Hardware Comparator Datasheet CMPHW V 1.0

Copyright © 2009-2013 Cypress Semiconductor Corporation. All Rights Reserved.

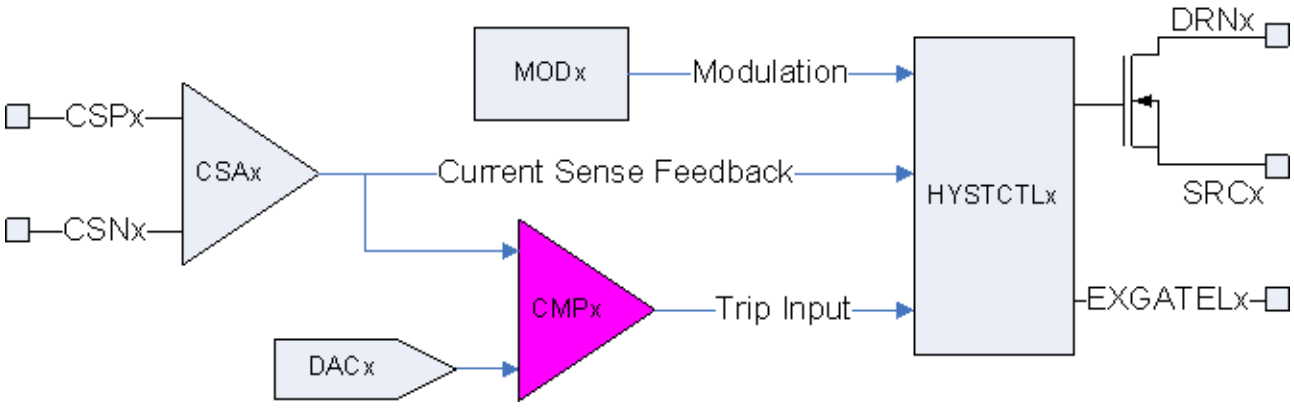
Resources	PSoC® Blocks	API Memory (Bytes)		Pins (per External I/O)
	CMP	Flash	RAM	
CY8CLED0xD, CY8CLED0xG	1	84	0	2

Features and Overview

- 10 mV hysteresis that can be enabled or disabled.
- Programmable output polarity.
- Programmable speed and power.
- Direct output connection available to trip input of the HYSTCTRL User Module blocks.
- Direct input selection from DualDAC8HW UM to achieve a precise, programmable comparator reference.

The CMPHW User Module (UM) is a dedicated hardware comparator block (not the comparator block implemented with PSoC analog and digital blocks) that compares two input signals and switches the output to indicate the larger signal. The comparator has rail-to-rail operation with a 10 mV hysteresis that can be enabled or disabled.This UM can operate in both fast and slow mode, a useful feature depending on the end application. For example, a typical over-current protection circuit is designed to respond faster to changes in the output variable and therefore would benefit from the fast mode feature. On the other hand, an application where the comparator is used to sense a slow changing variable, such as temperature, can benefit from the slow mode of this comparator using less system power.

Figure 1. CMPHW Block Diagram



Functional Description

The analog input levels can range from 0V to Vdd and are supplied from either the 8-bit DACs, current sense amplifiers (CSA), or from an external source. Note that the 8-bit DAC can only supply 2.6V. The input source is selected using the Device Editor.

The output can be fed to FN_0_0, FN_0_1, FN_0_2, FN_0_3 or to HYSTCTRL trip inputs.

Note

1. In other documentation for this device, the Function IO is written FN0[0], FN0[1], FN0[2] and so on. It is only written here with underscores (FN_0_0) for code requirements.
2. While using this block with the Hysteretic Controller User Module to give trip inputs, you can enable trip operation by writing '1' to the Trip bit in the HYSTCTRL_HYST_CONTROL_REG register as defined in the HYSTCTRL UM datasheet.

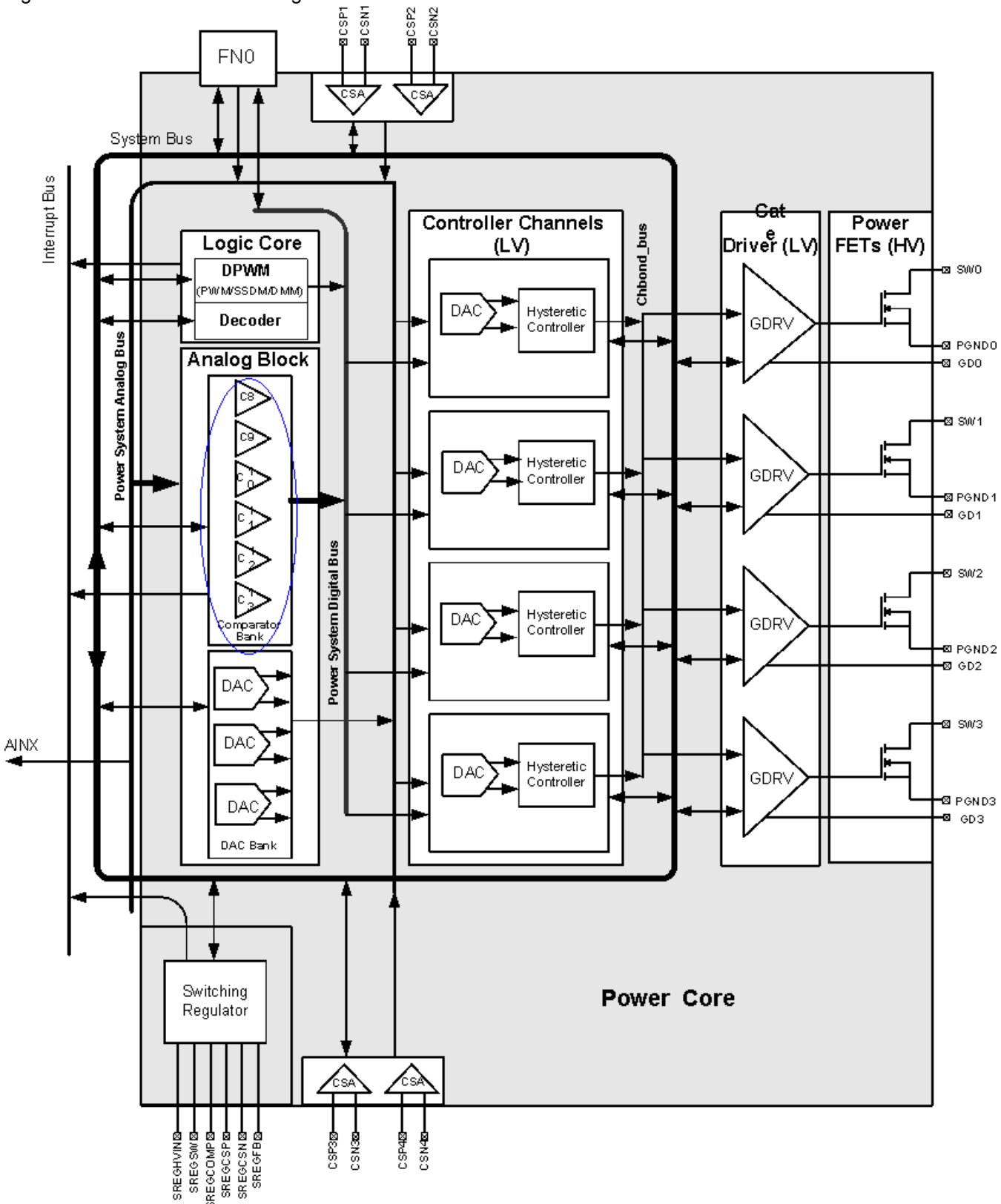
The output routing is selected using the user module parameters.

The speed of operation, hysteresis, and output polarity are selected using the corresponding user module parameters. Software APIs are also given to change these settings at run time.

Application Description

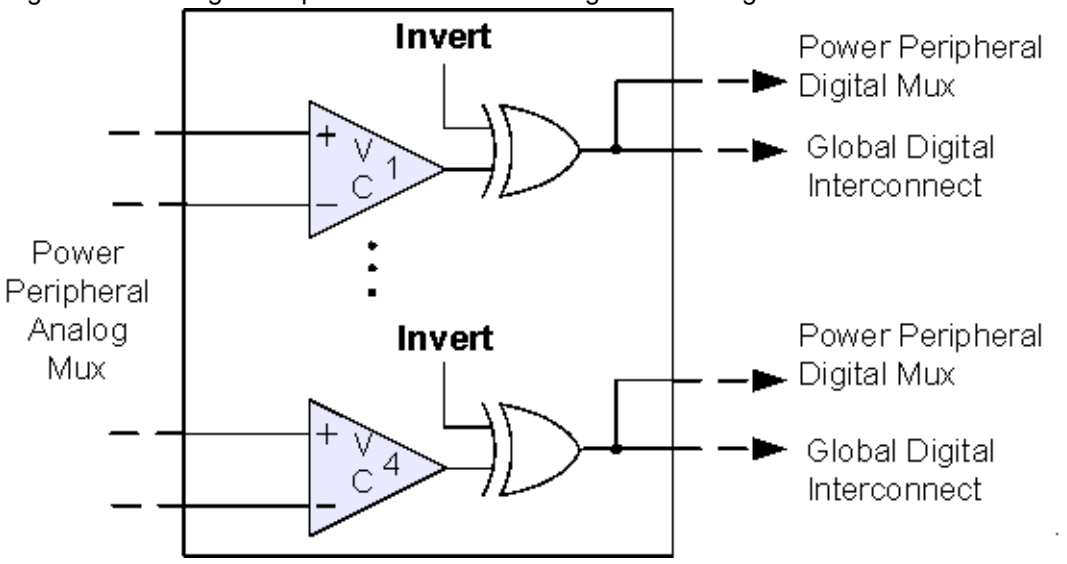
This CMPHW User Module is used in several different applications for the Power PSoC device family.

Figure 2. Functional Block Diagram



The CMPHW UM supports the Voltage Comparator, in which the voltage comparator bank is between the analog mux and the digital mux. This is shown in Figure 3. One example application is the Over Temperature Shut Out, in which it turns OFF the power section of the device by comparing the output of a temperature sensor and a known reference voltage. The voltage comparator is a valid application for this user module.

Figure 3. Voltage Comparators Between Analog Mux and Digital Mux



DC and AC Electrical Characteristics

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

Placement

The CMPHW occupies one of the possible CMPx blocks: CMP8, CMP9, CMP10, CMP11, CMP12, or CMP13.

Note The parameters of CMP0-7 can be edited using the HYSTCTL User Module since these comparators are directly tied to the hysteretic controller.

Parameters and Resources

Different devices have differing numbers of available resources, so not all noted selection values are available on all devices.

Positive Input

The Positive Input parameter allows you to choose the comparator positive input and contains the following input options, depending on placement of the comparator.

Negative Input

The Negative Input parameter allows you to choose the comparator negative input and contains the following input options, depending on placement of the comparator.

Negative Input	CMP8 Value	CMP9 Value	CMP10 Value	CMP11 Value	CMP12 Value	CMP13 Value
FN_0_0	●			●	●	
FN_0_1	●	●				●
FN_0_2		●	●		●	
FN_0_3			●	●		●
DAC8	●					
DAC9		●				
DAC10			●			
DAC11				●		
DAC12					●	
DAC13						●
VGND	●	●	●	●	●	●

Speed

The Speed parameter contains the following selections:

Speed	Current Consumption, µA	Response Time, ns (5 mV Overdrive)
Fast (Default)	600	100
Slow	165	360

Hysteresis

Enables or disables the 10 mV hysteresis.

Output Invert

Enables or disables the comparator output invert. Disabled is the default, where Vout = 5V when V+ Š V- else Vout = 0V.

Application Programming Interface

The Application Programming Interface (API) routines are given 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 given by the include files.

Each time a user module is placed, it is assigned an instance name. By default, PSoC Designer assigns the CMPHW_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 CMPHW for simplicity.

Note

** In this, as in all user module APIs, the values of the A and X register may be altered 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 functions 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.

CMPHW_Start

Description:

Enables the user module. The output is driven.

C Prototype:

```
void CMPHW_Start(void)
```

Assembly:

```
lcall CMPHW_Start
```

Parameters:

None.

Return Value:

None.

Side Effects:

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

CMPHW_Stop

Description:

Disables the user module. The output is low.

C Prototype:

```
void CMPHW_Stop()
```

Assembly:

```
lcall CMPHW_Stop
```

Parameters:

None.

Return Value:

None.

Side Effects:

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

CMPHW_EnableInt

Description:

Enables the interrupt associated with the analog comparator bus.

C Prototype:

```
void  CMPHW_EnableInt(void)
```

Assembly:

```
lcall  CMPHW_EnableInt
```

Parameters:

None.

Return Value:

None.

Side Effects:

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

CMPHW_DisableInt

Description:

Disables the interrupt associated with the analog comparator bus.

C Prototype:

```
void  CMPHW_DisableInt(void)
```

Assembly:

```
lcall  CMPHW_DisableInt
```

Parameters:

None.

Return Value:

None.

Side Effects:

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

CMPHW_SetSpeed

Description:

Sets the speed of operation. It contains the following selections:

Speed	Current Consumption, µA	Response Time, ns (5 mV Overdrive)
Fast (Default)	600	100
Slow	165	360

C Prototype:

```
void  CMPHW_SetSpeed(BYTE bSpeed)
```

Assembly:

```
mov   A, bSpeed
lcall  CMPHW_SetSpeed
```

Parameters:

bSpeedValue: This byte indicates the speed of operation in active mode. Symbolic names are given in C and assembly and their associated values are listed in the following table:

Symbolic Name	Value	Description
CMPHW_FAST	0x00	Active output with fast response time.
CMPHW_SLOW	0x04	Active output with slow response time

Return Value:

None.

Side Effects:

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

CMPHW_SetInvert

Description:

Sets the polarity of the output signal.

C Prototype:

```
void CMPHW_SetInvert(BYTE bInvert)
```

Assembly:

```
mov A, bInvert
lcall CMPHW_SetInvert
```

Parameters:

bInvert This byte indicates the polarity of the comparator output. Symbolic names are given in C and assembly and their associated values are listed in the following table:

Symbolic Name	Value	Description
CMPHW_INVERT_DISABLE	0x00	Noninverted output (default value)
CMPHW_INVERT_ENABLE	0x08	Inverted output

Return Value:

None.

Side Effects:

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

CMPHW_SetHysteresis

Description:

Enables or disables the hysteresis.

C Prototype:

```
void CMPHW_SetHysteresis(BYTE bHysteresis)
```

Assembly:

```
mov A, bHysteresis
lcall CMPHW_SetHysteresis
```

Parameters:

bHysteresisValue: This byte indicates whether to enable or disable the hysteresis. Symbolic names are given in C and assembly and their associated values are listed in the following table:

Symbolic Name	Value	Description
CMPHW_HYST_ENABLE	0x00	Hysteresis enabled.
CMPHW_HYST_DISABLE	0x02	Hysteresis disabled.

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 CMPHW User Module:

```
CMPHW_Start();
CMPHW_EnableInt();
```

The same code in assembly is:

```
call  CMPHW_Start;
call  CMPHW_EnableInt;
```

Configuration Registers

Table 1. CMPHW_CONTROL_REG

Bit	7	6	5	4	3	2	1	0
Value	0	0	0	0	Invert	Speed	Hysteresis	Enable

The Enable bit allows the UM to function and is modified by calling the Start or Stop API routine.

The Hysteresis bit enables or disables the 10 mV hysteresis. The value of this bit is determined by the choice made for the parameter of the same name under user module parameters in the Device Editor. The value can also be changed by calling the SetHysteresis API.

The Speed bit determines the speed and power consumption of the comparator. The value of this bit is determined by the choice made for the parameter of the same name under user module parameters in the Device Editor. The value can also be changed by calling the SetSpeed API.

The Invert bit determines inverted or noninverted output. The value of this bit is determined by the choice made for the Invert Output parameter under user module parameters in the Device Editor. The value can also be changed by calling the SetInvert API.

Version History

Version	Originator	Description
1.0	FRE	Corrected error in Parameters table of the CMPHW_SetInvert API.

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

Copyright © 2009-2013 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.