Overview Of Documentation

This engraving system was created to fill in the gap left by CAM systems inability to create dynamic engravings, and the difficulty of making nice looking engravings without using a CAM system. Unlike most macro subroutines you might find in the modern machine shop, this one is packed with a substantial number of features that would make using it, to its full potential, very difficult to decipher without this document. In this document you will learn how this system works on both a theoretical and practical level. It will break down the fundamental concepts that made this system possible including, theories, diagrams and some of the formulas used. Detailed explanations of the subprograms and their inputs, complete with practical examples. A List of Error codes, their remedies, and a handful of other troubleshooting solutions.

Why Use This System

This engraving program is incredibly versatile, it was designed to be easy to use for both simple and complex engravings, while at the same time allowing for extreme control on positioning and output. This system will allow you to easily program engravings that tightly follow contours, edges, and fit neatly between different part features all without a requiring complicated CAM software. Furthermore, this system allows complete control of the engraving text and changes to the positioning directly from the machine controller. 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 limitations and dozens of customizable features and options to fine tune every aspect of how the text is engraved. This system will produce smooth engravings with true arcs that were designed and handcrafted specifically for this application, complete with over 60 characters.

Overview Of 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 manage the placement of the characters and call on a *Character* subprogram. The *Character* program will machine 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. 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 run 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* program will then determine its start point, using the *Queue,* then 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 length of 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.

Future, File Repository, and Reporting Bugs

Version 1.0 currently doesn't support a robust date and time detail. This will be added in a future update.

You can always find the latest up to date files for this system at "https://github.com/Metalsoul212/Engraving-Subprogram". If you find issues in the code, you can submit a bug report on the GitHub page by creating a new issue. You can also email me directly at steinjj9@yahoo.com. This system was created purely as a project of passion and I make no guarantees on future updates.

Table Of Contents

How To Use It

Creating The Detail Block

Before starting anything, it is critical to have all three subroutine programs loaded into your controller, these come in two flavors: Annotated and Reduced Memory. The Annotated version has comments and headers for the programs. If you wanted to look though the programs yourself to try and understand how it works, I encourage you to do so. The Reduced Memory version has all comments removed, along with extra spaces and any other unessential code. I recommend using the Reduced Memory programs for your machines as they will run slightly faster, and they require significantly less memory on your controller.

Next there are 2 persistent variables that need to be set on your machine #500 and #503, they determine what variables your Queue will use, for more information on how to set these variables and what they do, look at the Queue section on this guide.

Now you need to plan out what you want to engrave. Does it need serial numbers? Will it need to be separated into multiple lines? Do you need certain parts to be easily editable at the top of the part program by the operator? All serial/sequential (SS) need to have their own variable assigned to them. Anything that needs to be easily editable from the top of a program will also require their own unique variables, determine these variables ahead of time.

Packaged with this system I created an Excel workbook called "Engraving Helper", this workbook contains everything you need to efficiently create new engravings, including Cheat Sheets, a word to decimal converter, and most importantly the 'Engraving Programmer' sheet. Using the programmer sheet, you can very quickly generate the Detail Block you need to create any basic engraving in seconds. The only thing this sheet will not do is special SS, user editable, or any other type of dynamic engravings. It is up to you the programmer to add the dynamic elements to the engraving. The programmer sheet will also show you the size of your engraving and how many characters are in it.

Creating dynamic engravings is a straightforward process, for example we want to engrave a Job Number followed by a space and a Serial number with a dash and 3 digits, so that the finished engraving will look like this, **A53842-001**. We want every element of this engraving to be editable at the top of the program. In this case our entire engraving is dynamic. Comments enclosed with {} are not part of the program, they are an explanation of the above line.

Detail Block Example

```
N123(Engrave Job Number and Serial Number)
G65 P80088 A#100 (JOB LETTER)
G65 P80088 B#101 S1.  (JOB NUMBER)
G65 P80088 D102. K1. E3. (-###)
M99
```

Explanation

N123(Engrave Job Number and Serial Number)

{N block to reference this engraving in the shape program, also a comment explaining what it engraves}

G65 P80088 A#100 (JOB LETTER)

{'A' specifies that we want letters, #100 will be editable by the operator at the top of the program to change the Jobs Letter.}

G65 P80088 B#101 S1. (JOB NUMBER)

{'B' specifies that we want numbers, #101 will be editable by the operator at the top of the program to change the Jobs Letter. 'S1.' will add 1 space after the job number.}

G65 P80088 D102. K1. E3.

{'D' specifies that we want a serial number, #102 will be set by operator at the start of the job to set the starting serial number. 'K1.' will put a dash in front of the serial number. E3. will add zeros to the serial number so that it is always at least 3 digits long}

M99

{An M99 is critical for the routine to work correctly}

Creating The Shape Subprogram

Now that we have prepared a Detail Block, we can add this to the very end of our part program and begin working on Shape routine part of the program. The Shape routine will tell the machine how we want this detail block to be engraved. The Shape routine is the command you use to machine the engraving, by default the engraving will be centered at the given coordinates with an angle of zero, a .125" letter height, .003 deep, and the default shape is linear. In this example we are simply going to engrave the detail block we made previously right at the center of our work coordinate at G54 X0, Y0.

Complete Program Example

```
O1000(Simple Dynamic Engraving Program)
IF [ #1 EQ 123] GOTO123
/GOTO1 (Block Delete On Only For First Part)
#101= 1. (Job Letter, A=1, B=2...)
#100= 53842. (Job Number)
#102= 1. (Starting Serial Number)
G28 G91 Z0
T1 M06
G54 M08
G65 P80089 X0 Y0 B1000. C123. U1.
M30

N123(Engrave Job Number and Serial Number)
G65 P80088 A#100 (JOB LETTER)
G65 P80088 B#101 S1.  (JOB NUMBER)
G65 P80088 D102. K1. E3. (-###)
M99
```

Explanation

O1000(Simple Dynamic Engraving Program)

{Program name, since this program is O1000 we will use 1000 to reference this program later}

IF [#1 EQ 123] GOTO123

{When the shape routine is running it will call O1000 as a subprogram, we will tell it to pass 123. into variable #1, when the machine controller reads this line as a subprogram, it will jump ahead to the Detail Block at N123, otherwise it will do nothing.}

/GOTO1 (Block Delete On Only For First Part)
#101= 1. (Job Letter, A=1, B=2...)
#100= 53842. (Job Number)
#102= 1. (Starting Serial Number)

{This block of code will cause the program to only run this section of the code when block delete is on. This is where the machine operator would set the Engraving Data for the job they are running.}

N1 (Start Machining)
G28 G91 Z0
T1 M06
G54 M08

{Tool 1 is our engraving tool, and we want to use work offset G54 and machine with the coolant on. Notice we do not need to turn on the spindle or position the machine, all of this is managed entirely by the Shape Routine. You can set these parameters by adding <u>additional arguments</u> to the routine.}

G65 P80089 X0 Y0 B1000. C123. U1.

{X0 Y0 sets the origin of the engraving. 'B1000.' references the program that contains the Detail Block we want to engrave. 'C123.' will set 123. to variable 1# when the reference program is run. 'U1.' Specifies that we want a linear engraving.}

M30

The Queue

How The Queue Works

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 is a based off a common programming data structure called an array which is used extensively in higher level programming languages such as C+ and Java. An array is a basic programming paradigm that allows you to create a block of memory and reference distinct parts of it or order to read and write the values contained in it. To recreate this type of data structure in GCode we need to use 5 different variables to 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.

The other two variables are used to manage operations with the Queue. These two variables should NEVER be manually changed.
#501 is the Upper Bound, this is how many characters are currently in the Queue.
#502 is the Index, this is the variable the program is currently reading when it is steps though the queue.
The Upper bound will never exceed #503(Maximum Length).

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

Visualization Of The Queue Variables

Persistant Variables					
Variable	#500	#501	#502	#503	#504
Value	140	9	0	17	0.8

#500 is set to 140 to mark the start of the queue, #503 is set to 17 so that the queue will never use more then 17 characters. #501 says there are only 9 characters in this engraving. #502 says the program is currently looking at the first element in the Queue (Zero). #504 says this engraving is a total of .8 inches long when it is engraved.

Queue Variables Containing Character Codes																		
Variable	#140	#141	#142	#143	#144	#145	#146	#147	#148	#149	#150	#151	#152	#153	#154	#155	#156	#157
Value	5	14	7	18	1	22	9	14	7									

#500 is 140 so the first element of our Queue is #140. #503 is 17 so the maximum range of our Queue is #417 (140+17=417). The values contained in this engraving spells, 'ENGRAVING'. Notice that this engraving doesn't use all of the queues slots. Everytime a detail block is read it will only modify essential variables, and when the engraving is machined the Shape routine will only engrave the characters codes in the queue up to the number of slots determined by the value contained in #501. Since #501 is 9 it will only engrave up to #148 (140+9=148).

When the Shape program reads the Queue, it will start from the first position in the list which is #140(140+0=140). Then after that character is machined it increments #501 by 1 so it reads the next character at #141(140+1=141). It will do this until #501 matches #503 at which point the program knows it has reached the end of the Queue and all the characters have been engraved.

Shape Subprogram

Main Purpose/ Usage

The Shape program is what will be in your main program to machine the engraving. It is used to determine the placement, size, and shape of the engraving. Currently only line, arc, and linear rotary shapes 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 character code and other machining information to the Character Subprogram that machines the individual characters. There are many optional variables to fine tune the final engraving, however all of them have default values that will work for most engravings. Rotary type engravings currently only work on Haas Controls, or any control that has an equivalent to the Haas G107 cylindrical mapping.

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 Type Engravings
G65 P80089 'BXY' [UHJAWFSDQ]
Arc Type Engravings
G65 P80089 'BXYRE' [UHJAWFSDQ]
Rotary Type Engravings (Haas Control Only)
G65 P80089 'BXYRVE' [UHJAWFSDQ]

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

G65 P80089 B1000. X0. Y0. R1.9 U2. A90. J2.

Shape Subprogram 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, 1. for a linear shaped engraving, 2. for an arc shaped engraving, or 3 for linear rotary shaped engravings (Haas Only).
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 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.
In the case of a rotary engraving this is the A axis alignment point.
If omitted, it Defaults to 0.

E = Rotation direction, Optional

For Arc Type Engravings

E represents the direction the engraving faces.
1. for CCW, making the engraving face the inside the arc.
-1. for CW, making the engraving face the outside the arc.

For Rotary Type Engravings

E represents the direction the Rotary Indexer is pointing towards.
±1. for ±X
±2. for ±Y

If omitted defaults to 1.

V = Axis Mapping Direction, Optional

This determines which direction the engraving faces for Rotary Engravings.
1. Faces away from the indexer
2. Faces towards the indexer
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 for 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 for extra logic in detail blocks. Referenced with #2.

Additional Notes on Inputs

C and K will set variables #1 and #2 respectively when the Detail Subprogram is called.
R is only mandatory for arc engravings and will have no effect with linear engravings.
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.

Alignment Codes

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. There are 9 points that matter,

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 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. There is also a folder in the main directory with a set of images showing exactly how each alignment code effects the position with a variety of shapes and alignment codes.

Detail Subprogram

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.
The Detail subprogram forms a new type of GCode that allows you to 'Program' the engraving text. The Detail Subprograms work by decoding its inputs into individual character codes and adds them to the queue.
Numbers are separated into their constituent digits including decimal and their sign. Letters are decoded from decimal format into a constituent base26 number system. It also manages and queues dynamic serial number and sequential letter inputs. This system requires you to make a Detail Block to define the engraving text that the Shape subprogram can use.
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 a method that will allow you to make a detail block into a type of local subprogram, to learn more see this section.

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. (A)
G65 P80088 D100. K1. E3. (Serial Number)
M99

Detail Subprogram User Inputs

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

All inputs need a decimal point.

A = Alphabetical Input

Letters/Words are represented as numbers, A=1 - Z=26... AA=27, AB=28...
*See **Converting Words into Decimal Format** 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...'

K = Dash/ Pound Prefix, Optional

When this is set to 1 a Dash will be added before the SS.
When this is set to 2 a Pound symbol will be added before the SS.
The K value is indexed to the Character Codes, any number other than 1 or 2 will produce the character code equal to K+37.
For Example, B2. K1. E3. Will output '-002', B2.891 K18. Will output 'Ø2.891'

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 **Converting Words into Decimal Format** 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.

Converting Words Into Decimal Format

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. 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 ‘TEST’. To convert it to base10 we need to multiply each letter by $26^{(digit-1)}$ then add them together. First convert the Letters into Base10 numbers where A=1, B=2… Z=26 20, 5, 19, 20. Then compute the following expression:

$$(20x26^3)+(5x26^2)+(19x26^1)+(20x26^0) = (351,520)+(3,380)+(494)+(20) = 355,414$$

This means that 355414. will produce the word *TEST* when the detail subroutine decodes the number.
Converting in the other direction is significantly more difficult. In short, the decoder first determines the digits, then recursively calculates the powers starting with the highest digit and works its way down. This process can cause the machine to stall for a couple seconds since the computation is intensive.
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 one line.

Character Code Sheet

This is a list of all currently supported Characters, their respective Character codes, and the Detail Variable that would command that individual character. The detail subprograms will decode its inputs and update the queue with the character codes as it is in this sheet. There is also a cleaner version of this Character Code Sheet in the 'Engraving Helper.xlms' workbook that is included with this system.

Detail Variable	Character Code	Character		Detail Variable	Character Code	Character		Detail Variable	Character Code	Character
S1. Or A0.	0	{SPACE}		A21.	21	U		C6.	42	)
A1.	1	A		A22.	22	V		C7.	43	[
A2.	2	B		A23.	23	W		C8.	44	]
A3.	3	C		A24.	24	X		C9.	45	<
A4.	4	D		A25.	25	Y		C10.	46	>
A5.	5	E		A26.	26	Z		C11.	47	=
A6.	6	F		B0.	27	0		C12.	48	/
A7.	7	G		B1.	28	1		C13.	49	,
A8.	8	H		B2.	29	2		C14.	50	'
A9.	9	I		B3.	30	3		C15.	51	"
A10.	10	J		B4.	31	4		C16.	52	%
A11.	11	K		B5.	32	5		C17.	53	°
A12.	12	L		B6.	33	6		C18.	54	@
A13.	13	M		B7.	34	7		C19.	55	Ø
A14.	14	N		B8.	35	8		C20.	56	$
A15.	15	O		B9.	36	9		C21.	57	!
A16.	16	P		C1.	37	.		C22.	58	?
A17.	17	Q		C2.	38	-		C23.	57	!
A18.	18	R		C3.	39	#		C24.	58	:
A19.	19	S		C4.	40	+		C25.	57	±
A20.	20	T		C5.	41	(				

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. This subprogram is not intended to be used on its own, but if you wanted to use them all of the inputs are mandatory, and the machine must be positioned Z+.01 above the center of the character.

The only configuration of the subprogram is as follows:
' ' Denotes Mandatory Inputs, all of 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.005 A20. H.25

Character Subprogram Inputs

All inputs need a decimal point.

C = Character Code, Mandatory
: Number Representing the desired character, refer to Character Code Sheet above.

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.

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. For clockwise or 1. For 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 are sure the alignment code is right, then check the work coordinate system your working in.
Another issue might be that you are using the wrong Shape type (U). Read up more on Shape Types.

The letters of my engraving or the spacing between the letters are too 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 quarter of the value in H.

My serial numbers are blank the first time I run them, or the numbers are not what they are 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 variable that your positive is not being used.

My machine does not have a big enough range of free variables to hold the engraving I want.

This is the only major limitation to this program that does not 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. Alternatively, you can try to break your engraving into smaller blocks of text.

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 is 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 an 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 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.

```
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 is 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 and K 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)
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 Subprogram?

Yes, it is not as complicated as it might appear. First all characters are designed to fit within 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 incremental center of the box which is X<distance>+#24 Y<distance>+#25.

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(X#24, Y#25 is the center of the box)

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 character is programmed incrementally from here.
If the tool is raised use G00 Z[#26+.01] followed by a rapid move to the next point and then 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.