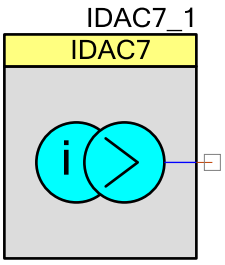
PSoC 4 Current Digital to Analog Converter (IDAC7)

1.0



Features

- Six current ranges (4.76 uA to 609 uA)
- Sink or Source current
- 7-bit resolution
- Two IDACs can be put in parallel to form an 8-bit IDAC
- Add external resistor for VDAC functionality

General Description

The IDAC7 component provides a programmable current with a resolution of 7 bits. The six overlapping ranges are from 4.76 uA to 609 uA, to allow sufficient resolution for most applications.

When to Use a IDAC7

- 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 input and output connections for the IDAC7 component.

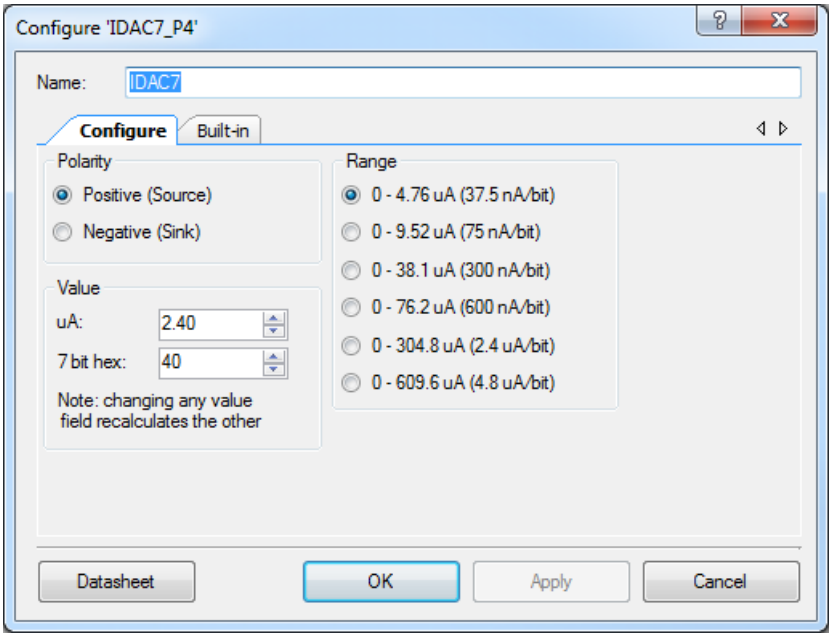
Iout – Analog

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

Component Parameters

Drag an IDAC7 component onto your design and double-click it to open the Configure dialog. This dialog has the following tabs with different parameters.

Configure Tab



Parameter Name	Description
Polarity	Selects either a current sink or source initial configuration. **Parameter Options:** • Positive (Source) (default) • Negative (Sink)
Range	Selects one of six overlapping current ranges. **Parameter Options:** • 0-4.76uA (37.5nA/bit) (default) • 0-9.52uA (75 nA/bit) • 0-38.1uA (300nA/bit) • 0-76.2uA (600nA/bit) • 0-304.8uA (2.4uA/bit) • 0-609.6uA (4.8uA/bit)
Value	Selects the initial value of the current sink or source. **Parameter Options:** • 0 to 7F (default 40)

Application Programming Interface

Application Programming Interface (API) routines allow you to configure the component using software. The following sections list and describe each function and dependencies.

By default, PSoC Creator assigns the instance name **IDAC7** 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 **IDAC7**.

General APIs

Description

General APIs are used for run-time configuration of the component during active power mode. These include, initializing, starting, stopping, reading from registers and writing to registers.

Functions

- void IDAC7_Init (void)
- void IDAC7_Enable (void)
- void IDAC7_Start (void)
- void IDAC7_Stop (void)
- void IDAC7_SetValue (uint32 current)
- void IDAC7_SetPolarity (uint32 polarity)
- void IDAC7_SetRange (uint32 range)

Function Documentation

void IDAC7_Init (void)

Initializes all initial parameters and operating modes.

void IDAC7_Enable (void)

Enables the IDAC for operation.

void IDAC7_Start (void)

Initializes all the parameters required to setup the component as defined in the customizer.

void IDAC7_Stop (void)

The Stop is not required.

void IDAC7_SetValue (uint32 current)

Sets the IDAC current to the new value.

Parameters:

uint32	current: The current value

- Valid range : [0 - 127]

void IDAC7_SetPolarity (uint32 polarity)

Sets polarity to either sink or source.

Parameters:

uint32	polarity: Current polarity

- IDAC7_POL_SOURCE : Source polarity
- IDAC7_POL_SINK : Sink polarity

void IDAC7_SetRange (uint32 range)

Sets the IDAC range to one of the six ranges.

Parameters:

uint32	range: Current range

- IDAC7_RNG_4_76UA : 37.5 nA/bit current range
- IDAC7_RNG_9_52UA : 75 nA/bit current range
- IDAC7_RNG_38_1UA : 300 nA/bit current range
- IDAC7_RNG_76_2UA : 600 nA/bit current range
- IDAC7_RNG_304_8UA : 2.4 uA/bit current range
- IDAC7_RNG_609_6UA : 4.8 uA/bit current range

Power Management APIs

Description

Power management APIs perform the necessary configurations to the components to prepare it for entering low power modes.

These APIs must be used if the intent is to put the chip to sleep, and then to continue component operation when it comes back to active power mode.

This component does not stop the CSD IP block. One possible way to turn off the entire CSD block before sleep is to use a specific define (IDAC7_CSD_CONFIG_ENABLE) for the m0s8csdv2 IP block control register (IDAC7_CSD_CONTROL_REG):

```
IDAC7_CSD_CONTROL_REG &= ~IDAC7_CSD_CONFIG_ENABLE
```

Functions

- void IDAC7_Sleep (void)
- void IDAC7_Wakeup (void)

Function Documentation

void IDAC7_Sleep (void)

Stores all volatile settings and powers down the IDAC7.

void IDAC7_Wakeup (void)

Restores settings and power saved by the Sleep function after wakeup.

Global Variables

The following global variables are used in the component.

- uint32 IDAC7_initVar – This variable is used to indicate the initial configuration of this component. The variable is initialized to zero and set to 1 the first time IDAC7_Start() is called. This allows the component initialization without re-initialization in all subsequent calls to the IDAC7_Start() routine.

Code Examples and Application Notes

This section lists the projects that demonstrate the use of the component.

Code Examples

PSoC Creator provides access to code examples in the Code Example 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 "Code Example" topic in the PSoC Creator Help for more information.

There are also numerous code examples that include schematics and example code available online at the Cypress Code Examples web page. Examples that use this component include:

- CE204022 - IDAC7_Sawtooth.

API Memory Usage

Shows the Flash, SRAM and stack usage of the component.

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 an 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.

PSoC 4 (GCC)

Configuration	PSoC 4000S / PSoC 4100S / PSoC Analog Coprocessor	
	Flash Bytes	SRAM Bytes
All	226	4

Functional Description

The 7-bit current DAC (IDAC7) is based on the CapSense block (CSDV2).

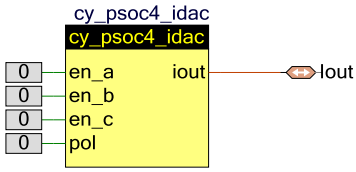
Definitions

- DAC – Digital to Analog Converter
- IDAC – Current Digital to Analog Converter

- VDAC – Voltage Digital to Analog Converter
- CSDV2 – Capacitive Sigma Delta version 2

Block Diagram and Configuration

A simplified diagram of the IDAC7 hardware is shown below:



The IDAC7 component uses cy_psoc4_idac primitive. It is configured using the CSDV2 block configuration registers only.

DMA Support

The DMA component can be used to transfer data from the component registers to RAM or another component.

You can use the DMA Wizard to configure DMA operation as follows:

Name of DMA Source/ Destination in the DMA Wizard	Length	Direction	DMA Req Signal	DMA Req Type	Description
IDAC7_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 IDAC7 component is limited due to the following reasons:

- The IDAC7_IDAC_CONTROL register is common for the IDAC setting and IDAC current value.

Before using the DMA channel with the IDAC7 component, review the description of this register in the device registers Technical Reference Manual (TRM).

Placement

The PSoC 4 IDACs are part of the CapSense CSDV2 hardware block. Two of the 7-bit IDACs are available.

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 IDAC7 component does not have any specific deviations.

Registers

For fixed-function blocks, refer to the device *Technical Reference Manual (TRM)* for more information about the registers.

Component Debug Window

PSoC Creator allows you to view debug information about components in your design. Each component window lists the memory and registers for the instance. For detailed hardware registers descriptions, refer to the appropriate device technical reference manual.

To open the Component Debug window:

1. Make sure the debugger is running or in break mode.
2. Choose **Windows > Components…** from the **Debug** menu.
3. In the Component Window Selector dialog, select the component instances to view and click **OK**.

The selected Component Debug window(s) will open within the debugger framework. Refer to the "Component Debug Window" topic in the PSoC Creator Help for more information.

Resources

The IDAC7 uses the following device resources:

- CSDV2 IDAC block

DC and AC Electrical Characteristics - TBD

Specifications are valid for -40° C = TA = 85° C and TJ = 100° C, except where noted.

Note Final characterization data for PSoC 4000S, PSoC 4100S and PSoC Analog Coprocessor devices is not available at this time. Once the data is available, the component datasheet will be updated on the Cypress web site.

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	Edited datasheet.	Final characterization data for PSoC 4000S, PSoC 4100S and PSoC Analog Coprocessor devices is not available at this time. Once the data is available, the component datasheet will be updated on the Cypress web site.
1.0	Initial release	

© Cypress Semiconductor Corporation, 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.