AD9410 Prototype Evaluation

Grant Hampson

May 13, 2002

Introduction

This document describes the evaluation of a 200MHz analogue-to-digital converter acquisition evaluation board. The evaluation board contains an AD9410 10-bit ADC [1], which was purchased from Analog Devices for $200.

The evaluation board is interfaced via an Altera APEX FPGA to a custom capture card [2]. The APEX FPGA currently contains minimal logic for synchronizing the two data streams from the AD9410. (In future work the APEX FPGA will contain the Digital IF processor.) Figure 1 illustrates the AD9410 evaluation board and interface card (with the APEX FPGA.) The custom capture card is not shown.

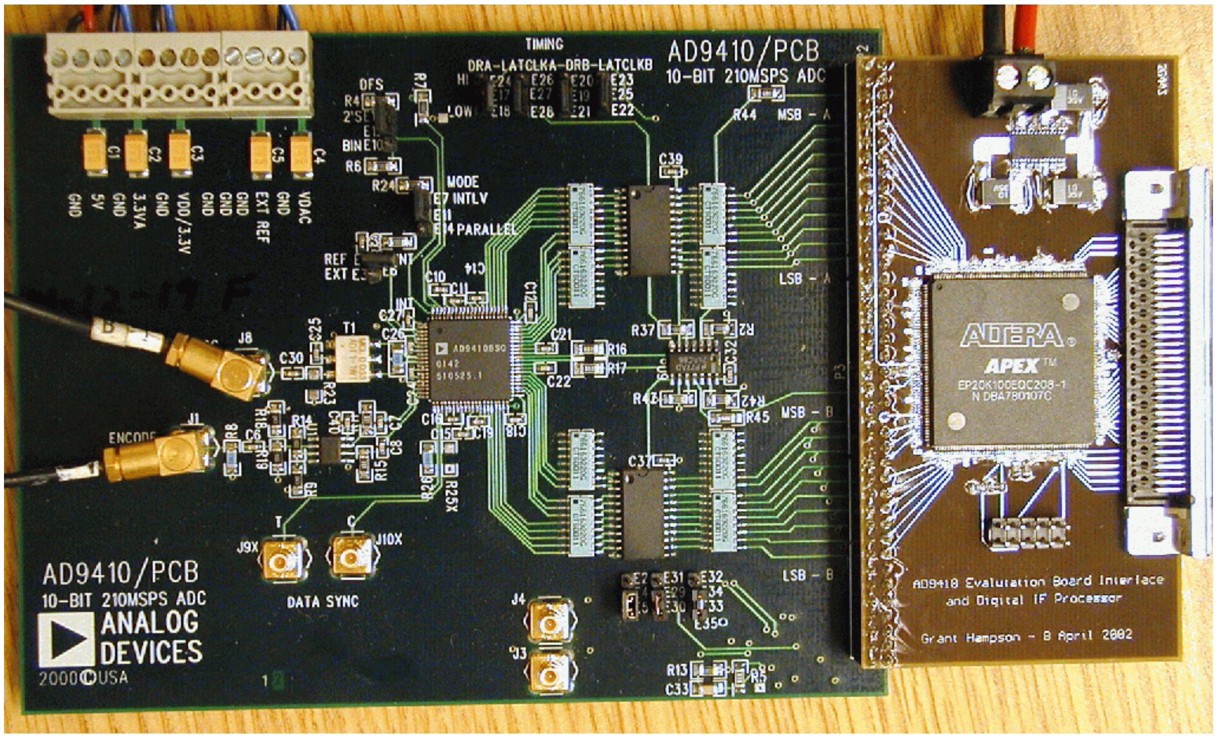


Figure 1: AD9410 evaluation board and APEX FPGA interface card. The APEX FPGA contains logic to synchronize the two data outputs to a single clock. The capture card plugs into the output of the APEX FPGA and data is captured into the PC using a PCI-DIO-32HS digital acquisition card.

1 AD9410 Evaluation Setup

The AD9410 evaluation setup is similar to that of the AD9054 evaluation board [3]. The output of the AD9410 is demultiplexed into two 100MSPS 10-bit buses since it is difficult to obtain a 200MSPS digital bandwidth across a single bus. These output buses are also skewed by half a clock cycle so as to distribute load on the system. The resulting outputs of the AD9410 evaluation board are two 10-bit buses, as well as two 100MHz clocks.

This data is fed into a custom board containing an APEX FPGA. For the moment though this FPGA simply synchronizes the two data streams and passes data unprocessed. This board will contain the Digital IF processor in future developments [4].

The sampled data is captured into the custom capture board. This data is then transferred from the capture card to the PC via PCI-DIO-32HS digital acquisition card.

The LAB Windows CVI interface is very similar to that of the AD9054 evaluation [3]. However, there is one function which changes; the function which reads data from the capture card. Refer to Appendix A for this functions source code.

2 Measurements

The following three sections illustrate the results from three different stimuli. The first is narrow band noise, the second low frequency sinusoids from an arbitrary waveform generator, and finally sinusoids generated by a network analyzer and sampled in the second Nyquist zone.

2.1 Noise Input

The first input to the system was a noise input generated by the ARB. The noise bandwidth of the ARB is relatively small compared to the sampling rate of the system, so a 10.7MHz low pass filter was used on the input to limit the output BW. The result of 100 integrations of length 32k FFTs is illustrated in Figure 2. The frequency response at 10.7MHz is -6dB and rolls off sharply to almost 55dB. Two tones exist in the result, one at 50MHz and another at 98.75MHz, which are likely to be the result of clock leakage.

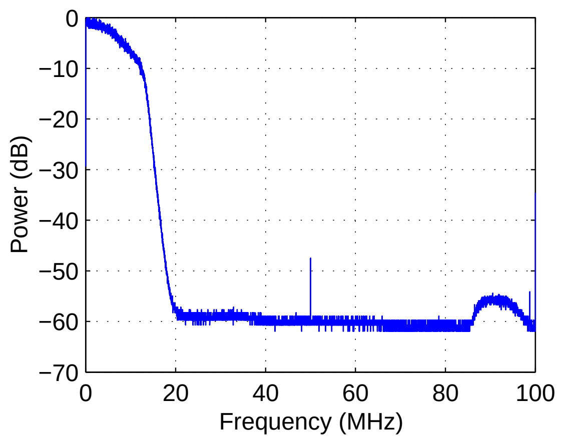


Figure 2: Noise spectrum from the ARB with a 10.7MHz low pass filter. 100 length 32k FFTs are integrated for this result. Clock leakage at 50 and 98.75MHz occurs.

2.2 Low Frequency (1-30MHz) Sinusoidal Inputs

The next tests use an arbitrary function generator to generate low frequency sinusoids in the range of 1 to 30MHz. The amplitude is set 0.5dB below full scale (7dBm or $1.5V_{p-p}$.)

The first sinusoidal test to be performed was to determine how accurate the frequency of the sampling frequency and ARB are. To do this the ARB was stepped in increments of 1MHz between 1.141592 and 30.141592MHz and a measurement taken. Figure 3(a) illustrates the difference between the ideal and measured frequencies. Using a 32k FFT, the frequency resolution is 6.1kHz. All of the results are within this range. It is possible to sample up to 256k samples (frequency resolution of 762Hz) but this was felt unnecessary.

The same data collected for the frequency tests was also used to determine the approximate Signal-to-Noise Ratio (SNR) of the AD9410. An example spectrum (30MHz input sinusoid) is shown in Figure 3(b). Several pieces of information can be extracted from this spectrum using the Matlab script listed in Appendix B.

First the fundamental frequency power is noted (which is the RMS value of "B" FFT-bins.) These bins are zeroed, as well as the DC FFT-bins. Secondly the first six harmonics are removed. The remaining *noise* is summed [5, 6]. The ratio of the fundamental power and total noise power is taken to be the SNR.

Figure 3(c) illustrates the fundamental power for frequencies between 1 and 30MHz. The anti-aliasing filter has a specification of 1dB ripple in the pass band.

Figure 3(d) illustrates the power level of the first six harmonics which ranges between 63 and 42dB below the fundamental. The SNR is illustrated in Figure 3(f) and ranges from 52 to 41dB. The AD9410 specifications state that a SNR of 53dB is possible. The arbitrary waveform generator is not a great device for testing the resolution of this ADC. Note the way the noise floor changes in Figure 3(b) suggesting the generator produces significant output noise above the ADC noise floor.

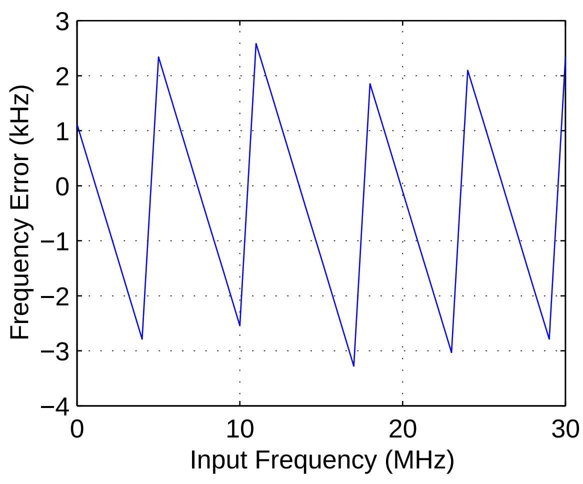
2.3 High Frequency Sinusoidal Inputs

The next test performed on the AD9410 was conducted using an Agilent 8722ET network analyzer which has a much greater frequency range. The network analyzer output power is limited to 0dBm and consequently an amplifier (a mini-circuits ZFL-500HLN) was used to boost the signal to 7dBm. Secondly, the anti-aliasing filter used is a band-pass filter with cut-off frequencies of 110 and 190MHz.

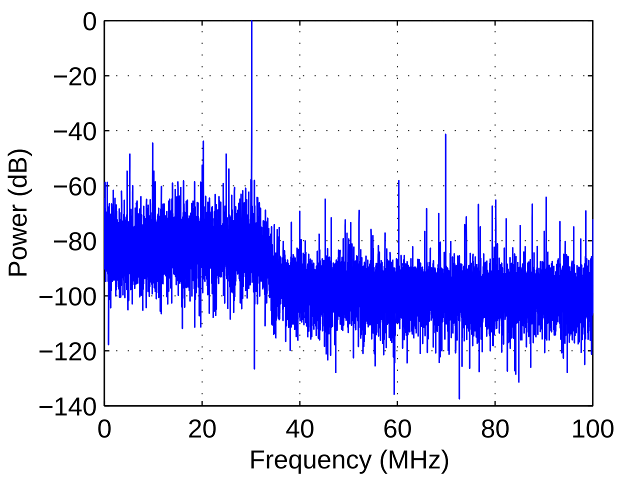
An input frequency ranging from 105.141592 to 195.141592MHz in 5MHz increments was input into the AD9410. The resulting data was analyzed in the same way. The results of this analysis are shown in Figure 4.

Note firstly that the network analyzer has a significantly lower noise floor than the arbitrary function generator. Consequently, the SNR measurements are likely to be more accurate. The fundamental power plot (c) illustrates the anti-aliasing filter function. A consequence of using a network analyzer is that a relatively strong harmonic (at double the frequency) could still pass through the anti-aliasing filter. Its power is noted as the largest harmonic.

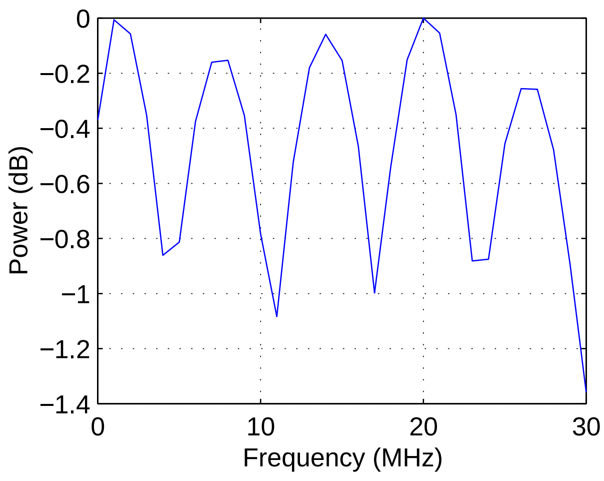
When disregarding this out-of-band harmonic in SNR calculations results in a measured SNR of approximately 50dB - which is very close to the specification of 53dB.



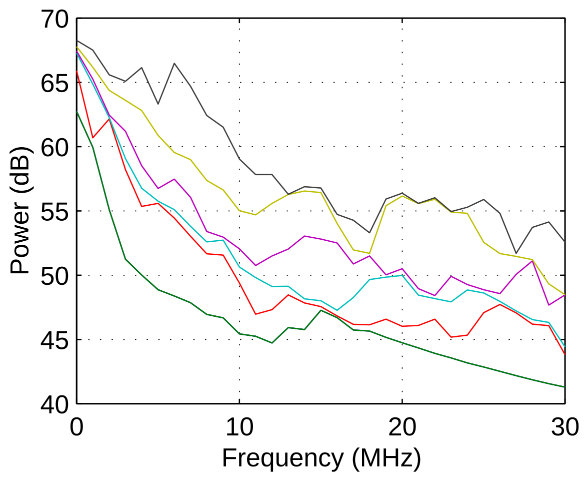
(a) Frequency Error



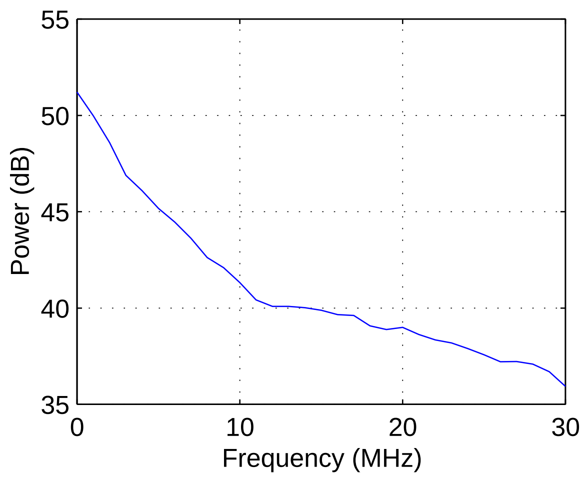
(b) 30.141592MHz Sinusoid



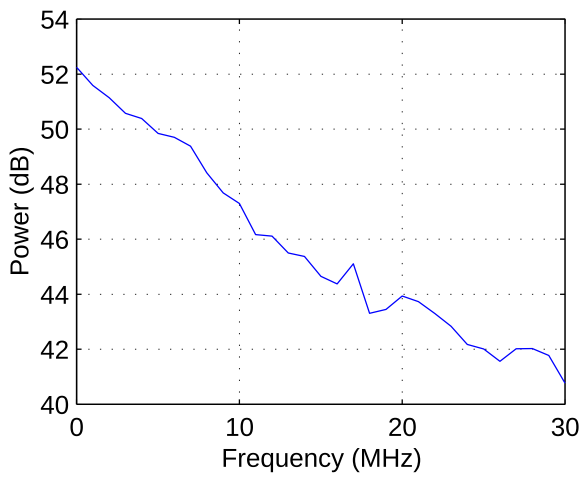
(c) Fundamental Power



(d) First 6 Harmonics

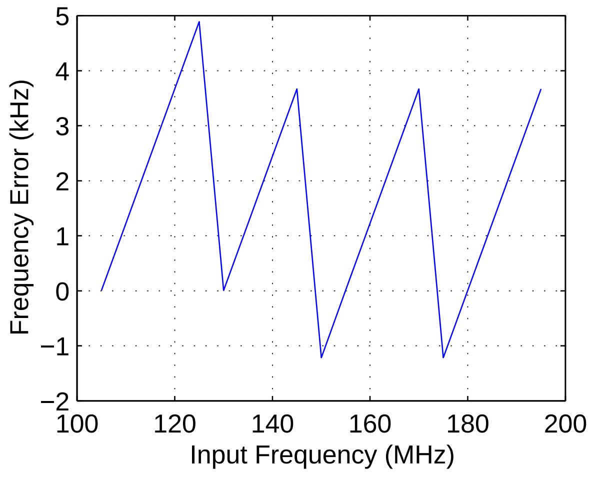


(e) SINAD (SNR with harmonics)

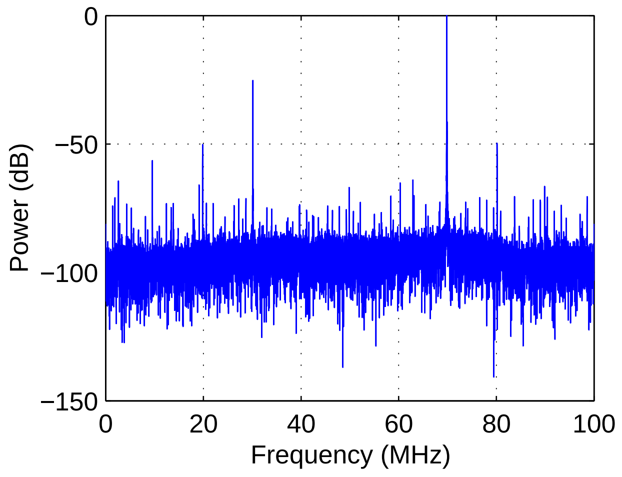


(f) Signal-to-Noise Ratio (without harmonics)

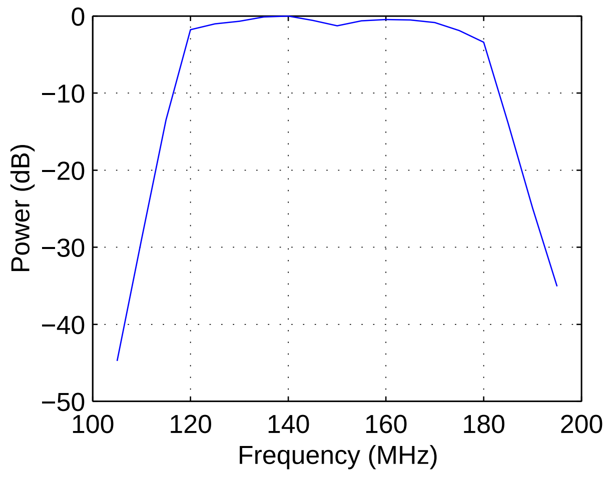
Figure 3: Results from the ARB for sinusoids between 1 and 30MHz, using a 50MHz low pass anti-alias filter.



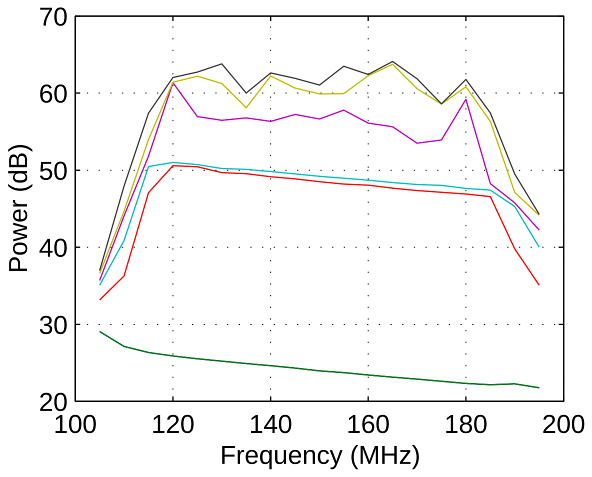
(a) Frequency Error



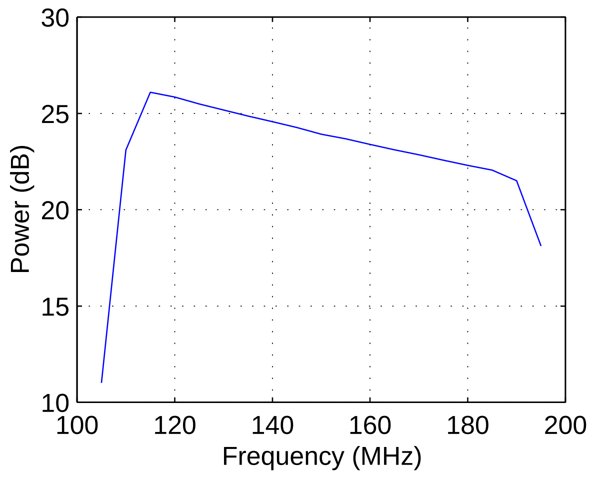
(b) 130.141592MHz Sinusoid



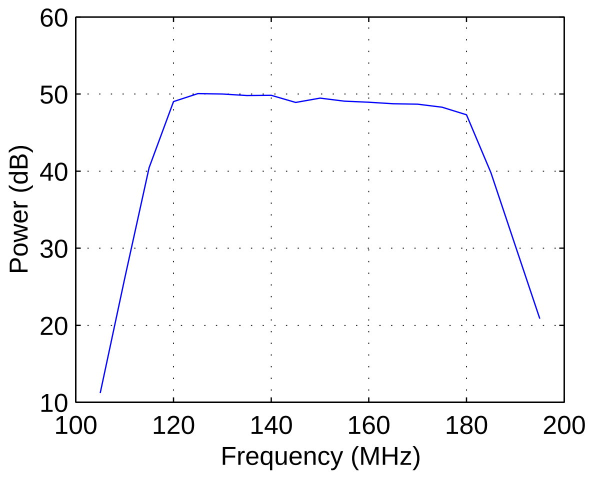
(c) Fundamental Power



(d) First 6 Harmonics



(e) SINAD (SNR with harmonics)



(f) Signal-to-Noise Ratio (without harmonics)

Figure 4: Results from the Network Analyzer CW sinusoids between 105 and 195MHz, using a 110 to 190MHz band-pass anti-alias filter.

3 Summary and Conclusions

This document has attempted to evaluate the AD9410 10-bit 200MSPS analog to digital converter evaluation board. In order to evaluate the AD9410 ADC several other custom boards were built. These additional boards are described in other documents.

The resulting system could be clocked at 200MSPS and many measurements were taken, both at low and high frequencies. A Matlab script was designed to analyze the recorded data. The measured SNR's were in close agreement to the ADC specifications, but the measurements revealed the need for test sources with low noise and high gain (a difficult combination.)

Personally, a greater appreciation of anti-aliasing filters, measurement equipment and ADCs has been achieved.

References

[1] *10-bit, 210MSPS A/D Converter, AD9410*, Analog Devices Corporation, October 2000. http://www.analog.com/productSelection/pdf/AD9410_0.pdf.

[2] G. A. Hampson, "A 256@32-bit Capture Card for the IIP Radiometer," May 10 2002. http://esl.eng.ohio-state.edu/rfse/iip/fifocapture.pdf.

[3] G. A. Hampson, "AD9054 Prototype Evaluation," March 5 2002. http://esl.eng.ohio-state.edu/rfse/iip/ad9054proto.pdf.

[4] G. A. Hampson, "An FPGA Implementation of the Digital IF Processor," March 7 2002. http://esl.eng.ohio-state.edu/rfse/iip/digitalif.pdf.

[5] *High-Speed ADC FIFO Evaluation Kit*, Analog Devices Corporation, 8 April 2002. http://www.analog.com/techSupport/DesignTools/evaluationBoards/HSC_ADC_Eval-SC_DC_prh.pdf.

[6] *Selecting Mixed-Signal Components for Digital Communications Systems – Part V : Aliases, Images, and Spurs*, Analog Devices Corporation, 31 March 1997. http://www.analog.com/library/analogDialogue/archives/31-3/Selecting.html.

Appendix A: LAB Windows CVI Source Fragment

```
void get_raw_data(int numsamples, int wtf)
{
   int i, raw_handles[8];
   char rawsample, filename[100];
   long nRemaining;
   FILE *filepntr;
   double *real_ptr, *imag_ptr;
   short *short_ptr;

   DIG_Out_Line(PCI_DIO_32HS, 0, 7, 0); // reset FIFO
   Delay(0.001);                        // Reset for 1ms
   DIG_Out_Line(PCI_DIO_32HS, 0, 7, 1); // start capturing

   DIG_Out_Line(PCI_DIO_32HS, 0, 6, 0); // Select Even Sample FIFO
   DIG_Block_In (PCI_DIO_32HS, 1, iBuf, (numsamples/2)+2);
   nRemaining = 1;
   while(nRemaining>0) DIG_Block_Check (PCI_DIO_32HS, 1, &nRemaining);

   real_ptr = &xreal[0];                // point to start of memory block
   for (i=0; i<(numsamples/2); i++)
   {
      *real_ptr++ = ((double)iBuf[i+2])/64.0;  // offset of 2 to avoid FIFO start
      real_ptr++;
   }

   DIG_Out_Line(PCI_DIO_32HS, 0, 6, 1); // Select Odd Sample FIFO
   DIG_Block_In (PCI_DIO_32HS, 1, iBuf, (numsamples/2)+2);
   nRemaining = 1;
   while(nRemaining>0) DIG_Block_Check (PCI_DIO_32HS, 1, &nRemaining);

   real_ptr = &xreal[1];                // point to start of memory block
   for (i=0; i<(numsamples/2); i++)
   {
      *real_ptr++ = ((double)iBuf[i+2])/64.0;  // offset of 2 to avoid FIFO start
      real_ptr++;
   }

   if (wtf)
   {
      sprintf(filename,"rawdata.dat");
      if((filepntr=fopen(filename,"w"))==NULL)
         printf("File could not be opened");
      for(i=0; i<numsamples; i++)
         fprintf(filepntr,"%f\n",xreal[i]);  // write to file
      fclose(filepntr);
   }

   DeleteGraphPlot (hpMain, MAIN_GRAPH_RAW, -1, VAL_IMMEDIATE_DRAW);
   PlotWaveform (hpMain, MAIN_GRAPH_RAW,
      xreal, numsamples, VAL_DOUBLE, 1.0, 0.0, 0.0, 1,
      VAL_THIN_LINE, VAL_NO_POINT, VAL_SOLID, 1, VAL_BLUE);
}
```

Appendix B: Matlab Data Analysis Script `calcsnr.m`

```
% Script to calculate SNR by Grant Hampson May 8, 2002

clear all

N = 32768;             % number of samples captured
sw = blackman(N);      % pre-calculated spectral window

Fs = 200;              % sampling frequency in MHz
f = (0:N/2-1)*Fs/N;    % frequency vector

ft = 0:30;             % name of saved data files
B = 15;                % 2B bins are blanked per fundamental/harmonic

for fi = 1:length(ft)
   X = fft(sw.*load(['rawdata' num2str(ft(fi)) '.dat']));
   X = X(1:N/2);                                % only half the spectrum is used
   X = real(X).*real(X) + imag(X).*imag(X);     % squared spectrum (no sqrt)

   [maxX b] = max(X);                           % find peak in spectrum
   freqfund(fi) = f(b);                         % frequency of fundamental

   fund(fi) = sum(X(b-B:b+B)/maxX);             % sum of fundamental energy
   X(b-B:b+B) = 0;                              % and remove it

   X(1:B) =0;                                   % remove DC component

   sinad(fi) = 10*log10(fund(fi)/sum(X/maxX)); % SNR with harmonics

   for h = 1:6                                  % remove first six harmonics
      [a b] = max(X);                           % find largest remaining harmonic
      X(b-B:b+B) = 0;                           % remove harmonic
      firstharm(fi,h) = -10*log10(a/maxX);
   end

   snrwh(fi) = 10*log10(fund(fi)/sum(X/maxX)); % SNR without harmonics

   fprintf('F=%f MHz, P=%3.0fdB, FH=%3.0fdB, SINAD=%3.0f dB, SNR=%3.0fdB\n', ...
      freqfund(fi), 10*log10(fund(fi)), firstharm(fi,1), sinad(fi), snrwh(fi))
end
```