ADC Functional Performance Characterization on the Intel® Quark™ microcontroller D1000

White Paper

October 2015

Document Number: 332915-001US

Contents

Figures

Tables

Revision History

Date	Revision	Description
October 2015	0.1	Initial release.

§

1 Abstract

The Intel® Quark™ microcontroller D1000 has an integrated SAR ADC. While the resolution of this ADC is twelve bits, errors internal to the ADC reduce its precision to something less than twelve bits. This precision is known as effective number of bits (ENOB). In addition to internal errors, impairments such as noise and distortion can further degrade precision. It is important for the user to have some realistic expectations regarding ADC precision. This paper presents a method for characterizing the performance of that ADC in the Intel® Quark™ microcontroller D1000 operating in its typical user configuration. This paper reviews the quantitative measures characterized, identifies impairments that could degrade those measurements, describes the methods used to characterize ADC performance, presents results, discusses outcomes, and draws conclusions based on those outcomes.

§

2 Introduction

The first objective of this paper is to familiarize the user with ADC performance parameters and present methods for quantifying ADC performance limitations when the Intel® Quark™ microcontroller D1000 is operating in functional mode. ADC performance is characterized by the quantitative measures signal to noise ratio (SNR), total harmonic distortion (THD), spurious free dynamic range (SFDR), signal to noise and distortion ratio (SINAD), effective number of bits (ENOB), integral non-linearity (INL) and differential non-linearity (DNL). These quantities are each defined in IEEE Std 1241™-2010 [1] and. discussed in detail in the methods section below.

The second objective is to characterize ADC performance while the Intel® Quark™ microcontroller D1000 operates in functional mode (the typical user mode, rather than any factory test mode). The Intel® Quark™ microcontroller D1000 has test modes that allow automated test equipment to control the ADC directly, but it's important to know what the performance limits are with the Intel® Quark™ microcontroller D1000 executing a user's application.

Performance is impaired when I/O switching noise couples to the analog signal. In order to measure performance limitations of the ADC, it is essential that harmonic distortion and noise on the test signal are well below ADC limits. Therefore, this paper outlines methods to minimize the coupling of I/O switching noise to the analog test signal.

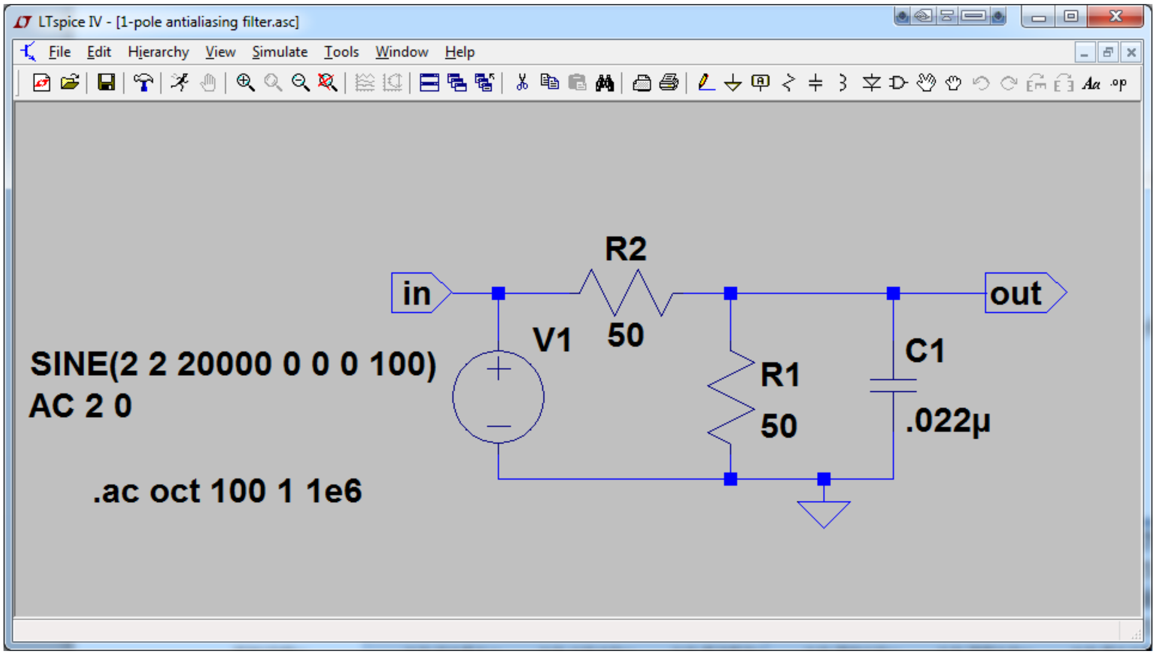
§

3 Methods

The best-known methods of measuring ADC performance require large amounts of memory and floating point arithmetic. The Intel® Quark™ microcontroller D1000 has no floating point unit and its internal memory is limited to 8 kB of SRAM – not nearly enough to measure ADC performance to twelve bits of precision. Therefore, a method is needed to get ADC sample data out of the Intel® Quark™ microcontroller D1000 and into a host computer for processing. This paper outlines two methods that are described in detail in Section 3.6 on GPIO and Section 3.7 on SPI.

In both methods, the MCU is configured to use an external clock source and to receive coherent clock and test signals from an Agilent 33522A Function Generator, with a 22 nF capacitor placed between the analog input used for test and ground. This capacitor, along with 50 Ω source and load resistances, implements the single pole antialiasing filter shown in Figure 1. This filter has the frequency and phase response shown in Figure 2.

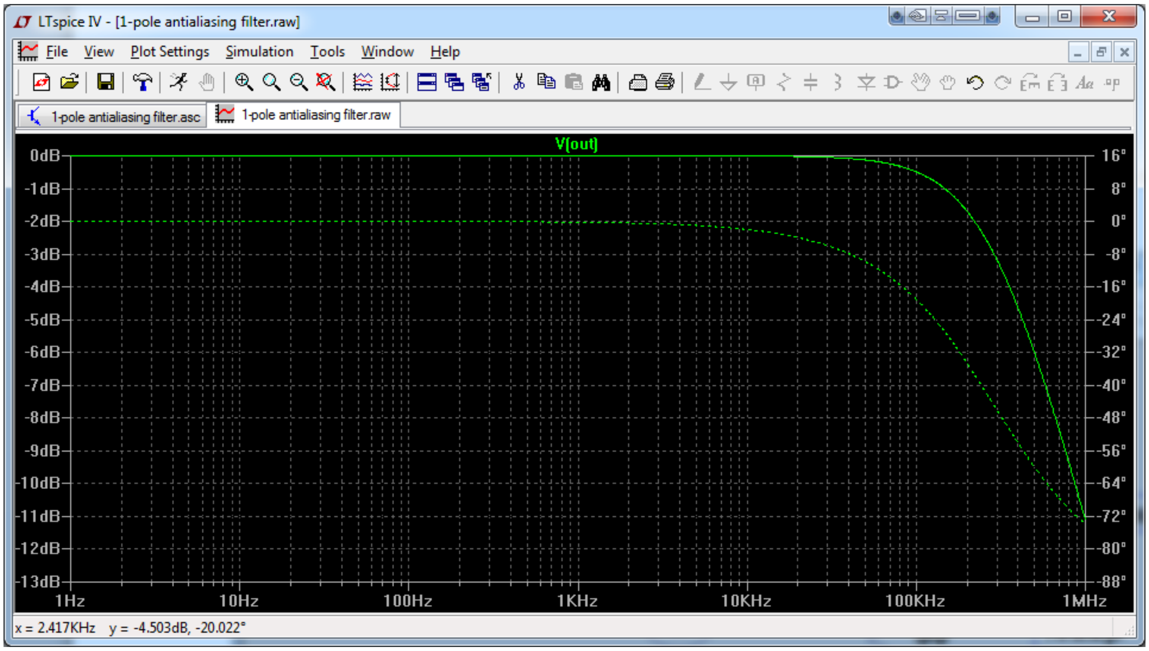
Figure 1. Schematic Diagram of Single Pole Antialiasing Filter Utilizing 50Ω Source and Load Resistance


LTspice IV - [1-pole antialiasing filter.asc]
File Edit Hierarchy View Simulate Tools Window Help
in
R2
50
out
V1
R1
50
C1
.022µ

```
SINE(2 2 20000 0 0 0 100)
AC 2 0

.ac oct 100 1 1e6
```

Figure 2. Bode Plot of Single Pole Antialiasing Filter Utilizing 50Ω Source and Load Resistance



x = 2.417KHz y = -4.503dB, -20.022°

IEEE Std 1241™-2010 [1] defines quantitative measures of ADC performance. This paper is concerned with SNR, THD, SFDR, SINAD, ENOB, INL and DNL. Multiple ways of measuring the quantities are described, including using the coherent sampling method and averaging the magnitude spectra of a test tone. The test tone frequency selected maximizes the number of unique phase samples while being orthogonal to the sample rate.

Section 5.4 of IEEE Std 1241™-2010 [1] gives the following guidance:

$$f_t = \frac{J}{N} f_s$$

…where f_t is the test tone frequency, f_s is the sample rate, N is the FFT size and J is relatively prime to N. Average magnitude spectra are defined in clause 8.8.1.1 as:

$$X_{avm}(f) = \frac{1}{K} \sum_{k=1}^{K} |X_k(f)|$$

…where f is the frequency of interest, K is the number of spectra to be averaged, and $|X_k(f)|$ is the magnitude of the k^{th} complex spectral coefficient at frequency f. Spectrum averaging does not reduce ADC noise or distortion; it reduces variation in the measurement of noise and distortion producing a high fidelity estimate of these quantities. The FFT is used to calculate the complex spectral coefficients.

The length of the FFT and the number of magnitude spectra to average are largely determined by trial and error. However, the FFT noise floor must be well below the expected noise and distortion of the ADC. The FFT noise floor is defined as SQNR plus FFT processing gain. Expressed in dB:

$$\eta_{FFT} = B \times 20\log_{10}(2) + 10\log_{10}\left(\frac{3}{2}\right) + 10\log_{10}\left(\frac{N}{2}\right)$$

…where B is the bit precision and N is the number of points. The following sections provide a brief description of each of the performance measures.

3.1 Signal to Noise Ratio

SNR is defined in IEEE Std 1241™-2010 [1] clause 9.3 as:

$$SNR = \frac{A_{rms}}{\eta}$$

…where A_{rms} is the RMS average amplitude of the test tone and η is the amplitude of the noise. This ratio is normally expressed in dB. Note that harmonics of the test tone are not included in either factor.

3.2 Total Harmonic Distortion

THD is defined in IEEE Std 1241™-2010 [1] clause 8.8.1 as:

$$THD = \frac{\sqrt{\frac{1}{M^2}\sum_h |X(f_h)|^2}}{A_{rms}}$$

…where M is the number of frequency bins, h are the test tone harmonics, $X(f_h)$ are the complex spectral coefficients at the harmonic frequencies, and A_{rms} is the RMS average amplitude of the test tone. This ratio is normally expressed in dB. Note that this paper only includes harmonics up to the Nyquist frequency in the calculations and relies on an anti-aliasing filter to attenuate those harmonics above Nyquist.

3.3 Spurious Free Dynamic Range

SFDR is defined in IEEE Std 1241™-2010 [1] clause 8.8.2 as:

$$SFDR = 20\log_{10}\left[\frac{A_{rms}}{A_{nf}}\right]$$

…where A_{rms} is the RMS average amplitude of the test tone and A_{nf} is the RMS average of the remaining non-fundamental frequencies. This range is expressed in dB.

3.4 Signal to Noise and Distortion Ratio

SINAD is defined in IEEE Std 1241™-2010 [1] clause 8.8.1 as:

$$SINAD = \frac{A_{rms}}{A_{nad}}$$

…where A_{rms} is the RMS average amplitude of the test tone and $A_{nad}$is the RMS average amplitude of noise and distortion. $A_{nad}$is defined in clause 9.2.2 as:

$$A_{nad} = \frac{1}{\sqrt{M(M-3)}} \sqrt{\sum_{n \in S_0} X_{avm}(f_n)}$$

…where M is the number of frequency bins, S_0 is the set of all frequency bins not including DC and the two corresponding to the test tone, and $X_{avm}(f_n)$ is the averaged spectral magnitude for frequency f_n. SINAD is normally expressed in dB.

3.5 Effective Number of Bits

ENOB expresses the effective precision of the ADC based on the full scale root mean squared signal amplitude versus the root mean squared error. This is customarily computed from SINAD:

$$ENOB = \frac{SINAD - 10\log_{10}\left(\frac{3}{2}\right)}{20\log_{10}(2)}$$

…where $SINAD$ is expressed in dB.

3.6 Transmitting ADC Samples using GPIO

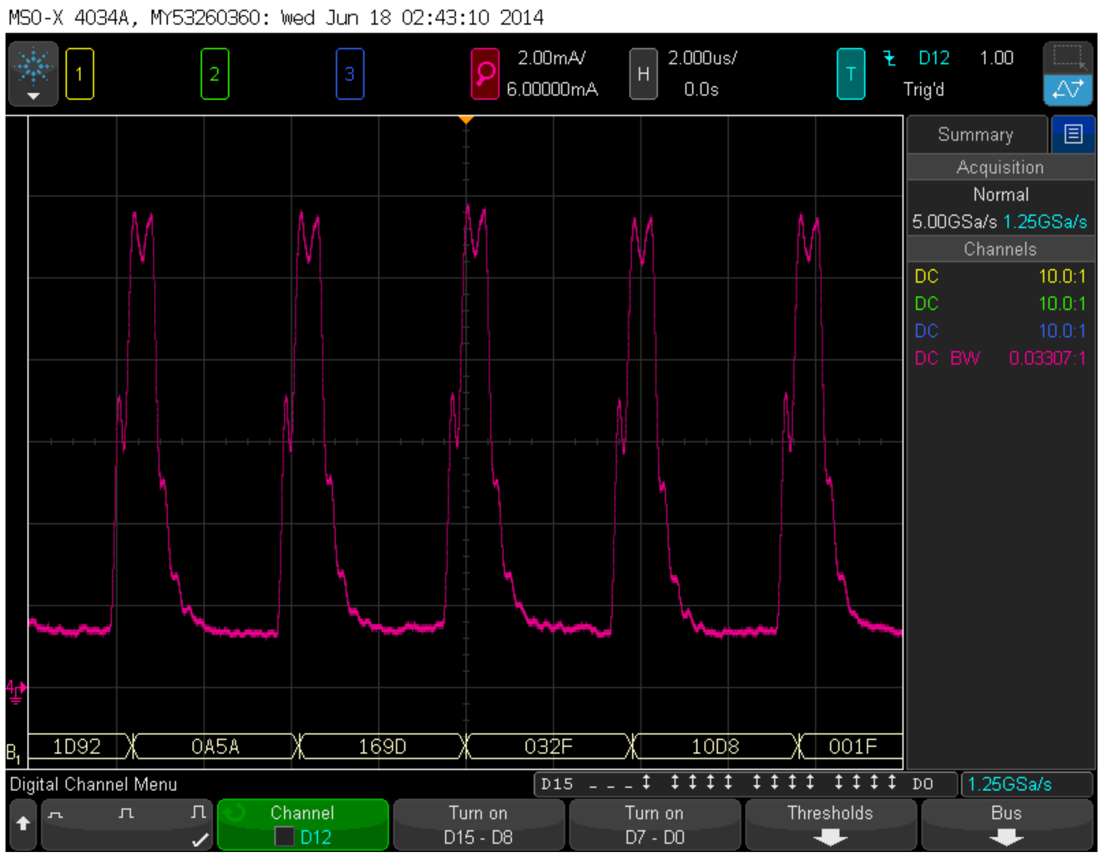
One method of getting samples out of the Intel® Quark™ microcontroller D1000 and into a PC is GPIO. In this method, a program running on the Intel® Quark™ microcontroller D1000 outputs ADC samples on GPIO. A logic analyzer or oscilloscope collects and stores the samples. A PC reads the stored samples from oscilloscope via USB and processes them using MATLAB.

One of the difficulties of using this method is the large interference on the analog signal when GPIO changes state. To minimize this interference, the program running on the Intel® Quark™ microcontroller D1000 puts the system into ADC sleep state between samples. In ADC sleep state, all clocks in the entire system except ADC are halted. When a sample is available from the ADC, the system clock resumes, a sample is unloaded from the ADC FIFO and the sample is written to GPIO. The sample interval immediately precedes the wake-up signal that resumes system operation. This ensures that interference is at a minimum during the sample interval. A waveform showing core supply current along with GPIO is shown in Figure 3.

Looking carefully at the B_1 bus in Figure 3, it is evident that that bit 12 is toggling. This is a DDR clock that the program running on the Intel® Quark™ microcontroller D1000 outputs along with the sample data on bits 0 through 11.

The MATLAB algorithm uses this clock to find the bit centers. After finding the bit centers, the MATLAB algorithm performs all of the calculations described above and displays the results.

Figure 3. Core Current Waveform on Channel 4 in Red and ADC Data on Bus 1 in White Showing Quiet During ADC Sample Interval Just Prior to Wake Up



MSO-X 4034A, MY53260360: Wed Jun 18 02:43:10 2014

2.00mA/ 6.00000mA

H 2.000us/ 0.0s

T D12 1.00 Trig'd

Acquisition: Normal, 5.00GSa/s 1.25GSa/s

Channels: DC 10.0:1; DC 10.0:1; DC 10.0:1; DC BW 0.03307:1

B1: 1D92 0A5A 169D 032F 10D8 001F

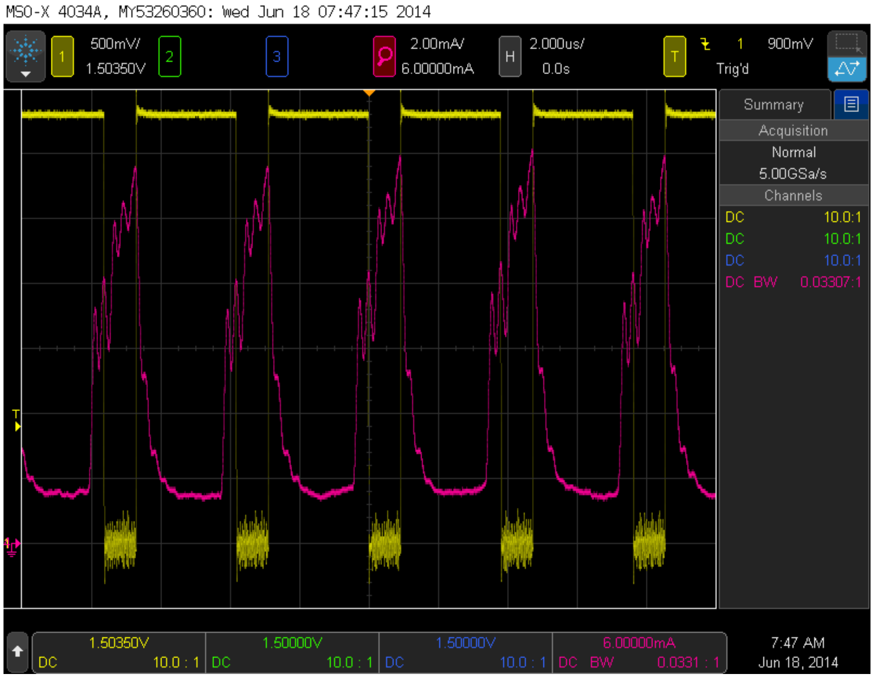
For consistent results, average as many discrete spectra as practical. By turning off all channels except the B_1 bus and adjusting the oscilloscope time base to approximately four times oversampling, it is possible to collect enough ADC samples to compute eleven discrete spectra.

3.7 Transmitting ADC Samples using SPI

Another method of getting samples out of the Intel® Quark™ microcontroller D1000 and into a PC is SPI. In this method, a program running on the Intel® Quark™ microcontroller D1000 moves samples as they become available from the ADC FIFO to SPI. A Beagle* I²C/SPI protocol analyzer from Total Phase* sniffs the SPI traffic and sends it to the PC via USB. The PC processes the samples using MATLAB.

One of the difficulties of using this method is the large interference during SPI transfers. To minimize this interference, the program running on the Intel® Quark™ microcontroller D1000 puts the system into ADC sleep between samples. In the ADC sleep state, all clocks in the entire system except ADC are. When a sample is available from the ADC, the system clock resumes, a sample is moved from the ADC FIFO to the SPI transmit FIFO. The sample interval immediately precedes the wake-up signal that resumes system operation. This ensures that interference is at a minimum during the sample interval. A waveform showing core supply current along with SPI slave select is shown in Figure 4.

Figure 4. Core Current Waveform on Channel 4 in Red and SPI Slave Select on Channel 1 in Yellow Showing Quiet During ADC Sample Interval Just Prior to Wake Up



For consistent results, average as many discrete spectra as practical. By adjusting the size of the capture buffer, it is possible to collect enough ADC samples to compute 44 discrete spectra. This produced consistent results.

§

4 Results

The ADC was tested with a clock frequency of 29,360,128 Hz, a sample rate of 262,144 Sps, an FFT size of 65,536 points, and a test tone frequency of 19,996 Hz. This choice of parameters conforms to the guidance given in the IEEE 1241™-2010 standard [1], yields six harmonics below Nyquist for distortion estimation, provides sufficient time between samples to fully enter ADC sleep, and yields consistent results close to theoretical expectations. The MATLAB script prints results in the command window, an example of which is shown below. In addition, the script plots power spectrum, sample histogram, INL and DNL, such those shown in Figure 5, Figure 6, and Figure 7.

```
Fc     = 29360128.000000000000000 Hz
Fs     =   262144.000000000000000 Hz
Fin    =    19996.000000000000000 Hz
Res BW =        4.000000000000000 Hz
Filename [Trace.mat]:
Peak   =     1948.5235 LSB
Offset =     2051.9438 LSB
SNR    =       69.3643 dB
THD    =      -80.4342 dB
SFDR   =       81.3998 dB
SINAD  =       69.0373 dB
ENOB   =       11.1744
DNL    = [-0.4583, 0.7751] LSB
INL    = [-0.5496, 0.8579] LSB
First full code bin =  105
Last full code bin  = 4001
No missing codes
```

Figure 5. Example Power Spectrum Plot from ADC Test of Sample Number 5 Using GPIO Method

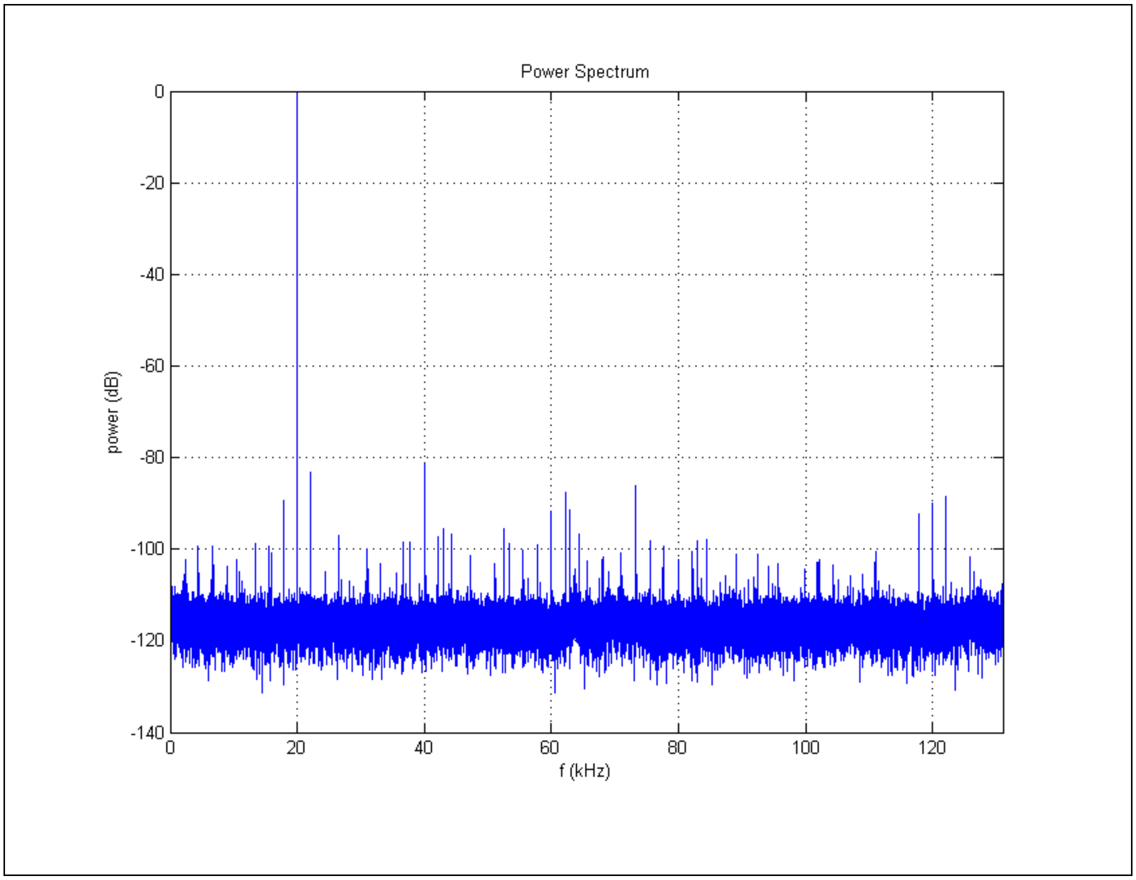


Figure 6. Example Sample Histogram Plot from ADC Test of Sample Number 5 Using GPIO Method

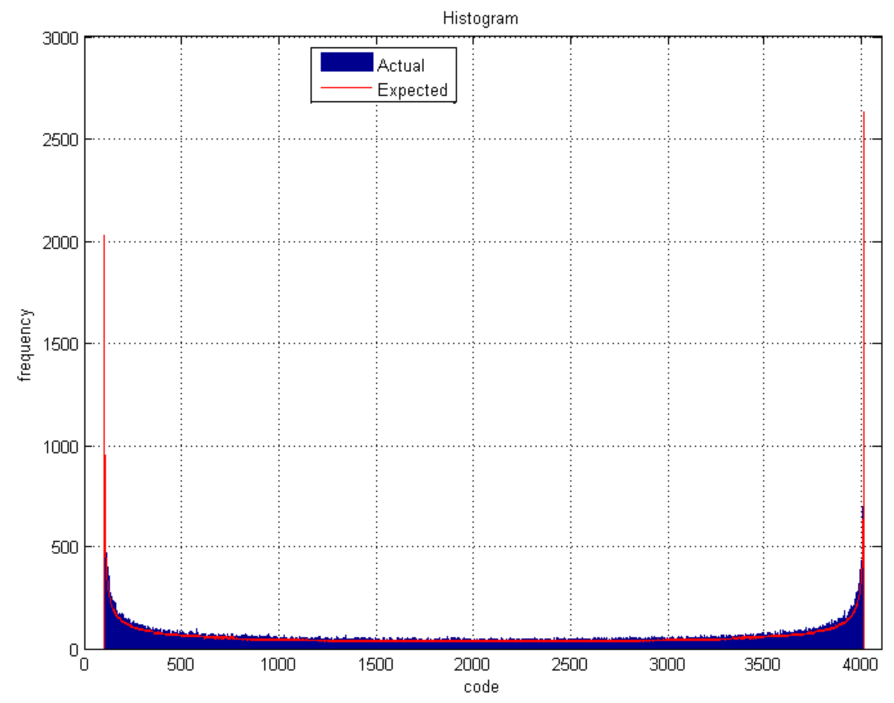
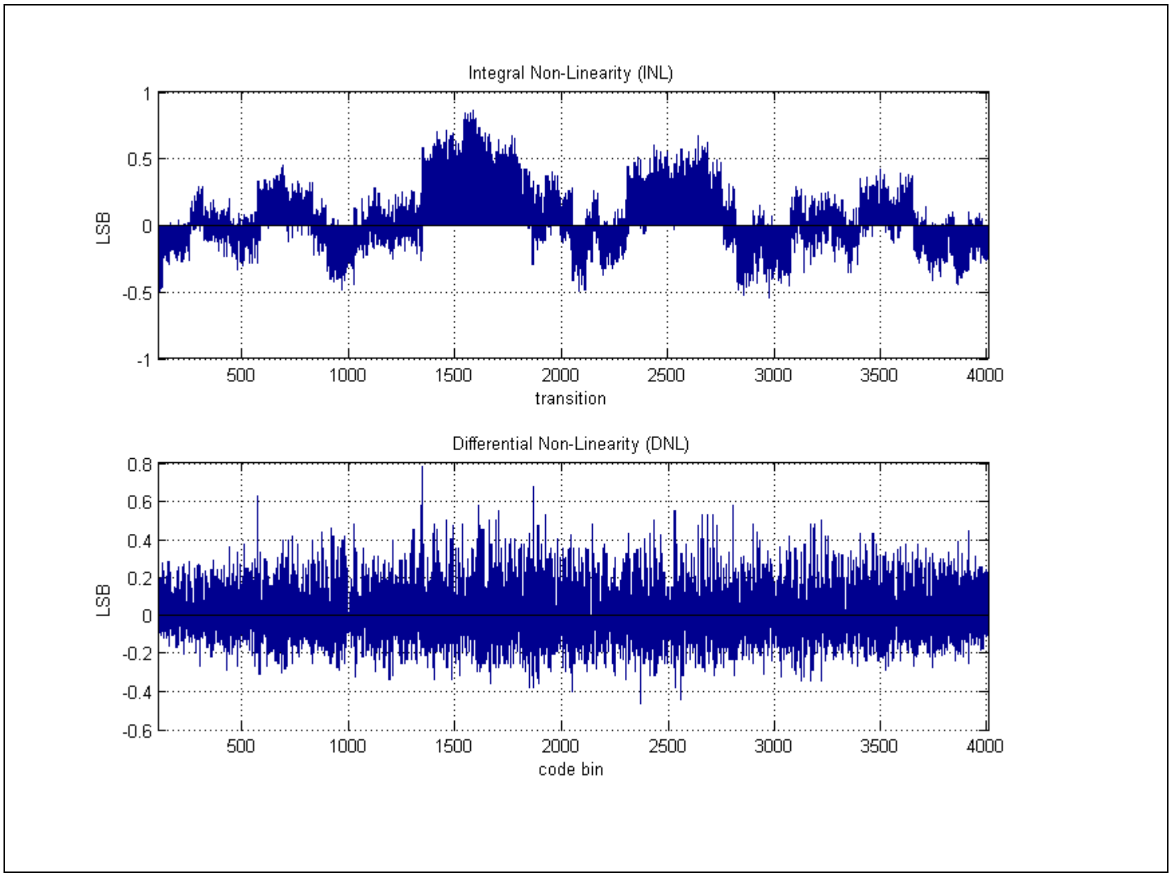


Figure 7. Plots of INL and DNL from ADC Test of Sample Number 5 Using GPIO



Finally, results of ADC testing on five random samples configured for 12 bit resolution using both methods are summarized in Table 1.

Table 1. Mean Values of Performance Measures Using GPIO and SPI Methods

Measure	GPIO Method			SPI Method			Units
	3.3V	2.5V	2.0V	3.3V	2.5V	2.0V	
SNR	68.0682	67.1561	66.0078	67.7297	67.7180	65.8947	dB
THD	-74.0357	-74.0427	-74.0042	-74.0779	-74.2881	-74.0355	dB
SFDR	76.1979	76.4277	76.3603	76.2323	76.6820	76.4688	dB
SINAD	66.8743	66.0903	65.1298	66.6991	66.6089	65.0465	dB
ENOB	10.8151	10.6849	10.5253	10.7860	10.7710	10.5115	Bits
INL	0.7425	0.8187	0.7824	1.0705	0.9393	0.7894	LSB
DNL	1.3746	1.4234	1.5505	1.7786	1.6051	1.4840	LSB

§

5 Discussion

In comparing the GPIO and SPI methods, it can be seen that the GPIO method yielded slightly better results. Most of the difference appears to be attributable to SNR, with THD and SFDR nearly the same. This could be explained by the shortened sleep interval when SPI is used. In comparing the test results with expectations, ENOB is lower than expected. Referring to the Agilent* 30 MHz Function/Arbitrary Waveform Generators Data Sheet [2], the specified THD of the function generator is less than -68dB (0.04%). This means that the test results in this paper might be limited by the THD of the function generator used.

§

6 Conclusion

Testing the Intel® Quark™ microcontroller D1000's SAR ADC in two functional modes found the ADC performance to be exceptional for its class of microcontroller, most of which generally contain10-bit ADCs. Furthermore, test results should show improved performance if a more pure test tone source is used. This leads to a conclusion that users can achieve this high level of performance if they use an appropriate antialiasing filter on their circuit board, AC isolate the AVDD power supply from the IOVDD and PVDD supplies, and write their software in a way that puts the chip into ADC sleep mode prior to the ADC's sample interval.

§

7 References

[1] IEEE Std 1241™-2010, “IEEE Standard for Terminology and Test Methods for Analog-to-Digital Converters,” January 2011.

[2] “Agilent 30 MHz Function/Arbitrary Waveform Generators Data Sheet,” Agilent, June 2011.

§