Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

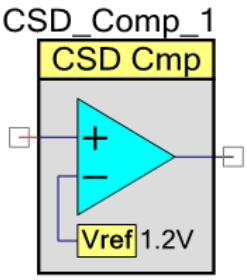
Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

1.2 Volt Comparator (CSD Comp)

1.0

Features



- Two speed modes
- Any pin in Port 0, Port 1 and Port 2 can be connected to the non-inverting input
- The inverting input is connected to a 1.2 V reference

General Description

The 1.2 Volt Comparator (CSD_Comp) component provides a hardware solution to compare one input voltage with the internal 1.2 V reference (Vref) of the PSoC 4000 family of devices. The CSD_Comp component makes use of the comparator that is present in the Capacitive Sensing Delta-Sigma Modulator (CapSense® CSD) block. This comparator functionality is only present on the PSoC 4000 family of devices and only available when not using the CapSense CSD block for Capacitive Sensing. You can sample the output in software or route it to a GPIO. There are two speed levels to allow you to optimize for response time or power consumption.

Although the CSD_Comp operates from a fixed Vref, it can be made to trigger at different voltage levels using external/internal components. With the help of an additional GPIO, the CSD_Comp can be used to generate interrupts. The CSD_Comp can also be operated with multiplexed inputs. Refer to the Functional Description section for more information.

When to Use the CSD_Comp

The CSD_Comp can be used in many applications to monitor external voltages and analog signals from transducers. Such applications include a thermal printer, a battery monitor, and an on-off controller.

Note This CSD_Comp component uses resources from the PSoC 4 CapSense CSD block. Therefore, the CSD_Comp cannot be used in applications that also require capacitive sensing.

Input/Output Connections

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

Non-inverting Input – Analog Input

This input is usually connected to the voltage that is being compared. This input can be routed from any GPIO in Port 0, Port 1 and Port 2.

Comparator Out – output

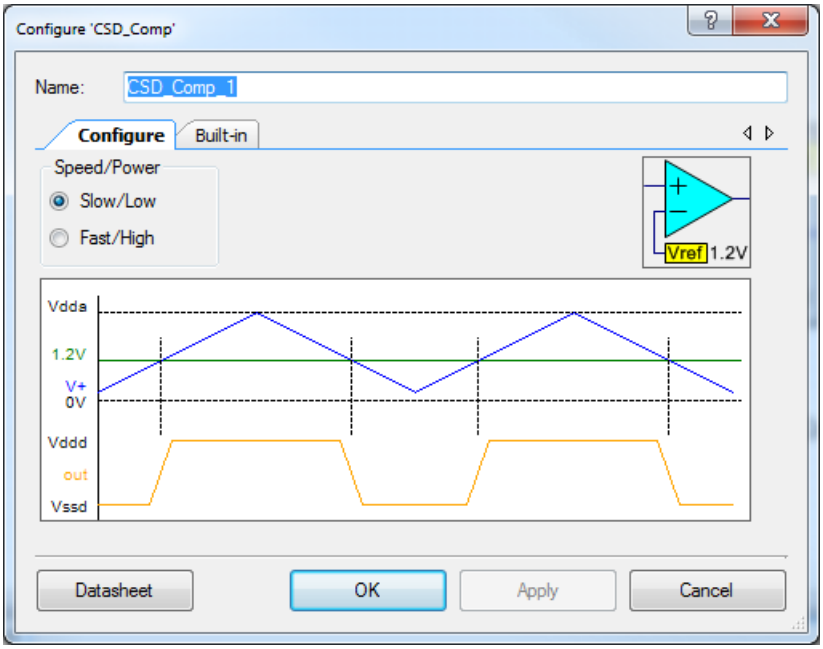
This is the digital comparison output. This output goes high when the non-inverting input voltage is greater than the Vref. The CSD_Comp output is always asynchronous.

Note The output can be routed only to three dedicated pins: P0[1], P0[4] or P1[6].

Component Parameters

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

Configure Tab



Speed/Power

This parameter provides a way to optimize speed versus power consumption. The **Speed/Power** parameter provides two levels: Slow/Low and Fast/High. The High Power mode gives better response time.

Placement

The CSD_Comp is placed as a Fixed Function block and all placement information is provided to the API through the *cyfitter.h* file.

Resources

The CSD_Comp uses the AMUXBUS-A and Sense Comparator from the CSD fixed-function block.

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 4000 (GCC)	
	Flash Bytes	SRAM Bytes
Default	332	2

Application Programming Interface

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

By default, PSoC Creator assigns the instance name “CSD_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 “CSD_Comp”.

Function	Description
CSD_Comp_Start()	Performs all of the required initialization for the component, enables power to the CSD block and enables the comparator.
CSD_Comp_Stop()	Turns off the Comparator block.
CSD_Comp_Init()	Initializes or restores the component according to the customizer Configure dialog settings.

Function	Description
CSD_Comp_Enable()	Activates the hardware and begins component operation.
CSD_Comp_GetCompare()	Returns compare result.
CSD_Comp_SetSpeed()	Sets the power and speed to one of two settings: SLOW_SPEED or FAST_SPEED.
CSD_Comp_Sleep()	Prepare the component for sleep.
CSD_Comp_Wakeup()	Restore the component to the state when Comp_Sleep() was called.

Global Variables

Knowledge of these variables is not required for normal operations.

Function	Description
CSD_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 CSD_Comp_Start(void)

Description: Performs all of the required initialization for the component, enables power to the CSD block and enables the comparator. The first time the routine is executed the power level is set. When called to restart the comparator following a Stop() call, the current component parameter settings are retained.

Parameters: None

Return Value: None

Side Effects: None

void CSD_Comp_Stop(void)

Description: Disable the comparator. Power to the CSD block remains enabled. All other comparator settings remain unchanged and will continue to be the settings used when the comparator becomes active again by calling Start().

Parameters: None

Return Value: None

Side Effects: None

void CSD_Comp_Init(void)

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

Parameters: None

Return Value: None

Side Effects: None

void CSD_Comp_Enable(void)

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

Parameters: None

Return Value: None

Side Effects: None

uint32 CSD_Comp_GetCompare(void)

Description: This function returns a nonzero value when the voltage connected to the non-inverting input is greater than the Vref.

Parameters: None

Return Value: uint32: Comparator output state. Nonzero value when the non-inverting input voltage is greater than the Vref; otherwise, the return value is zero.

Side Effects: None

void CSD_Comp_SetSpeed(uint32 speed)

Description: Sets the speed to one of two settings: slow or fast.

Parameters: (uint32) speed: Comparator response time parameter. See table below.

Value	Description
CSD_Comp_SLOW_SPEED	Use this setting for very low power applications
CSD_Comp_FAST_SPEED	Use this setting for fastest response time (50 nsec)

Return Value: None

Side Effects: None

void CSD_Comp_Sleep(void)

Description: Prepare the component for sleep. Call the Sleep() function before calling the CySysPmDeepSleep() functions.

Parameters: None

Return Value: None

Side Effects: None

void CSD_Comp_Wakeup(void)

Description: Restore the component to the state when Sleep() was called. If the component was enabled before the Sleep() function was called, the Wakeup() function will also re-enable the component.

Parameters: None

Return Value: None

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

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 Comparator component has the following specific deviations:

MISRA-C: 2004 Rule	Rule Class[1]	Rule Description	Description of Deviation(s)
19.7	A	A function should be used in preference to a function-like macro.	Deviated since function-like macros are used to allow more efficient code.

This component has the following embedded component: Clock. Refer to the corresponding component datasheet for information on its MISRA compliance and specific deviations.

[1] Required / Advisory

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.

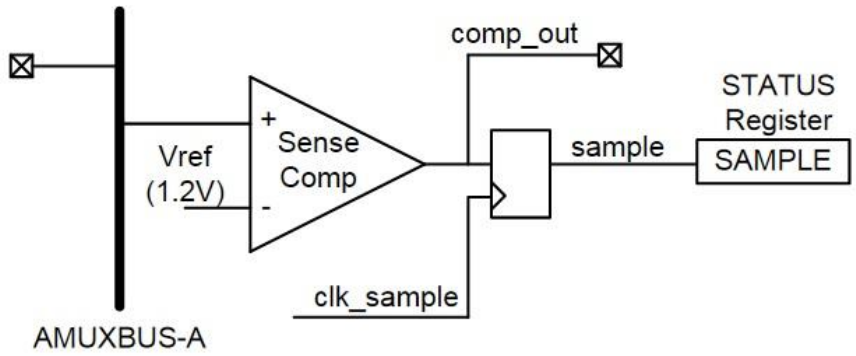
Refer to the “Find Example Project” topic in the PSoC Creator Help for more information.

Functional Description

This component uses the sense comparator in the CSD fixed-function block of the PSoC 4000 family of devices. The inverting input of this comparator is connected to a fixed reference of 1.2 V. The non-inverting input can be connected to any pins in Port 0, Port 1 and Port 2 using AMUXBUS-A. The output can be routed to any of the three dedicated pins P0[1], P0[4] or P1[6].

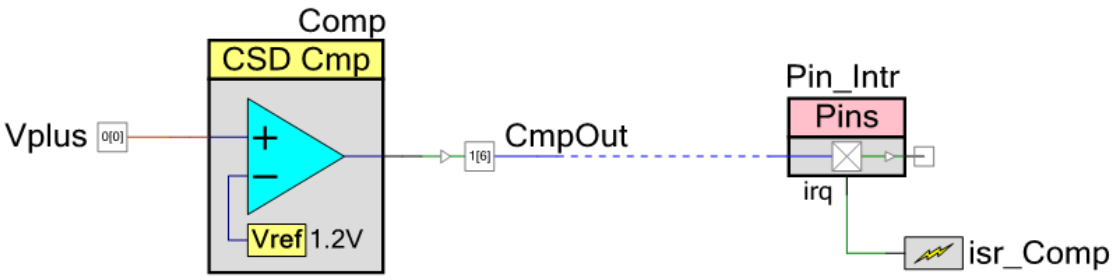
When the input voltage at the non-inverting input is above the Vref, the comparator output (comp_out) will be at a logic HIGH state. When the input voltage is below the reference, the output will be at a logic LOW state.

The component uses the CSD_CONFIG register to enable, disable and select the speed of the comparator. The CSD_STATUS register could be used for polling via the GetCompare() API. A high-level block diagram is shown below. Refer to the appropriate device *Technical Reference Manual (TRM)* for a detailed description of each register.



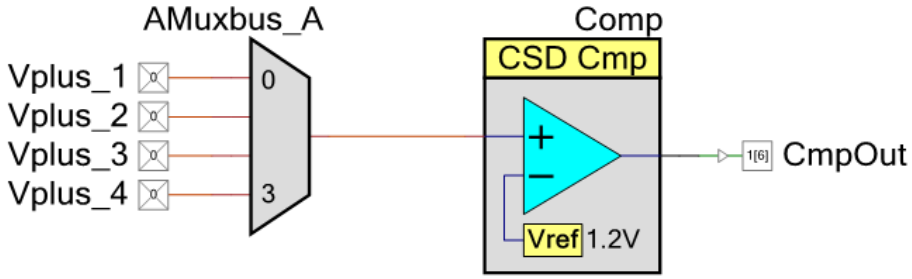
Interrupt generation

There are no Interrupts available with the Comparator component. But with the help of an additional GPIO, the comparator can be used to generate interrupts. For this the comparator output pin should be externally connected to another GPIO as shown in the image below.



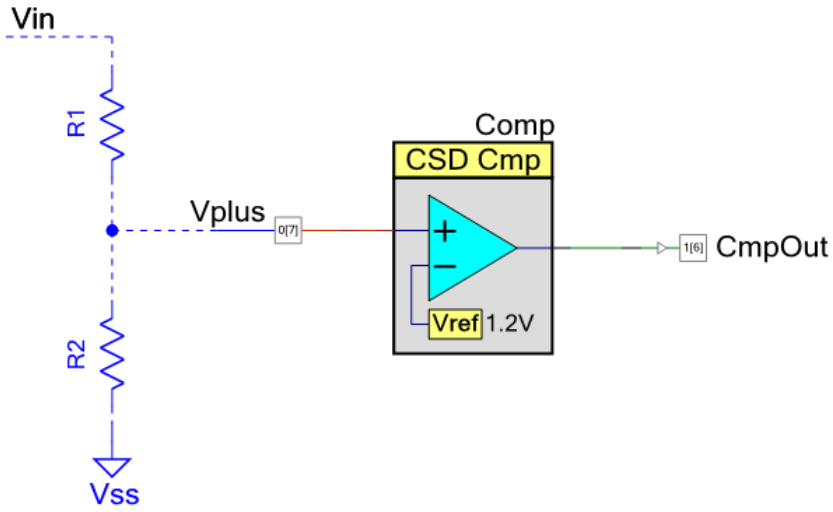
Multiplexed operation

By using an Analog Multiplexer component (AMuxbus_A), the input of the comparator can be multiplexed to all available GPIOs in Ports 0, 1 and 2.



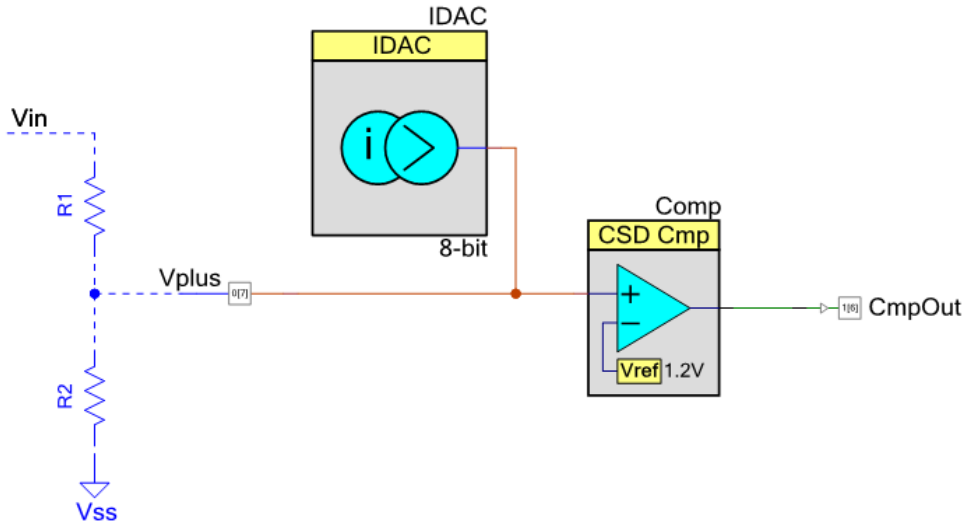
Operating at a voltage level above 1.2 V

By using an external resistive divider, the comparator can be used to detect voltages above 1.2 V. The output transition happens when Vin goes above/below 1.2*(1 + R1/R2) volts. The R1 and R2 must have high values to reduce errors due to source resistance.



Operating at a voltage level below 1.2 V

By using an IDAC and an external resistive divider, the comparator can be used to detect voltages below 1.2 V. The output transition happens when Vin goes above/below 1.2*(1 + R1/R2) − (R1*IDAC) volts.



Wakeup from Deep-Sleep Mode

This component operates in Active power mode only. The following is the correct Deep-Sleep mode entry procedure:

```
/* Prepares Comparator to wake up from Sleep mode */
Comp_Sleep();

/* Switches to the Deep-Sleep mode */
CySysPmDeepSleep();

/* Prepares Comparator to work in Active mode */
Comp_Wakeup();
```

Registers

See the device *Technical Reference Manual* for more information about registers.

Component Debug Window

The Comparator component supports the PSoC Creator component debug window. Refer to the appropriate device *Technical Reference Manual* for a detailed description of each register. The following registers are displayed in the Comparator component debug window:

- CONFIG – Configuration register
- STATUS – Status register

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.

DC Specifications

Parameter	Description	Min	Typ	Max	Units	Conditions
I_{CMP1}	Block current, High Bandwidth mode	–	–	120	µA	
I_{CMP2}	Block current, Low Power mode	–	–	90	µA	
$V_{OFFSET1}$	Offset voltage, High Bandwidth mode	–	10	30	mV	
$V_{OFFSET2}$	Offset voltage, Low Power mode	–	10	30	mV	

Parameter	Description	Min	Typ	Max	Units	Conditions
Z_{CMP}	DC input impedance of comparator	35	-	-	MΩ	
V_{INP_COMP}	Comparator input range	0	-	3.6	V	Max input voltage is lower of 3.6 V or V_{DD}

AC Specifications

Parameter	Description	Min	Typ	Max	Units	Conditions
T_{COMP1}	Response Time High Bandwidth mode, 50-mV overdrive	–	–	50	ns	
T_{COMP2}	Response Time Low Power mode, 50-mV overdrive	–	–	100	ns	

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 edits.	
1.0	First component version.	First component version.

© Cypress Semiconductor Corporation, 2014-2017. 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.