Analog Engineer's Circuit

Digital input to PWM output circuit using smart DACs

Smart DAC *Katlynne Jones*

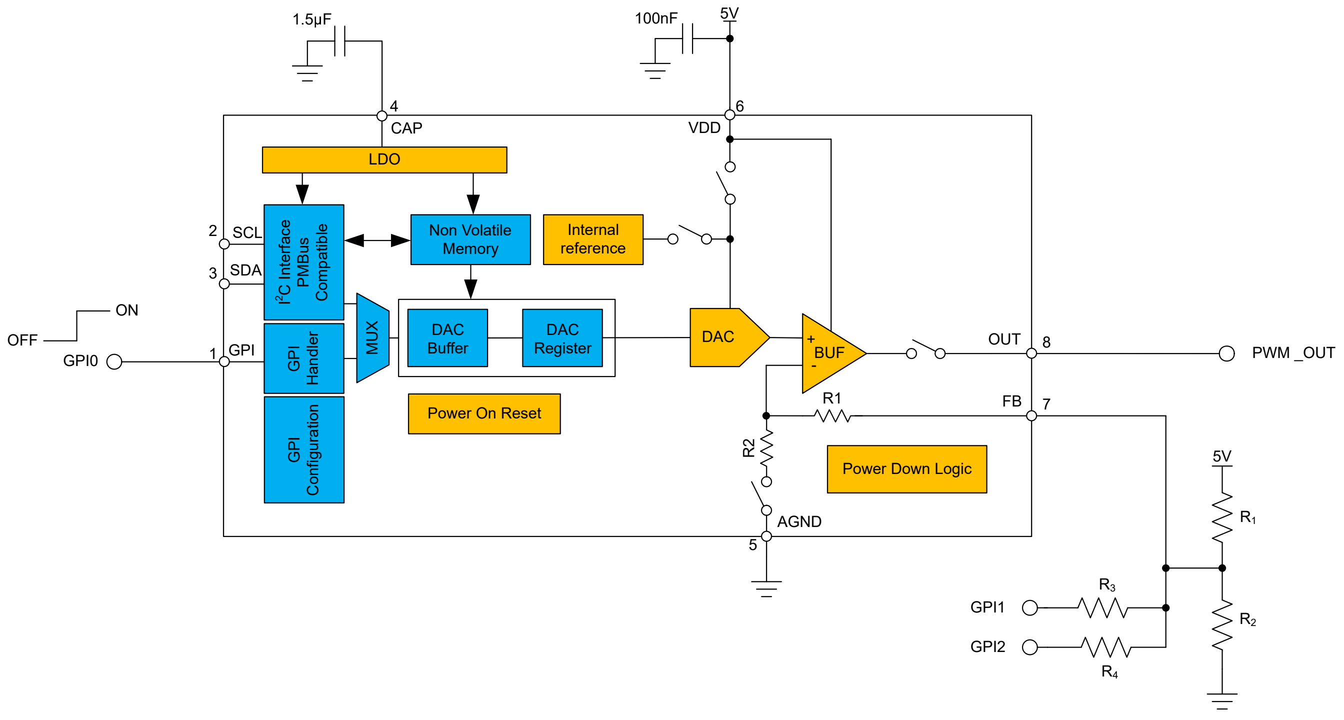
Design Objective

Key Input Parameter	Key Output Signal	Recommended Device
0-V to 5.5-V analog input, programmable triangular waveform	Pulse-width modulation (PWM) output	DAC53701, DAC43701, DAC53701-Q1, DAC43701-Q1

Objective: *Translate digital inputs to a variable duty cycle PWM output*.

Design Description

This design uses a buffered voltage output smart DAC to decode two general-purpose inputs (GPIs) into a constant-frequency PWM output with four selectable duty cycle levels. This design can be expanded to increase the number of GPIs and duty cycle levels. The 8-bit DAC43701 and 10-bit DAC53701 have an integrated continuous waveform generator (CWG) that can produce square, triangular, and sawtooth waveforms. In this design, the integrated buffer acts as a comparator and a triangle waveform generated by the CWG acts as the threshold for the comparator. The DAC43701 and DAC53701 integrated buffer has an exposed feedback path via the feedback pin (FB) which acts as the voltage input to the comparator. The comparator generates a PWM output with the same frequency as the triangle wave, and a duty cycle dependent on the FB input. All register settings can be saved using the non-volatile memory (NVM) on the DAC43701 and DAC53701 meaning that the devices can be used without a processor, even after a power cycle. This circuit can be used in applications such as automotive rear lights, rear light fault indication, and fault communication in factory automation and control designs.



Design Notes

1. The *DACx3701 10-Bit and 8-Bit, Voltage-Output Smart DACs With Nonvolatile Memory and PMBus™ Compatible I2C Interface With GPI Control 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. An external reference of 1.8V to 5.5V can be applied to the VDD pin of the device. In addition, there is an internal precision 1.21-V reference with ×1.5, ×2, ×3, and ×4 gain options. When using VDD as a reference, noise on VDD is directly translated to noise on the triangular waveform produced by the DAC53701. This noise is ratiometric to the voltage applied to the FB pin.
3. In this design, the 5-V VDD supply input is used as the reference. The duty cycle of the PWM output is decided by the high- and low-voltage levels of the triangular waveform, and the voltage applied to the FB pin of the DAC53701 (V_{FB}):

$$DUTY\_CYCLE = \frac{V_{TRIANGLE\_HIGH} - V_{FB}}{V_{TRIANGLE\_HIGH} - V_{TRIANGLE\_LOW}} \times 100\%$$

Values of 4V and 1V are used for $V_{TRIANGLE\_HIGH}$ and $V_{TRIANGLE\_LOW}$ to avoid zero-code and full-scale errors. With an example V_{FB} of 2.5V and unity gain, the equation becomes:

$$DUTY\_CYCLE = \frac{4V - 2.5V}{4V - 1V} \times 100\% = 50\%$$

The DAC codes for $V_{TRIANGLE\_HIGH}$ and $V_{TRIANGLE\_LOW}$ are stored in the DAC_MARGIN_HIGH and DAC_MARGIN_LOW registers. The codes programmed to these registers, in decimal, are calculated using:

$$DAC\_MARGIN\_HIGH = \frac{V_{TRIANGLE\_HIGH}}{V_{REF} \times GAIN} \times 1024$$

$$DAC\_MARGIN\_LOW = \frac{V_{TRIANGLE\_LOW}}{V_{REF} \times GAIN} \times 1024$$

The equation becomes:

$$DAC\_MARGIN\_HIGH = \frac{4V}{5V} \times 1024 = 819.2d$$

$$DAC\_MARGIN\_LOW = \frac{1V}{5V} \times 1024 = 204.8d$$

This is rounded to 819d and 205d to give a $V_{TRIANGLE\_HIGH}$ of 3.999V and $V_{TRIANGLE\_LOW}$ of 1V.

4. The frequency of the PWM output is equal to the frequency of the triangular waveform. This is calculated by:

$$f_{TRIANGLE\_WAVE} = \frac{1}{2 \times SLEW\_RATE \times \left(\frac{MARGIN\_HIGH - MARGIN\_LOW + 1}{CODE\_STEP}\right)}$$

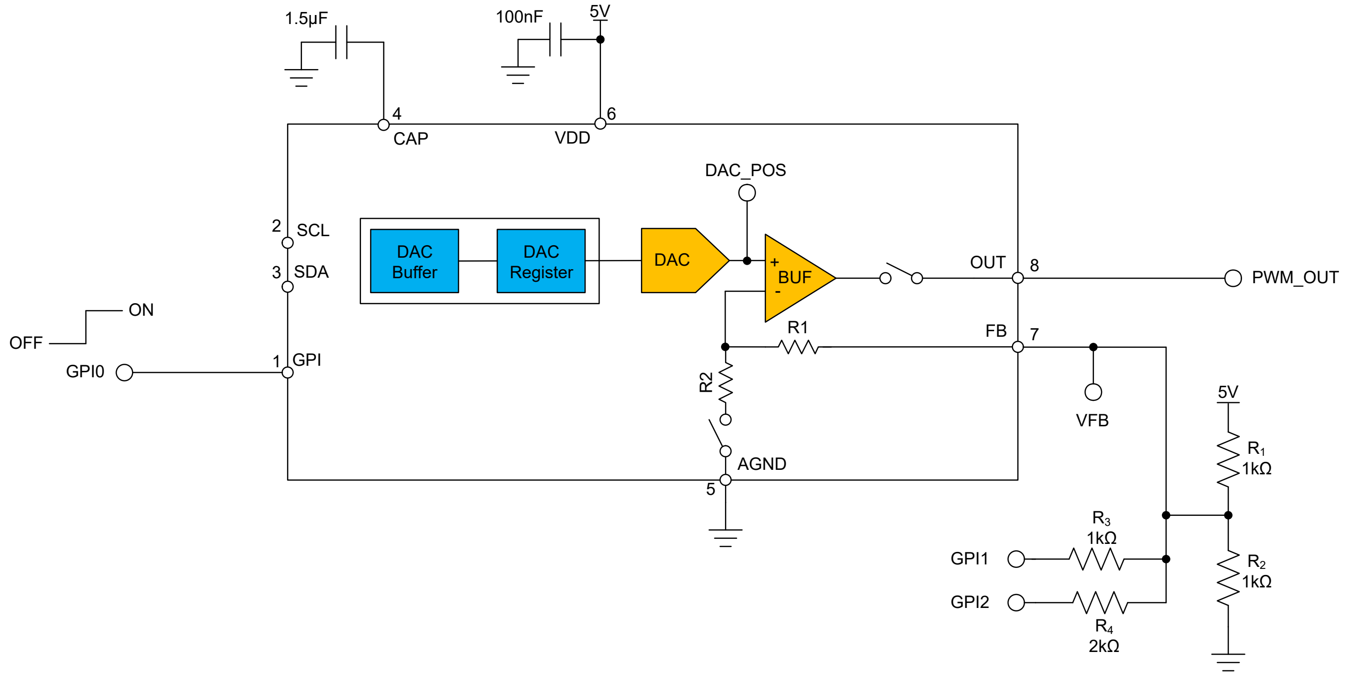
   SLEW_RATE and CODE_STEP are selected in the GENERAL_CONFIG Register. A CODE_STEP of 8 least significant bits (LSBs) and SLEW_RATE of 32µs per code step can be selected to produce a frequency of 203.25Hz:

$$f_{TRIANGLE\_WAVE} = \frac{1}{2 \times 32\mu s \times \left(\frac{819 - 205 + 1}{8}\right)} = 203.25Hz$$

5. Using a 5-V reference and the 10-bit DAC53701, the LSB size, or step size between each code, is about 4.88mV. Using lower reference voltages decreases the LSB size and thus increases the resolution of the MARGIN_HIGH and MARGIN_LOW voltages.
6. In this design, GPI0 is used for the Power-Up, Down (10kΩ) function. A high start pulse must be applied to GPI0 to power-on the DAC53701. The function of GPI0 can be modified in the GPI_CONFIG field of the CONFIG2 register.
7. GPI1 and GPI2 along with $R_{1–4}$ determine the voltage at V_{FB}. GPI1 and GPI2 toggle between 0 and VDD which changes the values in the resistor divider created by the different parallel combinations of resistors $R_{1–4}$. For example, when GPI1 and GPI2 are both off, R_{2-4} are added in parallel, and create a resistor divider with R_1. If the digital values applied to GPI1 and GPI2 are different from VDD or ground, MOSFETs can be added as level translators to each GPI.
8. The DAC53701 can be programmed with the initial register settings described in the Register Settings section using I2C. The initial register settings can be saved in the NVM by writing a 1 to the NVM_PROG field of the TRIGGER register. After programming the NVM, the device loads all applicable registers with the values stored in the NVM after a reset or a power cycle.

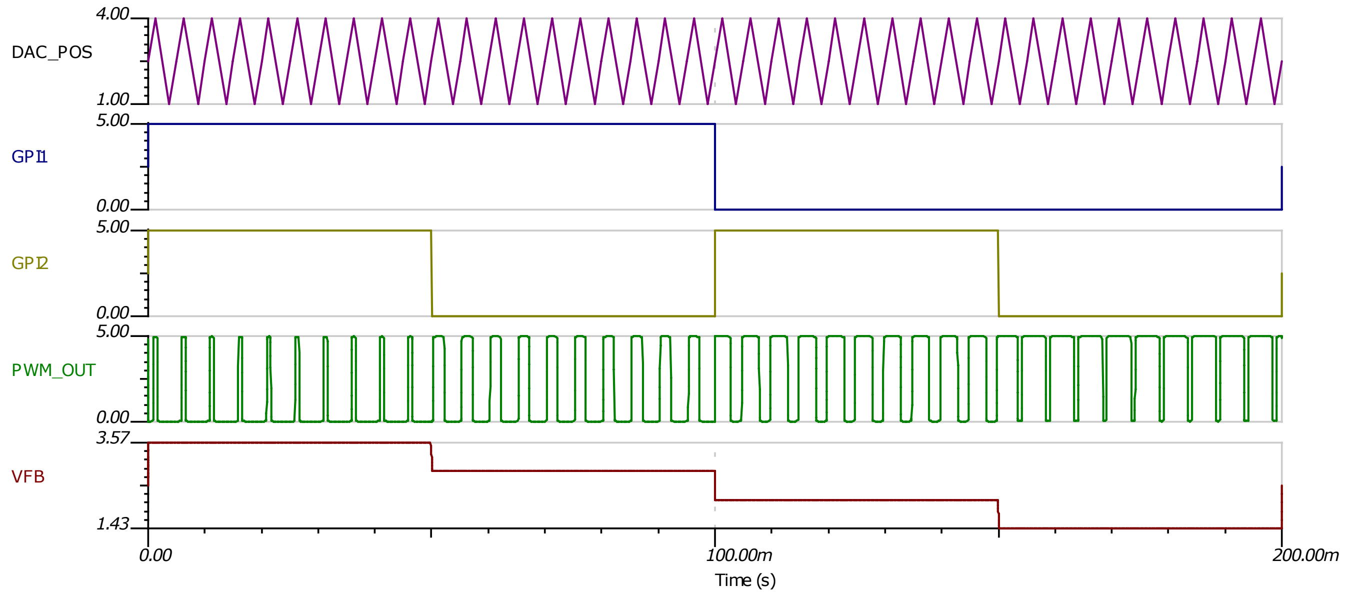
Design Simulations

This schematic is used for the following simulations of the DAC53701 GPI to PWM.



Transient Simulation Results

The simulation shows the DAC53701 output responding to the changes on GPI1 and GPI2. The DAC_POS value reflects the triangle wave generated by the CWG. The value at VFB changes with the voltage levels on the GPI pins and adjusts the duty cycle of the PWM output.



Register Settings

Register Settings for the DAC53701 GPI to PWM

Register Address	Register Name	Setting	Description
0xD1	GENERAL_CONFIG	0x0A20	[15:14] 0b00: Selects triangular waveform to be generated by the continuous waveform generator (CWG)
			[13] 0b0: Write 0b1 to lock device. Unlock by writing 0b0101 to DEVICE_UNLOCK_CODE field in the PROTECT register
			[12] 0b0: Write 0b1 to enable PMBus mode
			[11:9] 0b101: Selects code step of 8 LSB used for programmable slew rate control
			[8:5] 0b0001: Selects slew rate of 32µs per code step used for programmable slew rate control
			[4:3] 0b00: Powers up the device output
			[2] 0b0: Disables the internal reference
			[1:0] 0b00: Selects the gain to be applied to the internal reference
0xD2	CONFIG2	0x0800	[15:14] 0b00: Configures the device I2C address
			[13:11] 0b001: Configures the GPI pin as Power-Up, Down (10kΩ) trigger
			[10] 0b0: Write 0b1 to generate a high priority medical alarm
			[9] 0b0: Write 0b1 to generate a medium priority medical alarm
			[8] 0b0: Write 0b1 to generate a low priority medical alarm
			[7:6] 0b00: Always write 0b00
			[5:4] 0b00: Selects the interburst time for medical alarms
			[3:2] 0b00: Selects the pulse off time for medical alarms
			[1:0] 0b00: Selects the pulse on time for medical alarms
0xD3	TRIGGER	0x0510	[15:12] 0b0000: Write 0b0101 to unlock the device
			[11] 0b0: Don't care
			[10] 0b1: Write 0b1 to enable the GPI pin
			[9] 0b0: Write 0b1 to load all registers with factory reset values
			[8] 0b1: Write 0b1 to start continuous waveform generation output
			[7] 0b0: Write 1 to initiate PMBus MARGIN_HIGH command
			[6] 0b0: Write 1 to initiate PMBus MARGIN_LOW command
			[5] 0b0: Write 0b1 to reload applicable registers with existing NVM settings
			[4] 0b1: Write 0b1 to store applicable register settings to the NVM
			[3:0] 0b0000: Write 0b1010 to reset registers with existing NVM settings or default settings
0x25	DAC_MARGIN_HIGH	0x0CCC	[15:12] 0b0000: Don't care
			[11:2] 0x333: 10-bit data updates the MARGIN_HIGH code
			[1:0] 0b00: Don't care
0x26	DAC_MARGIN_LOW	0x0334	[15:12] 0b0000: Don't care
			[11:2] 0xCD: 10-bit data updates the MARGIN_LOW code
			[1:0] 0b00: Don't care

Pseudo Code Example

The following shows a pseudo code sequence to program the initial register values to the NVM of the DAC53701. The values given here are for the design choices made in the Design Notes.

Pseudo Code Example for GPI to PWM

```
//SYNTAX: WRITE <REGISTER NAME (Hex code)>, <MSB DATA>, <LSB DATA>
//Power-up the device, internal reference disabled, set code step and slew rate
WRITE GENERAL_CONFIG(0xD1), 0x0A, 0x20
//Configure GPI for Margin-High, Low function
WRITE CONFIG2(0xD2), 0x08, 0x00
//Write DAC margin high code
WRITE DAC_MARGIN_HIGH(0x25), 0x0C, 0xCC
//Write DAC margin low code
WRITE DAC_MARGIN_LOW(0x26), 0x03, 0x34
//Enable the GPI, start CWG, save settings to NVM
WRITE TRIGGER(0xD3), 0x05, 0x10
```

Design Featured Devices

Device	Key Features	Link
DAC53701	10-bit buffered voltage-output smart digital-to-analog converter	ti.com/product/DAC53701
DAC53701-Q1	Automotive 10-bit buffered voltage-output smart digital-to-analog converter	ti.com/product/DAC53701-Q1
DAC43701	8-bit buffered voltage-output smart digital-to-analog converter	ti.com/product/DAC43701
DAC43701-Q1	Automotive 8-bit buffered voltage-output smart digital-to-analog converter	ti.com/product/DAC43701-Q1

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, DAC53701 Evaluation Module
- Texas Instruments, *DAC53701EVM User’s Guide*
- Texas Instruments, Precision Labs - DACs

For direct support from TI Engineers, use the E2E community:

e2e.ti.com