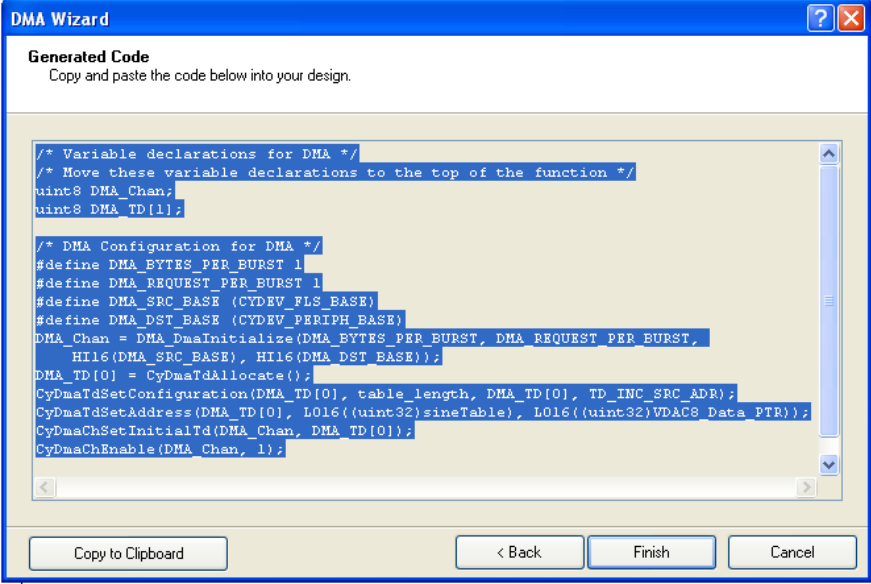
Destination: This indicates the destination address for the DMA transfer. VDAC data pointer is chosen as the destination.

Inc (Destination): Enables incrementing of the destination address as the DMA progresses through the specified number of bytes. In this code example, it is disabled as the destination address remains the same.

Auto Next: Specifies whether or not to automatically execute the next TD after this TD completes without requiring another request. This is unchecked as a TD has to be executed only upon receiving a request (Terminal count of Timer).

Next TD: Specifies the next logical TD in the chain of TDs. It is set to End if this is the last TD, else set to 0.

Once the configuration shown above is done, click **Next** to go to the **Generate Code** page.



```

DMA Wizard
Generated Code
Copy and paste the code below into your design.

/* Variable declarations for DMA */
/* Move these variable declarations to the top of the function */
uint8 DMA_Chan;
uint8 DMA_TD[1];

/* DMA Configuration for DMA */
#define DMA_BYTES_PER_BURST 1
#define DMA_REQUEST_PER_BURST 1
#define DMA_SRC_BASE (CYDEV_FLS_BASE)
#define DMA_DST_BASE (CYDEV_PERIPH_BASE)
DMA_Chan = DMA_DmaInitialize(DMA_BYTES_PER_BURST, DMA_REQUEST_PER_BURST,
    HI16(DMA_SRC_BASE), HI16(DMA_DST_BASE));
DMA_TD[0] = CyDmaTdAllocate();
CyDmaTdSetConfiguration(DMA_TD[0], table_length, DMA_TD[0], TD_INC_SRC_ADR);
CyDmaTdSetAddress(DMA_TD[0], L016((uint32)sineTable), L016((uint32)VDAC8_Data_PTR));
CyDmaChSetInitialTd(DMA_Chan, DMA_TD[0]);
CyDmaChEnable(DMA_Chan, 1);
  
```

After the DMA channels and TD configuration is complete, the wizard generates code for the respective DMA channel. This code includes the configuration for the DMA channel and the TDs. Copy the code and paste it into the *main.c*.

In PSoC 3, the Keil compiler needs the keyword 'code' to store the lookup table to Flash. In PSoC 5, the compilers (GCC, ARM-RVDS, and ARM-MDK) can store the data to Flash with the keyword 'const' itself.

The keyword CYCODE should be used when the user wants to make the variable declaration compatible with PSoC 3 and PSoC 5.

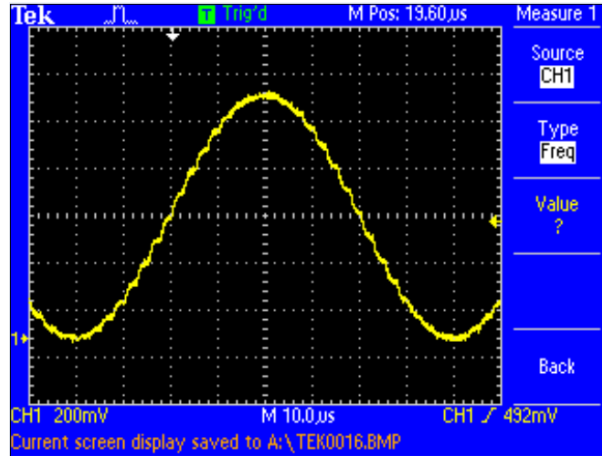
Design Wide Resources

The code example uses default configuration. Refer to *DAC_SineWaveGenerator.cydwr* for the default settings.

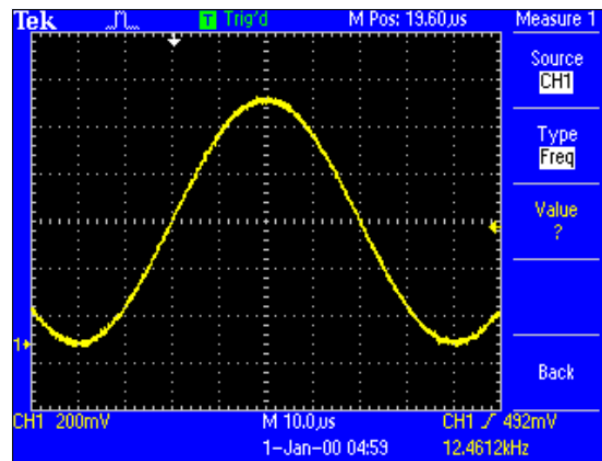
Operation

Sine lookup table consisting of 32 points and 128 points are stored in Flash whose values are updated sequentially to a DAC to obtain a sine wave. The update rate and the number of points in the sine lookup table determine the frequency of the output sine wave. The values from the lookup table are updated into the DAC with a DMA. The DMA is set to update values on a hardware trigger. DMA operates on a single channel and one TD. The TD loops itself so that the sine wave is repeating. The hardware trigger is given from terminal count of a timer. The output frequency of the sine wave generated equals the update rate divided by number of points in the sine lookup table. The code example generates a 12.5 kHz sine wave with 32 points in the sine lookup table with an update rate (timer frequency) of 400 kHz. A low pass filter can be put on the VDAC output to make the sine wave smoother.

- The sine output can be made smoother by using higher number of points in the lookup table. In this code example 128 points lookup table for sine wave is also stored in Flash. The user can set the 'SMOOTH' to non-zero value to use 128 points lookup table by changing its value in #define. The snap shot below shows the sine wave output using 32 and 128 points lookup tables.



Sine wave output using 32 point lookup table



Sine wave output using 128 points lookup table

- The frequency of sine wave generated can be varied by varying the update rate of DMA. The frequency of sine wave can be increased by increasing the update rate of the DMA that in turn is achieved by decreasing the period of Timer used to trigger the DMA. To obtain sine wave of frequency 12.5 KHz using 128 points lookup table, the update rate is set to 1.6 MHz.

Hardware Connections

This code example can be tested on the CY8CKIT-001 development board. The following connection must be made on the board to make the code example work.

- Connect Oscilloscope to Pin P3_2 to observe DAC output.
- Change the value of 'SMOOTH' in #define with a non-zero value and again observe the DAC output at Pin P3_2.

For the remaining basic settings of the development kit, refer to the *CY8CKIT-001 PSoC Development Kit Board Guide*, provided with the kit.

Output

- Use device selector (Code example → Device Selector) window in PSoC Creator to select the appropriate device and Device Revision.
- If you are using PSoC 3 device (for example, [CY8C3866AXI-040](#)) with production revision, then use the following selection.

Part Number	Design Fits on Device	Family	Temperature Grade	CPU Speed (MHz)	Flash (KB)	SRAM (KB)	EEPROM (bytes)	Trace Buffer (KB)	DMA Channels	PLL	LCD Drive (max ratio)	CapSense	ADC	8-bit DAC
CY8C3866AXA-039	?	PSoC3 (8051)	Automotive	67	64	8	2048	4	24	1	x16	✓	1x 20-bit Delta Sigma	4
CY8C3866AXA-040	?	PSoC3 (8051)	Automotive	67	64	8	2048	4	24	1	x16	✓	1x 20-bit Delta Sigma	4
CY8C3866AXA-054	?	PSoC3 (8051)	Automotive	67	64	8	2048	4	24	1	-	✓	1x 20-bit Delta Sigma	4
CY8C3866AXA-055	?	PSoC3 (8051)	Automotive	67	64	8	2048	4	24	1	-	✓	1x 20-bit Delta Sigma	4
CY8C3866AXI-033	?	PSoC3 (8051)	Industrial	67	64	8	2048	4	24	1	-	✓	1x 20-bit Delta Sigma	4
CY8C3866AXI-035	?	PSoC3 (8051)	Industrial	67	64	8	2048	4	24	1	x16	✓	1x 20-bit Delta Sigma	4
CY8C3866AXI-038	?	PSoC3 (8051)	Industrial	67	64	8	2048	4	24	1	x16	✓	1x 20-bit Delta Sigma	4
CY8C3866AXI-039	?	PSoC3 (8051)	Industrial	67	64	8	2048	4	24	1	x16	✓	1x 20-bit Delta Sigma	4
CY8C3866AXI-040	?	PSoC3 (8051)	Industrial	67	64	8	2048	4	24	1	x16	✓	1x 20-bit Delta Sigma	4
CY8C3866AXI-054	?	PSoC3 (8051)	Industrial	67	64	8	2048	4	24	1	-	✓	1x 20-bit Delta Sigma	4
CY8C3866AXI-055	?	PSoC3 (8051)	Industrial	67	64	8	2048	4	24	1	-	✓	1x 20-bit Delta Sigma	4
CY8C3866LTI-020	?	PSoC3 (8051)	Industrial	67	64	8	2048	4	24	1	-	✓	1x 20-bit Delta Sigma	4
CY8C3866LTI-023	?	PSoC3 (8051)	Industrial	67	64	8	2048	4	24	1	-	✓	1x 20-bit Delta Sigma	4

347 of 347 devices found [Clear Filters](#) **Device Revisions:** PRODUCTION
 PRODUCTION
 ES3
 ES2

Start Auto Select

- Similarly, select appropriate device number (for example, CY8C5588AXI-060) to work with PSoC 5 Device family.
Note For engineering samples, device revision is marked on the package as part of the device number. Production silicon will not have ES marking.
- Build the code example and program the device.
- Reset the device by Pressing SW4 (Reset Switch)
- Observe 12.5 kHz Sine waveform on the scope.

Related Application Notes and Code Examples:

- DMA Memory Transfer in PSoC[®] 3 / PSoC 5
- DMA Peripheral Transfer in PSoC[®] 3 / PSoC 5
- AN52705 - Using DMA on PSoC[®] 3 and PSoC 5

Document History

Document Title: CE56171 – PSoC® 3 / PSoC 5 – Sine Wave Generator with DAC

Document Number: 001-56171

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	2765677	PFZ	09/22/09	New application note
*A	2913238	PFZ	04/25/10	Removed warning, updated top design, changed copyright
*B	2944347	DASG	06/17/10	Updated the project and document to work with PSoC 3/PSoC 5 Updated the documents to meet the spec 001-55157 and 001-58307
*C	3178394	PFZ	02/21/2011	Updated the title and replaced 'example project' with 'code example'. The snap shots and explanation are added to describe the use of DMA Wizard. The project is changed to store the sine look-up table in Flash. Both 128 points and 32 points look-up table is included to show smooth sine wave generation. Explanation is added to describe the action to be taken to vary the frequency of the output sine wave.

PSoC is a registered trademark of Cypress Semiconductor Corp. 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
Phone: 408-943-2600
Fax: 408-943-4730
<http://www.cypress.com/>

© Cypress Semiconductor Corporation, 2009-2011. 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.

This 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.