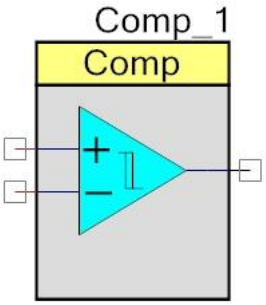
PSoC 4 Voltage Comparator (Comp)
1.0

Features



- Low input offset
- User controlled offset calibration
- Multiple speed modes
- Low-power mode
- Output routable to digital logic blocks or pins
- Selectable output polarity

General Description

The Voltage Comparator component gives you a hardware solution to compare two analog input voltages. You sample the output in software or routed to a digital component. There are three speed levels to allow you to optimize for speed or power consumption. You can also connect a reference or external voltage to either input.

The input offset is designed to be less than 1 mV over temperature and voltage. The hysteresis is selectable between 10 mV and no hysteresis.

When to Use theComparator

The Comparator gives you a fast comparison between two voltages, compared to using an ADC. Although you can use an ADC with software to compare multiple voltage levels, applications requiring fast response or little software intervention are good candidates for this comparator. Some example applications include CapSense®, power supplies or simple translation from an analog level to a digital signal.

A common configuration is to create an adjustable comparator by connecting a voltage DAC to the negative input terminal.

Input/output Connections

This section describes the various input and output connections for the Comparator.

Positive Input–Analog Input

This input is usually connected to the voltage that is being compared. This input can be routed from a GPIO or from an internal source.

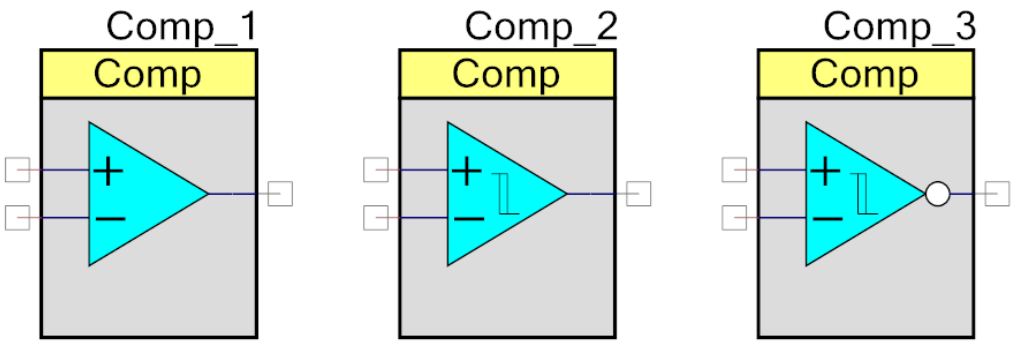
Negative Input– Analog Input

This input is usually connected to the reference voltage. This input can be routed from a GPIO or from an internal source.

Comparator Out–output

This is the digital comparison output. For a non-inverting configuration, this output goes high when the positive input voltage is greater than the negative input voltage. If the polarity is set to inverting, the output goes high when the negative input voltage is greater than the positive input voltage. The inverting configuration is implemented using an inverter in the UDB blocks and therefore requires a device that has UDB resources. The output can be routed to other component digital inputs such as interrupts, timers, etc.

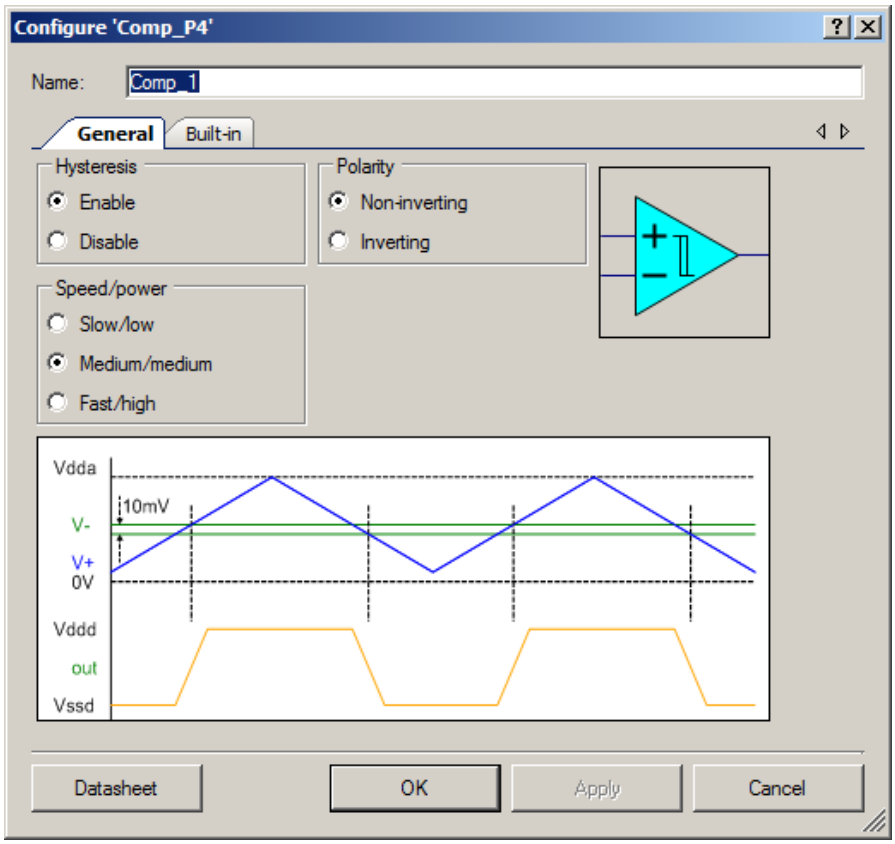
The symbol for this component is annotated to denote the selection of hysteresis or inversion of the output.



Component Parameters

Drag a Comparator onto your design and double click it to open the Configure dialog, as shown in Figure 1.

Figure 1. Configure Dialog

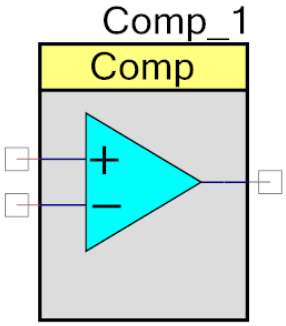


The component has the following parameters.

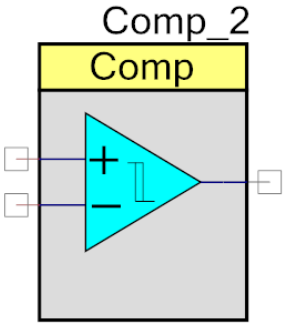
Hysteresis

This parameter gives you the ability to add approximately 10 mV of hysteresis to the Comparator. This helps to make certain that slowly moving voltages or slightly noisy voltages do not cause the output to oscillate when the two input voltages are nearly equal.

Hysteresis Disabled



Hysteresis Enabled



Speed/Power

This parameter provides a way for the user to optimize speed verses power consumption. The **Power** parameter allows you to select the power level: High Power, Medium Power, and Low Power.

Polarity

This parameter allows you to invert the output. This is useful for peripherals that require an inverted signal from the Comparator. The sampled signal state returned by the software API and the output seen by the power manager (see the *System Reference Guide* section on Alt Active and Sleep) is not affected by this parameter.

Note Inversion logic for the comparator is implemented using UDB resources.

Placement

Each Comparator is directly connected to specific GPIOs for its inputs along with being connected to the internal fabric. The output connection is routed to the digital fabric. Refer to the device datasheet for the part being used for the specific physical pin connections.

Resources

The Comparator uses one of the opamp (Constant Time Block – mini (CTBm)) blocks in PSoC 4. If the inverting output option is chosen a single macrocell in the UDB array is also used.

Application Programming Interface

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

By default, PSoC Creator assigns the instance name "Comp_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 "Comp"

Functions

Function	Description
Comp_Init()	Initializes or restores the component according to the customizer Configure dialog settings.
Comp_Enable()	Activates the hardware and begins component operation.
Comp_Start()	Performs all of the required initialization for the component and enables power to the block.
Comp_Stop()	Turns off the Comparator block.
Comp_GetCompare()	Returns compare result.
Comp_SetSpeed()	Sets the power and speed to one of three settings; LOWSPEED, MEDSPEED, HIGHSPEED.
Comp_ZeroCal()	Performs custom calibration of the input offset to minimize error for a specific set of conditions.
Comp_LoadTrim()	This function writes a value into the comparator trim register.
Comp_Sleep()	This is the preferred API to prepare the component for sleep.
Comp_Wakeup()	This is the preferred API to restore the component to the state when Comp_Sleep() was called.
Comp_SaveConfig()	Saves the configuration of the component.
Comp_RestoreConfig()	Restores the configuration of the component.

Global Variables

Function	Description
Comp_initVar()	Indicates whether or not the Comparator has been initialized. The variable is initialized to 0 and set to 1 the first time Comp_Start() is called. This allows the component to restart without reinitialization after the first call to the Comp_Start() routine. If reinitialization of the component is required, call Comp_Init() before calling Comp_Start(). Alternatively, the Comparator can be reinitialized by calling the Comp_Init() and Comp_Enable() functions

void Comp_Init(void)

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

Parameters: None

Return Value: None

Side Effects: All registers will be set to values according to the customizer Configure dialog.

void Comp_Enable(void)

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

Parameters: None

Return Value: None

Side Effects: None

void Comp_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 power level, and hysteresis are set. When called to restart the comparator following a Comp_Stop() call, the current component parameter settings are retained.

Parameters: None

Return Value: None

Side Effects: None

void Comp_Stop(void)

Description: Turn off the Comparator block.

Parameters: None

Return Value: None

Side Effects: Does not affect Comparator modes or power settings

uint32 Comp_GetCompare(void)

Description: This function returns a nonzero value when the voltage connected to the positive input is greater than the negative input voltage. This value is not affected by the Polarity parameter. This value always reflects a non-inverted stateconfiguration.

Parameters: None

Return Value: uint32: Comparator output state. Nonzero value when the positive input voltage is greater than the negative input voltage; otherwise, the return value is zero.

Side Effects: None

void Comp_SetSpeed(uint32 speed)

Description: Sets the power and speed to one of three settings; slow, medium, or fast.

Parameters: (uint32) speed:Comp_SLOWSPEED, Comp_MEDSPEED, Comp_HIGHSPEED

Return Value: None

Side Effects: None

uint32 Comp_ZeroCal(void)

Description: Performs custom calibration of the input offset to minimize error for a specific set of conditions: comparator reference voltage, supply voltage, and operating temperature. A reference voltage in the range at which the comparator will be used must be applied to the Negative and Positive inputs of the comparator while the offset calibration is performed. This can be done external to the device or by using an internal analog mux on the positive input that selects between the positive input signal for normal operation and the negative input signal for calibration.

Parameters: None

Return Value: uint32: The value from the comparator trim register after the offset calibration is complete. This value has the same format as the input parameter for the Comp_LoadTrim() API routine.

Side Effects: During the calibration procedure, the comparator output may behave erratically. The comparator output should be ignored during calibration.

void Comp_LoadTrim(uint32 trimVal)

Description: This function writes a value into the comparator offset trim register.

Parameters: uint32 trimVal: Value to be stored in the comparator offset trim register. This value has the same format as the parameter returned by the Comp_ZeroCal() API routine.

Return Value: None

Side Effects: None

void Comp_Sleep(void)

Description: This is the preferred API to prepare the component for sleep. The Comp_Sleep() API saves the current component state. Then it calls the Comp_Stop() function and calls Comp_SaveConfig() to save the hardware configuration. Call the Comp_Sleep() function before calling the CySysPmDeepSleep() or the CySysPmHibernate() functions.

Parameters: None

Return Value: None

Side Effects: None

void Comp_Wakeup(void)

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

Parameters: None

Return Value: None

Side Effects: Calling the Comp_Wakeup() function without first calling the Comp_Sleep() or Comp_SaveConfig() function may produce unexpected behavior.

void Comp_SaveConfig(void)

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

Parameters: None

Return Value: None

Side Effects: None

void Comp_RestoreConfig(void)

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

Parameters: None

Return Value: None

Side Effects: 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 and specific deviations – deviations that are applicable only for this component. This section provides information on component specific deviations. The project deviations are described in the MISRA Compliance section of the *System Reference Guide* along with information on the MISRA compliance verification environment.

The Comparator component does not have any specific deviations.

Sample Firmware Source Code

PSoC Creator provides numerous 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.

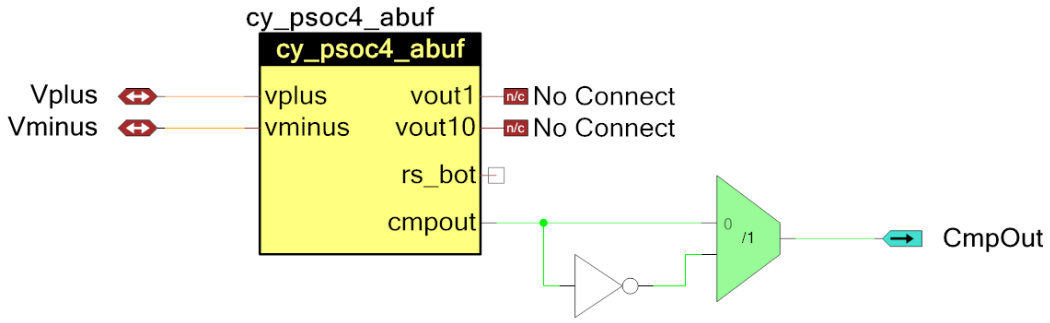
Refer to the "Find Example Project" topic in the PSoC Creator Help for more information.

Functional Description

The PSoC 4 Comparator component uses the opamp in the CTBm configured as a comparator. The CTBm supports 3 power choices that can be used to balance the response rate of the comparator with the power usage.

Block Diagram and Configuration

The component uses cy_psoc4_abuf primitive with the optional inverted output implemented using UDB resources.



Registers

See the chip Technical Reference Manual (TRM) for more information about registers.

API Memory Usage

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

The measurements were 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 4 (GCC)	
	Flash Bytes	SRAM Bytes
Default	280	8

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.

Comp DC Specifications

Parameter	Description	Min	Typ	Max	Units	Conditions
I_{DD}	Opamp Block current. No load.	–	–	–	–	
I_{DD_HI}	Power = high	–	1000	1300	µA	
I_{DD_MED}	Power = medium	–	320	500	µA	
I_{DD_LOW}	Power = low	–	250	350	µA	
V_{IN}	Charge pump on, $V_{DDA} \geq 2.7$ V	–0.05	–	VDDA – 0.2	V	
V_{CM}	Charge pump on, $V_{DDA} \geq 2.7$ V	–0.05	–	VDDA – 0.2	V	
V_{OS_TR}	Offset voltage, trimmed	1	±0.5	1	mV	High mode
V_{OS_TR}	Offset voltage, trimmed	–	±1	–	mV	Medium mode
V_{OS_TR}	Offset voltage, trimmed	–	±2	–	mV	Low mode
$V_{OS_DR_TR}$	Offset voltage drift, trimmed	–10	±3	10	µV/C	High mode
$V_{OS_DR_TR}$	Offset voltage drift, trimmed	–	±10	–	µV/C	Medium mode
$V_{OS_DR_TR}$	Offset voltage drift, trimmed	–	±10	–	µV/C	Low mode
CMRR	DC	70	80	–	dB	VDDD = 3.6 V
Vhyst_op	Hysteresis	–	10	–	mV	

Comp AC Specifications

Parameter	Description	Min	Typ	Max	Units	Conditions
Comp_mode	Comparator mode; 50 mV drive, Trise = Tfall (approx.)	–	–	–		
T_{PD1}	Response time; power = high	–	150	–	nsec	
T_{PD2}	Response time; power = medium	–	400	–	nsec	
T_{PD3}	Response time; power = low	–	1000	–	nsec	
T_op_wake	From disable to enable, no external RC dominating	–	300	–	µSec	

Component Changes

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

Version	Description of Changes	Reason for Changes / Impact
1.0.a	Minor datasheet edit.	
1.0	New Component	

© Cypress Semiconductor Corporation, 2013-2016. 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® is a registered trademark, and PSoC Creator™ and Programmable System-on-Chip™ are trademarks 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.