## Analog and Mixed-Signal Products

# Analog Applications Journal

First Quarter, 2005

Texas Instruments

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

| **Products** | | **Applications** | |
|---|---|---|---|
| Amplifiers | amplifier.ti.com | Audio | www.ti.com/audio |
| Data Converters | dataconverter.ti.com | Automotive | www.ti.com/automotive |
| DSP | dsp.ti.com | Broadband | www.ti.com/broadband |
| Interface | interface.ti.com | Digital control | www.ti.com/digitalcontrol |
| Logic | logic.ti.com | Military | www.ti.com/military |
| Power Mgmt | power.ti.com | Optical Networking | www.ti.com/opticalnetwork |
| Microcontrollers | microcontroller.ti.com | Security | www.ti.com/security |
| | | Telephony | www.ti.com/telephony |
| | | Video & Imaging | www.ti.com/video |
| | | Wireless | www.ti.com/wireless |

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

# Contents

**To view past issues of the**
***Analog Applications Journal*, visit the Web site**
**www.ti.com/sc/analogapps**

# Introduction

*Analog Applications Journal* is a collection of analog application articles designed to give readers a basic understanding of TI products and to provide simple but practical examples for typical applications. Written not only for design engineers but also for engineering managers, technicians, system designers and marketing and sales personnel, the book emphasizes general application concepts over lengthy mathematical analyses.

These applications are not intended as “how-to” instructions for specific circuits but as examples of how devices could be used to solve specific design requirements. Readers will find tutorial information as well as practical engineering solutions on components from the following categories:

- Data Acquisition
- Power Management

Where applicable, readers will also find software routines and program structures. Finally, *Analog Applications Journal* includes helpful hints and rules of thumb to guide readers in preparing for their design.

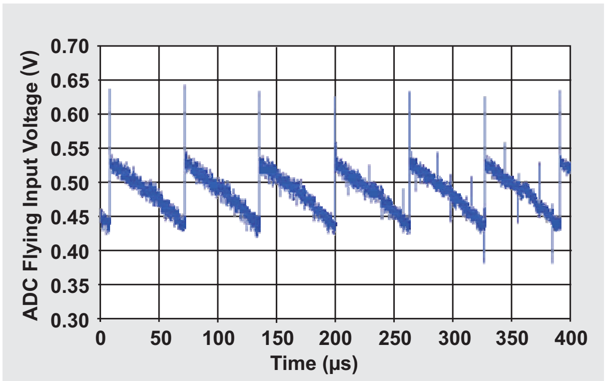
# Supply voltage measurement and ADC PSRR improvement in MSC12xx devices

**By Michael Gurevich** (Email: gurevich_michael@ti.com)
*Systems Engineer*

## Introduction

This article describes a simple method to measure the analog supply voltage in Texas Instruments MSC12xx devices with one of the unconnected ADC inputs of the ΔΣ ADCs. The ability to measure the supply's voltage change permits the ADC gain to be adjusted, which leads to a better ADC power supply rejection ratio (PSRR). This method is applicable to the MSC1210, MSC1211/1212/1213/1214, MSC1200/1201/1202, and ADS1216 families.

**Figure 1. ADC flying input voltage taken with 10-MΩ, 8-pF probe**



$f_{MOD}$ = 15.625 kHz, buffer is off, programmable gain amplifier (PGA) = 1, $AV_{DD}$ = 5 V, and the opposite ADC input is grounded

During the ΔΣ ADC conversion cycle, the input signal is sampled on the internal capacitors at the modulation frequency ($f_{MOD}$). Switching capacitor techniques are then used to compare the input voltage to the integrator voltage, eventually leading to the ADC result. Reference 1 provides a detailed description of how the ADC input circuits work. What is important here is that when the next sampling occurs, the sampling capacitor is not fully discharged but has a voltage proportional to the analog supply voltage, $AV_{DD}$. This weak voltage can be seen with a high-impedance (>10 MΩ) voltmeter or oscilloscope. When the oscilloscope probe is connected to the ADC input selected by the internal program and the ADC is running with the internal buffer disabled, the oscilloscope shows a voltage in the range of 0.3 to 0.7 V as shown in Figure 1.

## Measuring the flying ADC inputs

If both ADC differential inputs are disconnected from the external circuits, then both nodes will have the same potential and the measured differential voltage will be zero. But if one of the inputs is grounded, the conversion result will no longer be zero and will remain stable until the analog power supply is changed. These measurements can be done only if the ADC input buffer is disabled. If the buffer is on, the high-impedance buffer inputs fluctuate due to small-pin leakages, which force the buffer outputs to follow and suppress the voltage coming from the ADC inputs. The next natural question is: What affects the ADC readings in this condition?

## What affects the flying input readings?

Experiments were performed to determine whether the ADC results are affected by any of the following factors:

- ADC modulator frequency
- ADC input sampling frequency (sampling frequency changes when the programmable gain amplifier [PGA] setting is changed)
- Decimation frequency
- Chosen ADC inputs
- Device clocking frequency
- Reference voltage and reference voltage common mode
- Device temperature
- Proximity of clocks or other noise sources

In all cases it was found that if one ADC input is grounded and the opposite input is flying, the ADC readings are stable and don't show any dependence on the preceding parameters. But if the flying input has some resistance or capacitance connected to it, the ADC readings change because the additional dc and ac paths affect the flying input voltage (see Figure 1). The ADC reading also has a minor dependence on the digital supply voltage. When the digital supply changes from 5 V to 3 V, the ADC readings change less than 1%; therefore, the ADC readings mainly depend on the analog supply voltage, $AV_{DD}$. A stable ADC reading does not mean that flying input voltage measured with an external voltmeter is constant; it means that all these changes are self-compensated during conversion and lead to the stable ADC result.

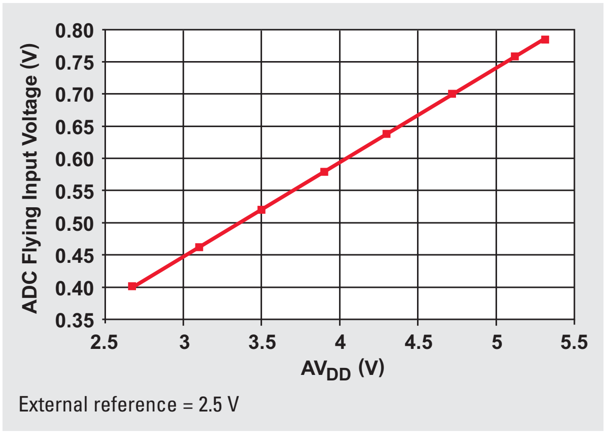
## ADC flying input readings and analog power supply voltage

When the dependence of the ADC flying input readings on the analog supply voltage was investigated, it was found that in all MSC12xx devices, the ADC readings have a strictly linear relationship to the analog supply voltage:

$$V = \frac{AV_{DD}}{KS}, \tag{1}$$

where V is the ADC flying input reading and KS is a proportional coefficient. KS varies slightly from device to device and between device types and packages. The average KS value is around 6.9. Figure 2 shows the ADC flying input readings for the MSC1210.

Figure 2. MSC1210 ADC flying input readings from $AV_{DD}$ supply



Equation 2 shows the correlation of the flying voltage on an unused ADC input to the analog supply voltage, $AV_{DD}$:

$$AV_{DD} = KS \times V \tag{2}$$

In Table 1, the KS value and its variations are determined for different MSC12xx devices. As you can see, the KS value slightly depends on the device type, as some devices have different internal layouts, different packages, and pins with different parasitic capacitances. Different device packages are also forced to use different types of testing boards.

It is necessary to understand that flying input measurements are absolute-value measurements, and therefore the precision of $V_{REF}$, ADC offset and gain calibration plays some role in the results.

**Table 1. Value of coefficient KS and its variations for different devices***

| DEVICE | NUMBER OF DEVICES TESTED | KS (AVERAGE SLOPE) | KS VARIATION (±%) |
|---|---|---|---|
| MSC1200 | 35 | 6.818 | 3.6 |
| MSC1201 | 5 | 7.014 | 1.1 |
| MSC1210 | 30 | 6.911 | 2.4 |
| MSC1211 | 10 | 6.843 | 2.1 |

*KS is determined with external reference voltage.

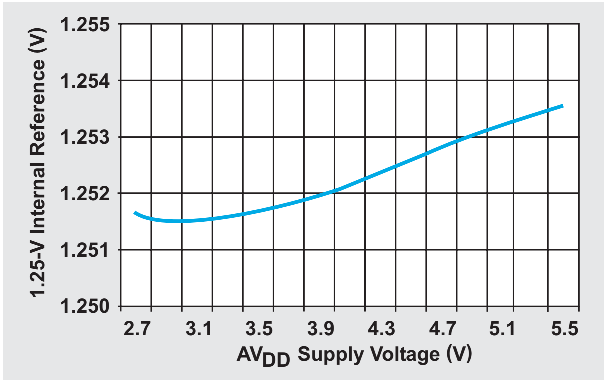
Table 1 also shows that KS variation between devices is around ±2.5%. Therefore, the KS value that determines $AV_{DD}$ for one user board can apply to all other similar boards with ±2.5% precision. Usually there is no need to track the supply voltage with a precision better than ±5%, and 2.5% is usually sufficient. The KS values shown in Table 1 are a good starting point to investigate the KS magnitude for the user board and settings. If an internal reference is used, remember that it has some supply dependence, which can affect the KS value.

The flying input voltage can also be used to detect whether the ADC inputs are still connected to the external signal source.

## Compensating for changes in internal reference voltage

The typical ADC PSRR value for MSC12xx devices is around 85 dB. However, the internal reference PSRR is much lower, around 60 dB, which will restrict the whole system PSRR if an internal reference is used. Usually the internal reference has a nearly linear dependence on the analog supply voltage (see Figure 3).

Figure 3. Typical MSC1210 1.25-V internal reference dependence on analog supply voltage



This linear dependence allows us to estimate the internal reference voltage change from the supply voltage change. Since the ADC flying input gives us a good tool to measure the supply voltage change, we can calculate the new ADC gain value to compensate the ADC reference shift:

$$GAIN_n = GAIN_c[1+(SPLY_n - SPLY_c)\times KR], \tag{3}$$

where

$GAIN_c$ is the data value loaded in the ADC gain calibration register (GCR) during ADC self-calibration with a stable supply voltage ($AV_{DD} = V_c$);

$SPLY_c$ is the ADC flying input data value after ADC self-calibration with $AV_{DD} = V_c$;

$SPLY_n$ is the ADC flying input data value with the new supply voltage, $V_n$ (the GCR still has the $GAIN_c$ value);

$GAIN_n$ is a new ADC gain for the GCR for supply voltage $V_n$; and

KR is a proportional coefficient, defined later in Equation 4.

The sequence of actions is this: After a reset, when the power supply is stabilized at $V_c$ and the device is warmed up, the program makes the ADC self-calibrate and copies the GCR value to the variable $GAIN_c$. Then the ADC flying input, $SPLY_c$, is measured; and the user program starts its routines. When the supply voltage has been changed to $V_n$, the program measures the new flying input voltage, $SPLY_n$. If it is different from the stored $SPLY_c$ value, the program calculates the new $GAIN_n$ value with Equation 3 and loads it to the GCR. Since the program stores the original $GAIN_c$ and $SPLY_c$ values, this procedure can be repeated many times.

There is an unknown constant, KR, which can be derived from Equation 3:

$$KR = \frac{\frac{GAIN_n}{GAIN_c} - 1}{SPLY_n - SPLY_c} \quad \textbf{(4)}$$

Even with the plot in Figure 3, it is possible to make only a coarse estimation for the KR value because of variations from device to device. Also affecting the KR value is the ADC PSRR; and very often the same supply used for $AV_{DD}$ is used to power the signal external buffer, which also has some PSRR, and so on. Much better results can be achieved if a stable external control signal, independent of $AV_{DD}$ supply changes, is used to determine the working KR value. Under these conditions, Equation 4 can be converted to Equation 5:

$$KR = \frac{\frac{CNTR_c}{CNTR_t} - 1}{SPLY_t - SPLY_c}, \quad \textbf{(5)}$$

where

$CNTR_c$ is the ADC control signal data value after ADC calibration with $AV_{DD} = V_c$;

$CNTR_t$ is the control signal data value after $AV_{DD}$ has been changed to some new voltage $V_t$; and

$SPLY_t$ is the ADC flying input data value for $V_t$.

This approach makes it fast and easy to find the precise value for the KR coefficient, which itself includes all device features and environment variations. For better KR precision, it is recommended that $V_c – V_t$ be at least 1 V and that the control signal value be bigger than REF/2. For MSC1200 parts, the typical KR value is $1.076 \times 10^{-9}$.

In addition to ADC supply compensation, this approach can be used to compensate ADC errors due to other factors if there is a linear dependence between ADC readings and the other factors. System temperature is a good example of such ADC gain compensation.

Table 2 shows the ADC PSRR with a 1-V input signal and the 1.25-V internal reference for the MSC1200 and MSC1210. $AV_{DD}$ was changed from 5 V to 4 V during the PSRR measurements. To find the exact KR value with Equation 5, a 1-V control signal was used, and $AV_{DD}$ was changed from 5 V to 3 V.

**Table 2. ADC PSRR with and without ADC gain correction**

| DEVICE NUMBER | ADC PSRR WITHOUT GAIN CORRECTION (dB) | ADC PSRR AFTER CORRECTION (dB) | PSRR DIFFERENCE (dB) |
|---|---|---|---|
| MSC1200 #1 | 51.83 | 72.76 | 20.93 |
| MSC1200 #2 | 52.56 | 72.58 | 20.02 |
| MSC1200 #3 | 53.88 | 81.42 | 27.54 |
| MSC1210 #1 | 63.53 | 80.91 | 17.38 |
| MSC1210 #2 | 68.63 | 93.97 | 25.34 |
| MSC1210 #3 | 66.74 | 79.16 | 12.43 |

## USupply program

The “C” program beginning on page 9 provides a way to verify the precision of the method discussed and to find the constants KS and KR. Communication with the program is via the device and computer serial port. By typing one-character commands and comparing the program response with real voltages applied to the device, the user can estimate the degree of precision. If necessary the program can be easily adjusted to the desired clock frequency, ADC decimation value, average number of ADC samples, reference voltage used, and input pins used. To work with the USupply program it is necessary to have the adjustable power supply (in the 2.5- to 6-V range), the source of stable dc voltage (in the 1- to 2.5-V range), and a dc voltmeter.

The ADC input pin AINCOM should be grounded, AIN4 should be flying (not connected), and AIN5 should be connected to the control signal (dc voltage, 1 V recommended). The default program clocking frequency is 1.8432 MHz. The user can switch the program to another clocking frequency but should remember that, for $DV_{DD}$ = 2.7 V, the maximum clock frequency is less than 12 MHz. The program uses a 1.25-V internal reference to give the user more room to change the supply voltage.

The default version of the program works with MSC1200/1201 devices. For the MSC1210/1211 devices, the code in the `Wait()` and `ADC_Avrg()` routines should be altered according to the comments in the routines.

After reset, the program uses the default values for the KR and KS constants. By manipulating program commands and the power supply, the user will be able to find the KR and KS values that best match the device environment.

For the ADC result averaging, the program uses the summation register. Constants are defined to simplify the summation register setup. The number of averages is chosen by defining the AVRG constant to be equivalent to either AVRG4, AVRG8, AVRG16, or AVRG256, which represents averaging of 4, 8, 16, or 256 points, respectively.

The program code size is slightly less than 8 Kbytes, which means that devices with Flash memory version Y3, Y4, or Y5 should be used.

### Routines

`Wait(cycles)`: Delays the number of ADC data cycles.

`ADC_Avrg()`: Averages the ADC result as specified in the AVRG constant with the ADC summation register. The subroutine returns the measured voltage as a float variable. The averaged ADC code is located in the global `u.lng` variable. The ADC is used in unipolar mode.

`ADCsupply()`: Measures the flying input AIN4 voltage and calculates the $AV_{DD}$ supply voltage with the KS constant routine. Reports both voltages to the PC and returns to the main program ADC code averaging for flying input measurements.

`ADCcntrl()`: Measures the control voltage between inputs AINCOM and AIN5. Converts it to volts and reports to the PC. Returns to the main program averaged ADC code.

### Main program

After reset, the USupply program expects the "CR" character from the PC serial port and, after receiving it, performs ADC self-offset and gain calibration. It then shows a greeting message on the screen. When the user types "H" for "Help," the program displays the following options:

```
Enter
'S' To check Supply voltage
'M' To Measure control signal
'A' To Adjust ADC gain
'C' ADC self offset and gain Calibration
'F' Find ADC gain cor.coef.
'V' Find supply Voltage coef.
```

Entering "S" will cause the program to measure the supply voltage using the flying input method and the KS coefficient. The user can compare the result with the real $AV_{DD}$ voltage. If option "V" was not used before, the program uses Equation 2 with a default $KS_z$ value.

Entering "M" will cause the program to measure the control signal on pin AIN5 and report its current voltage to the PC.

Entering "A" will cause the program to adjust the ADC GCR value using the KR coefficient. The magnitude of the adjustment depends on how much the supply voltage $AV_{DD}$ has been changed from the supply level $V_c$, which existed during ADC calibration. This option should be used after ADC calibration is done and $AV_{DD}$ has changed to $V_n$. The program reports to the PC the old ADC gain, the adjustment coefficient, and the new ADC gain. If option "F" was not used before, the program performs an adjustment with a default $KR_z$ value.

Entering "C" will cause the program to perform ADC self-offset and gain calibration. The program reports the ADC GCR value to the PC, copies it to $GAIN_c$, and measures and stores the ADC flying input and control signal readings.

Entering "F" will cause the program to find the KR coefficient with supply voltage $V_n$. For this purpose the program uses the control signal readings with supply voltages $V_c$ and $V_n$. The new KR value is reported to the PC, and with it the ADC gain adjustment (option "A") can then be more precisely performed.

Entering "V" will cause the program to find the supply proportional coefficient, KS. The program asks the user to enter the current value of $AV_{DD}$ in volts, then it measures the flying input voltage and calculates the precise value for KS. The updated KS value is also reported to the PC and then used with subsequent supply voltage measurements.

## Conclusion

The suggested method for measuring analog supply voltage in MSC12xx devices without additional external parts can be important in many battery applications. When the device current power supply voltage is known, the method allows adjustment of the ADC gain, which provides better ADC PSRR, especially in the cases with an internal ADC reference. The same method can be used to compensate system gain change due to temperature variations. The USupply program enables the user to verify the method and precision of measurements for the desired boards and settings and to find the KS and KR values.

## References

For more information related to this article, you can download an Acrobat Reader file at www-s.ti.com/sc/techlit/*litnumber* and replace *"litnumber"* with the **TI Lit. #** for the materials listed below.

| **Document Title** | **TI Lit. #** |
|---|---|
| 1. Joseph Wu, "Input Currents for High-Resolution ADCs," Application Report | sbaa090 |
| 2. Michael Gurevich, "ADC Offset in MSC12xx Devices," Application Report | sbaa097 |
| 3. Michael Gurevich, "ADC Gain Calibration—Extending the ADC Input Range in MSC12xx Devices," Application Report | sbaa107 |

## Related Web sites

**analog.ti.com**

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

Replace *partnumber* with ADS1216, MSC1200Y2, MSC1201Y2, MSC1202Y2, MSC1210Y2, MSC1211Y2, MSC1212Y2, MSC1213Y2 or MSC1214Y2

## USupply program

```
/* 1.08 ##################### Usupply ##############################
 Texas Instruments, G.M. September 2004;   MSC1200/MSC1210;
 To work with MSC1210/1211 parts, the code should be changed in the
 <ADC_Avrg> and <Wait> subroutines (see comments there).
 The program demonstrates part AVdd measurement using the ADC flying input.
 It also adjusts ADC gain when AVdd is changed.
 Internal 1.25 V reference. ADC buffer off. Unipolar ADC mode.
 Communication through UART0. After reset hit the 'Enter' key.
 --------------- Pins --------------------------------
 AINCOM        Grounded
 AIN5          Control signal input, always positive, bigger then REF/2
 AIN4          Flying input, used to measure AVDD
 --------------- Commands ----------------------------
 "S" Measure supply voltage with flying AIN4
 "M" ADC converts the control signal at AIN5
 "A" Adjust the ADC gain when AVdd is changed
 "C" Calibrate ADC
 "F" Find KR, the ADC correction coefficient, when AVdd is changed
 "V" Find KS, the supply voltage conversion coefficient
 "H" Print help
####################################################################*/
#include <MSC1210.H>
#include <stdio.h>

#define FCLOCK   2            //External clock freq. is 1.842 MHz
#define DCMTN    781          //ADC decimation ratio, 781==> fdata = 20.0 Hz
#define LSB      1.25/ 16777216       //LSB for 1.25-V reference, ADC unipolar mode
#define KSz      7.0          //Supply voltage coefficient (6.621 for MSC1200), default value
#define KRz      1.076E-9     //ADC gain correction coefficient, default value for MSC1200/1201
//#define KRz    2.28E-10     //ADC gain correction coefficient, default value for MSC1210/1211

//---------- Constants for summation register ------------------
#define AVRG2          0xC0   //Summator average 2 ADC samples
#define AVRG4          0xC9   //Summator average 4 ADC samples
#define AVRG8          0xD2   //Summator average 8 ADC samples
#define AVRG16         0xDB   //Summator average 16 ADC samples
#define AVRG256        0xFF   //Summator average 256 ADC samples
#define AVRG           AVRG8  //Constant used for ADC averaging

//----------- Subroutines ---------------
float          ADC_Avrg(void);
void           Wait(char );
unsigned long  ADCcntrl(void);
unsigned long  ADCsupply(void);
extern  void   autobaud(void);       //Boot ROM subroutine

float          KR;     //Coefficient to adjust ADC gain when supply is changed
float          KS;     //Coefficient to calculate supply voltage when flying input is measured

//######################### Union ####################################
//u is unsigned long variable with separate byte addressing
//Long = u.lng;  Byte= u.bt.b4, ... , u.bt.b1
//###################################################################
union { signed long lng;
       struct  {
                      unsigned char b4 ;     //MSB
                      unsigned char b3 ;
                      unsigned char b2 ;
                      unsigned char b1 ;     //LSB
               } bt ;
      } u;
//------------------------------------------------------------------
```

**Continued on next page**

## USupply program (Continued)

```
//######################### Main program  #############################
//####################################################################
//####################################################################
void main(void)
{       char k;
        unsigned long gainC;
        signed long sply, splyC, cntr, cntrC;
        float x;

        KR= KRz;               //Preset gain correction coefficient
        KS= KSz;               //Preset supply coefficient
//--------------- Set MSC --------------------------
        PDCON = 0x37;          //Turn on ADC
        CKCON = 0x10;          //Need for ser. communication in MSC1200/1201 only //////
        TCON = 0x00;           //Need for ser. communication in MSC1200/1201 only //////
        ACLK = FCLOCK-1;       //Aclock
        USEC = FCLOCK-1;       //One µsec clock
        ONEMS = 18420;         //Setting for 1.842 MHz Clock
        ADMUX = 0x48;          //Set ADC multiplexer, AIN4-AINCOM
        ADCON0 = 0x20;         //Internal Vref = 1.25 V on, buffer off, BOD off, PGA = 1
        DECIM = DCMTN;         //Set ADC decimation ratio
        autobaud();            //Set and start serial communication with PC
        ADCON1 = 0x71;         //Unipolar, Sinc3, start ADC offset and gain calibration

   printf("\nStart USupply\nAINCOM->GND\n");
   printf("\AIN5->Cntrl signal\nAIN4->Flying\n");
   printf("ADC Decimation=%i\n\n", DECIM);
   k='C';                      //Start with calibration
//===================== MAIN LOOP ==================================
        while(1)
        {       switch (k)
//--------------- Measure supply voltage at AIN4 ---------
        case 's': case 'S':
                ADCsupply();           break;
//--------------- Measure control voltage at AIN5 --------
        case 'm': case 'M':
                ADCcntrl();                    break;
//---------------- Calibrate ADC ------------------------
//Control signal bigger then REF/2 is recommended
        case 'c': case 'C':
                ADCON1 = 0x71;         //Unipolar, Sinc3, start ADC self-offset and gain calibration
                Wait(1);               //Wait when ADC calibration is done
                u.bt.b4=0; u.bt.b3=GCH; u.bt.b2=GCM; u.bt.b1=GCL; //Copy the ADC gain
                gainC=u.lng;           //Store the ADC gain
                printf ("ADC Gain= %li\n", gainC);

                splyC = ADCsupply();   //Measure supply voltage, store ADC readings
                cntrC = ADCcntrl();    //Measure control signal, store ADC reading
                break;

//------------ Find the ADC gain correction coefficient --------------------
//Before this, ADC calibration with valid control signal (V > REF/2)
//and valid AVdd should be performed. Then change the supply voltage and type 'F'.
        case 'f': case 'F':
                sply = ADCsupply();            //Measure supply voltage
                cntr = ADCcntrl();             //Measure control voltage at AIN5
        //------------- Calculate coefficient -----------------
                x= (float) cntrC/cntr -1 ;
                KR= x/(float)(sply-splyC);     //ADC gain correction coefficient
                printf ("ADC Gain Cor.Coef.KR= %E\n", KR );
                break;
```

Continued on next page

## USupply program (Continued)

```
//----------------- Adjust the ADC gain ----------------------------
//Before this, the ADC gain correction coefficient should be found.
//Supply voltage should be changed from the calibration value to the new one.
        case 'a': case 'A':
                sply= ADCsupply();                    //Measure current supply voltage
//              printf ("ADC Correction Coef.= %.12lf\n", KR );//////////////////
                x= 1.+(float)(sply-splyC) * KR;       //Calculate ADC gain correction
                printf ("ADC Adjust coef.= %lf\n", x );
                printf ("Old ADC gain= %li\n", gainC );
                u.lng= gainC * x;                     //New ADC gain
                printf ("New ADC gain= %li\n", u.lng );
                GCH=u.bt.b3; GCM=u.bt.b2; GCL=u.bt.b1;       //Store ADC new gain
                break;

//-------------- Find the supply conversion coefficient -------------------------
        case 'v': case 'V':
                ADCsupply();                          //Measure supply
                printf ("Enter current supply voltage.(V)\nAVdd=");
                scanf("%f", &x);                      //Read voltage as float
                KS= x/(u.lng * LSB);                  //New supply voltage coefficient
                printf ("\nSupply voltage coef.KS= %.3f\n", KS );
                break;

//----------------- Print Help text -----------------------------------
        case 'h': case 'H':
                printf ("Enter\n'S' To check Supply voltage\n");
                printf ("'M' To Measure control signal\n");
                printf ("'A' To Adjust ADC gain\n");
                printf ("'C' ADC self offset and gain Calibration\n");
                printf ("'F' Find ADC gain cor.coef.\n");
                printf ("'V' Find supply Voltage coef.\n");
                break;
                }
                printf ("-----------------\nFor help type 'H'\n" );
                k = getchar();          printf("\n");
        }
} //main------------------------------------------------------------------


//################### ADC supply ###########################################
//Measure supply voltage, using flying ADC input. AINCOM is grounded. Buffer off.
//########################################################################
unsigned long ADCsupply(void)
{               float z;
                ADMUX= 0x48;            //Switch to flying input
                z = ADC_Avrg();         //Measure flying (supply) voltage with ADC
                printf ("Flying input= %.3fv\n", z);
                z = z * KS;             //Calculate AVdd
                printf ("Supply voltage= %.3fv\n", z );
                return u.lng;
}//---------------------------------------------------------------------
```

Continued on next page

## USupply program (Continued)

```
//################### ADCcntrl ###########################################
//Measure control signal at ADC inputs AINCOM and AIN5.
//ADC unipolar mode, AINCOM is grounded. Buffer off.
//######################################################################
unsigned long ADCcntrl(void)
{              float y;
               ADMUX= 0x58;                  //AINCOM-AIN5, control voltage
               Wait(1);                      //Wait for better precision
               y = ADC_Avrg();               //Measure voltage with ADC
               printf ("Control voltage= %lfv\n", y );
               return u.lng;
}//----------------------------------------------------------------------


//######################## Wait #####################################
//Make a pause for (t) ADC data cycles
//#################################################################
void Wait(char t)
{       char n;
        n=ADRESL;                            //Dummy read to clear ADC IRQ
        n=SUMR0;                             //Dummy read to clear SUM.IRQ

        for (n=0; n<t; n++)                  //Wait for t ADC conversions
        {
               while (!(AIPOL & 0x20));      //Wait for data ready MSC1200/1201
//          while (!(AIE & 0x20));           //Wait for data ready MSC1210/1211
               u.bt.b1 = ADRESL;             //Dummy read to clear ADCIRQ
        }
}//Wait -------------------------------------------------------------


/*##################### ADC_Avrg ##############################################
 Get ADC average from AVRG samples using summation register.
 ADC is in unipolar mode. Result is always positive voltage. ADC code is in u.lng
##############################################################################*/
float ADC_Avrg (void)
{       float y;
        Wait(4);                      //Wait for 4 ADC conversions, settle the filter
        SSCON = 0;                    //Clear summation registers
        SSCON = AVRG;                 //Set the averaging
        while (!(AIPOL & 0x40)) {}    //Wait for summator interrupt MSC1200/1201
//      while (!(AIE & 0x40)) {}      //Wait for summator interrupt MSC1210/1211
        u.bt.b4=SUMR3;  u.bt.b3=SUMR2; u.bt.b2=SUMR1; u.bt.b1=SUMR0;       //Copy ADC data
        y= LSB * u.lng;               //ADC result in volts
//      printf ("ADC code = %ld\n",u.lng);/////////////////////
        return y;
}//ADC_Avrg -----------------------------------------------------------------

//##############################################################################
//##############################################################################
```

# 14-bit, 125-MSPS ADS5500 evaluation

**By Hui-Qing Liu** (Email: liu_hui-qing@ti.com)
*Applications Engineer, High-Speed ADC*

## Introduction

The ADS5500 is a high-speed, pipeline, CMOS ADC with 14-bit resolution and a 125-MSPS sampling rate. In March 2004 Texas Instruments (TI) introduced the device, which is the first ADC in the world market with such high sampling speed and high resolution. The ADS5500 is suitable for applications such as wireless communication, test and measurement instrumentation, control systems, medical imaging, and high-speed digitization.

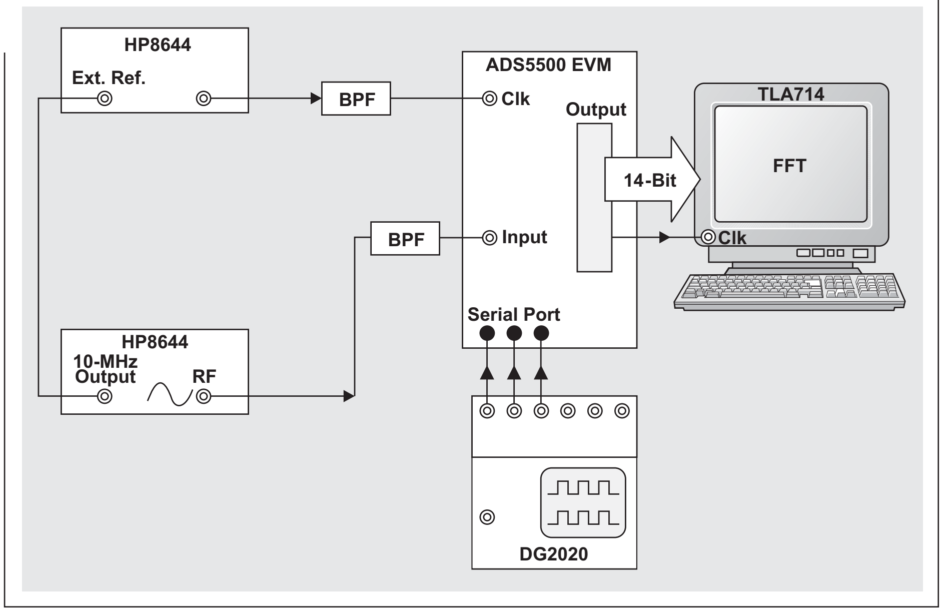
The ADS5500 consists of an input sample-and-hold stage; a 14-bit, pipeline ADC core; an internal voltage reference; a clocking circuit; digital error correction; a digital output driver; and a single 3.3-V voltage supply. The leading features and advantages of the ADS5500 are a wide signal input bandwidth of 750 MHz, a large dynamic differential input-signal range of 2.3 V peak-to-peak, a high signal-to-noise ratio (SNR) and spurious-free dynamic range (SFDR) at wide frequency range, a good SNR of up to 74 dBFS with low signal amplitude for receiver applications, and very low power dissipation (780 mW) at normal operation with a large load.[1] For example, with a 125-MSPS sampling rate, a –1-dBFS signal amplitude, and an input frequency of 190 MHz with proper input configuration, the ADS5500 has a typical SNR of about 70 dB and an SFDR above 82 dB. With the same sampling frequency, the same input frequency, and a –15-dBFS input signal amplitude, the ADS5500 has an SNR of 74 dBFS and an SFDR of 83 dBFS.

Since the ADS5500 has such high performance, it has a wide application; however, because of its high speed and high resolution, the device is sometimes challenging to evaluate. For this reason, this article introduces an ADS5500 evaluation system that includes the test equipment, system configuration, test circuit, basic high-speed ADC test concept, and test data.

## The ADS5500 test system

One of the ADS5500 bench test systems used for dynamic performance evaluation is shown in Figure 1. It basically consists of a signal source (HP8644), a clock source (HP8644), a digital logic analyzer (TLA714), a data generator (DG2020), bandpass filters (BPFs), a test board, and a fast Fourier transform (FFT) program. The signal source generates a pure tone signal with the amplitude and frequency necessary to test the ADS5500. The clock source generates a sine wave to trigger an external clock circuit of the ADS5500, which can be a transformer or PECL driver to produce an ideal sampling clock. The data generator is used to generate serial data for the control register. The Tektronix digital logic analyzer (TLA) is used to capture the data from the ADS5500 and analyze it with the FFT. If

**Figure 1. ADS5500 bench evaluation system**



no such instrumentation is available, a FIFO or an FPGA board with a PC can be used as a TLA or data generator.

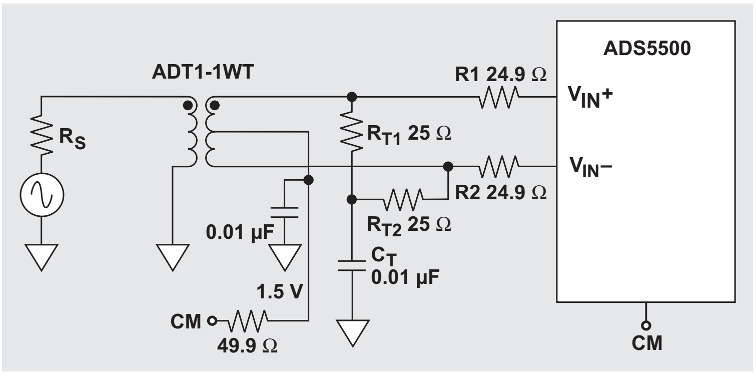
Some important specifications of the instrumentation in this system are signal frequency bandwidth, signal power, source impedance, noise, harmonics, jitter, phase lock function, and the ADS5500 digital load. The low input capacitance of devices such as the digital buffer, digital data capture board, or logic analyzer is important. The logic analyzer shown in Figure 1 has a 2-pF input capacitance. The maximum digital output load of the ADS5500 is 12 pF; a larger load will affect the ADS5500 evaluation. The timing between the data capture clock and the ADC output data can affect the test result if an external capture clock is used, so the timing must meet the data sheet specification. Data must be captured during the data-valid time. Using the ADS5500 output clock to capture ADS5500 data is strongly recommended, since the output clock is synchronized with the output data. There is a delay variation between the input clock and the output clock across devices and temperature as well as supply voltage; therefore, using the input clock for output data capture is less preferred and is not recommended, particularly at high speeds.

On the analog side of the ADS5500, all the signal and clock generators shown in Figure 1 have phase lock function, low-noise mode (Mode 3), 50-Ω source impedance, and 20-dBm source power. To avoid energy reflection on the signal transmission path, the equivalent input impedance of the ADC, including the external input circuit, should match the source impedance. This is shown in Figure 2, a configuration similar to one shown in Reference 2. At very high input frequency with certain test conditions, the source may not have enough power for the ADC due to the attenuation from the BPF and input circuit. In this case a wideband amplifier (for example, the THS900x or ZHL-6A) with a certain gain and a 50-Ω input/output impedance is needed to provide sufficient power to the ADS5500. Properly reducing R1 and R2 and increasing $R_{T1}$ and $R_{T2}$ will reduce the power requirement from the source at very high input frequency.

In a real test system, a BPF is used in the ADS5500 signal input path to minimize the harmonics and noise from the source. A narrow BPF is also used in the clock input path to minimize jitter from the clock source and to provide a good clock duty cycle when the sampling speed is very high. The bench test shows that with or without a BPF on the input signal path, the FFT result is dramatically different. The bench test also shows that, with the system in Figure 1 and a high input frequency, a narrow (3-MHz-bandwidth) BPF on the signal input path results in an SNR at least 0.3 dB better than a wide (10-MHz-bandwidth) BPF. The BPF on the clock input path can improve SNR more than 0.5 dB and SFDR more than 2 dB on the current EVM board with a 125-MHz clock and a very high input frequency. In any case the BPF itself should be tested to make sure it is functioning well. The input and output impedance of the BPF should be matched with the signal source impedance and the ADC input impedance. A good BPF choice would be the TTE KC series with a stopband attenuation of 50 dBC (minimum) to optimize the evaluation.

In the ADS5500 test it was observed that the ADS5500 performance is sensitive to the system jitter, analog input configuration, and test-board layout. These are discussed in the following sections.

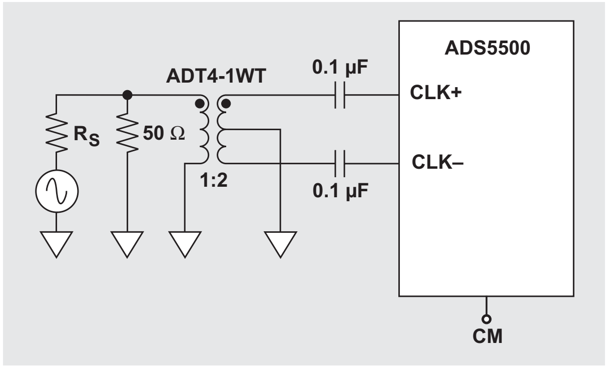
**Figure 2. Differential transformer input configuration for ADS5500 evaluation**



## The clock requirement

For the best evaluation, the ADS5500 requires its input clock to have low jitter; a 50% duty cycle; and a differential amplitude of 3 $V_{PP}$ if the input clock signal is a sine wave, or 1.5 $V_{PP}$ if it is a pulse. A sharper input clock signal edge provides a better SNR. The ADS5500 internally supports a dc offset voltage to the internal input clock circuit; therefore, a clock ac coupling path is recommended. To provide a sharp clock signal edge and the lowest external circuit noise and thus to get the best performance from the ADS5500 EVM, a 1:2 turns ratio transformer is used to couple a sine wave into the ADS5500 clock input as shown in Figure 3.[2] With a high clock frequency, using a BPF such as the TTE KC4T-125M-3M-50-69A BPF on the input

**Figure 3. Sine wave clock input circuit for the ADS5500 evaluation**



clock path is recommended if necessary to reduce the clock jitter noise from the source. To preserve the excellent ac performance of the ADS5500 and get the best evaluation, a low-jitter clock for this test is critical because the ADS5500 itself has a very low jitter of about 300 fs.[1] Any input clock circuit or driver must not carry extra jitter. Generally, as the input frequency increases, the clock jitter becomes more dominant in the system for maintaining a good SNR. The following equation can be used to calculate the achievable SNR for a given input frequency and system jitter in psrms.

$$\mathrm{SNR} = 20\log\frac{1}{2\pi f_{IN} t_{ja}},$$

where $t_{ja}$ is the rms aperture jitter from all jitter sources such as the clock edge, input signal, test board, and device; and $f_{IN}$ is the input frequency.

In addition, the input clock is treated as an analog signal, and its power supply should be separated from the digital driver power supply to limit the digital noise.

## Analog input configuration

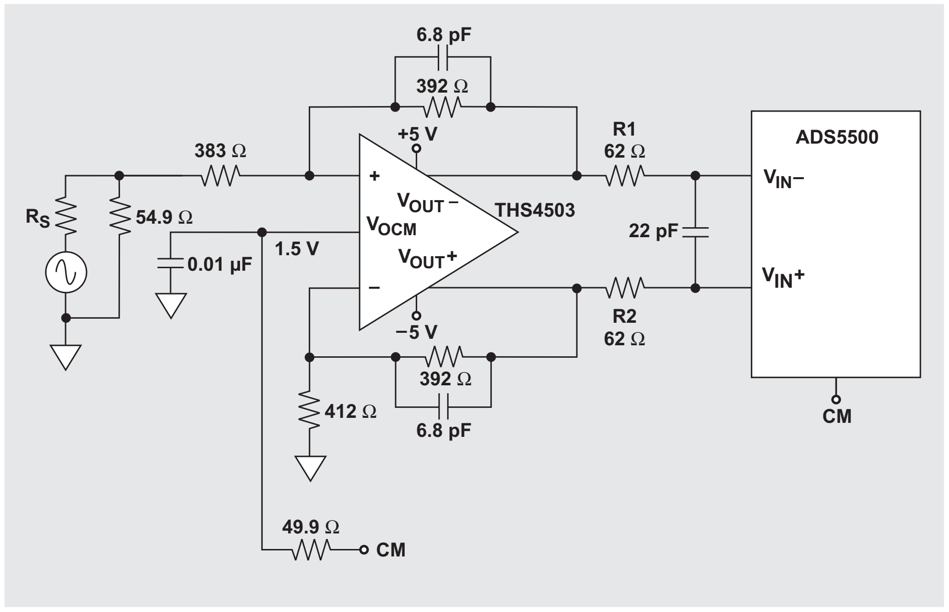
The analog input configuration of the ADS5500 is important for the evaluation and typically includes a transformer-coupled differential input (see Figure 2) or an operational amplifier-driven differential input (see Figure 4). This configuration can be found in Reference 2. The transformer configuration provides low noise and harmonics over a wide frequency range. It also provides ac coupling, differential input, and a wide signal bandpass, making it a good way to evaluate the ADS5500 even though the transformer has some insertion loss. The op amp, widely used for signal conditioning and power boosting, is also used for dc coupling.

In the transformer configuration, the input impedance of the ADS5500 is an important consideration. The ADS5500 input impedance is capacitive and is the function of both the sampling clock frequency and the input signal frequency. When the sampling and input frequency speeds are relatively low, the ADS5500 input impedance is relatively high, and matching the source impedance is not difficult. When the sampling speed is very high (at 125 MHz) and the input frequency is very high (above 150 MHz), the input impedance of the ADC is low. In this case the equivalent input impedance could be smaller than 50 Ω, which may cause a mismatch with the source impedance and require more source current. This needs to be accounted for in the evaluation.

Figure 2 shows a transformer-coupled analog input circuit for the evaluation of the ADS5500. $R_{T1}$ and $R_{T2}$ are the termination resistors for the source impedance match; they also form the low-pass filter with $C_T$. We have seen that at input frequencies below 150 MHz, $R_{T1} + R_{T2} = 50\ \Omega$ provides the best SNR and SFDR; while at input frequencies above 150 MHz, higher values of $R_{T1}$ and $R_{T2}$ can be used in the evaluation due to low ADC input impedance and high transformer roll-off. R1 and R2 are the analog input serial resistors for isolation between the ADC switch capacitor input and the signal source. They also form a low-pass filter with the ADS5500 input capacitance. Proper R1 and R2 values are needed for the best performance. If R1 and R2 are too small, the SFDR could decrease; and if they are too big, the signal source power will increase. We have seen that a value of 25 Ω for R1 and R2 provides the best result when a transformer coupling is used in the input circuit. In Reference 2, two transformers are used in the input circuit to reach the best differential signal balance; but this causes a 9-dB input signal roll-off from 70 MHz to 350 MHz at the front end of the ADC and requires a large signal source. For this reason only one transformer was used in some tests. We have seen that there is no significant difference in the performance whether one or two transformers is used. The transformer is used in our test for signal ac coupling and single-ended to differential signal conversion.

Figure 4 shows the ADS5500 EVM configuration used to evaluate the ADS5500 when the input signal is driven by the THS4503 fully differential op amp. The dc-coupled THS4503 is set for unity gain by the input and feedback resistors. A low-pass filter is set by the feedback resistor and capacitors on the feedback paths to limit signal

**Figure 4. The op amp input configuration for ADS5500 evaluation**



bandwidth. The THS4503 receives a single-ended analog signal and dc offset voltage, then outputs a differential signal with the dc offset voltage for the ADS5500. A small (22-pF) input capacitor is needed to form a low-pass filter with serial resistors R1 and R2. The serial resistors are important for isolation between the op amp and the ADC input stage. A value around 60 Ω is chosen for these resistors in the circuit. With this configuration the ADS5500 can provide good performance with input frequencies of up to 20 MHz. This is described later under "ADS5500 test data."

## Board layout and decoupling

The evaluation board layout and signal decoupling are equally important factors in the ADS5500's performance. The ADS5500 package is designed with the analog inputs on one side and the digital outputs on the other, providing good physical isolation between them. To achieve optimum ADS5500 performance, it is important to use a short signal trace, separated digital and analog signal locations, and ground planes with a multilayer board. When the analog inputs to the ADS5500 are driven differentially, it is especially important to make the layout highly symmetrical to avoid phase differences between the two signal paths, as an asymmetrical parasitic on the PCB creates input signal distortion. The differential input clock signal traces should be symmetrical and short to avoid mismatches in propagation delays. The clock lines should not cross any other signal traces. Short circuit traces on the digital outputs will minimize capacitive loading.

The ADS5500 must be treated as an analog component, and its power supply pins should be connected to the analog supply. For the best performance, the analog supply ($AV_{DD}$) and the digital driver supply (VDRV) should be separated to limit substantial current transient noise from the digital driver. The VDRV pins must also be connected to a low-noise supply. The supply voltage must be thoroughly filtered before connecting to the supply of the converter. The recommended supply decoupling scheme for the ADS5500 is shown in Reference 2. All supply pins can be bypassed with a combination of 0.1-µF ceramic capacitors and a 10-µF tantalum tank capacitor or ceramic capacitor. To minimize the lead and trace inductance, the capacitors must be located as close to the supply pins as possible. In addition, larger (10-µF to 47-µF) bipolar decoupling capacitors effective at lower frequencies must be used on the main supply pins.[2] All ground connections on the ADS5500 are internally bonded to the metal flag (bottom of the package) that forms a large ground plane. All ground pins must directly connect to the analog ground plane that is under the converter. All the supply pins and reference pins must be sufficiently bypassed due to the clock feedthrough (switch noise) caused by the high-frequency clock of the ADS5500. Insufficient bypassing will add noise to the conversion process. Besides the factors already discussed, others affecting SNR and SFDR, such as external bias resistance, the serial port, the digital output load, the external buffer, etc., can contribute a small performance variation to the evaluation.

## FFT analysis

In the ADS5500 dynamic performance evaluation, the FFT is used to determine the SNR, total harmonic distortion (THD), and SFDR of the device in the test system shown in Figure 1. The FFT signal frequency domain analysis provides a signal spectrum. When a pure tone signal passes through a nonideal or real system, the signal spectrum always changes. The changed signal spectrum presents the system dynamic characteristics.

When a high-performance ADC is tested, a perfectly synchronized system is critical. Therefore the user must make sure that the input signal and sampling clock sources are synchronized and that they meet the following coherent sampling requirement to avoid spectrum leakage on the FFT.

$$f_{IN} = m\frac{f_S}{N},$$

where $f_{IN}$ is the input signal frequency, $f_S$ is the sampling clock frequency, N is the sampling size, and m is an odd number. If coherent sampling can't be ensured, then windowing techniques can be used in the FFT.[3]

## ADS5500 test data

As good application examples of this test system, some test data is presented here for users whose designs are not covered by the current ADS5500 data sheet.[2] This data includes an FFT at very high input frequency; two-tone intermodulation distortion (IMD); an FFT of the ADS5500 combined with an op amp; as well as differential nonlinearity (DNL) and integral nonlinearity (INL). The data is measured using an EVM similar to the ADS5500 EVM shown in Reference 2. Some of this data is better than the data sheet specification because the test system used here was optimally set for the frequency conditions.

Figure 5 shows an FFT plot of the ADS5500 with a very high input frequency and a 125-MHz sampling clock through a transformer-coupled differential input configuration. The input frequency is 190 MHz and the amplitude is –1 dBFS. The FFT analysis shows that the SNR is 69.6 dBFS and

**Figure 5. FFT plot of ADS5500 with 190-MHz input frequency and 125-MHz sampling clock**

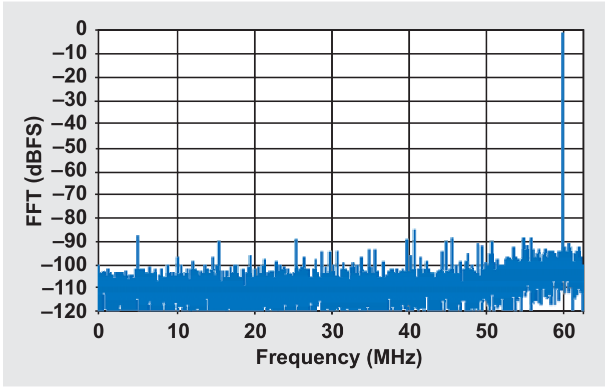
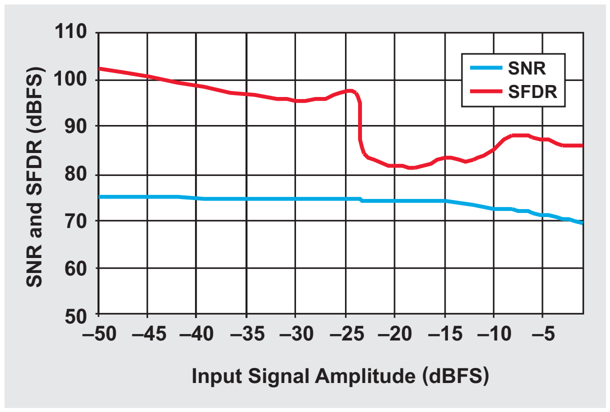


**Figure 6. Dynamic performance of ADS5500 with different input amplitudes, 190-MHz input frequency, and 125-MHz sampling clock**



the SFDR is 85 dBFS. In this case the input impedance due to $R_{T1}$ and $R_{T2}$ is about 200 Ω, and R1 and R2 are each less than 25 Ω in the input circuit.

Figure 6 shows the dynamic performance of the ADS5500 with different input amplitudes, a 190-MHz input frequency, and a 125-MHz sampling clock. This data shows that at such a high input frequency and with low input amplitudes, the ADS5500 has a 74-dBFS SNR and a high SFDR.

Figure 7, Figure 8, and Table 1 show the two-tone IMD test data of the ADS5500 in an undersampling condition. This test uses the ADS5500 EVM shown in Reference 2. The test signal is a two-tone signal combined from two sine waves with different frequencies, such as any two frequencies above the ADC's Nyquist frequency with a 1-MHz separation. The two-tone signal amplitude should not exceed the ADS5500's full scale. Just as with a single-tone FFT, when the signal amplitude is higher, the spur and IMD from the ADC are higher. In this test the amplitude of –7 dBFS or –10 dBFS for each tone is used, providing the ADS5500 with a combined input signal at full scale or 3 dB below full scale. The input signal frequency ranges from 76 to 125 MHz, and the clock frequency ranges from 96 to 125 MHz. The IMD rejection is the ratio of the rms value of one input tone to the value of the worst third-order intermodulation product.

Figures 7 and 8 show FFT plots for two different sampling speeds at an input amplitude of –10 dBFS for each tone. In Figure 7 the two-tone SFDR is 96 dBFS with about a 100-MHz input frequency, a 1-MHz separation, and a 125-MHz sampling clock. In Figure 8 the two-tone SFDR is 97 dBFS with about a 125-MHz input frequency, a 1-MHz separation, and a 96.5-MHz sampling clock.

**Figure 7. Two-tone IMD of ADS5500 with 100-MHz input frequency, 1-MHz separation, and 125-MHz sampling clock**

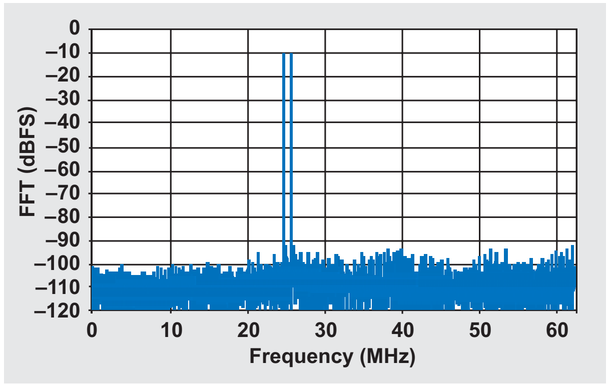


**Figure 8. Two-tone IMD of ADS5500 with 125-MHz input frequency, 1-MHz separation, and 96.5-MHz sampling clock**

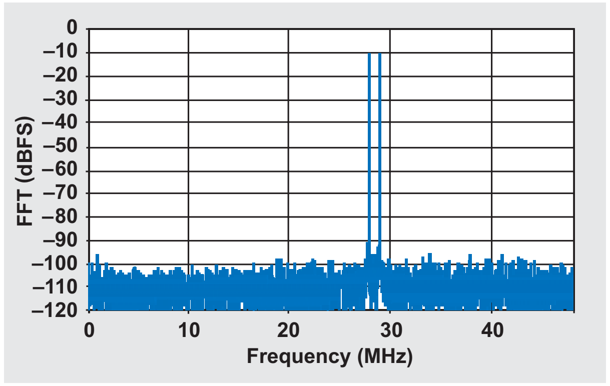


**Table 1. Two-tone IMD of ADS5500**

| CLK (MHz) | INPUT FREQUENCY 1 (MHz) | INPUT FREQUENCY 2 (MHz) | SFDR (dBFS) | IMD REJECTION (dBFS) | IMD2 (dBFS) | IMD3 (dBFS) | AMPLITUDE (dBFS) |
|---|---|---|---|---|---|---|---|
| 125 | 99.0 | 100.0 | 96 | — | 98 | 94 | –10 |
| 125 | 81.0 | 82.0 | 95 | — | 104 | 92 | –10 |
| 125 | 99.0 | 100.0 | — | 87 | 95 | 86 | –7 |
| 125 | 81.0 | 82.0 | — | 88 | 102 | 88 | –7 |
| 102.4 | 76.1 | 77.1 | 91 | — | 101 | 93 | –10 |
| 102.4 | 76.1 | 77.1 | — | 95 | 101 | 91 | –7 |
| 96.5 | 124.5 | 125.5 | 97 | — | 102 | 97 | –10 |
| 96.5 | 124.5 | 125.5 | — | 83 | 97 | 81 | –7 |

Table 1 shows that with a 125-MHz sampling clock and an 80- to 100-MHz input frequency, the two-tone IMD rejection is above 87 dBFS at full-scale input (–7-dBFS input amplitude for each tone). Table 1 also shows the ADS5500's good undersampling IMD performance with a 96.5- to 125-MHz sampling clock and a 76- to 125-MHz input frequency. This data shows that the ADS5500 has leading IMD performance at such high input frequencies and sampling speeds.

Figure 9 shows the dynamic performance of the ADS5500 and THS4503 combined. This data is measured on the circuit shown in Figure 4 with a 125-MHz sampling clock and different input frequencies. Note that the serial resistors R1 and R2 in Figure 4 each have a value of 62 Ω instead of 25 Ω as mentioned in Reference 2. The data shows that with the THS4503, at input frequencies of up to 18 MHz, the ADS5500 can provide an SNR of 71 dBFS and an SFDR above 86 dBFS. At input frequencies above 20 MHz, the performance starts degrading due to the bandwidth limitation of the THS4503. The THS4503 is a fully differential op amp; detailed information can be found in Reference 4. For wideband applications with an op amp input configuration, the OPA695 is recommended.[5]

The linearity of the ADS5500 is important in its applications. Two specifications, DNL and INL, are used to describe ADC nonlinearity. DNL is measured from sampling a pure sine wave with a large sample size, then calculating the deviation of the transition from each code. INL is integrated from DNL. Figure 10, Figure 11, and Table 2 show a typical linearity measurement of the ADS5500 from the test system just discussed and a typical EVM board. With a 125-MHz sampling clock, an input frequency of 10 MHz, and at room temperature, the maximum/minimum DNL is ±0.5 LSB, and the maximum/minimum INL is ±2.5 LSB. Table 2 shows that the ADS5500 has good linearity, with similar DNL/INL at input frequencies of 10, 45, and 100 MHz.

Table 2. DNL/INL of ADS5500 vs. input frequency with 125-MHz sampling clock

| INPUT FREQUENCY (MHz) | MAX/MIN DNL (LSB) | MAX/MIN INL (LSB) | MISS CODE |
|---|---|---|---|
| 10 | 0.46/–0.52 | 2.44/–2.40 | 0 |
| 45 | 0.47/–0.48 | 2.59/–2.57 | 0 |
| 100 | 0.43/–0.44 | 2.05/–3.05 | 0 |

Figure 9. Dynamic performance of ADS5500 with THS4503 vs. input frequency with 125-MHz sampling clock

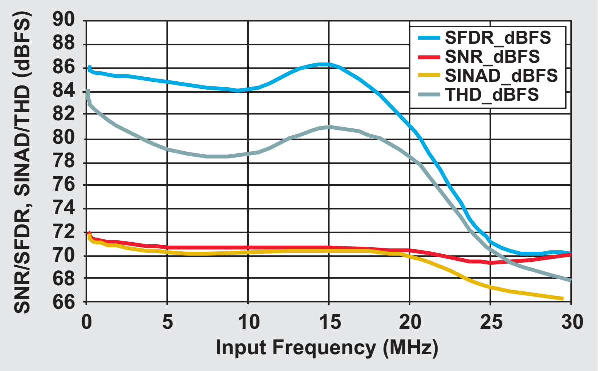


Figure 10. ADS5500 DNL with 10-MHz input frequency and 125-MHz sampling clock

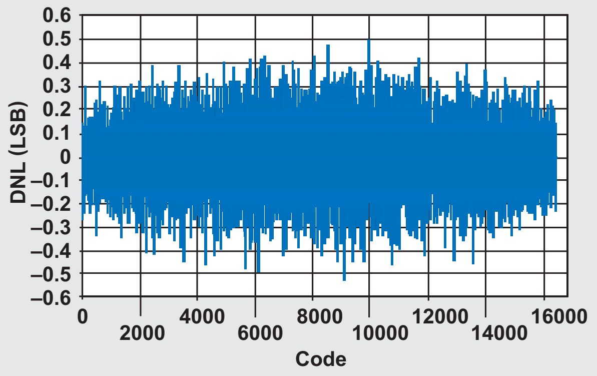
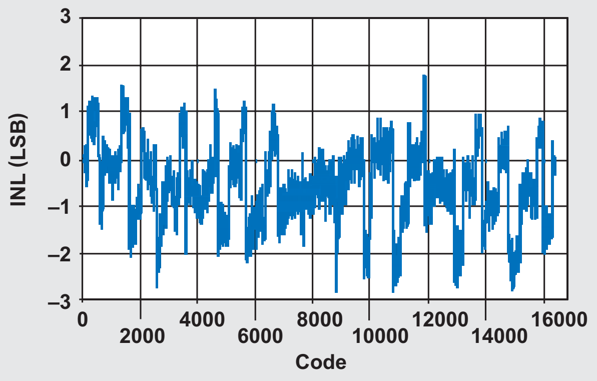


Figure 11. ADS5500 INL with 10-MHz input frequency and 125-MHz sampling clock



## Conclusion

The ADS5500 is a very high-performance device with wide application, but evaluating it can present some test setup challenges due to its high frequency and high resolution. This article has introduced an ADS5500 evaluation system including the test equipment, system configuration, test circuit, clock requirements, basic high-speed ADC test concept, and test data. This evaluation has shown the leading performance advantages of the ADS5500: wide signal input bandwidth, large dynamic differential input signal range, high SNR and SFDR over a wide frequency range, good SNR of up to 74 dBFS with a low signal amplitude for receiver applications, and very low power dissipation with a high (125-MSPS) sampling rate.

## References

For more information related to this article, you can download an Acrobat Reader file at www-s.ti.com/sc/techlit/*litnumber* and replace *"litnumber"* with the **TI Lit. #** for the materials listed below.

| **Document Title** | **TI Lit. #** |
|---|---|
| 1. "14-Bit, 125MSPS Analog-to-Digital Converter," ADS5500 Data Sheet | sbas303 |
| 2. "ADS5500/5541/5542/5520/5521/5522 14- and 12-Bit Single Channel ADC EVM," User's Guide | slwu010 |
| 3. Alan V. Oppenheim, *Discrete-Time Signal Processing* (Prentice-Hall, Inc., 1989). | — |
| 4. "Wideband, Low-Distortion Fully Differential Amplifiers," THS4502/4503 Data Sheet | slos352 |
| 5. "Ultra-Wideband, Current-Feedback Operational Amplifier With Disable," OPA695 Data Sheet | sbos293 |

## Related Web sites

**analog.ti.com**
**www.ti.com/sc/device/ADS5500**
**www.ti.com/sc/device/THS4503**

# Clocking high-speed data converters

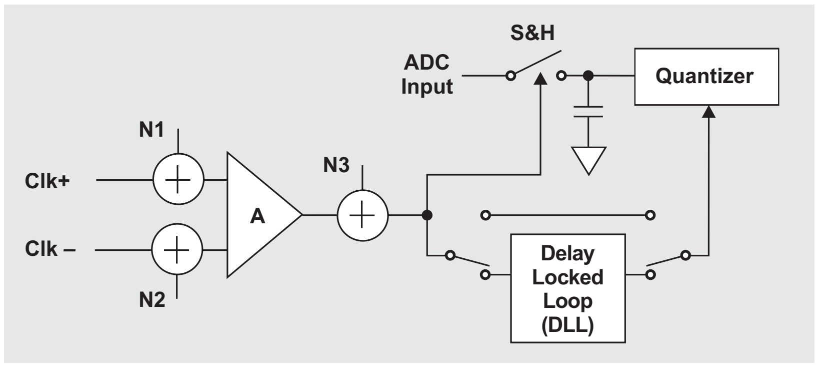
**By Eduardo Bartolome,** *High-Speed ADC Systems and Applications Manager* (Email: e-bartolome1@ti.com),
**Vineet Mishra,** *High-Speed ADC Design Engineer* (Email: vineetm@ti.com),
**Goutam Dutta,** *High-Speed ADC Test Engineer* (Email: g-dutta2@ti.com),
**and David Smith,** *High-Speed ADC Test Engineer* (Email: w-smith13@ti.com)

## Introduction

In circuit design that involves the use of a high-performance, high-speed analog-to-digital converter (ADC) such as the ADS5500, one of the main careabouts is the clocking scheme. Questions about the type of clock to be used (sinusoidal or square), the voltage levels, or the jitter are common. The purpose of this article is to explain the general theory to support the circuit designer in making the right choices.

Figure 1 shows a simplified model of the clock circuit inside a high-speed ADC like the ADS5500. Although not all ADCs have exactly the same internal blocks in their clock distribution, this diagram can be modified to fit your particular ADC. Since nowadays most of the circuits sold as ADCs include a front sample-and-hold (S&H), for the purpose of this article we will differentiate between them. The circuit that takes an instantaneous analog snapshot of the input signal will be called the S&H; and the ADC itself, which converts the analog value being held by the S&H into quantized digital output, will be called the quantizer. Analyzing what parameters of the internal clock are important for these two circuits will help us understand the main careabouts in our external clock design.

**Figure 1. Simplified model of clock circuit in high-speed ADC**

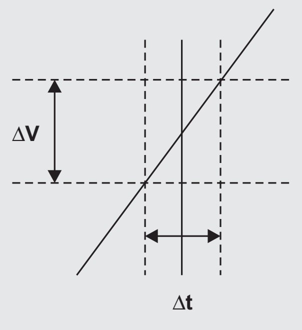


## Errors in the sampling instant

The conversion process starts when a clock signal tells the S&H to take the sample. Up to that instant, the internal switch on the S&H circuit has been closed, allowing the voltage across the capacitor to track the input signal (which is why other literature more properly calls this circuit “track and hold”). One of the edges of the input clock then indicates when to open this switch, and the capacitor holds the voltage at that instant in time. This instant is represented in Figure 2 by a vertical solid line. Any error in that instant (Δt) will translate as an error in voltage (ΔV) dependent on the input signal slope. The error in that instant is what we will call jitter.

A mathematical estimation of the best-case signal-to-noise ratio (SNR) (without other noise sources), given a certain amount of jitter, can be extracted from Figure 2. Given a sinusoidal input of amplitude A and frequency $f_{IN}$ (1/T), the uncertainty of the sampled voltage at a given point will be proportional to the slope of the input signal at that instant and to the uncertainty of the sampling instant (jitter, which is the rms value of that variation,

**Figure 2. Voltage error relation to sampling jitter**



uncorrelated with the input level). The total uncertainty is the addition of all the uncertainties at each point of the sinusoid weighted by the probability of sampling each of the points:

$$\sigma^2_{\text{Jitter}} = \frac{1}{T}\int_0^T (\text{Slope}(\tau)\times\text{Jitter})^2 d\tau = \frac{1}{T}\int_0^T \left[\frac{d\left(A\sin\frac{2\pi\tau}{T}\right)}{d\tau}\text{Jitter}\right]^2 d\tau$$

$$= \frac{1}{T}\text{Jitter}^2\int_0^T \left(\frac{2\pi A\cos\frac{2\pi\tau}{T}}{T}\right)^2 d\tau = \left\langle a = \frac{2\pi\tau}{T}, \frac{da}{d\tau} = \frac{2\pi}{T}\right\rangle$$

$$= \frac{T}{2\pi}\left(\frac{2\pi A}{T}\right)^2 \text{Jitter}^2 \frac{1}{T}\int_0^{2\pi}(\cos^2 a)da = \frac{T}{2\pi}\left(\frac{2\pi A}{T}\right)^2 \text{Jitter}^2 \frac{1}{T}\frac{1}{2}(a+\sin a\times\cos a)\Big|_0^{2\pi}$$

$$= \frac{T}{2\pi}\left(\frac{2\pi A}{T}\right)^2 \text{Jitter}^2 \frac{1}{T}\pi = \frac{1}{2}\left(\frac{2\pi A}{T}\right)^2 \text{Jitter}^2$$

The theoretical limitation of the SNR due to jitter is given by

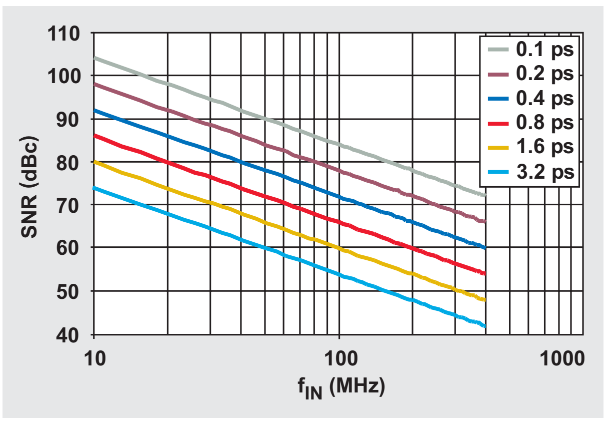
$$\text{SNR (dBc)} = \frac{S}{N} = 10\log_{10}\left[\frac{\frac{A^2}{2}}{\frac{1}{2}\left(\frac{2\pi A}{T}\right)^2\text{Jitter}^2}\right] = -20\log_{10}(2\pi f_{IN}\,\text{Jitter}). \quad \textbf{(1)}$$

Figure 3 shows this limitation as a function of the input frequency.

Observe that increasing or decreasing the input amplitude ($A_{IN}$) has no effect on the SNR component coming from jitter. In other words, as we decrease the input amplitude, the amount of error due to the jitter also becomes smaller. Nevertheless, there are other sources of error, like thermal noise, that do not get smaller. Assuming all these sources of noise are uncorrelated, the total noise is the addition of a noise term independent of input frequency and a noise term dependent on input frequency (jitter):

$$\text{SNR (dBc)} = 10\log_{10}\left[\frac{\left(\frac{A}{\sqrt{2}}\right)^2}{\text{Thermal}+\text{Quantization}+\frac{1}{2}\left(\frac{2\pi A}{T}\right)^2\text{Jitter}^2}\right] \quad \textbf{(2)}$$

**Figure 3. Limitation of the SNR due to jitter as a function of input frequency**



For a given ADC, below a certain input amplitude, the SNR is dominated by those other sources and jitter becomes irrelevant. One way of showing this is plotting the noise floor in dBFS with respect to the $A_{IN}$. Figure 4 shows a real noise-floor example of the ADS5542 operating at 78 MSPS and 230-MHz input.

Note that the theoretical trace, computed by adding the effect of 250 fs of jitter and about 1 LSB of thermal noise (idle channel noise; i.e., noise floor with no ADC input), very closely follows the measured performance. This effect is especially important when we compute the theoretical effects of jitter in our system and try to choose the right ADC and clocking. Specifically, it shows that for certain systems, ADC data taken at –1 dBFS may give us too pessimistic a result, as the signal may seldom reach those levels.

Figure 5 (extracted from Reference 1) shows the SNR versus input and sampling frequencies. This measured plot correlates with Equation 2. How can we explain the degradation observed in the SNR as input frequency ($f_{IN}$) increases for a fixed sampling frequency ($f_S$)? Assume that a full-scale sinusoid at low input frequencies is unaffected by jitter and that, as we increase the frequency of that sinusoid, the SNR will be degraded exclusively by clock jitter. In that case, for a given $f_S$, we can estimate a value for clock jitter. Table 1 shows the measured SNR for $f_S$ = 60 MSPS (see the red horizontal dashed line in Figure 5) versus the estimated SNR using Equation 2 and assuming a jitter of 200 fs.

So Equation 2 seems to model variation in the SNR versus $f_{IN}$ very well. What about SNR variation versus $f_S$? Does the jitter effect on the noise floor depend on the $f_S/f_{IN}$ ratio? In other words, given a certain amount of phase noise in our clock, will its effect be much worse in the case of 65-MSPS/150-MHz input frequency than in the case of 125-MSPS/150-MHz input frequency? From Equation 2 it is obvious that $f_S$ has nothing to do with SNR jitter. Nevertheless, we observe that as we decrease $f_S$ (following the blue vertical dashed line in Figure 5), the SNR degrades, which seems counterintuitive.

We know that by increasing the input waveform's number of hits per cycle, even with the same likelihood of error on each sample, we will "average out" the bigger portion of the noise. In other words, as we get more samples, each

**Figure 4. ADS5542 typical noise-floor measurement**

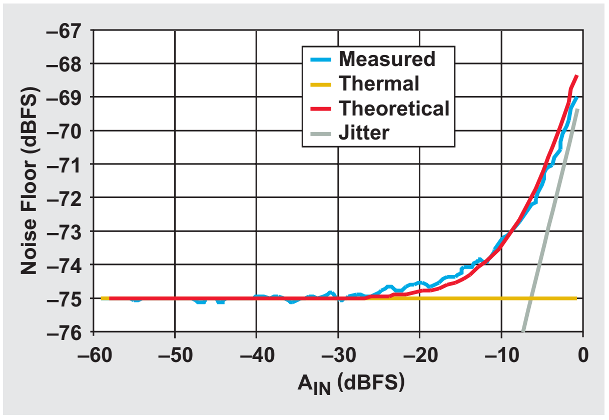
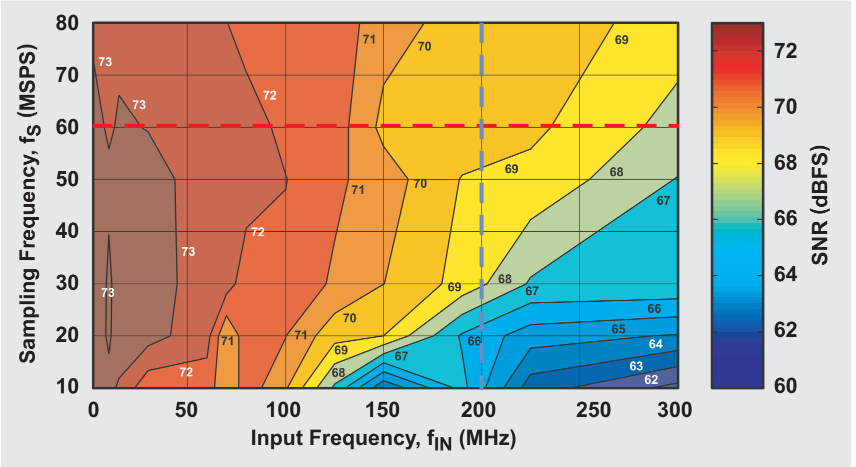


**Table 1. Measured vs. estimated SNR**

| $f_{IN}$ (MHz) | 2.00 | 10.01 | 15.51 | 30.00 | 60.04 | 70.04 | 80.01 | 100.00 | 125.01 | 150.00 | 190.04 | 225.03 | 300.02 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Measured SNR (dBFS) | 73.35 | 72.84 | 73.13 | 72.96 | 72.75 | 72.54 | 72.55 | 71.68 | 71.50 | 69.72 | 69.61 | 69.22 | 67.63 |
| Estimated SNR (dBFS) | 73.35 | 73.34 | 73.32 | 73.22 | 72.85 | 72.68 | 72.49 | 72.08 | 71.49 | 70.88 | 69.86 | 68.99 | 67.25 |

**Figure 5. SNR with digital phase lock loop off**



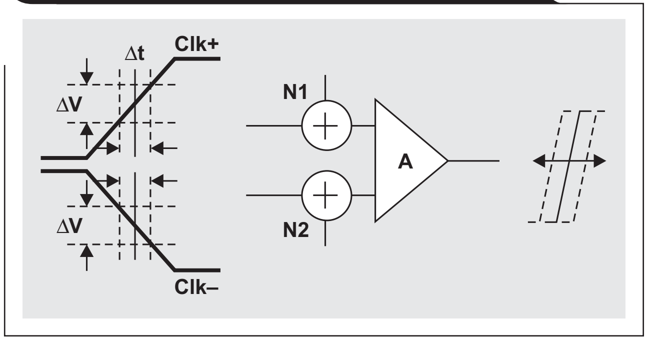
one with the same amount of error, the "average" of those will be more precise. This is no different than the standard concept of process gain. Considering the frequency domain, an increase in the sampling frequency will spread the same amount of noise over a bigger bandwidth, effectively reducing the noise floor. Usually we care about the in-band noise in our signal. Doubling the frequency of our clock will reduce the noise floor by half (i.e., 3 dB). Nevertheless, when we compute the SNR, we integrate all the noise; so this does not explain the SNR degradation that occurs as we decrease $f_S$. For an explanation we have to introduce a new factor, a new jitter that will not be the same for different sampling frequencies. A further look at Figure 1 will help us understand this SNR degradation.

Note that so far we have been talking about jitter generically as a total jitter budget allocated to our system. The sources for jitter can be external (such as a clock provided by the user) or internal in the ADC. Jitter is a form of simplification ("integration") of a more specific parameter called phase noise. The relationship between jitter and phase noise is explained in References 2 and 3. Disregarding those details, ADC users should at least be aware that not all phase noise in the clock will affect the system equally. Phase noise very close to the carrier reflects slow variation in the sampling instant, which may not be relevant in systems with a "short" observation time. Phase noise for larger offsets away from the carrier may be more important but at the same time easier to filter out. It is important to note that currently ADCs do not provide any mechanism to reject any incoming jitter. On the contrary, the clock chain inside the ADC adds to the jitter degradation. There will always be a delay between initiation of the clock signal and the time when the S&H goes into the hold mode. While the mean of that delay, commonly referred to as "aperture delay," does not produce a nondeterministic error, the variation in that delay will create an error. Some of that variation will come from noise sources inside the ADC (represented as N3 in Figure 1), which the user cannot influence except by modifying some of the external conditions such as temperature or supply. Nevertheless, there are some techniques the user can apply to minimize the influence of other internal sources of noise (N1 and N2). We will center the discussion around this topic.

As shown in Figure 1, the first stage in the internal clock chain is an amplifier. Texas Instruments usually includes such a circuit to present a more clock-friendly interface to the user, with the following features:

- High input impedance to reduce clock driver load.
- Differential amplifier that supports either differential or single-ended inputs.
- Amplifier to support smaller clock swings.

**Figure 6. Evaluating the clock rising edge**



- "Squares" the input clock signal (if sinusoidal) to generate the internal digital clocks. The amplifier is usually followed by logic circuits to distribute the clock to the internal blocks of the ADC.

Although ADC designers optimize this circuit to minimize jitter, there are always limitations set by the process, power, and other trade-offs. ADC users can model this circuit knowing that it adds a certain noise voltage to the input clock before amplification takes place.

In Figure 6 we represent only the rising edge of the clock, assuming that this is the edge used to open the switch on the S&H circuit (i.e., the edge indicating when to hold the sample). Note that the jitter on the other edge theoretically has no effect on the SNR. For simplicity we are also assuming that the slopes of the edge on the positive (Clk+) and negative (Clk–) lines of the clock are the same (which may not be true, but this has no effect on the following reasoning).

Although the sources of noise are internal (N1, N2), their final effect on performance can be minimized by the user. Specifically, if the input clock has infinite slope, the addition of any (voltage) noise to that edge will not affect the time position of the edge. As the edge becomes slower, the effect of adding any voltage noise will produce a bigger variation in time. In the case where the clock is a sinusoidal signal, increasing the amplitude or the sampling frequency will increase the slope of the edge. We should have the same jitter when we double the sampling frequency as when we double the clock amplitude. All this can be expressed in the following equation:

$$\begin{aligned}(\text{Jitter}_{\text{Total}})^2 &= (\text{Jitter}_{\text{External}})^2 + (\text{Jitter}_{\text{N3}})^2 + (\text{Jitter}_{\text{N1,N2}})^2 \\ &= (\text{Jitter}_{\text{External}})^2 + (\text{Jitter}_{\text{N3}})^2 + \left[\frac{\text{K(N1, N2)}}{\text{Clk\_slope}}\right]^2, \end{aligned} \qquad (3)$$

where K(N1, N2) represents the input amplifier jitter contribution and is constant for each ADC.

Let’s take a look at Figure 7 (adapted from Figure 17 in Reference 4). This figure really shows how to squeeze the last dB from the SNR at high input frequency. In line with the previous discussion, one of the first things that catches our attention is that as we increase the clock amplitude, the SNR improves. We are just minimizing the effect of the third term of Equation 3, improving the SNR to a point where the jitter coming from external sources and from N3 will be dominant and any further improvement in the third term will be irrelevant. Fitting the results from this model to the measured data (in this case, for differential clocking with 3.3-V $OV_{DD}$, using the curve labeled “SNR Diff 3.3”), we can obtain the constants for Equation 3:

- Adding terms 1 and 2 (we cannot distinguish between them unless other measuring techniques are applied) will give us a total jitter of 300 fs. Given the purity of our sources in the lab, we can assume that all this jitter is coming from the ADS5413 and thus from N3.
- K(N1,N2) will be 160 µV.

The final equation will be

$$(\text{Jitter}_{\text{Total}})^2 = (300\text{ fs})^2 + \left(\frac{160\ \mu\text{V}}{\text{Clk\_slope}}\right)^2. \qquad \textbf{(4)}$$

Figure 8 and Table 2 compare the result of Equation 4 with the real data.

Note that a clock slope of about 1 V/ns—i.e., a CMOS edge of 3.3 ns—will be sufficient to obtain the maximum performance from our ADC. If the clock is sinusoidal, we will require a peak-to-peak differential clock of about 4 V. With a single-ended clock, 3.3 $V_{PP}$ will be about the maximum we can provide without exceeding the supply rails and turning on the internal protections. Using a differential clock will let us increase the clock amplitude to double that amount. This is one of the advantages of using differential clocking. The other is the rejection of common-mode noise signals. Nevertheless, observe that in Figure 7 the performance of the single-ended clock for small amplitudes is actually slightly better than that of the differential clock. Besides a possible repeatability error between measurements, this could also be due to an imbalance between the two clock lines (a different N1 versus N2), so that the differential clock performance is actually an average of the independent single-ended performances. Another trend to observe is that when the digital output voltage is increased, in this case from 1.8 V to 3.3 V, more switching noise is produced that seems to couple to the clock circuit, as the degradation seems to be smaller at lower input frequencies (see Figure 9, which was adapted from Figure 19 in Reference 4). Finally, note that the clock amplitude has little or no effect on the distortion spurious-free dynamic range.

**Figure 7. AC performance vs. clock level**

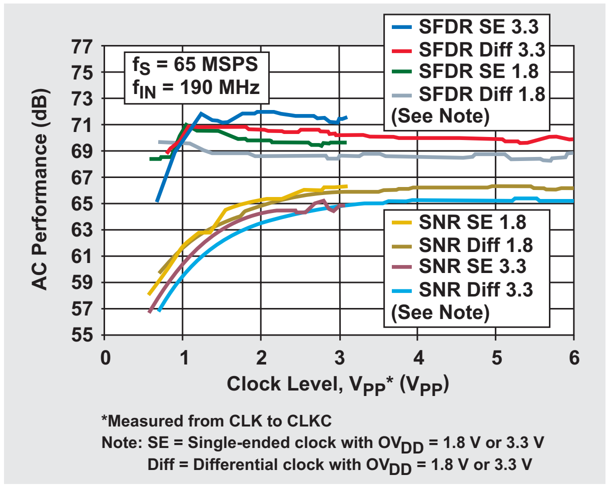


**Figure 8. Measured SNR vs. SNR estimated with Equation 4**

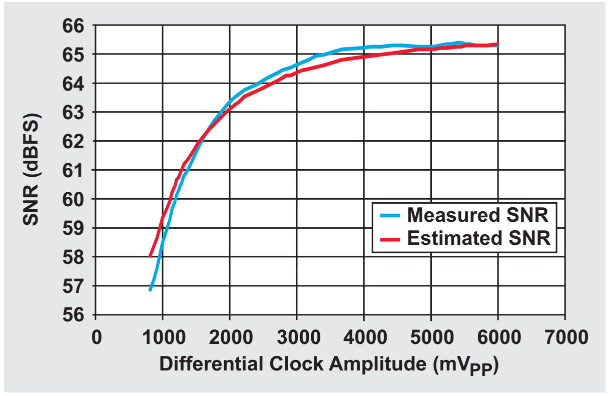


**Table 2. Measured SNR vs. SNR estimated with Equation 4**

| CLOCK ($mV_{PP}$) | MEASURED SNR (dBFS) | ESTIMATED JITTER (ps) | ESTIMATED SNR (dBFS) | SLOPE (V/ns) |
|---|---|---|---|---|
| 820 | 56.79 | 1 | 58.00 | 0.17 |
| 1180 | 59.95 | 0.73 | 60.44 | 0.24 |
| 1380 | 60.97 | 0.64 | 61.35 | 0.28 |
| 1700 | 62.44 | 0.55 | 62.40 | 0.35 |
| 2070 | 63.47 | 0.48 | 63.23 | 0.42 |
| 2400 | 63.96 | 0.44 | 63.74 | 0.49 |
| 2720 | 64.34 | 0.41 | 64.10 | 0.56 |
| 3000 | 64.59 | 0.4 | 64.35 | 0.61 |
| 3420 | 65 | 0.38 | 64.62 | 0.70 |
| 3970 | 65.21 | 0.36 | 64.87 | 0.81 |
| 4570 | 65.28 | 0.34 | 65.06 | 0.93 |
| 5050 | 65.24 | 0.34 | 65.17 | 1.03 |
| 5390 | 65.39 | 0.33 | 65.23 | 1.10 |
| 5680 | 65.26 | 0.33 | 65.27 | 1.16 |
| 6020 | 65.26 | 0.33 | 65.32 | 1.23 |

With this new piece of information, we can now go back to Figure 5 and see if we can estimate the SNR degradation as we decrease the sample rate. Using a process similar to the one we used before, we can estimate the coefficients for Equation 3 that will predict the SNR across sampling and input frequencies with less than 1 dB of error:

$$(\text{Jitter}_{\text{Total}})^2 = (250\text{ fs})^2 + \left(\frac{40\ \mu\text{V}}{\text{Clk\_slope}}\right)^2 \qquad (5)$$

The result of applying this model can be seen in Figure 10a. Figure 10b shows the original data from Figure 5 for easy comparison.

Following are some implications from this model:

- The SNR degradation at lower $f_S$ seen in Figure 5 occurred because the plot was taken with a fixed 3-$V_{PP}$ sinusoidal clock that reduced the slope on its edges as we decreased its frequency.
- A direct implication of this is that the data sheet actually shows worse performance than what a user could obtain with a different clocking scheme.

Figure 9. SNR vs. input frequency

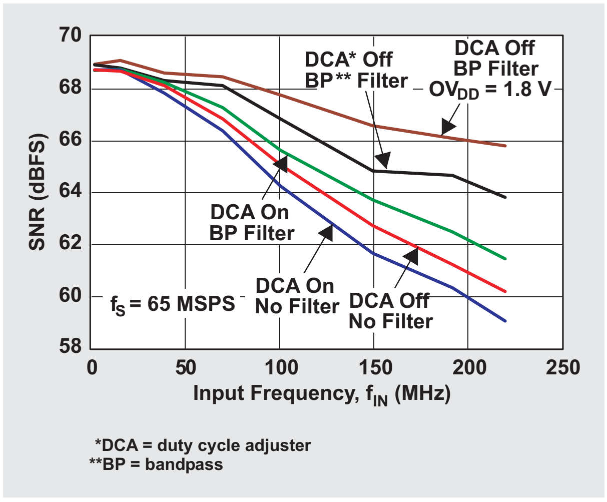
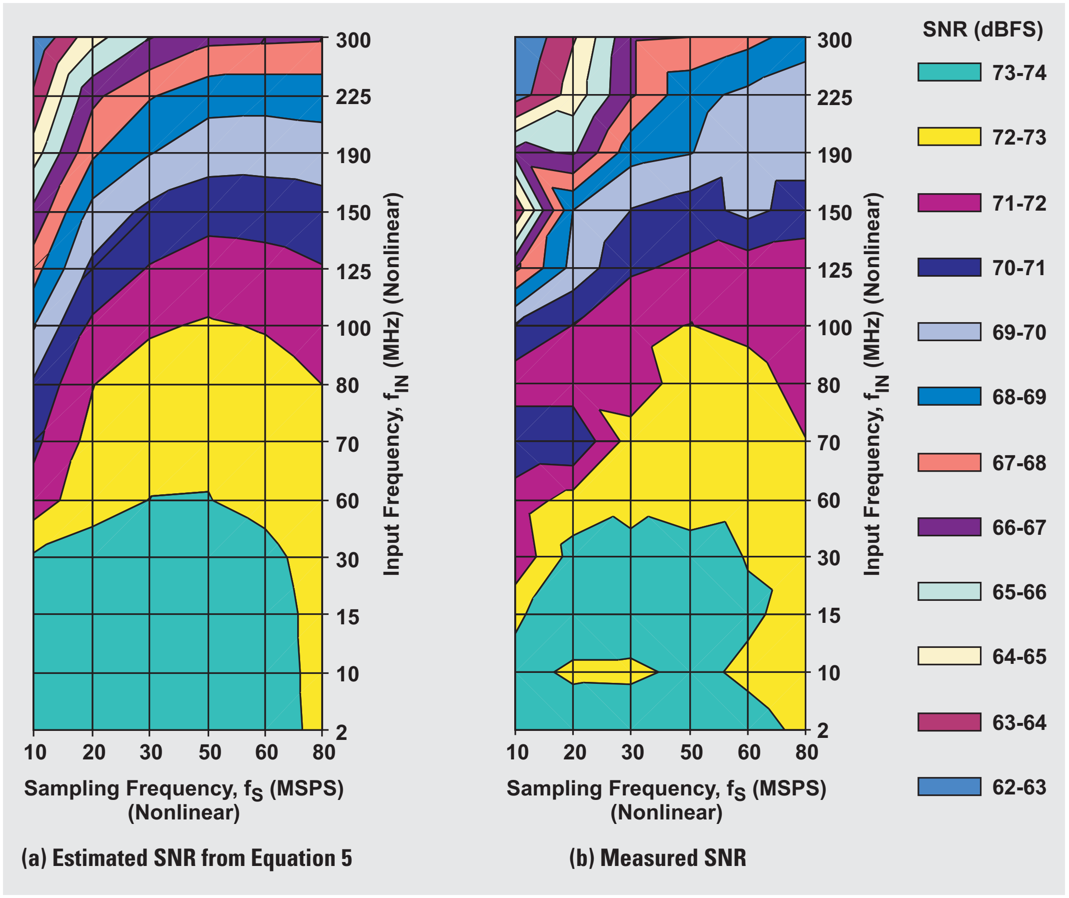


Figure 10. Measured SNR vs. SNR estimated with Equation 5



(a) Estimated SNR from Equation 5

(b) Measured SNR

- A first possible approach is to use a step-up transformer to increase the slope of the sinusoidal clock signal. A pair of clipping diodes (for example, MAX3X71600LCT-ND) can be used to limit the amplitude and avoid exceeding the supply rails of the ADC. This is a clean way of generating a square-like clock signal.
- Another method to square the sinusoidal clock signal is to use an external gate (like a PECL device) acting as a comparator. This minimizes the effect of N1 and N2 but transfers the problem to the equivalent N1 and N2 at the input of the comparator. Many of these commercial circuits have very good jitter numbers, but these numbers stem from the assumption that the input is square; so important degradation could be seen if a sinusoidal clock were used. In many cases the input of the ADC will be a much better squaring circuit than the external gate.
- Ideally we would like to use a very low-jitter clock source with square outputs. One good approximation is the use of a voltage-controlled crystal oscillator (VCXO) with the CDC7005. The use of that circuit will be limited, in principle, by the phase noise quality of the VCXO and some degradation that the CDC7005 may add. Also, this circuit will save a transformer to generate the differential clocking.
- Using an external bandpass filter[3,5] will clean up the jitter on the clock. Nevertheless, the insertion loss of the filter will attenuate the amplitude of the clock, reducing the slope and increasing the effects of N1 and N2. Further amplification (prior to filtering) or a step-up transformer can be used to minimize this attenuation.
- So far we have been estimating jitter indirectly from the SNR degradation, but there could be other reasons for SNR degradation. A method to measure jitter directly on an ADC is described in Reference 5.

## Errors in the quantizer

The quantizer will, after the sample-and-hold, take the voltage across the capacitor and convert it into a digital code. As the S&H is holding the signal steady, the exact time to clock the quantizer is not critical. Nevertheless, other problems arise that are related not to jitter but just to pure timing.

One problem is that in a pipeline ADC, usually both phases of the clock (clock high and clock low) are used. Each stage performs a task during half of the clock and another task during the other half. Both tasks are equally important and require at least a minimum time for accurate execution; so the user has to provide a minimum clock duty cycle. Duty-cycle specifications are usually included on the data sheet of the device. Also, some ADCs (such as the ADS5413) have an internal duty-cycle stabilizer that, when enabled, creates the right internal duty cycle from any external clock duty cycle within a certain range.

Another problem, as ADCs become faster and faster, is to squeeze the maximum performance from the timing design. Open-loop designs of the internal clock circuit tend to leave some margin for supply and temperature variations, which at high clock rates means that time that could be used to settle the stage is being left just for a margin. To optimize the timing, closed-loop designs, like delay locked loop (DLL)-based clocks, can be employed. Many users wonder if the jitter of the DLL will affect the performance. Notice in Figure 1 that the DLL is not in the path of the S&H. Nevertheless, the issue is that basic DLL designs have a range of frequencies of operation; so, if they are designed for the higher clocking rates, they will not be able to operate properly at the lower ones. Also, use of the DLL means that the clock is synchronous—i.e., periodic, not burst or pulsed. For applications requiring lower clock rates or asynchronous clocking, the ADS5500 includes the possibility of bypassing the DLL.

## References

For more information related to this article, you can download an Acrobat Reader file at www-s.ti.com/sc/techlit/*litnumber* and replace *“litnumber”* with the **TI Lit. #** for the materials listed below.

| **Document Title** | **TI Lit. #** |
|---|---|
| 1. “14-Bit, 125MSPS Analog-to-Digital Converter,” ADS5500 Data Sheet, p. 17 . . . . . . | sbas303 |
| 2. A. Zanchi, A. Bonfanti, S. Levantino, and C. Samori, “General SSCR vs. cycle-to-cycle jitter relationship with application to the phase noise in PLL,” *Proceedings of the 2001 IEEE Southwest Symposium on Mixed-Signal Design* (Austin, TX, Feb. 25–27, 2001), pp. 32–37. | — |
| 3. Alfio Zanchi, “How to Calculate the Period Jitter $\sigma_T$ from the SSCR $L(f_n)$ with Application to Clock Sources for High-Speed ADCs,” Application Note . . . . . . . . . . . . . . . . . . . | slwa028 |
| 4. “Single 12-Bit, 65-MSPS IF Sampling Analog-to-Digital Converter,” ADS5413 Data Sheet, p. 9 . . . . . . . . . . . . . . . . . . . . . . . . . | slws153 |
| 5. A. Zanchi, I. Papantonopoulos, and F. Tsay, “Measurement and Spice prediction of sub-picosecond clock jitter in A/D converters,” *Proceedings of the 2003 IEEE International Symposium on Circuits and Systems* (Bangkok, Thailand, May 25–28, 2003), Vol. 5, pp. V-557–V-560. | — |

## Related Web sites

**analog.ti.com**
**www.ti.com/sc/device/***partnumber*
Replace *partnumber* with ADS5413, ADS5500, ADS5542, or CDC7005

# 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**

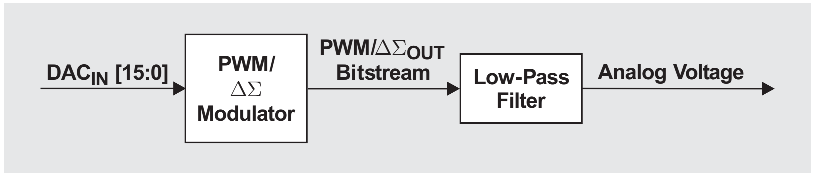
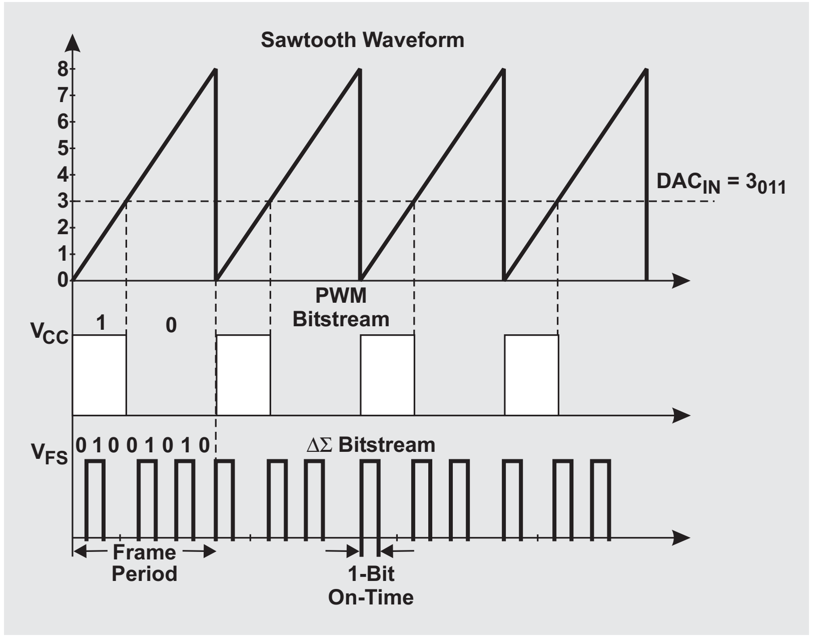


**Figure 2. Bitstream output for PWM and ΔΣ DAC for $DAC_{IN} = 3_{011}$**



## 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**

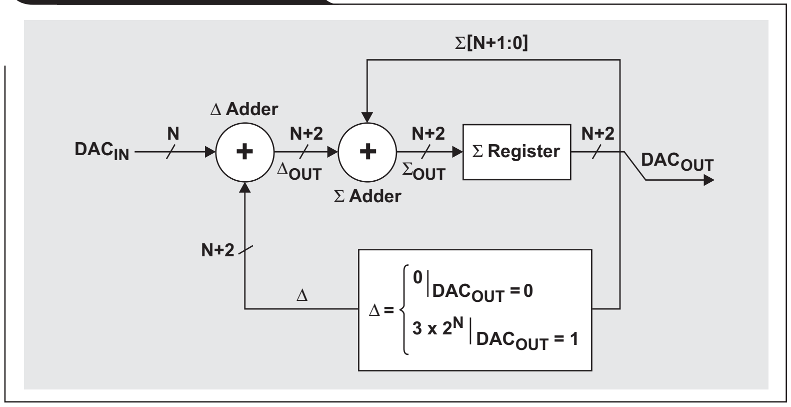


**Table 2. Bitstream and average analog output voltage for 3-bit DAC inputs**

| $DAC_{IN}$ | $DAC_{OUT}$ 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 $\Delta_{OUT}$:

```
If DAC_OUT = 0
        Δ_OUT = DAC_IN;
else
        Δ_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 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.

**Figure 4. Simplified block diagram of ACCSH block**

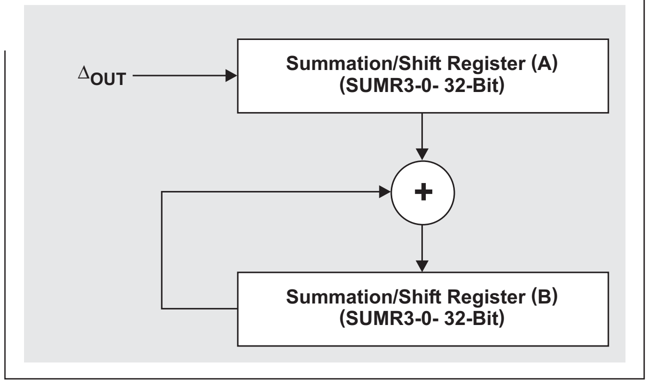
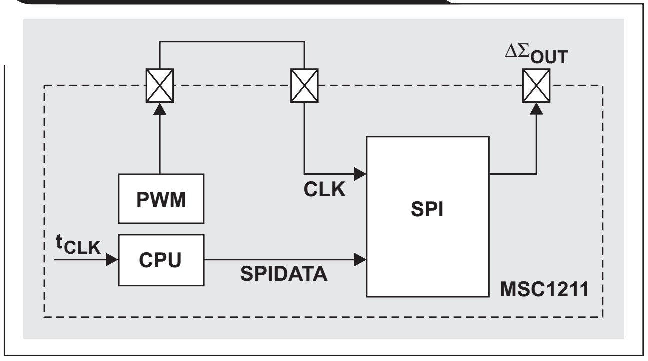

Δ_OUT
Summation/Shift Register (A) (SUMR3-0- 32-Bit)
+
Summation/Shift Register (B) (SUMR3-0- 32-Bit)

**Figure 5. SPI port for DAC output**



## 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 $\Delta_{OUT}$ 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:

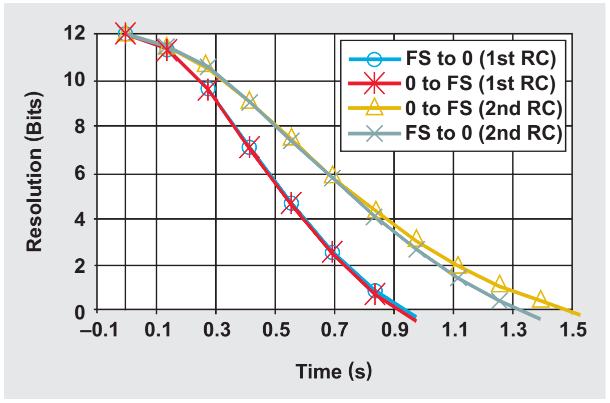
$$H(\omega) = \left|\frac{V_{OUT}}{V_{IN}}\right| = \frac{1}{\sqrt{1+(\omega RC)^2}} \qquad \textbf{(2)}$$

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

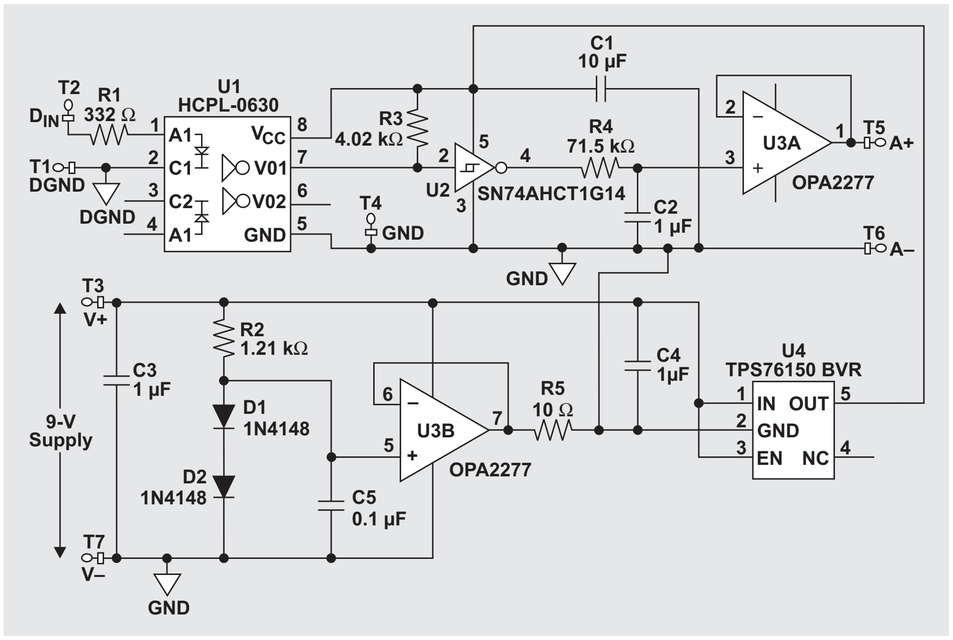


Figure 8. DNL measurements for ΔΣ DAC

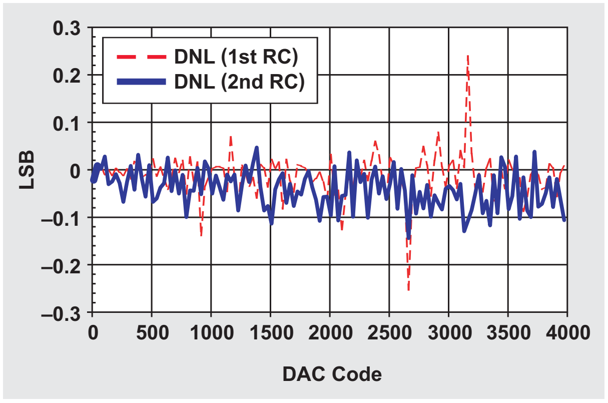
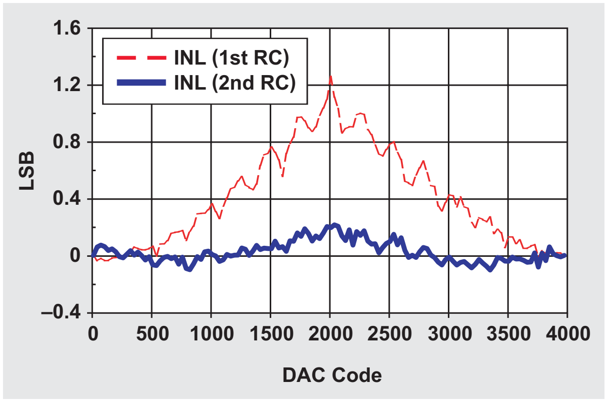


Figure 9. INL measurements for ΔΣ DAC



## 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

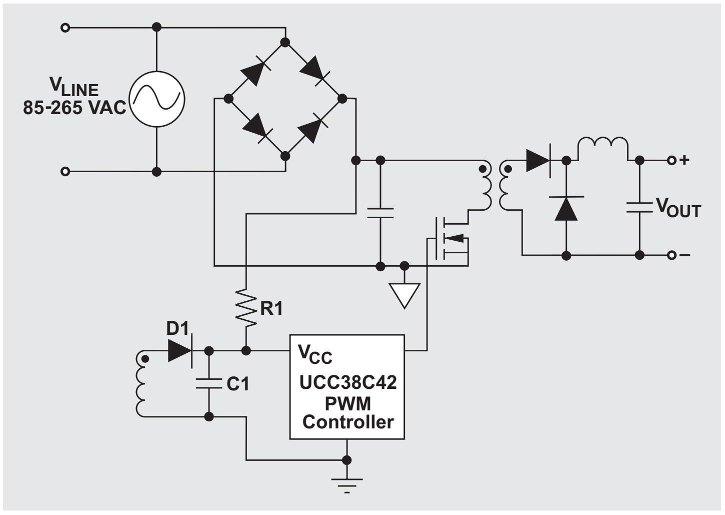
```
#include <REG1210.H>
#include <stdio.h>
void autobaud ();
sbit REQ = P3^7;
sbit ACK = P3^6;
union intU
{
        int i;
        struct{
                unsigned char b1;
                unsigned char b0;
        } byt;
};
union intU dacbuf;
void INT0_isr(void) interrupt 0
{
        T0=1;
        // 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
```

# A better bootstrap/bias supply circuit

**By Michael O'Loughlin** (Email: michael_oloughlin@ti.com)
*Member, Applications Engineering*

In some power-supply applications, the pulse width modulator (PWM) controller is powered up by the configuration in Figure 1 from an auxiliary winding tapped off the power stage's transformer. This technique is used to reduce power loss and keep the overall efficiency high. The auxiliary winding, D1, and C1 provide power and hold-up energy for the PWM. Resistor R1 is used to trickle charge C1 off the input voltage. C1 has to be sized to hold up the supply voltage ($V_{CC}$) long enough for the PWM to start switching the gate of the power MOSFET, causing energy to be stored in the power transformer and delivered to the PWM's $V_{CC}$ through the auxiliary winding. This technique is known as bootstrapping. However, at light-load conditions this circuit is problematic for the power-supply designer.

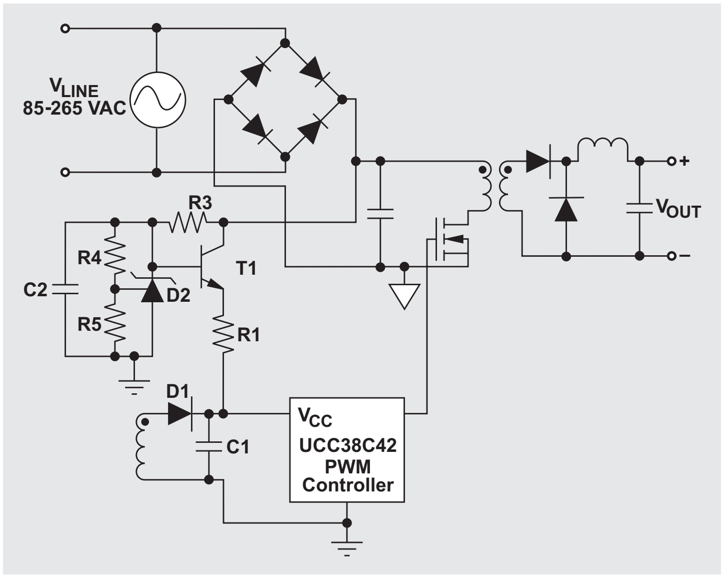
**Figure 1. Traditional bootstrapping/bias supply**



## Problems with traditional bootstrapping/bias supply scheme

Under light-load conditions, C1 has to supply all the energy. C1 generally has to be large enough to hold up $V_{CC}$ for at least 10 switching cycles. However, under light-load and no-load conditions, the current into the PWM ($I_{CC}$) will discharge $V_{CC}$ to the point that the PWM will go into undervoltage lockout (UVLO), causing the output to become unstable. The designer might think that he can reduce the size of R1 or increase the size of C1. Reducing the size of R1 only increases losses and decreases the overall efficiency. Increasing the size of C1 only decreases the start-up time of the PWM. The circuit in Figure 2 will reduce the size of the hold-up capacitor and provide power under no-load conditions while maintaining high efficiencies at the converter's full output power.

**Figure 2. Adding series-pass regulator reduces size of C1**



## Theory of operation

Electrical components C2, D2, R3, R4, R5, and T1 form a series-pass regulator that provides power to the PWM's $V_{CC}$ under light- to no-load conditions. Resistor R1 provides current limiting to protect the series-pass transistor (T1). All of these components together form a bootstrap/bias supply circuit. The series-pass regulator is designed to supply a bias voltage that is less than the bias voltage developed by the auxiliary winding. When the auxiliary winding starts to supply $V_{CC}$ voltage, it produces a voltage large enough to back bias the base emitter junction of T1, causing the series-pass regulator to turn off. This circuit enables the designer to take advantage of lower losses from powering the PWM with an auxiliary winding and also supplies energy to the control circuitry at light loads.

## Setting up the circuit

First we set up the shunt regulator voltage. Knowing the voltage from the auxiliary supply ($V_{AUX}$), we can use the following equation to calculate the shunt voltage ($V_{SHUNT}$):

$$V_{SHUNT} = V_{AUX} - 1\text{ V}$$

It is important to note that $V_{SHUNT}$ minus $V_{AUX}$ should not exceed the maximum reverse voltage of the base-to-emitter junction of T1. This information can be found in the transistor's data sheet.

Once we have decided on the shunt voltage, we can easily set up the series-pass regulator. To minimize losses, R3 is typically sized to allow just enough current to bias the shunt regulator. Typically this is set for twice the shunt regulator's minimal bias current ($I_{D2}$):

$$R3 = \frac{V_{IN} - V_{SHUNT}}{2 \times I_{D2}}$$

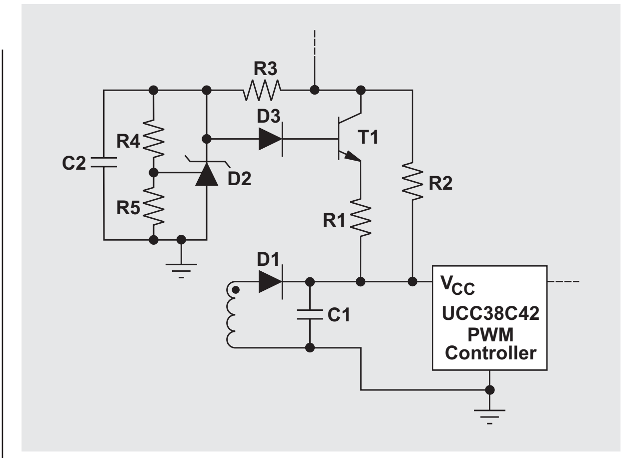
R4 and R5 program the shunt voltage. R4 can be selected by choosing a resistor for R5 and plugging the reference voltage ($V_{REF}$) into the following equation:

$$R4 = R5 \times \frac{V_{SHUNT} - V_{REF}}{V_{REF}}$$

R1 limits the current through T1, protecting it from overcurrent conditions. It can easily be selected based on the maximum current rating ($I_{max}$) and Ohm's law:

$$R1 = \frac{V_{SHUNT} - V_{BE}}{I_{max}}$$

**Figure 3. Lower bias voltage provided by R3 and D3**



C1 is typically sized for hold-up time ($t_{HOLDUP}$) and can be sized by knowing $V_{BIAS}$ and the PWM's UVLO and operating current ($I_{CC}$):

$$C1 = \frac{I_{CC} \times t_{HOLDUP}}{V_{BIAS} - \text{UVLO}}$$

C2 is a hold-up and filtering capacitor for most applications. A 1-µF ceramic capacitor will work fine for this electrical component.

In some applications, due to efficiency requirements, the power-supply designer may want the bias voltage to run lower than the PWM integrated circuit's turn-on threshold. This would require an extra resistor and diode (see Figure 3). R2 is a trickle-charge resistor used for bootstrapping the PWM. It allows capacitor C1 to charge up to the PWM's turn-on threshold voltage. Diode D3 is required to ensure that T1's maximum reverse base-to-emitter voltage is not exceeded. It is also worth mentioning that the shunt voltage would have to be adjusted to accommodate the forward voltage drop of D3.

## Related Web sites

**analog.ti.com**
**www.ti.com/sc/device/UCC38C42**

# Index of Articles

## Data Acquisition

| Title | Issue | Page |
|---|---|---|
| Aspects of data acquisition system design | August 1999 | 1 |
| Low-power data acquisition sub-system using the TI TLV1572 | August 1999 | 4 |
| Evaluating operational amplifiers as input amplifiers for A-to-D converters | August 1999 | 7 |
| Precision voltage references | November 1999 | 1 |
| Techniques for sampling high-speed graphics with lower-speed A/D converters | November 1999 | 5 |
| A methodology of interfacing serial A-to-D converters to DSPs | February 2000 | 1 |
| The operation of the SAR-ADC based on charge redistribution | February 2000 | 10 |
| The design and performance of a precision voltage reference circuit for 14-bit and 16-bit A-to-D and D-to-A converters | May 2000 | 1 |
| Introduction to phase-locked loop system modeling | May 2000 | 5 |
| New DSP development environment includes data converter plug-ins (PDF - 86 Kb) | August 2000 | 1 |
| Higher data throughput for DSP analog-to-digital converters (PDF - 94 Kb) | August 2000 | 5 |
| Efficiently interfacing serial data converters to high-speed DSPs (PDF - 80 Kb) | August 2000 | 10 |
| Smallest DSP-compatible ADC provides simplest DSP interface (PDF - 120 Kb) | November 2000 | 1 |
| Hardware auto-identification and software auto-configuration for the TLV320AIC10 DSP Codec — a “plug-and-play” algorithm (PDF - 105 Kb) | November 2000 | 8 |
| Using quad and octal ADCs in SPI mode (PDF - 94 Kb) | November 2000 | 15 |
| Building a simple data acquisition system using the TMS320C31 DSP (PDF - 235 Kb) | February 2001 | 1 |
| Using SPI synchronous communication with data converters — interfacing the MSP430F149 and TLV5616 (PDF - 182 Kb) | February 2001 | 7 |
| A/D and D/A conversion of PC graphics and component video signals, Part 1: Hardware (PDF - 191 Kb) | February 2001 | 11 |
| A/D and D/A conversion of PC graphics and component video signals, Part 2: Software and control | July 2001 | 5 |
| Intelligent sensor system maximizes battery life: Interfacing the MSP430F123 Flash MCU, ADS7822, and TPS60311 | First Quarter, 2002 | 5 |
| SHDSL AFE1230 application | Second Quarter, 2002 | 5 |
| Synchronizing non-FIFO variations of the THS1206 | Second Quarter, 2002 | 12 |
| Adjusting the A/D voltage reference to provide gain | Third Quarter, 2002 | 5 |
| MSC1210 debugging strategies for high-precision smart sensors | Third Quarter, 2002 | 7 |
| Using direct data transfer to maximize data acquisition throughput | Third Quarter, 2002 | 14 |
| Interfacing op amps and analog-to-digital converters | Fourth Quarter, 2002 | 5 |
| ADS82x ADC with non-uniform sampling clock | Fourth Quarter, 2003 | 5 |
| Calculating noise figure and third-order intercept in ADCs | Fourth Quarter, 2003 | 11 |
| Evaluation criteria for ADSL analog front end | Fourth Quarter, 2003 | 16 |
| Two-channel, 500-kSPS operation of the ADS8361 | First Quarter, 2004 | 5 |
| ADS809 analog-to-digital converter with large input pulse signal | First Quarter, 2004 | 8 |
| Streamlining the mixed-signal path with the signal-chain-on-chip MSP430F169 | Third Quarter, 2004 | 5 |
| Supply voltage measurement and ADC PSRR improvement in MSC12xx devices | First Quarter, 2005 | 5 |
| 14-bit, 125-MSPS ADS5500 evaluation | First Quarter, 2005 | 13 |
| Clocking high-speed data converters | First Quarter, 2005 | 20 |
| Implementation of 12-bit delta-sigma DAC with MSC12xx controller | First Quarter, 2005 | 27 |

## Power Management

| Title | Issue | Page |
|---|---|---|
| Stability analysis of low-dropout linear regulators with a PMOS pass element | August 1999 | 10 |
| Extended output voltage adjustment (0 V to 3.5 V) using the TI TPS5210 | August 1999 | 13 |
| Migrating from the TI TL770x to the TI TLC770x | August 1999 | 14 |
| TI TPS5602 for powering TI’s DSP | November 1999 | 8 |
| Synchronous buck regulator design using the TI TPS5211 high-frequency hysteretic controller | November 1999 | 10 |

## Power Management *(Continued)*

| Title | Issue | Page |
|---|---|---|
| Understanding the stable range of equivalent series resistance of an LDO regulator | November 1999 | 14 |
| Power supply solutions for TI DSPs using synchronous buck converters | February 2000 | 12 |
| Powering Celeron-type microprocessors using TI's TPS5210 and TPS5211 controllers | February 2000 | 20 |
| Simple design of an ultra-low-ripple DC/DC boost converter with TPS60100 charge pump | May 2000 | 11 |
| Low-cost, minimum-size solution for powering future-generation Celeron™-type processors with peak currents up to 26 A | May 2000 | 14 |
| Advantages of using PMOS-type low-dropout linear regulators in battery applications (PDF - 216 Kb) | August 2000 | 16 |
| Optimal output filter design for microprocessor or DSP power supply (PDF - 748 Kb) | August 2000 | 22 |
| Understanding the load-transient response of LDOs (PDF - 241 Kb) | November 2000 | 19 |
| Comparison of different power supplies for portable DSP solutions working from a single-cell battery (PDF - 136 Kb) | November 2000 | 24 |
| Optimal design for an interleaved synchronous buck converter under high-slew-rate, load-current transient conditions (PDF - 206 Kb) | February 2001 | 15 |
| –48-V/+48-V hot-swap applications (PDF - 189 Kb) | February 2001 | 20 |
| Power supply solution for DDR bus termination | July 2001 | 9 |
| Runtime power control for DSPs using the TPS62000 buck converter | July 2001 | 15 |
| Power control design key to realizing InfiniBand℠ benefits | First Quarter, 2002 | 10 |
| Comparing magnetic and piezoelectric transformer approaches in CCFL applications | First Quarter, 2002 | 12 |
| Why use a wall adapter for ac input power? | First Quarter, 2002 | 18 |
| SWIFT™ Designer power supply design program | Second Quarter, 2002 | 15 |
| Optimizing the switching frequency of ADSL power supplies | Second Quarter, 2002 | 23 |
| Powering electronics from the USB port | Second Quarter, 2002 | 28 |
| Using the UCC3580-1 controller for highly efficient 3.3-V/100-W isolated supply design | Fourth Quarter, 2002 | 8 |
| Power conservation options with dynamic voltage scaling in portable DSP designs | Fourth Quarter, 2002 | 12 |
| Understanding piezoelectric transformers in CCFL backlight applications | Fourth Quarter, 2002 | 18 |
| Load-sharing techniques: Paralleling power modules with overcurrent protection | First Quarter, 2003 | 5 |
| Using the TPS61042 white-light LED driver as a boost converter | First Quarter, 2003 | 7 |
| Auto-Track™ voltage sequencing simplifies simultaneous power-up and power-down | Third Quarter, 2003 | 5 |
| Soft-start circuits for LDO linear regulators | Third Quarter, 2003 | 10 |
| UCC28517 100-W PFC power converter with 12-V, 8-W bias supply, Part 1 | Third Quarter, 2003 | 13 |
| UCC28517 100-W PFC power converter with 12-V, 8-W bias supply, Part 2 | Fourth Quarter, 2003 | 21 |
| LED-driver considerations | First Quarter, 2004 | 14 |
| Tips for successful power-up of today's high-performance FPGAs | Third Quarter, 2004 | 11 |
| A better bootstrap/bias supply circuit | First Quarter, 2005 | 33 |

## Interface (Data Transmission)

| Title | Issue | Page |
|---|---|---|
| TIA/EIA-568A Category 5 cables in low-voltage differential signaling (LVDS) | August 1999 | 16 |
| Keep an eye on the LVDS input levels | November 1999 | 17 |
| Skew definition and jitter analysis | February 2000 | 29 |
| LVDS receivers solve problems in non-LVDS applications | February 2000 | 33 |
| LVDS: The ribbon cable connection | May 2000 | 19 |
| Performance of LVDS with different cables (PDF - 57 Kb) | August 2000 | 30 |
| A statistical survey of common-mode noise (PDF - 131 Kb) | November 2000 | 30 |
| The Active Fail-Safe feature of the SN65LVDS32A (PDF - 104 Kb) | November 2000 | 35 |
| The SN65LVDS33/34 as an ECL-to-LVTTL converter | July 2001 | 19 |
| Power consumption of LVPECL and LVDS | First Quarter, 2002 | 23 |
| Estimating available application power for Power-over-Ethernet applications | First Quarter, 2004 | 18 |
| The RS-485 unit load and maximum number of bus connections | First Quarter, 2004 | 21 |
| Failsafe in RS-485 data buses | Third Quarter, 2004 | 16 |

| Title | Issue | Page |
|---|---|---|
| **Amplifiers: Audio** | | |
| Reducing the output filter of a Class-D amplifier | August 1999 | 19 |
| Power supply decoupling and audio signal filtering for the Class-D audio power amplifier | August 1999 | 24 |
| PCB layout for the TPA005D1x and TPA032D0x Class-D APAs | February 2000 | 39 |
| An audio circuit collection, Part 1 (PDF - 93 Kb) | November 2000 | 39 |
| 1.6- to 3.6-volt BTL speaker driver reference design (PDF - 194 Kb) | February 2001 | 23 |
| Notebook computer upgrade path for audio power amplifiers (PDF - 202 Kb) | February 2001 | 27 |
| An audio circuit collection, Part 2 (PDF - 215 Kb) | February 2001 | 41 |
| An audio circuit collection, Part 3 | July 2001 | 34 |
| Audio power amplifier measurements | July 2001 | 40 |
| Audio power amplifier measurements, Part 2 | First Quarter, 2002 | 26 |
| **Amplifiers: Op Amps** | | |
| Single-supply op amp design | November 1999 | 20 |
| Reducing crosstalk of an op amp on a PCB | November 1999 | 23 |
| Matching operational amplifier bandwidth with applications | February 2000 | 36 |
| Sensor to ADC — analog interface design | May 2000 | 22 |
| Using a decompensated op amp for improved performance | May 2000 | 26 |
| Design of op amp sine wave oscillators (PDF - 56 Kb) | August 2000 | 33 |
| Fully differential amplifiers (PDF - 51 Kb) | August 2000 | 38 |
| The PCB is a component of op amp design (PDF - 64 Kb) | August 2000 | 42 |
| Reducing PCB design costs: From schematic capture to PCB layout (PDF - 28 Kb) | August 2000 | 48 |
| Thermistor temperature transducer-to-ADC application (PDF - 97 Kb) | November 2000 | 44 |
| Analysis of fully differential amplifiers (PDF - 96 Kb) | November 2000 | 48 |
| Fully differential amplifiers applications: Line termination, driving high-speed ADCs, and differential transmission lines (PDF - 185 Kb) | February 2001 | 32 |
| Pressure transducer-to-ADC application (PDF - 185 Kb) | February 2001 | 38 |
| Frequency response errors in voltage feedback op amps (PDF - 184 Kb) | February 2001 | 48 |
| Designing for low distortion with high-speed op amps | July 2001 | 25 |
| Fully differential amplifier design in high-speed data acquisition systems | Second Quarter, 2002 | 35 |
| Worst-case design of op amp circuits | Second Quarter, 2002 | 42 |
| Using high-speed op amps for high-performance RF design, Part 1 | Second Quarter, 2002 | 46 |
| Using high-speed op amps for high-performance RF design, Part 2 | Third Quarter, 2002 | 21 |
| FilterPro™ low-pass design tool | Third Quarter, 2002 | 24 |
| Active output impedance for ADSL line drivers | Fourth Quarter, 2002 | 24 |
| RF and IF amplifiers with op amps | First Quarter, 2003 | 9 |
| Analyzing feedback loops containing secondary amplifiers | First Quarter, 2003 | 14 |
| Video switcher using high-speed op amps | Third Quarter, 2003 | 20 |
| Expanding the usability of current-feedback amplifiers | Third Quarter, 2003 | 23 |
| Op amp attenuators | Fourth Quarter, 2003 | 28 |
| Calculating noise figure in op amps | Fourth Quarter, 2003 | 31 |
| Op amp stability and input capacitance | First Quarter, 2004 | 24 |
| Integrated logarithmic amplifiers for industrial applications | First Quarter, 2004 | 28 |
| Active filters using current-feedback amplifiers | Third Quarter, 2004 | 21 |
| **General Interest** | | |
| Synthesis and characterization of nickel manganite from different carboxylate precursors for thermistor sensors (PDF - 194 Kb) | February 2001 | 52 |
| Analog design tools | Second Quarter, 2002 | 50 |

## 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 |
| Finland (English) | +358 (0) 9 25173948 |
| France | +33 (0) 1 30 70 11 64 |
| Germany | +49 (0) 8161 80 33 11 |
| Israel (English) | 1800 949 0107 |
| Italy | 800 79 11 37 |
| Netherlands (English) | +31 (0) 546 87 95 45 |
| Russia | +7 (0) 95 7850415 |
| Spain | +34 902 35 40 28 |
| Sweden (English) | +46 (0) 8587 555 22 |
| United Kingdom | +44 (0) 1604 66 33 99 |
| 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 |
| Australia | 1-800-999-084 |
| China | 800-820-8682 |
| Hong Kong | 800-96-5941 |
| Indonesia | 001-803-8861-1006 |
| Korea | 080-551-2804 |
| Malaysia | 1-800-80-3973 |
| New Zealand | 0800-446-934 |
| Philippines | 1-800-765-7404 |
| Singapore | 800-886-1028 |
| Taiwan | 0800-006800 |
| Thailand | 001-800-886-0010 |
| Fax | 886-2-2378-6808 |
| Email | tiasia@ti.com<br>ti-china@ti.com |
| Internet | support.ti.com/sc/pic/asia.htm |

**A011905**

SLYT072