Current DAC Datasheet IDAC V 1.00

Copyright © 2013 Cypress Semiconductor Corporation. All Rights Reserved.

Resources	PSoC® Blocks			API Memory (Bytes)		Pins (per External I/O)
	Digital	Analog CT	Analog SC	Flash	RAM	
Supported Devices: CY8C28x13, CY8C28x33, CY8C28x43, CY8C28x45, CY8C28x52, CY8C21x45, CY8C22x45, CY8C24x93						
CY8C28x13, CY8C28x33, CY8C28x43, CY8C28x45, CY8C28x52, CY8C21x45, CY8C22x45	0	0	1	251	0	1 or 2
CY8C24x93	0	0	0	182	0	1

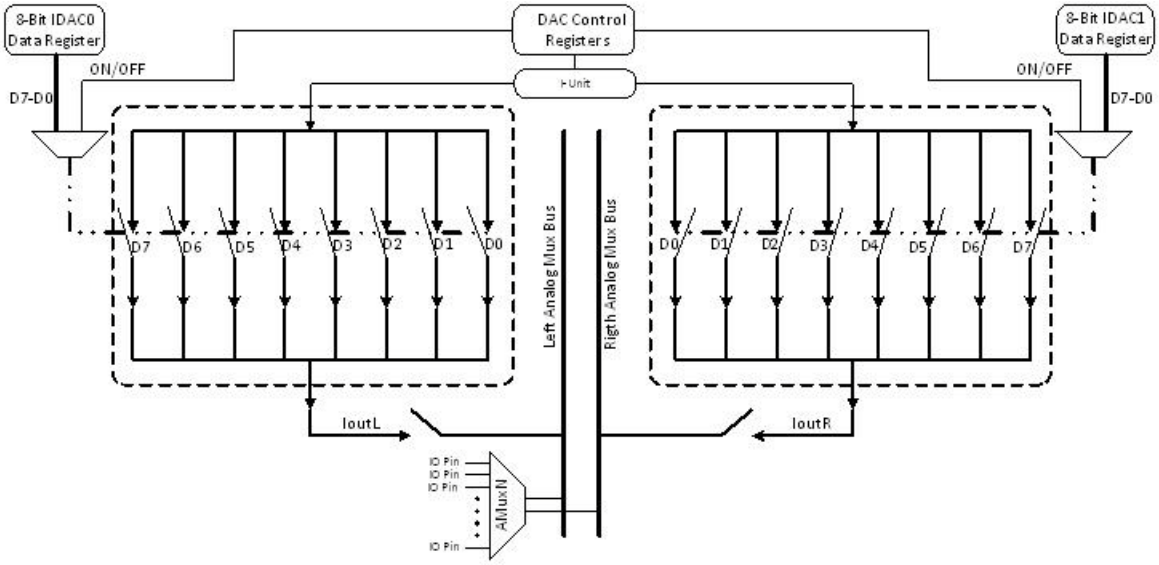
Features and Overview

- 8-bit resolution
- Current output
- Two channels for the CY8C28xxx, CY8C21x45, and CY8C22x45 device families
- One channel for the CY8C24x93 device family
- Three ranges of output current for the CY8C28xxx, CY8C21x45, and CY8C22x45 device families
- Four ranges of output current for the CY8C24x93 device family of devices
- Possibility to set the output current in µA

Functional Description

The IDAC User Module provides a current source to the analog mux bus. The IDAC User Module can be used with the existing AMuxN User Module, which connects the analog mux bus to any GPIO pin. This user module can be used in any application where a current source is required.

Figure 1. CY8C28xxx, CY8C21x45, and CY8C22x45 IDAC Block Diagram



In the CY8C28xxx, CY8C21x45, and CY8C22x45 device families, the dual channel IDAC can provide two independent current sources to any analog mux buses. The IDAC output current is based on the input current reference IUNIT. The IUNIT is generated from the PLL and can be trimmed in the IDAC_CR1 register. These two current sources provide the current in the range from 0 to 637.5 µA with 255 steps (8-bit IDAC data registers - IDACL_D and IDACR_D), and they are connected to the left analog mux bus (AmuxBus0) and the right analog mux bus (AmuxBus1) independently. In the CY8C28xxx, CY8C21x45, and CY8C22x45 device families, three ranges of current sources are implemented. Each range can vary in 255 steps. The current range depends on the IRANGE bit in the IDAC_CR0 register and the Double_Current bit in the IDAC_CR1 register.

Table 1. Current Range Values for IDAC_CR0 and IDAC_CR1 Registers

IDAC_CR1[0]. Double_Current	IDAC_CR0[3]. IRANGE	Current Range
0 (default)	0 (default)	Reserved
1	0	Maximum 91.03 µA
0	1	Maximum 318.75 µA
1	1	Maximum 637.5 µA

The functional block diagram of the IDAC User Module for the CY8C24x93 family of devices is as follows.

Figure 2. CY8C24x93 IDAC Block Diagram

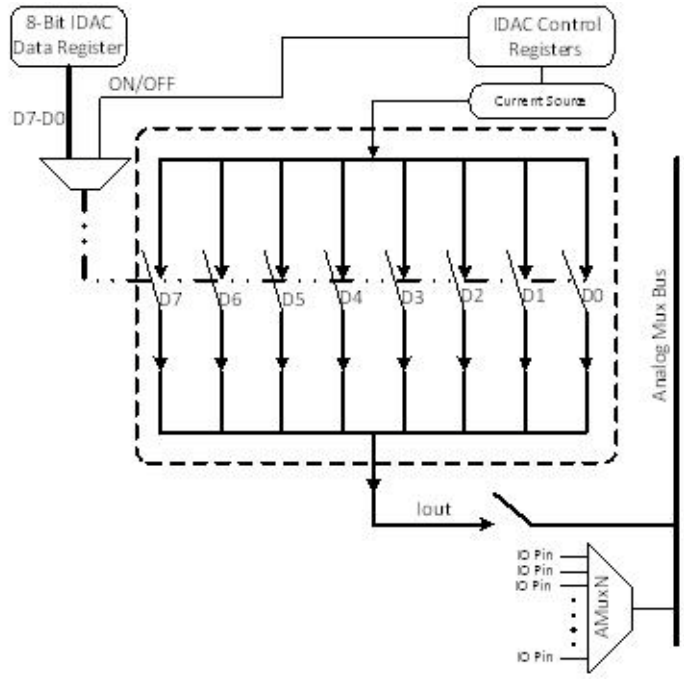


Table 2. Current Range Values for CS_CR2 Register

Supported Devices			
CY8C24193-24LQXI, CY8C24493-24LTXI		CY8C24093-24LKXI,CY8C24293-24LQXI,CY8C24393-24LQXI,CY8C24693-24LQXI	
CS_CR2[6-7]	Current Range	CS_CR2[6]	Current Range
0	Maximum 64 µA	0	Maximum 256 µA
1	Maximum 128 µA	1	Maximum 512 µA
2	Maximum 256 µA	–	–
3	Maximum 512 µA	–	–

In this family of devices, the internal current DAC provides the current source to the analog mux bus. In the CY8C24x93 family of devices, two ranges of current source are implemented. Each range can vary in 255 steps (8-Bit IDAC data register - IDAC_CODE). The current range depends on the IRANGE bit in the CS_CR2 register.

The IDAC User Module can be used with the existing AMuxN User Module, which connects the analog mux bus to any GPIO pin. You can set the MUX_CRx register to connect any pin you desire on the AMUX to route the IDAC output out.

DC and AC Electrical Characteristics

The following table lists maximum and minimum specifications for the voltage and temperature ranges: 4.75 V to 5.25 V and –40 °C < T_A < 85 °C or 3.0 V to 3.6 V and –40 °C < T_A < 85 °C, respectively. For

CY8C24x93, the voltage and temperature ranges are –40 °C < T_A < 85 °C, 1.71 V < V_{DD} < 5.5 V. Typical parameters apply to 5 V and 3.3 V at 25 °C and are for design guidance only.

Table 3. IDAC DC and AC Electrical Characteristics

Symbol	Description	Min	Typ	Max	Units
FSE1	Full Scale Error	–	1	–	% of FS
FSE2	Full Scale Error	–	1	–	
FSE3[1]	Full Scale Error	–	1	–	
DNL	DNL Error	–	2	–	LSB
INL	INL Error	–	2	–	
VOH	Head Room Voltage - Maximum output voltage that IDAC can drive	–	1	–	V Less than supply voltage
Monotonic	Monotonicity of the DAC	–	–	–	

Parameters and Resources

Full Scale Range

This parameter sets the full scale current range for both the IDAC channels. The range for the CY8C28xxx, CY8C21x45, and CY8C22x45 families is 91.03 µA, 318.75 µA, and 637.5 µA, while the default is 637.5 µA. For the CY8C24193-24LQXI and CY8C24493-24LTXI devices (CY8C24x93 family), the range is 64 µA, 128 µA, 256 µA, and 512 µA, while the default is 512 µA. For the CY8C24093-24LKXI, CY8C24293-24LQXI, CY8C24393-24LQXI, and CY8C24693-24LQXI devices (CY8C24x93 family), the range is 256 µA and 512 µA, while the default is 512 µA. The affected registers in the CY8C28xxx, CY8C21x45, and CY8C22x45 families are IDAC_CR1 - bit[0] Double_Current and IDAC_CR0 - bit[3] IRANGE. In the CY8C24x93 family, the affected register is CS_CR2 - bits [6;7].

Used AMUX Bus

This parameter sets the analog bus to be used for the IDAC User Module. The range varies from Left for the single-channel IDAC that uses the left analog bus and Right for the single-channel IDAC which uses the right analog bus, or Both for the dual-channel IDAC, which uses two analog buses (each channel is independent). The default setting is Left. The affected registers are IDAC_CR1 - bit [7] EN1 and IDAC_CR0 - bit [0] EN0. This parameter is available only for CY8C28xxx, CY8C21x45, and CY8C22x45 families.

Left IDAC Value

This parameter sets the initial value for the left channel of the IDAC User Module. The range is 0 to 255 and the default setting is 50. The affected register in the CY8C28xxx, CY8C21x45, and CY8C22x45 families are IDAC0_D - bits [7:0] IDAC0. This parameter is available only for CY8C28xxx, CY8C21x45, and CY8C22x45 families.

Right IDAC Value

This parameter sets the initial value for the right channel of the IDAC User Module. The range is 0 to 255 and the default setting is 50. The affected register in the CY8C28xxx, CY8C21x45, and CY8C22x45 families are IDAC1_D - bits [7:0] IDAC1. This parameter is available only for CY8C28xxx, CY8C21x45, and CY8C22x45 families.

IDAC Value

This parameter sets the initial value for the IDAC User Module. The range is 0 to 255 and the default setting is 50. In the CY8C24x93 family, the affected registers are IDAC1_CODE - bits [7:0] IDAC_CODE. This parameter is available only for CY8C24x93 family.

Application Programming Interface

The Application Programming Interface (API) routines are provided as part of the user module to allow the designer to deal with the module at a higher level. This section specifies the interface to each function together with related constants provided by the “include" files.

Note

In this, as in all user module APIs, the values of the A and X register may be altered by calling an API function. It is the responsibility of the calling function to preserve the values of A and X prior to the call if those values are required after the call. This “registers are volatile" policy was selected for efficiency reasons and has been in force since version 1.0 of PSoC Designer. The C compiler automatically takes care of this requirement. Assembly language programmers must ensure their code observes the policy, too. Though some user module API function may leave A and X unchanged, there is no guarantee they will do so in the future.

For Large Memory Model devices, it is also the caller's responsibility to preserve any value in the CUR_PP, IDX_PP, MVR_PP, and MVW_PP registers. Even though some of these registers may not be modified now, there is no guarantee that will remain the case in future releases.

Entry points are provided to initialize the DAC6 User Module, write updated values, and disable the user module.

IDAC_Start

Description:

Starts the IDAC with the default value.

C Prototype:

```
void  IDAC_Start(void)
```

Assembler:

```
lcall  IDAC_Start
```

Parameters:

NA

Return Value:

None

IDAC_Stop

Description:

Stops the IDAC.

C Prototype:

```
void IDAC_Stop(void)
```

Assembler:

```
lcall IDAC_Stop
```

Parameters:

NA.

Return Values:

None

IDAC_SetValue

Description:

Sets the IDAC output current value for the selected channel.

C Prototypes (for CY8C28xxx, CY8C21x45, CY8C22x45):

```
void IDAC_SetValue(BYTE bChannel, BYTE bValue)
```

C Prototypes (for CY8C24x93):

```
void IDAC_SetValue(BYTE bValue)
```

Assembly (for CY8C28xxx, CY8C21x45, CY8C22x45):

```
mov A, bChannel
mov   X, bValue
lcall  IDAC_SetValue
```

Assembly (for CY8C23x93):

```
mov   A, bValue
lcall  IDAC_SetValue
```

Parameters:

bChannel: Number of the current channel. For IDAC_LEFT, the value is 0 and for IDAC_RIGHT, the value is 1 (only for the CY8C28xxx, CY8C21x45, CY8C22x45).

bValue: Output current value (0-255) - for both the CY8C28xxx, CY8C21x45, CY8C22x45, and CY8C24x93 families.

Return Values:

None

IDAC_SetRange

Description:

Sets the IDAC output current range for both channels.

C Prototypes:

```
void IDAC_SetRange(BYTE bRange)
```

Assembler:

```
mov   A, bRange
lcall  IDAC_SetRange
```

Parameters:

bRange: Decides the output current range of the IDAC, which can be varied in 255 steps. These are the values for the CY8C28xxx, CY8C21x45, and CY8C22x45 families:

Symbolic Name	Value
IDAC_RANGE_91UA	0x01
IDAC_RANGE_318UA	0x08
IDAC_RANGE_637UA	0x09

These are the values for the CY8C4x93 family:

Symbolic Name	Supported Devices	
	CY8C24193-24LQXI, CY8C24493-24LTXI	CY8C24093-24LKXI, CY8C24293-24LQXI, CY8C24393-24LQXI, CY8C24693-24LQXI
	Value	Value
IDAC_RANGE_64UA	0x00	Not available
IDAC_RANGE_128UA	0x40	Not available
IDAC_RANGE_256UA	0x80	0x00
IDAC_RANGE_512UA	0xC0	0x40

Return Values:

None

IDAC_SetCurrent

Description:

Sets the output current value for the IDAC. Input argument is in uA.

C Prototype (for CY8C28xxx, CY8C21x45, CY8C22x45):

```
void IDAC_SetCurrent(BYTE bChannel, WORD wCurrentInUA)
```

C Prototype (for CY8C4x93):

```
void IDAC_SetCurrent(WORD wCurrentInUA)
```

Assembly (for CY8C28xxx, CY8C21x45, CY8C22x45):

```
mov A, >wCurrentInUA
push A
mov A, <wCurrentInUA
push A
mov A, bChannel
push A
lcall IDAC_SetCurrent
add SP, -3
```

Assembly (for CY8C4x93):

```
mov A, [wCurrentInUA]
```

```
mov X, [wCurrentInUA+1]
lcall IDAC_SetCurrent
```

Parameters:

bChannel: Number of the current channel. The value for IDAC_LEF is 0 and for IDAC_RIGHT, it is 1.

wCurrentInUA - Output current value in uA; value up to 637 uA (max for CY8C21x45, CY8C22x45, CY8C28xxx)

wCurrentInUA: Output current value in uA; value up 512 uA (max for CY8C24x93)

Return Value:

None

Sample Firmware Source Code

The following is the C sample code for the CY8C28xxx, CY8C21x45, CY8C22x45 families.

```
//--------------------------------------------------------------------
// C main line
//--------------------------------------------------------------------

#include <m8c.h>        // Part specific constants and macros
#include "PSoCAPI.h"    // PSoC API definitions for all User Modules

void main(void)
{
    // M8C_EnableGInt;   // Enable Global Interrupts
    MUX_CR0 |= 0x02;     // Connect P0[1] to Analog Mux Bus
    IDAC_Start();        // Start IDAC with default range and value
    IDAC_SetRange(IDAC_RANGE_637UA); // set IDAC output current range
    IDAC_SetValue(IDAC_LEFT, 255); // set IDAC output current value
    for(;;)
    {
        // Add user code here to use or display result
    }
}
```

The following is the C sample code for CY8C24x93 family.

```
//--------------------------------------------------------------------
// C main line
//--------------------------------------------------------------------

#include <m8c.h>        // Part specific constants and macros
#include "PSoCAPI.h"    // PSoC API definitions for all User Modules

void main(void)
{
    // M8C_EnableGInt;   // Enable Global Interrupts
    MUX_CR0 |= 0x02;     // Connect P0[1] to Analog Mux Bus
    IDAC_Start();        // Start IDAC with default range and value
    IDAC_SetRange(IDAC_RANGE_512UA); // set IDAC output current range
    IDAC_SetValue(255); // set IDAC output current value
    for(;;)
    {
        // Add user code here to use or display result
```

```
    }
}
```

The following is the ASM sample code. for CY8C28xxx, CY8C21x45, CY8C22x45 families.

```
;-----------------------------------------------------------------
; Assembly main line
;-----------------------------------------------------------------

include "m8c.inc"    ; Part specific constants and macros
include "memory.inc" ; Constants & macros for SMM/LMM and Compiler
include "PSoCAPI.inc"; PSoC API definitions for all User Modules

export _main

area text(ROM,REL)
_main:

    ;M8C_EnableGInt   ; Enable Global Interrupts
    M8C_SetBank1
    or    reg[MUX_CR0], 0x02 ; Connect P0[1] to Analog Mux Bus
    M8C_SetBank0
    lcall IDAC_Start  ; Start IDAC with default range and value
    mov   A, IDAC_RANGE_637UA
    lcall IDAC_SetRange   ; set IDAC output current range
    mov   A, IDAC_LEFT
    mov   X, 255
    lcall IDAC_SetValue   ; set IDAC output current value

.terminate:
    ; Add user code here to use or display result
    jmp .terminate
```

The following is the ASM sample code for CY8C24x93 family.

```
;-----------------------------------------------------------------
; Assembly main line
;-----------------------------------------------------------------

include "m8c.inc"    ; Part specific constants and macros
include "memory.inc" ; Constants & macros for SMM/LMM and Compiler
include "PSoCAPI.inc"; PSoC API definitions for all User Modules

export _main

area text(ROM,REL)
_main:

    ;M8C_EnableGInt   ; Enable Global Interrupts
    M8C_SetBank1
    or    reg[MUX_CR0], 0x02 ; Connect P0[1] to Analog Mux Bus
    M8C_SetBank0
    lcall IDAC_Start  ; Start IDAC with default range and value
    mov   A, IDAC_RANGE_512UA
    lcall IDAC_SetRange   ; set IDAC output current range
    mov   A, 255
```

```
    lcall IDAC_SetValue   ; set IDAC output current value

.terminate:
    ; Add user code here to use or display result
    jmp .terminate
```

Configuration Registers

The IDAC registers used to configure this user module are described in this section. The following registers are used by the IDAC UM in CY8C28xxx, CY8C21x45, and CY8C22x45 families of devices.

Table 4. IDAC Control Register 1 - IDAC_CR1

Bit	7	6	5	4	3	2	1	0
Value	EN1	–	–	–				Double_Current

Table 5. IDAC Control Register 0 - IDAC_CR0

Bit	7	6	5	4	3	2	1	0
Value	–	–	–		IRANGE	–		EN0

Double_Current and IRANGE - These bits set the full scale current range for IDAC.

EN0 - This bit controls whether or not the left channel IDAC mode is enabled.

EN1 - This bit controls whether or not the right channel IDAC mode is enabled.

Table 6. IDAC Data Registers - IDACx_D

Bit	7	6	5	4	3	2	1	0
Value	IDACx[7:0]							

IDACx - This is an 8-bit value that selects the number of current units that combine to form the DAC current. This current then drives the analog mux bus when the DAC mode is enabled in the IDAC_CRx register. For example, a setting of 80h means that the charging current is 128 current units. The current unit size depends on the range setting in the IDAC_CRx register.

The following registers are used by IDAC User Module in the CY8C24x93 family of devices:

Table 7. CapSense Control Register 2 - CS_CR2

Bit	7	6	5	4	3	2	1	0
Value	–	IRANGE	–	IDAC_EN	–	–	–	–

IRANGE - This bit sets the full scale current range.

IDAC_EN - This bit provides connection of the IDAC to the analog global bus.

Table 8. Current DAC Data Register - IDAC1_CODE

Bit	7	6	5	4	3	2	1	0
Value	IDAC1_CODE[7:0]							

IDAC1_CODE - This is an 8-bit value that selects the number of current units that combine to form the IDAC current. This current then drives the analog mux bus when IDAC mode is enabled. For example, a setting of 80h means that the charging current is 128 current units.

Version History

Version	Originator	Description
1.00	HPHA	Initial version.

Note PSoC Designer 5.1 introduces a Version History in all user module datasheets. This section documents high level descriptions of the differences between the current and previous user module versions.

Copyright © 2013 Cypress Semiconductor Corporation. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

PSoC Designer™ and Programmable System-on-Chip™ are trademarks and PSoC® is a registered trademark of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are property of the respective corporations.

Any Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.