**Engraving Subprogram Documentation**

**Why use it**

This engraving program is incredibly versatile. It was designed to be easy to use for simple engravings, while at the same time allowing for extreme control on positioning. This system will allow you to easily program engravings that tightly follow contours, edges, and fit neatly between different part features.

While a CAM system can directly program engravings, they lack one feature that this system does easily and very well; serial numbers, sequential letters, and the ability to have dynamically changing engravings. With a creative programmer this system has very few limits and dozens of customizable features and options to fine tune every aspect of how the text is engraved.

**How it works**

The programmer will use the main shape subprogram (*Shape*) into the program of the part to be engraved and create an accompanying small (*Detail*) subprogram that details what the engraved text will be. The *Shape* program will handle the placement of the characters and call on a *Character* subprogram, the *Character* program will machine actual the individual characters themselves.

Defining an engraving is done by combining *Detail* subprograms to create a *Detail Block*. These *Detail* subprograms let the programmer define the letters, numbers, and symbols that make up the engraving, including special Serial numbers, Sequential letters, Date, and Logo. Every engraving will need one of these *Detail Block's* to define what the engraving is.

The main idea is to have the main *Shape* program look though a *detail block* to populate a range of variables to create a list (*Queue*) of what needs to be engraved and in what order. The *Shape* will then determine its start point and work down from the top of the *Queue* to position and machine the individual characters using the *Character* subprogram.

For the *Shape* program to look ahead and determine its start points it will look at the elements in the *Queue* and increment a spacing sum (*SSum*). The *SSum is* a sum of all the character widths in the entire *Queue.* Using the *SSum*, the *Upper Bound* and the *Letter Height* we can calculate the dimensional *Finished Size* of the engraving. From the *Finish Size* we can calculate where to start the engraving based on the justification and alignment requirements.

#504 holds a value that represents how many units of space the finished size of the engraving will take up when it is machined.

**The Queue**

The Queue is a range of variables that make a type of list. This list is the core element of the engraving program and is what makes the complicated shapes, alignment, justification, and serial/sequential numbers all possible. The Queue has 5 different variables that create and manage it.

There are two variables that create the Queue and manage the memory it uses.

#500 'points' to the variable that will be the start of the Queue.

#503 is the maximum length of the Queue, this directly impacts how many characters you can engrave.

For example, if #500 is 100 and #503 is 30 then the Queue will be #100 - #130.

There other two variables are used to manage operations with the Queue. These two variables should never manually changed.

#501 is the Upper Bound, this is how many characters are currently in the Queue.

#502 is the Index, this is what variable the program is currently reading.

The Upper bound will never exceed #503.

When a *Detail Block* is run it will add numbers to the variables in the queue. Each number stands for a different character that will be engraved.

It is extremely important to choose a range of variables for the *Queue* that does not interfere with any other programs on the machine. The *Queue* is refreshed every time an engraving is run.

**Shape Subprogram**

**Main Purpose/ Usage**

The Shape program is what will be in your main program and is used to determine the placement, size, and shape of the engraving. Currently only line and arc shaped engravings are supported. This subprogram will move the machine to the center of the first character in the engraving, start the spindle, and move the tool Z+.010 from the face of the part. The Shape program keeps track of the absolute position of the characters and passes the to the and other machining information to the Character Subprogram that finally machines the individual characters. There is many optional variables to fine tune the final engraving, however all of them have default values that will work for the majority of engravings.

**The two common configurations of the subprogram are as follows;**

*[] Denotes Optional Inputs, These inputs are optional and have defaults values if omitted*

*' ' Denotes Mandatory Inputs, All these inputs are not optional.*

Line Shaped Engraving

G65 P80089 'BXY' [UHJAWFSDQ]

Arc Shaped Engraving

G65 P80089 'BXYRE' [UHJAWFSDQ]

**Example of a Shape Program to engrave text on an Arc along the edge of a 4" Día part.**

***We want to engrave Detail Block O1000 centered along the Y+ axis and we want the characters to come within .1 away from the edge.***

G65 P80089 B1000. X0. Y0. R1.9 A90. J2.

**User Inputs**

All inputs need a decimal point.

**B = Detail Block, Mandatory**

Program number of the Detail Block that you want to engrave.

Be aware that the Shape program will run the program this variable points to. If it is not a detail block it will run that entire program and the chances of a crash are high.

**X = X Origin, Mandatory**

In the case of a linear engraving this is the X coordinate of the alignment point.

In the case of an arc engraving this is the X coordinate of the arc center.

**Y = Y Origin, Mandatory**

In the case of a linear engraving this is the Y coordinate of the alignment point.

In the case of an arc engraving this is the Y coordinate of the arc center.

**U = Shape, Optional**

The shape of the engraving, currently either a 1. for a linear shaped engraving or a 2. for an arc shaped engraving. If omitted Defaults to a 1.

**H = Letter Height, Optional**

Letter height is the height of each character. This will allow you to scale the characters to any size. Character width is half of H. If omitted Defaults to .125

**J = Alignment, Optional**

Alignment code for positioning of the engraving. If omitted Defaults to 5.

*See* ***Final Note*** *below for more information on Alignment Codes.*

**A= Angle, Optional**

In the case of a linear engraving this is the angle of the engraving.

In the case of an arc engraving this is the angle of the alignment point from arc center.

If omitted it Defaults to 0.

**E = Rotation direction, Optional**

1. for CCW, making the engraving face the inside the arc.

-1. for CW, making the engraving face the outside the arc.

If omitted defaults to 1.

**W = Letter Spacing, Optional**

Scale factor between two consecutive characters in terms of Letter Height.

If omitted defaults to .25

**D = Engraving Depth, Optional**

If omitted defaults to .003

**F = Cutting Feedrate, Optional**

If omitted defaults to 20.

**Q = Plunge Feedrate, Optional**

If omitted defaults to 10.

**S = Spindle RPM, Optional**

If omitted defaults to 5000.

**C = Pass Through Variable 1, Optional**

This variable will be passed along when the Detail Block is called. On its own does nothing, but allows extra logic in detail blocks. Referenced with #1.

**K = Pass Through Variable 2, Optional**

This variable will be passed along when the Detail Block is called. On its own does nothing, but allows extra logic in detail blocks. Referenced with #2.

**Additional Notes on Inputs**

R is only mandatory for arc engravings, but will have no effect with linear engravings.

All Optional Inputs have default values and only need to be used for advanced functionality.

There are many built in errors that will alarm out the machine when invalid inputs are entered.

**Final Note**

The alignment point system is one of the most powerful features of this macro. If you can imagine a box that fits perfectly around your engraving, the alignment codes allow you to choose a part of the box that the engraving is defined from. By default, the X, Y coordinate you give will be the center of the engraving, but there are several different options to choose from. In short that are 9 points that matter and the parts of the box are described on the next page.

**Alignment Codes**

|  |  |  |
| --- | --- | --- |
| 1  Upper Left Corner | 2  Upper Middle | 3  Upper Right Corner |
| 4  Center Left | 5  Center Middle | 6  Center Right |
| Lower Left Corner  7 | Lower Middle  8 | Lower Right Corner  9 |

By picking one of these alignment codes, you can control how engraving's box reacts when you add characters to it or when sequential/serial increase the engravings size. The codes make it easier to precisely position the engraving around, along, and between certain features of a part.

**Compatibility**

This subprogram uses mostly local variables #1-#33 for the bulk of its computations, which are not persistent. It does access persistent variables #500-#503. It also has access to all of the persistent variables within the queues range. To learn more about the queue go its section in this guide.

#504 contains the total size of the Engraving when its computed. With linear engravings this number will represent its true size from on end to the other. With arc engravings this number will represent the total angle of the arc the engraving will encompass. This can be used in your main programs logic if your engraving needed to be a certain size.

**Detail Subprogram**

**This subprogram has been completed on 2/28/2022, it has not been unit tested.**

**Main Purpose/ Usage**

Will append Letters, Numbers, Special Characters, and Serial/Sequential values to the queue.

Does NOT move the machine, this subprogram only appends operations to the queue.

A block of Detail Subprograms that starts with a Unique O# and ends with a M99 is used to define what needs to be engraved on the part. A detail block is in technically its own subprogram. There is no other way around this without severely complicating what needs to be in the main part program.

**The two common configurations of the subprogram are as follows;**

*| Denotes Exclusive Inputs, Only one of these can be used at a time.*

*[] Denotes Optional Inputs, These inputs are optional and have defaults values if omitted*

*' ' Denotes Mandatory Inputs, All these inputs are not optional.*

Normal Character Input

G65 P80088 'A|B|C' [ES]

Sequential/Serial Input

G65 P80088 'D' [EFHMR]

**Example of a Detail block to engrave '16701767 MBM 16058A -051'**

*Assuming #100 = 51 and #503 >= 25*

O1000

G65 P80088 B167017. (Part Number 1 of 2)

G65 P80088 B67. S1. (Part Number 2 of 2)

G65 P80088 A8853. S1. (MBM )

G65 P80088 B16058. (Job Number)

G65 P80088 A1. S2. (A )

G65 P80088 C1. (-)

G65 P80088 D100. E3. (Serial Number)

M99

**User Inputs**

*SS will stand in short for the special Sequential/Serial object*

All inputs, excluding F, need a decimal point.

**A = Alphabetical Input**

Letters/Words are represented as numbers, A=1 - Z=26... AA=27, AB=28...

Use special excel program to convert words into their base10 number.

*See* ***Final Note*** *below for more information on this.*

**B = Numerical Input**

Any number is acceptable, including decimal and negative numbers.

Decimal numbers will be rounded to 4 decimal places.

**C = Special Character Input**

Special characters are defined in the Character Subprogram.

Special Characters include Symbols, Date, Time, and Company Logos

**D = SS Base Variable**

This variable will 'point' to the variable containing the Incremental SS number.

For example, D100. will set variable #100 as being the variable that the SS will use.

The number in #100 will be the number that is incremented every cycle.

This allows for multiple SS values that can be changed independently.

**E = Padding, Optional**

Padding places leading 0's in front of a number to match a certain number of digits.

It is most useful in combination with D to make good looking serial numbers.

For example, B1. E3. outputs '001' , B12. E3. outputs '012' ,

B123. E3. outputs '123'. Note that when the digits match or exceed E it will have no effect. B1234. E3. outputs '1234'

**F = SS Number/Letter, Boolean, Optional**

When this is set to 0 the SS will output Numeric Serial numbers i.e. 1, 2, 3, ... 10

When this is set to 1 the SS will output Sequential Alpha letters i.e. A, B, C, ... AA

F defaults to 0 if it is omitted, so by default D will output Numeric Serial Numbers

**H = SS Increment, Optional**

This will change how the SS increments its value, only positive values are valid.

Decimals values allow for fraction increments.

If omitted H is defaulted to 1.0 and will increment the value by one every cycle.

For Example, H0.5 F1. outputs 'A, A, B, B, C, C...' , H2.0 F0. outputs '0, 2, 4, 6...'

**M = SS Maximum Value, Optional**

This will set a maximum value for the SS to increment to.

When omitted defaults to -1, which will disable itself and have no effect.

When M is reached the variable D 'points' to will be reset to the value of R.

This is useful for parts that are separated and match marked.

For Example, F1.0 M3.0 R1.0 outputs 'A, B, C, A, B, C...'

**R = SS Reset Value, Optional**

This will be the value that the variable D 'points' to will be set to when a reset is triggered by M being reached.

Only positive values are valid.

When omitted will default to 1.

For Example, assuming (#100 = 0), D100. H2. M5. R1. will output;

'0, 2, 4, 1, 3, 5, 1, 3, 5...'

Note that since the initial number was 0 it caused the start of the output to act oddly, also notice how it skipped 6 and went to 1 since 6 was bigger than M(5).

You must take this into account when setting up complicated SS engravings.

**S = Spaces, Optional**

This variable allows you to add spaces before or after your input.

Negative values will add leading spaces, Positive values will add trailing spaces.

When omitted will default to 0, which will not add any spaces.

Example A355414. S1. will output 'TEST ' , A355414. S-2. will output ' TEST'

*See* ***Final Note*** *below for more information on A355414.*

**Additional Notes on Inputs**

ABCD are all Exclusive Inputs, it is Mandatory to have exactly one of these Inputs.

EHMSR are all Optional Inputs

FHMR are only used with D

E has no effect when used with A or C, or when F = 1

All Optional Inputs have default values and only need to be used for advanced functionality.

There are many built in errors that will alarm out the machine when invalid inputs are entered.

**Final Note**

I understand finding the number that represents a word for the A input might be hard to wrap your head around. Of course, you can always program each individual letter separately, the main purpose of this 'word' functionality is for the Sequential letter system which fundamentally required it. If you wish to understand it fully it is further explained below;

The letters need to be converted from a Bijective base26 hexavigesimal number system into a base10 decimal number system. It is possible to convert it by hand or on your calculator. Take the word 'TEST' for example. This word has 4 'digits' in base26. To convert it to base10 we need to multiply each letter by 26(digit-1) then add them together. First convert the Letters to Base 10 numbers: 20, 5, 19, 20. Then compute the following expression;

*(20x263)+(5x262)+(19x261)+(20x260) = (351,520)+(3,380)+(494)+(20) = 355,414*

So this means that *A355414.* will produce the word *TEST.*

Converting in the other direction is significantly more difficult and is beyond the scope of this guide. There is an excel sheet for converting between the two number systems, it should be in the same folder you found this document in, named EngraveConvert.xlsm.

Also note that the numbers get significantly bigger with 5 letter words. Since most CNC inputs are limited to 999,999 the biggest word you can effectively command in one line is *BDWGM*. Meaning any 5-letter word that doesn’t start with an A, BA, BB, or BC is essentially going to be too big to program with a one liner.

**Compatibility**

This subprogram uses mostly local variables #1-#26 for the bulk of its computations, which are not persistent. It does access persistent variables #500-#503. It also has access to all of the persistent variables within the queues range. To learn more about the queue go its section in this guide.

**Character Subprogram**

**Main Purpose/ Usage**

The Character Subprogram is used to engrave the individual characters. The main Shape Program calls this subprogram to do the actual machining of the engraving. The characters are all hand designed and programmed in such a way that they are completely rotatable and scalable without any additional optional machine codes such as G68. The consequence of this is that the Character Subprogram is a fairly large file with hundreds of trigonometric functions. This program also has the ability to be used completely on its own if you wish to directly machine an individual character without using the whole Shape and Detail system or if you want to use the character set for a simpler engraving program, for more information on this see the Final Note below.

**The only configuration of the subprogram is as follows;**

*' ' Denotes Mandatory Inputs, All these inputs are not optional.*

G65 P80090 'CXYZAHFQ'

**Example of the Character Subprogram engraving a .25" letter T on a 30 Deg angle centered at X1.0 Y1.0 AND .005 deep.**

G65 P80090 C20. X1. Y1. Z.003 A30. H.25 F20. Q10.

**User Inputs**

All inputs need a decimal point.

**C = Character Code, Mandatory**

Number Representing the desired character, refer to Character Code Sheet below.

**X = X Origin, Mandatory**

X center point of the character.

**Y = Y Origin, Mandatory**

Y center point of the character.

**Z = Depth, Optional**

Depth to engrave the character.

**H = Letter Height, Optional**

Letter height is the height of each character. This will allow you to scale the characters to any size. Character width is half of H. If omitted Defaults to .125

**A= Rotation Angle, Optional**

Rotation of the character from its center point. If Omitted Defaults to 0.

**F = Cutting Feedrate, Optional**

If omitted defaults to 20.

**Q = Plunge Feedrate, Optional**

If omitted defaults to 10.

**Additional Notes on Inputs**

All Optional Inputs have default values and only need to be used for advanced functionality.

There are many built in errors that will alarm out the machine when invalid inputs are entered.

**Final Note**

Each character is programed from its center point. The first move is an absolute G90 move, going directly to start point of the character. Each character starts and finishes .01 thou from Z0. To program an individual character, you will need to call a tool and start the spindle before engraving, unlike the shape program which will has the startup built in.

**Compatibility**

This subprogram uses mostly local variables #1-#33 for the bulk of its computations, which are not persistent. It does not access persistent variables #500-#503 nor does it access to all of the persistent variables within the queues range. To learn more about the queue go its section in this guide.

**Error Codes**

**Detail Subprogram Errors**

**3001 (Missing Mandatory Input A, B, C or D)**

Detail subprograms can only use one A, B, C, or D input, they cannot be used together.

Break your detail into smaller pieces and separate the words, numbers, and symbols into their own details.

**3002 (Character Limit Reached)**

There are too many characters in the engraving. This can be fixed by increasing the memory the program is able to use.

This can be done by increasing the value in #503. Be careful that the expanded memory does not include variables other programs rely on.

**3003 (R cannot be greater than M)**

M sets the Max number for the SS detail. R sets the value it's Reset to when that Max number is reached or exceeded.

Having the Reset number greater than the Max number would cause an infinite loop. Rethink how the SS should work.

**Shape Subprograms Errors**

**3010 (DETAIL BLOCK MISSING)**

Mandatory B input is missing. This is the program number for the detail block of the engraving. Without a detail block the program doesn't know what to engrave.

**3011 (X MISSING)**

It is mandatory to have an X position. With linear engravings this is the X adjustment point. With arc engravings this is the X center of the arc.

**3012 (Y MISSING)**

It is mandatory to have a Y position. With linear engravings this is the Y adjustment point. With arc engravings this is the Y center of the arc.

**3013 (INVALID SHAPE)**

Currently only shapes that are supported is U1. for Linear Engravings, and U2. for Arc Engravings. Any other U value currently produces this error.

**3014 (INVALID ALIGNMENT)**

The only allowed values for J is 1.0 to 9.0 for more information on alignment codes see its section in this guide.

**3015 (NO DETAILS IN SUB)**

The detail block set with B didn't contain any valid Detail subprograms. Make sure you have the correct program number for the detail block or create a new one.

**3016 (ENGRAVING WILL OVERLAP)**

This error occurs when an arc engraving is too big and the total angle it takes up would be over 360. degrees.

Try reducing the character size (H) and/or the character spacing with (W).

**3017 (INVALID DIRECTION)**

Directions are either -1. clockwise or 1. counter clockwise.

**Troubleshooting**

**My engraving is not going to the coordinates I expected it to.**

This is most likely an issue with the O80089 (J) Value. Read up more on Alignment Codes.

If you’re sure the alignment code is right then check the work coordinate system your working in.

Another issue might be that you’re using the wrong Shape type (U). Read up more on Shape Types.

**The letters of my engraving or the spacing between the letters are to big.**

This is most certainly an issue with the O80089 (H), H will control the height of the characters. (W) will control the space between the letters, W is a factor of H so W.5 will make the space between the characters half the value in H.

**My serial numbers are blank the first time I run them, or the numbers are not what they're supposed to be.**

This is an issue with the variable D points to not being initialized before starting the first part. If D is set to D100. then variable #100 should be set to the number you want it to start with. If there is no value in #100 then the first part will be Null which will cause the program to ignore the whole serial number on the first run.

If the number is changing between parts, then there is another program that is interfering and using the variable you set for D. Try changing the D value to a different number that your positive is not being used.

**My machine doesn't have a big enough range of free variables to hold the engraving I want.**

This is the only major limitation to this program that doesn’t have an easy fix. If your machine has extra tool slots that it does not use you can use the appropriate #2000 range of system variables.

**The count increment (H) in my detail block is having an issue with rounding errors.**

Try using a fractional computation instead of an approximated decimal input. For example if you want the first 6 parts to have the same serial number instead of using H.166666 use H[1/6] instead, this should allow the machine to use the highest precision float number it can for the calculations.

If the issue persists you can highjack the increment system by setting H0. This will stop the subprogram from increasing the D value every time it's ran. Now you can set some logic inside the detail block itself for example;

*N1000(Engraving Detail Block)*

*IF[#101 EQ 6] THEN #100 = #100 + 1*

*IF[#101 EQ 6] THEN #101 = 0*

*G65 P80088 D100.*

*#101 = #101 + 1*

*M99*

This would have a similar effect to the previous fix and cause the engraving to increase its serial number once every 6 cycles, but has the benefit of not relying on an irrational decimal. #101 would be a extra counter that counts to 6 once it hits 6 it will increment #100 by 1 and cause the serial number to increase by one.

**Can I have the detail block directly inside my main program instead of being its own file?**

This is entirely possible and for most programmers is probably desirable! The Shape program has a variable (C) that is passed through when the detail block is called internally. You can use this variable as a jump condition when your main program is called as a subprogram.

Example;

%

O0001(My Main Program)

IF[#3EQ1000]GOTO1000

(Part Machining..)

G65 P80089 B0001 X0 Y-1. C1000. (B is referencing main program)

(Some more Machining..)

M30 (End of main Program)

*N1000(Engraving Detail Block)*

*G65 P80088 D100. (Serial Number)*

*M99*

*%*

In that example since #3(C) will normally = #0, which is a null variable, it's safe to have at the top of the main program. The Shape program calls the main program again, but this time as a subprogram with C set to 1000. This causes the program to jump forward to the Detail Block and update the queue as expected.

**What are some clever uses of the Pass-Through Variable C in the shape program?**

You can use it to partition up a program with several different common Detail Blocks then reference them as needed.

In this example you can call G65 P80089 B1000. C{1.-3.} to select a common detail block.

*%*

*O1000(Common Engravings)*

*GOTO#1(Jump to requested engraving)*

*N1(Three Place serial Number)*

*G65 P80088 D100. E3. K1. (-###)*

*M99*

*N2(Company Logo)*

*G65 P80088 A8853. (MBM)*

*M99*

*N3(Common Customer Name)*

*G65 P80088 A13083. (SIE)*

*G65 P80088 A8932. (MEN)*

*G65 P80088 A19. (S)*

*M99*

*%*

Another useful example is to have a detail block that references something like a job number that can be easily changed on the fly, from the top of a main part program.

In this example you can change the engraved job number right from the top of the main program by editing #100.

%

O0001(My Main Program)

#100 = 123456. (Edit for Job Number)

(Part Machining..)

G65 P80089 B1000. X0 Y1. C#100. (C is referencing the Variable at the top of the program)

(Some more Machining..)

M30 (End of Main Program)

%

*O1000(Variable Job Number Detail)*

*G65 P80088 B#1 (Job Number passed through from C)*

*M99*

*%*

You could also use the second Pass-Thru variable K to combine the two previous examples. By making C(#1) the jump address and K(#2) the Part Number.

**Can I add new characters to the Character Program?**

Yes, it’s not as complicated as it might appear. First all characters are designed to fit snug in a box that is 1in. tall and .5in. wide. It is not advised to make any characters exceeding that size. All characters are programmed to start from the center of this box.

All programmed moves need to follow this formula, replacing <X> and <Y> as needed with the incremental coordinates from the previous point, the first point is always center of the box.

*X[[#33\*<X>]+[-#32\*<Y>]] Y[[#32\*<X>]+[#33\*<Y>]]*

*I[[#33\*<X>]+[-#32\*<Y>]] J[[#32\*<X>]+[#33\*<Y>]]*

If any of the coordinates are 0, then delete the whole bracket containing the 0. Reduce brackets as needed.

*G01 X[[#33\*.1]+[-#32\*0]] becomes G01 X[#33\*.1]*

The N# is to be a unique character code, it can be referenced with a detail subprogram as C{N#-36}.

The first move needs to be a G90 rapid move with +#24 on the X and +#25

*G90 G00 X[[#33\*<X>]+[-#32\*<Y>]+#24] Y[[#32\*<X>]+[#33\*<Y>]+#25]*

Following that is always the same plunge move, G91 G01Z-[#26+.01] F#17

The rest of the letter is programmed incrementally from here.

If the tool is raised use G00 Z[#26+.01] followed by another plunge move.

The first move after a plunge needs a F#9 at the end of the line.

At the end of the character raise the tool again and finish with an M99.