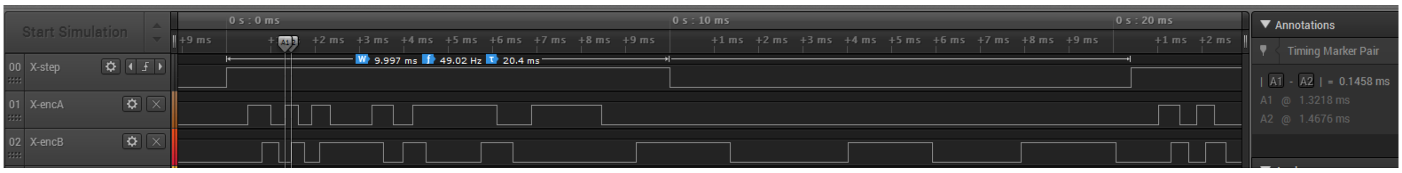
Today I attached a logic analyser to the electronics to monitor the step and encoder operation, mainly to look at my jerky X axis, but also to get a feel for what was happening overall. Here are some of my findings:

1. Motor Start

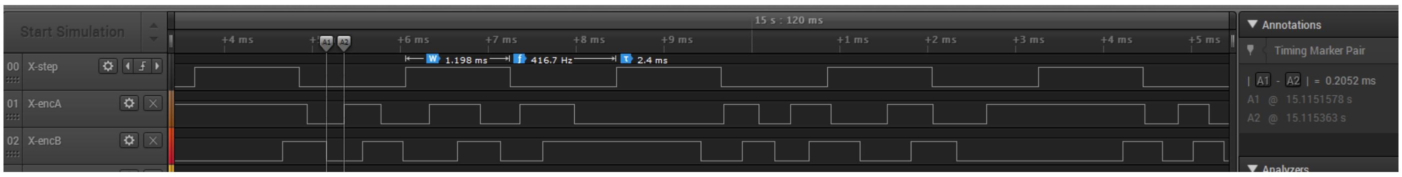


Above is the first step on the X axis, the step pulse width is 10ms, a frequency of 50Hz, which is correct for a minimum speed of 50 steps/s.

You can see here that there are 11 encoder edges in the correct direction, then we get some reverse ones, and then the B oscillates before the next step pulse. Thinking about it, that's what you might expect, the gantry isn't 'stiff' so what we're really doing is making it vibrate! Only when we're moving at speed do the encoder oscillations stop.

The above is not a problem, the encoder waveform is correct and should be decoded ok, except that the shortest time between two edges in the above waveform is <150us (between the A1/A2 markers).

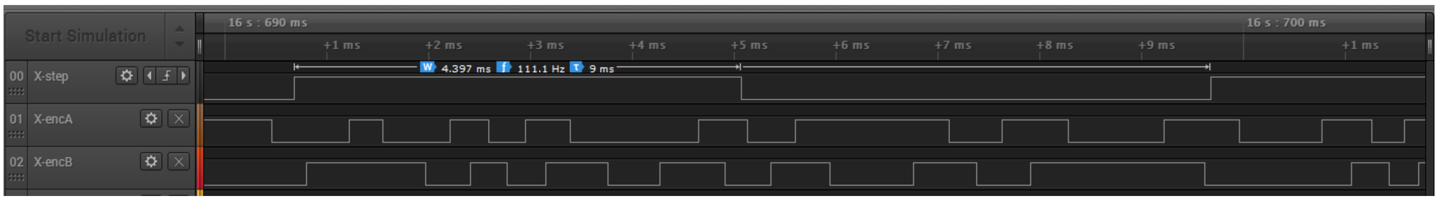
2. Motor Running at 500 steps/s



When running at 500 steps/s the operation is noticeably smoother. The above shows a frequency of 416.7Hz, this is partly due to the granularity of the stepper frequency generation (which is poor, should be using DDS), and partly due to the ISR missing interrupts due to controlling 2 or 3 steppers simultaneously.

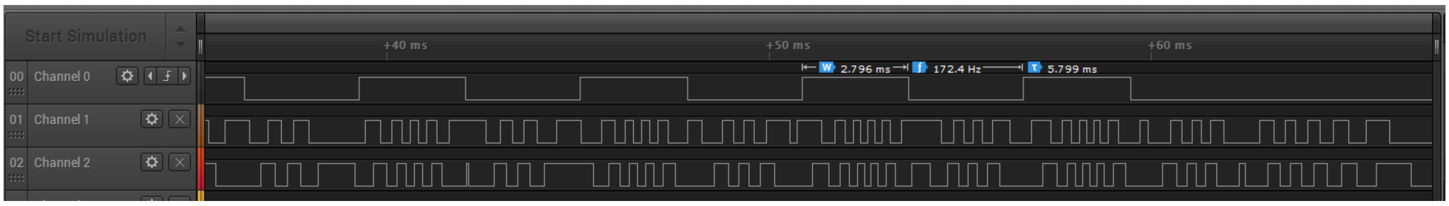
Worth noting is that the encoder is running consistently in the correct direction. The time between encoder edges is 200us, so is actually easier to monitor at this speed. Also note that the encoder output is very 'uneven', although there is consistent timing between edges.

3. Slowing down



The above capture is in the middle of slowing down, at about 100 steps/s. You can see that the vibration has returned and the encoder is again giving both forward and reverse counts.

4. Premature stop



The above capture shows the software 'stopping' the step pulses during deceleration. The frequency should slow down to 50Hz, but here it's still 170Hz and just stops. There is nothing in the encoder waveform to suggest any problem, other than the previously mentioned forward/reverse counts (but all valid waveforms). The very short pulse on 'Channel 2' (X-encB) about 8 pulses in is about 50us, and is perfectly valid.

5. Encoder timing

The shortest time between encoder edges I've seen is about 120us, although there are some occasional shorter pulses at low speed. Interestingly, the shortest time between encoder edges occurs at low speed, so slowing down the motors isn't going to help with monitoring the encoders.

6. Step frequency

The generation of the step frequency occurs in discrete 'steps', it is also highly dependent on whether there are other axes in motion at the same time. I'm guessing this is because in the Arduino code you are trying to do everything in one 200us interrupt, so when more than one axis is operational the ISR is likely taking longer than the 200us, so your timing counter is incorrect. I can suggest more optimal ways of organising this code, but you might already be on to it.

Obviously, sampling the encoder inputs at 200us is insufficient based on the results I've just obtained.

The step frequency would be better generated using DDS (direct digital synthesis), that way you could change frequency smoothly without the noticeable discrete steps.