

CNC for Hobbyists
Alan Marconett KM6VV
Hobbit Engineering
1/7/03

The time has come! Hobbyists are now able to afford a “home shop” CNC!

What is CNC?

- CNC stands for Computer Numeric Control. It uses a computer to read a part program, and command motions on the mill/lathe to make parts.
- A CAD program draws the part, while the CAM program generates the part program.
- A hobby CNC system consists of the following parts:
 - Computer
 - Controller program
 - Part Program
 - Stepper drivers (or servos)
 - Power supply
 - Stepper motors
 - CAD/CAM program(s)
 - Oh yeah, and a mill/lathe!

Computer

- IBM PC based computers are plentiful, and cheap.
 - A Pentium, 486, or even an old 386 can often be used.
 - You will probably want to run DOS 6.0 or 6.22 to support the Controller program, although programs like FlashCut, DeskWinNC and Master5 can run in Windows 95 or newer.
 - The computer’s parallel port is used to send step/direction signals to the drivers (FlashCut and DeskWinNC use a serial port).

Controller Program

- The controller program runs under DOS (typically).
 - Its job is to read your part program, and generate signals over the parallel port.
 - The parallel port is connected to the stepper motor drivers and the limit switches.
 - The controller program can also accept input via MDI (manual data input), and make moves for you, select Gcode settings, or issue Mcode commands.
 - Keyboard Jog keys (usually arrow keys) also move an axis, useful to “touch off” on an edge, or just make simple cuts.
 - Examples are CNCpro (Yeager), TurboCNC (free), Master5, FlashCut, DeskWinNC, and my STEP4.
 - “RUN” screen. This is where we will machine a part!
 - This screen has a window for viewing the lines (blocks) being executed, and the next line.
 - The Command menu allows the user to Reset, Start, Stop, and Single Step a program, or goes to a particular program line.
 - Gcode and Mcode command menus.
 - Several modal Gcodes and Mcodes can be set.
 - Modal codes maintain their value until explicitly set again.
 - The controller program may also have these other screens:
 - A “HELP” screen. One of the most important!
 - Index and contents functions allow access to keywords.

- A search function allows you to search with your own keywords.
- An editor to allow the user to write/edit a program.
- A “VIEWER” (Plot) screen to allow the user to preview tool paths, which is good way to study the operation of a part program.
- A “PARAMETER” screen, where axis resolution, axis acceleration, rapid rates, and a host of other parameters may be set.
 - Gcode executed by the controller program can be saved to a “logging” file by selecting logging.
 - Block deletes.
 - Program stops.
 - Tool change.
 - Fast updates.
 - Backlash correction.
 - Motor polarity.
 - Axis scaling.
 - Estop and limit switch controls.
 - Pulse width.
- A “FILE MENU” screen to allow selection and load/save of files.
- File searches by search pattern.
- A “JOG” screen that allows the user to move the axis manually.
 - The STEP4 controller also allows the user to move by calculated incremental steps. This is useful for cutting gears and pulleys, and indexing on the rotary table.
 - Jog distance and rate controls.
- A “FIXTURE” screen for viewing and setting fixture settings.
 - G54, G55 and G56 fixture offsets.
 - Tool lengths and diameters.
 - A “DIGITIZER” screen.

Part Program

- The part program is a file consisting of a list of moves needed to produce a part.
 - Generated by the CAD/CAM program.
 - An ASCII text file that can be edited with Notepad, if desired.
 - Read by the controller program.
 - The part program is in the Gcode (RS-274) language, or sometimes HPGL (plotter files).
 - Gcode consists of blocks and words.
 - A block is simply a line of ASCII text, consisting of words and coordinates.
 - A word is a letter and a number (G01).
 - Coordinates are words, an axis letter and a value (X1.000).
 - These words are interpreted and used to create motion commands (a series of step and direction pulses) to be sent to the motors.
 - The block (G01 X1.000) instructs the CNC system to move the X-axis to the 1.000” position.
 - Moves can be in absolute (G90) or incremental coordinate (G91).
 - Absolute locations are always given from the origin.
 - If the above move were incremental, then the X-axis would move 1” FROM where it was (the immediately preceding point).
 - The Set Origin Gcode (G92) allows the machine’s coordinates to be set by the program.
 - Moves can be in either inch (G20) or millimeters (G21).
 - There are three or four basic types of moves.
 - Linear moves (G01) command the hardware to move at a selected rate (feed rate).
 - Rapid moves (G00) move the hardware at a selected rapid rate.
 - Arc clockwise (G02) and counterclockwise (G03) move the hardware in arcs.

- Arc moves, move a pair of axis in a plane. The XY plane is selected by a Gcode command (G17). The two other planes (XZ and YZ) can also be selected by Gcodes (G18 and G19).
- More then one axis can be moved at a time, as in “G01 X1.000 Y1.000”
- Gcodes G42 and G43 accomplish cutter diameter compensation.
- Gcode G41 allows cutter height compensation.
- Subprograms allow portions of a program to be repeated (G22).
 - A label (\$20) marks the target subroutine.
 - The return from subroutine Gcode (G02) is used to return to normal program flow.
- Jumps (G23) can be used to alter the program flow.
 - Useful for program testing. Otherwise its use results in “spaghetti code” (to be avoided).
- Canned cycles for drilling, reaming and boring (G80-G83).
- Feed rate Gcode (F2.0).
- Spindle speed Gcode (S1000).
- Mcodes (M02) or miscellaneous codes control the execution of the part program, or turn certain machine functions ON or OFF.
 - M02 signifies the end of the program.
 - M01 provides a “stop” in the program.
 - They can also turn auxiliary equipment on and off (M3, M5 spindle control), or coolant (M7, M8).
 - Tool change (M06) can be programmed to allow the program to pause for the operator.
 - End of program (M30).
 - Program stop (M00).
 - Optional stop (M01).
 - Sub program end (M02).
 - There are many other G and M codes!
- Line numbers (optional) can be added for program clarity (N10).
- Variables allow programs to reference “canned” values (V1 = 2.222).
- Block deletes (/) are used to temporarily prevent program blocks (lines) from executing.
- Comments are enclosed in (comment). Recommended!

Stepper Drivers

- Use the step/direction commands to energize the windings of the stepper motors in a sequence, so that the motors turn.
- Occasionally 4-phase signals or CW/CCW signals are used.
- Stepper drivers come in two basic types:
 - Unipolar (characterized by a large L/R resistor) drive 5 or 6 wire motors.
 - Cheap, but typically have lower power.
 - Other low cost drivers are also available (HobbyCNC).
 - Bipolar choppers (high efficiency), drive 4, 6 or 8 wire motors.
 - Geckos are excellent bipolar driver examples (one module per axis). They are small modules, and you’ll have to connect them together in your driver/PS chassis.
 - Geckos can use 24-80V.
 - Camtronics boards (bipolar) are also a good choice (3 axis per board). These boards are available as kits.
 - Camtronics boards and others using integrated driver chips are usually limited to about 36V.
- Greatest efficiency is obtained when the motors are driven from voltage sources 5-20 times their rated voltage (it won’t hurt them, if a chopper drive is used).
- Often built into the same enclosure as the power supply.
- Use DB9, DB15, Mike connectors, or DIN connectors for the wires to the motors.
- Use a DB25 for the parallel interface.

- Microstepping drivers (Geckos) can be set up for full step, or 2, 5, or 10 microsteps per motor step. This increases resolution of the drive system, and gives smoother motion.
 - Half step is common for other drivers, and full step is also available.

Power Supply.

- The DC power to run the motors.
- Can be from 12 to perhaps 80 volts for our purposes.
- Generally a “linear” supply, with a current rating anywhere from 2 to 20 amps (as required for the steppers).
- No regulation necessary.
- Build one!
 - Transformer, Rectifier Bridge, and filter cap.
 - Don’t forget switches, fuses, power indicators, and a line cord.
- My Sherline example uses two 28V 10A transformers.

Stepper Motors

- Cheap and available as surplus.
- Look for 4, 6 or 8 wires (for bipolar use).
- 5 or 6 wires for Unipolar.
- Motor ratings of 2-6 volts are generally the most useful (bipolar), while 8 – 24V is typical for unipolar.
- Current will vary depending on the torque of the motor. Between 0.5 to 6 amp motors are useful. (A big Bridgeport may need as much as 15A, while a 24V unipolar motor may want as little as 0.18A).
- Steppers are also rated in steps per revolution. We want 200 step motors, although 400 and 500 step motors are also available.
- Really cheap steppers may be 24 steps per revolution. They work, but don’t have much resolution. Good for experiments!
- My Sanyo Denki steppers are 180 oz/in, 4V, 2A.

Servo Motors

- Also available.
- Gecko drivers RECOMMENDED! (G310).
 - Can also be driven by step/direction signals.
 - Closed loop
 - Uses encoders for feedback.
 - My next CNC project will be to put 300 oz/in servos on my RF-31 mill.

The Iron.

- We’ll assume a mill here, and my example is a Sherline mill converted for CNC.
- Three stepper motors (X, Y, Z) are mounted on the mill with Sherline motor mounts. With a fourth motor on the rotary table.
 - These mounts are the BEST! They contain a pair of bearings set up to contain linear motion of the connected lead screw, and a split coupler to aid in alignment.
 - The Sherline example uses 20-tpi lead screws driven 1:1 by the steppers. This results in 8000 steps per inch, assuming ½ step mode. That’s .000125” resolution (not that the mill in this example is that accurate)! More resolution is available with higher microstepping, but probably not warranted.
 - The 4th axis (A) is a rotary table (Sherline). It has a 72:1 worm drive. When driven with a 200 step/rev stepper in half step, results in 80 steps per degree. Or .0125 degrees per step.
 - Limit switches can be added to the axis to prevent overrun.
 - Home switches give the controller program a constant position on the mill to return to.
 - Limit and home switches would be nice, I’ll probably add them some day to my Sherline mill!

CAD/CAM Programs.

- This is where you design your part!
 - A low cost or free program like IntelliCad or TurboCad can be used. Save tool paths in .DXF format (an AutoCad format).
 - A good CAD/CAM program like Vector CAD/CAM is highly recommended. Vector can generate the Gcode for you!
 - Also available are new programs like Dolphin, which can also generate Gcode.
 - If you need to convert DXF to Gcode, free programs exist, like Yeager's (CNCpro fame).
 - You can even write the Gcode manually, I often do short programs to mill off a surface, trim an edge, or drill some holes.

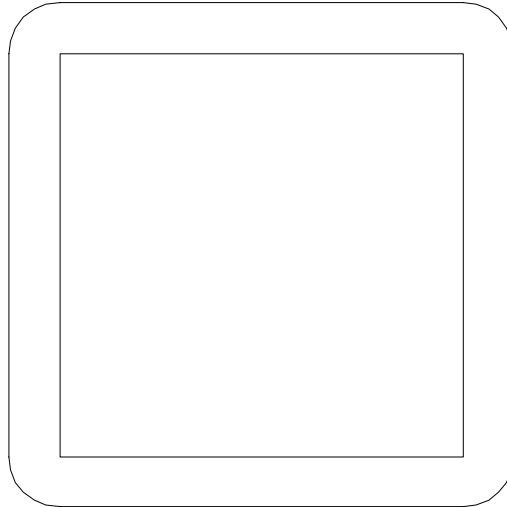
Coordinate System.

- In order to understand CNC, you need to have a basic understanding of the 3D Cartesian coordinate system.
- Devised by the French mathematician René' Descartes.
 - There are three axes, mutually perpendicular, the intersection of which is called the origin.
 - We can reference any point inside this "space" by measuring along the three axes (plus rotation).
 - All moves are done in one or more of these axis at a time. We move from one "point" to another. This is 3D.
 - On a vertical milling machine, the X-axis moves horizontal (left or right) of the table, the Y-axis moves across the table, and the Z-axis moves toward or away from the spindle.
 - On a lathe, the X-axis moves across the bed, while the Z-axis moves toward or away from the spindle (just like the mill).
 - We can do a LOT with just 2D! Often, all we need to do is add just a little more to a 2D drawing (2 ½ D), and we can cut out things like gaskets, signs, plane surfaces, make pockets, drill holes; well, all the major tasks a hobbyist might want to do! Just by moving the cutter up and down occasionally!
 - Movements are either point-to-point or continuous path (contouring).
 - Positioning the spindle to various locations for holes is an example of point-to-point positioning.
 - Contouring is where the cutting tool stays in contact with the part as it moves. This requires an accurate path.
 - Contouring cuts require interpolation. We are interested in linear and circular interpolation.
 - Linear interpolation is required for straight-line moves between points.
 - Circular interpolation is needed for arcs and circles.
 - There is also helical, parabolic and cubic interpolation for handling complex shapes.
 - Complex 3D shapes (like a sphere or bowling pin) will not be explored here!

Drawing

- If you draw a box in a CAD program, you essentially have four coordinates, and four vectors connecting them.
- In this simple example, we'll draw a box in our CAD program. 1" on each side, and put the origin at the LL corner of the box.
- We will define part surfaces from the three perpendicular reference planes (our axes).
- The part normally has one or more of its surfaces in these planes.
- We dimension from a specific point on the part's surface (a reference point).
- What the controller program needs is a starting location, and a series of points to travel to. We can cut a square by moving the X, Y and Z-axis at the appropriate times.
- We'll let the controller program convert inches to steps for us.
- We'll align the horizontal and vertical sides of our box with the X and Y-axis, and set the Z-axis at the surface (Z=0).

- It is common practice to have the top surface of the part at $Z=0$ and in the XY plane. This way, all machining happens below $Z=0$. Any moves that are above this plane (watch out for those clamps) will not cut the part.
- The moves we'll have to make to create a box are:
 - Rapid to start (assume LL corner, and drawing clockwise).
 - Move to first corner (moves Y +).
 - Move to 2nd corner (moves X +).
 - Move to 3rd corner (move Y -).
 - Move to start (move X -).
- We've made a box! The Gcode commands to do this would be:
 - N20 G00 X0.000 Y0.000 (rapid to LL corner).
 - N30 G01 Y1.000 (Y up)
 - N40 G01 X1.000 (X right)
 - N50 G01 Y0.000 (Y down)
 - N60 G01 X0.000 (X left)
- What we didn't mention is doing the rapid move with Z ABOVE ($Z0.5000$) the surface, lowering it for the four moves, and then raising it after we're done. Sort of like a plotter!
- If we add the Z-axis moves, and a few other nice things we get:
 - N05 (simple box program)
 - N06 G90 G20 G17 F2 (ABS coordinates, inch mode, XY plane, and 2 ips feed)
 - N10 G00 Z0.500 (rapid out of the way of the surface)
 - N15 M03 (spindle on)
 - N20 G00 X0.000 Y0.000 (rapid to LL corner).
 - N25 G01 Z-0.100 (move at feed rate to our cut depth)
 - N30 G01 Y1.000 (Y up)
 - N40 G01 X1.000 (X right)
 - N50 G01 Y0.000 (Y down)
 - N60 G01 X0.000 (X left)
 - N65 G00 Z0.500 (rapid up out of the way)
 - N70 M5 (spindle off)
 - N80 M02 (end of program)
- We've written our first Gcode or part program! Now if we were to put a scribe or a pen in the spindle, we would get an accurately drawn box, 1" on each side. If we put a 1/4" endmill in the spindle, and run the part program, we WON'T get a 1" square! Why?
 - The answer is that the endmill has some width. We need to compensate for this.
- This is why a CAD/CAM package is SO useful! In addition to generating the Gcodes, it can be used to draw an "offset" around the box to follow. We call this a "tool path".
- A CAD drawing (the square inside).



- This simple box was drawn in a CAD program (Vector CAD/CAM). An offset line around the box was created (note the rounded corners), and Vector generated the Gcode for the tool path (offset line).

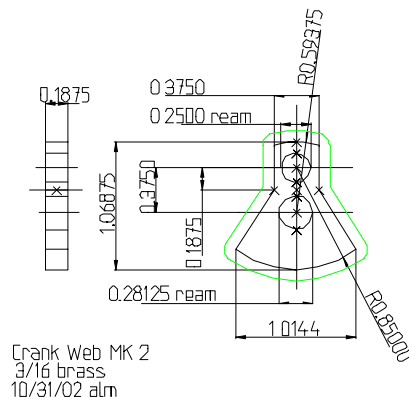
```

N01000 (box)
N01010 G90 G20 G17 F2
N01020 G00 Z.500
N01030 M03
N01040 G00 X-0.12500 Y0.00000 Z0.800
N01050 X-0.12500 Y0.00000 Z0.10000
N01060 G01 X-0.12500 Y0.00000 Z0.000
N01070 X-0.12500 Y1.00000
N01080 G02 X0.00000 Y1.12500 I0.1250
N01090 G01 X1.00000 Y1.12500
N01100 G02 X1.12500 Y1.00000 I0.0000
N01110 G01 X1.12500 Y0.00000
N01120 G02 X1.00000 Y-0.12500 I-0.12
N01130 G01 X0.00000 Y-0.12500
N01140 G02 X-0.12500 Y0.00000 I0.000
N01150 G00 X-0.12500 Y0.00000 Z1.600
N01160 M05
N01170 M02
□

```

- Above is an actual program generated by Vector CAD/CAM of a simple box.
 - You'll notice that arcs are used at the corner (G02), and that the coordinates of the vectors (the moves) are adjusted to allow the endmill to cut a proper size box (X-0.1250).
- If a CAD/CAM program is not available, then one would export the tool path in a .DXF file format, and convert it with a Gcode utility program (Yager).
- Another way to get the tool paths if the controller supports it (STEP4 does not, yet...) is to use G42 and G43.
 - Many CAD/CAM programs do this for you! Thus offsets at ½ of the tool diameter are not needed in the drawing. They are input into the controller program's fixture screen instead. Neat!

A more complex contour:



- You'll see that once again, an outline is drawn around the part. The holes could be either milled, or drilled.
 - A G83 drill cycle is useful here, because it tells the controller program to "peck drill" the holes for us!
- By using the CAD/CAM program, all dimensions are taken from the drawing, and tool paths need not be calculated. Gcode part programs like this can run into the 1000's of lines (blocks). Try writing THAT by hand!

Cutting Parts

- Now to cut the part!
- We have a few steps to follow to run the controller program:
 - Run the controller program (STEP4) on a DOS computer.
 - Load the part program (SimpleBox).
 - If the part reference point is not positioned at the machine zero point (home), then we move the axis (use jog) to the part's reference point, and set the machine coordinates (or the coordinate system) to 0,0 (G92 X0.0 Y0.0).
 - If the tool tip is not at the part's top surface when Z=0, then we can use G92 to set it as well.
 - The G54-G59 reference frames can also be used. Useful for multiple parts.
 - Run the part program by going to the appropriate screen or menu (RUN screen).
 - Select RUN, and the program executes through the lines (blocks) of the program automatically.
 - That's it! WE'VE CUT A PART!
 - If we need to pause or stop the program, the STEP4 program utilizes the SPACE key to pause, and for emergencies, the ESC key.

Viewing Tool Paths

- Before actually cutting a part, we may want to view it first.
- Several "viewers" are available; Vector furnishes one (Qsim) with its CAD/CAM program.

- CutView is one I have and like.
- STEP4 and some other controller programs allow you to view the tool path before cutting.
- Windows programs (FlashCut) allows you to view the tool path WHILE you're cutting.
- For STEP4, go to the Plot Gcode screen, and select PLOT on the first (configuration) screen.
- Various offsets can also be set on the configuration screen, to allow a better "view" of a particular area of the presented tool path.
- For convenience, a simple ENTER keystroke when the graphics screen comes up will run the entire tool path. The execution can be halted by pressing the "S" key, or aborted by pressing ESC.
- The "S" key actually changes to single step, and allows the part program to be executed one step at a time, a "walk" through the program.
-