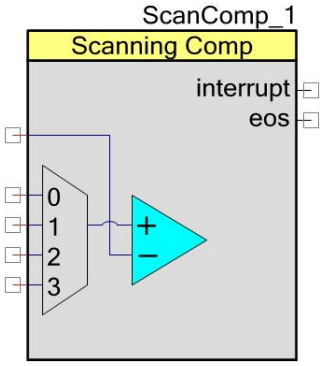
Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the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

Scanning Comparator (ScanComp)
1.10



Features

- Scan up to 64 single ended or differential channels automatically

  Note The number of input and output channels will be limited by the hardware available in the device being used.
- Up to 64 outputs routable to digital logic blocks or pins
- Multiple comparison modes

General Description

The Scanning Comparator (ScanComp) component provides a hardware solution to compare up to 64 pairs of analog input voltages signals using just one hardware comparator. The sampled comparator outputs can be enabled for connection in digital hardware. A reference or external voltage can be connected to each input.

When to Use a Scanning Comparator

The Scanning Comparator component can provide a comparison of up to 64 analog signals. Although an ADC can be used with software to compare multiple voltage levels, applications requiring fast response or little software intervention are good candidates for this scanning comparator. Some example applications include power supplies, or simple translation from an analog level to a digital signal. This component is useful when more signals must be compared than the number of comparators present in the device.

Input/Output Connections

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

clock - Digital Input*

The Clock for multiplexing between comparator inputs. Does not exist when the internal clock configuration is used.

vplus[n] – Analog

Differential pair voltage positive input, usually the signal being compared.

vminus[n] – Analog*

Differential pair voltage negative input, usually used as a reference. Does not exist when all inputs have a common negative reference, or when the VDAC is used.

vminus – Analog*

Voltage input used as a negative reference for all of the positive inputs. Does not exist when all inputs have independent references, or when the VDAC is used.

interrupt - Digital Output

ORed result of all channels interrupts**.**

eos – Digital Output

End of scan.

cmpout[n] – Digital Output*

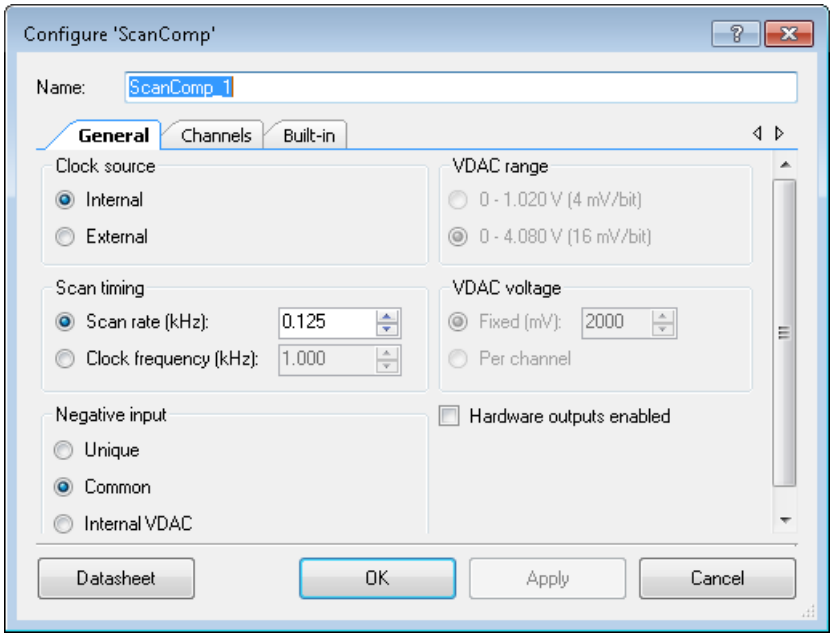
Individual comparator outputs for each input.

Note: These may be disabled in the customizer.

Component Parameters

Drag a Scanning Comparator onto your design and double click it to open the Configure dialog. This dialog has the following tabs with different parameters.

General Tab



Clock source

This parameter selects a sequencing clock implementation: internal or external. Default setting is internal.

Scan Rate

This parameter sets a scan rate in cycles per second when the clock source is internal. This will affect or be affected by the clock frequency parameter. Not available if the clock source is external.

The valid range for the scan rate parameter depends on the range of the Clock Frequency parameter and on the number of channels.

Clock Frequency

This parameter sets a clock frequency in kHz when the clock source is internal. This will affect or be affected by the scan rate parameter. Frequency selection is not available if the clock source is

external. The actual internal clock frequency can be observed in the Design-Wide Resources Clock Editor as “ScanComp_Clock_int”.

Negative Input Options	Range of the clock frequency parameter (kHz)
Unique	1–10000. Default is 1.
Common (default)	1–10000. Default is 1.
Internal VDAC. VDAC range: 0–1.020 V	1–2000. Default is 1.
Internal VDAC. VDAC range: 0–4.080 V	1–500. Default is 1.

Hardware Outputs Enabled

This check box enables the latched outputs of the component. Default setting is disabled.

Negative Input

This parameter selects the negative input mode of the component.

Negative Input Options	Description
Unique	Each channel has its own negative analog input that is multiplexed.
Common (default)	All channels use the same negative analog input.
Internal VDAC	All channels use for their negative analog input the output from a VDAC that is configured by the component. Not available for PSoC 4 devices.

VDAC Configuration - Range

This parameter selects one of the two voltage ranges for the internal VDAC. Not applicable if the Negative Input parameter is set to Unique or Common.

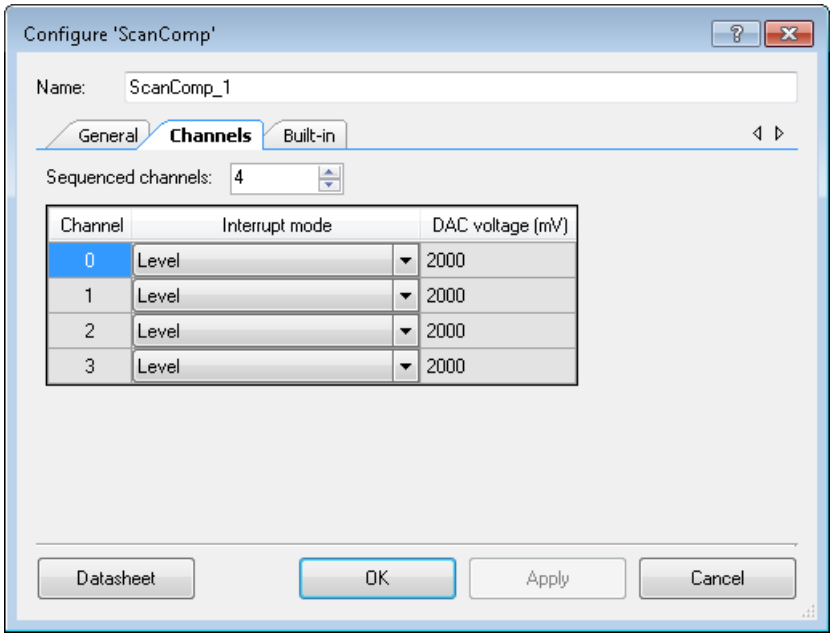
Range	Lowest Value	Highest Value	Step Size	Output Equation
0- 1.020 V	0.0 mV	1.020 V	4 mV	V_{OUT} = (value/256) × 1.024 V
0 – 4.080 V (default)	0.0 mV	4.080 V	16 mV	V_{OUT} = (value/256) × 4.096 V

VDAC Configuration - Voltage

This parameter selects the output voltage mode of the internal VDAC. Not applicable if the Negative Input parameter is set to Unique or Common.

Voltage	Description
Fixed (default)	A fixed initial value is written to the DAC. It may be changed at run-time using an API call.
Per Channel	Each channel has its own DAC output voltage that is configured in the Channels tab.

Channels Tab



Sequenced channels

This parameter selects up to 64 channels. The number of input and output channels will be limited by the hardware available in the device being used. If more channels are used than the device can support, the project will fail to build. Default setting is 4.

Interrupt mode

This parameter selects the interrupt detect mode for each channel.

Interrupt mode	Description
Rising Edge	Interrupt is generated when a rising edge on the output is detected.
Falling Edge	Interrupt is generated when a falling edge on the output is detected.
Both Edges	Interrupt is generated when either (rising or falling) edge on the output is detected.
Disable	Interrupt is disabled.
Level(default)	Interrupt is generated when the output is high.

DAC Voltage

This parameter sets the initial voltage value for each channel in mV. Not applicable if the output voltage mode of the internal VDAC is set to Fixed. Default setting is ½ of the selected DAC range (500 mV or 2000 mV).

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 "ScanComp_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 "ScanComp."

Functions

Function	Description
ScanComp_Start()	Performs all of the required initialization for the component and enables power to the block.
ScanComp_Init()	Initializes or restores the component according to the customizer settings.
ScanComp_Enable()	Activates the hardware and begins component operation.
ScanComp_Stop()	Turns off the Scanning Comparator.
ScanComp_SetSpeed()	Sets the drive power and speed.
ScanComp_SetDACRange()	Sets the DAC to a new range.
ScanComp_GetDACRange()	Gets the DAC range setting
ScanComp_SetDACVoltage()	Sets the DAC output to a new voltage.
ScanComp_GetDACVoltage()	Gets the current DAC output voltage setting
ScanComp_SetChannelDACVoltage()	Sets the DAC output for a specific channel to a new voltage.
ScanComp_GetChannelDACVoltage()	Gets the DAC output voltage for a specific channel.
ScanComp_GetCompare()	Gets the current comparison result for the selected channel.
ScanComp_GetInterruptSource()	Gets the pending interrupt requests from the selected block. Even masked interrupts are returned
ScanComp_GetInterruptSourceMasked()	Gets the pending interrupt requests from the selected block. Masked interrupts are not returned.
ScanComp_GetInterruptMask()	Gets the current interrupt mask from the selected block.
ScanComp_SetInterruptMask()	Sets the interrupt masks for the selected block.
ScanComp_Sleep()	This is the preferred API to prepare the component for low power mode operation.
ScanComp_Wakeup()	This is the preferred API to restore the component to the state when ScanComp_Sleep() was called.

Global Variables

Function	Description
ScanComp_initVar	The ScanComp_initVar variable is used to indicate initial configuration of this component. This variable is pre-pended with the component name. The variable is initialized to zero and set to 1 the first time ScanComp_Start() is called. This allows for component initialization without re-initialization in all subsequent calls to the ScanComp_Start() routine. If re-initialization of the component is required the ScanComp_Stop() routine should be called followed by the ScanComp_Init() and ScanComp_Enable().

void ScanComp_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. Power/speed is set based on the configured sample rate and the comparator response time specs, or if an external clock is used, it is set to the maximum. When called to restart the comparator following a ScanComp_Stop() call, the current component parameter settings are retained.

Parameters: None

Return Value: None

Side Effects: None

void ScanComp_Init(void)

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

Parameters: None

Return Value: None

Side Effects: None

void ScanComp_Enable(void)

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

Parameters: None

Return Value: None

Side Effects: None

void ScanComp_Stop(void)

Description: Turns off the Scanning Comparator by turning off the comparator itself and halting the muxing of inputs, and turning off the DAC if it is used.

Parameters: None

Return Value: None

Side Effects: None

void ScanComp_SetSpeed(uint8 speed)

Description: Sets the drive power and speed to one of three settings. Power/speed is set by ScanComp_Start() based on the configured sample rate and the comparator response time specs, or if an external clock is used, it is set to the maximum.

Parameters: uint8 speed: Enumerated speed mode value

Speed Setting	Notes
ScanComp_SLOWSPEED	Slow speed / Ultra low power
ScanComp_MEDSPEED	Medium speed / Medium power
ScanComp_HIGHSPEED	High speed / High power

Return Value: None

Side Effects: None

void ScanComp_SetDACRange(uint8 DACRange)

Description: Sets the DAC to a new range. Used only when the Internal DAC is selected.

Parameters: uint8 DACRange – new range to be written to the DAC

Range	Notes
ScanComp_INTERNAL_RANGE_1V	Full-scale range of 1.020 V
ScanComp_INTERNAL_RANGE_4V	Full-scale range of 4.080 V

Return Value: None

Side Effects: None

uint8 ScanComp_GetDACRange(void)

Description: Gets the DAC range setting. Used only when the Internal DAC is selected.

Parameters: None

Return Value: uint8: Present DAC range.

Range	Notes
ScanComp_INTERNAL_RANGE_1V	Full-scale range of 1.020 V
ScanComp_INTERNAL_RANGE_4V	Full-scale range of 4.080 V

Side Effects: None

void ScanComp_SetDACVoltage(uint8 DACVoltage)

Description: Sets the DAC output to a new voltage. Used only when the Internal DAC is selected.

Parameters: uint8 DACVoltage – voltage to be written to the DAC, in units depending on the selected range (4 or 16 mV per bit in 1 or 4V range, respectively)

Return Value: None

Side Effects: None

uint8 ScanComp_GetDACVoltage(void)

Description: Gets the current DAC output voltage setting. Used only when the Internal DAC is selected.

Parameters: None

Return Value: uint8: Voltage the DAC is configured for, in units depending on the selected range (4 or 16 mV per bit in 1 or 4V range, respectively)

Side Effects: None

void ScanComp_SetChannelDACVoltage(uint8 channel, uint8 DACVoltage)

Description: Sets the DAC output for a specific channel to a new voltage. Used only when the Internal DAC is selected and voltage is “Per channel”.

Parameters: uint8 channel – specifies the channel whose DAC voltage should be changed.

uint8 DACVoltage – voltage to be written to the DAC, in units depending on the selected range (4 or 16 mV per bit in 1 or 4V range, respectively)

Return Value: None

Side Effects: None

uint8 ScanComp_GetChannelDACVoltage(uint8 channel)

Description: Gets the DAC output voltage for a specific channel. Used only when the Internal DAC is selected and voltage is “Per channel”.

Parameters: uint8 channel – specifies the channel whose DAC voltage should be returned.

Return Value: uint8: Voltage the DAC is configured for, for the given channel, in units depending on the selected range (4 or 16 mV per bit in 1 or 4V range, respectively).

Side Effects: None

uint8 ScanComp_GetCompare(uint8 channel)

Description: Gets the current comparison result for the selected channel.

Parameters: uint8 channel – specifies the channel whose comparison result should be returned.

Return Value: uint8: Comparison result of the selected channel. Returns a non-zero value when the input is greater than the reference value. Otherwise, the return value is zero.

Side Effects: None

uint8 ScanComp_GetInterruptSource(uint8 inputBlock)

Description: Gets the pending interrupt requests from the selected block. This function can determine which of the channels generated an interrupt. Even masked interrupts are returned. This function clears the interrupt status for that input block.

Parameters: uint8 inputBlock – specifies the “block” of 8 or less channels whose interrupt requests should be returned.

Return Value: uint8: Bit field of interrupt sources from the selected block of 8 or less inputs. Each input has a mask value: ScanComp_INTR_MASK.

Side Effects: None

uint8 ScanComp_GetInterruptSourceMasked(uint8 inputBlock)

Description: Gets the pending interrupt requests from the selected block. This function can determine which of the channels generated an interrupt. Masked interrupts are not returned. This function clears the interrupt status.

Parameters: uint8 inputBlock – specifies the “block” of 8 or less channels whose interrupt requests should be returned.

Return Value: uint8: The interrupt source bit-fields for the selected block of 8 or less inputs. Each input has a mask value: ScanComp_INTR_MASK.

Side Effects: None

uint8 ScanComp_GetInterruptMask(uint8 inputBlock)

Description: Gets the current interrupt mask from the selected block. This function can determine which of the channels' interrupts are currently masked.

Parameters: uint8 inputBlock – specifies the "block" of 8 or less channels whose interrupt masks should be returned..

Return Value: uint8: The interrupt mask for the given block of channels. A '1' indicates that the interrupt is enabled, a '0' indicates that it is masked off. Each input has a mask value: ScanComp_INTR_MASK.

Side Effects: None

void ScanComp_SetInterruptMask(uint8 inputBlock, uint8 mask)

Description: Sets the interrupt masks for the set block of 8 or less channels.

Parameters: uint8 inputBlock – specifies the "block" of 8 or less channels whose interrupt masks will be written.

uint8 mask - interrupt mask value for the specified block of 8 or less channels. A '1' indicates that the interrupt is enabled, a '0' indicates that it is masked off.

Return Value: None

Side Effects: None

void ScanComp_Sleep(void)

Description: This is the preferred API to prepare the component for low power mode operation. The scanning comparator cannot operate in sleep mode in PSoC 3/5LP or in deep sleep mode in PSoC 4.

Parameters: None

Return Value: None

Side Effects: None

void ScanComp_Wakeup(void)

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

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 Scanning comparator component does not have any specific deviations.

This component has the following embedded components: Comparator, VDAC8, DMA, Clock, Status Register, Control Register. Refer to the corresponding component datasheet for information on their MISRA compliance and 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 Scanning Comparator component allows analog comparison of multiple sets of signals using just one hardware comparator. The trade-off is between the increased number of comparison channels versus reduced response time to a comparison transition and additional hardware usage. This component is useful when more signals must be compared than the number of comparators present in the device.

This component uses a hardware comparator (CTBm in PSoC 4) and additional hardware to mux between inputs and outputs appropriately. The inputs are automatically multiplexed at a set rate using analog multiplexers. The comparator output is latched once the inputs are stable, and an interrupt output is generated based on the configured interrupt mode.

The interrupts due to comparison results are written into status registers. The individual comparator interrupt states can be observed in firmware by polling the status registers using provided APIs. A single component-wide interrupt output signal is the ORed combination of all unmasked interrupts.

The sampled comparator outputs can be enabled for connection in digital hardware. This consumes additional hardware and is therefore optional.

The Scanning Comparator component operates in one of three compare modes which are selected via the Negative input parameter in the customizer.

- Unique – In this mode each channel has its own negative analog input that is multiplexed.
- Common – In this mode all channels use the same negative analog input.
- Internal VDAC – In this mode all channels use for their negative analog input the output from a VDAC that is configured by the component. The VDAC output value may be changed for each cannel as the channels are scanned using DMA. This mode is not available for PSoC 4 devices.

Registers

The Scanning Comparator component has several status registers that are used by the firmware APIs to monitor the status of the comparator output value and to store interrupts after edge detection. None of these registers are accessible directly by user firmware.

Resources

The Scanning Comparator component uses the following device resources:

Configuration		Resource							Pins (per External I/O)
Negative Input Mode	Seq. Ch.	Macro cells	Status Reg.	Control Reg.	Cntr7	Comp	DMA	VDAC	
Unique	7	38	2	1	1	1	–	–	Inputs: 1 [1] + 2 * Nch [2] Outputs: 2 + Nch [3]
	8	32	3						
	16	50	5						
	24	77	7						
	32	85	9						
Common	7	38	2	1	1	1	–	–	Inputs: 1 + 1 [1] + Nch Outputs: 2 + Nch
	8	32	3						

[1] Depends on the Clock source parameter. This Input is not available for internal clock.

[2] Nch – Number of sequenced channels.

[3] Depends on the Hardware output enable parameter.

Configuration		Resource							Pins (per External I/O)
Negative Input Mode	Seq. Ch.	Macro cells	Status Reg.	Control Reg.	Cntr7	Comp	DMA	VDAC	
	16	50	5						
	24	77	7						
	32	85	9						
Internal VDAC/ Fixed [4]	7	38	2	1	1	1	–	1	Inputs: 1 + Nch Outputs: 2 + Nch
	8	32	3						
	16	50	5						
	24	77	7						
	32	85	9						
Internal VDAC/Per channel [4]	7	39	2	1	1	1	1	1	Inputs: 1 + Nch Outputs: 2+ Nch
	8	41	3						
	16	67	5						
	24	101	7						
	32	117	9						

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 4 (GCC)		PSoC 5LP (GCC)	
Negative Input Mode	Seq. Channels	Flash (bytes)	RAM (bytes)	Flash (bytes)	RAM (bytes)	Flash (bytes)	RAM (bytes)
Unique / Common	4	–	–	500	7	–	–
	8	698	5	–	–	688	5
	16	709	5	–	–	756	5

[4] Not applicable for PSoC 4.

Configuration		PSoC 3 (Keil_PK51)		PSoC 4 (GCC)		PSoC 5LP (GCC)	
Negative Input Mode	Seq. Channels	Flash (bytes)	RAM (bytes)	Flash (bytes)	RAM (bytes)	Flash (bytes)	RAM (bytes)
	24	724	5	–	–	780	5
	32	737	5	–	–	804	5
Internal VDAC/Fixed	8	1055	10	–	–	1184	10
	16	1067	10	–	–	1248	10
	24	1080	10	–	–	1272	10
	32	1093	10	–	–	1300	10
Internal VDAC/Per channel	8	1279	20	–	–	1328	20
	16	1292	28	–	–	1392	28
	24	1305	36	–	–	1416	36
	32	1318	44	–	–	1448	44

Component Debug Window

The Scanning Comparator component supports the PSoC Creator component debug window. Refer to the appropriate device datasheet for a detailed description of each register. The following registers are displayed in the Scanning Comparator component debug window.

Register	Description
ScanComp_Out1_8_STATUS_REG	Status of outputs from 1 to 8 channels
ScanComp_Out9_16_STATUS_REG	Status of outputs from 9 to 16 channels
ScanComp_Out17_24_STATUS_REG	Status of outputs from 17 to 24 channels
ScanComp_Out25_32_STATUS_REG	Status of outputs from 25 to 32 channels
ScanComp_Out33_40_STATUS_REG	Status of outputs from 33 to 40 channels
ScanComp_Out41_48_STATUS_REG	Status of outputs from 41 to 48 channels
ScanComp_Out49_56_STATUS_REG	Status of outputs from 49 to 56 channels
ScanComp_Out57_64_STATUS_REG	Status of outputs from 57 to 64 channels
ScanComp_Comp_Comp_PM_ACT_CFG	Active power mode configuration register 7
ScanComp_Comp_Comp_PM_STBY_CFG	Standby power mode configuration register 7
ScanComp_Comp_Comp_TR0	Comparator trim register TR0
ScanComp_Comp_Comp_TR1	Comparator trim register TR1

Register	Description
ScanComp_Comp_Comp_CR	Comparator control register
ScanComp_Comp_Comp_SW0	Comparator analog routing register 0
ScanComp_Comp_Comp_SW1	Comparator analog routing register 1
ScanComp_Comp_Comp_SW2	Comparator analog routing register 2
ScanComp_Comp_Comp_SW3	Comparator analog routing register 3
ScanComp_Comp_Comp_SW4	Comparator analog routing register 4
ScanComp_Comp_Comp_SW5	Comparator analog routing register 5
ScanComp_Comp_Comp_SW6	Comparator analog routing register 6
ScanComp_Comp_Comp_CLK	Comparator clock control register
ScanComp_Comp_Comp_WRK	Comparator output working register
VDAC8_PM_ACT_CFG	Active power mode configuration register 8
VDAC8_PM_STBY_CFG	Standby power mode configuration register 7
VDAC8_TR	DAC trim register
VDAC8_CR0	DAC block control register 0
VDAC8_CR1	DAC block control register 1
VDAC8_SW0	DAC analog routing register 0
VDAC8_SW2	DAC analog routing register 1
VDAC8_SW3	DAC analog routing register 2
VDAC8_SW4	DAC analog routing register 3
VDAC8_STROBE	DAC strobe register
VDAC8_DATA	DAC data register

DC and AC Electrical Characteristics

The following values indicate expected performance and are based on initial characterization data.

Scanning Comparator DC Specifications

Parameter	Description	Conditions	Min	Typ	Max	Units
V_{OS}	Input offset voltage in High speed / High power mode [5]	Factory trim, V_{DDA} > 2.7 V, V_{IN} ≥ 0.5 V	–	–	10	mV
	Input offset voltage in Medium speed / Medium power mode [5]	Factory trim, V_{IN} ≥ 0.5 V	–	–	9	mV
	Input offset voltage in Slow speed / Ultra low power mode [5]	V_{DDA} ≤ 4.6 V	–	±12	–	mV
Icmp	Quiescent current		–	–	Icmp = Idac [6] + Icomp [7] + Ibase [8]	uA
CMRR	Common mode rejection ratio		30	50		dB
V_{CMP}	Input common mode voltage [5]	Ultra Low Power mode	0		V_{DDA} – 1.1	V
		Medium Power mode	0		V_{DDA} – 0.1	V
		High Power mode	0		V_{DDA} – 0.1	V

Scanning Comparator AC Specifications

Parameter	Description	Conditions	Min	Typ	Max	Units
fclk	Clock frequency	Without internal VDAC	0.001	–	10	MHz
		With internal VDAC (scale – 1V)	0.001	–	2	MHz
		With internal VDAC (scale – 4V)	0.001	–	0.5	MHz

5 The Speed Power parameter (which depends on internal clock or High power for External clock) can be observed on the component Display on Hover window.

6 Idac – VDAC operating current. This value is described in the VDAC8 component datasheet. The VDAC is not used when the negative input mode of the component is set to "Unique" or "Common".

7 Icomp – Comparator operating current. This value is described in the Comparator component datasheet.

8 Ibase – UDB block's current. The typical value of this current is 45 uA / MHz for the Scanning Comparator component.

Parameter	Description	Conditions	Min	Typ	Max	Units
T_{RESP}	Response time [5]	Ultra Low Power mode	55 / 1 [9]	–	(2*Nch)/fclk [10]	us
		Medium Power mode	0.155 / 0.4 [9]	–	(2*Nch)/fclk [10]	us
		High Power mode	0.075 / 0.15 [9]	–	(2*Nch)/fclk [10]	us

Component Changes

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

Version	Description of Changes	Reason for Changes / Impact
1.10.a	Minor datasheet edits.	
1.10	Added support for PSoC 4200-BL devices.	Updates to support PSoC 4200-BL devices.
1.0.a	Edited datasheet for Knowledge Base Article KBA94159 (www.cypress.com/go/kba94159).	Document that the component was changed, but there is no impact to designs.
1.0	New component.	

© 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.

9 Response time for PSoC 4.

10 Nch – The number of the sequenced channel; fclk = clock frequency.