

This is my first attempt at writing a stepper motor driver for the Propellor 2.

The driver itself is written in propellor assembly. There are no tricky code sections and the code is well commented so it should be easy to follow. Minimal effort has been applied to optimize the code primarily because this is my first attempt at prop 2 assembly. The code makes extensive use of the cordic math solver.

The driver produces a constant acceleration where the delay between each step is calculated in real time from the kinematic equations.

Velocity (at time t) = Initial Velocity + acceleration * time (t) velocity in steps/sec, Vo = initial velocity

$\Delta X = (V+V_o)/(2 * a)$ ΔX (change in position) in steps

$V = \text{sqrt}(V_o^2 + (2 * a * \Delta X))$

The time delay between steps = Clock frequency / velocity

With a clock frequency of 200,000,000 and a velocity of 20,000 steps/ sec the delay becomes 10, 000 clock ticks. More than enough time for the math.

I am using this with a 5 phase stepper motor.

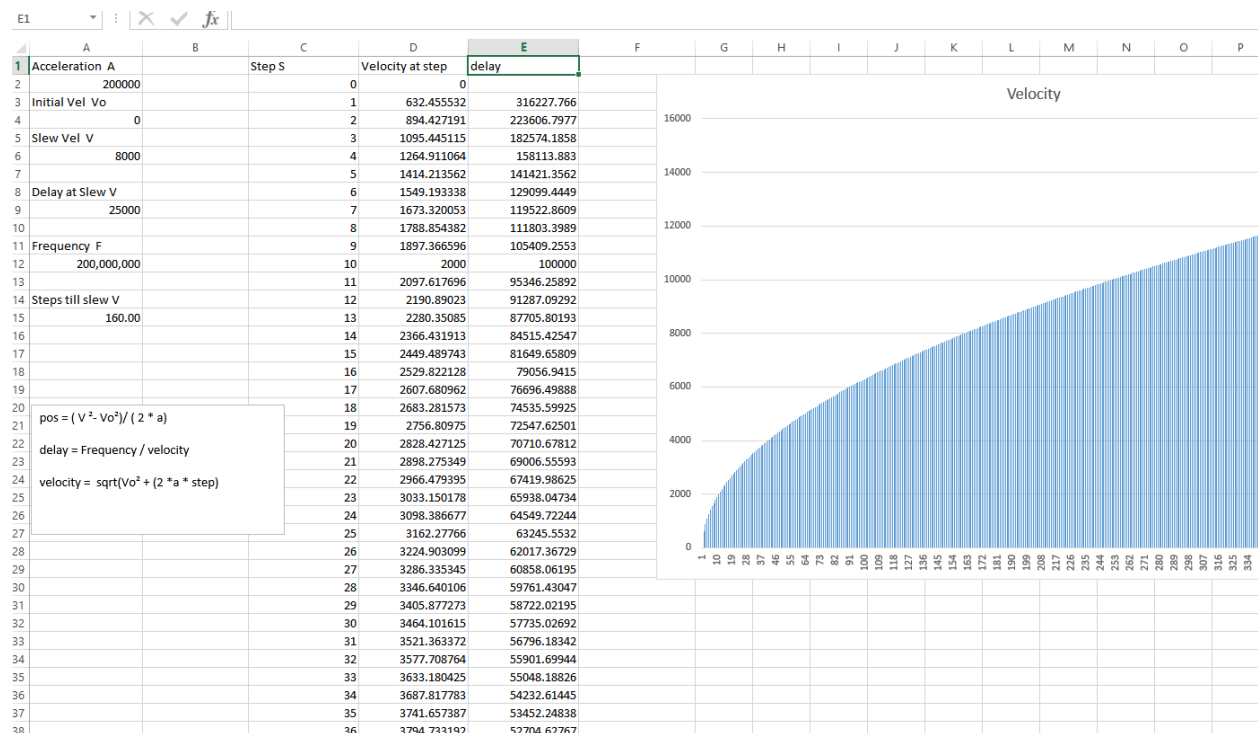
.72 degrees / step

360 degrees / .72 = 500 steps / revolution

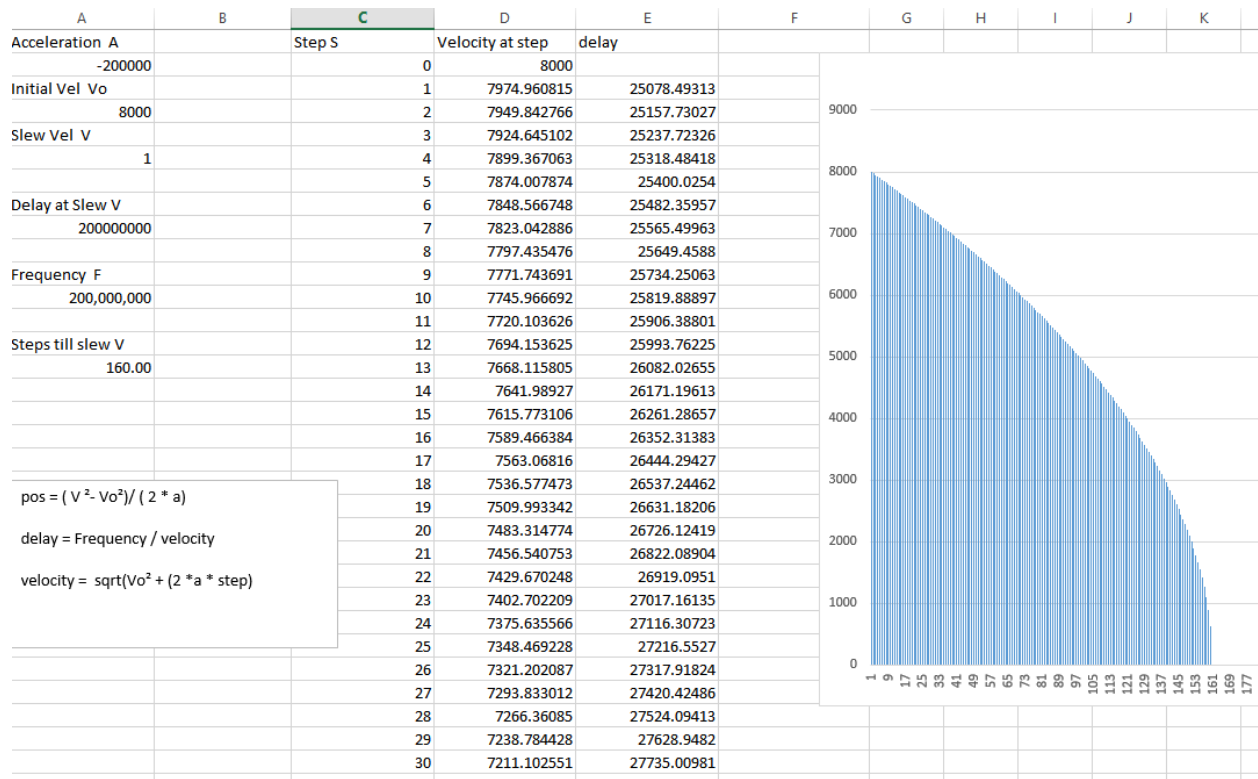
20,000 steps/sec divided by 500 steps/ revolution = 40 revolutions / sec

40 * 60 = 2400 rpm

There is an included excel spreadsheet that calculates the delay and instantaneous velocity at every step.



Deceleration



The included flowchart was drawn using a program called draw.io – diagrams.net - It's free to use and easy to use and for me anyway it makes it a lot easier to visualize the program flow.

There is a demo program that lets you input steps (plus and minus) and speed. You can run the demo without a motor connected.

In the demo program the following two lines calculate the acceleration and deceleration. You can modify them to suit your purposes.

$a := (sp * sp) / (4 * \sqrt{sp})$ 'calculate values for acceleration and deceleration

$d := (3 * a) / 2$ '*1.5 most systems can stop faster than they start (friction, etc.)

To high a value for a or d will cause you to lose steps. To small a value will lead to long delays starting and stopping and the possibility you will never reach the desired speed.

In normal use the speed of the motor accelerates to the desired speed, maintains that speed for some time and then decelerates to a stop.

Run the demo program.

Press F12 to start the parallax serial terminal. Check Echo On.

Press Enter to start.

You will be prompted for steps and speed and the the motor will run.

If you press zero 0 while the motor is running it will decelerate to a stop.

The driver sets steps = 0 to indicate it is ready for a new command.

In the demo program the start/stop pin is held high by the following line.

`pinhigh(spin)` 'start pin must be high for motor to run - could be connected to a switch instead

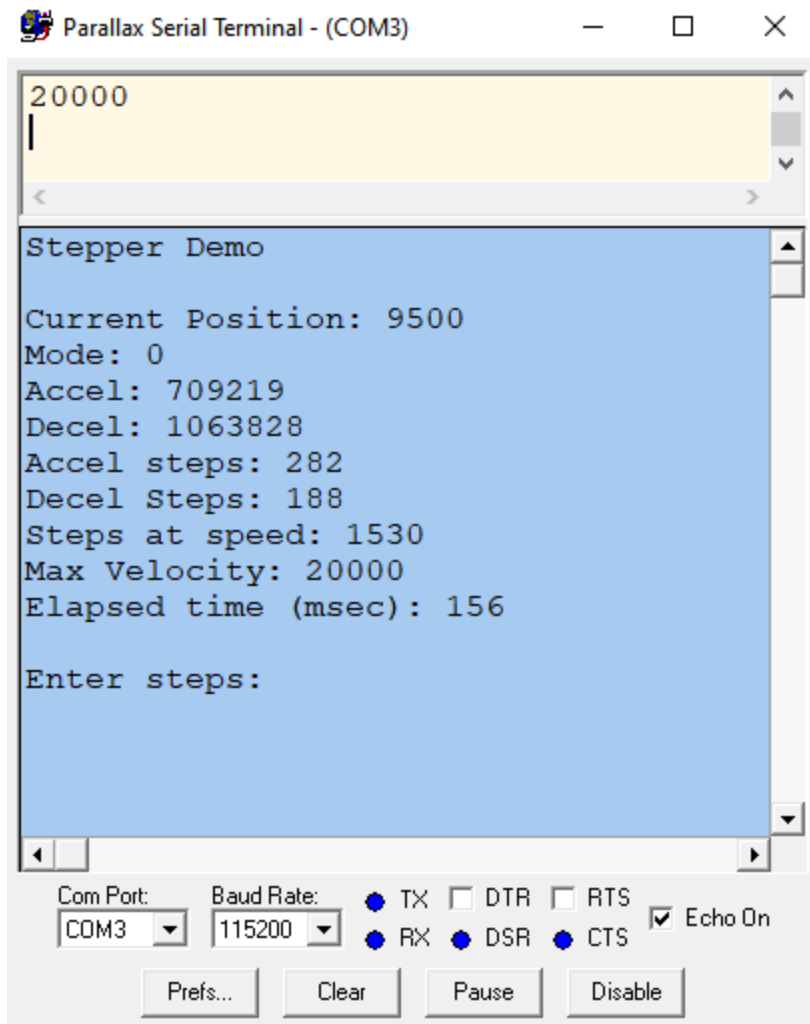
you could comment out that line and connect the pin to a switch to start or stop the motion.

Comment out the following lines as well

```
if char == "0"      'set the start/stop pin to 0 (stop)
    pinlow(spin)
    waitms(10)
    pinhigh(spin)      'set the start/stop pin to 1 (start) for next time
```

Best practice would be to connect the switch to the pin through a resistor to prevent pulling the pin to ground while the program is driving it high.

I have also moved the 3 pins to a different group of outputs after learning that a shorted pin in outputs 24-31 could take out the power supply for the oscillator section thus bricking the p2



Enjoy...