DotG

V0.631

Post-processor configuration.

The DotG post processor enables full user configuration of the generated GCode so that it can be tailored to suit almost any CNC machine controller. It allows for the inclusion of text strings such as the bitmap file name and final image size / machine extents as well as machine specific initialization strings and final homing codes etc.
The post processor itself is a text file (having a .dgp file extension) and this consists of four separate sections which are categorised as follows;

Format - This specifies the format of the variables.
Header - Strings and code which will be placed at the beginning of the GCode output file.
Moving - This section substitutes the defined variables and sequence into the GCode output file.
Pgmend - Code which will be placed at the end of the output file.

Each of these separate sections are contained within curly brackets **{ }** and care must be taken when making changes or additions to preserve the existing format / layout / syntax to avoid conflicts.

Variables used:

n – Automatically incremented line number.
xa – The current position of the X axis.
ya – The current position of the Y axis.
xh – X axis start position and home position.
yh – Y axis start position and home position.
zh – Z axis start position and home position.
za – Z axis safe distance above work surface.
zd – Z depth (depth of machining).
zt – The pause or dwell time for the Z axis (P).
sl – The X axis position if **Extend** has been selected.
xmin – X axis minimum.)
ymin – Y axis minimum.)
zmin – Z axis minimum.) Machine axis extents for the loaded image.
xmax – X axis maximum.)
ymax – Y axis maximum.)
zmax – Z axis maximum.)
pname – File output name (without extension).
tnum – Tool number (T).
sp – Spindle speed (S).
m34 – The M code to be set for spindle (M3, M4 or M5).
m78 - The M code that has been set for coolant (M7, M8 or M9).
bmpfile – The currently loaded bitmap file name.
f – X and Y axis G01 feed rate (F).
fz – Z axis G01 feed rate (F).
cs1 - User defined string.
cs2 - User defined string.
cs3 - User defined string.

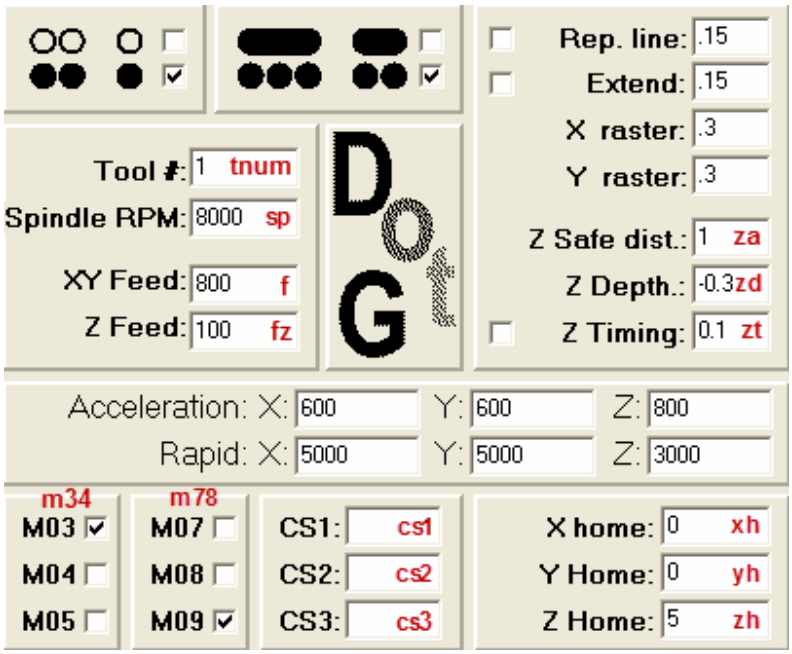

Fig.1 – The post processor area of the User Interface.

Variables are usually set within the User Interface and the relationship between the various fields and the associated variables are shown in red in the above illustration.

The following details the function of each of the four sections within the post processor.

Format: This section defines the output of the coordinate data (X, Y, Z), the feed rate, and format of the output file including line numbering if required.

{format	
[ext] tap	output file extension (typically .tap).
[seq] N | 0 | 1 |	defining a line number. N – the character that is placed before numbers (not mandatory). | 0 | - starting number. | 1 | - number increment.
[xy] d3 | x0 |	X and Y variables are defined (d and x unbound order). d3 = 3 decimal places. x0 - multiplying the base value. (x0 = 1x : x1 = 10x : x2 = 100x : x3 = 1000x)
[z] d3 | x0 |	
[f] d0 | x0 |	(typically no decimal places are required for feed rate).
}	

Header: The data here forms the start of the generated, GCode, output file. Text information, such as the file name and machine extents together with any initialization strings, tool number, spindle speed / switching and additional M codes are placed here.

```
{Header
(- DotG program -)
(- [bmpfile] -)
(Xmin: [xmin] Ymin: [ymin] Zmin [zmin])
(Xmax: [xmax] Ymax [ymax] Zmax [zmax])
G17 G21 G40
G80 G90 G94
T [tnum]
S [sp] M [m34]
G0 X [xh] Y [yh] Z [zh]
}
```

The contents between the square brackets **[ ]** is the name of the variable that will be replaced with the value that has been entered into the related field in the User Interface. This is true for all cases except for the format section.
If the text between the parentheses does not match exactly with the name of a variable (see variables used) then the entered text will be written to the output file in that particular location.

Moving: This section determines how the lines of GCode are to be written to the output file where axis movement is performed (This also determines the order or sequence in which they are written .). If the post-processor contains a subroutine (for example cs1), then the data entered into the relevant field (**cs1**) in the User Interface will be inserted at that position in the output file.

```
{Moving
G0 X [xa] Y [ya] Z [za]
G1 Z [zd] F [fz]
ifzt
G04 P [zt]
endzt
ifsl
G1 X [sl] F [f]
endsl
G0 Z [za]
}
```

Pgmend: This section contains the closing code and home moves (spindle off, Z safe height, home X and Y axis, program end). If there is a subprogram then it must be written after the main program closing sign.

```
{Pgmend
G0 X [xh] Y [yh] Z [zh]
M30
}
```

As mentioned earlier care must be taken to preserve the correct syntax and it is recommended that a new post is created, from a copy of the original, for testing prior to use for production work.

Post processor operating principal:

The post processor area of the User Interface contains a number of fields into which values for the various variables can be placed (see Fig.1). These values will then be automatically entered into the output file, at run time, in the position determined by the layout of the selected post processor file (*name* .dgp).
The post processor file itself can also be edited directly but if the contents between the square brackets **[ ]** do not match exactly with a variables name (see Variables used) then the contents will be written to the output file without change (including the square brackets).

The following is an example of this when **xa = 12.46** and **ya = -7.813**

Example:

G0 X [xa] Y [ya]	processed as: **G0 X12.46 Y-7.813**
G1 X [# 3]	processed as: **G1 X [# 3]** (no change)

Ifym: This test will insert a line of text into the output file which tracks the value of its associated variable. As the code progresses and the value of the variable changes then a line of text will be inserted that reflects the new value.

Example:

ifym
 # 2 = [ya]

When processed, assuming **ya=213.87** (Y axis position), then the output file will contain the line **#2=213.87** then as the GCode progresses to the next raster line on the work the new text line will contain the new co-ordinate position of the Y axis.

ifsl – endsl: This test will extend a single point, in the +X direction, by the value entered into the **Extend** field of the User Interface.
Its purpose is to increase the width of an individual dot (pixel) and depending on the value entered, can have the effect of merging neighboring dots into a continuous line. The function cannot have a zero value and is only active when the **Extend** check box has been enabled in the User Interface. The following is a typical example (when the Z axis has been lowered to Z Depth) the X axis is moved the specified distance at feed rate **[f]**.

Example:

ifsl
G1 X [sl] F [f]
endsl

ifzt – endzt: This test relates to Z axis timing which is essentially a ‘dwell’ with the time period being determined by the value entered in the **Z Timing** field of the User Interface. It would typically be used to allow time for the cutting tool to ‘bite’ into the work at the end of the Z axis travel but it could also be used for other purposes where a delay is required in the process. The **Z Timing** check box in the User Interface must be enabled for this function to be active.

Example:

ifzt
G04 P [zt]
endzt

Test conditions should only be placed in the **moving** and **pgmend** sections and they must always be placed at the beginning of the line. They can be nested and when this is done those opened first must be closed last as shown below.

Example:

```
ifsl
G1 X [sl] F [f]
ifzt
G04 P [zt]
endzt
endsl
```

A typical Mach3 post processor:

```
{Format
[Ext] tap
[Seq] N | 1 | 1 |
[Xy] d3 | x0 |
[Z] d3 | x0 |
[F] d0 | x0 |
}

{Header
( - DotG program - )
( - [Bmpfile] - )
(Xmin: [xmin] Ymin: [ymin] Zmin [zmin])
(Xmax: [xmax] Ymax [ymax] Zmax [zmax])
G17 G21 G40
G80 G90 G94
T [tnum]
S [sp]
M [m34]
M [m78]
G0 X [xh] Y [yh] Z [zh]
}

{Moving
G0 X [xa] Y [ya] Z [za]
G1 Z [zd] F [fz]
ifzt
G04 P [zt]
endzt
ifsl
G1 X [sl] F [f]
endsl
G0 Z [za]
}

{Pgmend
G0 X [xh] Y [yh] Z [zh]
M30
}
```

Notes:

Please remember that the post processor does not contain error trapping - the layout and correct syntax must be preserved so always observe care when editing.

The DotG generated GCode should be tested before any real work. There are a lot of good GCode simulators and most of them have built in test functions and warn of any conflicts etc.

Free CNC simulator: http://www.cncsimulator.com/

DotG latest versions, contact, user guides: http://dotg.weebly.com/

Any feedback is welcome. If you have any suggestions for improvements or corrections please do not hesitate to contact me from my website. http://dotg.weebly.com/

Please contact me if you require a post processor written for a particular application or machine.

Terms of use:

Enjoy !

Beni.

04th February 2012.