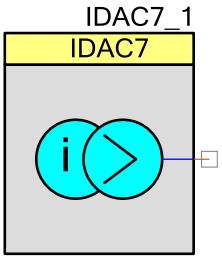
PSoC 6 Current Digital to Analog Converter (IDAC7)

1.0



Features

- Six current ranges (4.96 uA to 635 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 4.96 uA to 635 uA, which allows sufficient resolution for most applications.

When to Use a IDAC7 Component

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

Definitions

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

Quick Start

1. Drag an IDAC7 Component from the Component Catalog *Cypress/Analog/DAC* folder onto your schematic (the placed instance takes the name IDAC7_1).
2. Double-click to open the Configure dialog, and set the desired range and current parameters.
3. Drag an Analog Pin from the Component Catalog *Cypress/Ports and Pins* folder onto your schematic (the placed instance takes the name Pin_1). Connect it to the IDAC7_1 output.
4. Build the project in order to verify the correctness of your design and generate the configuration data for the IDAC7_1 instance.
5. In *main.c*, initialize the peripheral and start the application:

```
IDAC7_1_Start();
```

6. Build and program the device.

Input/Output Connections

Iout – Analog

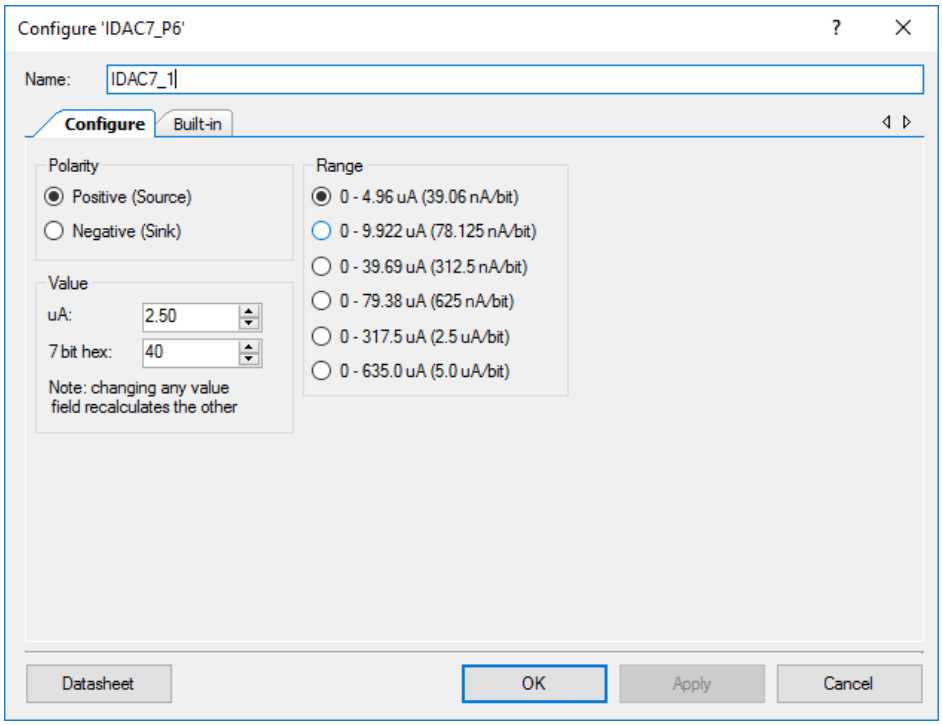
This is the connection to the DAC's current source/sink.

Component Parameters

The IDAC7 Component Configure dialog allows you to edit the configuration parameters for the Component instance.

General Tab

This tab contains the Component parameters used in the general peripheral initialization settings.



Name	Description
Polarity	Selects either a current sink or source initial configuration. **Parameter Options:** • Positive (Source) (default) • Negative (Sink) **Note:** If Polarity is Negative, the current value is correct for output voltages between Vdda and 0.6V. If Polarity is Positive, the current value is correct for output voltages between Vdda-0.6V and ground (0V).
Range	Selects one of six overlapping current ranges. **Parameter Options:** • 0-4.96 uA (39.06 nA/bit) (default) • 0-9.922 uA (78.125 nA/bit) • 0-39.69 uA (312.5 nA/bit) • 0-79.38 uA (625 nA/bit) • 0-317.5 uA (2.5 uA/bit) • 0-635.0 uA (5.0 uA/bit)

Name	Description
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.

By default, PSoC Creator assigns the instance name IDAC7_1 to the first instance of a Component in a given design. It can be renamed 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
  General APIs are used for run-time configuration of the Component during active power mode.
- Power Management APIs
  Power management APIs perform the necessary configurations to the Components to prepare it for entering low power modes.

General APIs

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 parameters required to setup the Component as defined in the customizer.

Global Variables

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.

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: Current value

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 six ranges.

Parameters:

uint32	range: Current range – IDAC7_RNG_4_96UA : 39.06 nA/bit current range – IDAC7_RNG_9_922UA : 78.125 nA/bit current range – IDAC7_RNG_39_69UA : 312.5 nA/bit current range – IDAC7_RNG_79_38UA : 625 nA/bit current range – IDAC7_RNG_317_5UA : 2.5 uA/bit current range – IDAC7_RNG_635_0UA : 5.0 uA/bit current range

Power Management APIs

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, then to continue the Component operation when it comes back to active power mode.

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 after wakeup that were saved by the Sleep function.

Global Variables

The IDAC7 Component populates the following peripheral initialization data structure(s). The generated code is placed in C source and header files that are named after the instance of the Component (e.g. IDAC7_1.c). Each variable is also prefixed with the instance name of the Component.

uint8 IDAC7_initVar

The instance-specific configuration structure. This should be used in the Init() function.

Code Examples and Application Notes

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.

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 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 6 (GCC)

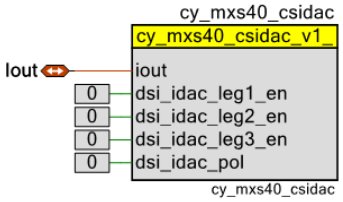
Configuration	PSoC 6	
	Flash Bytes	SRAM Bytes
All	252	4

Functional Description

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

Block Diagram and Configuration

A simplified diagram of the IDAC7 hardware is shown below:



The IDAC7 Component uses cy_mxs40_csidac primitive. It is configured using the CSDV2 block configuration registers only.

DMA Support

The IDAC7 Component supports Direct Memory Access (DMA) transfers. The Component may transfer to/from the following sources.

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 6 IDACs are part of the CapSense CSDV2 hardware block. Two 7-bit IDACs are available.

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).

Industry Standards

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. Refer to PSoC Creator Help > Building a PSoC Creator Project > Generated Files (PSoC 6) for information on MISRA compliance and deviations for files generated by PSoC Creator.

The IDAC7 component has the following specific deviations:

Rule	Rule Class (Required/Advisory)	Rule Description	Description of Deviation(s)
1.1	R	Number of macro definitions exceeds 1024 - program does not conform strictly to ISO:C90.	The IDAC component generates macros for the registers. For some configurations numbers of macros can exceeds 1024 (ISO90).
10.1	A	A non-constant expression of 'essentially unsigned' type (%1s) is being converted to signed type, '%2s' on assignment.	The cast from unsigned int does not have any unintended effect, as it is a consequence of the definition of a structure based on hardware registers.

Registers

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

Resources

The IDAC7 Component uses the CSDV2 IDAC peripheral block.

DC and AC Electrical Characteristics

Note Final characterization data for PSoC 6 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.b	Minor datasheet edits.	
1.0.a	Edited 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 blocks.
1.0	Initial Version	

© Cypress Semiconductor Corporation, 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.