Analog Engineer's Circuit

Smart DAC LED Biasing Circuit with Low-Power Sleep Mode

Smart DAC

Katlynne Jones

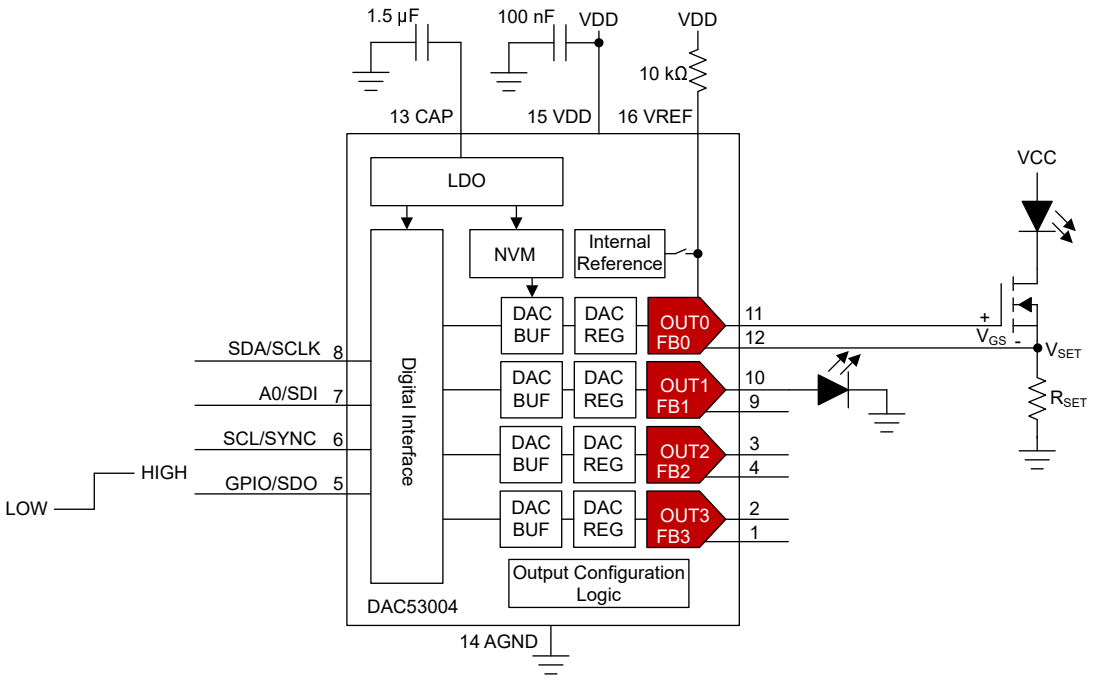
Design Objective

Key Input Parameter	Key Output Signal	Recommended Device
SPI or I^2C communication to program DAC codes 0x000 to 0xFFF, GPI trigger	0 A to 250 µA and 0 mA to 20 mA LED current	DAC53004 (10-bit), DAC63004 (12-bit)

Objective: *Bias an LED using a low-power smart DAC*

Design Description

This design uses a four-channel buffered voltage or current output low-power smart DAC such as the DAC53004 or DAC63004 (DACx3004) to bias an LED. The smart DAC can be connected in a force-sense configuration with a MOSFET in LED biasing applications needing a few milliamps of current. The DAC will set the source current of the MOSFET and control the amount of current through the LED, connected between the power supply and drain of the MOSFET, by varying the gate voltage. The DAC can be used in current output mode to drive the LED directly with up to 250 µA for low-current LED biasing applications. The V_{FB} pin of the DACx3004 compensates for the gate-to-source voltage (V_{GS}) drop and the drift of the MOSFET when using the MOSFET configuration. The DACx3004s have a general-purpose input-output (GPIO) pin that allows the DAC to enter and exit deep-sleep mode. All register settings can be saved using the non-volatile memory (NVM) on the smart DAC, meaning that the device can be used without a processor, even after a power cycle. This circuit can be used in applications such as barcode scanners, barcode readers, currency counters, POS printers, optical modules, and appliance lighting.



Design Notes

1. The *DACx3004 12-Bit and 10-Bit, Ultra-Low-Power, Quad Voltage and Current Output Smart DACs With Auto-Detected I2C, PMBus™, or SPI* data sheet recommends using a 100-nF decoupling capacitor for the VDD pin and a 1.5-µF or greater bypass capacitor for the CAP pin. The CAP pin is connected to the internal LDO. Place these capacitors close to the device pins.
2. When the external reference is not used, the VREF pin should be connected to VDD through a pullup resistor.
3. The example circuit shows two methods for controlling the LED current. Current can be set via an R_{SET} resistor and varying the gate voltage of an external MOSFET with the DACx3004 output, or the LED current can be set using the current output mode of the DACx3004.
   a. To adjust the LED current with the external MOSFET, select an R_{SET} resistor and vary the gate voltage with the DAC output. R_{SET} is calculated by:

   $$R_{SET} = \frac{V_{SET}}{I_{LED}}$$

   If the DAC output voltage range is chosen to be 0 V to 2.5 V, and the required LED current range is 0 mA to 20 mA, R_{SET} is calculated to be:

   $$R_{SET} = \frac{2.5\ V}{20\ mA} = 125\ \Omega$$

   The DAC codes can be calculated by:

   $$Code = \frac{V_{DAC}}{V_{REF}} \times 1024$$

   In this design, the internal reference is powered down to limit the power consumption. This configuration compensates the gate-source voltage drop caused by temperature, drain current, and aging of the MOSFET. Assuming a typical gate-source voltage of 1.2 V and a power supply headroom of 200 mV, the VDD for the DAC must be a minimum of (2.5 V + 1.2 V + 200 mV) = 3.9 V. If a 5-V VDD is used as the reference, the high and low DAC values for the 10-bit DAC53004 become:

   $$Code = \frac{2.5\ V}{5\ V} \times 1024 = 512\ d$$

   $$Code = \frac{0\ V}{5\ V} \times 1024 = 0\ d$$

   where
   - d = decimal

   b. The DAC can be used in current output mode to drive the LED directly with up to 250 µA. With the ±250 µA range selected, the 8-bit current DAC53004 codes are calculated by:

   $$Code = \frac{(I_{DAC} - I_{MIN}) \times 256}{I_{MAX} - I_{MIN}}$$

   The high and low DAC53004 codes become:

   $$Code = \frac{(250\ \mu A + 250\ \mu A) \times 256}{250\ \mu A + 250\ \mu A} = 256\ d$$

   $$Code = \frac{(0\ \mu A + 250\ \mu A) \times 256}{250\ \mu A + 250\ \mu A} = 128\ d$$

   256 decimal (256 d) is rounded down to 255 d to give a high value of 248.04 µA.

4. The power consumption of the DACx3004 will vary based on the configuration used, and the power down settings. The power consumption is given by:

$$P = (VDD \times IDD_{SLEEP}) + \sum_{x=0}^{N-1} (VDD \times IDD_X)$$

where

- IDD_{SLEEP} is the quiescent current for the device in sleep mode
- N is the number of channels powered on
- IDD_X is the quiescent current per channel powered on

a. One DAC channel is powered on in voltage output mode in the voltage output configuration with the external MOSFET. The quiescent current in voltage output mode is 35 µA typical per channel. The quiescent current of the DAC in sleep mode is 21 µA maximum. With a VDD of 5 V, the power consumption equation becomes:

$$P = (5\ V \times 21\ \mu A) + (5\ V \times 35\ \mu A) = 280\ \mu W$$

This calculation does not include the load current sourced from VCC though R_{SET}.

b. One DAC channel is powered on in current output mode in the current output configuration. With a current output range of 0 µA to 250 µA, the quiescent current is 18 µA typical per channel. In this configuration, the load current sourced by the DAC output channel also needs to be added. The power consumption equation becomes:

$$P = (5\ V \times 21\ \mu A) + (5\ V \times (18\ \mu A\ +\ 250\ \mu A)) = 1.445\ mW$$

c. All of the DAC channels are powered down and the device quiescent current is 3 µA maximum in deep-sleep mode. The power consumption equation becomes:

$$P = (5\ V \times 3\ \mu A) = 15\ \mu W$$

5. The slew rate between the high and low DAC codes can be programmed if these two values are stored in the MARGIN-HIGH and MARGIN-LOW DAC registers. The slew time is determined by the settings in the SLEW-RATE and CODE-STEP fields in the DAC-X-FUNC-CONFIG register. The slew time is given by:

$$Slew\ Time = \frac{(MARGIN\_HIGH\_CODE\ -\ MARGIN\_LOW\_CODE + 1)}{CODE\_STEP} \times SLEW\_RATE$$

If the CODE-STEP is set to 1 LSB, and the SLEW-RATE is set to 4 µs/step, the slew time for the voltage configuration becomes:

$$Slew\ Time = \frac{(512 - 0 + 1)}{1} \times 4\ \mu s = 2.05\ ms$$

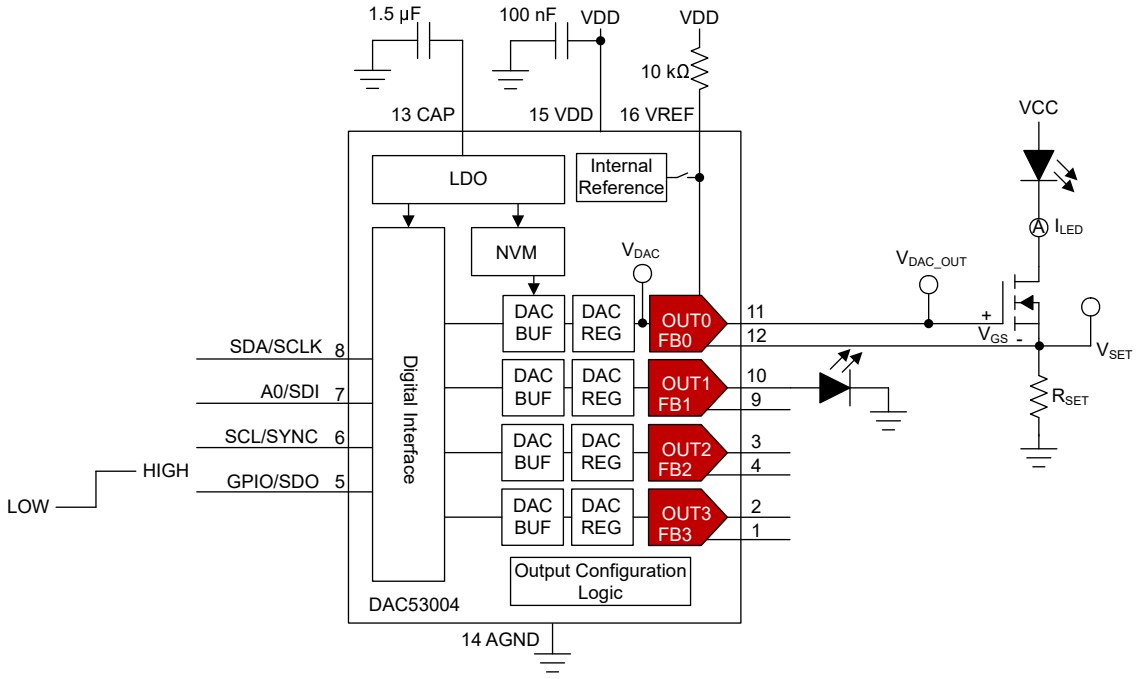
The slew time for the current output configuration becomes:

$$Slew\ Time = \frac{(255 - 128 + 1)}{1} \times 4\ \mu s = 512\ \mu s$$

6. The GPIO pin is used as an input to enter and exit deep-sleep mode. A falling edge on the GPIO pin puts the device in deep-sleep mode. The LDO takes approximately 550 µs to switch off and the device remains in deep-sleep mode as long as the GPIO input is low. A rising edge brings the device out of deep-sleep mode. The digital circuitry and the LDO take approximately 550 µs to switch on. The register settings to enable the GPIO for this function are described in the Register Settings section.
7. The DACx3004 can be programmed with the initial register settings described in the Register Settings section using I^2C or SPI. The initial register settings can be saved in the NVM by writing a 1 to the NVM-PROG field of the COMMON-TRIGGER register. After programming the NVM, the device loads all registers with the values stored in the NVM after a reset or a power cycle.

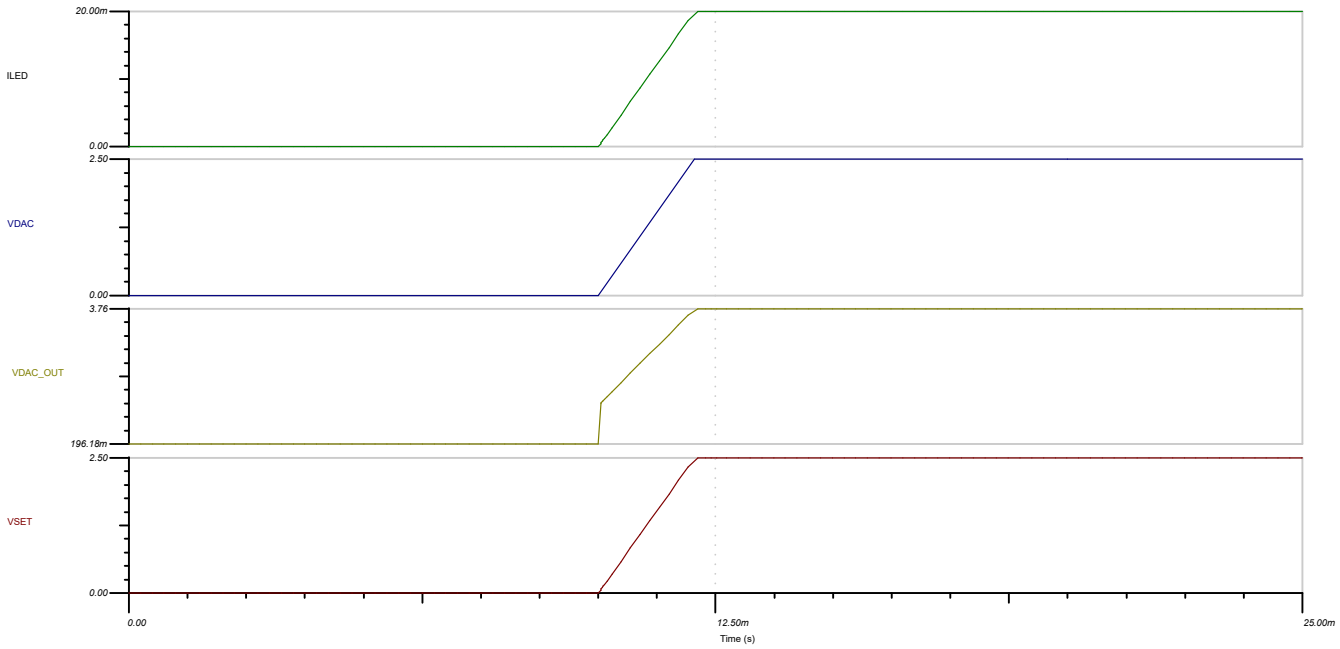
Design Simulations

This schematic is used for the following simulation of the DAC53004.



Transient Simulation Results

The simulation shows the LED current when the DAC slews from the margin low to margin high code.



Register Settings

Register Settings for Voltage Output Configuration

Register Address	Register Name	Setting	Description
0x01	DAC-0-MARGIN-HIGH	0x8000	[15:4] 0x800: 10-bit data left adjusted updates the MARGIN-HIGH code
			[3:0] 0x0: Don't care
0x02	DAC-0-MARGIN-LOW	0x0000	[15:4] 0x000: 10-bit data left adjusted updates the MARGIN-LOW code
			[3:0] 0x0: Don't care
0x06	DAC-0-FUNC-CONFIG	0x0001	[15] 0b0: Write 0b1 to set DAC-0 clear setting to mid-scale
			[14] 0b0: Write 0b1 to update DAC-0 with LDAC trigger
			[13] 0b0: Write 0b1 to enable DAC-0 to be updated with broadcast command
			[12:11] 0b00: Selects phase for function generator
			[10:8] 0b000: Selects waveform generated by the function generator
			[7] 0b0: Write 0b1 to enable logarithmic slew
			[6:4] 0b000: Selects code-step of 1 LSB
			[3:0] 0b001: Selects slew-rate of 4 µs/step
0x1F	COMMON-CONFIG	0x0FF9	[15] 0b0: Write 0b1 to set window-comparator output to a latching output
			[14] 0b0: Write 0b1 to lock device. Unlock by writing 0b0101 to DEV-UNLOCK field in the COMMON-TRIGGER register
			[13] 0b0: Write 0b1 to set fault-dump read enable at address 0x01
			[12] 0b0: Write 0b1 to enables the internal reference
			[11:10] 0b11: Powers-down VOUT3
			[9] 0b1: Powers-down IOUT3
			[8:7] 0b11: Powers-down VOUT2
			[6] 0b1: Powers-down IOUT2
			[5:4] 0b11: Powers-down VOUT1
			[3] 0b1: Powers-down IOUT1
			[2:1] 0b00: Powers-up VOUT0
			[0] 0b1: Powers-down IOUT0
0x20	COMMON-TRIGGER	0x0002	[15:12] 0b0000: Write 0b0101 to unlock the device
			[11:8] 0b0000: Write 0b1010 to trigger a POR reset
			[7] 0b0: Write 0b1 to trigger LDAC operation if the respective SYNC-CONFIG-X bit in the DAC-X-FUNC-CONFIG register is 1
			[6] 0b0: Write 0b1 to set the DAC registers and outputs to zero-code or mid-code based on the respective CLR-SEL-X bit in the DAC-X-FUNC-CONFIG register
			[5] 0b0: Don't care
			[4] 0b0: Write 0b1 to trigger fault-dump sequence
			[3] 0b0: Write 0b1 to trigger PROTECT function
			[2] 0b0: Write 0b1 to read one row of NVM for fault-dump
			[1] 0b1: Write 0b1 to store applicable register settings to the NVM
			[0] 0b0: Write 0b1 to reload applicable registers with existing NVM settings

Register Settings for Voltage Output Configuration (continued)

Register Address	Register Name	Setting	Description
0x24	GPIO-CONFIG	0x4001	[15] 0b0: Write 0b1 to enable glitch filter on GPI
			[14] 0b1: Enables deep-sleep function
			[13] 0b0: Write 0b1 to enable output mode on GPIO pin
			[12:9] 0b0000: STATUS function setting mapped to GPIO as output
			[8:5] 0b0000: Determines channels affected by channel-specific GPI functions
			[4:1] 0b0000: Selects GPI to trigger deep-sleep mode
			[0] 0b1: Enables input mode for GPIO pin

Register Settings for Current Output Configuration

Register Address	Register Name	Setting	Description
0x01	DAC-0-MARGIN-HIGH	0xFF00	[15:4] 0xFF0: 8-bit data left adjusted updates the MARGIN-HIGH code
			[3:0] 0x0: Don't care
0x02	DAC-0-MARGIN-LOW	0x8000	[15:4] 0x800: 8-bit data left adjusted updates the MARGIN-LOW code
			[3:0] 0x0: Don't care
0x06	DAC-0-FUNC-CONFIG	0x0001	[15] 0b0: Write 0b1 to set DAC-0 clear setting to mid-scale
			[14] 0b0: Write 0b1 to update DAC-0 with LDAC trigger
			[13] 0b0: Write 0b1 to enable DAC-0 to be updated with broadcast command
			[12:11] 0b00: Selects phase for function generator
			[10:8] 0b000: Selects waveform generated by the function generator
			[7] 0b0: Write 0b1 to enable logarithmic slew
			[6:4] 0b000: Selects code-step of 1 LSB
			[3:0] 0b001: Selects slew-rate of 4 µs/step
0x1F	COMMON-CONFIG	0x0FFE	[15] 0b0: Write 0b1 to set window-comparator output to a latching output
			[14] 0b0: Write 0b1 to lock device. Unlock by writing 0b0101 to DEV-UNLOCK field in the COMMON-TRIGGER register
			[13] 0b0: Write 0b1 to set fault-dump read enable at address 0x01
			[12] 0b0: Write 0b1 to enables the internal reference
			[11:10] 0b11: Powers-down VOUT3
			[9] 0b1: Powers-down IOUT3
			[8:7] 0b11: Powers-down VOUT2
			[6] 0b1: Powers-down IOUT2
			[5:4] 0b11: Powers-down VOUT1
			[3] 0b1: Powers-down IOUT1
			[2:1] 0b11: Powers-down VOUT0
			[0] 0b0: Powers-up IOUT0

Register Settings for Current Output Configuration (continued)

Register Address	Register Name	Setting	Description
0x20	COMMON-TRIGGER	0x0002	[15:12] 0b0000: Write 0b0101 to unlock the device
			[11:8] 0b0000: Write 0b1010 to trigger a POR reset
			[7] 0b0: Write 0b1 to trigger LDAC operation if the respective SYNC-CONFIG-X bit in the DAC-X-FUNC-CONFIG register is 1
			[6] 0b0: Write 0b1 to set the DAC registers and outputs to zero-code or mid-code based on the respective CLR-SEL-X bit in the DAC-X-FUNC-CONFIG register
			[5] 0b0: Don't care
			[4] 0b0: Write 0b1 to trigger fault-dump sequence
			[3] 0b0: Write 0b1 to trigger PROTECT function
			[2] 0b0: Write 0b1 to read one row of NVM for fault-dump
			[1] 0b1: Write 0b1 to store applicable register settings to the NVM
			[0] 0b0: Write 0b1 to reload applicable registers with existing NVM settings
0x24	GPIO-CONFIG	0x4001	[15] 0b0: Write 0b1 to enable glitch filter on GPI
			[14] 0b1: Enables deep-sleep function
			[13] 0b0: Write 0b1 to enable output mode on GPIO pin
			[12:9] 0b0000: STATUS function setting mapped to GPIO as output
			[8:5] 0b0000: Determines channels affected by channel-specific GPI functions
			[4:1] 0b0000: Selects GPI to trigger deep-sleep mode
			[0] 0b1: Enables input mode for GPIO pin

Pseudo Code Example

The following shows a pseudo code sequence to program the initial register values to the NVM of the DAC53004. The values given here are for the design choices made in the Design Notes.

Pseudo Code Example for Voltage Output Configuration

```
1:  //SYNTAX: WRITE <REGISTER NAME (Hex code)>, <MSB DATA>, <LSB DATA>
2:  //Configure GPI for deep-sleep trigger and enable deep-sleep function
3:  WRITE GPIO-CONFIG(0x24), 0x40, 0x01
4:  //Write DAC0 margin high code
5:  //With 16-bit left alignment 0x200 becomes 0x8000
6:  WRITE DAC-0-MARGIN-HIGH(0x01), 0x80, 0x00
7:  //Write DAC0 margin low code
8:  WRITE DAC-0-MARGIN-LOW(0x02), 0x00, 0x00
9:  //Set the CODE-SETP to 1 LSB and SLEW-RATE to 4 µs/step
10: WRITE DAC-0-FUNC-CONFIG(0x06), 0x00, 0x01
11: //Power-up voltage output on channel 0, internal reference disabled
12: WRITE COMMON-CONFIG(0x1F), 0x0F, 0xF9
13: //Enable the GPI, save settings to NVM
14: WRITE COMMON-TRIGGER(0x20), 0x00, 0x02

15: //Trigger for channel 0 margin high
16: WRITE COMMON-DAC-TRIG(0x21), 0x02, 0x00
17: //Trigger for channel 0 margin low
18: WRITE COMMON-DAC-TRIG(0x21), 0x04, 0x00
```

Pseudo Code Example for Current Output Configuration

```
1:  //SYNTAX: WRITE <REGISTER NAME (Hex code)>, <MSB DATA>, <LSB DATA>
2:  //Configure GPI for deep-sleep trigger and enable deep-sleep function
3:  WRITE GPIO-CONFIG(0x24), 0x40, 0x01
4:  //Write DAC0 margin high code
5:  //With 16-bit left alignment, 0xFF becomes 0xFF00
6:  WRITE DAC-0-MARGIN-HIGH(0x01), 0xFF, 0x00
7:  //Write DAC0 margin low code
8:  //With 16-bit left alignment, 0x80 becomes 0x8000
9:  WRITE DAC-0-MARGIN-LOW(0x02), 0x80, 0x00
10: //Set the CODE-SETP to 1 LSB and SLEW-RATE to 4 µs/step
11: WRITE DAC-0-FUNC-CONFIG(0x06), 0x00, 0x01
12: //Power-up current output on channel 0, internal reference disabled
13: WRITE COMMON-CONFIG(0x1F), 0x0F, 0xFE
14: //Enable the GPI, save settings to NVM
15: WRITE COMMON-TRIGGER(0x20), 0x00, 0x02

16: //Trigger for channel 0 margin high
17: WRITE COMMON-DAC-TRIG(0x21), 0x02, 0x00
18: //Trigger for channel 0 margin low
19: WRITE COMMON-DAC-TRIG(0x21), 0x04, 0x00
```

Design Featured Devices

Device	Key Features	Link
DAC53004	Ultra-low-power, 4-channel, 10-bit, smart DAC with I2C, SPI and PWM	DAC53004
DAC63004	Ultra-low-power, 4-channel, 12-bit, smart DAC with I2C, SPI and PWM	DAC63004

Find other possible devices using the Parametric search tool.

Design References

See *Analog Engineer's Circuit Cookbooks* for TI's comprehensive circuit library.

Additional Resources

- Texas Instruments, DAC63204 Evaluation Module
- Texas Instruments, DAC63204 EVM User's Guide
- Texas Instruments, Precision Labs - DACs

For direct support from TI Engineers, use the E2E community: e2e.ti.com

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265