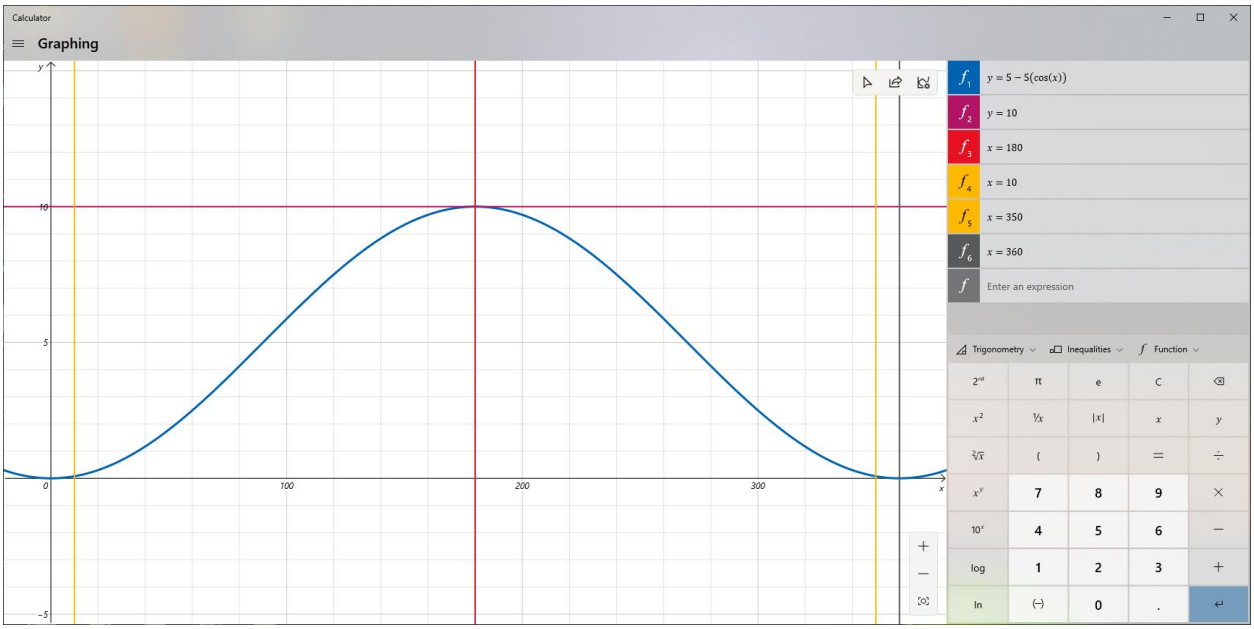
This is my second attempt at writing a stepper motor driver for the Propeller 2.

The driver itself is written in propeller assembly. There are no tricky code sections and the code is well commented so it should be easy to follow. The code makes extensive use of the cordic math solver.

It occurred to me that the desired s- curve motion profile resembles a sine wave. This driver attempts to utilize that fact.



I am mapping the number of steps (say 0- 5000) to degrees (0-360), calculating the cosine and scaling it by the desired velocity. (qrotate in the cordic solver)

The time delay between steps = Clock frequency / velocity at step

With a clock frequency of 200,000,000 and a velocity of 50,000 steps/ sec the delay becomes 4000 clock ticks. More than enough time for the math.

The number of steps represents the acceleration. So what I am actually doing is mapping a smaller number of steps (say 500) to the (0-360) degree range then adding the excess steps (ex. 5000-500=4500) at the top of the curve (the purple line) where the acceleration is zero.

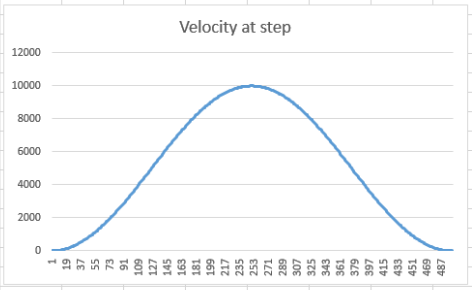
In practice the areas at the bottom of the curve where the acceleration is close to zero contribute little to the smoothness of the profile but they do add considerable time to the motion. For that reason I actually map the steps to the range (0 + offset) to (360-offset), represented by the yellow lines above.

The mapping is done by the following formulas.
Steps = oldrange = (oldmax-oldmin)= (steps – 0)
Degrees = newrange = (newmax – newmin)= (0 + offset) to (360-offset)
Newvalue(in degrees) = (oldvalue – oldmin)/(newrange/oldrange) + newmin
Nv(degrees) = step number*(360-2*offset)/(steps to move) + offset
Velocity at step = (cos(nv)*vel/2) + vel/2
Delay(at step) = clock frequency / velocity

You can send a second move command before the first one is completed. If the direction and speed are the same, the additional steps will add to the xsteps and there won't be an intervening decelerate and accelerate between the moves. I use this as a jog mode where I can hold a key down and get steady motion while the key is depressed.

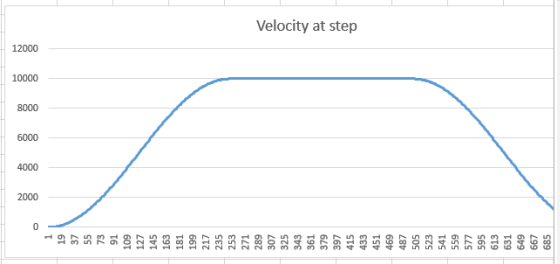
There is an included excel spreadsheet that calculates the delay and instantaneous velocity at every step.

Frequency	step	pos converted to new range	Velocity at step	delay	elapsed time
200,000,000	1	0	0	1000000000	5.00
	2	1.44	1.579053584	112419323	5.56
vel	3	2.16	3.552636797	53295859	5.83
10000	4	2.88	6.31521697	30697366	5.98
	5	3.6	9.866357859	19868159	6.08
oldRange = oldMax - oldMin	6	4.32	14.2054987	13883587	6.15
steps	7	5.04	19.33195428	10239631	6.20
500	8	5.76	25.24491509	7860117	6.24
	9	6.48	31.9434474	6222108	6.27
	10	7.2	39.42649343	5047128	6.30
	11	7.92	47.69287152	4175987	6.32
	12	8.64	56.74127631	3512391	6.34
	13	9.36	66.57027896	2995345	6.35
newRange =360-0	14	10.08	77.17832735	2584703	6.36
360	15	10.8	88.56374636	2253172	6.38
	16	11.52	100.7247381	1981675	6.39
	17	12.24	113.6593822	1756553	6.39
	18	12.96	127.3656361	1567820	6.40
	19	13.68	141.8413354	1408041	6.41



With extra steps inserted at top of curve

Frequency	step	pos converted to new range	Velocity at step	delay	elapsed time
200,000,000	1	0	0	1000000000	5.00
	2	1.44	1.579053584	112419323	5.56
vel	3	2.16	3.552636797	53295859	5.83
10000	4	2.88	6.31521697	30697366	5.98
	5	3.6	9.866357859	19868159	6.08
oldRange = oldMax - oldMin	6	4.32	14.2054987	13883587	6.15
steps	7	5.04	19.33195428	10239631	6.20
500	8	5.76	25.24491509	7860117	6.24
speed/2	9	6.48	31.9434474	6222108	6.27
5000	10	7.2	39.42649343	5047128	6.30
	11	7.92	47.69287152	4175987	6.32
	12	8.64	56.74127631	3512391	6.34
	13	9.36	66.57027896	2995345	6.35
newRange =360-0	14	10.08	77.17832735	2584703	6.36
360	15	10.8	88.56374636	2253172	6.38
	16	11.52	100.7247381	1981675	6.39
	17	12.24	113.6593822	1756553	6.39
	18	12.96	127.3656361	1567820	6.40
	19	13.68	141.8413354	1408041	6.41
	20	14.4	157.0841944	1271584	6.42



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 * 60sec = 2400 rpm

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 driver program the following lines calculate the acceleration (number of steps) and offset. You can modify them to suit you purposes.

```
asteps := sp/10 +2          'effectively sets max acceleration
t := abs(st/2)
offset := 40
if asteps > t
  asteps := t               'limits accel/ decel steps to 1/2 the total steps
ss := abs(st) - (asteps*2)
if ss == 0
  sp := 10*(asteps -2)      'limit acceleration for short moves
```

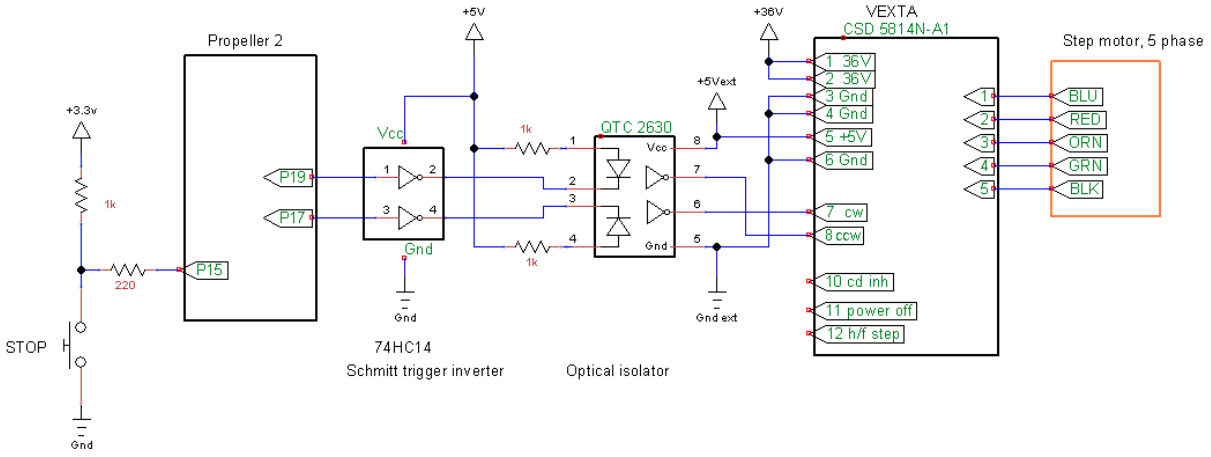
Sp = desired speed (maximum speed)

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.

Pulling the stop pin low causes the driver to decelerate to a stop in a controlled fashion. The step count will be correct so the move could be continued by a later move command.

There are two versions of the Stepper Driver. Step_driver2_sd is for motor drivers that have a step and direction input. Step_driver2_cc is for motor drivers that have cw, and ccw inputs.

This is a schematic of how my motor is connected.



Run the demo program.

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

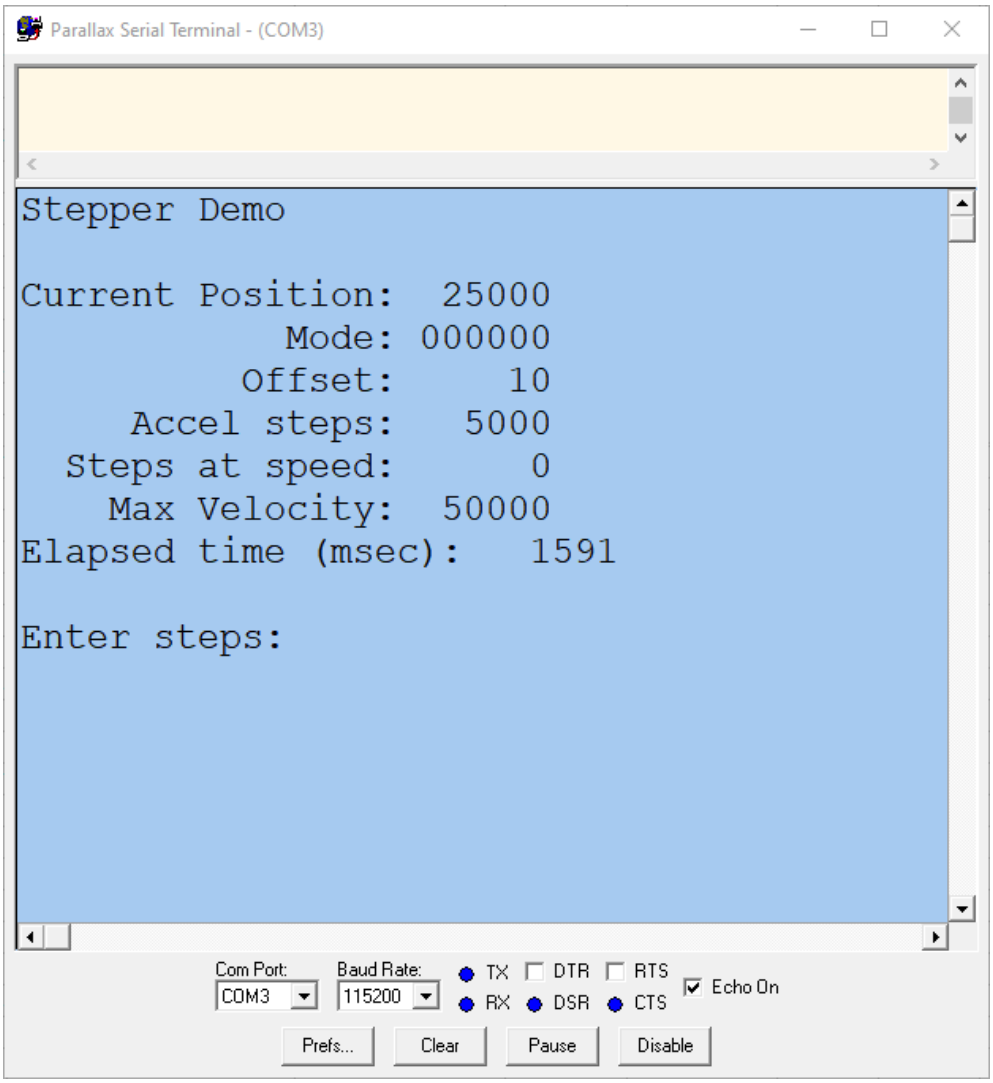
Press Enter to start.

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

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

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

You can also monitor the mode value to determine when you can send a new command.



Enjoy...

From the forum

So what is the API, what input does you driver software accept?

obj
stepper : "step_driver2_sd" ' pasm stepper routine for step and direction type
Pub
stepper.start(sspin, cpin, dpin) 'start/stop pin, (step pin, dir pin to driver) .....Where sspin - specifies a pin that must be held high to run, pulled low causes a controlled deceleration to stop.

stepper.move(speed,steps) ..... This is the call to the stepper routine, where speed is steps/sec and steps = number of steps to move (+/-)

The driver updates a position variable as the move is in progress
A mode variable that informs if a move is in progress, direction and current phase (acceleration, at_speed or deceleration)
and a steps to move variable(STM) that is set to zero to indicate that the routine is ready for another move.