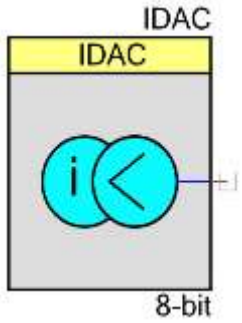
PSoC 4 Current Digital to Analog Converter (IDAC)

1.10



Features

- 7 or 8-bit resolution
- 7-bit range: 0 to 152.4 or 304.8 µA
- 8-bit range: 0 to 306 or 612 µA
- Current sink or source selectable

General Description

The IDAC component gives you a programmable current with a resolution of either 7 or 8 bits. The 8-bit ranges are approximately 612 and 306 µA and the 7-bit ranges are approximately 304.8 and 152.4 µA.

When to Use IDAC

- Resistance measurements
- Current sink or source
- Capacitance measurements other than CapSense
- Sensor current
- Temperature measurement (diode sensor)

Input/Output Connections

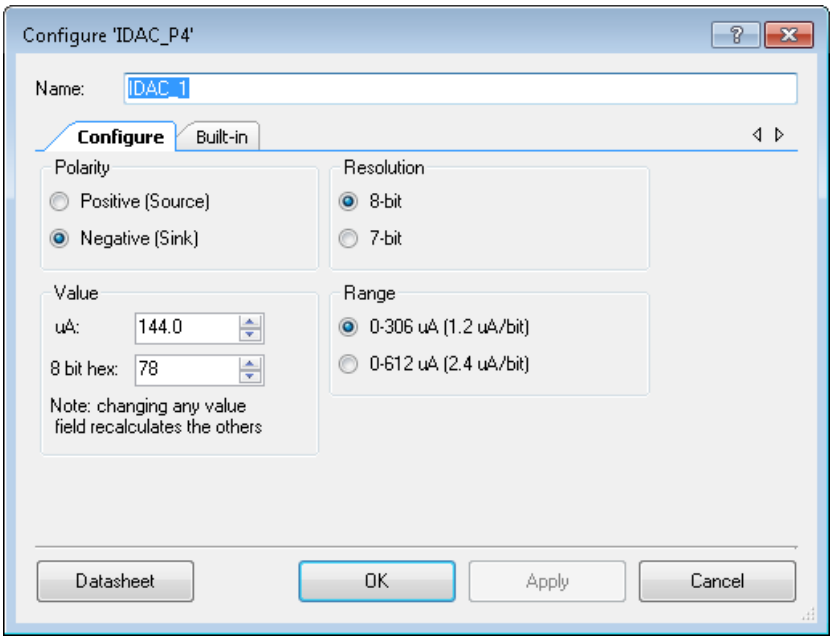
This section describes the various IDAC input and output connections.

Iout – Analog

The connection to the DAC's current source/sink.

Component Parameters

Drag the IDAC onto your design desktop and double-click it to open the Configure dialog box.



Polarity

Mode of operation. Negative/Sink (default) or Positive/Source.

Resolution

Resolution of the IDAC. 8-bit (default) or 7-bit.

Range

IDAC dynamic range:

- 8-bit resolution - 306 µA (default) or 612 µA.
- 7-bit resolution 152.4 µA or 304.8 µA

Value

IDAC hexadecimal value (default is 78).

When changing modes the code value is fixed and the current value changes. When you switch from 8-bit to 7-bit and the value exceeds the 7-bit range, the value automatically changes to 7F. When the code value changes the current value updates and vice versa.

Placement

The PSoC 4 IDACs are part of the CapSense CSD hardware block. Two IDACs are available. The 8-bit IDAC is connected to AmuxBusA and the 7-bit IDAC is connected to AmuxBusB.

Note The IDAC Component cannot be placed on the schematic if both AMUXA and AMUXB buses are used by the other blocks (for example, the CapSense Component with the enabled Shield electrode).

Resources

Resolution (bits)	Resource Type
	CSD IDAC block
7	1
8	1

Application Programming Interface

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

By default, PSoC Creator assigns the instance name "IDAC_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 table is "IDAC".

Functions

Function	Description
IDAC_Start()	Performs all of the required initialization for the component and enables power to the block.
IDAC_Stop()	Turn off the IDAC block.
IDAC_Init()	Initializes or restores the component according to the customizer Configure dialog settings.
IDAC_Enable()	Activates the hardware and begins component operation.
IDAC_SetValue()	Sets the DAC’s output value.
IDAC_Sleep()	This is the preferred API to prepare the component for sleep.
IDAC_Wakeup()	This is the preferred API to restore the component to the state when IDAC_Sleep() was called.
IDAC_SaveConfig()	Saves the configuration of the component.

Function	Description
IDAC_RestoreConfig()	Restores the configuration of the component.

void IDAC_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 configured settings. When called to restart the IDAC following a IDAC_Stop() call, the current component parameter settings are retained.

Parameters: None

Return Value: None

Side Effects: None

void IDAC_Stop(void)

Description: Turn off the IDAC block.

Parameters: None

Return Value: None

Side Effects: Does not affect the IDAC settings.

void IDAC_Init(void)

Description: Initializes or restores the component according to the customizer Configure dialog settings. It is not necessary to call IDAC_Init() because the IDAC_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 IDAC_Enable(void)

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

Parameters: None

Return Value: None

Side Effects: None

void IDAC_SetValue(uint32 value)

Description: Sets the DAC's output value. The least significant 7 or 8 bits are used depending on the resolution of the DAC. This function sets the value from 0 to 0xFF (for 8-bit IDAC) or from 0 to 0x7F (for 7-bit IDAC). The user is responsible for calculation of the correct IDAC value depending on selected resolution and range.

Parameters: (uint32) value

Return Value: None

void IDAC_Sleep(void)

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

Parameters: None

Return Value: None

Side Effects: None

void IDAC_Wakeup(void)

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

Parameters: None

Return Value: None

Side Effects: Calling the IDAC_Wakeup() function without first calling the IDAC_Sleep() or IDAC_SaveConfig() function may produce unexpected behavior.

void IDAC_SaveConfig(void)

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

Parameters: None

Return Value: None

Side Effects: None

void IDAC_RestoreConfig(void)

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

Parameters: None

Return Value: None

Side Effects: None

Global Variables

Function	Description
IDAC_initVar()	Indicates whether or not the IDAC was initialized. The variable is initialized to 0 and set to 1 the first time IDAC_Start() is called. This allows the component to restart without reinitialization after the first call to the IDAC_Start() routine. If reinitialization of the component is required, call IDAC_Init() before calling IDAC_Start(). Alternatively, you can reinitialize the IDAC by calling the IDAC_Init() and IDAC_Enable() functions.

Sample Firmware Source Code

PSoC Creator has many 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 diagram. 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.

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

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 gives you 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 IDAC component does not have any specific deviations.

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 were taken with the associated compiler configured in release mode with optimization set for size. For a specific design, you can analyze the map file generated by the compiler to determine the memory usage.

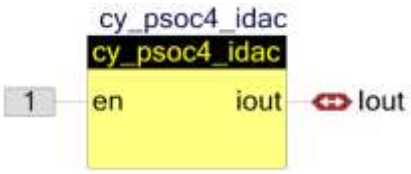
Configuration	Flash Bytes	SRAM Bytes
7 or 8-bit	472	8

Functional Description

Only one instance of each of the 7-bit and 8-bit IDAC components is available in a design. These are shared with the CapSense CSD component. If the CapSense component is present in the design it will use the 8-bit IDAC and depending on the configuration it may also use the 7-bit IDAC.

Block Diagram and Configuration

The component uses the cy_psoc4_idac primitive with hardware enable connected to Logic High. It is configured using the CSD block configuration registers.



DMA Support

The DMA component can be used to transfer data from RAM to the component registers. Refer to the applicable device datasheet to check the DMA feature availability.

Name of DMA Source/ Destination	Length	Direction	DMA Req Signal	DMA Req Type	Description
IDAC_1_IDAC_CONTROL_PTR	32 bit	Source/ Destination	N/A	N/A	This register is intended to control the IDAC settings. See the device Technical Reference Manual (TRM) for details.

Note DMA support in the IDAC_P4 component is limited due to the following reasons:

- The IDAC_1_IDAC_CONTROL register is common for two IDACs (8-bit IDAC and 7-bit IDAC).
- The IDAC_1_IDAC_CONTROL register is common for the IDAC setting and IDAC data.

Before using the DMA channel with the IDAC_P4 component, review the description of this register in the device registers Technical Reference Manual (TRM).

Registers

See the device TRM for more information about registers.

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.

IDAC DC Specifications

Parameter	Description	Conditions	Min	Typ	Max	Units
Resolution			7		8	bits
I_{out}	Output current at code = 255 Resolution = 8 bits	Range = 600uA, R_{LOAD} = 600 Ω	-	612	-	uA
		Range = 300uA, R_{LOAD} = 600 Ω	-	306	-	uA
	Output current at code = 127 Resolution = 7 bits	Range = 300uA, R_{LOAD} = 600 Ω	-	304.8	-	uA
		Range = 150uA, R_{LOAD} = 600 Ω	-	152.4	-	uA
Monotonicity			-	-	Yes	
E_{ZS}	Zero scale error	Temp = -40C to +70C	-	0	+/- 1	LSB
Eg	Gain error	Range = 600uA, 8-bit	-10	-	10	%
		Range = 300uA, 8-bit	-10	-	10	%
		Range = 300uA, 7-bit	-10	-	10	%
		Range = 150uA, 7-bit	-10	-	10	%
INL	Integral nonlinearity	Range = 600uA, 8-bit	-3	-	3	LSB
		Range = 300uA, 8-bit	-3	-	3	LSB

Parameter	Description	Conditions	Min	Typ	Max	Units
		Range = 300uA, 7-bit	-3	-	3	LSB
		Range = 150uA, 7-bit	-3	-	3	LSB
DNL	Differential nonlinearity	Range = 600uA, 8-bit	-1	-	1	LSB
		Range = 300uA, 8-bit	-1	-	1	LSB
		Range = 300uA, 7-bit	-1	-	1	LSB
		Range = 150uA, 7-bit	-1	-	1	LSB
Vcompliance	Dropout voltage	Range = 600uA, 8-bit, Source	0.8V	-	-	V
		Range = 600uA, 8-bit, Sink	0.8V	-	-	V
		Range = 300uA, 7-bit, Source	0.8V	-	-	V
		Range = 300uA, 7-bit, Sink	0.8V	-	-	V
IDAC_SET8	Settling time to 0.5 LSB for 8-bit IDAC	Full-scale transition. No external load. For PSoC 4000 / PSoC 4100 / PSoC 4200 devices only	-	-	10	µs
IDAC_SET7	Settling time to 0.5 LSB for 7-bit IDAC	Full-scale transition. No external load. For PSoC 4000 / PSoC 4100 / PSoC 4200 devices only	-	-	10	µs

Component Errata

This section lists the problems known with the IDAC_P4 Component.

Cypress ID	Component Version	Problem	Workaround
290222	1.0 to 1.10	The Vref buffer can cause noise spikes on the IDAC output when its connection is switched from one AMUX bus to another on the fly (even it is turned off). As an example, the CapSense Component powers off the Vref buffer and connects its output to the AMUX-B bus during a scan. This causes the glitch on the IDAC output pin.	Do not switch the Vref buffer from one AMUX bus to another on the fly.
290234	1.0 to 1.10	Writing a value to one IDAC on the fly can cause spikes on another IDAC.	Do not update an IDAC on the fly when two IDACs are used in a project.

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.c	Edited the datasheet.	Added a note that the IDAC Component cannot be placed on the schematic if both AMUXA and AMUXB buses are used by the other block. Added errata items 290222 and 290234.
1.10.b	Edited datasheet.	Removed note about data for PSoC 4200L devices being preliminary.
1.10.a	Edited datasheet.	Updated the IDAC DC Specifications
1.10	Added DMA support. Updated the IDAC DC Specifications	Support for PSoC 4100M/PSoC 4200M devices. Data was missing from previous revision.
1.0.a	Edited datasheet.	Updated the current ranges.
1.0	Initial release	

© Cypress Semiconductor Corporation, 2016-2017. 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, 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.