A Simple GWBASIC Approach to CNC

I have built several "CNC" lathes and X-Y platforms of various sizes since 1978. I haven't the patience to learn all the caveats involved with servo/shaft-encoder control, though I understand the new Gecko 300 series make that plausible for a dummy like me.

I began with simple unipolar stepper control in my first CNC lathe, using an "8K PET" to generate the stepping-pulses to the home-brew stepper-motor drivers. Later, I used much-better bipolar KML-series steppers with Superior Electric bipolar chopper drives. First, a detail about computer-output to these drives: One can control a stepper-drive with two bits (plus ground, of course!) of the printer-port (aka "LPT-1") in either of two ways: One way, as I have done, with one bit carrying "CW" pulses for the "X stepper" while the second bit is held LO, and that second bit then carrying the "CCW" pulses for the "X stepper" while the first bit is held LO. Then the third and fourth bits are the CW and CCW lines for the Y stepper in exactly the same manner.

I wish now I had realized it is more "industry-standard" to have "STEP and DIRECTION" lines, not separate CW and CCW lines. The only difference would be the simple machine-code patch used to "toggle" the respective X- and Y- bit-pairs of the LPT-1 would be slightly different so that it ALWAYS stepped the FIRST of a pair, and direction of stepping was determined by (first!) setting the second "direction bit" of a pair either HI or LO. For any future work, I WILL do it the S & D way, to be sure! The Gecko drives want S & D inputs, so, to GET those inputs, I must use a simple 74HC02 latch and inverter lashup to "translate" CW and CCW to S & D, instead, for both X and Y stepper drives.

For my lathe, I have the tailstock ram activated pneumatically, the "shop air" to that controlled by a small DC solenoid-valve (Clippard EVO-6, which works fine on 5.0 VDC) Controlling that, I use the next TWO bits connected to the input of a solenoid-driver Darlington (ULN 2003) through a 74HC86 exclusive OR gate. This way, upon power-ON, both bits may be both HI, and later, after "power-up-boot settling", both go low, and the XOR gate then precludes that valve opening during power-up. Then, software makes only ONE of those bits go HI to extend the tailstock ram.

A similar arrangement is used for the SSR that controls the spindle-motor (or the drill DOWN/up motor in the PCB-drill). If BOTH the seventh and eighth bits are HI (or LO) at the same time, the motor will not start. Only when ONLY ONE bit is made HI, will the spindle motor run.

Some PC's have "byte-wide buffers" for their LPT-1 outputs, I guess, as a couple of the desktops AND notebook computers I use for this can source 10 mA on all eight bits, maintaining a bit-voltage of 4 volts. But "standard specs." don't guarantee more than, what, 2.8 VDC on a line sourcing much lower current, measured in microamps. Though these same specs do guarantee a good LO of 0.8 volts maximum, if SINKING 10 mA or so, I guess. So, on my machines, I use a 74HCT541 octal TTL-in buffer, which will accept a "worst-cast TTL valid level", and can SOURCE 25 mA at 4.5 VDC with ease, between the LPT-1 and the stepper-drive opto inputs. This requires some careful layout, component soldering, proper cable, etc.! With all these connections made as described so far, the only thing left to do is write some software that can control those first four bits controlling the two steppers in both directions, and the following two pairs of bits to control tailstock, spindle-motor, or drill down/up.

I wrote a bit of very simple ASM-86 machine-code which has eight "entry points" and eight "RETurn" points. When, say, the FIRST point is CALLed by BASIC, it "toggles" bit-zero of the LPT-1. That is, it makes it "go high" for a few microseconds, then low, again, then returning to BASIC control. That causes ONE step of the X stepper in the CW direction. If the SECOND point in the ASM-86 code is CALLed, it "toggles" bit-one of the LPT-1, which causes ONE step of the X stepper in the CCW direction. Same for bit-two and three and the Y stepper, respectively. If that ASM-86 point is CALLed in a FOR-NEXT loop in BASIC such as:

150 FOR X=1 TO 200:CALL X1:NEXT

This causes the X motor to make 200 steps—exactly ONE turn—in a direction that moves the carriage rightward. Then,

160 FOR X=1 TO 200:CALL X0:NEXT

causes the X motor to make 200 steps in the opposite direction. Note "X1" and "X0"; those are the addresses of that machine-code which cause the step-pulse outputs on the LPT-1.

I also write four simple patches which toggle BOTH the X and Y motors simultaneously in each of four directions we can call "N.E., N.W., S.E., and S.W." There is also a delay loop in the machine-code which is a subroutine each stepping patch must go through, and the "delay-loop count" is a byte which the BASIC can POKE, setting the STEPPING RATE. This delay-loop serves to WIDEN the HI pulse-width, the LO timing being that during rest, or, at least the time it takes control to return to BASIC, which is, of course, much longer than that HI pulse-width.

For example, the Address of the Delay-loop number is "AD", and useful numbers for the notebook which controls my lathe are 3, 8, 15, 30, and 60. Note, these values are respectively for rapid-traverse, rouging, smooth-finishing, and two rates of "plunge", and are optimum for hard plastics and woodworking. I have not yet figured any metalworking rates, etc., as my machine was built specifically for woodworking, not metalworking. Other PC's will require different delay-loop numbers, of course.

So, if I first use a line:

120 POKE AD,30 then,

130 FOR Y=1 TO 2000:CALL Y1:NEXT

then the large "delay value" 30 is in AD, so the Y motor will step fairly slowly, ten turns. For my Tandy 1500HD notebook, a delay-value of 3 is a practical minimum, as a 1 or 2 causes jitter, and zero causes lock-up in the ASM-86 code.

So, for "rapid traverse", I'd have this line first:

120 POKE AD,3 then,

```
130 FOR X=1 TO 1000:CALL X0:NEXT          This causes the X motor to step 5 turns rapidly CCW.
```

I have NO clue how to do a drawing here, so I will try to describe a very simple shape to be turned, and how a BASIC program would handle that: Say we have a cylindrical blank 50 mm. in diameter, fixed between centers, which is 200 mm long (yes, I used METRIC lead-screws, 4 mm pitch, that is, 50 steps per mm. of linear move). The bit is a right-hand bit, the corner of which has been set to the extreme right end of the stock, and JUST-tangent to the 50 mm. O.D.   Now, we will turn the entire O.D. 2 mm smaller in diameter, and a 30 mm length of the right end down to 40 mm, taking no passes deeper than 2.5 mm. per pass. For our purposes here, we will assume the tailstock has been manually set-up (though I CAN engage/release it via computer-control, and always do!), and we must manually switch the spindle-motor ON or OFF (though, again, I ALWAYS do this via software, controlling those last two LPT-1 bits!).

Motor on, and,

```
100 POKE AD,15                          (A fairly slow stepping-rate)
110 FOR Y=1 TO 50:CALL Y1:NEXT          (Moves the bit 1 mm. deeper into the work)
120 FOR X=1 TO 10000:CALL X0:NEXT       (Moves the bit 200 mm. to the left)
130 POKE AD,3                           (Set rate to "rapid-traverse")
140 FOR X=1 TO 10000:CALL X1:NEXT       (Move rapidly back all the way to the right)
150 POKE AD,15                          (Set rate to slow, again)
160 FOR I=1 TO 4                        (Repeat following four times)
170 FOR Y=1 TO 110:CALL Y1:NEXT         (Plunge toward the axis 2.20 mm)
180 FOR X=1 TO 1500:CALL X0:NEXT        (Move leftward 30 mm)
190 POKE AD,3                           (Set rate to "rapid-traverse")
200 FOR X=1 TO 1500:CALL X1:NEXT        (Move rapidly back all the way to the right)
210 NEXT I
220 POKE AD,15                          (Set rate to slow, again)
230 FOR Y=1 TO 10:CALL Y1:NEXT          (Plunge toward the axis 0.2 mm; a "trim-cut")
240 FOR X=1 TO 1500:CALL X0:NEXT        (Move leftward 30 mm)
250 FOR Y=1 TO 450:CALL Y0:NEXT         (Move back out of work)
260 POKE AD,3                           (Set rate to "rapid-traverse")
270 FOR X=1 TO 1500:CALL X1:NEXT        (Move rapidly back all the way to the right)
```

Turn off spindle-motor and wait for it to stop spinning (if no manual brake). Release tailstock and remove piece.

Now, see my "Preamble" GWBASIC listing ALSO just FILEd. I use that, plus whatever BASIC for each individual piece I machine. AND, this "Preamble" permits "rapid traverse" set-up moves of 200 steps each, or SINGLE steps, using the "arrow keys".

Jan Rowland.