Dual 8-bit Hardware DAC Datasheet DualDAC8HW V 1.0

Copyright © 2009-2014 Cypress Semiconductor Corporation. All Rights Reserved.

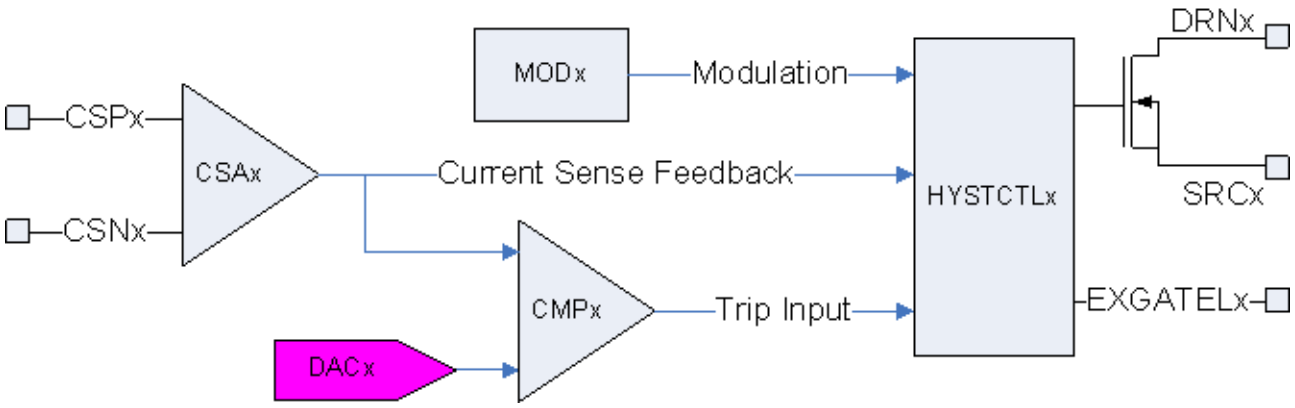
Resources	PSoC® Blocks	API Memory (Bytes)		Pins (per External I/O)
	DAC	Flash	RAM	
CY8CLED0xD, CY8CLED0xG	2	27	0	-

Features and Overview

- Two independent channels with common configuration settings
- 8-bit resolution
- Voltage output
- Monotonic operation
- Low gain errors
- Binary input data format
- Intended to be statically configured references
- Update rate of 10 µs

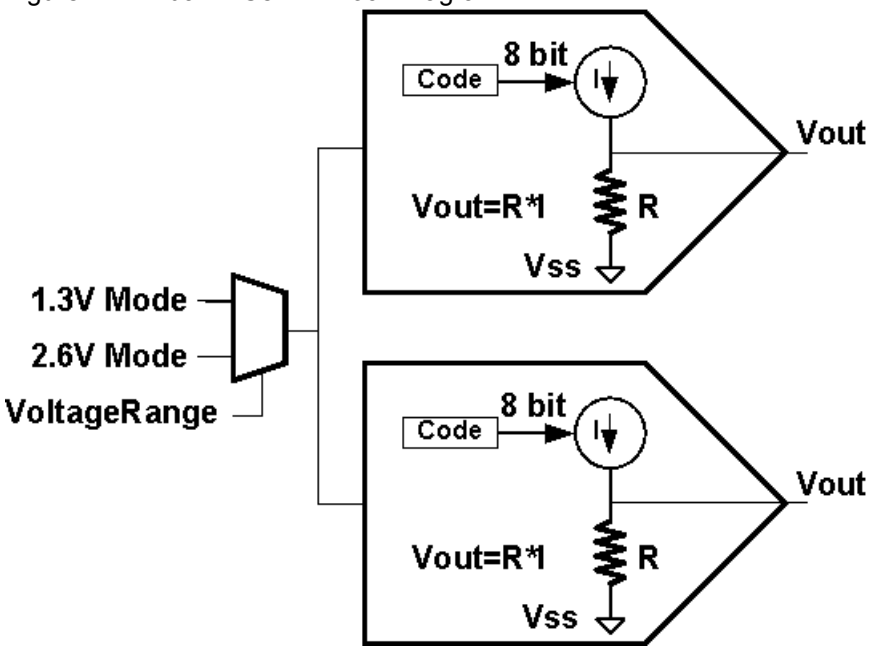
The DualDAC8HW User Module can be used to provide a reference for internal comparators. A typical application for this user module is to provide a reference for comparators in an overcurrent protection or a voltage protection circuit. The following diagram shows connection for one of two DACs. The another "half" of the DualDAC8HW is connected in the same manner.

Figure 1. DualDAC8HW Functional Relation Diagram



The DualDAC8HW User Module internal diagram is represented in the following figure. Each of the "halves" contains its own current source that is driven by the 8-bit code. Both DACs have a common mode control.

Figure 2. DualDAC8HW Block Diagram



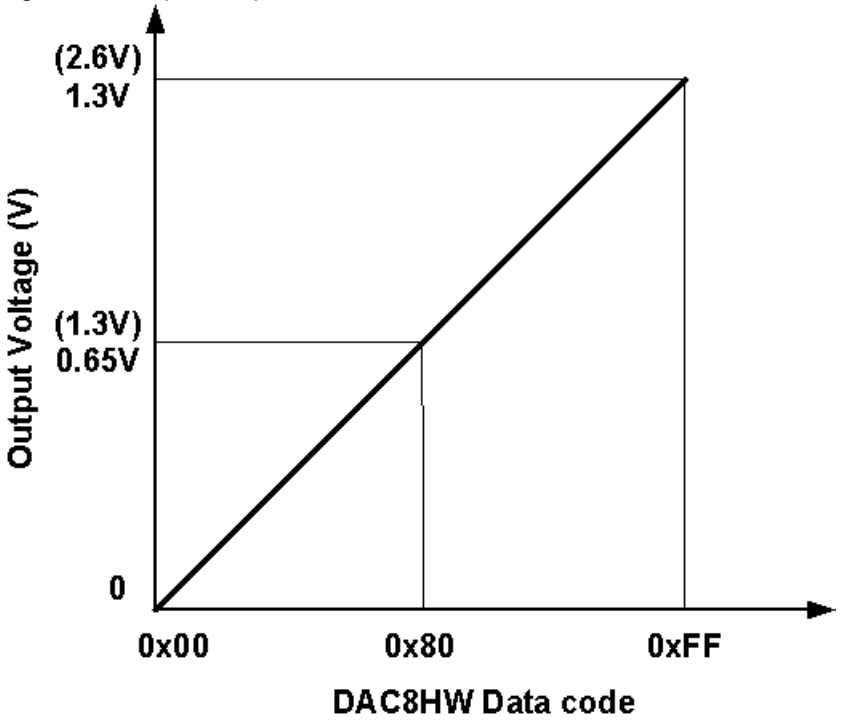
Functional Description

The output of the DualDAC8HW User Module can be routed to internal comparators only. The DAC has two outputs and both of these are enabled or disabled together. However, the data code can be individually updated.

The DualDAC8HW User Module can operate in two modes. The range of the DACs varies from 0 V (00h) to 1.3 V (FFh) when the Voltage Range bit in the VDACx_CR register is one. The range is from 0 V (00h) to 2.6 V (FFh) when the Voltage Range bit is zero. Use the Device Editor or the provided APIs to enable or disable the DACs and select the output voltage range.

Figure 3 illustrates the transfer characteristics of the DACs. The figure shows that DualDAC8HW is a linear monotonic DAC that has a zero offset. The step size is 5 mV for every bit if the 1.3 V mode is selected. The step size is 10 mV for every bit if the 2.6 V mode is selected. As shown, if the 1.3 V range is selected, a data code of 0x80 results in an output voltage of 0.65 V. If the 2.6 V range is selected (shown in parentheses), the same data code results in an output voltage of 1.3 V.

Figure 3. Input-Output Transfer Characteristics of DualDAC8HW



The output from the DACs is based on the 8-bit data code written into the DualDAC8HW_DATA0_REG and DualDAC8HW_DATA1_REG registers respectively. This value can be written using the API only. The default data code is set to zero.

The following equation mathematically explains how the desired voltage is derived from given data code and voltage range (1.3 V or 2.6 V).

Equation 1

$$Voltage = \frac{Code \times Range}{255}$$

DC and AC Electrical Characteristics

See the device characterization data in the DC and AC Electrical Characteristics section of the device datasheet.

Placement

The DualDAC8HW User Module can be placed on any of the DAC8-DAC9, DAC10-DAC11, or DAC12-DAC13 block pairs.

Note The DAC0-DAC7 are used to provide reference voltage for the hysteretic controller only. The properties of DAC0-DAC7 are configured with the HYSTCTRL User Module.

Parameters and Resources

VoltageRange

The Mode parameter allows selecting the output voltage range for both DACs:

Parameter	Description
1.3 V Mode	Sets the voltage range from 0 V to 1.3 V in steps of 5 mV per bit.
2.6 V Mode	Sets the voltage range from 0 V to 2.6 V in steps of 10 mV per bit.

Application Programming Interface

The Application Programming Interface (API) routines are provided as part of the user module to allow the designer to deal with the module at a higher level. This section specifies the interface to each function together with related constants provided by the include files.

Each time a user module is placed, it is assigned an instance name. By default, PSoC Designer assigns the DualDAC8HW_1 to the first instance of this user module in a given project. It can be changed to any unique value that follows the syntactic rules for identifiers. The assigned instance name becomes the prefix of every global function name, variable, and constant symbol. In the following descriptions the instance name has been shortened to DualDAC8HW for simplicity.

Note

** In this, as in all user module APIs, you can alter the values of the A and X register by calling an API function. It is the responsibility of the calling function to preserve the values of A and X before the call if those values are required after the call. This "registers are volatile" policy was selected for efficiency reasons and has been in force since version 1.0 of PSoC Designer. The C compiler automatically takes care of this requirement. Assembly language programmers must ensure their code observes the policy, too. Though some user module API function may leave A and X unchanged, there is no guarantee they may do so in the future.

For Large Memory Model devices, it is also the caller's responsibility to preserve any value in the CUR_PP, IDX_PP, MVR_PP, and MVW_PP registers. Even though some of these registers may not be modified now, there is no guarantee that will remain the case in future releases.

DualDAC8HW_Start

Description:

Starts the DualDAC8HW operation.

C Prototype:

```
void DualDAC8HW_Start(void)
```

Assembler:

```
lcall DualDAC8HW_Start
```

Parameters:

None

Return Value:

None

Side Effects:

See Note ** at the beginning of the API section.

DualDAC8HW_Stop

Description:

Stops the DualDAC8HW operation.

C Prototype:

```
void DualDAC8HW_Stop()
```

Assembler:

```
lcall DualDAC8HW_Stop
```

Parameters:

None

Return Value:

None

Side Effects:

See Note ** at the beginning of the API section.

DualDAC8HW_SetVoltageRange

Description:

Sets the output voltage range.

C Prototype:

```
void  DualDAC8HW_SetVoltageRange(BYTE bVoltageRange)
```

Assembler:

```
mov   A, bVoltageRange
lcall  DualDAC8HW_SetVoltageRange
```

Parameters:

bVoltageRange: indicates the output range of the DualDAC8HW. Symbolic names provided in C and assembly, and their associated values, are given in the following table:

Symbolic Name	Value	Description
DualDAC8HW_1_3 V	0x02	Sets the voltage range from 0 V to 1.3 V
DualDAC8HW_2_6 V	0x00	Sets the voltage range from 0 V to 2.6 V

Return Value:

None

Side Effects:

See Note ** at the beginning of the API section.

DualDAC8HW_Write1

Description:

Write the 8-bit digital code on the first DAC channel. DAC output depends on VoltageRange selection.

C Prototype:

```
void  DualDAC8HW_Write1(BYTE bData)
```

Assembler:

```
mov   A, bData
lcall  DualDAC8HW_Write1
```

Parameters:

bData: Defines the 8-bit data code to be loaded in the DAC. The format of the bData parameter is binary. The following table shows values for the lowest, mid, and the highest reference voltage setting.

Code	1.3V Mode value	2.6V Mode value
00h	0 V	0 V
80h	0.65 V	1.3 V
FFh	1.3 V	2.6 V

Return Value:

None

Side Effects:

See Note ** at the beginning of the API section.

DualDAC8HW_Write2

Description:

Write the 8-bit digital code on the second DAC channel. DAC output depends on VoltageRange selection.

C Prototype:

```
void  DualDAC8HW_Write2(BYTE bData)
```

Assembler:

```
mov   A, bData
lcall  DualDAC8HW_Write2
```

Parameters:

bData: Defines the 8-bit data code to be loaded in the DAC. The format of the bData parameter is binary. The following table shows values for the lowest, mid, and the highest reference voltage setting.

Code	1.3V Mode value	2.6V Mode value
00h	0 V	0 V
80h	0.65 V	1.3 V
FFh	1.3 V	2.6 V

Return Value:

None

Side Effects:

See Note ** at the beginning of the API section.

Sample Firmware Source Code

The C code illustrated here shows you how to use the DualDAC8HW User Module.

```
DualDAC8HW_SetVoltageRange(DualDAC8HW_2_6V);  // set 0-2.6V operation range
DualDAC8HW_Start();                           // start user module
DualDAC8HW_Write1(0x80);                      //set output voltage 1 to 1.3V
DualDAC8HW_Write2(0xFF);                      //set output voltage 2 to 2.6V
```

The same code in assembly is:

```
mov   A, DualDAC8HW_2_6V
call  DualDAC8HW_SetVoltageRange  ; set 0-2.6V operation range
call  DualDAC8HW_Start            ; start user module
mov   A, 0x80
call  DualDAC8HW_Write1           ; set output voltage 1 to 1.3V
mov   A, 0xFF
call  DualDAC8HW_Write2           ; set output voltage 2 to 2.6V
```

Configuration Registers

Table 1. DualDAC8HW_CONTROL_REG

Bit	7	6	5	4	3	2	1	0
Value	0	0	0	0	0	0	Mode	Enable

Enables user module functioning and is modified by calling Start or Stop API routine.

Mode determines the DACs output range. The value of this bit is determined by the choice made, for the parameter of the same name in the user module parameter of the Device Editor. The value can be changed by DualDAC8HW_SetVoltageRange() API.

Table 2. DualDAC8HW_DATA0_REG

Bit	7	6	5	4	3	2	1	0
Value	DATA[7:0]							

This register is the VDAC1 data register.

Table 3. DualDAC8HW_DATA1_REG

Bit	7	6	5	4	3	2	1	0
Value	DATA[7:0]							

This register is the VDAC2 data register.

Version History

Version	Originator	Description
1.0	DHA	Initial version

Note PSoC Designer 5.1 introduces a Version History in all user module datasheets. This section documents high level descriptions of the differences between the current and previous user module versions.

Copyright © 2009-2014 Cypress Semiconductor Corporation. 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 Designer™ and Programmable System-on-Chip™ 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.