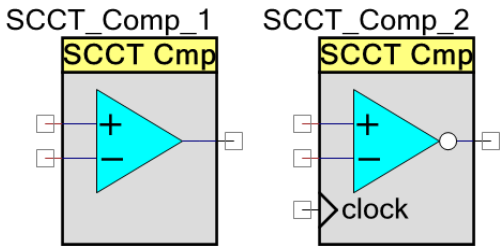
SC/CT Comparator (SCCT_Comp)

1.0

Features

- Output routable to digital logic blocks or pins
- Selectable output polarity



General Description

The SC/CT Comparator (SCCT_Comp) component provides a hardware solution to compare two analog input voltages. The implementation uses a mode of the Switched Capacitor / Continuous Time (SC/CT) analog block to implement the comparator. The output can be digitally routed to another component. A reference or external voltage can be connected to either input. You can also invert the output of the comparator using the Polarity parameter.

When to Use a Comparator

The SCCT_Comp is not as high performance as the dedicated comparator. Hysteresis support is not available. However, it is still useful in applications that don't have strict requirements for offset voltage and response time parameters and the number of required comparators exceeds the available dedicated comparators. This does yield a solution that requires less software intervention and provides a faster response time when compared with an ADC implementation.

This Comparator type should only be used if more comparators are needed than can be satisfied by the fixed function comparators in the device.

Input/Output Connections

This section describes the input and output connections for the SCCT_Comp. An asterisk (*) in the list of I/Os 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 or to a selection of internal references.

Negative Input – Analog

This input is usually connected to the reference voltage. This input can be routed to GPIOs or to a selection of internal references.

clock – Digital Input *

The clock input synchronizes 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.

This clock is intended for synchronization of the component output for use with logic in the UDB array. This is implemented using a Sync component.

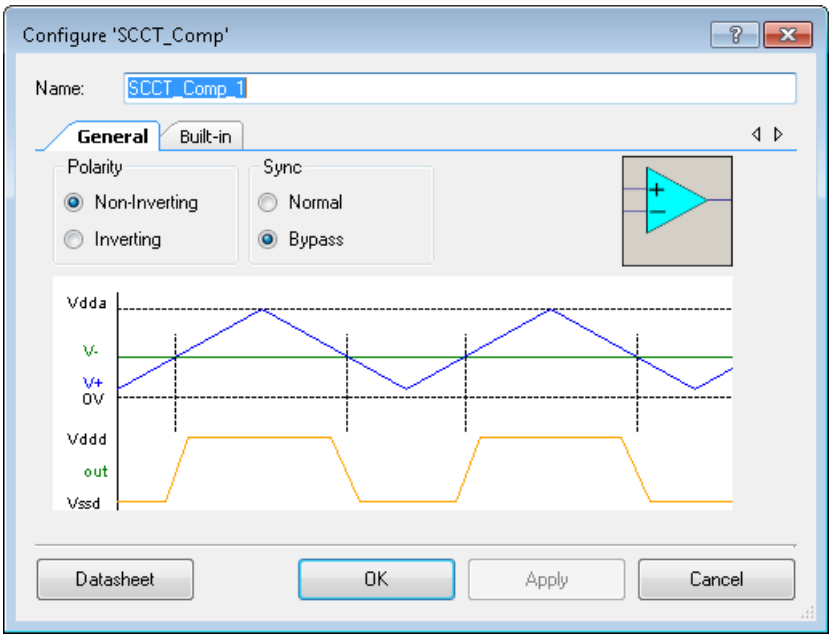
Comparator Output – Digital Output

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. In this case an inverter in the UDB array is used to invert the comparator's output signal. The output can be synchronized with the UDB array at the provided input clock frequency through a Sync component, when the **Sync** parameter is set to **Normal.**

The output can be routed to other component digital inputs such as interrupts, timers, etc.

Component Parameters

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



The Comparator provides the following parameters.

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.

Note Inversion logic for the comparator is implemented using UDBs.

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

Sync

This parameter selects between synchronizing the output with a clock or routing the output directly. 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	Route the comparator output directly

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 “SCCT_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
Comp_Start()	Initializes the component with default customizer values and enables operation.
Comp_Stop()	Turns off the component.
Comp_Sleep()	Stops component operation and saves the user configuration.
Comp_Wakeup()	Restores and enables the user configuration.

Function	Description
Comp_Init()	Initializes or restores default component configuration.
Comp_Enable()	Enables the component.

Global Variables

Variable	Description
Comp_initVar	Indicates whether the component 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 re-initialization after the first call to the Comp_Start() routine. If re-initialization of the component is required, then the Comp_Init() function can be called before the Comp_Start() or Comp_Enable() function.

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 component is initialized to the configuration from the customizer. When called to restart the component following a Comp_Stop() call, the current component parameter settings are retained.

Parameters: None

Return Value: None

Side Effects: If the initVar variable is already set, this function only calls the Comp_Enable() function.

void Comp_Stop(void)

Description: Turns off the component. It will disable the related SC/CT block.

Parameters: None

Return Value: None

Side Effects: None

void Comp_Sleep(void)

Description: This is the preferred API to prepare the component for low power mode operation (disable for this case). If the component is enabled it configures the comparator for low power operation.

Call the Comp_Sleep() function before calling the CyPmSleep() or the CyPmHibernate() function. Refer to the PSoC Creator *System Reference Guide* for more information about power management functions.

Parameters: None

Return Value: None

Side Effects: In the inverting mode of component, the output is implemented using UDB logic. Hence, the component output level is high after this sleep API is called and before sleep mode is entered.

void Comp_Wakeup(void)

Description: This is the preferred API to restore the component to the state befopre Comp_Sleep() was called.

Parameters: None

Return Value: None

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

void Comp_Init(void)

Description: Initializes or restores the component according to the customizer 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: None

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

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
- specific deviations – deviations that are applicable only for this component

This section provides information on component-specific deviations. Project deviations are described in the MISRA Compliance section of the *System Reference Guide* along with information on the MISRA compliance verification environment.

The SC/CT Comparator component does not have any specific deviations.

The component has the following embedded components: Clock.

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

This component provides access to the SC/CT block in PSoC 3/PSoC 5LP configured in comparator mode. It uses the open loop naked opamp as an analog comparator. To decrease the comparator response time, all of the amplifier compensation capacitors are disabled.

Hysteresis support is not available. The opamp is set to the highest speed configuration.

The output polarity can be either non-inverting or inverting. This capability is implemented using UDB logic.

When the comparator is disabled the output is low when in non-inverting mode and high when in inverting mode.

Registers

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

Resources

The SCCT_Comp component uses the following device resources:

- SC/CT (switched capacitor/continuous time) analog block
- Invertor from UDB logic (if Polarity parameter is set to Inverting)
- Sync cell from UDB logic (if Sync parameter set to Normal)

API Memory Usage

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

The measurements have been 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 3 (Keil_PK51)		PSoC 5LP (GCC)	
	Flash Bytes	SRAM Bytes	Flash Bytes	SRAM Bytes
Default	185	2	216	5

PSoC 3/PSoC 5LP DC and AC Electrical Characteristics

Parameter	Description	Conditions	Min	Typ	Max	Units
V_{OS}	Input offset voltage		TBD	TBD	TBD	mV
I_{CMP}	Quiescent current		TBD	TBD	TBD	µA
V_{HYST}	Hysteresis		N/A	N/A	N/A	N/A
CMRR	Common mode rejection ratio		TBD	TBD	TBD	dB
V_{ICM}	Input common mode voltage		TBD	TBD	TBD	V
T_{RESP}	Response Time		TBD	TBD	TBD	ns

Component Errata

This section lists known problems with the component.

Cypress ID	Component Version	Problem	Workaround
191257	v1.0	This component was modified without a version number change in PSoC Creator 3.0 SP1. For further information, see Knowledge Base Article KBA94159 (www.cypress.com/go/kba94159).	No workaround is necessary. There is no impact to designs.

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.b	Minor datasheet edits.	
1.0.a	Updates to datasheet.	Added Component Errata section to document that the component was changed, but there is no impact to designs.
		Minor changes to GUI screen capture and update to parameter text.
1.0	Initial release	

© Cypress Semiconductor Corporation, 2013-2016. 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, 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.