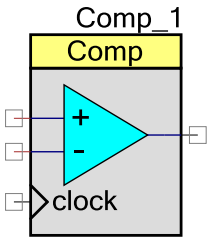
# Comparator (Comp)
1.50

## 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
- Configurable operation mode during Sleep and Hibernate

## General Description

The Comparator (Comp) component provides a hardware solution to compare two analog input voltages. The output can be sampled in software or digitally routed to another component. Three speed levels are provided to enable you to optimize for speed or power consumption. A reference or external voltage may be connected to either input.

You can also invert the output of the comparator using the Polarity parameter.

### When to use a Comparator

The Comparator can provide a fast comparison between two voltages as compared to using an ADC. Although an ADC can be used with software to compare multiple voltages 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 input and output connections for the Comp. An asterisk (*) in the list of I/O's states that the I/O may be hidden on the symbol under the conditions listed in the description of that I/O.

## Positive Input – Analog

This input is usually connected to the voltage that is being compared. This input can be routed to GPIOs and internal signals through analog globals, and to a selection of references.

## Negative Input – Analog

This input is usually connected to the reference voltage. This input can be routed to GPIOs and internal signals through the analog globals and to a selection of references.

## Comparator Out – Digital Output

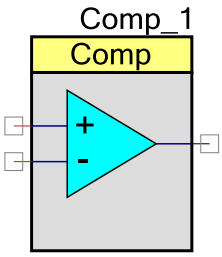
The output of the comparison. For the non-inverting configuration, this output goes high when the positive input voltage is greater than the negative input voltage. If the polarity is set to inverting, the output will go high when the negative input voltage is greater than the positive input voltage. The output can be routed to the digital interconnect and interrupt structures.

## clock – Digital Input *

The clock input will synchronize the comparator output to the rising edge of the clock when the Sync parameter is set to "Normal." This forces the comparator output to be sampled on the rising edge of the clock.

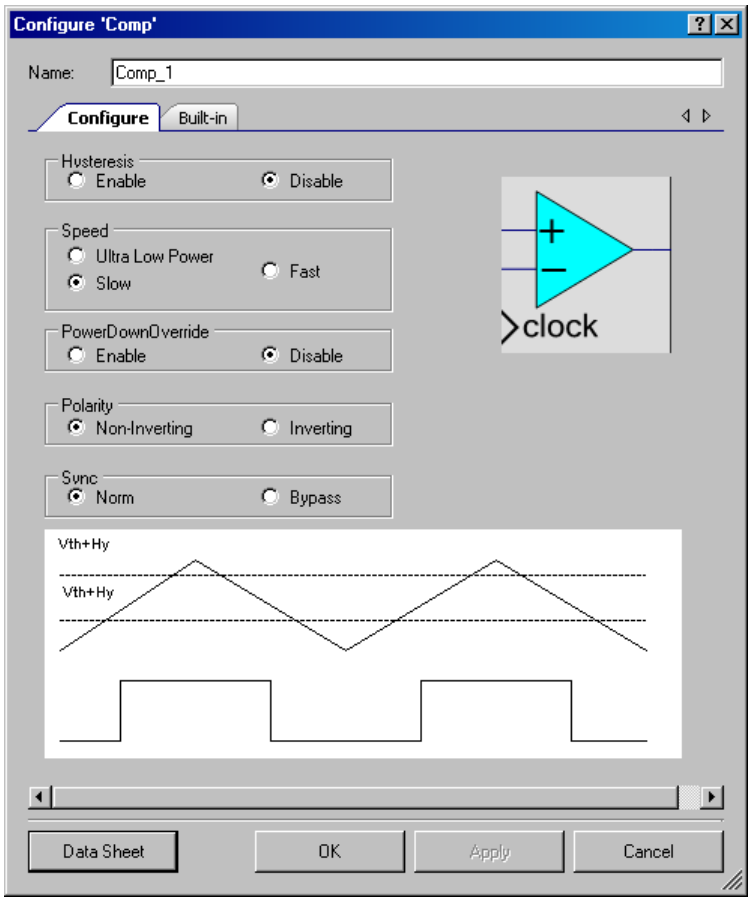
When the Sync parameter is set to "Bypass" the output is not synchronized and the clock input terminal no longer displayed on the component symbol.

Sync set to Bypass



# Parameters and Setup

Drag a Comparator onto your design and double-click it to open the Configure dialog.




The Comparator provides the following parameters.

## Hysteresis

This parameter enables allows you to add approximately 10 mV of hysteresis to the comparator. This will help to ensure that slowly moving voltages or slightly noisy voltages will not cause the output of the comparator to oscillate when the two input voltages are near equal.

## Speed

This parameter provides a way for the user to optimize speed verses power consumption.

| Speed Options | Description |
|---|---|
| Ultra Low Power | Use this setting for very low power applications. |
| Slow (default) | Use this setting for signals requiring response times slower than 80ns |

| Speed Options | Description |
|---|---|
| Fast | Use this setting for signals requiring response times faster than 80ns |

### Power Down Override

Enabling the power down override parameter causes the comparator to stay active during Sleep and Hibernate modes.

### Polarity

This parameter allows you to invert the output of the comparator. This is useful for peripherals that require an inverted signal from the comparator. The sampled signal state returned by the software API is not affected by this parameter.

| Polarity Options | Description |
|---|---|
| Inverting | Output goes high when positive input is less than the negative input |
| Non Inverting (default) | Output goes high when positive input is greater than negative input |

### Sync

This parameter selects between synchronizing the output with a clock and connecting directly to the comparator output. When Normal is selected, the output will change on the rising edge of the clock input.

| Sync Options | Description |
|---|---|
| Normal (default) | Sync the comparator output with the clock input. |
| Bypass | Connect the analog comparator directly to the output signal. |

## Placement

There are no placement specific options.

## Resources

The Comparator component uses one analog comparator block.

# Application Programming Interface

Application Programming Interface (API) routines allow you to configure the component using software. The following table lists the interface to each function. The subsequent 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 the instance 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".

| Function | Description |
|---|---|
| void Comp_Init(void) | Initialize or Restore default Comparator configuration. |
| void Comp_Enable(void) | Enable the Comparator. |
| void Comp_Start(void) | Initializes the Comparator with default customizer values. |
| void Comp_Stop(void) | Turn off Comparator |
| void Comp_SetSpeed(uint8 speed) | Set speed of comparator. |
| uint8 Comp_ZeroCal(void) | Zero the input offset of comparator. |
| uint8 Comp_GetCompare(void) | Returns compare result. |
| void Comp_LoadTrim(uint8 trimVal) | Write a value to the comparator trim register |
| void Comp_Sleep (void) | Stops the comparator operation and saves the user configuration. |
| void Comp_Wakeup (void) | Restores and enables the user configuration. |
| void Comp_SaveConfig(void) | Empty function. Provided for future usage. |
| void Comp_RestoreConfig(void) | Empty function. Provided for future usage. |
| void Comp_PwrDwnOverrideEnable(void) | Enables comparator operation in sleep mode |
| void Comp_PwrDwnOverrideDisable(void) | Disables comparator operation in sleep mode |

## Global Variables

| Variable | Description |
|---|---|
| Comp_initVar | Indicates whether 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.<br>If reinitialization of the component is required, then the Comp_Init() function can be called before the Comp_Start() or Comp_Enable() function. |

## void Comp_Init(void)

**Description:** Initializes/Restores default Comparator configuration.

**Parameters:** None

**Return Value:** None

**Side Effects:** None

## void Comp_Enable(void)

**Description:** Enables the Comparator.

**Parameters:** None

**Return Value:** None

**Side Effects:** None

## void Comp_Start(void)

**Description:** Initializes the Comparator with default customizer values. Enables and powers up the comparator.

**Parameters:** None

**Return Value:** None

**Side Effects:** None

## void Comp_Stop(void)

**Description:** Disable and power down the comparator.

**Note** This API is not recommended for use on PSoC 3 ES2 and PSoC 5 ES1 silicon. These devices have a defect that causes connections to several analog resources to be unreliable when not powered. The unreliability manifests itself in silent failures (e.g. unpredictably bad results from analog components) when the component utilizing that resource is stopped. It is recommended that all analog components in a design should be powered up (by calling the <INSTANCE_NAME>_Start() APIs) at all times. Do not call the <INSTANCE_NAME>_Stop() APIs.

**Parameters:** None

**Return Value:** None

**Side Effects:** None

## void Comp_SetSpeed(uint8 speed)

**Description:** This function selects one of three speed modes for the comparator. The comparator power consumption increases for the faster speed modes.

**Parameters:** (uint8) speed: Speed parameter, see table below for valid settings.

| Speed Options | Description |
|---|---|
| Comp_LOWPOWER | Use this setting for very low power applications. |
| Comp_SLOWSPEED | Use this setting for signals requiring response times slower than 80ns |
| Comp_HIGHSPEED | Use this setting for signals requiring response times faster than 80ns |

**Return Value:** None

**Side Effects:** None

## uint8 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 input of the comparator while the offset calibration is performed. The comparator component must be configured for Fast or Slow operation when calibration is performed. The calibration process will not work correctly if the comparator is configured in Low Power mode.

**Parameters:** None

**Return Value:** (uint8) 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. Refer to the *PSoC3, PSoC5 Technical Reference Manual* for a description of the comparator trim register.

**Side Effects:** During the calibration procedure the comparator output may behave erratically.

During the calibration procedure the analog routing switches for the comparator positive input will be reconfigured. This reconfiguration may affect the analog signal routing for other components that are connected to the comparator positive input.

When calibration is complete all routing and comparator configuration registers will be restored to the state they were in before calibration occurred.

## uint8 Comp_GetCompare(void)

**Description:** This function returns a non-zero 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 state.

**Parameters:** None

**Return Value:** (uint8) comparator output state – none zero value when the positive input voltage is greater than the negative input voltage, otherwise the return value is zero.

**Side Effects:** None

## void Comp_LoadTrim(uint8 trimVal)

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

**Parameters:** (uint8) trimVal: Value to be stored in the comparator trim register.
This value has the same format as the parameter returned by the Comp_ZeroCal() API routine. Refer to the *PSoC3, PSoC5 Technical Reference Manual* for a description of the comparator trim register.

**Return Value:** None

**Side Effects:** None

## void Comp_SaveConfig(void)

**Description:** Saves the user configuration.

**Parameters:** None

**Return Value:** None

**Side Effects:** Empty function. Implemented for future usage. No effect by calling this function.

## void Comp_RestoreConfig(void)

**Description:** Restores the user configuration.

**Parameters:** None

**Return Value:** None

**Side Effects:** Empty function. Implemented for future usage. No effect by calling this function.

## void Comp_Sleep(void)

**Description:** Stops the comparator operation and saves the user configuration.

**Parameters:** None

**Return Value:** None

**Side Effects:** None

### void Comp_Wakeup(void)

**Description:** Restores and enables the user configuration.

**Parameters:** None

**Return Value:** None

**Side Effects:** None

### void Comp_PwrDwnOverrideEnable(void)

**Description:** This is the power down override feature. This function allows the component to stay active during sleep mode.

**Parameters:** None

**Return Value:** None

**Side Effects:** None

### void Comp_PwrDwnOverrideDisable(void)

**Description:** This is the power down override feature. This function allows the comparator to stay inactive during sleep mode.

**Parameters:** None

**Return Value:** None

**Side Effects:** None

## Sample Firmware Source Code

The following is a C language example demonstrating the basic functionality of the Comp. This example assumes the component has been placed in the schematic and renamed to "Comp_1".

```
#include <device.h>

void main()
{
    uint8 cmpVal;
    Comp_1_Start( );                    /* Turn on comparator */
    Comp_1_SetSpeed(Comp_1_HIGHSPEED); /* Set high speed mode */
    cmpVal = Comp_1_GetCompare();       /* Return comparator state */
}
```

# Functional Description

The Comparator component is similar to most common analog comparators on the market. The Comparator component offer greater flexibility in supporting the configuration of options such as speed/power, hysteresis, and clock synchronization.

# DC and AC Electrical Characteristics

The Comp will operate at all valid supply voltages.

## Comparator DC Specifications

| Parameter | Description | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| VIOFF | Input offset voltage in fast mode | Factory trim | - | - | ±2 | mV |
| | Input offset voltage in slow mode | Factory trim | - | - | ±2 | mV |
| VIOFF | Input offset voltage in fast mode | Custom trim | - | - | ±2 | mV |
| | Input offset voltage in slow mode | Custom trim | - | - | ±1 | mV |
| VIOFF | Input offset voltage in ultra low power mode | | - | ±12 | - | mV |
| VHYST | Hysteresis | Hysteresis enable mode | - | 10 | 32 | mV |
| VICM | Input common mode voltage | Fast mode | 0 | - | VDDA-0.1 | V |
| | | Slow mode | 0 | - | VDDA | V |
| CMRR | Common mode rejection ratio | | 50 | - | - | dB |
| ICMP | High current mode/fast mode | | - | - | 400 | µA |
| | Low current mode/slow mode | | - | - | 100 | µA |
| | Ultra low power mode | | - | 6 | - | µA |

# Component Changes

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

| Version | Description of Changes | Reason for Changes / Impact |
|---|---|---|
| 1.50.d | Minor datasheet edits. | |
| 1.50.c | Minor datasheet edits. | |
| 1.50.b | Minor datasheet edits. | |
| 1.50.a | Added Known Problems and Solutions to datasheet | To provide a workaround for hysteresis problem in PSoC3 ES2 silicon. |
| 1.50 | Added Sleep/Wakeup and Init/Enable APIs. | To support low power modes, as well as to provide common interfaces to separate control of initialization and enabling of most components. |
| | Updated Configure dialog with customized interface. | The updated Configure dialog makes it easier to use. There is also a preview of how the component will change based on various selections. |
| | Added Power Down Override parameter to the Configure dialog. | To allow configuration of Comparator to operate during sleep and hibernate modes. |
| | Added _PwrDwnOverrideEnable / _PwrDwnOverrideDisable APIs. | To allow the component to stay active / inactive during sleep mode. |

# Known Problems and Solutions

| Problem & Solution | Cypress ID |
|---|---|
| **Hysteresis implementation for PSoC3 ES2 silicon in version 1.50 of comparator component is broken.**<br>Workaround for this issue is :<br>To disable the Hysteresis write: Comp_CR \|= 0x20;<br>To enable the Hysteresis write: Comp_CR &=0xDF; | 84266 |

© Cypress Semiconductor Corporation, 2009-2016. 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.