Implementation of 12-bit delta-sigma DAC with MSC12xx controller

By Hugo Cheung, *MSC Group, Data Acquisition Products* (Email: cheung_hugo@ti.com),
and Sreeja Raj, *MSC Group, Data Acquisition Products* (Email: s-raj2@ti.com)

Introduction

Digital-to-analog converters (DACs) are usually used as an interface between digital systems and continuous analog circuitry. To choose the type of DAC best suited for an application, the designer must consider many important performance measures.

- **Resolution:** Generally, a DAC is specified by the number of bits in the input, or the input width, which represents the number of voltage levels (2^N for an N-bit DAC) that can be generated by the DAC.
- **Full-scale (FS):** If a DAC is implemented to represent the voltages from 0 V to the power-supply voltage, V_{CC}, then the lowest DAC input code should represent 0 V and the highest should represent V_{CC}, or the full-scale voltage. Each analog voltage step of an N-bit DAC is given by

$$V_{LSB} = \frac{FS}{2^N}.$$

- **Output bitstream:** The output of a pulse-width modulator (PWM) or delta-sigma (ΔΣ) DAC is a stream of pulses, referred to as a bitstream, which is passed through a low-pass filter to get the precision analog output voltage (see Figure 1). The frequency of the bitstream decides the complexity and size of the filter design. Higher frequencies will result in smaller filters.
- **Average analog output voltage:** The output of the filter is an analog voltage corresponding to the average on-time of the bitstream input to the low-pass filter. If the frame period (Figure 2) is divided into 2^N parts, the on-time is represented as the number of parts in the frame, where the bitstream is 1. The analog voltage is given by

$$\frac{\text{On-time}}{2^N} \times FS.$$

For example, in Figure 2, the average analog output equals $3/8 \times V_{CC}$. There are various types of DACs based on different methods of conversion. Two of them are:

  – PWM DAC: This is the simplest type of DAC. In this method of conversion, a stream of pulses is passed through a low-pass analog filter, and the width of the pulse is determined by the digital input code. Generally, the implementation compares a sawtooth waveform and the DAC input to produce an output pulse with on-time proportional to the DAC input.

  – ΔΣ DAC: In this method, the output is a stream of pulses of equal width such that the average density of the pulses corresponds to the digital input value. The output stream is then passed through a low-pass filter to produce an analog voltage.

Figure 1. High-level block diagram of a PWM or ΔΣ DAC

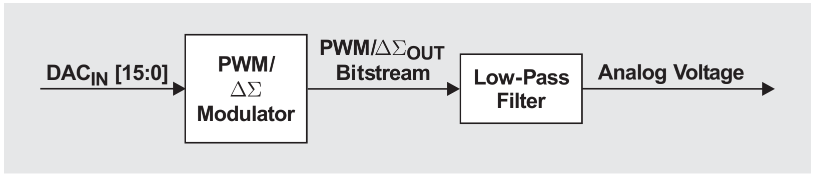
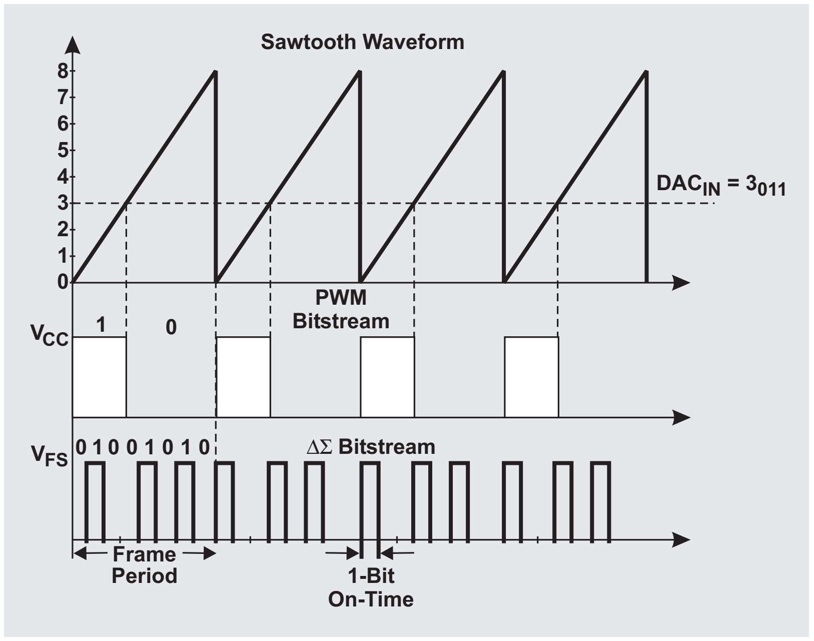


Figure 2. Bitstream output for PWM and ΔΣ DAC for DAC$_{IN}$ = 3_{011}



Design of ΔΣ DAC

This section explains the structure, operation, theory, and implementation of a ΔΣ DAC.

Structure

As the name suggests, a ΔΣ DAC makes computations using binary adders (see Figure 3). Their functionality is as follows:

- **Δ adder:** This adder is used to compute the difference between the DAC input and DAC output. The Δ feedback signal to the Δ adder (Figure 3) depends on the DAC output, which is either a 1 or a 0. If it is a 0, then Δ is an N+2 bit number with all 0s. If it is a 1, then Δ is the 1's complement of the highest N bit number, sign-extended to N+2 bits. It is equivalent to two 1s concatenated as MSBs to an N bit number of all 0s. The DAC_{IN} is an unsigned number; however, since the outputs of both adders represent signed numbers, it is sign-extended. Therefore, the outputs of the Δ adder and Σ adder are signed numbers. For example, in the 3-bit case, the output of the adders is 5 bits. When the DAC input is 0, the output is always 0 V.
- **Σ adder:** This adder is used to compute the sum of the Δ adder output and the current content of the Σ register. The output of the Σ adder is stored in the Σ register. The MSB of the Σ register gives the DAC output (DAC_{OUT}).

Operation

The ΔΣ operation can be explained with a 3-bit example. Table 1 details the bitstream computation steps for a single case, when the DAC_{IN} is equal to 3_{011}. At t_0, the value of Σ is initialized with 10000. The bitstream from t_0 to t_7 has three 1s. The ratio of on-time to frame time is 3⁄8. Therefore, the average analog output voltage is 3⁄8 × FS.

Table 2 shows the bitstream for all the inputs of a 3-bit DAC and their corresponding average analog output voltage.

Theory of ΔΣ conversion

The Σ adder functions like an integrator, which accumulates the input at a rate or slope proportional to the magnitude of the input. When Σ becomes a negative number—i.e., when the MSB equals 1—the Δ error signal is subtracted from Σ such that the accumulated value is reduced to a smaller positive value. Then the integration is continued until the overflow takes place again. The MSB of Σ is the DAC_{OUT}, and the rate at which the MSB becomes 1 is directly proportional to the DAC input. Therefore, the density of 1s in the DAC_{OUT} bitstream is also directly proportional to the input.

Implementation of ΔΣ DAC

There are different approaches for implementing a DAC, depending on the resources used for the computations. It can be completely implemented with hardware only, software only, or a combination of both.

- **Hardware-only implementation:** This is the best possible method in terms of both performance and accuracy. Since all the computations are done by hardwired circuits, this implementation is also the fastest.
- **Software-only implementation:** In this method, the microcontroller is programmed to perform all the operations involved in the ΔΣ conversion. Although this is inexpensive, as it does not use any hardware resources, it exacts huge penalties of lower speed and loss of accuracy from software-induced errors and uncertainties.

Table 1. DAC_{OUT} computation steps for $DAC_{IN} = 3_{011}$

	TIME								
	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
Δ	11000	00000	00000	11000	00000	00000	11000	00000	11000
$Δ_{OUT}$	11011	00011	00011	11011	00011	00011	11011	00011	11011
Σ	10000	01011	01110	10001	01100	01111	10010	01101	10000
$Σ_{OUT}$	01011	01110	10001	01100	01111	10010	01101	10000	01011
DAC_{OUT}	1	0	0	1	0	0	1	0	1

Figure 3. ΔΣ modulator

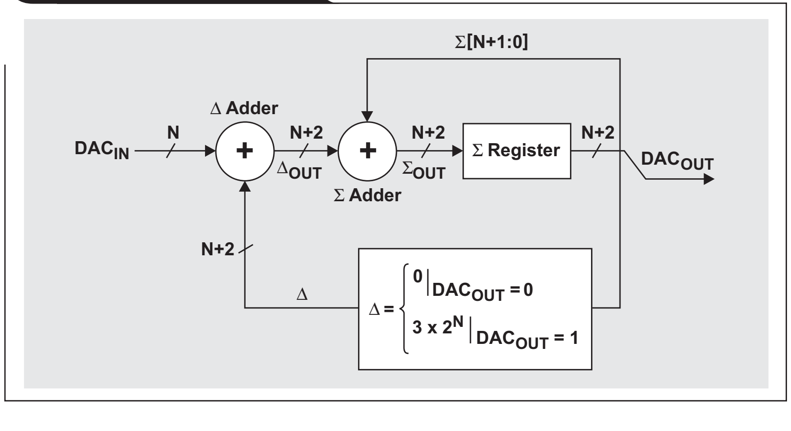

Σ[N+1:0]
Δ Adder
DAC_{IN}
N
N+2
$Δ_{OUT}$
N+2
$Σ_{OUT}$
Σ Adder
Σ Register
N+2
DAC_{OUT}
N+2
Δ
$Δ = \begin{cases} 0 & \vert DAC_{OUT} = 0 \\ 3 \times 2^N & \vert DAC_{OUT} = 1 \end{cases}$

Table 2. Bitstream and average analog output voltage for 3-bit DAC inputs

DAC_{IN}	DAC_{OUT} BITSTREAM $t_{1\ 2\ 3\ 4\ 5\ 6\ 7\ 8}$	ANALOG OUTPUT VOLTAGE (FS = V_{CC}) (V)
0_{000}	0 0 0 0 0 0 0 0	0
1_{001}	0 0 0 0 0 0 1 0	1⁄8 × FS
2_{010}	0 0 1 0 0 0 1 0	2⁄8 × FS
3_{011}	0 1 0 0 1 0 1 0	3⁄8 × FS
4_{100}	1 0 1 0 1 0 1 0	4⁄8 × FS
5_{101}	1 0 1 1 0 1 1 0	5⁄8 × FS
6_{110}	1 1 1 0 1 1 1 0	6⁄8 × FS
7_{111}	1 1 1 1 1 1 1 0	7⁄8 × FS
8_{1000}	1 1 1 1 1 1 1 1	FS

- **Hardware/software co-implementation:** This method, described next, uses part hardware and part software, which helps to minimize the errors induced by software computations. The use of hardware adders improves speed considerably.

Hardware/software co-implementation of ΔΣ DAC with MSC12xx

The implementation method of the ΔΣ DAC described in this article uses both hardware and software resources of the MSC12xx block (see the Appendix on page 32). One of the adders is implemented with the accumulator/shift (ACCSH) block of the MSC12xx (Figure 4). The summation/shift feature of this module can be used only when the ADCON bit is set to 0 in the power-down control register (PDCON), and the summation/shift control register (SSCON) has to be appropriately set to enable summation mode.

The output of the Δ adder is obtained by executing a software algorithm with the microcontroller. If the output of the DAC is 0, then the output of the Δ adder should be the DAC input itself. If the DAC output is 1, then we need to add the 1's complement of all the 1s sign-extended to N+2 bits. For example, the 1's complement of FF_H (for an 8-bit number, sign-extended to N+2 bits) will be 300_H. If this is added to the DAC input, it is equivalent to concatenating two 1s as the MSBs of the DAC input (Table 1). Therefore, we eliminate the Δ adder and use software to generate Δ_{OUT}:

```
If $DAC_{OUT}$ = 0
        $\Delta_{OUT}$ = $DAC_{IN}$;
else
        $\Delta_{OUT}$ = Concatenate 11 with $DAC_{IN}$;
end if;
```

The Σ adder of the ΔΣ modulator is implemented directly with the hardwired adder in the ACCSH block.

The next step is to send the DAC output to a port of the device so it can be filtered and used for some applications (see Figure 5). There are several means of writing the output of the DAC to an output port of the MSC12xx. The serial peripheral interface (SPI) output can be used to observe the DAC output as a series waveform. The SPI can be run either in the master mode, where it derives a clock

Figure 4. Simplified block diagram of ACCSH block

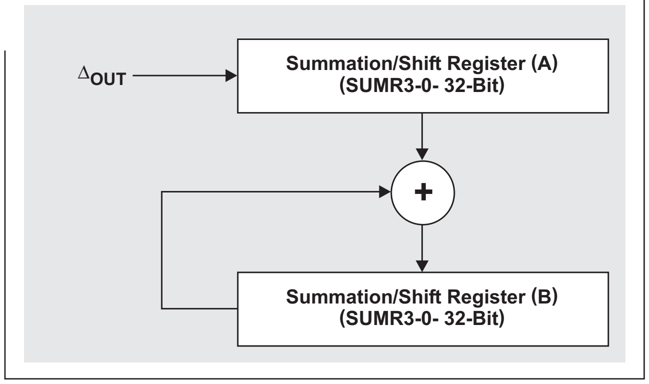
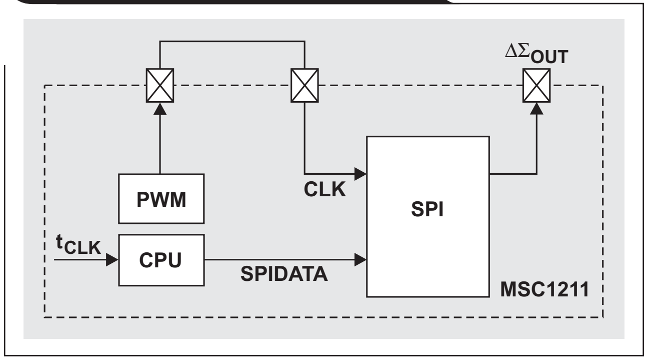

Δ_{OUT}
Summation/Shift Register (A) (SUMR3-0- 32-Bit)
+
Summation/Shift Register (B) (SUMR3-0- 32-Bit)

Figure 5. SPI port for DAC output



from the CPU itself, or in the slave mode, where the user can provide the clock or the clock can be generated from a timer or PWM. The disadvantage of running the SPI in the master mode is that some clock cycles may be wasted, as the SPI clock is a discrete multiple of the CPU clock speed in the master mode. With the CPU running with a 22-MHz crystal, it takes about 50% of the CPU's time to run the output bitstream at 150 kHz.

Comparing ΔΣ DAC with PWM DAC

To compare the performance of the ΔΣ DAC, a PWM DAC was implemented with software and the microcontroller on the same MSC12xx board. In the implementation, a counter was used to count from 0 to 2^N–1, and the count value was compared with the DAC input after each increment. The DAC output was maintained at 1 until the counter exceeded the DAC input, when it was pulled down to 0. When the counter expired, the DAC output was set back to 1; so a series of pulses was generated with an on-time proportional to the DAC input. The disadvantage of this type of DAC is that, since the frequency of the output bitstream is the same as the frame rate, it is not possible to achieve very high frequencies for the output as compared with the ΔΣ DAC. Therefore, to obtain an analog output voltage, we will need to design filters with large time constants.

For a fair comparison, we chose a PWM implemented with a timer running at a frequency of 10 MHz. For a 12-bit DAC, the frame rate would be equal to 2.5 kHz. With a ΔΣ DAC, more than 90% of the DAC_{IN} codes will result in a Δ_{OUT} bit frequency greater than 8 kHz. Therefore, the ΔΣ DAC output filter design is much easier than for a PWM DAC. Hence, the disadvantages of using a PWM DAC are as follows:

- Requires larger filters.
- For very low/high codes, the PWM on/off-time might not accurately represent the code because there might be software overhead that causes the on/off-time to be greater than the exact time representing the code.
- Some devices don't have double-buffered timers. This will cause some software uncertainties during the duty-cycle transition in the PWM DAC, which might result in poor integral/differential nonlinearity (INL/DNL) performance.

Filter design

In the DACs previously described, the DAC outputs are always a stream of pulses. To generate an analog output voltage corresponding to the digital input, the pulse stream is passed through an analog low-pass filter. The filter output is the average signal level of the pulse stream.

There are several issues in the design of an RC low-pass filter, including attenuation of high-frequency components, settling time, and INL/DNL performance. These issues will be discussed in detail next, with a DAC of 12-bit resolution used as an example.

Attenuation of high-frequency components

The maximum ripple allowed in the filtered output has to be less than the voltage corresponding to 1 LSB (see Table 3). For a 12-bit DAC, with the highest voltage (FS) equal to 5 V, this value is

$$20\log\left(\frac{1}{2^{12}}\right) = -72 \text{ dB.} \qquad \textbf{(1)}$$

Table 3. Required attenuation

NO. OF BITS (N)	MAX RIPPLE (1 LSB WITH FS = 5 V) (V)
10	0.0048
12	0.00122
14	0.00031
16	0.000076

This attenuation should be attained at the lowest possible frequency. To keep the filter design reasonable, we assume that the DAC input is always above 5% of the total number of codes (2^N for an N-bit DAC). For a 12-bit DAC, this value is given by CC_H. For the specifications of the board and crystal chosen for the design, the SPI clock rate (bitstream frequency) was found to be 150 kHz, and the frequency corresponding to CC_H was 8.9 kHz. The RC time constant of the filter is calculated as follows:

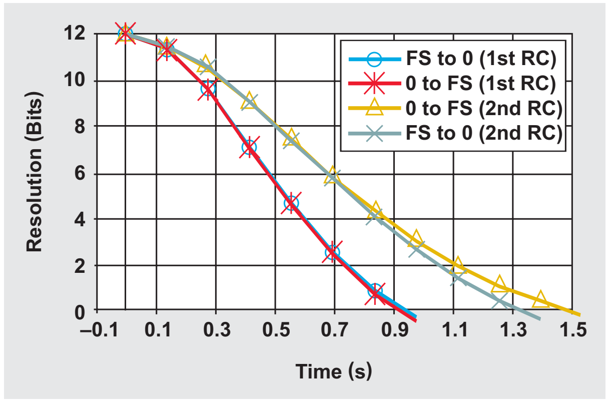
$$H(\omega) = \left|\frac{V_{OUT}}{V_{IN}}\right| = \frac{1}{\sqrt{1+(\omega RC)^2}} \qquad \textbf{(2)}$$

Solving for RC with Equations 1 and 2, we get RC = 0.0712 s.

Settling time

A very important parametric of the DAC performance is the settling time, which is the time required to settle within the range of 1-LSB voltage without error. The required settling times for 10-, 12-, 14-, and 16-bit DACs are shown in Table 4. The settling time is computed as a factor of the RC time constant. The factor is the number of time constants required to settle to a 1-LSB value. Figure 6 shows the DAC settling time versus bit resolution when the DAC input is changed from 0 to FS and vice versa.

Figure 6. DAC settling time vs. bit resolution



INL/DNL performance

The filter design under discussion is a very simplified form of the output circuitry of a ΔΣ DAC. If the output voltage is measured with only this filter, then a significant degradation in INL/DNL performance is expected. To achieve the desired INL/DNL performance, certain auxiliary circuits have to be designed along with the filter to minimize output transistor resistance and digital power-supply noise (see Figure 7). In addition, for impedance matching, a buffering circuit should follow the filter before the analog voltage-measuring instrument. The design of these analog circuits is briefly discussed here.

The digital bitstream from the DAC (denoted as D_{IN}) is passed through the conditioning circuit, resulting in an analog voltage output that is used for INL/DNL measurements. The optical coupler HCPL-0630 is used to provide isolation between the digital and analog sides, which

Table 4. Theoretical calculations for settling times for DACs with different bit resolutions

NO. OF BITS (N)	5% OF 2^N	BITSTREAM FREQUENCY (kHz)	RC TIME CONSTANT (s)	TYPICAL NO. OF RC TIME CONSTANTS REQUIRED	SETTLING TIME (s)
10	33_H	8.9	0.0179	8	0.143
12	CC_H	8.9	0.0712	9	0.641
14	333_H	8.9	0.283	11	3.1176
16	CCC_H	8.9	1.128	12	13.54

significantly reduces the impact of digital noise on the analog signal. The isolated signal is then passed through an inverter and the RC filter. Finally, the filtered output is passed through a voltage-follower op amp buffer to yield an analog voltage in the range of 0 to 5 V. The op amps use the 9-V supply voltage, but additional circuitry is required to generate a 5-V supply for the inverter and optical coupler. The results of the INL/DNL calculation are plotted in Figures 8 and 9. We can see that the conditioning circuit results in excellent DNL performance (±0.2 LSB) but does not help to correct INL errors (±0.6 LSB) to a large extent. The INL performance is degraded because of the different gains the RC filter offers to the higher bitstream frequencies and those closer to 0. The filter needs to have flatter frequency response in the passband. To improve the INL performance, another RC filter was cascaded, thus forming a second-order filter. Although this degraded the settling-time performance, it helped reduce the INL error to ±0.2 LSB and the DNL error to ±0.15 LSB.

Figure 7. Conditioning circuit to improve INL/DNL performance

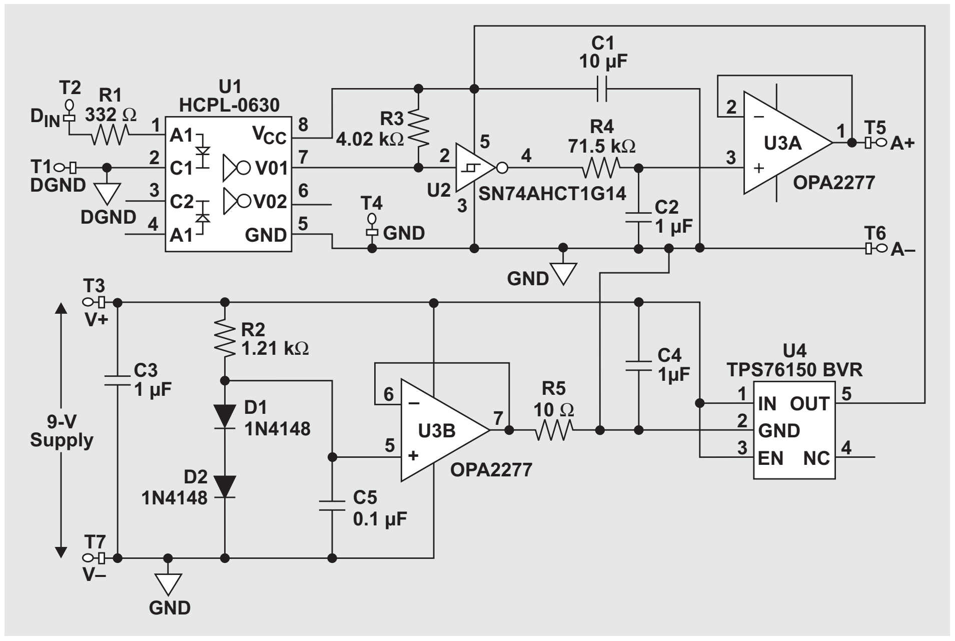


Figure 8. DNL measurements for ΔΣ DAC

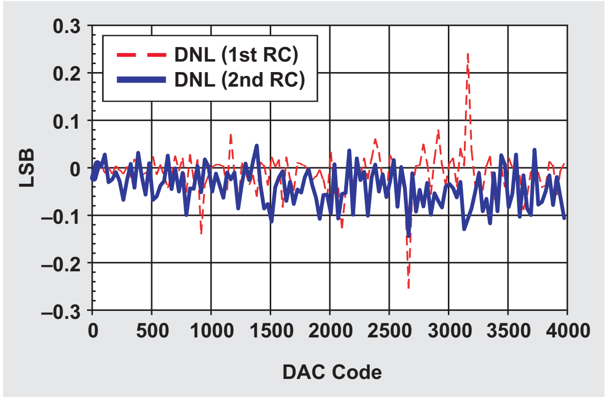
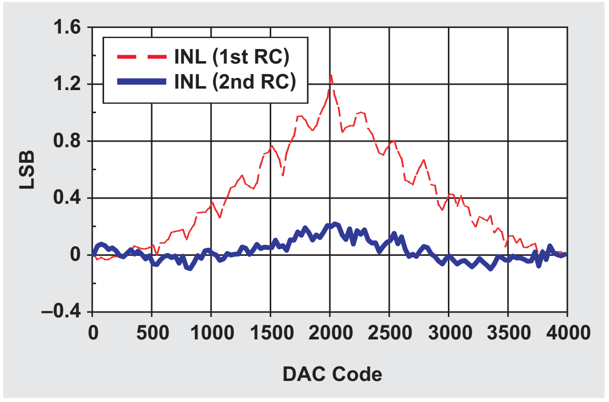


Figure 9. INL measurements for ΔΣ DAC



Conclusion

A ΔΣ DAC is implemented on an MSC1211 microcontroller board with both hardware and software resources. The Δ adder is implemented with software, and the Σ adder is implemented with the hardware adder on the MSC1211 board. The design results in a very efficient ΔΣ DAC, as it offers much better speed and error-free performance as compared with a software-only implementation. Also, with a higher bit rate, the frequency spectrum is better than that of a PWM DAC, resulting in smaller filters and faster settling times.

Related Web sites

analog.ti.com

www.ti.com/sc/device/*partnumber*

Replace *partnumber* with MSC1211Y2, OPA2277, SN74AHCT1G14, or TPS76150

Appendix—C program for 12-bit ΔΣ DAC with MSC12xx

```
#include <REG1210.H>
#include <stdio.h>
void autobaud ();
sbit REQ = P3^7;
sbit ACK = P3^6;
union intU
{
        int i;
        struct{
                unsigned char b1;
                unsigned char b0;
        } byt;
};
union intU dacbuf;
void INT0_isr(void) interrupt 0
{
        T0=1;
        // INL & DNL test
        /*      if ((dacbuf.i % 2)==0) dacbuf.i+=1;
        else dacbuf.i+=31;
        if(dacbuf.i==4096)  dacbuf.i=0;
        printf("\n**%d**\n",dacbuf.i); */
        // Step response test
        if (dacbuf.i==0) dacbuf.i=4095; else dacbuf.i=0;
}
void main(void)
{
        unsigned char lut[8] = {1,2,4,8,16,32,64,128};
        unsigned char  outbuf, bitcnt;
        autobaud();
        // Init SPI
        PDCON = 0x66;
        P1DDRH = 0xDD;          //b11011101;
        SS = 0;
        SSCON = 0;              //Clear summation registers
        SSCON = 0x10;           //Enabling summation mode of the ACCSH register
        SPICON = 0x00;          //Setting the SPI control to slave mode
        SPITCON = 0x08;         //Setting drive immediately
        SPIDATA = 0x00;
        //Init PWM, used for SPI clk, connect PWM o/p to SCLK
        PWMCON = 0x09;          // Set period, SysClk source, PWM mode
        PWM =  0x0085;          // PWM period
        PWMCON = 0x00;          // Disable PWM/tone
        PWMCON = 0x19;          // Set duty, SysClk source, PWM mode
        PWM =  0x0042;          // PWM duty
        // INT0 edge interrupt
        printf("Delta Sigma     DAC\n");
        dacbuf.i=0;
        EX0=1; IT0=1;
        EA=1;
        while(1) {
                        bitcnt = 0;
                        outbuf = 0;
                        for(bitcnt=0;bitcnt<8;bitcnt++) {
                                if ((SUMR1&0x20) != 0){
                                        SUMR1 = 0x30 | dacbuf.byt.b1;
                        outbuf = outbuf | lut[bitcnt];
                        //The i-th bit of OUTBUF is forced to 1
                                } else  SUMR1 = dacbuf.byt.b1;
                                SUMR0 = dacbuf.byt.b0;
                        }
                        while (!(AIE & 0x08)) {} // Wait for SPI TX empty
                        SPIDATA = outbuf;
                        outbuf = SPIDATA ; // Clear SPI RX buf

        } //Main
```

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products	
Amplifiers	**amplifier.ti.com**
Data Converters	**dataconverter.ti.com**
DSP	**dsp.ti.com**
Interface	**interface.ti.com**
Logic	**logic.ti.com**
Power Mgmt	**power.ti.com**
Microcontrollers	**microcontroller.ti.com**

Applications	
Audio	**www.ti.com/audio**
Automotive	**www.ti.com/automotive**
Broadband	**www.ti.com/broadband**
Digital control	**www.ti.com/digitalcontrol**
Military	**www.ti.com/military**
Optical Networking	**www.ti.com/opticalnetwork**
Security	**www.ti.com/security**
Telephony	**www.ti.com/telephony**
Video & Imaging	**www.ti.com/video**
Wireless	**www.ti.com/wireless**

TI Worldwide Technical Support

Internet

TI Semiconductor Product Information Center Home Page
support.ti.com

TI Semiconductor KnowledgeBase Home Page
support.ti.com/sc/knowledgebase

Product Information Centers

Americas

Phone	+1(972) 644-5580	Fax	+1(972) 927-6377
Internet/Email	support.ti.com/sc/pic/americas.htm		

Europe, Middle East, and Africa

Phone			
Belgium (English)	+32 (0) 27 45 54 32	Netherlands (English)	+31 (0) 546 87 95 45
Finland (English)	+358 (0) 9 25173948	Russia	+7 (0) 95 7850415
France	+33 (0) 1 30 70 11 64	Spain	+34 902 35 40 28
Germany	+49 (0) 8161 80 33 11	Sweden (English)	+46 (0) 8587 555 22
Israel (English)	1800 949 0107	United Kingdom	+44 (0) 1604 66 33 99
Italy	800 79 11 37		
Fax	+(49) (0) 8161 80 2045		
Internet	support.ti.com/sc/pic/euro.htm		

Japan

Fax			
International	+81-3-3344-5317	Domestic	0120-81-0036
Internet/Email			
International	support.ti.com/sc/pic/japan.htm		
Domestic	www.tij.co.jp/pic		

Asia

Phone			
International	+886-2-23786800		
Domestic	Toll-Free Number		Toll-Free Number
Australia	1-800-999-084	New Zealand	0800-446-934
China	800-820-8682	Philippines	1-800-765-7404
Hong Kong	800-96-5941	Singapore	800-886-1028
Indonesia	001-803-8861-1006	Taiwan	0800-006800
Korea	080-551-2804	Thailand	001-800-886-0010
Malaysia	1-800-80-3973		
Fax	886-2-2378-6808	Email	tiasia@ti.com
Internet	support.ti.com/sc/pic/asia.htm		ti-china@ti.com

C011905

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