AN64275

PSoC® 3 and PSoC 5LP: Getting More Resolution from 8-Bit DACs

Author: Mark Hastings

Associated Part Family: All PSoC 3 and PSoC 5LP parts

Associated Project: Yes

Software Version: PSoC® Creator™ 4.0 or higher

To get the latest version of this application note, or the associated project file, please visit http://www.cypress.com/go/AN64275.

AN64275 discusses several methods to increase the resolution of the DACs available in the PSoC® 3 and PSoC 5LP families. These methods can be used to extend the resolution up to 12 bits. An example application is supplied to demonstrate most of these concepts. A library is also included that implements three of the methods as PSoC Creator™ components.

Contents

1 Introduction....1
2 What are INL and DNL?....2
3 Summary of Results....3
4 Voltage or Current DAC....3
5 Parallel DACs Method (PIDAC)....4
6 Dithered Output DAC (DVDAC)....8
6.1 Dithered VDAC Limitations....10
7 Modulated IDAC (MIDAC)....12
8 ADC Feedback DAC....15
9 Which DAC is Right for You?....16
9.1 Test Setup....17
9.2 PSoC Advantage....17
10 Using these DACs in your Project....18
11 Summary....21
Document History....22
Worldwide Sales and Design Support....23
Products....23
PSoC® Solutions....23
Cypress Developer Community....23
Technical Support....23

1 Introduction

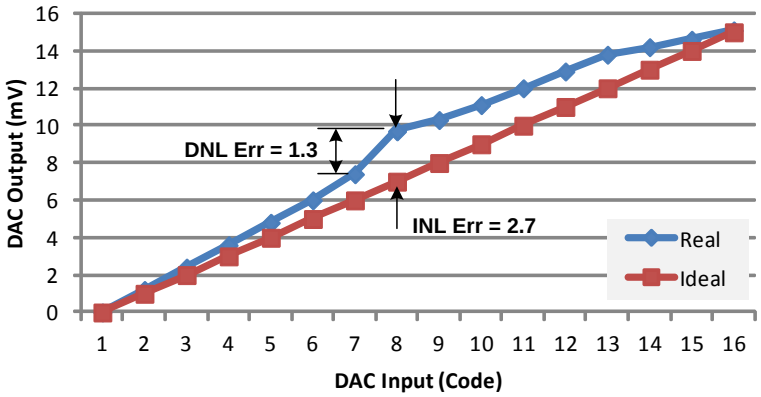
The PSoC 3 and PSoC 5LP families have up to four 8-bit voltage or current DACs (viDAC). These DACs have overlapping ranges, two in the voltage mode and three in current mode. In voltage mode, the full-scale voltage ranges are 1 and 4 volts. In current mode, the ranges are 32, 256, and 2048 µA. For many applications, eight bits may be sufficient, but there may be times where more resolution can save a design. This application note discusses four methods to extend the resolution up to 12 bits. The four methods presented in this application note use one or two of the existing 8-bit DACs and other PSoC components to achieve the higher resolution. Following is the list of DAC resolution enhancement techniques covered in this application note:

- Parallel IDACs (component included)
- Dithered Output DAC (component included)
- Modulated IDAC (component included)
- Parallel DACs with ADC feedback

2 What are INL and DNL?

Before we jump into designing a higher resolution DAC, it is best to understand two important DAC specifications, Differential nonlinearity (DNL) and integral nonlinearity (INL). DNL is probably the most important specification for a DAC. It is the difference between the ideal step size and the actual step size between two successive output codes. For example, if you have a 10-bit voltage DAC that has a full scale of 1.023 volts, the ideal step size would be 1 mV. If one or more steps are measured to be 1.5 mV, the DNL error would be 1.5 – 1.0, or 0.5 LSb. Ideally you want the DNL error to be zero, but a DNL less than 1.0 is usually acceptable. A 10-bit DAC with a DNL between 1 and less than 2 would be considered a 9-bit DAC. See Figure 1 for an example of DNL.

Figure 1. DNL and INL Error



INL is the deviation from the DAC's actual transfer function. Ideally, you would like the INL to be one or less, but many applications will not suffer with an INL of several counts. One example where a 16-bit DAC with an INL of 10 or 12 may be acceptable is the audio. Also, applications where waveform shape is important, but absolute accuracy is not required are applications where a higher INL may be acceptable. Applications that require absolute output accuracy may require a much lower INL, such as 1 or 2 counts. Examples of applications that require a low INL are voltage references, power supplies, or any application that requires an accurate reference without an ADC to close the loop. Figure 1 shows an example of a DAC that has an INL greater than 1.

The 8-bit current and voltage DACs in PSoC 3 and PSoC 5LP have an INL of about 2 and a DNL less than 1. Because of this, we are able to easily achieve useful higher resolution DACs. With each of the methods mentioned above, the goal is to increase the resolution until the DNL becomes 1 or greater. In most cases the INL increases as the resolution increases, but as stated before, a larger INL may be acceptable for many applications.

3 Summary of Results

If you do not care about all the testing or how the DACs work and want to get right to the results, take a look at Table 1. It shows each of the methods discussed in the application note, the achievable resolution, INL, DNL, and DAC speed. Use this table to find the DAC that fits your needs. You can then choose to jump right to the section of interest for more information.

Table 1. DAC Resolution, INL, DNL, and Speed Summary

DAC Type	Resolution	INL	DNL	Speed
PIDAC	9	1	0.25	4 Msps
	10	1.5	0.5	4 Msps
	11	3	1.1	4 Msps
DVDAC (1 V Range)	9	1.2	0.2	1.13 Msps
	10	3	0.25	190 ksps
	11	5.5	0.4	36 ksps
	12	11	0.8	7 ksps
MIDAC	9	0.6	0.35	4 Msps
	10	1.0	0.6	2.3 Msps
	11	2	1.0	430 ksps
ADC Feedback	12+	1	1	~100 sps

4 Voltage or Current DAC

Designers too often ignore current DACs in favor of voltage DACs when needing a variable voltage source. Since most voltage DACs have fixed ranges, you must adjust your design to make the best use of the DAC's native range. A current DAC on the other hand can be very flexible in providing just the right voltage range required for a given application. By adding a single external resistor, you can optimize the voltage range to your application instead of the other way around. For example, the internal voltage DACs in PSoC devices has a full-scale output range of either 1 or 4 volts. What if you need a full-scale range of 2.3 volts? You can use the 4 volt range and route the output to a voltage divider (2 resistors), or you could use a single current DAC and one resistor. For example, to achieve the 2.3 volts full scale, you could select the 256-µA range and a 9.09 K load resistor. The equation to calculate the resistor is just ohms law. (Resistor_Value = Full_Scale_Volts * Full_Scale_Current). To let you in on a little secret, many voltage DACs are actually current DACs with an internal resistor, including the ones inside PSoC 3 and PSoC 5LP. So next time you see a current DAC do not think of it as an inferior device, it may be just what you need.

One complaint with voltage DACs is that the output impedance is not low enough and unable to drive much of a load. With most microcontrollers that include a DAC, you need to add an external amplifier to buffer the output. PSoC 3 and PSoC 5LP devices have up to four opamps internally that can be used to buffer a voltage DAC output. These opamps are capable of sinking or sourcing 25 mA, enough for most applications. These internal amplifiers in combination with the DACs provide a wide range of solutions for almost any application.

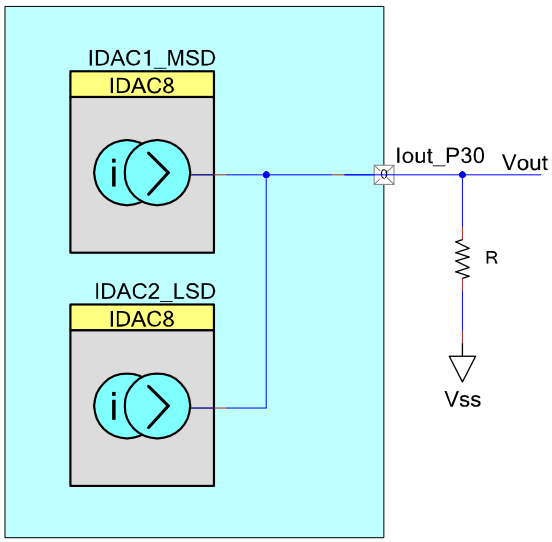
5 Parallel DACs Method (PIDAC)

The parallel DAC method requires two current DACs (iDAC) placed in parallel and set to two different overlapping current ranges. When two iDACs are put in parallel, the total current is the sum of both DACs. Notice in Table 2 how the three current ranges overlap. The left column indicates the weighting for each bit in the iDAC data register. For example, in the 256 µA range, the most significant bit (7) adds 128 µA when set. The least significant bit (0) adds 1 µA when set. Each two adjacent ranges overlap by 5 bits. Theoretically, you can construct a 14-bit iDAC by setting one DAC to the 2048 µA range and the second to the 32 µA range. Unfortunately, 8-bit DACs are seldom linear enough to achieve an INL or DNL that is low enough to for 14-bits.

Table 2. PSoC 3 and PSoC 5LP iDAC Overlapping Current Ranges

	Ranges		
µA	**2048 µA**		
1024	7		
512	6		
256	5	**256 µA**	
128	4	7	
64	3	6	
32	2	5	**32 µA**
16	1	4	7
8	0	3	6
4		2	5
2		1	4
1		0	3
0.5			2
0.25			1
0.125			0

Figure 2. Parallel Current DACs



In Figure 2, two iDACs are placed in parallel, configured in source mode and connected to the same load resistor. If the goal is to output a fixed voltage, a resistor can be used for the load and the desired voltage is across this resistor. The maximum voltage from the current source is the analog supply voltage (V_{DDA}) minus the compliance voltage of the current source, typically less than a volt. The routing resistance of the PCB and the internal PSoC signal path may also reduce the maximum voltage across the given load. This is usually only a concern for the 2048 µA range.

The load resistor is selected to provide a specific full-scale voltage level. R is determined by using Ohm's law.

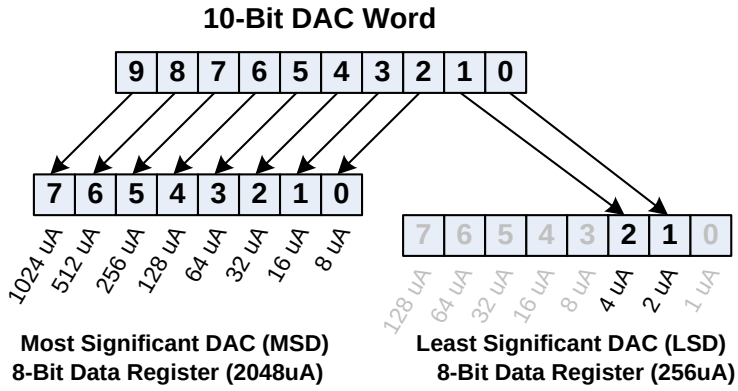
$$R = \frac{Full_Scale_Voltage}{Maximum_Current} \qquad \text{Equation 1}$$

For example, if the full-scale current is 2.048 mA and the desired full-scale voltage is 1.5 volts, the optimal resistor would be (1.5 V) / (0.002048 A) = 732 Ω.

In this example, the most significant DAC, 'IDAC1_MSD' is configured to source current and set to the 2048 µA range. The second iDAC, 'IDAC2_LSD' is also configured as a current source but set to the 256 µA range.

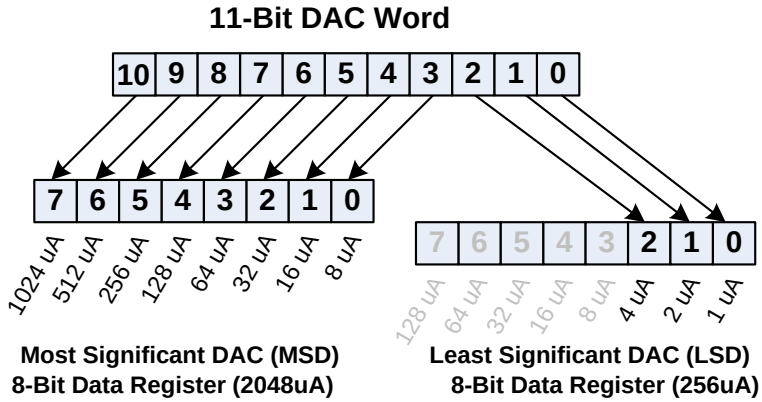
Since the two ranges (2048 µA and 256 µA) overlap by five bits as shown in Figure 3, there are six viable ways to configure the DACs to achieve a 10-bit iDAC. Figure 3 shows one possible alignment. The second iDAC could also have been configured in the 32 µA range and use bits 4 and 5 in the iDAC data register.

Figure 3. 10-bit DAC Data Register Bit Alignment



An 11-bit DAC would be a simple modification to the 10-bit DAC. The 8 most significant bits of the input word would be written to the higher current DAC and the 3 least significant bits would be written to bits 2, 1, and 0 in the lower current DAC. Figure 4 illustrates how the 11-bit control word is spread across the two DACs.

Figure 4. 11-bit DAC Data Register Bit Alignment



The code to configure the DACs and to write the value is rather straight forward. The following code fragment starts the two DACs, puts them both into source mode and puts the two DACs into two adjacent ranges.

Example code to configure DACs:

```
/* Start both DACs */
IDAC1_MSD_Start();
IDAC2_LSD_Start();

/* Sets both IDACs to source current */
IDAC1_MSD_SetPolarity(IDAC1_MSD_SOURCE);
IDAC1_MSD_SetPolarity(IDAC2_LSD_SOURCE);

/* Sets proper ranges */
IDAC1_MSD_SetRange(IDAC1_MSD_RANGE_2mA );
IDAC2_LSD_SetRange(IDAC2_LSD_RANGE_255uA);
```

The following example code shows how to split the data word into an MSB and LSB word to be written into the parallel DACs. The most significant DAC should be written first to minimize any glitch from one value to another since both DACs cannot be written at the exact same time. This glitch can be eliminated by setting the 'Strobe_Mode' parameter to 'External' in the customizer of each DAC. Then connect both strobe inputs to the same clock source. The code for implementing both a 10- and 11-bit iDAC is as follows.

```
/* 10-Bit SetValue function  */
void iDAC10_SetValue(uint16 dacValue)
    {
        uint8 msb, lsb;
        /* Split data into 2 bytes */
        msb = (uint8)(dacValue >> 2);
        lsb = (uint8)((dacValue << 1) & 0x06);
        /* Write values */
        IDAC1_MSD_SetValue(msb);
        IDAC2_LSD_SetValue(lsb);
    }

    /* 11-Bit SetValue function.  */
    void iDAC11_SetValue(uint16 dacValue)
    {
        uint8 msb, lsb;
        /* Split data into 2 bytes */
        msb = (uint8)(dacValue >> 3);
        lsb = (uint8)((dacValue << 0) & 0x07);
        /* Write values */
        IDAC1_MSD_SetValue(msb);
        IDAC2_LSD_SetValue(lsb);
    }
```

A PSoC Creator component using this method has been created and is part of the library accompanying this application note. The name of this component is the Parallel IDAC or PIDAC. The resolution for the PIDAC is selectable for 9, 10, or 11 bits. See PIDAC datasheet included with the component for more information.

The next step is to test this concept for increasing resolutions until the DNL exceeds 1. Following are the INL and DNL plots for 9, 10, and 11 bits.

Figure 5. 9-bit PIDAC DNL

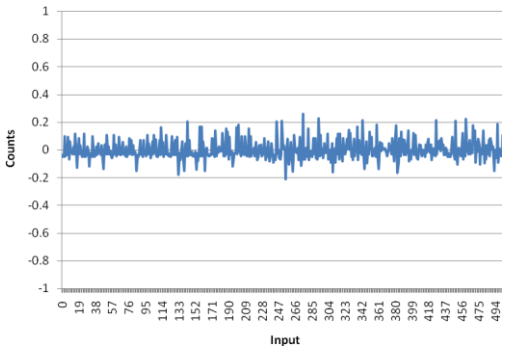
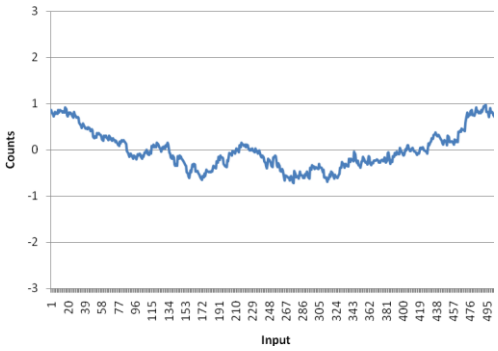


Figure 6. 9-bit PIDAC INL



As can be seen from the PIDAC INL and DNL plots we have very acceptable 9-bit performance.

Figure 7. 10-bit PIDAC DNL

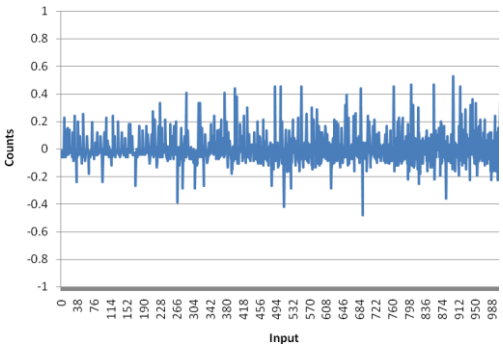


Figure 9. 11-bit PIDAC DNL

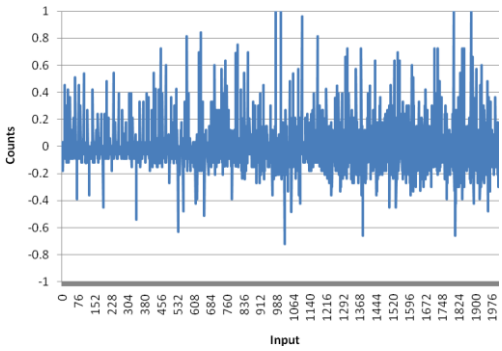


Figure 8. 10-bit PIDAC INL

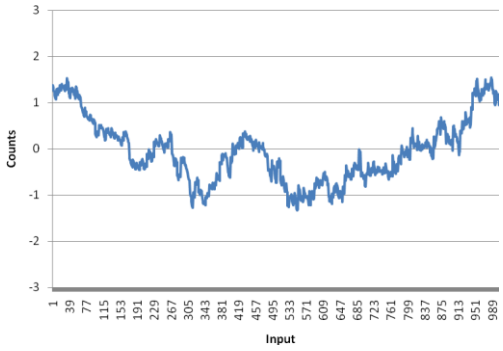
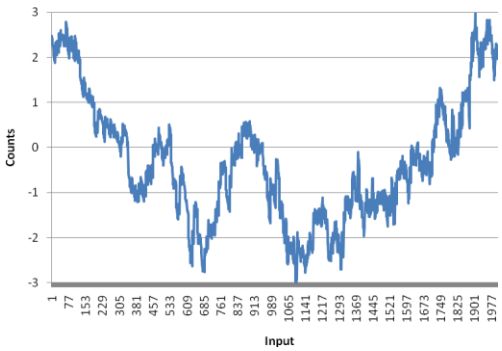


Figure 10. 11-bit PIDAC INL

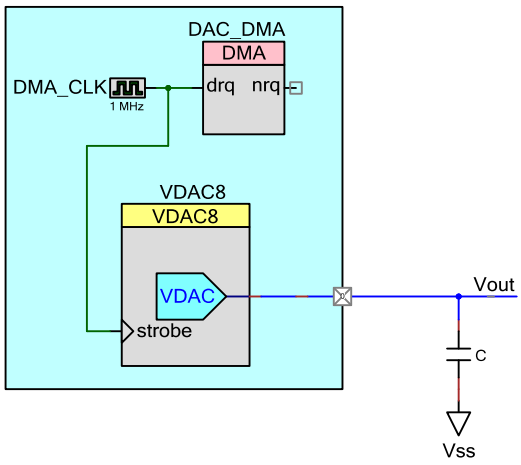


The 10-bit performance is still very respectable, but the DAC's 11-bit performance is becoming marginal. The INL is about 3 counts, but the DNL at some points is just above 1.

6 Dithered Output DAC (DVDAC)

The dithered output DAC uses a single current or voltage DAC, DMA channel, a clock, and a small array of RAM. The simple theory behind this method is that if you quickly write two or more different values to the DAC and filter the output, then the output is the average of the values written to the DAC. This assumes that values written to the DAC are periodic.

Figure 11. Dithered Outputs DAC



For example, to get 10-bits of resolution from an 8-bit DAC, you require a resolution of ¼ of the LSb. Suppose you are using a standard 8-bit VDAC with a full-scale voltage of 1.024 volts, the minimum resolution, or LSb is 4 mV. For 10 bits, the LSb is 1 mV. This means that you need two more bits of resolution or an LSb of ¼, you must average at least four values to achieve 10 bits of resolution with an 8-bit VDAC.

The trick is to output the data quickly and very periodically. With most average microcontrollers, you can use a timer to generate an interrupt every ‘n’ microseconds. The interrupt service routine would write the array of four values sequentially to the VDAC. The faster you write to the DAC, the simpler the filter will be and the faster the output will settle. The problem is that interrupting a processor every microsecond can consume a large share of your microcontroller’s CPU performance. Since the PSoC 3 and PSoC 5LP are not ordinary microcontrollers, DMA can be used to repeatedly write an array to the VDAC. A clock is used to trigger both the DMA and the VDAC to strobe the data output. The beauty of using the DMA is that after it is setup, there is ZERO CPU overhead. Except for a couple extra bytes of RAM and a small external capacitor, there is little cost to get 9 to 12 bits of resolution from an 8-bit VDAC in the PSoC 3 and PSoC 5LP parts.

Natively, the 8-bit VDAC in the 1-volt range provides a resolution of 4 mV (1.024 V / 256 = 0.004 mV). If you want an output of 500 mV you can simply write 125 to the DAC. (125 * 0.004 mV = 500 mV). But, if you require 501 mV, you have to settle for 500 or 504 mV. By dithering the output at a relatively high speed, the 501 mV output can be generated by averaging multiple output values. In this case a succession of four values can be periodically written to the VDAC. In the 1-volt range, a value of 125 written to the VDAC produces 500 mV. A value of 126 produces an output voltage of 504 mV. If you average the numbers 500, 500, 500, and 504, you get 501. The following table shows an example of how the output is dithered. The same pattern may be used between any two 8-bit steps to increase the resolution.

Table 3. Example Output of 10-bit VDAC

Sample	Array1	Array2	Array3	Array4
1	125	125	125	125
2	125	125	125	126
3	125	125	126	126
4	125	126	126	126
AVG	125.00	125.25	125.50	125.75
Average Volt (mV)	500	501	502	503

This dithering concept can be expanded beyond 10 bits, but there are some limitations to where the INL and DNL may become excessive. Usually the limit is where the DAC becomes non-monotonic or the DNL error exceeds one LSb.

A PSoC Creator component using this method has been created and is part of the library accompanying this application note. This DAC is called the DVDAC (Dithered Voltage DAC) and has a selectable resolution of 9 to 12 bits. See DVDAC datasheet included with the component for more information. The following plots show the DNL and INL for a 10-bit dithered VDAC.

Figure 12. DNL for 10-bit Dithered VDAC

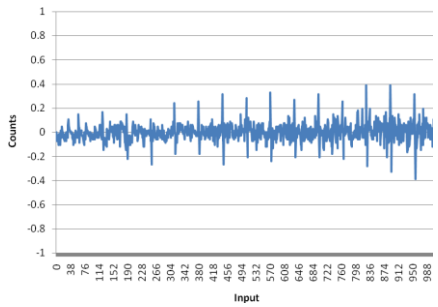
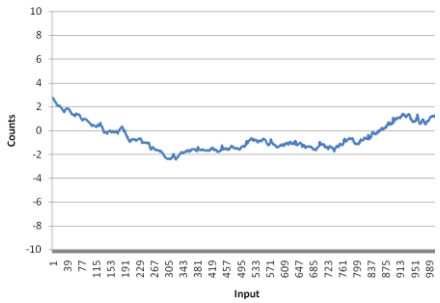


Figure 13. INL for 10-bit Dithered VDAC



The dithered VDAC concept can easily be extended beyond 10-bit version from the looks of the INL and DNL.

Further testing showed that the DNL error remained under 1 up to 12 bits or resolution. At 13 bits the DNL error exceeded 1.0 so currently the limit for the dithered VDAC will be 12 bits. Below are the INL and DNL plots for the 12-bit version.

Figure 14. DNL for the 12-bit Dithered VDAC

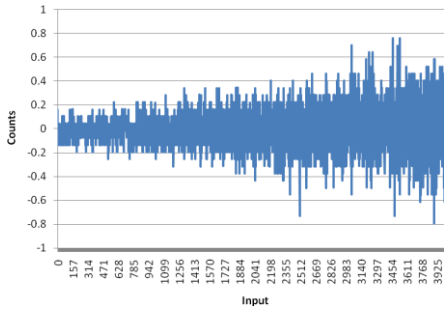


Figure 15. INL for the 12-bit Dithered VDAC



Notice that the shape of the INL for the 12-bit version is identical to that of the 10-bit version. This is really not a surprise since you expect it to have the same shape as the native 8-bit VDAC. The INL error is multiplied by the difference in resolution to the power of 2. Using this equation we would expect the INL error for 12 bits to be (12 bits – 10 bits)^2 * 2.5 INL = 10 counts. If we note the INL plot in Figure 15, we see that indeed the INL for 12 bits is about 10. The DNL also tends to double each time the resolution is increased by one bit.

6.1 Dithered VDAC Limitations

The dithered DAC cannot generate a true 2^{bits} unique output. The last N codes, where N = $2^{(bits - 8)}$ -1, all generate the same output voltage. This is due to the fact that the dithering requires two adjacent 8-bit DAC values to generate an average output signal. Once the internal 8-bit DAC's output is 255 (0xFF), there is no adjacent higher value. When the voltage DAC is configured for the 1 volt range and the highest output value is written to the VDAC, the output is 1.024 * (255 / 256) = 1.020 volts. This is the highest voltage the 8-bit VDAC can generate. Using the same equation for a 10-bit VDAC, we get a slightly higher output, 1.024 * (1023 / 1024) = 1.023 volts. But, as we are using a single 8-bit VDAC to simulate 10 or more bits, the maximum voltage is still that of the 8-bit VDAC. This means that any VDAC value higher than that of 8-bit VDAC is invalid. In the case of a 10-bit VDAC, the highest valid code is 1020, (1.024 * (1020 / 1024) = 1.020 volts. The following table specifies the code limit for each resolution.

Table 4. Valid Range of Dithered VDAC

Resolution (bits)	Valid Range	Invalid Codes	Flat Code Range
9	0–510	1	511
10	0–1020	3	1021–1023
11	0–2040	7	2041–2047
12	0–4080	15	4081–4095

Another more obvious limitation is the noise generated by the process of dithering. Since the output is the average of two adjacent values, the noise generated by dithering is small. In this case the noise is 4 mV (1.024 / 256) for the 1-volt range and 16 (4.096 / 256) mV for the 4-volt range. The actual dither frequency varies with the resolution of the DAC. If a 1 MHz dither clock is used for the PWM and the period is set to 4 (10-bits) the actual dither frequency is about 250 KHz (1 MHz/4).

A filter can be added to reduce the dither noise to an acceptable level. You are free to implement any type of active or passive filter required to reduce the dithered output noise. To keep external parts count low, a first order passive filter may be sufficient. A first order filter is simply a resistor and capacitor. Since the output resistance of the DAC is known, 4 kΩ for the 1-volt range and 16 kΩ for the 4-volt range, we get the resistor for free. This means that all we need to do is add a capacitor on the output. To calculate the capacitor value we first need to know just how much attenuation is required, and then determine the filter cutoff frequency. For each bit over 8-bits of resolution, the output needs to be attenuated by about 6 dB to attenuate the noise caused by the dither frequency. If we are making a 10-bit DAC, the dithered output would need to be attenuated by 12 dB, (6 db * (10 bits – 8 bits)). For an 11-bit DAC the attenuation will need to be 18 dB and so on. The filter's cutoff frequency is relative to the dither frequency. The VDAC8 specification states that the maximum clock rate is 1 MHz for the 1-volt range and 256 kHz for the 4 volt range, but this output is divided by 2^{B-8} where "B" is bits of resolution. For example if we want 10 bits of resolution in the 1 volt range, we divide the 1 MHz sample clock by 4 ($2^{(10-8)}$) or 1 MHz/4 = 250 kHz. The following table shows the attenuation required and dither frequency for each resolution and voltage range.

Table 5. Attenuation and Dither Frequency

Resolution	9	10	11	12	Bits
Attenuation	6	12	18	24	dB
1-volt dither frequency	500	250	125	62.5	kHz
4-volt dither frequency	125	62.5	31.3	15.6	kHz

Using the following equation, we can find the filter cutoff frequency.

$$Atten = 20 * \log\left(\frac{F_{dith}}{F_C}\right)$$ Equation 2

Where;

Atten is the amount of attenuation required for a given resolution. F_{dith} is the dither frequency and F_c is the filter cutoff frequency.

Solving for F_c;

$$F_c = \frac{F_{dith}}{10^{\frac{Atten}{20}}}$$ Equation 3

Now that we know the cutoff frequency, we can calculate the filter capacitor value.

$$F_c = \frac{1}{2\pi RC}$$ Equation 4

Solving for C;

$$C = \frac{1}{2\pi RF_c}$$ Equation 5

For example, if we build a 10 bit DAC using the voltage dithering method and want to calculate the value of C for the 1 volt range. The filter's cutoff frequency would be;

$$F_c = \frac{250kHz}{10^{\frac{12}{20}}} = 62.8kHz$$

Remember the internal resistance is about 4 kΩ for the 1 volt range and the filter cutoff frequency is 62.8 kHz from above we can solve for the filter capacitor value.

$$C = \frac{1}{2 * \pi * 4k\Omega * 62.8kHz} = 634\,pF$$

Using these equations we can solve for the required capacitor values for both ranges at each resolution. See the following table.

Table 6. Low-Pass Filter Capacitor Values

Resolution	9	10	11	12
C (1 volt Range)	160 pF	630 pF	2.5 nF	0.01 uF
Filter Cutoff	250 kHz	63 kHz	16 kHz	4 kHz
C (4 volt Range)	630 pF	2.5 nF	0.01 uF	0.04 uF
Filter Cutoff	16 kHz	4 kHz	1 kHz	250 Hz

The last thing to be concerned about is what the settling time of the DAC is at the given resolution.

$$V_S = V_{in}e^{\frac{-t}{RC}}$$ Equation 6

Where V_s is the settled voltage and V_{in} is the smallest step size of the 8-bit VDAC. If we make V_{in} one unit, then V_s is the faction of the smallest step that we need to settle to for the output to be accurate. Ideally, we want the output to be within one half the smallest step of the VDAC.

The step size of the DAC in terms of the initial VDAC can be expressed as follows.

$$V_s = 0.5 * \frac{1}{2^{B-8}}$$ Equation 7

Where B is the bits of resolution required. The "0.5" multiplier is because we want the error to be one half the step size.

If we combine these two equations, we get;

$$0.5 * \frac{1}{2^{B-8}} = V_{in} e^{\frac{-t}{RC}}$$ Equation 8

Solving for time (settling time);

$$t = -\ln(\frac{0.5}{2^{B-8}}) * R * C$$ Equation 9

If we use the example of the 10-bit, 1 volt full-scale VDAC, our settling time would be;

$$t = -\ln(\frac{0.5}{2^{2}}) * 4K * 630\,pF = 5.3uSec$$

These are rough calculations to get you quickly into the ballpark. The filter could easily be improved with a higher order passive or active filter.

A dithered current DAC could easily be constructed using the DVDAC as a template. All that would be needed is to change the output connection of the viDAC8 in the DAC's schematic, change the customizer to reflect the current range options, and a few lines of code in the API files. Since the current DAC can run at a higher sampling frequency, 8 MHz, the dither frequency is higher which makes the settling time much faster.

7 Modulated IDAC (MIDAC)

This method combines a standard 8-bit current DAC (IDAC8) and a PWM DAC. PWM DACs have been used for years as an inexpensive way to generate a linear voltage from digital hardware. The concept is simple. The PWM generates a waveform with an adjustable duty cycle. This is fed into a low-pass filter (LPF) and the DC output is a function of the PWM's peak-to-peak output times the duty cycle. If the LPF is a simple single pole RC filter, the modulation frequency needs to be close to three orders of magnitude higher than the filters cutoff frequency just for an 8-bit DAC. A higher order filter would probably be a better option.

Figure 16. PWM Block Diagram

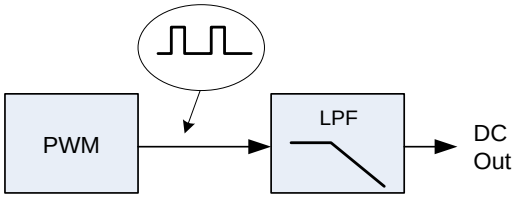
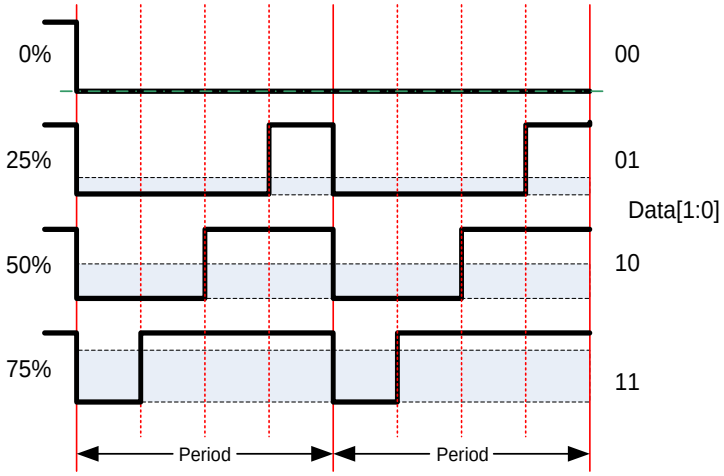
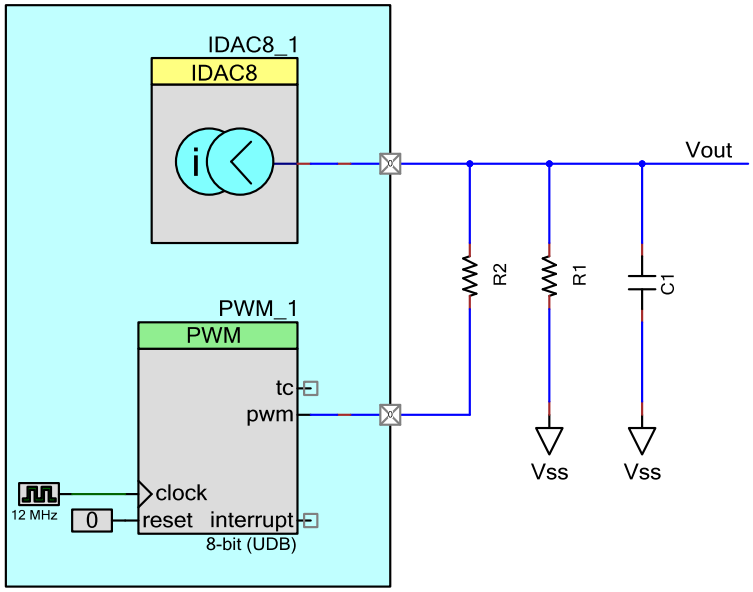


Figure 17. Example PWM Waveforms



With the modulated IDAC, the current is summed between the PWM DAC and the 8-bit current source (See Figure 18). The PWM's modulation amplitude is half the amplitude of the least significant bit of the 8-bit current DAC, because of the combination of R2 and R1. So the noise introduced by the PWM is more than 50 dB less than a typical PWM DAC and much easier to filter with a simple filter. The IDAC adjusts the most significant 8 bits of the current and the PWM adjusts the remaining few bits, depending on the resolution. In the case of a 10 bit DAC the PWM would add the additional 2 bits of resolution.

Figure 18. Modulated IDAC



This method requires a current DAC, two resistors, and a PWM. The output of the current DAC is connected to two resistors. One of the resistors is connected directly to V_{SSA} and the other is connected to the PWM output as shown in Figure 18. With just one resistor connected to the DAC's output and V_{SS}, the current DAC operates as a voltage DAC. The second resistor connected to the PWM must be much larger than the resistor connected to ground. The pin connected to the PWM should be set to "Strong Drive" so that it can drive the filter's capacitor. This DAC works much similar to the dithered voltage DAC except it uses a PWM to modulate the least significant bits instead of the DAC itself. One of the advantages is that the PWM can be modulated faster than that of the DAC itself. Unlike a typical PWM DAC where the output of the PWM that swings from V_{SS} to V_{DD}, the effective voltage swing of this PWM is much smaller and generates much less noise and therefore requires less filtering The PWM's output period should be sufficiently fast so that the modulated signal can be easily filtered.

To calculate the size of R1, simply divide the maximum output voltage required by the current range that is being used.

$$R1 = \frac{V\max}{I} \qquad \text{Equation 10}$$

Then use the value calculated for R1 to calculate R2.

$$R2 = R1 * \frac{Vdd + (V\max/256)}{(V\max/256)} \qquad \text{Equation 11}$$

Example:

If you require a DAC output voltage between 0 and 1.024 volts, using the 256 µA current range, and has a V_{DD} = 3.3 V solve for R1 and R2. (Using Equations 10 and 11 given earlier.)

$$R1 = \frac{1.024V}{256uA} = 4K\Omega \qquad \text{Equation 12}$$

$$R2 = 4K * \frac{3.3 + (1.024/256)}{(1.024/256)} = 3.3\ \text{MΩ}$$ Equation 13

The modulated IDAC has good performance up to 11 bits. The INL is better than that of the PIDAC or DVDAC at 2 counts for 11 bits. In 10 bit mode the maximum DNL was less than 0.6 counts and the INL was just under 1 count, not bad when derived from an 8-bit DAC.

A PSoC Creator component using this method has been constructed and is part of the library accompanying this application note. This DAC is called the MIDAC (Modulated Current DAC). Its parameters allow you to select both the range and the resolution between 9 and 11 bits. See MIDAC datasheet included with the component for more information. Following are the test result plots for INL and DNL for 9 to 11 bits.

Figure 19. DNL for 9-Bit MIDAC

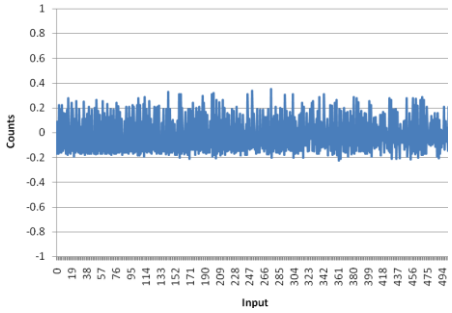


Figure 20. INL for 9-Bit MIDAC

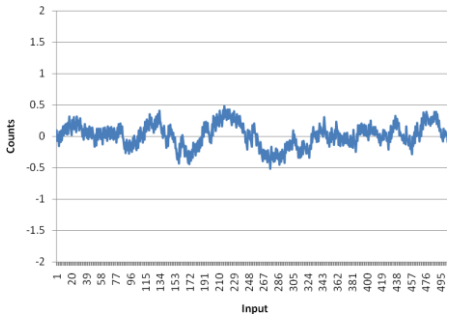


Figure 21. DNL for 10-Bit MIDAC

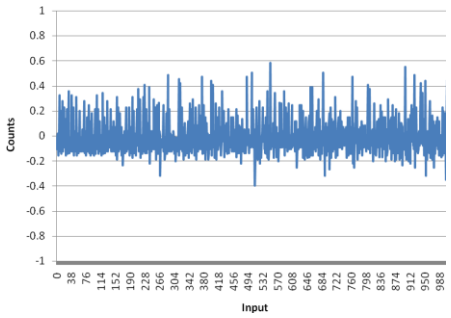


Figure 22. INL for 10-Bit MIDAC

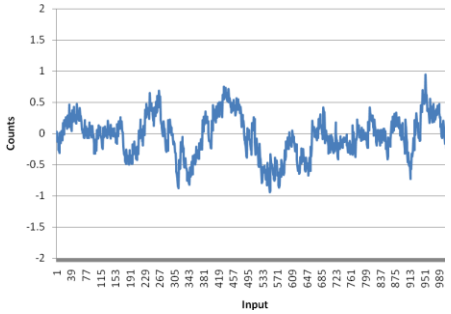


Figure 23. DNL for 11-Bit MIDAC

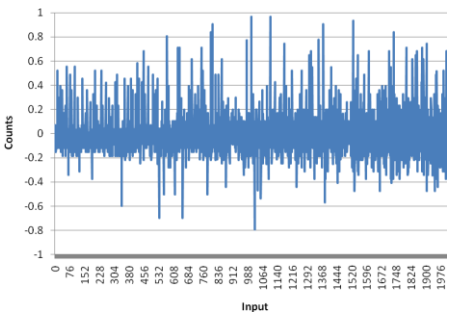
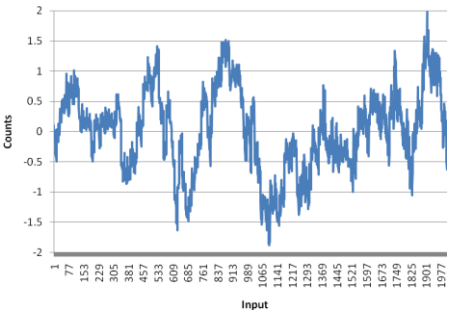


Figure 24. INL for 11-Bit MIDAC



As with the dithered voltage DAC, the PWM modulator generates noise on top of the signal. Although the DAC does use a PWM, the noise generated is less than one LSb in magnitude that is much less than a typical PWM DAC, where the output swings from V_{SS} to the maximum output voltage. To filter out this noise a capacitor may be added to the output of the DAC in parallel with the resistors, See Figure 18. The same method used to calculate the capacitor value for the dithered voltage DAC can be used here as well. As long as R2 is much larger than R1, you can ignore R2 and use R1 and C1 (Figure 18) for R and C respectively in Equations 3 and 5 given earlier. The following table shows examples of some capacitor values for a load of 4 K and a PWM clock of 12 MHz.

Table 7. Capacitor Value for R1 = 4 K

Resolution	9	10	11
Modulation Frequency	6 MHz	3 MHz	1.5 MHz
Attenuation	6 dB	12 dB	18 dB
Cap Value	13 pF	53 pF	210 pF

The settling time can also be calculated in the same fashion as we used with the dithered voltage DAC with Equation 9.

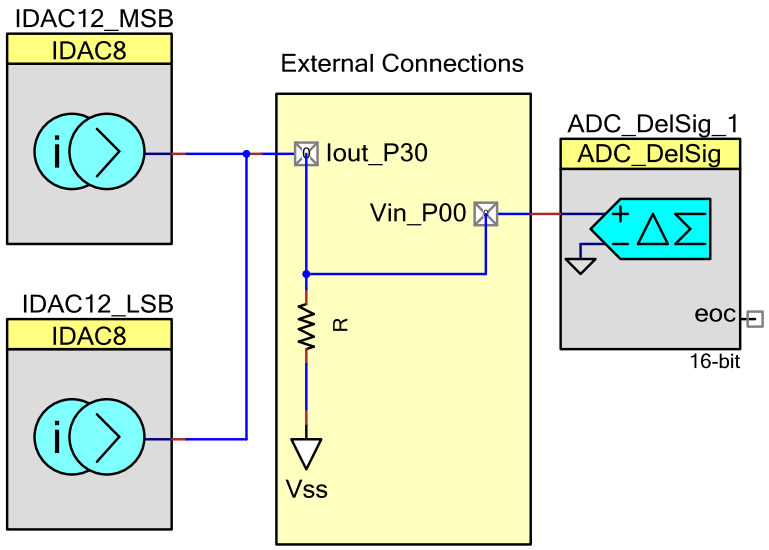
8 ADC Feedback DAC

This method uses the same DAC configuration used in the parallel IDAC method but is configured to provide at least 2 bits more resolution than required for the accuracy. For example, if 12 bits of accuracy is required, the IDAC resolution should be 14 bits. To achieve this, one IDAC should be configured to the 2 mA range and the other IDAC to the 32 µA range. These overlapping ranges provide a maximum resolution of 14 bits. The higher resolution parallel IDAC by itself may have excessive INL and DNL errors for 14 bit operation. The ADC used for this method should have an INL and DNL about half of the desired result for this method. The ADC is essentially continually calibrating the DAC and therefore needs to be more accurate than the DAC. The 20-bit Delta-Sigma ADC found in PSoC 3 and PSoC 5LP, has an INL and DNL of less than 1 at 16 bits. This is more than sufficient to trim a nonlinear 14-bit DAC to 12 bits.

In most cases, an external resistor is used as a linear load that converts the current output to a voltage. The ADC needs to be connected at the point where the voltage is used or buffered by the system to eliminate any IR (current * resistance) drop in the internal or external current path.

Firmware is required to complete a feedback loop between the IDAC output and the ADC. Each time the output is updated, an approximated value is applied. The output is allowed to settle and then measured with the ADC. If the output is not within 12 bits of accuracy, the IDAC output is trimmed to get closer to the desired value. This is an iterative process and may require 2 to 4 cycles to get within 12 bits of accuracy. The following figure is an example of a feedback IDAC using standard PSoC Creator components.

Figure 25. IDAC with Feedback



9 Which DAC is Right for You?

Four methods to create a higher resolution DAC have been presented in this application note. There are several factors that need to be considered when selecting which DAC is right for your application. Resolution, current versus voltage, speed, resources required, and need for external components all need to be considered in order to make a good decision. The current DACs are the most flexible as they can all be converted to a voltage DAC with the addition of an external resistor. The selection of the external resistor also allows you to optimize the full-scale voltage output as well.

For applications that require the fastest settling times, PIDAC and MIDAC are your best choices. The PIDAC provides 4 Msps for all ranges and does not require external components. The only downside is that the PIDAC uses two viDAC block resources. This may or may not be a concern depending on your overall application resource requirements. The MIDAC will be your next best selection for fast settling times and it uses only one viDAC block. Its speed does diminish as the resolution increases, but may be sufficient for many applications.

The DVDAC also provides relatively high update rates at low resolutions (9 and 10 bits), but slows down to 7 ksps at 12 bits. It has good DNL performance up to 12 bits and requires only one viDAC block and one DMA channel per DAC. This is a good option for many applications since it offers a good tradeoff between resolution and speed while using few analog resources.

Applications that require high accuracy (low INL), the ADC feedback method is the best approach, but its update rate is substantially slower than the other options, well under 1 ksps. It is possible to achieve 12 or maybe 13 bits with an INL of 1, but it will require the use of the DelSig ADC to close the feedback loop. In applications where a reference needs only be adjusted periodically and you can share the ADC, it could prove to be a very cost effective solution.

The following figure shows a comparison of INL between three of the methods discussed with a resolution of 10 bits. Table 8 shows a summary of resources required to implement the DAC.

Figure 26. INL Comparison for 10-bit DACs

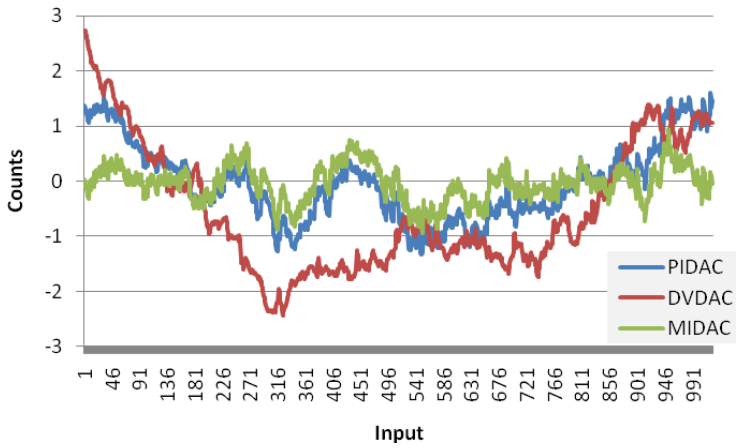


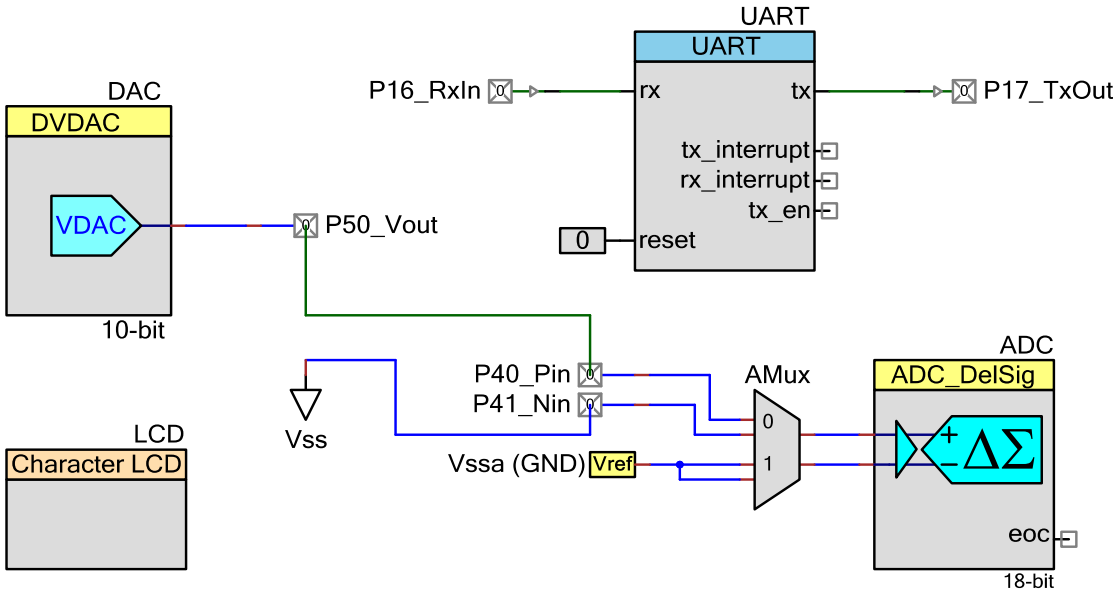
Table 8. Resources Required for DACs

DAC Type	Resources	Pins	External Components Required
PIDAC	viDAC8(2)	1	NA
DVDAC	viDAC8, DMA, Clock	1	Capacitor(1)
MIDAC	viDAC8, UDB(PWM)	2	Resistor(2) Capacitor(1)
ADC Feedback	viDAC8(2), 16-bit ADC	1 or 2	Resistor(1)

9.1 Test Setup

The test setup used for evaluation of these DACs consisted of a Cypress CY8CKIT-001 PSoC Development Board (DVK), a couple external components (Rs and Cs), a USB-to-Serial adapter, and a PC. The internal Delta-Sigma ADC, a UART, and the LCD were used for data collection. When testing current DACs, external resistors were used to convert the current to voltage so that the ADC could measure the output. Excel was used to convert the collected data into the INL and DNL plots in the document.

Figure 27. Test Setup



Other than a couple resisters and capacitors no special external hardware or development board is required to implement these methods. Any of the PSoC 3 or PSoC 5LP development systems (DVK, FTK) can be used to evaluate or test these concepts.

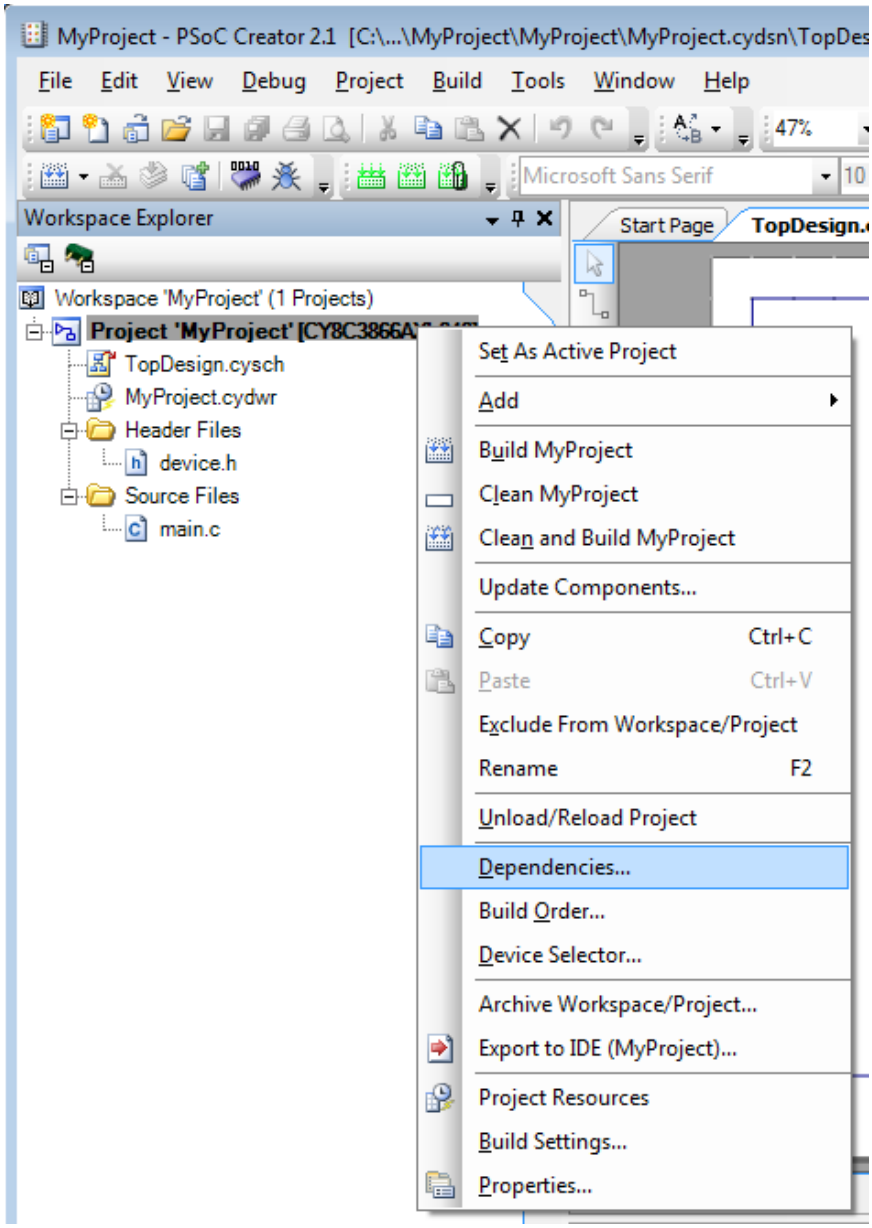
9.2 PSoC Advantage

Almost any microcontroller can do the old trick using a PWM and LPF to create a DAC, so can PSoC, times 50 or more (24 PWMs with dual outputs and 4 fixed function PWMs). Voltage DACs are very common to many controllers, but current DACs are not. Current DACs can prove to be very flexible in optimizing a voltage range to fit your application, not the other way around. DMA is becoming more and more common in high end microcontrollers, but with up to 24 channels, you will always have sufficient channels to implement something such as the DVDAC. Internal opamps to buffer a voltage DAC is not as common, but very useful. If you combine all these common and uncommon features, the PSoC 3 and PSoC 5LP devices have unmatched flexibility compared to any single part solution on the market today.

10 Using these DACs in your Project

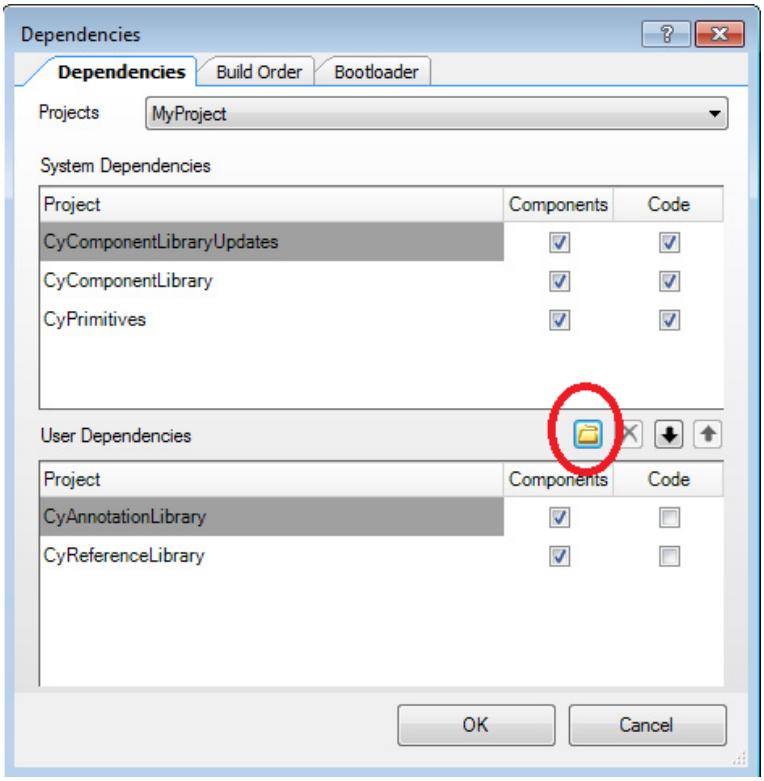
To add any of these DAC components to your project, you must add the HighResDacs library containing these components, as a dependency. To do this, right-click on your project's name in the Workspace Explorer on the left half of the PSoC Creator window. Select the Dependencies option in the pop-up menu as shown in Figure 28.

Figure 28. Select Project Dependency Option



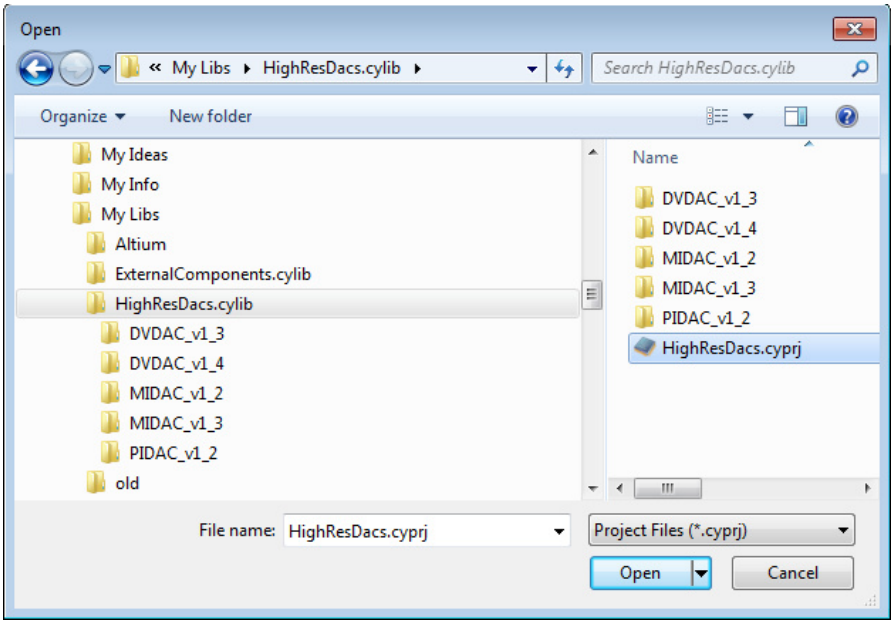
When the Dependencies dialog box opens, press the folder icon for User Dependencies as shown in Figure 29.

Figure 29. Adding a User Dependency



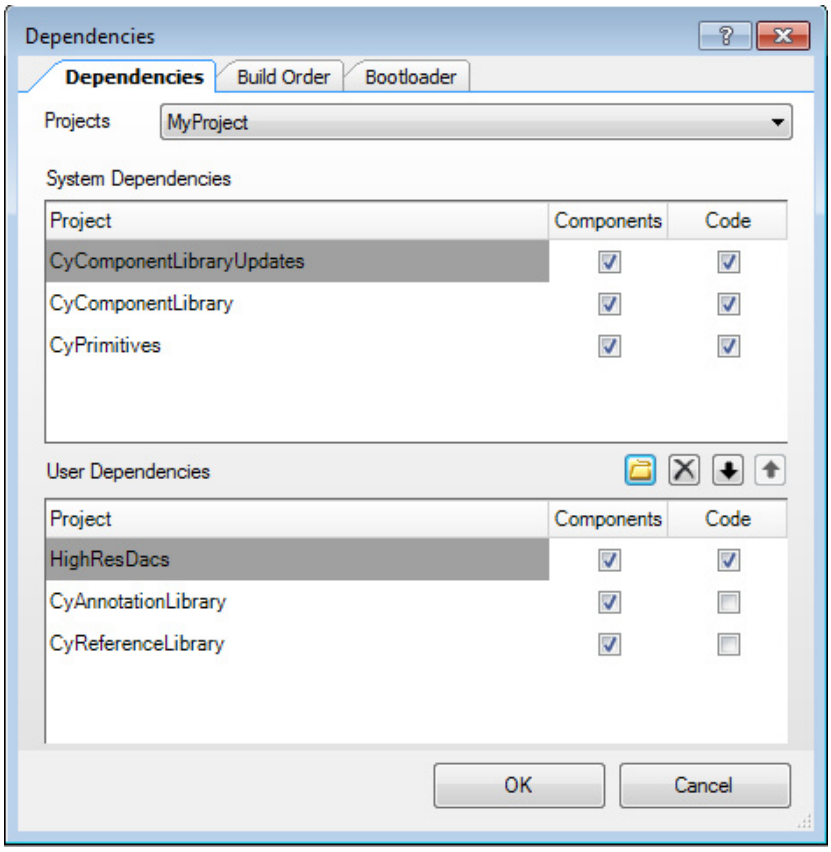
Click the folder icon and then navigate to the folder containing the library in which the high resolution DAC components are located. In this case, it is located in the folder HighResDacs.cylib. Select the file as shown in Figure 30. This will add the library to this project.

Figure 30. Select the DAC library



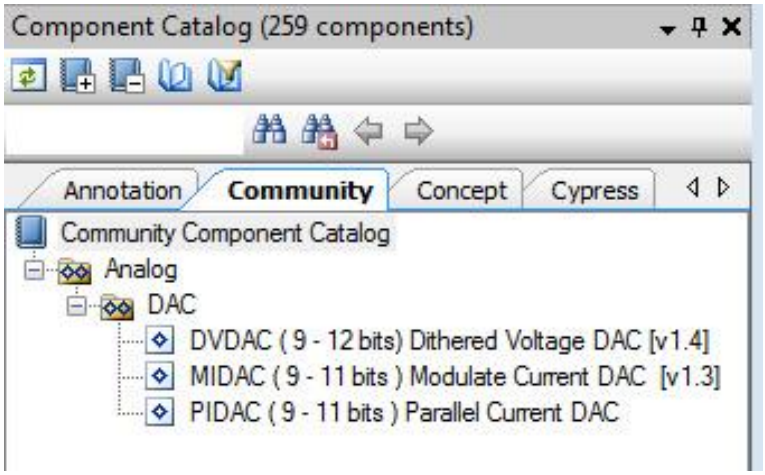
The **Dependencies** dialog box should now look similar to Figure 31, with the DAC library “HighResDacs” added to your project.

Figure 31. DAC Library Added to Project



After you add the library to your project, the Concept tab appears in the Component Catalog on the right side of PSoC Creator. You must be in the schematic entry mode to see the component library. Under that tab, you will see three entries under the Analog/DAC/ component path, as shown in Figure 32.

Figure 32. New DAC Components in Component Catalog



You can now add any of these DAC components to you project just as you add any standard components.

11 Summary

There are many ways to implement DACs or to increase the resolution of an existing DAC. The methods discussed in this application note focused on methods that may be unique to PSoC with its flexible analog and digital structure. The supplied reference components should give you a good head start in experimenting with the different methods and finding out what works best for a given application. These concepts and example components can be used as provided, or modified to fit your application even better. Three example projects are also included with this application note, one for each of the three components (DVDAC, PIDAC, and MIDAC). These projects are identical except for which DAC is used in the project.

As of version 3.0 of PSoC Creator, the DVDAC or “Dithered VDAC” is part of the standard library and no longer requires downloading the library associated with this application note to use it in your project.

About the Author

Name: Mark Hastings.

Title: Applications Engrg MTS

Background: Mark Hastings graduated from Washington State University in 1984. For most of the last twenty five years he has been involved in embedded and mixed signal designs.

Document History

Document Title: AN64275 - PSoC® 3 and PSoC 5LP: Getting More Resolution from 8-Bit DACs

Document Number: 001-64275

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	3095295	MEH	12/03/2010	New application note.
*A	3450302	MEH	11/30/2011	Updated Document Title. Updated Abstract. Updated to new template.
*B	3621214	MEH	05/18/2012	Added Using these DACs in your Project. Updated to new template.
*C	3702848	MEH	08/03/2012	Updated What are INL and DNL? (Updated Figure 1). Several minor changes.
*D	3811902	MEH	11/15/2012	Updated Associated Part Family as “All PSoC 3 and PSoC 5LP parts”. Updated Software Version as “PSoC® Creator™ 2.0 SP1 or later”. Updated Using these DACs in your Project (Updated Figure 28, Figure 29, Figure 30, Figure 31, Figure 32). Replaced PSoC 5 with PSoC 5LP in all instances across the document.
*E	4573105	MEH	11/18/2014	Fixed Contents. Fixed several headers. Updated Summary (To mention new DVDAC in PSoC Creator 3.0).
*F	5044672	MEH	12/10/2015	Updated Software Version as “PSoC® Creator™ 3.2” in page 1. Fixed low resolution graphics. Updated attached associated project. Updated to new template. Completing Sunset Review.
*G	5634981	MEH	03/23/2017	Updated projects to PSoC Creator 4.0 Updated template

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

© Cypress Semiconductor Corporation, 2010-2017. 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.