G-ZERO Mill

Power Tips

G-ZERO CAM Software

RAPID OUTPUT

(925) 264-4495
www.g-zero.com

Contents

Helix

Ramping 33)Zmove + 10)Round

To ramp into a pocket along a straight line, you normally program this:

MILL
POINT
ZMOVE
POINT or RECT or COMP or ROUGH

Now you may ramp down along a curved line (helix) by programming a 10)Round after the 33)ZMove:

MILL	zrapid.1 zcut0 passes 1...
POINT	x1 y1.25
ZMOVE	z-.55 ramp.5 f2.2
ROUND	dia-.875 x1 y1 thru90

Note that the Mill command zcuts to the start point of the ramp. Only 1 pass will work here. The Point is optional, but provides a safe place for the MILL to come down. Zmove gives the final Z depth. For a full helix, enter ramp 1. For half an arc of helix, enter .5. Numbers larger than 1 trigger threading. Round does most of the work. In this example, thru90 starts the helix at the top of the circle (90 degrees).

Threading 1 33)Zmove + 10)Round

With a single-point tool, cutting threads is almost as simple as helical ramping:

MILL	zrapid.1 zcut0 passes1...
POINT	x0 y.125
ZMOVE	z-.55 ramp8 f2.2
ROUND	dia-.875 x0 y0 thru90
POINT	x0 y.125

For 8 threads-per-inch, enter 8 at Zmove's ramp question. To conventional-cut (normal thread in this case) enter -8. A Point after the Round pulls the tool tip out of the thread. Note that G-ZERO calculates partial arcs to finish at exactly the Z depth given in the Zmove command.

Threading 2 33)Zmove + 10)Round

With multi-point tool, you may only need one pass around; so, set yourself up for one loop:

MILL	zrapid.1 zcut-.5 passes1...
ZMOVE	z-.55 ramp1 f2.2
ROUND	dia-.875 x0 y0 thru1
POINT	x0 y0

For 20 threads-per-inch, the total cut distance is a mere .05, so you can rapid almost all the way to the bottom of the hole. Tell Zmove to ramp just 1 loop, and then tell Round this is a thru hole (thru1) and it will loop on, helix one full circle, then loop off.

Note: ROUND's arc-on/arc-off feature will work when helixing, but the arc-on and arc-off are flat (no Z movement). That means you may occasionally need to add a Z block directly to your G-code, if the tool leaves a mark as it arcs-off.

Tapered Threads (T-Helix)

T-HELIX is a G-ZERO option which calculates a very special kind of HELIX. The thread gets larger (or smaller) in diameter as it goes along. To drive a single-point or multi-point tool along such a thread, T-HELIX breaks each full arc of the cut into 8 arc segments. Each segment is slightly larger (smaller) than the last. To use T-HELIX, you must first program a start point in your source:

```
1    MAT'L      xmin-.625 xmax.625 ymin-.625 ymax.625 thk2 type0=ALUM
2     TOOL 1    dia1.25 flutes1 type1=CARBIDE MILL rad0 ***SINGLE PT
3    MILL       zrapid.05 zcut-.4 passes1 zret.05 zf10 xyf10
4    POINT      x.1 y.1
```

The start point will ALWAYS be the center of the hole and for right handed threads will always be at the bottom. Start left handed threads from the top. Next, call the routine by entering ##THELIX:

```
5                ##THELIX.EXE
```

Note: There is no dash (-) between T and HELIX. Your screen will clear, then several questions come up:

Starting Diameter of thread?

How many TPI (-=G02)?

Taper (in degrees) [1.78333]?

Start Z?

End Z?

IJ Arcs= (1=yes, 0=no) [0]?

After you answer the questions, T-HELIX will add lines to your source. For example:

```
6                %G1Z-.4F10.
7                %G91G41X.5Y0D51F5.
8                %G03X-.1455Y.3545Z.0156I-.5007J.0016
9                %X-.3545Y.1482Z.0156I-.3561J-.3538
10               %X-.3564Y-.1463Z.0157I.0016J-.5034
11               %X-.149Y-.3564Z.0156I.3557J-.358
…
31               %X.157Y.3757Z.0156I-.375J.3773
32               %X-.1551Y.3776Z.0156I-.5334J.0016
33               %X-.3776Y.1578Z.0156I-.3792J-.3769
34               %G1G40X0Y-.5354
35               %G0G90Z.1
```

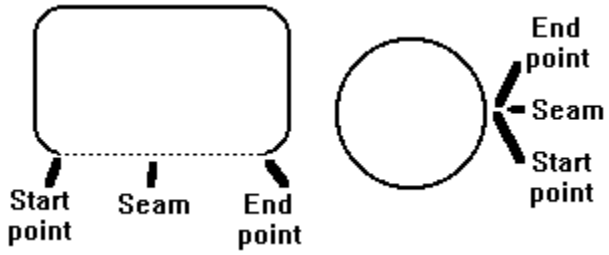
Note 1: T-HELIX produces G-code (%TEXT) and not source code. So, only the start point is displayed in graphics.

Note 2: G-code is in G91 or incremental mode. (In lines 8-33 of this example, Z values are positive because they are a positive increment cutting from bottom to surface -- right handed threads). This makes it easy to position to another start point and then just repeat the G-code:

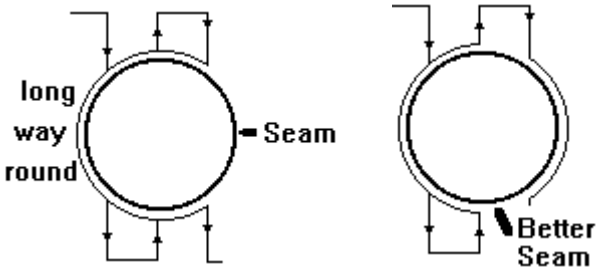
```
36   MILL       zrapid.05 zcut-.4 …
37   POINT      newX newY
38   REPEAT     from6 thru35
```

Island Roughing

Complete Control of Island Roughing can be maintained once you understand that ROUGH won't jump a seam. "What," you ask "is a seam?"



Each island is made up of a contour that has a start point and an end point. Between the start and end points is a seam which is usually very small (zero, in fact). When ROUGH chooses a path around the island, it will never touch the seam. Instead, the tool will run all the way around the other side of the island.



When the tool travels the long way around an island, all you need to do is move the seam. If you are using ROUND, then just alter the start angle.

For more complex contours, use 25) MOVE to start at a different point. A better seam could save many minutes of machine time on a complex job.

Seams can also help keep an oversized roughing tool from trying to squeeze into places it won't fit. When you know you have islands near a wall, put the seam near the wall. When islands are grouped closely together, put the seams toward the inside of the cluster.

Macros

A MACRO is similar to a CNC subprogram. It is a separate file created outside G-ZERO source programs and contains a set of source commands that will be used over and over as a group. A MACRO can be used in one or many different source programs. A typical MACRO may look like:

1,5,2,1
9,-5,2,-5,7,0
1,2/3*-1,2 <—mathematical expressions are always executed in LEFT to RIGHT order.

Create a Macro

1. Start Windows Notepad, WS, or your favorite text editor.
2. Type the Macro commands. They are the same as in the G-ZERO source program but the Macros use ONLY THE NUMBERS for the commands. For example, TOOL is number 6, MILL is number 7, etc.
   Type one command per line, and separate each element in a command with a comma.
3. Save and exit your text editor. Note: G-ZERO assumes all macro files have file extension .U; however, you can use any file extension you would like.

The examples below show G-ZERO source code lines that have been translated into equivalent MACRO code. The first number that appears in the MACRO code is the choice number you would select for an operation when writing normal source programs.

Source	Macro
MILL zrapid.05 zcut-1 passes1 zret.05 zf2.7 zyf11	7,.05,-1,1,.05,2.7,11
RECT xmin-5 xmax2 ymin-5 ymax2/3*7 thru0	9,-5,2,-5,2/3*7,0
COMPUTE ID PART XC-25498	0,COMPUTE ID PART XC-25498

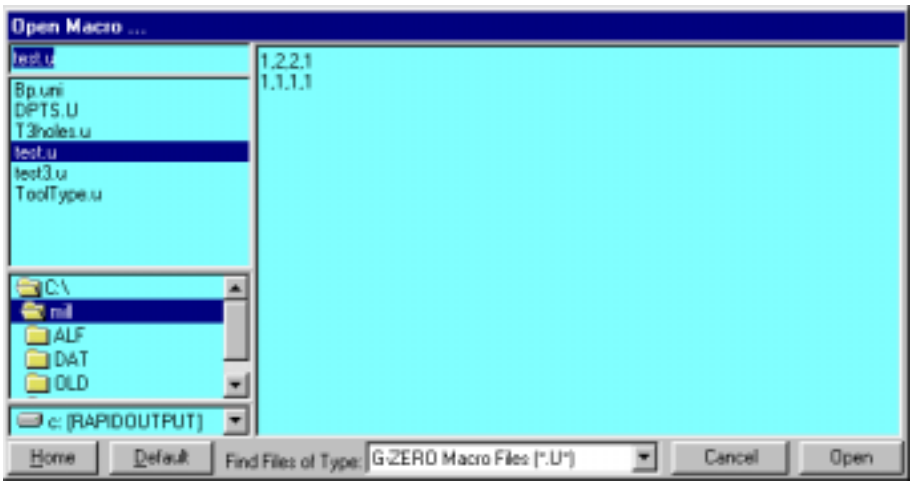
Call a Macro

To read the information of a Macro from within a source program, type # and the name of the MACRO as a comment in your source program. If the macro is located in a different directory, make sure to include the complete path. Example: #d:\lathe\test.u

The MACRO name must be placed exactly where the operations inside the MACRO need to take place in the source program.

```
Example:        21              COMPUTE OD PROFILE USING A MACRO
                22  POINT       x4 y2
                23              #MACRONAM
```

If you are using the G-ZERO Mill system, once you press the # key, an *Open Macro* window appears to assist you find the the macro file. If you are using a macro with file extension different from U, make sure to select *All Files* in the *Find Files of Type* section at the bottom of the window. The right section displays the contents of the macro file selected. Click the *Open* button to complete the macro comment line in your source program.



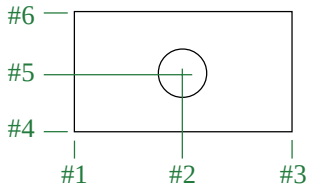
Variables

Your macro file can include variables to allow more flexibility. These variables are coded with the # symbol in front, and the actual values of the variables should be listed in parenthesis right after the name of the macro.

Examples:

We have a macro file named ShapeA.u with these values:

```
7,.1,-.51,1,.1,45,10
1,#1,#6
1,#3,#6,10
1,#3,#4,10
1,#1,#4,10
1,#1,#6,10
0,DRILL HOLE
8,83,.1,-.5,3,1,.1,1.3
1,#2,#5
```



(a) If we type this macro comment line in our source code: **#shapea.u (2,2.75,3.5,1,2,3)**

G-ZERO interprets the parameters are:

#1 = 2	#2 = 2.75	#3 = 3.5
#4 = 1	#5 = 2	#6 = 3

G-ZERO reads these source codes:

```
MILL   zrapid.1 zcut-.51 passes1 zret.1 zf45 xyf10
POINT  x2 y3
POINT  x3.5 y3 f10
POINT  x3.5 y1 f10
POINT  x2 y1 f10
POINT  x2 y3 f10
       DRILL HOLE
DRILL  g83=PECK zrap.1 zcut-.5 pecks3 tip1 zret.1 f1.3
POINT  x2.75 y2
```

(b) If we type this macro comment line in our source code: **#shapea.u (-2,0,2,-1,0,1)**

G-ZERO interprets the parameters are:

#1 = -2	#2 = 0	#3 = 2
#4 = -1	#5 = 0	#6 = 1

G-ZERO reads these source codes:

```
MILL   zrapid.1 zcut-.51 passes1 zret.1 zf45 xyf10
POINT  x-2 y1
POINT  x2 y1 f10
POINT  x2 y-1 f10
POINT  x-2 y-1 f10
POINT  x-2 y1 f10
       DRILL HOLE
DRILL  g83=PECK zrap.1 zcut-.5 pecks3 tip1 zret.1 f1.3
POINT  x0 y0
```

Super Macro

To import the contents of a Macro file into your source program, adding new lines of source, call the Macro as previously described followed by an exclamation mark "!". The contents of the macro file will be converted into source code lines and imorted into your source program right after the macro comment line. This is known as Super Macro. If you are using G-ZERO Mill, select the macro file as previously described. When the name of the file appears in the upper left section of the Open Macro window, type "!" right after the file name.

Example 1: **#mymacro.u!** imports the converted source codes of the macro mymacro.u

Example 2: If you would like to include the converted source codes of the macro ShapeA.u in example (b), type this macro comment line in your source code: **#shapea.u! (-2,0,2,-1,0,1)**

Hints

- Macros are a quick way to program a commonly used feature that occurs on more than one part or a “family” of parts.
- By adding variables, you can change the size, orientation and location of a feature described in a Macro or Super Macro. There is a limit of 10 variables per Macro.
- Regular MACROS cannot solve for unknown radius locations or REPEATs, so use a Super Macros.
- Macro names must begin with a letter, NOT a number.
- To program in the “User Macro-B” language of Fanuc and Yasnac, you will often need to program a line like: N1 #501=#501+1. Since G-ZERO wants to substitute for the ‘#501’ you must hide it by doubling the #. Example: 0,%N1=##501

Dup

When a G-ZERO Macro file becomes too limiting (not enough variables, or too complex a task required, or data needs to be looked up from another data file), then a DUP file offers virtually unlimited power and flexibility.

A DUP file is a Macro file with the extension .DUP, which includes prompts such as “How many steps per inch?” or “What is the X center dimension?” in its header. DUP files are “pulled apart” by a Basic program called DUP.B to create an output file of the same name as the DUP file, but without the .DUP extension. Users must import the output file using the # command.

Create a DUP

A DUP file starts with one or more questions coded with an apostrophe ' as the first character of each line. These questions are going to be prompted to allow users to key in the values for the variables. For example, in a Macro file you might put instructions like:

```
#Macro (xc,yc)
where xc = x center of circle,  yc = y center of circle
```

In a DUP, the actual prompts are used:

```
'Where is the center of the circle in X [0]
'Where is the Y center [0]
```

The second section of a DUP file looks the same as a macro. Each line of code contains numbers and symbols that represent G-ZERO commands with their parameters. An easy way to create a DUP is to:

1. Program the DUP source codes with G-ZERO.
2. Convert the source program to a macro.
   In version 4: Use File | Export Macro
   In version 3: Use "Copy a source file" from the G-ZERO front page, or "Backup a source file" to convert source to simple text.
3. Load the macro into your favorite text editor.

   Note for Version 3.0 users: Insert the command number in front of each line (eg: 6 TOOL, 1 POINT, etc.)
4. Type the questions for the first section of the DUP file. Make sure each line starts with an apostrophe `.
5. Replace each value of the source code that correspond to a variable with the number symbol # followed by a number that represents the sequence of the question. That is: #3 corresponds to the variable for the third question.
6. Save the DUP file under a name using the DUP extension.

Example:

```
'What is the diameter of the tool?
'Point 1: X=
'Point 1: Y=
'Point 2: X=
'Point 2: Y=
'Point 3: X=
'Point 3: Y=
6   TOOL 1 dia#1 flutes2 type1=CARBIDE MILL rad0 ***
8 DRILL    g83=PECK zrap.1 zcut-.5 pecks3 tip1 zret.1 f6
1 POINT    x#2 y#3
1 POINT    x#4 y#5
1 POINT    x#6 y#7
```

Use a DUP

You can use a DUP during a source code programming session whenever you are prompted to enter a command number.

1. Type 27 to activate the command Basic.
2. When you are asked for the program you would like to use, select or type in SUPDUP.
3. SUPDUP will list all .DUP files in the current directory. (Example: Bshape.dup).
4. Key in the name of the DUP file without the file extension. (Example: Bshape)
5. At this time, G-ZERO will ask you the questions included in the first section of the DUP file. Key in the values (parameters).
6. A new file will be created with the same name as the DUP file but without the .DUP extension. (Example: Bshape)
7. Import the file using the Super Macro format. (Example: #Bshape!)

SUPDUP

A SUPDUP is a G-ZERO Basic Program which includes prompts such as “How many steps per inch?” or “What is the X center dimension?” in its header.

SUPDUP files are “pulled apart” by a Basic program called DUP.B to create a G-ZERO Basic Program which is automatically run, once all the promts have been answered by the user. A well-written SUPDUP program creates AUTO.IMP as a final output file, thus G-ZERO automatically imports the new data, once the SUPDUP is finished.

SUPDUP files give programmers the benefits of G-ZERO’s prompting system plus the raw power of Microsoft Qbasic.