HEWLETT PACKARD

Analog-to-digital Converter Testing with the HP 16542A

Application Note 1225-4

Analog-to-Digital Converter Testing with the HP 16542A

Selecting a high speed analog-to-digital converter (ADC) for a design requires testing the ADC's performance. Many of these performance tests are difficult to do because they require storing large amounts of data at high clock speeds. The HP 16542A data acquisition card has 2 Mbytes of memory and 100-MHz acquisition speed, making it a good choice for these types of tests. One performance test is the differential non-linearity test.

Differential Non-linearity Test

There are two types of non-linearity errors which affect the performance of an ADC: integral non-linearity and differential non-linearity. Integral non-linearity is not discussed in this application note. Differential non-linearity is defined as the amount of deviation of any quantum (Q) from its ideal value. For an ideal ADC, Q is defined by the following formula:

$Q = FSR/2^n$

Where:

Q = quantum
FSR = full-scale range
n = number of bits in ADC

Differential non-linearity measurement varies with respect to the input frequency. Therefore, the test is normally performed at several different input frequencies.

The differential non-linearity measurement is specified in some manufacturer's ADC specification sheets as the differential linearity error.

Histogram Testing

One method used to make differential non-linearity measurements is called histogram testing. Histogram testing is performed by applying a full-scale sine wave to the input of the ADC. A statistically significant number of ADC output samples are collected and a histogram is created from these samples.
The resulting histogram, when compared against the histogram of an ideal ADC, shows the deviation from the ideal value of each output code.

[1]it is possible for the differential non-linearity error to fail due to excessive integral non-linearity error. For the purposes of this application note, the integral non-linearity error for the ADC being tested is assumed to be below the level which can affect the differential non-linearity measurement.

Creating a Histogram Plot

A histogram is created by plotting the different output codes along the X-axis, and the number of times each code appears divided by the total number of samples along the Y-axis. For an 8-bit ADC, the output codes are 0 to 255 inclusive. The number of times a particular code appears in the output of an ideal ADC, using a sine wave input signal, is given by the following probability density function:

$$P(i^{th}\ code) = \frac{1}{\pi}\left\{ \sin^{-1}\left[\frac{V\,[(I+1) - 2^{n-1}]}{A\,2^{n}}\right] - \sin^{-1}\left[\frac{V\,(I - 2^{n-1})}{A\,2^{n}}\right]\right\}$$

Where:

V = full-scale range of ADC
n = number of bits in ADC
I = output; 0 to 255 for an 8-bit ADC
A = peak amplitude of input sine wave

Calculation of the probability density function for every output code requires that the amplitude of the input frequency be equal to the full-scale range (2A) of the ADC. Setting V equal to the full-scale value, 2A, reduces the equation to:

$$P(i^{th}\ code) = \frac{1}{\pi}\left\{ \sin^{-1}\left[\frac{[(I+1) - 2^{n-1}]}{2^{n-1}}\right] - \sin^{-1}\left[\frac{(I - 2^{n-1})}{2^{n-1}}\right]\right\}$$

Plotting the probability density function for each output code of an ideal 8-bit ADC yields the U-shaped curve shown in figure 1.

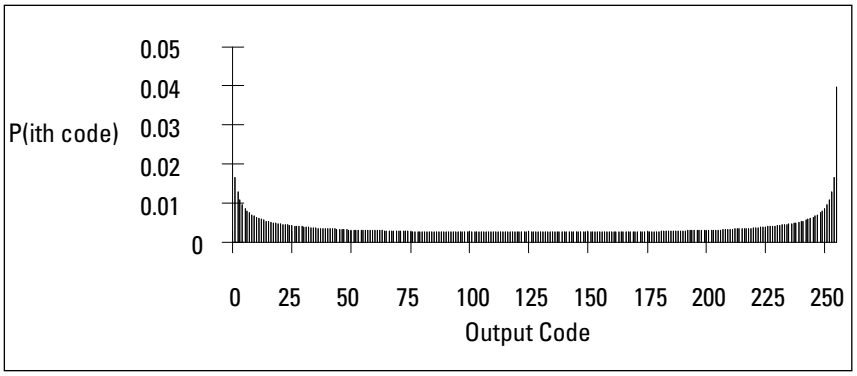

Figure 1. Ideal ADC Probability Density Function

Typically, the histogram for a non-ideal ADC varies from the ideal U-shaped curve. Any output codes in the histogram having zero occurrences are missing codes. Figure 2 shows the histogram for a typical non-ideal ADC.

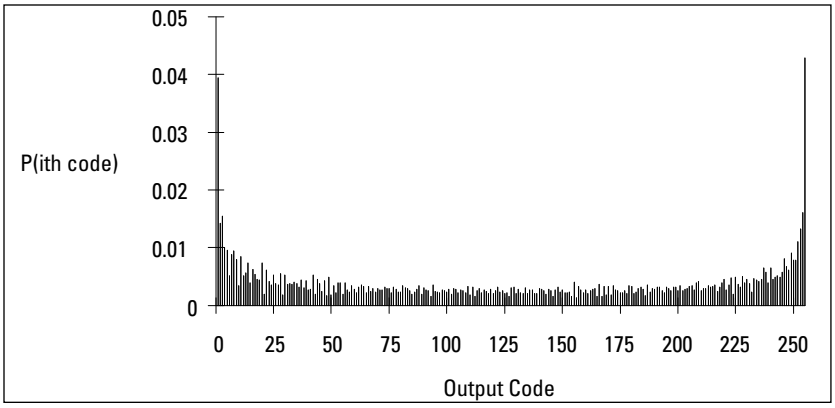

Figure 2. Non-ideal ADC Probability Density Function

Calculating Differential Non-Linearity

The differential non-linearity for each output code is calculated by dividing the probability of an output code for an ADC under test by the probability of an output code for an ideal ADC minus 1. The probability of an output code for an ADC under test is equal to the number of occurrences of an output code divided by the total number of samples. The differential linearity error is calculated using the following equation:

$$DNL = \frac{\text{actual probability } (i^{th} \text{ code})}{\text{ideal probability } (i^{th} \text{ code})} - 1$$

$$DNL = \frac{x_i / X}{p(i^{th} \text{ code})} - 1$$

Where:

DNL = Differential Non-linearity in least significant bits (LSBs)
x_i = number of occurrences of the ith code for ADC under test
X = total number of samples for ADC under test
$P(i^{th}$ code) = probability of ith code for ideal ADC

Selecting the Input Frequency

Testing an ADC using an input frequency which is well below the sample frequency of the ADC requires no special precautions. For example, using an input frequency of 10 kHz and a sample frequency of 100 MHz gives 1000 samples for each cycle of the input frequency. This input frequency provides a good distribution of sample points across the 256 possible values of an 8-bit ADC. However, when using a 12-bit ADC, this may not give good results.

The choice of input frequency for frequencies near to the sampling frequency requires special consideration. This is especially true for input frequencies whose period is an integral multiple of the sample period. For example, a 25 MHz input frequency has a period of 40 ns. If the sample period is 10 ns (100 MHz) then the input period is 4 times that of the sample period.

Sampling the 25 MHz input frequency at 100 MHz produces 3 output codes for each cycle as shown in figure 3.

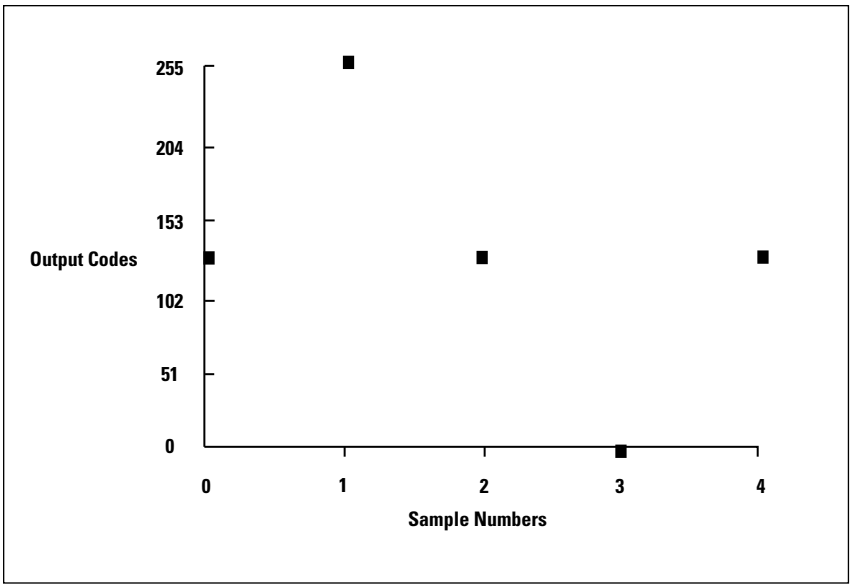

Figure 3. Sample Values for One Cycle of a 25-MHz Input Frequency

Because the period of the input frequency is an integer multiple of the sample period, every cycle is sampled at the same place. This produces the same 3 output codes across several thousand cycles. This is illustrated in figure 4.

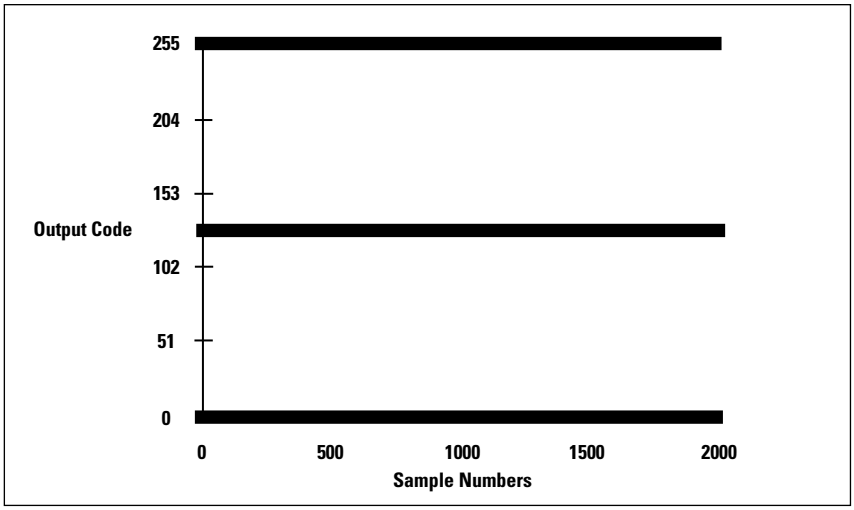

Figure 4. Sample Values for Several Cycles of a 25-MHz Input Frequency

This shows that only 3 out of the 256 total possible codes are present on the output of the ADC. This input frequency provides poor distribution of output codes for histogram testing.

One way to avoid this problem is to choose an input frequency which is close (within 5%) to the input frequency of interest. This causes the sampling positions to change slightly on each cycle of the input sine wave, creating a new frequency called a beat-frequency. The new input frequency can be calculated as follows:

$$f_{new} = f_{old} + \Delta f$$

where:

f_{new} = new input frequency
f_{old} = old imput frequency
Δf = beat-frequency

The beat-frequency is calculated as follows:

$$\Delta f = \frac{K}{MT_5}$$

where:

K = number of desired sampled cycles
M = number located in the Trigger Menu's ***Mem Length*** field
Ts = sample period of ADC

The following shows a sample input frequency calculation:

K = 1000
M = 2,096,128
T_s = 10 ns

$$\Delta f = \frac{1000}{2096128 \text{ x } 10nS} = 47{,}707$$

f_{old} = 25 MHz

f_{new} = 25 MHz + 47707 = 25,047,707

By using this equation to calculate the input frequency, each successive cycle of the input frequency is sampled at a different point resulting in a better distribution of the points across the 256 possible values for an 8-bit ADC.

Sampling several thousand points of this input frequency produces the the graph of figure 5. The result is 4 sine waves of frequency 47,707 Hz that are 90 degrees out of phase.

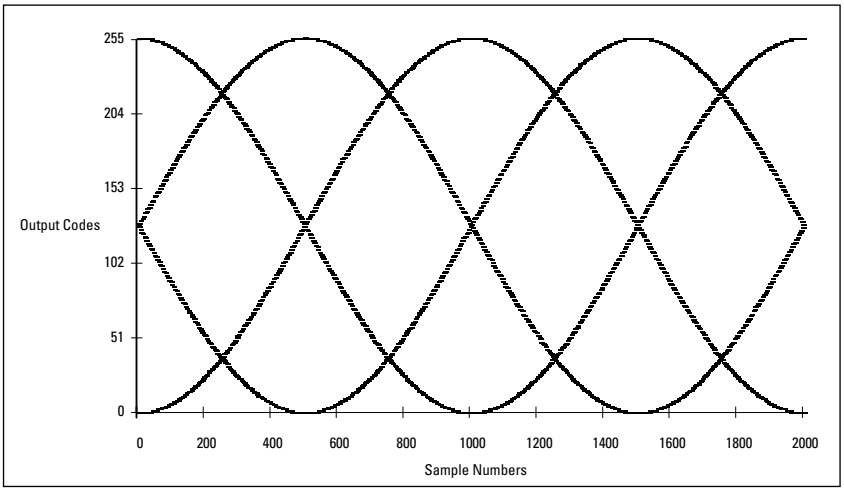

Figure 5. Sample Values for 25.047707-MHz Input Frequency

In practice, it is difficult to find an oscillator which can produce input frequencies this accurately. However, choosing an input frequency of 25.05 MHz is adequate because of the number of samples captured by the HP 16542A.

Test Setup

The following sections show the HP 16542A and test circuit setups used to make a differential non-linearity measurement. Also shown is the results of such a test for a typical 8-bit ADC.

HP 16542A Setup

The differential non-linearity test requires setting the Configuration, the Format, and the Trigger Menus as shown in figure 6 through figure 8.

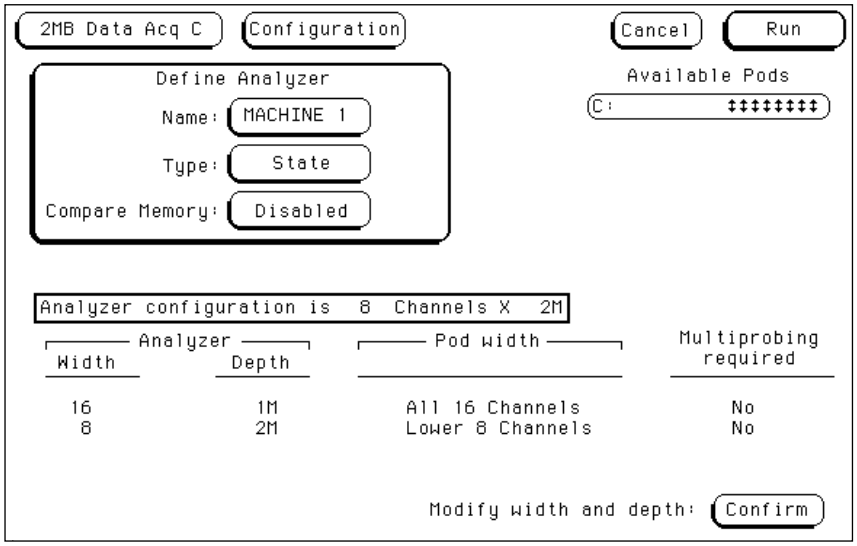

Figure 6. Configuration Menu Setup

The output stage of the ADC used in this test setup uses ECL technology. This requires both the Master Clock and Data Channels thresholds be set to ECL as shown in figure 7.

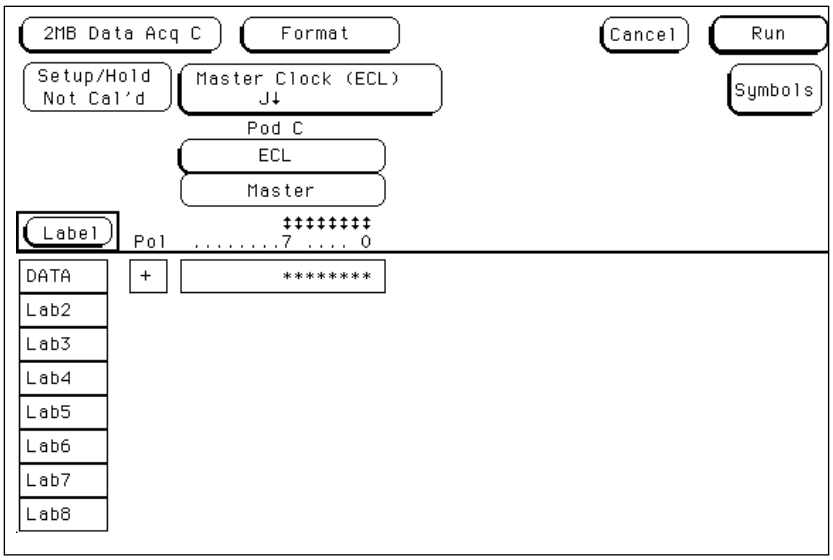

Figure 7. Format Menu Setup

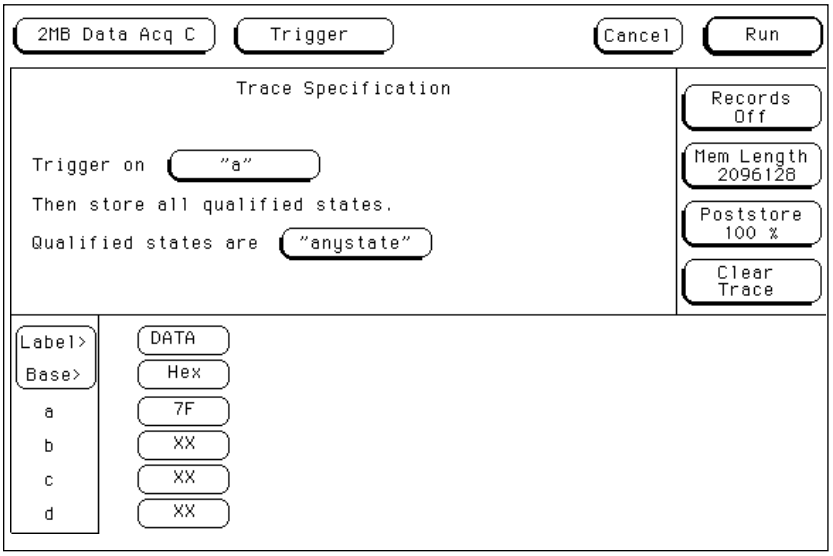

Figure 8. Trigger Menu Setup

The trigger term, 7F, is not important to the measurement and any valid output code can be used.

Test Circuit Setup

Testing an ADC for differential non-linearity requires an ADC evaluation board usually available from the ADC manufacturer. If an evaluation board is not available, you must design and build a test circuit. Figure 9 shows the block diagram of the test circuit used for this example.

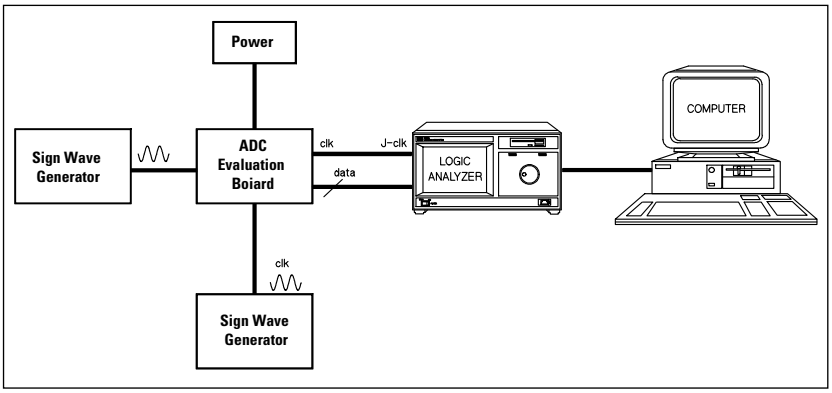

Figure 9. Test Circuit Setup

The logic analyzer Pod C channels 7 through 0 are connected to the ADC's data bits 7 through 0 respectively. The ADC's Clock Out signal is connected to the J-clk channel of Pod C. The peak-to-peak amplitude of the input signal to the ADC must be equal to the full-scale value of the ADC. One way to do this is by using Chart mode.

The logic analyzer's Chart mode plots output code values versus sample time of the captured data. The Chart mode effectively performs a digital-to-analog conversion of the captured data for display.

By using the Chart mode, it is possible to view the output of the ADC while adjusting the amplitude of the input signal until it reaches the full-scale value of the ADC without clipping. This is done by setting the logic analyzer's memory length, found in the Trigger menu, to the minimum required to capture at least one complete sine wave of output samples. Then run the analyzer repetitively while adjusting the amplitude of the input frequency to the ADC until the output reaches full scale. The picture of figure 10 shows an example Chart menu display.

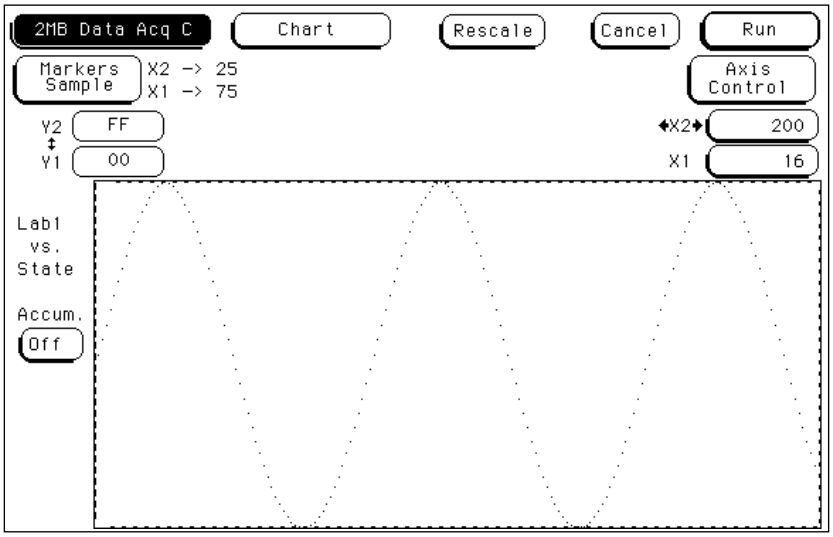

Figure 10. ADC sine wave output

Making the Measurement

Making a measurement requires several steps easily performed using a computer to control and upload information from the HP 16542A over the HP-IB interface. This requires writing a program which implements the following algorithm:

- Set up the HP 16542A Configuration, Format, and Trigger Menus
- Execute Run to capture the ADC under tests output codes
- Upload the output codes of the ADC under test to the computer
- Calculate probability for each output code of the ADC under test
- Calculate probability for ideal ADC output codes
- Calculate differential non-linearity
- Plot the differential non-linearity results

The *HP 16542A Programming Reference Manual* contains example programs written in C which show how commands are sent to the HP 16542A over the HP-IB communications interface.

The example programs, shown at the end of this application note, upload the captured ADC data from the logic analyzer to the computer and calculate the differential non-linearity. The differential non-linearity program saves the calculated data to a file which is imported easily into a spreadsheet. The spreadsheet is then used to plot the results. The graphs shown in this application note were all plotted using a spreadsheet program.

While the calculations are performed by this program, it is also possible to import the data into a spreadsheet program for calculating and plotting the information.

When making the calculations, it is important to note that not all the collected samples are used. The total number of samples collected by the logic analyzer depends on the maximum memory depth setting shown in the Trigger Menu's ***Mem Length*** field. This number most likely is not equal to an integer number of cycles of the sampled input sine wave. Accurate calculation of the probability for each output code requires using an integer number of sampled cycles of the input sine wave. You can calculate the last valid sample number as follows:

$S = f_s/f_o$

$N = \text{trunc}(M/S)$

Where:

f_s = sample frequency

f_o = input frequency or Δf for beat-frequency input signals

M = the number located in the Trigger Menu's ***Mem Length*** field

S = the number of samples per cycle

N = the number of cycles in memory rounded down to the nearest whole number

$L = \text{trunc}(S \times N) - 1$

Where:

L = the last valid sample number rounded down to the nearest whole number

The following shows a sample calculation:

f_s = 100 MHz

f_o = 1.5 MHz

M = 2,096,128

S = 100 MHz/1.5 MHz = 66.66666

N = trunc(2,096,128/66.66666) = 31,441

L = trunc(66.66666 x 31,441) - 1 = 2,096,065

Therefore, only the first 2,096,066 samples of the 2,096,128 total number of samples collected are used when creating the probability density function for the ADC under test. The resulting plot of differential non-linearity for a typical ADC is shown in figure 11.

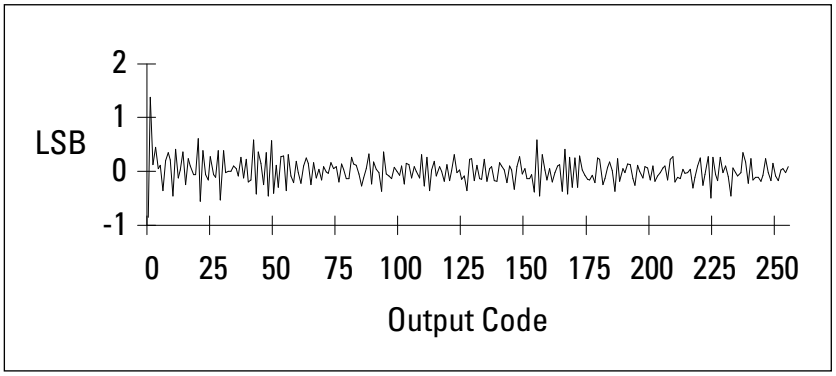

Figure 11. Differential Non-linearity

Higher Frequency ADCs

The plot shows that the worst case differential non-linearity for this ADC is 1.4 LSBs and occurs at output code 1.

Many ADCs currently available have sample rates which far exceed the 100 MHz capture rate of the HP 16542A. However, it is still possible to make the differential non-linearity measurement by modifying the test circuit similar to the example shown in figure 12.

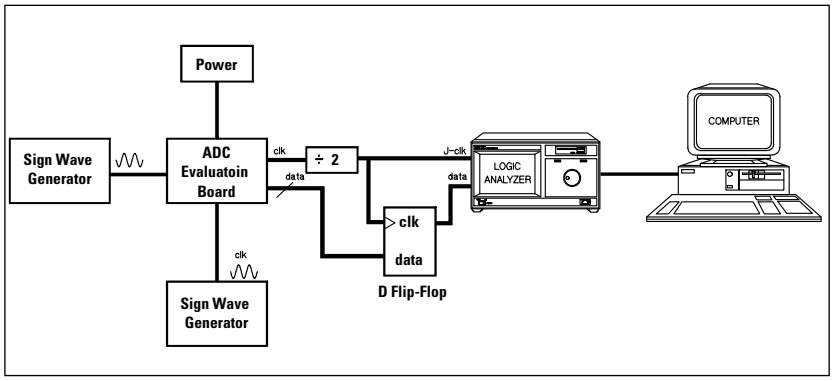

Figure 12. High Frequency Test Circuit

The ADC in this circuit has a sample rate of 200 MHz. Dividing the clock by a factor of two and re-clocking the data from the ADC, effectively reduces the data rate to 100 MHz. This now makes it possible for the HP 16542A to capture every other data sample from the ADC.

When calculating the last valid sample number for this example, 100 MHz is used as the sample period and the input frequency to the ADC is divided by 2.

Conclusion

Measuring an ADC's differential non-linearity error is one way of determining if the ADC being tested meets the requirements of a particular design. The histogram method of determining the differential non-linearity error, shown in this application note, used the HP 16542A logic analyzer module to capture the output codes of the ADC under test.

By writing a program to control the logic analyzer and to upload the collected data to a computer, much of the process of calculating the differential non-linearity error is automated. Once the program is written, testing any ADC requires only a minimum amount of hardware setup time.

Example Programs

Not shown in these programs are the HP-IB communication procedures: Init_IO, Read_IO, Write_IO, and Close_IO. See the Examples Chapter of the *HP 16542A Programmer's Guide* for details of these procedures.

```
/* Upload Programming Example */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_SIZE 16000  /* Maximum buffer size for
                           Read_IO and Write_IO */
#define TRUE 1
#define FALSE 0

/* IO prototypes */

int Init_IO( void );
int Read_IO( int, char *, size_t );
int Write_IO( int, char *, size_t );
int Close_IO( int );

#define CARDC 3      /* HP 16542A occupies card slot C */

void Initialize_16500( void );
int Select( int );
int Read_data( const char * );

void
main( void )
{

   /* Initialize IO port */

   hpib_id = Init_IO();

   /* Initialize the 16500 logic Analyzer */

   Initialize_16500();

   /* Select the HP16542A card */

   Select( CARDC );

   /* Upload the captured ADC data from the analyzer
    * and store it in a file named “adc.dat”
    */

   Read_data( “adc.dat” );

   /* Close the IO port */

   Close_IO( hpib_id );
}


/*************************************************
 * Function Name: Initialize_16500               *
 * Passed Parameters: None                       *
 * Return Value: None                            *
 * Description:                                  *
 *   The function initializes the HP16500 logic  *
 * analyzer by clearing all status registers and *
 * enabling all Service Request Status and Standard *
 * Event Status register bits.                   *
 *************************************************/
void
Initialize_16500( void )
{
   char id[80];
   int term;

   /* Clear the 16500 */
```

```
   Write_IO( hpib_id, "*CLS", 4 );

   /* Enable all Service Request Status Register bits */

   Write_IO( hpib_id, "*SRE 255", 8 );

   /* Enable all Standard Event Status Register bits */

   Write_IO( hpib_id, "*ESE 255", 8 );
}

/*****************************************************
 * Function Name: Select                             *
 * Passed Parameters: card_number - integer          *
 * Return Value: TRUE - successful completion        *
 *               FALSE - unsuccessful completion     *
 * Description:                                      *
 *      The function selects which module in the     *
 * HP16500 card cage receives commands.  This function *
 * must be invoked before sending any commands to the  *
 * HP16542A 100MHz State Analyzer Card.              *
 *****************************************************/
int
Select( int card_number )
{
   char command[10];

   sprintf( command, ":SELECT %d", card_number );

   Write_IO( hpib_id, command, strlen( command ) );

   return( TRUE );
}
/*****************************************************
 * Function Name: Read_data                          *
 * Passed Parameters:                                *
 *     data_file_name - char pointer to the name of the *
 *                      data file to which the binary   *
 *                      data read from the logic        *
 *                      analyzer is stored.             *
 * Return Value: TRUE - successful completion        *
 * Description:                                      *
 *      Open the file specified by the data_file_name   *
 * parameter and write the binary data from the logic   *
 * analyzer to this file.  Returns TRUE when all        *
 * information has been read.                        *
 *****************************************************/
int
Read_data( const char *data_file_name )
{
   char command[20];
   char buffer[MAX_SIZE];
   int bytes;
   FILE *data_file;

   data_file = fopen( data_file_name, "wb" );

   /* Turn off system header information
    * which is not required for binary
    * data upload
    */

   strcpy( command, ":SYSTEM:HEADER OFF" );
   Write_IO( hpib_id, command, strlen( command ) );

   strcpy( command, ":SYStem:DATA?" );
   Write_IO( hpib_id, command, strlen( command ) );

   bytes = Read_IO( hpib_id, buffer, (size_t) MAX_SIZE );
```

```
   while( bytes == MAX_SIZE )
   {
      fwrite( buffer, 1, size of( buffer ), data_file );
      bytes = Read_IO( hpib_id, buffer, (size_t) MAX_SIZE );
   }

   fwrite( buffer, 1, (size_t) bytes, data_file );

   fclose( data_file );
   return( TRUE );
}

/* Differential Non-linearity Programming Example */
#include <stdio.h>
#include <math.h>
#include <string.h>

#define MAX_CODES    /* Maximum number of output codes */
#define M_PI 3.14159265358979323846
#define HEADER_SIZE 98

void
main( void )
{
   double count[MAX_CODES];
   double P_Actual[MAX_CODES];
   double P_Ideal[MAX_CODES];
   double Diff[MAX_CODES];
   FILE *binary_file;
   int i;
   double x;
   double y;
   double sample_count = 0.0;
   char buffer[HEADER_SIZE];
   char byte[1];
   FILE *output;

   /* Open the uploaded data file */

   binary_file = fopen( “adc.dat”, “rb” );

   /* Read past the header to start of data */

   fread( buffer, 1, HEADER_SIZE, binary_file );

   /* Initialize output code count array to 0 */

   for( i = 0; i < MAX_CODES; i++ )
      count[i] = 0.0;

   /* Count the number of times each output code
    * appears in the uploaded data file using
    * only the first 2096066 samples.  See the
    * Making the Measurement section for an explanation.
    */

   while( sample_count < 2096066 )
   {
      fread( byte, 1, 1, binary_file );
      i = (int) ( 0x00ff & byte[0]);
      count[i] = count[i] + 1.0;
      sample_count = sample_count + 1.0;
   }

   /* Open the file to store probability
    * density function data for the ADC
    * under test
    */
```

```
   output = fopen( “pdata.act”, “w” );
   for( i = 0; i < MAX_CODES; i++ )
      fprintf( output, “%d,”, i );
   fprintf( output, “\n” );

   /* Calculate the probability density function
    * for the ADC and store the values in a lotus
    * spreadsheet import format.
    */

   for( i = 0; i < MAX_CODES; i ++ )
   {
      P_Actual[i] = count[i]/sample_count;
      fprintf( output, “%f,”, P_Actual[i] );
   }

   fprintf( output, “\n” );
   fclose( output );

   /* Open the file to store probability
    * density function data for an ideal ADC
    */

   output = fopen( “pdata.idl”, “w” );
   for( i = 0; i < MAX_CODES; i++ )
      fprintf( output, “%d,”, i );
   fprintf( output, “\n” );

   /* Calculate the probability density function
    * for an Ideal ADC and store the values in a
    * lotus spreadsheet format.
    */

   for( i = 0; i < MAX_CODES; i++ )
   {
      x = (double) (i-127.0)/128.0;
      y = (double) (i-128.0)/128.0;
      P_Ideal[i] = (1/M_PI) * (asin( x ) - asin( y ));
      fprintf( output, “%f,”, P_Ideal[i] );
   }

   fprintf( output, “\n” );
   fclose( output );


   /* Open the file to store the differential
    * non-linearity data for the ADC
    * under test
    */

   output = fopen( “pdata.dif”, “w” );
   for( i = 0; i < MAX_CODES; i++ )
      fprintf( output, “%d,”, i );
   fprintf( output, “\n” );

   /* Calculate the differential non-linearity
    * of the ADC and store the values in a
    * lotus spreadsheet format.
    */
   for( i = 0; i < MAX_CODES; i++ )
   {
      Diff[i] = (P_Actual[i]/P_Ideal[i]) - 1.0;
      fprintf( output, “%f,”, Diff[i] );
   }

   fprintf( output, “\n” );
   fclose( output );
}
```

HEWLETT PACKARD

For more information on Hewlett-Packard Test & Measurement products, applications or services please call your local Hewlett-Packard sales offices. A current listing is available via Web through AccessHP at http://www.hp.com.
If you do not have access to the internet please contact one of the HP centers listed below and they will direct you to your nearest HP representative.

United States:
Hewlett-Packard Company
Test and Measurement Organization
5301 Stevens Creek Blvd.
Bldg. 51L-SC
Santa Clara, CA 95052-8059
1 800 452 4844

Canada:
Hewlett-Packard Canada Ltd.
5150 Spectrum Way
Mississauga, Ontario
L4W 5G1
(905) 206 4725

Europe:
Hewlett-Packard
European Marketing Centre
P.O. Box 999
1180 AZ Amstelveen
The Netherlands

Japan:
Hewlett-Packard Japan Ltd.
Measurement Assistance Center
9-1, Takakura-Cho, Hachioji-Shi,
Tokyo 192, Japan
Tel: (81-426) 48-0722
Fax: (81-426) 48-1073

Latin America:
Hewlett-Packard
Latin American Region Headquarters
5200 Blue Lagoon Drive
9th Floor
Miami, Florida 33126
U.S.A.
(305) 267 4245/4220

Australia/New Zealand:
Hewlett-Packard Australia Ltd.
31-41 Joseph Street
Blackburn, Victoria 3130
Australia
131 347 ext. 2902

Asia Pacific:
Hewlett-Packard Asia Pacific Ltd
17-21/F Shell Tower, Times Square,
1 Matheson Street, Causeway Bay,
Hong Kong
(852) 2599 7070

Printed in U.S.A 6-96
5091-5448E