

Mastercam X Class

Machine Definition
Control Definition
&
Post Modifications

Sections

1. **Overview:** In this section we will briefly go over the files used in Mastercam X for controlling output and machine settings.
Pg: 3
2. **Machine Definition:** What is the machine definition, how does it effect Mastercam, How do I configure the machine definition correctly.
Pg: 4-11
3. **Control Definition:** What is the control definition, how does it effect Mastercam, How do I configure the control definition correctly.
Pg: 12-17
4. **Post Processor Intro:** What is the post processor, How does it work with the machine and control definitions to change program format.
Pg: 18
5. **Post Processors-1:** General overview of the parts of a post processor. Explaining: Variables, Postblocks, Post Switches
Pg: 19-20
6. **Post Processor-2:** Details about Variable Definitions and different methods of defining variables
Pg: 21
7. **Post Processor-3:** Details about Format Assignments. How to format variables and how to change formatting during posting
Pg: 22-24
8. **Post Processor-4:** Details about String Selector Tables. How they work, How to change existing selector tables.
Pg: 25-26
9. **Post Processor-5:** Details about Condition Statements, What they are, How to use them in practical posting.
Pg: 27-28
10. **Post Processor-6:** About the NCI File.
Pg: 29

1 – Overview

Mastercam X Uses 3 main files to control the output of the gcode program.
These files are the: Machine Definition (.mmd, .lmd, .rmd, or .wmd)
Control Definition (.control)
Post Processor (.pst)

These three files are used in conjunction with one another to configure Mastercam and it's options to allow you to program for your machine type, along with formatting the output of the gcode to allow the program to then run correctly and accurately on your machine tool .

The machine definition is a file that contains information about your machine. The machine def is used to determine what type of machine you are using, mill, lathe, router, wire, etc. In the machine def. you will setup what axes your machine has along with other parameters about the physical components of your machine tool. The machine def is also the file that selects what control definition file that machine will use, and also what control definition will be used from the selected control file.

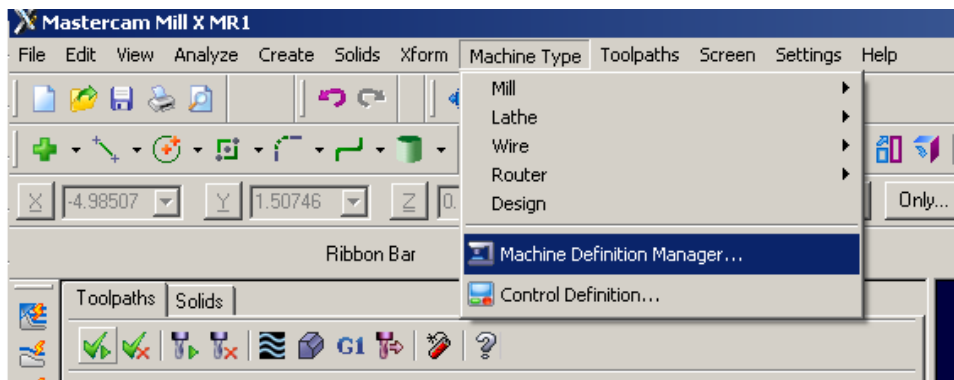
The control definition is stored in a .control file. The .control file can contain 1 or more control definitions. These control definitions are what controls what post processor will be use, along with options like how the arc moves and drill cycles will be output. The control definition holds a large amount of information about your specific program format that is required in order for the generated gcode program to run on your machine tool.

The post processor file is the file that has the largest control over the format of the generated gcode program. This file not only determines what gcodes are output but also what order they are output in the generated program. The post processor file is a logic based code file that can be modified to suit just about any requirement you may have. A post file is made up of variables, postblocks and condition statements to determine when and where things happen in your programs. We will get into more details about the post processor files as this guide continues.

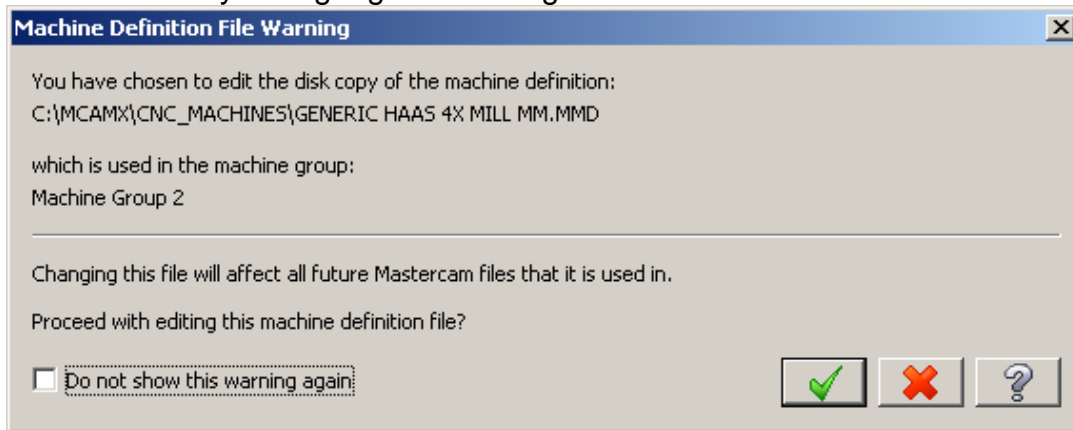
2– Machine Definition

In this section we will go over the machine definition and how we configure it. For the purpose of this document we will be creating a machine definition for a Haas 3 axis vertical mill with a 4th axis rotary option. Before we get too far ahead of ourselves we need to go into a generic machine definition that we will then use to create the machine definition we need.

1st select the “Machine Type” menu select the “Machine Definition Manager” option.(shown below)



Once selected you might get a message box that looks like the one below.



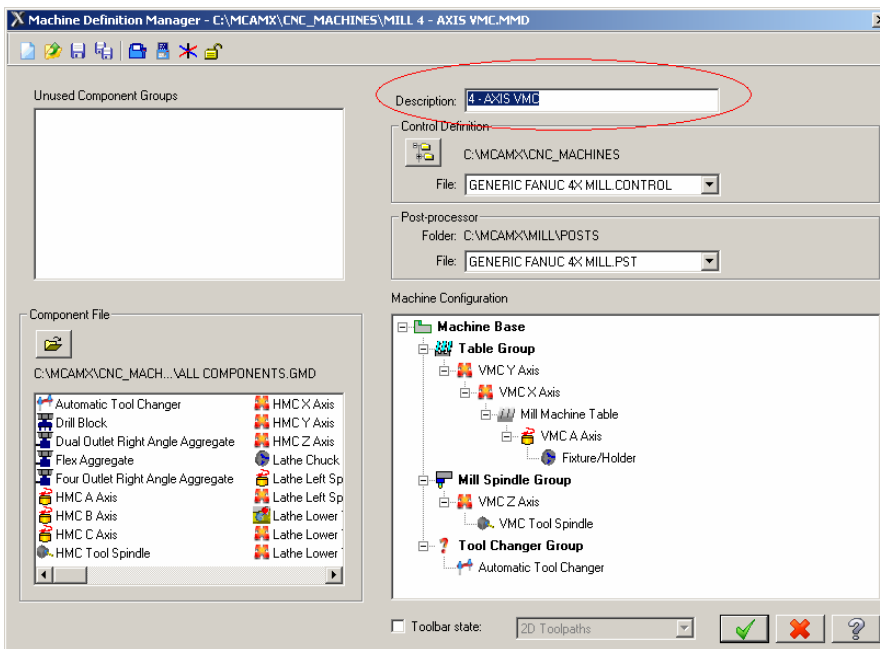
This message is simply stating that you are going to go into the “Disk Copy” of the machine definition. It will then have a line that says what actual machine definition it is opening. At this point we are not concerned about what machine definition it is opening because we are going to create our own. Select the green check mark in the lower right corner of this window to continue into the Machine Definition Manager.

2 – Machine Definition – Continued

Now the Machine Definition window will open. In this window we will be setting up options that relate to our machine.


Naming the Machine Definition


The 1st thing we want to do is to change the name that is displayed in the operations manager for this machine. This is done in the “Description field” shown below. This field is in the upper right hand corner of the machine definition manager window.



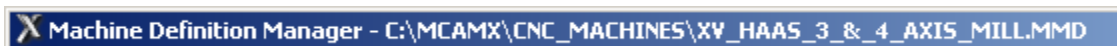
In the “Description” field we need to enter the name “Xv_Haas_3_&_4_Axis_Mill

Once we have entered the description we need to save the machine definition with the new name of our machine.

The  icon stands for “Save As” This is the option we want to use to save the machine def with our new machine name.

Select the  Icon and when the Save dialog box opens enter the name “Xv_Haas_3_&_4_Axis Mill” then press the “Save” button in the lower left. We have now saved our own copy of a machine definition, with a new name.

You can verify the name of a currently opened machine definition by looking at the blue bar at the top of the page. (sample below)



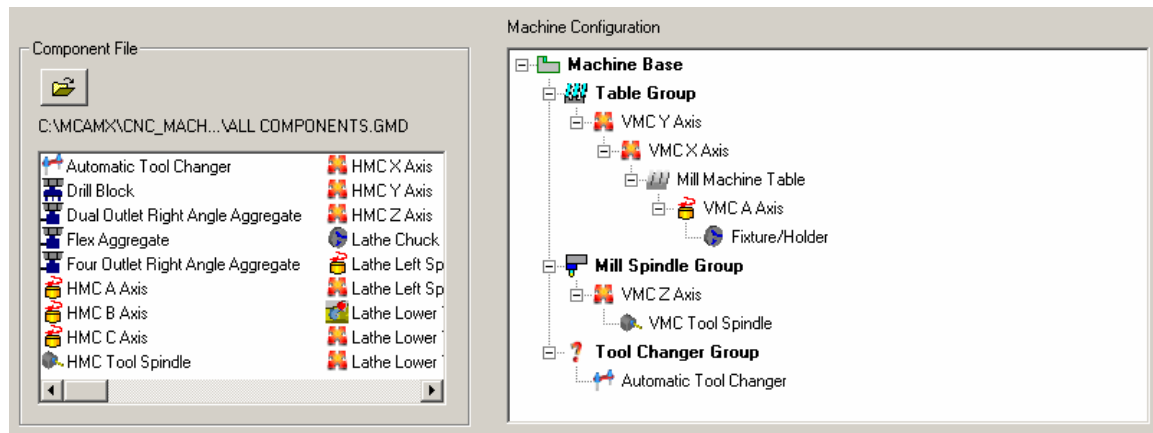
(As you can see in the example above it shows that the Xv_Haas_3_&_4_Axis_Mill.mmd file is open.)

2 – Machine Definition – Continued

Building the Axes of Our Machine

The lower portion of the Machine definition is for defining what axes our machine has. This will determine what our machine is able to cut. For example if you do not have a rotary axis defined for your machine then later when trying to create toolpaths you will not have rotary axis toolpaths available.

Here is a image of the Machine Configuration section.



On the left of the above image you can see where you select different “Component Files”. Component files are files that contain machine components that are associated with a particular type of machine, eg: vmc, hmc, vtl, etc. By default the “All Components” file is selected and the display window just below show components for all types of machine tools.

On the right of the above image is where we actually setup our machine. Now we opened an existing machine which is similar to the machine we are working on so we have most of this work done for us. In this document I am not going to go step by step through building machines but I will note important facts about how you must build your machine.

Notes on building Machines: When you are building your machine you must build things in what is referred to as Logical Order. In other words with a standard type VMC you will have:

- A Machine Base
- A Table Group
- A Spindle Group
- A Tool Changer

Now in each of these groups there are rules that apply. You want to build your machine the same way the physical machine is set up.

2– Machine Definition – Continued

For example. In the Machine Base group you have a Y axis. Now looking at your physical machine you can see that the X axis is physically mounted onto the Y axis, so in your machine build you will have the X axis mounted off of the Y axis like shown above.


Now on your actual machine you have a table mounted to your X axis so in here you will have a Mill Machine Table mounted to your X axis.

If your machine has a rotary axis the following method still applies. When you put your rotary axis on your physical machine tool you mount it to your table. You must do the same thing here.

Now you don't bolt your part to the rotary by itself, you mount a fixture plate or rotary chuck to the rotary axis and then hold your part with that. So here we put a Rotary Fixture Holder.

Why is this so important? This is important because the next thing we need to do is define what are called as "Axis Combinations".

Defining Axis Combinations

To get into the axis combinations page click on the  Icon at the top of your machine definition page. Axis combinations are combinations of axes that will be used together in a program to cut the part.

In order to create valid axis combinations we must have what are referred to as Axis Terminators. An axis terminator is:

- Mill Machine Table
- Rotary Chuck
- Fixture Holder
- Spindle
- Etc

Without an valid axis terminator you will not be able to use your machine definition. When the axis combinations page opens we will see that one combination has already been created. This is because we are using an existing machine def that we saved off to a new name to create our own.

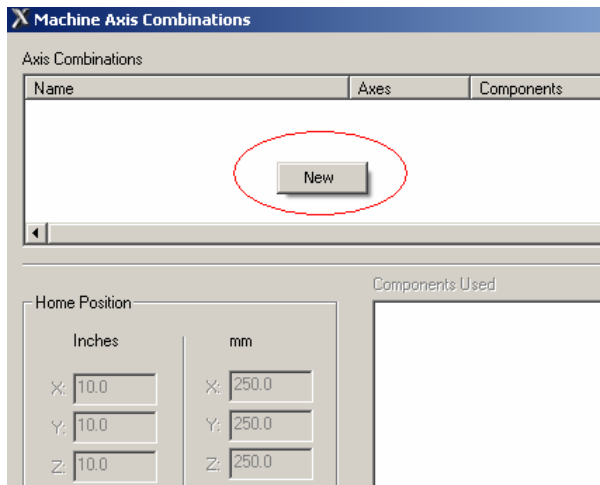
If you were starting from a new machine definition you would not have any axis combinations.

2 – Machine Definition – Continued

We are going to delete the existing axis combination so we can create our own. To do this “right click” on the axis combination that is named “Default(1)”. Select the delete option from the drop down menu.

Now we have a blank page that we can create our new axis combos.

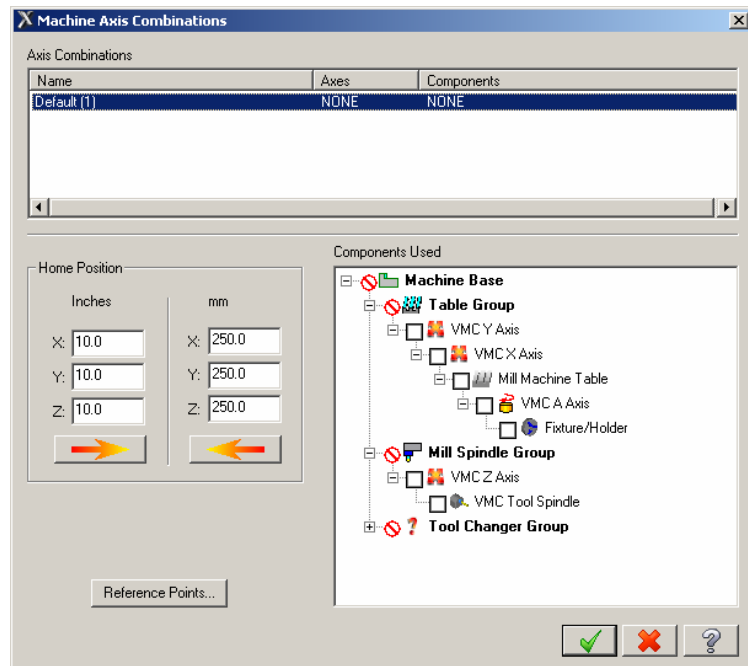
To create a new axis combo “right click” and in the white area shown in the image below and select “new”.



When we select “new” we will have a new axis combination that we can then configure.

With the new axis combination selected we need to select the components that will be used in this axis combo from the “Components Used” section.

(This is an image below is of the axis combinations page after we have added a new axis combination. We have not yet selected the components that will be used with this axis combo. We also have not yet given it a name)



2 – Machine Definition – Continued

The 1st axis combo we want to create is for a 3 axis mill. To do this we simply select all of the components used for a 3 axis mill.

Select: Y Axis
X Axis
Mill Machine Table (This is the terminator for this group)
Z Axis
Tool Spindle (This is the terminator for this group)

Now we want to “right click” on the axis combo we just created and choose the “rename” option. Name this: “3 – Axis”. We have now created the 3 axis component. We will now move on to creating the 4 axis combo.

We need to “right click” and create another new axis combo. This time we are going to select the following components.

Select: Y Axis
X Axis
A Axis
Fixture Holder (This is the terminator for this group)
Z Axis
Tool Spindle (This is the terminator for this group)

Note that this time we did not select the “Mill Table”. The mill table would be an axis terminator and we would not be able then to add the rotary. By not selecting the “Mill Table” it enables us to select the A Axis and Fixture Holder. The Fixture Holder has now become the axis terminator. “Right Click” on the new axis combo we just created and change the name to “4 – Axis”.

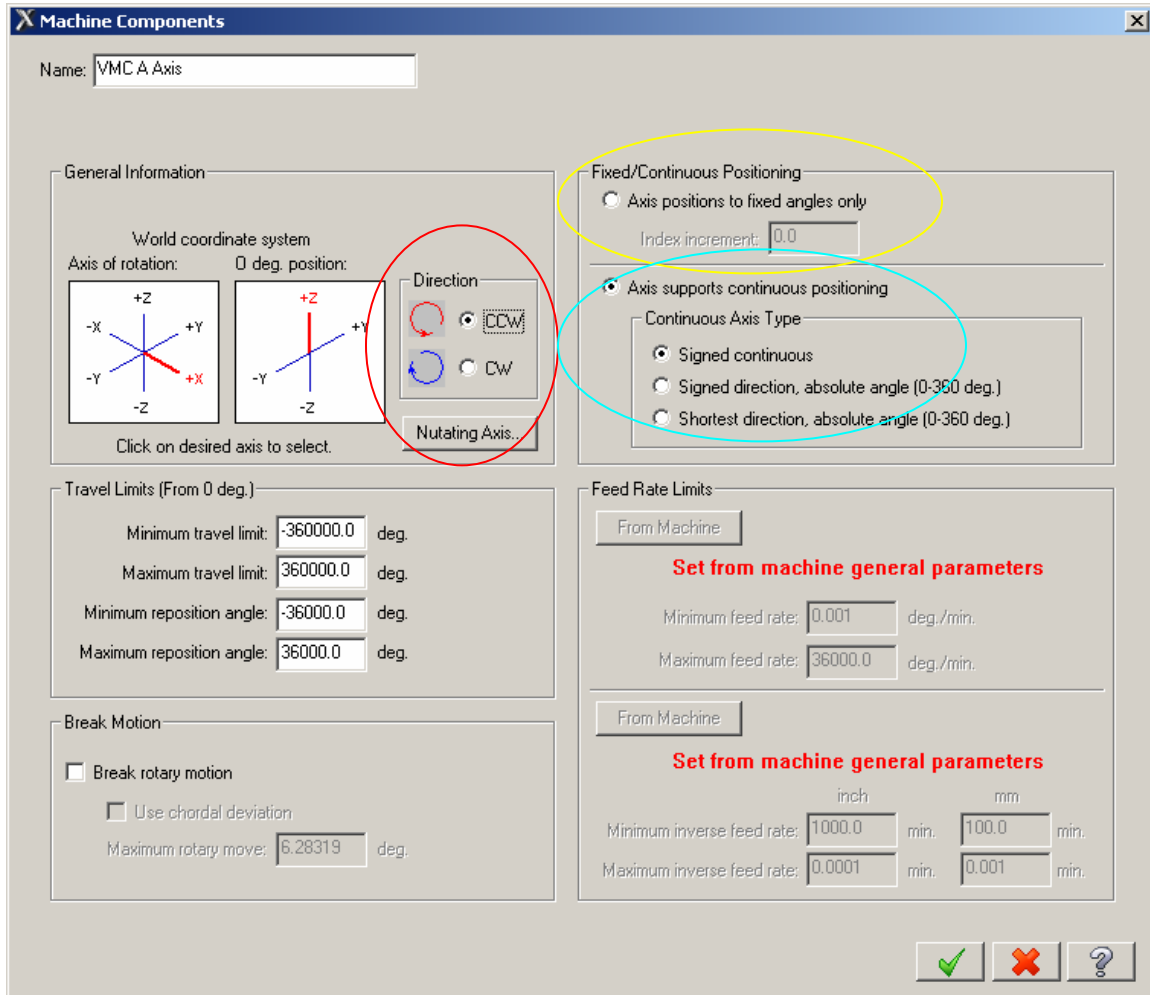
Also note that on this page you can define the machines “Home Position”. The values you enter on this page will be used by your post if you have the options in your control definition set to read the value from the machine definition. We will get into that more later.

We can now select the green check in the lower right corner to accept the changes and return to the machine definition manager.

2 – Machine Definition – Continued

Configuring Specific Axis Parameters

In the machine definition we can configure specific axis and component parameters by double clicking on that object in the machine configuration window. For example I am going to click on the A Axis and go into the options for this component as seen below.




For this axis we can specify the direction(1) of rotation, Whether or not this rotary is positioning(2) only for full rotary, and also the type(3) if a continuous axis is defined.

This is just an example of the options that can be found and configured for your machine components. Once you have made modifications to these components I recommend saving your machine definition.

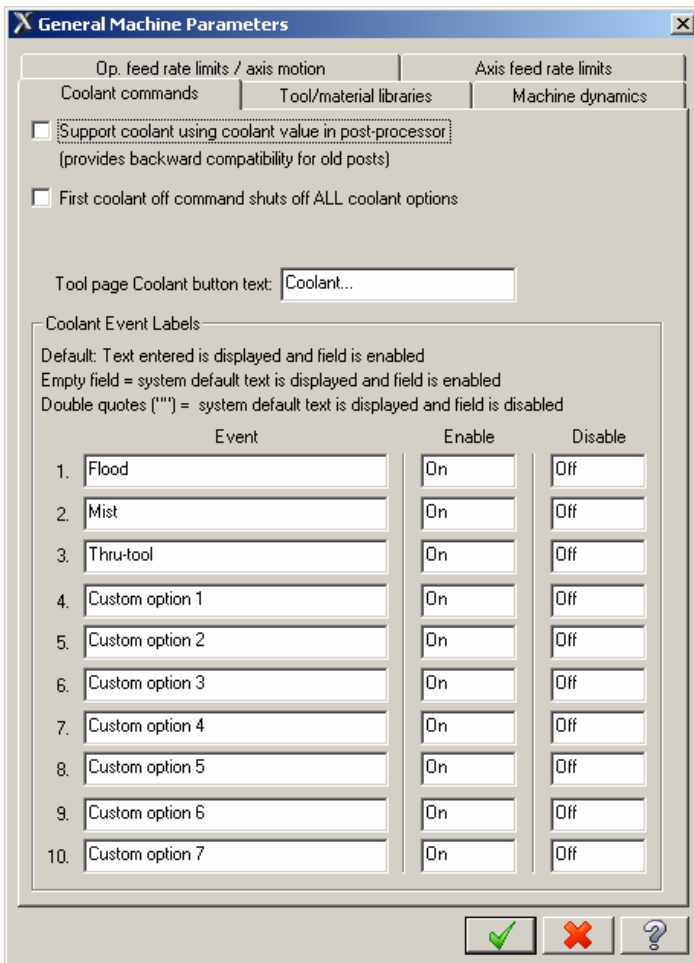
2 – Machine Definition – Continued

General Machine Parameters

The next thing we need to do is edit the General Machine Parameters. You get into this page by clicking on the  icon at the top left the machine definition page.

In the general machine parameters page you will set coolant support options and also parameters for axis feed rates. There is also a tab for selecting tool and material libraries that this machine definition should use. We are not going to cover everything in this page but we are going to focus on the coolant options.

Select the “Coolant Commands” tab at the top of the page. You will then see the following page.



On this page you have an option for supporting coolant values in the post processor. If you enable this option then you can use multiple coolant on commands in the post but you can only have one coolant off command..


If you choose not to use the post processor based coolant method then you will be able to have different on and off codes for each different coolant type. This method uses the Canned Text option in Mastercam to output your coolant. This is a method that Mastercam has implemented to aid in supporting newer styles of coolant but this might change in the near future. Use this option knowing that you might have to change it in the future if they decide to change posting methods.

By changing the text that is shown in the fields you can alter the options that are seen when selecting coolant in the tool path operations themselves.

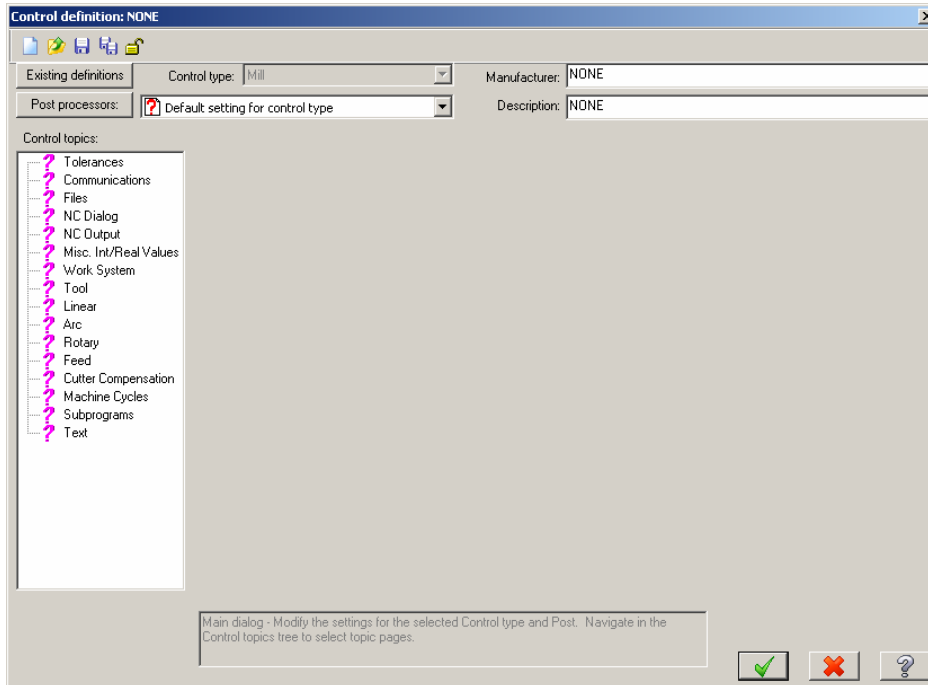
3– Control Definition

In this section we will go over the control definition and what some of the settings are that can be found in the control definition.

The control definition is information that is stored in the .control file. You can get into the control definition two ways.

- 1 - Machine type menu, Control Definition
- 2 – Open the Machine definition and select the  Icon.

Once the Control definition is open you will see the following screen



This is the Control Definition Manager window. This is where you Edit, Create, and delete control definitions that are within .control files.

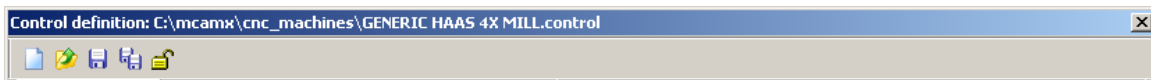
It is very important that you know how to correctly associate the control definition with a machine definition and also what happens when you select to add a post on this page. In the machine definition you will select what control file and what control definition you wish to use...but here you select define what post the control definition will use.

3– Control Definition – Continued

Lets get started by explaining how this all works, then we will continue on with creating our control definition.

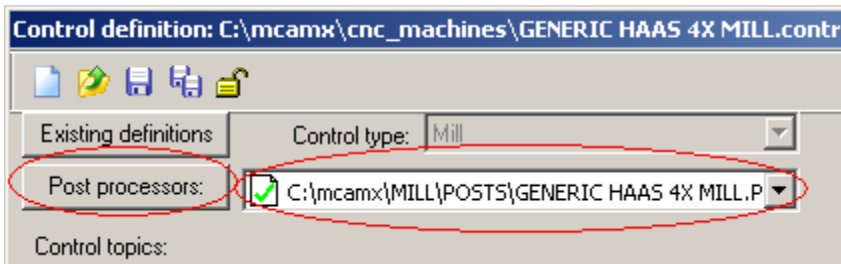
When you open the window you saw on the previous page you are in the control definition manager. The control definition manager shows you control definition files.

Now to see what control definition file (.control) you are in you can look at the top blue ribbon bar.



Now each .control file can have more than one control definition in it. In fact you can have one .control file that will hold all the control definition files for all the mills you have. This can get confusing so we are not going to get into this right now but you will see how this is possible as we go along.

Starting off we look at the dropdown box just to the right of the “Post Processor” button.



Note: The two red circles show the button and dropdown box.

In this dropdown box we can see all the control definitions that are saved in this control file.

Note that all control files will have a “Default Control Definition” that is saved in them and you cannot delete this.

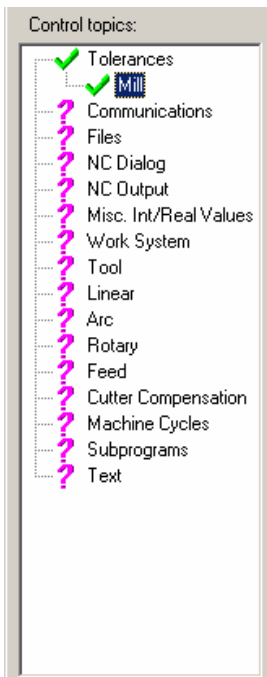
Now in the dropdown box it shows a directory path and post processor name. This shows the post that will be used and where it is stored. If the post is not in that location or the name has been changed you will get a red X over the icon to the left of the path. This means that your file is invalid.

3– Control Definition – Continued

Add Additional Post Processors

To add an additional post processor you simply click on the Post Processor button. This not only allows you to select a different post but when a new post is selected it creates a whole new control definition inside the current .control file.

In the control definition manager there is an area to the left of the window called “Control Topics” This is where the settings and parameters go for each control definition.



As you select different control topics you will see the options appear to the right of the Control Topics box. Please note that these settings are related to the control definition that is being shown in the drop down list that is beside the post processor button. If you decide to change to a different control def in the drop down list you must save the .control file in the upper toolbar or you will lose your changes.

In these control topics you can set things like how your arc moves are output in the gcode program. Eg: IJK output or R output.

You can control sequence number output and many, many other options. We are not going to cover all of these options here as you can explore them.

Re-naming, or Changing the location of a post processor


If you wish to re-name a post processor or simply change the location that the post processor is stored on your computer or network you must realize that when you go into the control definition and select the “Post Processor” button and then choose to “Add” your post you will end up with a “New Blank” control definition. So any options you had defined for your post previously will be lost unless you follow these steps.

- 1 – Open your existing control definition
- 2 - select the “Post Processor” button.
- 3 - In the Control Definition Post List Edit box choose the “Add” button

3 – Control Definition – Continued

- 4 - Select your post processor
- 5 - Select the green checkmark in the lower right side of the screen to go back to the control definition page.
- 6 - Now in the drop down box just to the right of the post processor button select the post that you just added.
- 7 - In the control topics menu select any of the control topics. (I choose NC Output)
- 8 - off to the right of the control topics, where the parameters are displayed for that options “Right Mouse” click and select “Import” then “All Pages”.
- 9 - you will then be prompted for a .control file. Select your .control file.
- 10 - you will then be shown a list of all saved control definitions in that control file. Select the definition that has the correct parameters in it for your machine.
- 11 - Now save your control def.
- 12 - You now can go back into the post processor button and select and delete the control definition that has the invalid post name or path.

Setting up our Control Definition

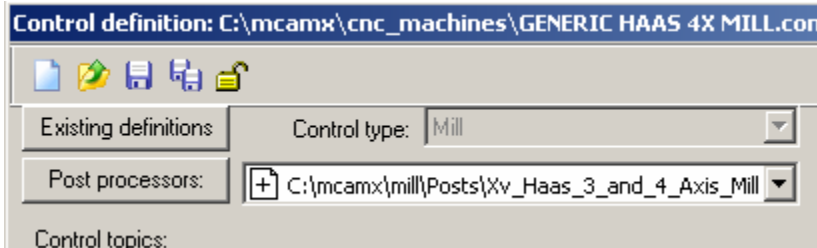
Now that we have a better understanding of how all this control definition stuff works we can move on and create our control definition. Open the control definition manager from our machine definition and select the “New Control Definition”,  option.

We now have a completely blank control definition to work from. Now I would like you to go through windows explorer and go into the c:\Mcamx\mill\posts directory and make a copy of the Generic Fanuc 4X Mill.pst and name it Xv_Haas_3_&_4_Axis.pst.

Back in the control definition I would like you to add this post processor to our blank control def by selecting the “Post Processor” button, and then in the next window selecting the “Add” button, then choosing our Xv_Haas_3_&_4_Axis.pst.

3– Control Definition – Continued

Now the back in the control def select our Xv_Haas_3_&_4_Axis.pst. from the drop down list next to the “Post Processor” button.



Notice how the icon next to the path in the drop down list has a “+” in it. That means that the file is there and valid but it has not been saved into this .control file yet. We must now select the save option to go ahead and save the file into our control def. This will put a green check mark where the “+” was.

Importing the parameters from a Generic Haas Control Def

Now we have our post added we need to setup our parameters for this machine. Mastercam X comes with a default machine for a Haas mill and it works pretty good. We are going to import the control parameter from there default machine but if you have a machine where no defaults are available then you would just go through each of the parameter pages and set the parameters according to what your machine requires.

To import the parameters from the default Haas machine we need to select any of the control topics. Then simply right mouse click in the grey area where the options are displayed and select “Import” and then “All Pages” from the drop down menu that appears.

You will then be prompted to select a .control file. Choose the file “GENERIC HAAS 4X MILL.control”. You will then be prompted to choose what control def you wish to import the parameters from. Select the one with c:\mcamx\cnc_machines\Haas 4X Mill.pst

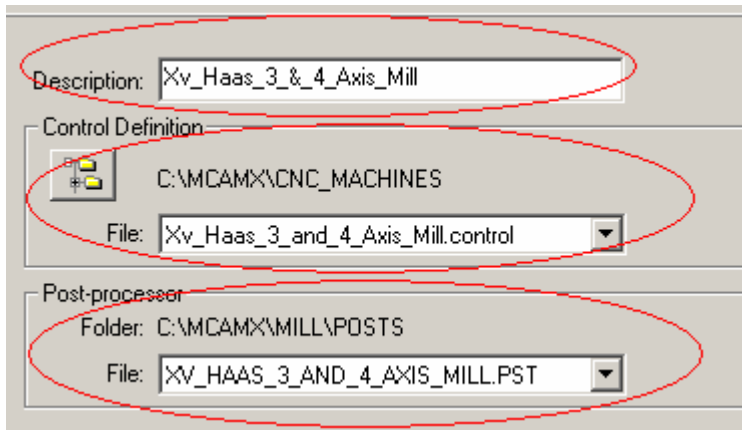
Accept all boxes until you get back to the control definition page.

We have now imported the parameters from the Generic Haas definition into our control definition.

3 – Control Definition – Continued

We can add comments and notes about the control type in the upper right side of the control definition window, and then save our control definition.

Going back to our machine definition you should see the following in the circled fields.



Note: if you used different file names or data paths then you should see the names and paths that you used in these fields. You also can select them from the drop down list if they are not shown.

We have now created and setup our machine definition and we are now ready to start setting up our post processor to modify the format of the gcode program.

Please note that some things in the program format will be effected by the control definitions control topics

4– Post Processor Intro

The post processor is the file that processed by Mastercam to format the gcode that is output when a file is post processed. The post file has a .pst file extension and by default is stored in the c:\mcamx\mill\posts directory. This directory will vary based on what type of machine you are using. Eg if you are using a lathe then the path would be c:\mcamx\lathe\posts\.

The post processor file is just a txt file with a .pst file extension so you can open it in just about any editor. One thing to keep in mind is you do not want to use any editors that might apply formatting to the post as they will make the post file unusable. This means don't use Microsoft Word and other programs like it. You can use and I recommend using the editors that come with Mastercam to do your post editing.

When you decide to generate the gcode program from a given MCX file Mastercam does the following (Not necessarily in this order)

- Creates the NCI file
- Looks at the machine definition to determine settings
- Looks at the control definition to determine settings
- Opens the post processor
- Runs mp.dll using the files listed above

Then using the information it gets from the steps listed above it uses the machine def., control def., and post processor to start generating code. Based on the settings in the machine and control definitions settings are configured in the post processor and in mp.dll. These settings will affect some of the aspects as to how the program format will look.

All of the different settings that are effected by the machine and control definitions are outside the scope of this guide and will not be covered entirely here. Some things will be mentioned in the following sections as they come up in our project.

5– Post Processors-1

Before I get started in the following sections I would like to explain how I intend to present some of the information to you. When I am putting example post code in this document I will surround the text with a box. Example below.

```
This is my example
```

Whenever you see text in a box like above, it will be code from the post.

Parts of a Post Processor

The post processor is made up of many parts. There are comments, variables, strings, selector tables, format statements, postblocks, and more.

I am going to start at the top of a post processor and work my way down.

Comments & Headers – Comments are output all over a post processor and are used to store notes that the post editor will use and need to remind him/her of what they were doing in a particular piece of code.

A comment line is a line that starts with a # sign. Example below

```
# Post Name      : Xv_Haas_3_and_4_Axis_Mill.pst
```

The # sign tells Mastercam to ignore anything to the right of it on that line.

Comments are used to output the post header information which is useful in telling about what options the post processor has and also how things have been implemented. Example post header below

```
# Post Name      : Xv_Haas_3_and_4_Axis_Mill.pst
# Product        : Mill
# Machine Name   : Haas
# Control Name   : Hass
# Description     : Generic 4 Axis Mill Post
# 4-axis/Axis subs. : Yes
# 5-axis         : No
# Subprograms    : Yes
# Executable     : MP 10.0
```

5– Post Processors-1 Continued

Using this header information we can tell that this post is for a Haas mill. This post supports 4 axis, axis substitution, and subprograms. We can tell that this post also is setup for Mastercam MP Version 10.

It is very good in post editing to make a habit of creating good comments about the work you are doing. It will make things easier when future work is done to the post processor. Near the top of almost all post processors is a revision log. Please update this revision log each time you make changes to the post to aid others that might make edits in the future.

Variables – Variables are used to hold data in the post processor. A variable can hold a number or a string(text). Variables can be used to output the number or string that is stored in it...or it can be tested against to determine if or when something should be output. We will get more into variables in the next section.

Postblocks – Postblocks are sections of code that are defined with a name that is in column position 1 in a post processor. Postblocks contain the code that produces the gcode output. When a postblock is called Mastercam processes the code that is listed under that postblock and generates any output to the NC program that is defined in that postblock. Postblocks typically start with the letter “p” but can also start with the letters “l” and “m”.

Postlines – Postlines are the lines of code that are in postblocks that test and output things from a postblock. Postlines can start in any column position except for the 1st column position. All code listed under a postblock is part of the proceeding postblock until another postblock label is encountered.

String Literals – String literals are characters that are in “ “ statements. Any text or codes you put into “ “ statements will be output as literal code in the program. Example:

n\$, “output this”, e\$

For now ignore the n\$ and the e\$ as we will be getting to those after a while.

The Gcode program would look like:

N100 output this

You can see where this can be used to output some code that you need to be in your program. Note that this is forced out so anytime that postline is processed it will be output.

Post Switches – Post switches are variables that are set at the top of the post that will be used either in mp.dll or later in the post to alter output.

6– Post Processors-2

Variable Definitions

We briefly went over what variables we in section 5. We now are going to go into variable and how to work with them in more detail.

To start, think of a variable as a blank line of paper. We will say that line is called “var1”. Now anything that we write on that line of paper is stored in the variable called “var1”. So lets say we write the number 2 on that line. If we then output the variable “var1” we would get the value of 2.

Example of output in program:

```
var1 2.
```

Now the reason the output has “var1” in there is because this variable is not formatted.

There are two main types of variable in Mastercam posts. Pre-defined variables, which are variables that are defined in mp.dll. User-defined variables, which are variables that are defined in the post processor file.

Basic Variable Initialization

Now to be able to use a user defined variable it must 1st be defined in the post. There are several different ways to define a variable but for now we are going to cover basic variable initialization. We do this by creating the variable with a default value. Ex:

```
var1 : 0
```

This statement will initialize the variable var1 with a value of zero or “0”. We could initialize this variable with any numeric value we wanted just by changing the value after the “:”.

You can also create a variable and format it all at once using format statements. Format statements allow you to control the decimal output along with any prefix you would like the variable to have. This is covered in Section 7.

String Variables

You can also define strings in Mastercam posts. Strings are variables that hold text instead of numeric values. For example we could have a variable “strpost” with a value of “Xv_Haas_3_&_4_Axis.pst”. Now if we output strpost we would get the following output:

```
Xv Haas 3 & 4 Axis.pst
```

Strings are nice for outputting comments and other useful information about the type of pass that is being preformed...etc.

7 – Post Processors-3

Format Assignments

How you control the decimal output of a variable or what letter pre-fixes the output of the variable is controlled through format assignments. The different parts of format assignments are:

- Format Statements
- Variable Formats

Format Statements

Format statements consist of general format information organized into a table of information and labeled with a number. Example:

```
# -----  
# Format statements - n=nonmodal, l=leading, t=trailing, i=inc, d=delta  
# -----  
#Default english/metric position format statements  
fs2 1 0.7 0.6 #Decimal, absolute, 7 place, default for initialize (:)  
fs2 2 0.4 0.3 #Decimal, absolute, 4/3 place  
fs2 3 0.4 0.3d #Decimal, delta, 4/3 place  
#Common format statements  
fs2 4 1 0 1 0 #Integer, not leading  
fs2 5 2 0 2 0l #Integer, force two leading  
fs2 6 3 0 3 0l #Integer, force three leading  
fs2 7 4 0 4 0l #Integer, force four leading  
fs2 9 0.1 0.1 #Decimal, absolute, 1 place  
fs2 10 0.2 0.2 #Decimal, absolute, 2 place  
fs2 11 0.3 0.3 #Decimal, absolute, 3 place  
fs2 12 0.4 0.4 #Decimal, absolute, 4 place
```

There are two types of format statements:

fs – which is a single format statement for that number. (there is only one format specified)

fs2 – There are 2 formats specified for that format number.

The format statement consists of the following information:

1st column – this is where it is defined as a fs statement or a fs2 statement.

2nd column – this is the formats identifying number. Use this number in your variable formats.

3rd column – This column is made up of 3 parts. The first is the number of places before the decimal. The next is the decimal itself. The last is how many after the decimal. This column is used for Inch output. (Note: You can choose to not enter a decimal and no decimal will be output with any variable using this format statement in the program)

4th column – This column is the same as the 3rd column but used for metric Output

Format statements are setup so they can be used with one or many variables.

7 – Post Processors-3 Continued

if you look at the example of the format statements above you will notice that format statements 5, 6, and 7 have an “l” at the end of the last column. At the end of this last column you can control some aspects of the variable, and this is usually defined in the comments at the head of the format statements. The values that can be used here are:

- l – Output leading zeros
- t – Output trailing zeros
- d – Delta value output
- n - Non-modal output. (makes variable non modal)

There may be others that can be found using Mastercam’s post reference guide.

Variable Formats

Now if you want to assign the format of variable it will be done with variable formats. Variable formats are normally located in the upper portion of the post processor before any postblocks. A variable format look like the following sample:

```
# -----  
# Toolchange / NC output Variable Formats  
# -----  
fmt T 4 t$      #Tool number  
fmt T 4 first_tool$ #First tool used  
fmt T 4 next_tool$ #Next tool used  
fmt D 4 tloffno$ #Diameter offset number  
fmt H 4 tlngno$  #Length offset number
```

Here is an explanation of the columns in the variable format line.

- 1st column - “fmt” this tells the post that we are doing a variable format.
- 2nd column - This is any letter you wish to prefix the variable value.
- 3rd column – This is what format statement number you want the variable to be formatted with.
- 4th column - This is the variable name.

7 – Post Processors-3 Continued

You can change the format of a variable while posting using a conversion function. This conversion function will take a variable that has been formatted with one format and re-format that variable with the one you specify in the function. The syntax of this function is: `result = newfs(#, var)`

Explained:

“`result = newfs()`” – This is the conversion function.

- This is the format statement that you wish to format the variable with

“`var`” – This is the variable you wish to re-format

In the below example I have a variable “`tool_no`” which I want to change its current format statement to 4. To do this I put the following code in my post.

```
result = newfs(4, tool_no)
```

This new format will be used whenever this variable is output from now on unless I re-format the variable again elsewhere.

Changing the prefix of a variable that has already been defined

If I have a variable that has already been defined I can change the address of that variable in a similar fashion using “`result = nwadrs(str, var)`”.

Explained:

“`result = nwadrs()`” - This is the string function.

“`str`” – This is the string I wish to change the prefix of the variable to.

“`var`” – This is the variable I wish to change the prefix address of.

In the example below I have the variable “`tool_no`” which I want to change the prefix address to “`T`”. To do this I put the following code in my post.

```
result = nwadrs(strt, tool_no)
```

Notes on functions. I only use these functions when changing things mid program. You can change the initial definitions of variables and strings to effect their format throughout the entire program. Using these functions lets you change them mid program and then change them back to something else all while posting the same program.

8– Post Processors-4

In this section we are going to discuss string selector tables. String selector tables are tables that hold multiple different values that can be output through a single variable. The value that is output is determined by the value of an index variable. An example of a string selector table is shown below.

```
# Generate string for spindle
sm04 M4 #Spindle reverse
sm05 M5 #Spindle off
sm03 M3 #Spindle forward
spindle #Target for string

fstrsel sm04 spdir2 spindle 3 -1
```

Lets break down this table and explain what each element is. First off this selector table has 3 values that it holds. Those values are “M4”, “M5”, and “M3”.

Now the string variable that holds the value in the selector table is:

sm04 holds “M4”

sm05 holds “M5”

sm03 holds “M3”

The line that has “spindle #Target for string” is the definition of the variable that is going to hold the value of this table for output into the gcode program.

Below is the next line broken down into parts.

- 1 - “fstrsel” this is the format string select function.
- 2 – “sm04” is defining where we want to start in the tables values
- 3 – “spdir2” this is the index. Whatever value is in this determines which value is selected from the table.
- 4 – “spindle” again this is the variable that is going to get the value from the table.
- 5 – This is the number of values the selector table has. Here we have 3 different values listed in our table so this value is 3.
- 6 – “-1” for your purposes at this time this will always be -1. It’s definition is not within the scope of this guide.

8 – Post Processors-4 Continued

Now to explain all of this. String selector tables hold a list of values that are stored in strings. One of these strings will be selected based on the value of the index variable, and then the contents of the string will be put into the output variable for output in the program. Anywhere in the post the output variable is called this process is executed. The string select tables select the appropriate string based on the value of the index. They select starting at 0.

Using our example selector table shown on the previous page the following would be true.

If I wanted the variable “spindle” to have the value of “M3” then the index variable(spdir2) would need to equal the value of “0”.

So if the index variable(spdir2) equaled 2, then the value that would be output from the variable “spindle” would be “M04”.

Changing Existing String Selector Tables

One of the more common post changes is when someone needs to change the gcode set that is being output for a specific thing. Example would be the post is setup to use “M20” and “M21” do determine weather the program is in inch values or metric value. Now most post processors have a string selector table that is used for outputting these codes. Lets say we need to change these values to use “M70” and “M71”. We can do this simply by changing the values that are listed in our string selector table. Compare the string selector tables below to see how I accomplished this.

Original:

```
# -----  
#Select english/metric code  
sg20  G20  #Inch code  
sg21  G21  #Metric code  
smetric      #Target string  
  
fstrsel sg20 met_tool$ smetric 2 -1
```

Changed:

```
# -----  
#Select english/metric code  
sg20  G70  #Inch code  
sg21  G71  #Metric code  
smetric      #Target string  
  
fstrsel sg20 met_tool$ smetric 2 -1
```

Now when the variable “smetric” is output it will have a value of “G70” or “G71” based on weather met_tool\$ = 0 or 1.

9– Post Processors-5

In posing some times you need to be able to determine if things should happen or not. You can do this using a Condition Statement. There are defined operators that you can get additional information about in the Mastercam post ref guide but for the purpose of this guide we are going to cover IF statements.

IF Statements

You can use IF statements to determine if a condition is true, and based on that condition you can choose to do or output something. A written out example of this would be. If *var1* equals 1 then *do this*. Using an IF statement allows you to have great power in detecting and controlling output in a program.

Lets say that every time you have a particular tool output into your program you would like a message or gcode to appear. You could test against the variable holding the value of the tool number and then output what you would like if it is true.

When using IF statements you can also put a result if the condition is not true. This is an else statement. You will see IF and Else statements throughout the post processors that come with your Mastercam installation.

Example:

If t\$ = 1, "Tool One"

In this IF statement I am testing against the variable "t\$" to see if its value is 1. If the value of "t\$" is one then I will output a string literal of "Tool One".

I can change that up a bit by adding an Else statement in.

Example:

If t\$ = 1, "Tool One" else, "Not Tool One"
--

Now in this 2nd example I am doing the same IF statement but I added the else. What this will do is any time "t\$" is equal to one the output will be "Tool One" but any time "t\$" is not equal to one it will output "Not Tool One".

Note: In an if statement the "," separates the statement from the action. In this case I have been using a string literal output. We could be calling a post block, or doing some math function or many other possible things here.

9 – Post Processors-5 Continued

Here is a list of operators you can use in an If statement.

- = - This means that the value equals something.
- > - This means greater than
- < - This means Less than
- <> - This means not equal too
- <= - This is smaller than or equal to
- >= - This is larger than or equal to

Multi Line IF Statements

You can create multi line if statements by using the “[” signs. Example shown below.

```
If t$ = 1, [      # you start with open bracket
               "output some code"
               "test some stuff"
               "output more code"
               ]   # you end with close bracket
```

In the example above I show how you can have several lines of logic associated with the result of the condition statement. Any lines of logic in between the “[” and the “]” are tied to the IF statement.

You can also nest one if statement inside of another if statement but I recommend you keep good notes and try to keep your code organized in a way which makes it easy to understand.

Testing Multiple Items In one If Command

You can test more than one variable in one condition statement line by using the “|” or the “&” commands.

- | - Means OR
- & - Means AND

Examples of how this is applied shown below.

```
If t$ > 10 | if coolant = 0, "ERROR"
```

- This states that if “t\$” is larger than 10 or if coolant = 0 (off) the output the word “ERROR”

```
If t$ > 10 & coolant = 0, "ERROR"
```

- This states that if t\$ is larger than 10 and coolant = 0(off) then output “ERROR”

10– Post Processors-6

What is the NCI file? The NCI file is the intermediate file that Mastercam generates for MP. The NCI file contains all the information about your tool path that is needed to generate your gcode program. It also contains the XYZ locations of the cut profile that was programmed. The NCI file is where the data comes from when you are using variables in post processors. We use the NCI file to get our information when creating customizing or editing posts. It is important to understand that we only get 2 NCI data lines of information at any given time. There are large amounts of information available about the NCI file in the Mastercam Post Reference guide and I will not copy them here. I recommend you read and look that the provided post ref guide from Mastercam to learn more about the NCI file and how it works.