DECUS-8-250

This demonstration is based on DECUS-8-250 which was written for capturing signals on a AX08 and calculating and displaying the frequency spectrum using an FFT. DECUS was the Digital Equipment Computer Users' Society where users submitted useful software and it was distributed at nominal cost. The software could be distributed as binary and/or source code. No restrictions were put on use. DECUS-8-250 was distributed as source code.

I first noticed this software in an eBay auction that I saw after it closed. I looked around and found two of the five files were available online. I asked around and nobody responded that they had the missing files or won the auction. There were some other FFTs in the DECUS library and I used 8-143 to make something that operated similar to 8-250 for a previous demo.

Bitsavers put up a large dump of DEC files. While I was going through it, I noticed it had more of DECUS-8-250. With the previous files that left only one file missing, and that was the definition of the data storage so I should be able to recreate it. I did so and got it to assemble but results were garbage. After figuring out correct values of some constants and fixing other errors it started to generate a spectrum that looked somewhat like expected but lots of bad values.

The first problem I found was that the first tape which had a table of sin values was truncated. There were likely two separate reads of that tape so it may be the original DECUS distribution tape was bad. In recreating the proper table I found errors in the original table. One value was dropped so all following samples were off by one. Also there were some typos in other values in the table.

Next I found that the multiply routine had errors. The multiply routine takes 2 12 bit values and returns (v1 * v2) / 2048. N * 0 didn't always give 0. The worst case was -1 times 0 giving -2048 as the result. The error in multiplying positive numbers was +-1. For negative numbers, many were off by 4 with a -2047 * 2047 off by 2044. I fixed these errors in the original code and spectrum looked pretty good. Still -2048 times -2048 or -2047 give the wrong answer. -2048 * -2048 can't be represented in 12 bits. The other should be but the algorithm gives -2048 instead of 2047. I also added support for the hardware multiply my machine has. 1024 point FFT 3 seconds with hardware multiply, 6.8 without.

Program also had a couple "features" which I fixed. In bar mode it would frequently draw the last point at the top of the screen instead of the bottom and horizontal shift would allow shifting such that it displayed data past end of spectrum. Fixed that but left in displaying the mirrored spectrum from transform of real data which results in two copies of the N/2 point spectrum. It also applied the Hanning window twice. This may have been to suppress some of the spikes from the multiply bug. I turned the second one off.

N bit digital signals have theoretical maximum SNR of 6.02 * N + 1.76 dB. People normally just say 6 dB per bit. A/D noise will reduce that. The 9 bit A/D converter of the AX08 has about 54 dB SNR. The calculations are done in 12 bit integers using block floating point. Code implements block floating point. After each stage of calculation the results are divided by 2 if any value is outside -1023 to 1023 to prevent overflow so calculations limit dynamic range to 66 dB. FFT gives complex values. When the original code squared and summed the complex signal to show real linear power that limited the signal to 33 dB dynamic range so less than the A/D. I added an approximation (¼ smaller plus larger) to do a magnitude calculation to retain the input signal range in the frequency output. This gives about maximum of 12% error in magnitude or -18.7 dB in power. dB is 10*log10(power) or 20*log10(voltage) since power = voltage^2/resistance.

Like always I spent a lot more time than planned on getting this code working so didn't get time to move and improve my code to plot the results on a pen plotter. It will be back some other year plotting the input signal and a spectrogram with proper axis labels. Also want to add dB amplitude and log frequency display.

Later I found another bug/feature. To make it run faster the first and second FFT pass were optimized. The first pass the imaginary part is always zero and the first and second pass only require multiplying by +- 1 so can be implemented with only addition and subtraction. The rest of the passes use multiply. The original code had a *377 before PASS2 in decus-250-3.pa which made PASS2 overwrite most of PASS1. When I fixed it that made results wrong since during the initial debugging I had set the number of passes for the later code to compensate. I was suspicious about the number I used but didn’t know why it needed that value at the time. When I set that back to the proper value it then operated properly. You could remove the PASS1 and PASS2 code to save memory with some performance hit.

Original files from https://svn.so-much-stuff.com/svn/trunk/pdp8/src/decus/8-250/ and http://www.bitsavers.org/bits/DEC/mixed_media/