Debugger Quick Start

How do I...? / FAQ

1 *Enable the debugger*

Can't find it? Your .config file has an option that turns it on. Learn more...

2 *Start the debugger*

Post your toolpaths and click the button. Learn more...

3 *Set your screen layout*

Only do this once—select and lay out the windows you want to use. Learn more.

4 *Choose your Run mode*

First use **Turbo** to quickly preview NC output. Then use **Run** (normal) or **Step** (detailed, but slow) to analyze post logic and output. Learn more...

5 *Set watch lists and break points*

Isolate the information you want to focus on. Learn more...

6 *Analyze output and edit post*

Run post and use debugger windows to analyze output. Use editor to make changes to your post. Learn more...

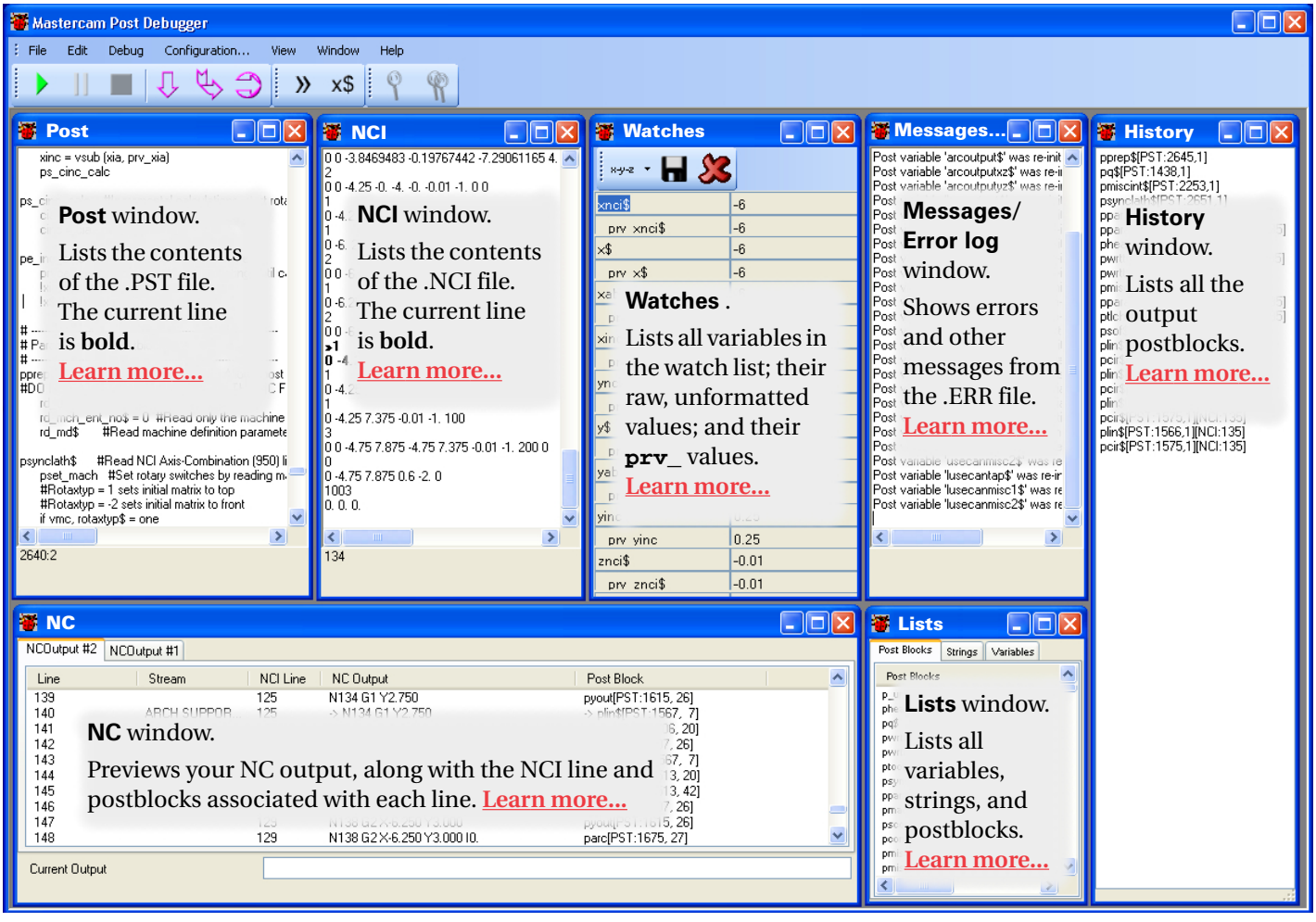
7 *Re-post and see changes*

Re-post from Mastercam to see changes—return to step 2. Learn more...

Shortcut keys

Find Next	F3
Run	F5
Stop	Shift+F5
Toggle Breakpoint	F9
Step Over	F10
Step Into	F11
Step Statement	Ctrl+F11
Pause	F12
Select All	Ctrl+A
Find	Ctrl+F
Add Watch	Ctrl+W

What the heck are all these windows??

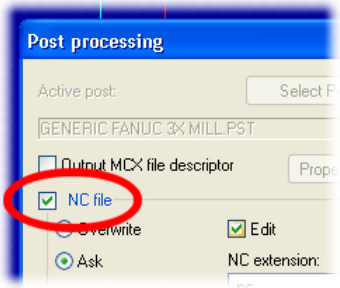


How do I…?

- *Make it run like fastmode?*
  Use **Run Turbo** mode—this is close to legacy fastmode output. Learn more...
- *Make changes to my post?*
  Open the PST file in your regular editor and make changes there. Re-post and run the debugger again to see your changes. You can't directly edit the PST file in the debugger window. Learn more...
- *Add something to the watch list?*
  Select it in the **Post** window and press **Ctrl+W**. Learn more...
- *Delete a single variable from the watch list?*
  Make the **Watches** window wider so you can see the **Delete** button next to the variable name. Learn more...
- *Make the font size bigger?*
  Select **Configuration** from the menu. Learn more...
- *Rewind the post and run it again?*
  Select **File > Quit** from the menu and re-post from inside Mastercam. There is no "rewind" function. Learn more...
- *I just want to step through a section of my post, not the whole thing. How do I do that?*
  Set a breakpoint before the section you want to step through. Run the post. When you hit the breakpoint, use one of the **Step** commands. Learn more...
- *Print the contents of one of the windows?*
  You can't print from the debugger. Open the NC, NCI, or PST file in your editor and print from there.
- *Set a breakpoint?*
  Move the cursor at the beginning of a postline or NCI line and press **F9**. Learn more...
- *Debug a "binned" post?*
  This works just like a regular post except that binned portions won't be visible in the **Post** window. Only unbinned portions will be shown. Learn more...

FAQ—

- *Why is it so slow?*
  You might be running in **Run Step** mode. Try regular **Run** mode or **Run Turbo**. Learn more...
- *I added variables to my watch list, but I don't see any values.*
  Use **Run Step** mode to see the values update in real time. In regular **Run** mode, they will update at breakpoints. In **Run Turbo** mode, they won't update at all. Don't forget to include the **$** for pre-defined variables. Learn more...
- *Nothing is happening, it looks like I'm stuck in an infinite loop.*
  Your post might be in the middle of a parameter read loop, which can take a long time to complete. If you're not interested in debugging this portion of your post, consider setting a breakpoint after your parameters are processed; then, use **Run** mode (not **Run Step** mode) until the breakpoint is reached. Learn more...
- *All my toolbar buttons are grayed out and nothing's working—what do I do now?*
  Your post has finished running. Return to Mastercam and re-post your part to continue the debugger session. Learn more...
- *I re-posted my part but the debugger isn't appearing. Mastercam looks like it's stuck.*
  Click the debugger icon in the Taskbar on your desktop or click on the debugger window itself.
- *The debugger is showing NC output that isn't actually in my NC file.*
  This can happen with canned cycles. Sometimes the NC window shows additional moves or data that the NC file—properly—omits. Verify the NC output in your editor.
- *I want to search for some text but the Find button is grayed out.*
  Find only works when you're in the **Post** window.
- *I can see the debugger button in Mastercam, but it's grayed out.*
  Select the **NC file** option:



Mastercam Post Debugger User's Guide

January 2019

Information may have been changed and/or added since this document was published. The latest version of this document is installed with Mastercam or can be obtained from Mastercam.

Mastercam® Post Debugger User's Guide

Contents

chapter 1

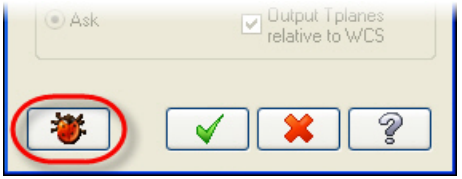
Using the Post Debugger

The Mastercam Post Debugger helps you find and correct errors in your posts. Using the debugger, you can step through the posting process a line at a time, while watching the post produce the NC code. As you manipulate the posting process, you can watch how variables change value and view a stack trace of the post execution. This chapter shows you how to use some of the debugger's essential features:

Enabling the debugger

Quick Start

The debugger is launched from a button on the **Post processing** dialog box.

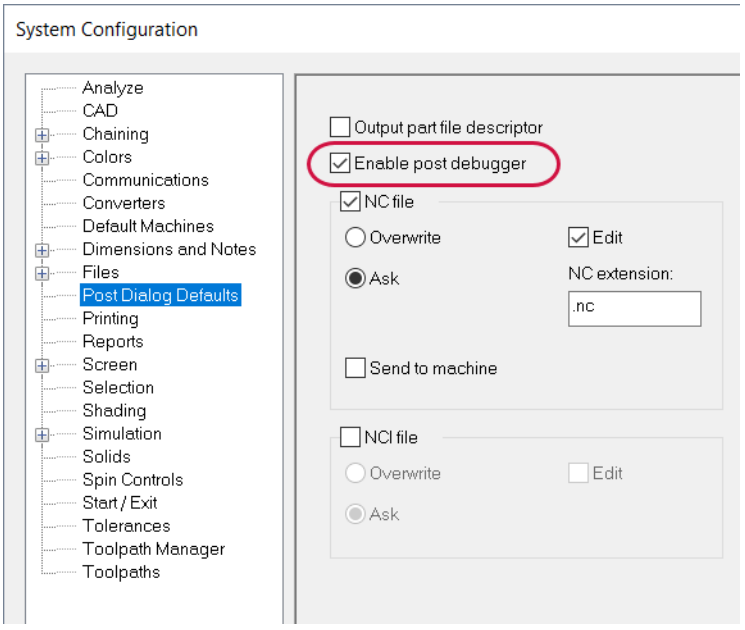


By default, this button is not visible. Before you can use the debugger, you need to enable the button. This is controlled by an option in your **.config** file. Follow these steps:

How do I...?

Enabling the debugger

1. Inside Mastercam, choose **File > Configuration**.
2. Select the **Post Dialog Defaults** page.



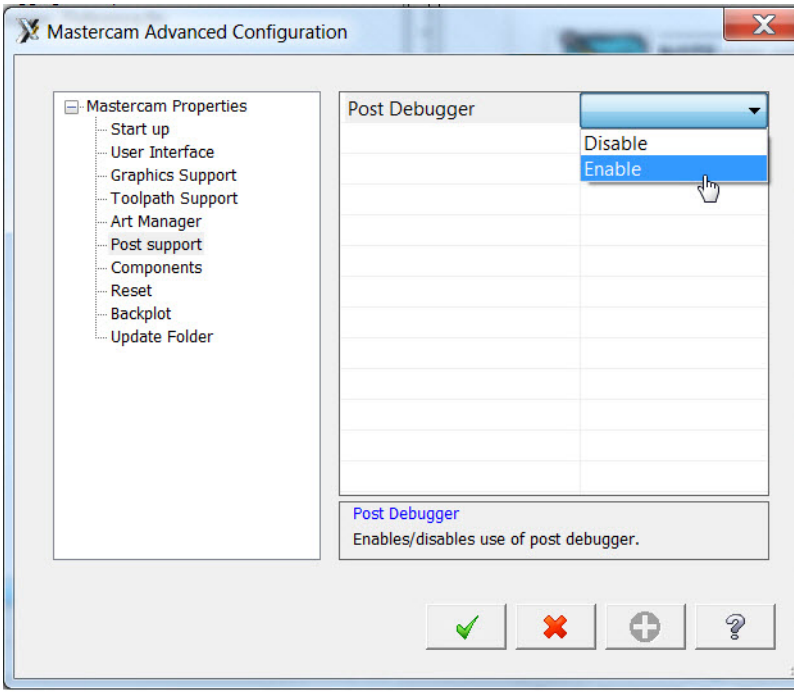
3 Choose **Enable post debugger**.

4 Click **OK** to save the configuration file.

Once the debugger is enabled, you will see its button in the **Post processing** dialog box:

Repeat these steps for both your inch and metric configuration files.

NOTE: In versions of Mastercam earlier than Mastercam 2020, use the **Advanced Configuration** utility to enable the debugger.



Launching the debugger

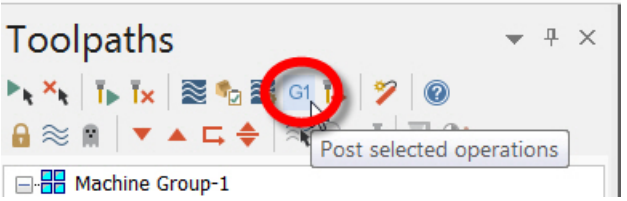
Quick Start

Launch the debugger as part of the regular posting process.

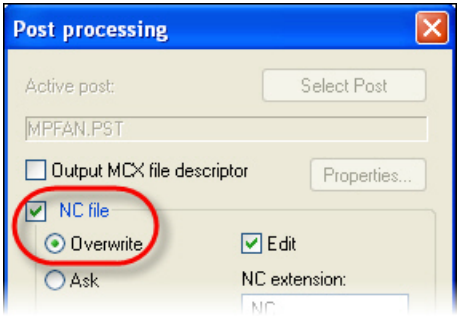
How do I...?

Launching the debugger

1. Select the operations that you want to post.
2. Click the **Post** button.



3. In the **Post Processing** dialog, select the **NC file** option.



4. Choose which **Edit** options you want. These will cause your default editor to automatically open when the debugger session is finished.

   Note that the **Edit** options (or the **NCI file** option) do not affect the debugger itself. You will always be able to see NC output and the NCI file in the debugger even if no **Edit** options are selected. Select them only if you want the **Editor** to open in addition to the debugger.

Best practice—*Unless you really want to see the NCI or NC file in your editor, leave the* **Edit** *options unchecked.This will greatly simplify the interaction between Mastercam and the debugger.*

5 Click the debugger button.

NOTE: If you don't see this button, the debugger has not been enabled. See **Enabling the debugger** on page 2.

6 Select the names of the NC and/or NCI files, if prompted.

The debugger will open automatically.

IMPORTANT: If a debugger session is already open—in other words, if you have been using the debugger and you are simply re-posting—Mastercam might look like it has locked up. All you need to do, though, is click on the title bar of the debugger window to make it active.

Customizing the screen display

You can choose which windows to display and how they are arranged. Typically, you only need to do this once—Mastercam will remember your layout between sessions.

How do I...?

Changing the font size

1. Select **Configuration** from the menu.
2. Choose **Select font**.
3. Select the desired font size.

   If you wish, you can also change the font, style, and other attributes.

Best practice—*Don't select a* **Bold** *font. The debugger uses bold to highlight the current NCI and post line—if you choose bold for everything, you won't be able to see this.*

How do I...?

Showing or hiding windows

Learn more about each window—see **Debugger windows** on page 44.

1. Use the **View** menu to select which windows will be visible.
2. Drag and resize them as desired, or use the **Window** menu to automatically position them.

Best practice—*If you need to conserve screen space, the Lists, History, and Messages windows are less important and can typically be turned off while you are running the post. You can always turn them on and inspect them after the post has run.*

You can also customize the information that gets displayed in the NC window. See **Customizing the NC window** on page 52.

Running the post

How you use the debugger depends on what you are looking for in your post. As you use the debugger, you will discover different ways to find the information you need. To get you started, the following sections describe some basic debugging techniques.

The simplest way to use the debugger is to click the **Run** button. Different **Run** modes are available from the **Debug** menu. Each runs the post from beginning to end, populating the NC window with output. See **Understanding the different run options** on page 8 to learn more about them.

While the post is running, you can pause it and examine the contents of the windows.

- To pause the post while it is running, do either of these two things:
  - Click the **Pause** button: 
  - Set a breakpoint.
- Often while the post is running, it is difficult to stop it exactly where you want with the pause button. Setting breakpoints gives you a lot more control.
- Once the post is paused, you can either choose to step (or trace) through it. This lets you run one line at a time so you can see what is happening in great detail.

When the post finishes, you can use your editor to make changes to the post, and/or return to Mastercam to make changes to your part and re-post.

Understanding the different run options

The debugger gives you three different run modes. Choose the right one by balancing your need for speed with the desired level of detail.

Table 1: Comparing the different Run modes

Mode	Speed	Expanded NC output?	Trace enabled?	Watch list?
Run Turbo	Fastest	No	No	No
Run (normal)	Fast	Yes	Yes	Only at breakpoints
Run Step	Slow	Yes	Yes	Yes

See **NC window** on page 49 to learn more about these terms.

Pay attention to these guidelines:

- If your post reads operation parameters, avoid **Run Step** mode. The parameter read loops can take a long time.
- You can mix-and-match run modes. In other words, you can start in one mode, pause, and resume in a different mode. So, for example, you can run in normal mode until your post is finished with parameter reads; then you can pause it (or set a breakpoint) and resume in **Run Step** mode to see watch list variables.
- Consider first running in **Turbo** mode to quickly see the NC output and look for problem areas. This will let you identify areas where you want to set breakpoints or identify variables to add to the watch list. Then return to Mastercam and re-post.
- **Turbo** mode will not stop for breakpoints.
- In **Run** (normal) mode, you can see watch list values only when the debugger is paused. You will not see them update while it is running.

See **Making the debugger work like legacy debug variables** on page 25 for more information about running the debugger.

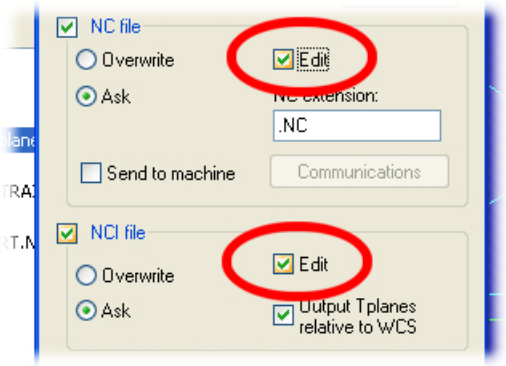
What happens when the post is finished running?

You can only run the post once during each debugger session. The session ends when either of two things happens:

- The debugger finishes processing the NCI file and the last postblock has been executed.
- You click the **Stop** button.

Once the debugger finishes processing, control passes back to MP.

- If either of the **Edit** buttons was selected when you launched the debugger:



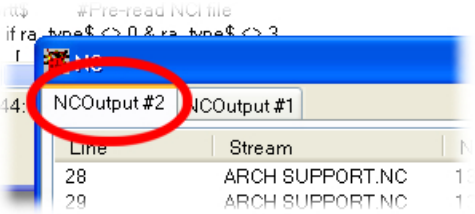
  your default editor will automatically start. The debugger window will still be open in the background. You can return to it and review the output in the different windows.

- If no **Edit** options were selected, the debugger window remains open, but the toolbar buttons will be grayed out.

You can now do one of two things:

Quit the debugger—Select **File > Quit** from the menu, or simply click on the Mastercam or Editor window. This leaves the debugger session active. You can edit your PST file with the Editor, make changes to your part in Mastercam—or both—then re-post your part.

- When you return to the debugger, your output will appear in a new NC tab:



- Your watch list and breakpoints will be preserved.

Exit or close the debugger—Select **File > Exit** from the menu, or simply close the window. The next time you launch the debugger, the NC output window, variable watch list, and breakpoints will all be cleared.

IMPORTANT: If you are returning to an active debugger session from Mastercam—in other words, if you have been using the debugger and you are simply re-posting—Mastercam might look like it has locked up. All you need to do, though, is click on the title bar of the debugger window to make it active.

Tracing through a post

Instead of just running the post and examining the output, you will often want to trace through a post step by step.

- As each instruction executes, you see the resulting output (if any).
- You can more easily verify the post's logic.
- You can see exactly when variable values change in the watch list.

There are a number of different ways to trace through a post. The following procedures describe some common techniques.

Best practice—*Typically, to set up a trace, you will set a breakpoint before running the post to automatically pause the post at the proper spot.*

If you are not using breakpoints, remember to click the **Pause** *button, not the* **Stop** *button. If you click* **Stop***, the debugger session will end.*

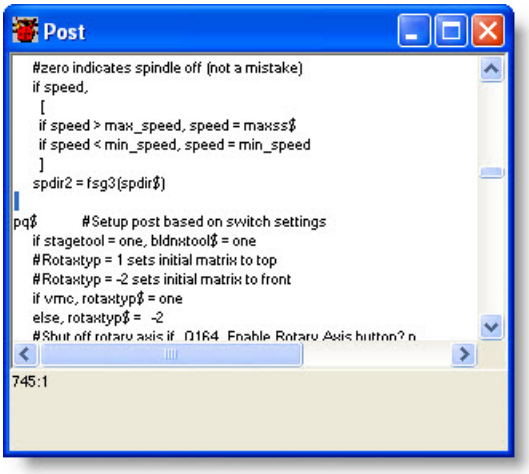
TIP: Tracing is especially effective when **Expanded NC output** is turned on. See **What is "Expanded NC output" ?** on page 52 to learn more.

How do I...?

Tracing with Step Into

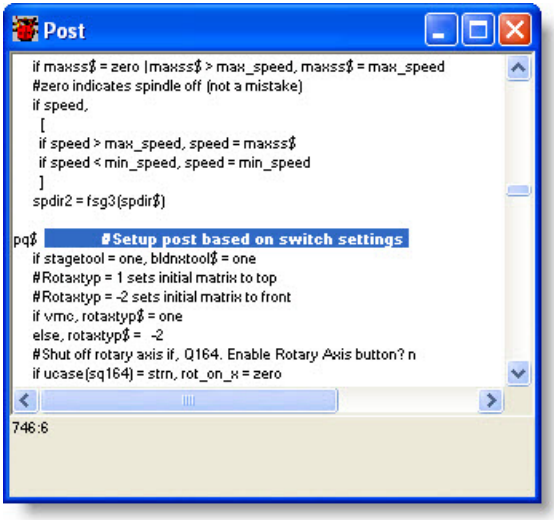
The debugger's **Step Into** command lets you watch the post execute one line at a time. Each time you click **Step Into**, the debugger executes a line and updates the windows to show the results. The debugger then waits for you to issue another command.

1 Start the debugger. The **Post** window displays your PST file.



2 Click the **Step Into** button.

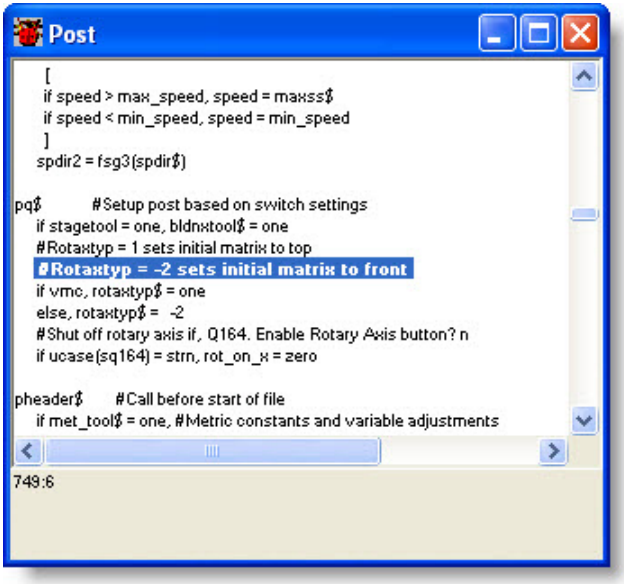
The debugger moves to the first code line inside of the current post block.



3 Click **Step Into** again.

- If the current line is an instruction, the debugger executes the instruction and moves to the next line, as shown in the following figure.

- If the current line calls a post block or function, the debugger jumps to that post block or function.



4 Keep pressing **Step Into** to trace deeper and deeper into the post, watching to see that the instructions execute in the order that you expect.

How do I...?

Tracing with Run Step

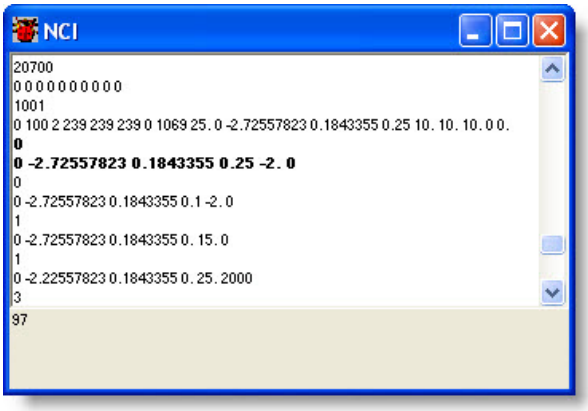
The **Run Step** command lets you watch post processing as it happens, without your having to continually click a button.

1 Start the debugger.

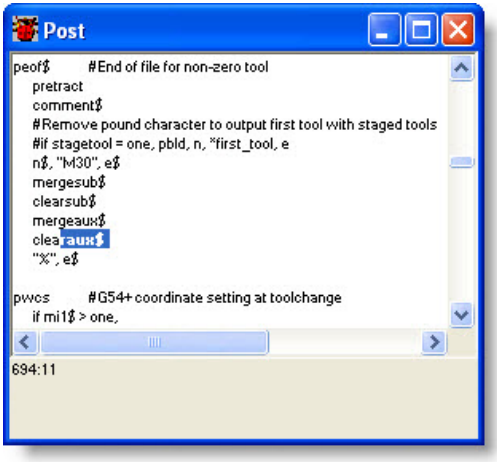
2 Select **Run Step** from the **Debug** menu. The debugger starts executing the post.

See **Understanding the different run options** on page 8 to learn more about the run modes.

3 Watch the **NCI** window to see the current line the post is processing.



4 Watch the **Post** window to see the debugger quickly processing post lines.



Using breakpoints

Often, you don't want to debug an entire post from the beginning. More likely, you want to trace from a specific location in your post. Breakpoints make this type of debugging possible.

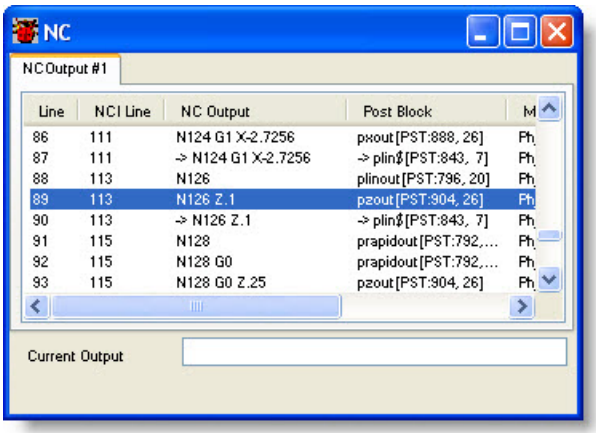
How do I...?

Tracing with breakpoints

1. Start the debugger.
2. Select **Run Turbo** from the **Debug** menu to quickly run through the entire post and populate the debugger's windows with output.

NOTE: Select **Run**, not **Run Turbo**, if you want to see expanded NC output.

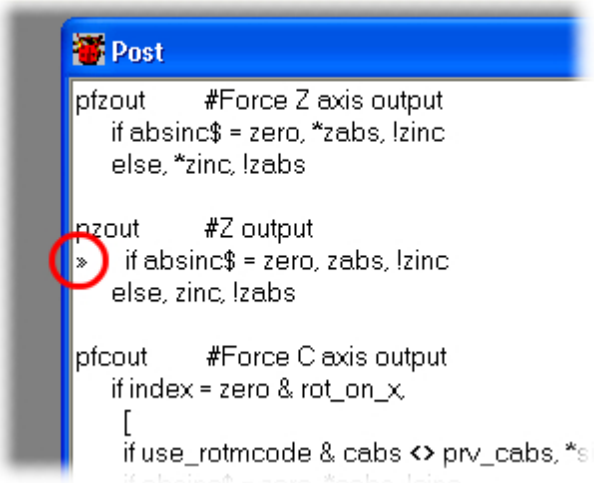
3. In the **NC** window, locate the instruction where you want to start debugging.



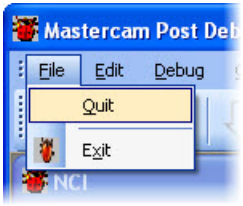
4 Double-click the NC line. The debugger highlights the NCI and post lines that generated the code.

You can also simply press **F9** to set the breakpoint.

5 Right-click the highlighted post line, and choose **Toggle Breakpoint** from the pop-up menu. A small double-arrow appears next to the selected line.



6 Choose **File > Quit** to stop the debugging session, but leave the debugger open.



7 Go back to Mastercam and re-start the debugger session.

   a Make sure the same operations are selected.

   b Click the **Post** button.

   c Click the debugger button.

   d Click on the debugger window to activate it.

8 Click the **Run** button.

   The debugger processes the post until it gets to the breakpoint you set. It will pause and waits for your command.

NOTE: Run Turbo mode does not stop for breakpoints.

9 Click the **Step Into** button to start tracing from the breakpoint. (See **Tracing through a post** on page 11 to learn more.)

How do I...?

Clearing breakpoints

1 To clear a single breakpoint, click anywhere in the line with the breakpoint and press **F9**.

2 To clear all breakpopints, select **Remove All Breakpoints** from the **Debug** menu, or press **Ctrl + Shift + F9**.

Watching variables

Quick Start

Often, a wrong variable value is the cause of a post problem, so an important part of debugging is watching to see how and where post variable values change. Use the **Watches** window to monitor variable values. When you add a variable to this window, you can step through a post and see the variable's value at any point in the posting process.

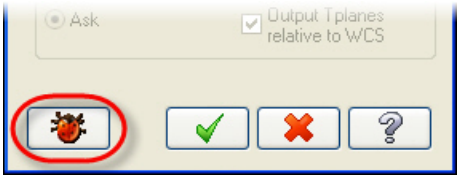
- You can see the values updating when you are stepping through the post or running in **Run Step** mode.
- If you are in normal **Run** mode, you will not see the values update in real time, but they will update when the post pauses at breakpoints.

TIP: You can watch both numeric variables and strings.

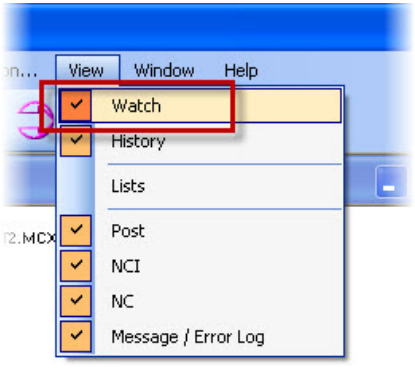
How do I...?

Watching a variable

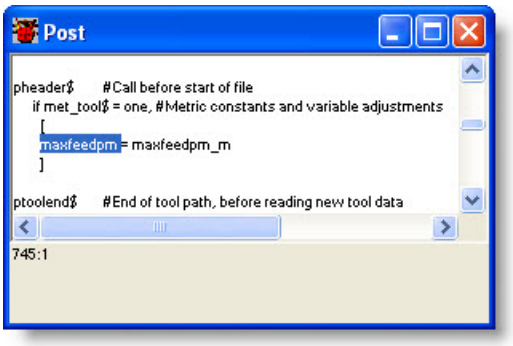
1 Start the debugger.



2 If you do not see the **Watches** window, choose **View > Watch** from the menu.



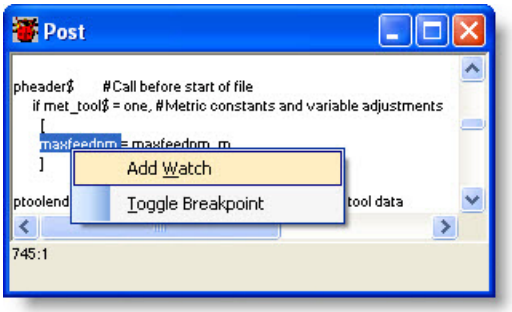
3 In the **Post** window, locate the variable that you want to watch.



4 Select the variable.

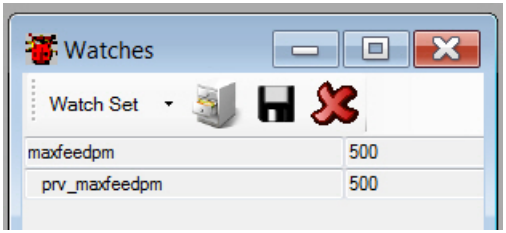
TIP: If it is a pre-defined variable, make sure you select the **$** following the name. If you double-click on the name, Mastercam will automatically select the **$** too.

5 Right-click and select **Add Watch** from the pop-up menu.



You can also right-click and select **Add Watch** when nothing is selected. Mastercam will display a window where you can simply type in the name of a variable to watch.

6 Trace through your post, watching for variable values and changes in the **Watches** window.

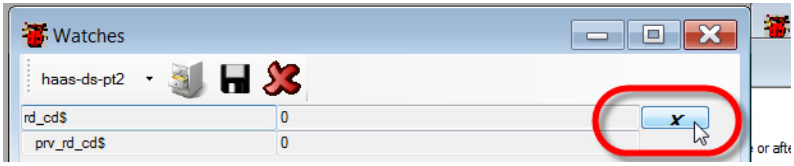


NOTE: The debugger automatically adds the **`prv_`** variable.

How do I...?

Removing a variable from the watch list

1 Click the **Clear** button next to the variable name.



Note that this button is typically not visible unless you expand the **Watches** window and make it much wider than usual.

How do I...?

Clearing the entire watch list

1 Click the **Clear** button in the **Watches** toolbar.

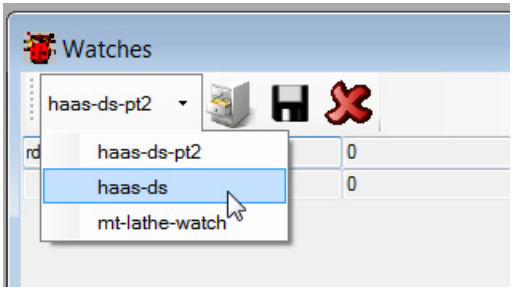
Saving and loading watch lists

The debugger lets you save a watch list to a file. This lets you recall sets of variables with a single keystroke.

This can be very powerful when the debugger is paused. You can quickly open sets of variables and check sets of values, one after the other, and get a detailed picture of the state of your post session at any particular point in time. For example, you can create saved lists of tolerance settings, control definition settings, rotary axis settings, or whatever is useful to you.

Use the **Select path** button in the **Watches** toolbar to tell Mastercam where your watch lists are stored. When you set the folder, Mastercam will:

- Automatically populate the list on the toolbar with all the watch lists that are available in that folder.
- Use that folder as the default location for saving new watch lists.



How do I...?

Saving a watch list

1. Click the **Save** button in the **Watches** toolbar.
2. Navigate to the desired folder.
3. Enter the name of the file.

How do I...?

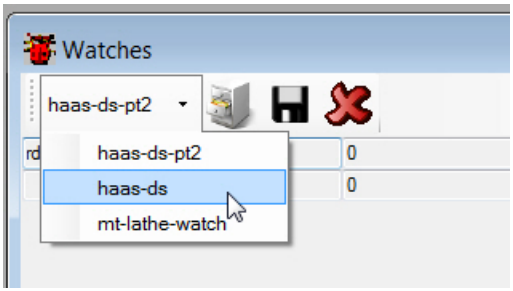
Setting the folder where watch lists are stored

1. Click the **Select path** button in the **Watches** toolbar.
2. Navigate to the desired folder, or click the **Make New Folder** button to create one.

How do I...?

Loading a watch list

1 Select it from the **Watches** toolbar.



2 If the watch list you want is not displayed, click the **Select path** button in the toolbar and navigate to the proper folder.

Working with watch list files

Watch list files have a **.MPWatch** extension. Despite the extension, they are simple XML files. If you want to create large watch lists, or many watch list files, you can easily create or edit the files directly with a text editor or XML editor.

The files look like this:

```
<?xml version="1.0" encoding="utf-8"?>
<ArrayOfString xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" xmlns:xsd="http://www.w3.org/
2001/XMLSchema">
  <string>xnci$</string>
  <string>x$</string>
  <string>xabs</string>
  <string>xinc</string>

  <string>ynci$</string>
  <string>y$</string>
  <string>yabs</string>
  <string>yinc</string>

  <string>znci$</string>
  <string>z$</string>
  <string>zabs</string>
  <string>zinc</string>

</ArrayOfString>
```

To add new variables to a file, all you need to do is simply enclose the variable name between **`<string> ... </string>`** tags.

As long as the file has the proper extension, the debugger will automatically recognize it and add it to the **Set watch** list when its folder is selected.


Quick Start

chapter 2

Editing and Debugging Posts

In this chapter, you will learn about:

In addition, there is a complete tutorial example:

The previous chapter concentrated on how to use the debugger tools. The sections in this chapter will show you how to use the debugger to create the proper workflow for accomplishing specific post development and debugging tasks.

Editing your post

It is important that you understand that the debugger doesn't actually let you edit your PST file. It simply provides a controlled environment where you can run the PST and watch it closely. To actually write, edit, and save the PST file itself, you will use the same editor that you already use.

When you start the debugger from inside Mastercam, the debugger loads your PST file from disk; reads and parses it; and loads it in memory. When you run it inside the debugger, it runs your PST file from its memory.

Because of this, you can't actually see changes to your post in real-time. You can't see the effect of changes to your PST file until you re-post your part and MP re-loads your post in its memory.

Follow this general workflow when working on your post.

How do I...?

Making changes to your post while using the debugger

1. Start the debugger.
2. Run your post, and see the contents of the different output windows.
3. Start your editor and load the PST file.
4. Use the debugger tools to identify where in the PST file you need to make changes.
5. Make the changes in your editor.
6. Save the PST file.
7. In the debugger, select **File > Quit** from the menu.
8. Re-post the toolpaths from inside Mastercam and launch the debugger again.

Now when you run the post, you will see the effect of your changes. The debugger will create a new tab in the **NC** window for the output from the updated post. This lets you easily compare the two and identify how your changes affected the NC output.

Making the debugger work like legacy debug variables

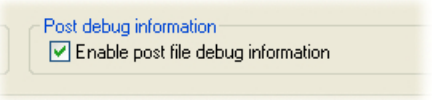
This section discusses some of the legacy debugging tools that MP supported before the debugger was released, and how they relate to the debugger. If you are used to debugging your posts with the traditional debug variables, this section will tell you how to reproduce that same functionality with the debugger.

This section also explains what uses, if any, the traditional debug variables still have and how they should be set.

If you are not familiar with editing posts in Mastercam X2 and earlier, you do not need to read this section.

fastmode$

The **`fastmode$`** variable was the traditional way to turn on debugging. Setting **`fastmode$ = 0`** (or **`no$`**) enabled the other bug switches. Starting with Mastercam X, it was toggled with a control definition setting:



This setting was taken out in Mastercam X3. **`fastmode$`** is now automatically initialized to **`yes$`** and you do not need to include it in your PST files.

NOTE: **`fastmode$`** can still be used to enable scaling. If you manually set it to **`no$`**, it enables scaling via **`scalex$`**, **`scaley$`**, and **`scalez$`**.

The debug output that you got with **`fastmode$`** depended on the state of the other debug switches. The next section tells you how to reproduce these results with the debugger.

The bug2$ switch

The **bug2$** switch was one of the most common tools for debugging a post. To refresh your memory about how it worked:

- When **fastmode$ = 0** and **bug2$** was set to a positive value, each line of NC file included the name of the postblock that output the line. If the line of NC code was output by more than one postblock, the NC file contained the names of the first and last postblocks that wrote to that line.
- If **bug2$** was set to a negative value, displayed all the postblock labels that have been called in the post, even those that do not result in output.

The **bug2$** switch no longer has any function and can be removed from your post. The debugger lets you mimic its features in the following ways:

- To simply see the first output postblock for each line with the fastest possible output, choose **Run Turbo** from the **Debug** menu.
- To see *all* the output postblocks for each line, follow these steps:
  - **a** Turn on **Expanded NC output**.
  - **b** Choose **Run** from the **Debug** menu.
- To see all the postblocks that are executed, whether or not they produce output, follow these steps:
  - **a** Choose **Run Step** from the **Debug** menu.
  - **b** Make the **History** window visible. The postblocks will be listed there.

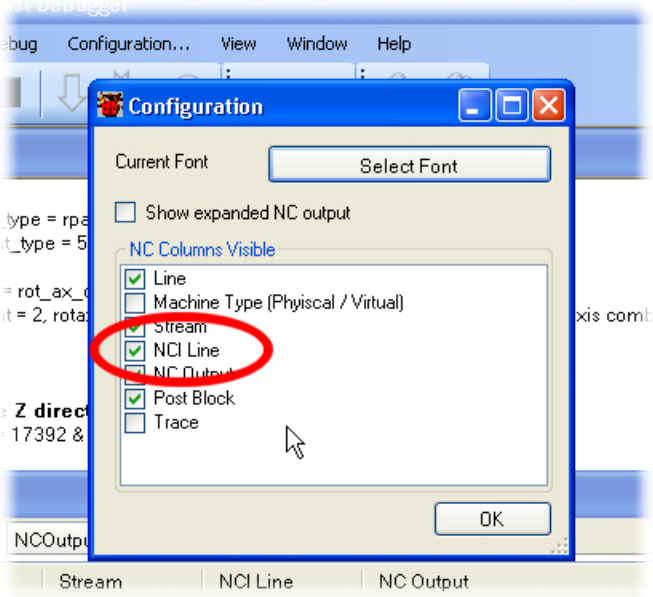
See **What is "Expanded NC output" ?** on page 52.

See **History window** on page 57.

The bug4$ switch

The **bug4$** switch was used to output the current line number of the NCI file at the end of each line of NC code. Setting its value to **1** activated this feature.

This is no longer necessary: all three **Run** modes in the debugger will include the NCI line number in the **NC** window. Just make sure that the **NCI Line** option is turned on in the **Configuration** dialog box:



However, **bug4$** is still useful when you want to output raw, unformatted variable values with the **~** (tilde) operator.

- When **bug4$ = -1**, the **~** operator is active. The ~ operator will force raw, unformatted output.
- When **bug4$ = 0** or **1**, the **~** operator will force output, but the output will be formatted according to the applicable format statement.

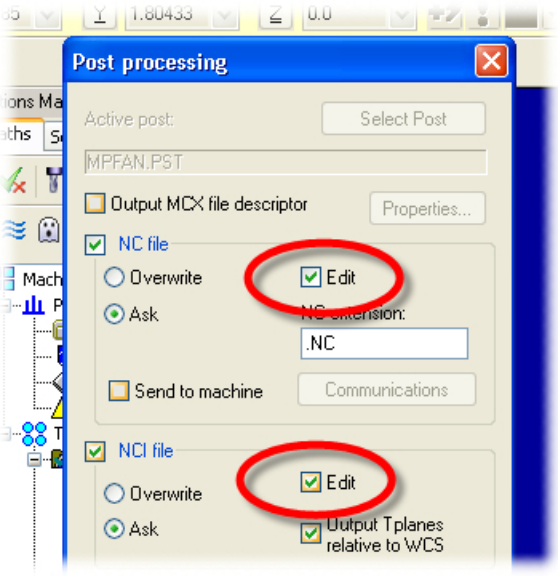
You need to use your editor to set **bug4$** and manually add the **~** in front of the desired variables.

Note, however, that the debugger can often give you the same information more easily. Just add the desired variable to a watch list; the watch list window always shows all values unformatted. This way, you do not need to constantly keep making changes to your post.

Other debug switches

The variables **bug1$** and **bug3$** can also be removed from your PST file, they no longer have any function. The only remaining use is for ATP: you can set **bug1$ = 1** to see the NC output in a list box. For normal posting, though, **bug1$** is replaced by the **Edit** options that you select in the **Post Processing** dialog box.

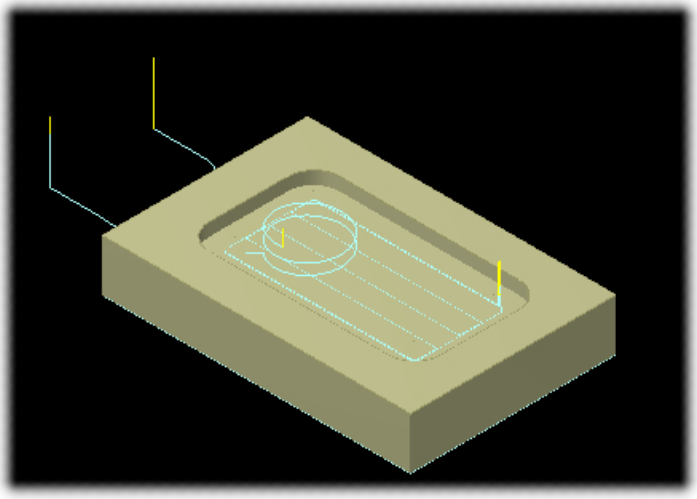
Quick Start



Tutorial example: Finding out where a value comes from

This section walks you through an example of how you might use the debugger to perform a real-world debugging task.

In this example, you will look at the toolpaths for the part shown here:



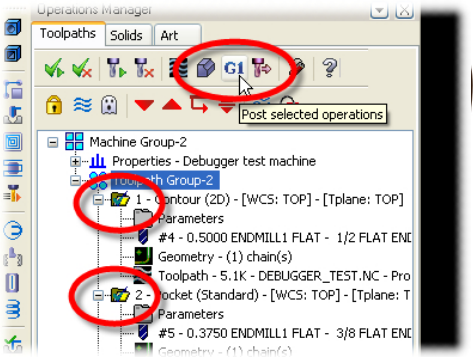
There are two simple toolpaths—a contour toolpath and a pocket toolpath—using two different tools. In this example, you will look at the G43 codes for each toolpath and figure out how they are output:

- what postblocks and postlines output the codes
- what variables are used to store the values
- what NCI lines are current when the G43 is processed

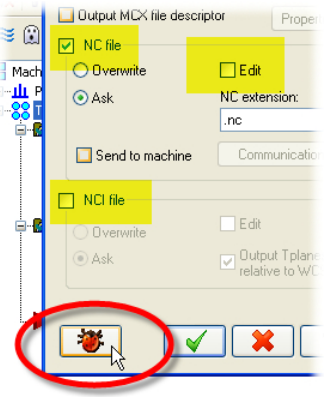
Once you learn this information, you will know where in your post to make changes if you need to modify the output.

Example

Example 1: Starting the debugger



1 Select the operations and click the **Post** button.



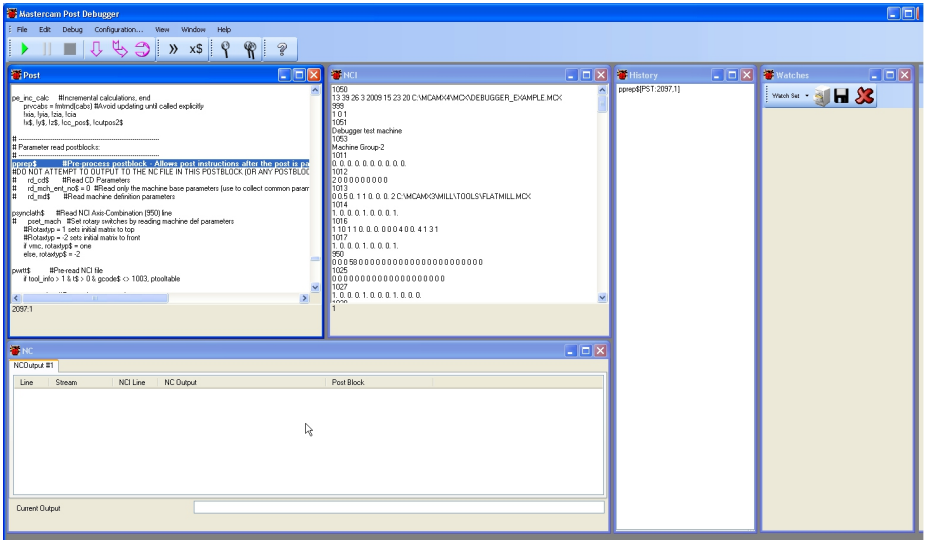
2 Set your edit options as shown here and click the **Debug** button.

See **Enabling the debugger** on page 2 if you do not see this button.

3 Enter a file name for the NC file and click **OK**.

4 If necessary, take a few moments and arrange your windows in a useful layout.

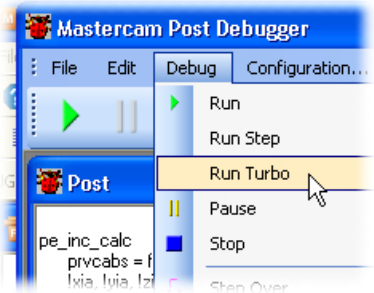
See **Customizing the screen display** on page 6.



Example

Example 2: Previewing the NC output

When the debugger first starts, your PST file is loaded in the **Post** window. The first postblock that will be executed—typically, **`pprep$`**—will be highlighted. The **NCI** window is populated with your entire NCI file.

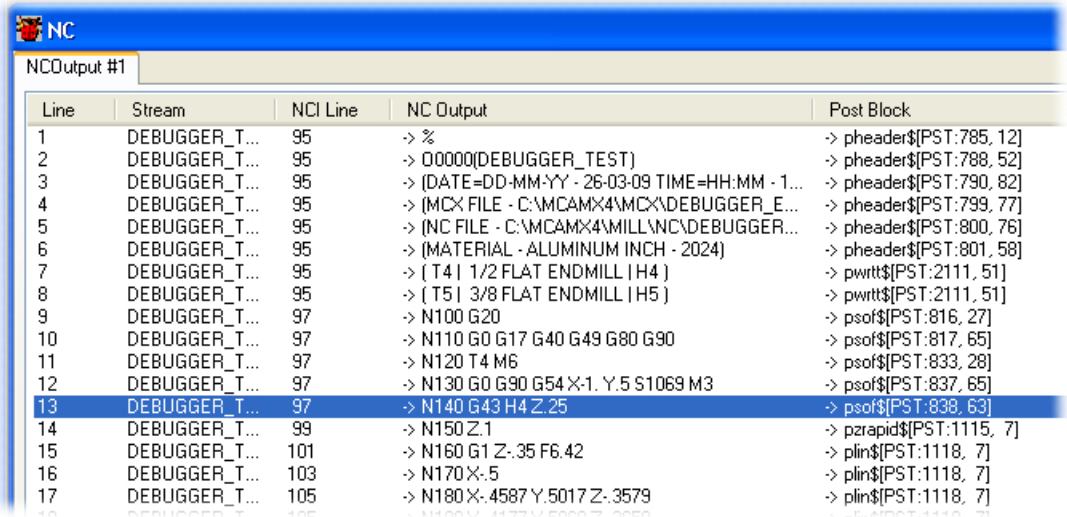


1 Select **Run Turbo** from the **Debug** menu. .

Mastercam runs the post in Turbo mode and quickly populates just the **NC** window with output

2 Go to the **NC** window and find the G43.

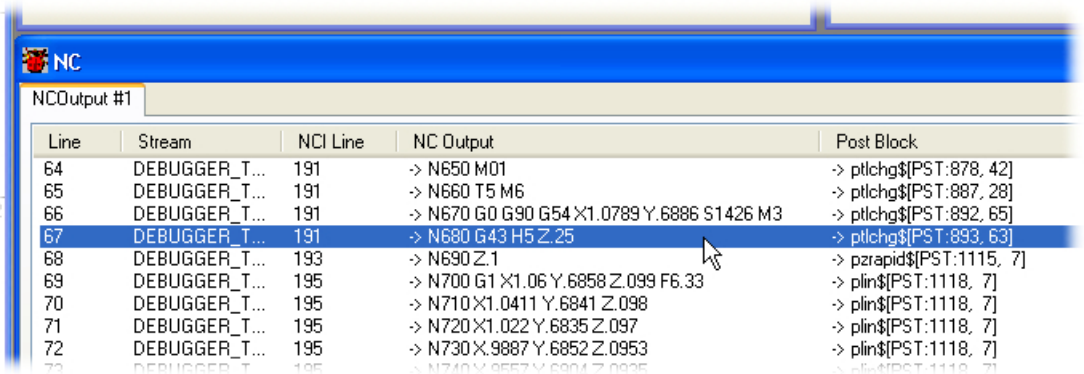
When you find the line, you can see that is was output by the **`psof$`** postblock, and that line 97 was the current NCI line.



Line	Stream	NCI Line	NC Output	Post Block
1	DEBUGGER_T...	95	-> %	-> pheader$[PST:785, 12]
2	DEBUGGER_T...	95	-> O0000(DEBUGGER_TEST)	-> pheader$[PST:788, 52]
3	DEBUGGER_T...	95	-> (DATE=DD-MM-YY - 26-03-09 TIME=HH:MM - 1...	-> pheader$[PST:790, 82]
4	DEBUGGER_T...	95	-> (MCX FILE - C:\MCAMX4\MCX\DEBUGGER_E...	-> pheader$[PST:799, 77]
5	DEBUGGER_T...	95	-> (NC FILE - C:\MCAMX4\MILL\NC\DEBUGGER...	-> pheader$[PST:800, 76]
6	DEBUGGER_T...	95	-> (MATERIAL - ALUMINUM INCH - 2024)	-> pheader$[PST:801, 58]
7	DEBUGGER_T...	95	-> (T4 | 1/2 FLAT ENDMILL | H4)	-> pwrtt$[PST:2111, 51]
8	DEBUGGER_T...	95	-> (T5 | 3/8 FLAT ENDMILL | H5)	-> pwrtt$[PST:2111, 51]
9	DEBUGGER_T...	97	-> N100 G20	-> psof$[PST:816, 27]
10	DEBUGGER_T...	97	-> N110 G0 G17 G40 G49 G80 G90	-> psof$[PST:817, 65]
11	DEBUGGER_T...	97	-> N120 T4 M6	-> psof$[PST:833, 28]
12	DEBUGGER_T...	97	-> N130 G0 G90 G54 X-1. Y.5 S1069 M3	-> psof$[PST:837, 65]
13	DEBUGGER_T...	97	-> N140 G43 H4 Z.25	-> psof$[PST:838, 63]
14	DEBUGGER_T...	99	-> N150 Z.1	-> pzrapid$[PST:1115, 7]
15	DEBUGGER_T...	101	-> N160 G1 Z-.35 F6.42	-> plin$[PST:1118, 7]
16	DEBUGGER_T...	103	-> N170 X-.5	-> plin$[PST:1118, 7]
17	DEBUGGER_T...	105	-> N180 X-.4587 Y.5017 Z-.3579	-> plin$[PST:1118, 7]

3 Continue scrolling through the **NC** window and find the other G43.

When you find the line, you can see that the G43 for the second operation was output by the **`ptlchg$`** postblock.



Example

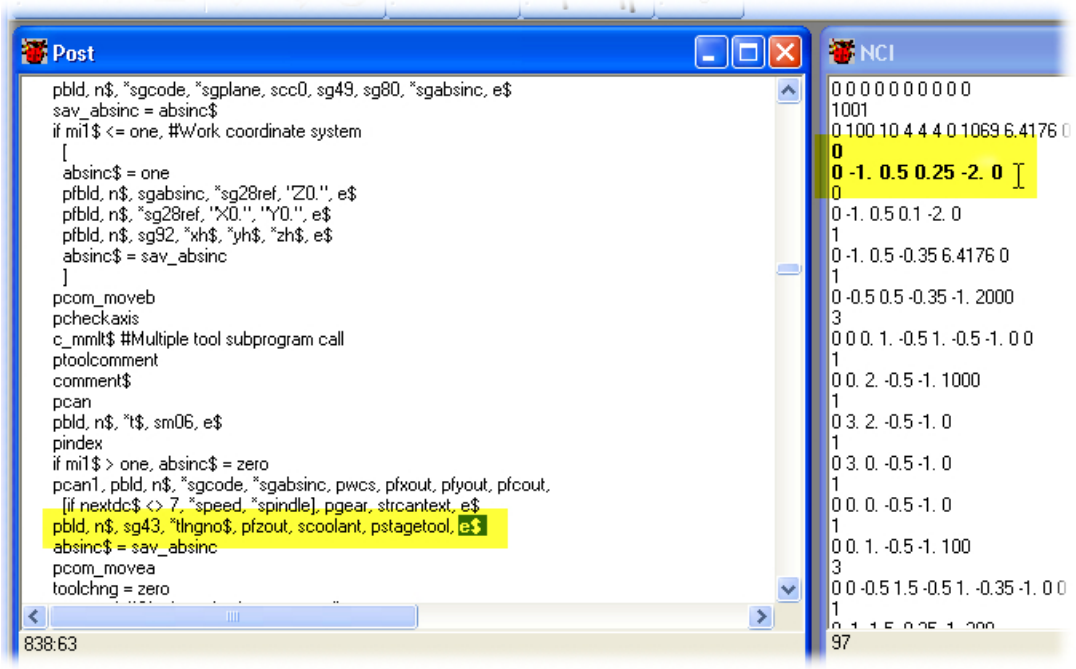
Example 3: Setting breakpoints and watching variables

Quick Start

Once you know where the output is coming from, set breakpoints just before those locations, so you can study the output in detail. Once you see the output postlines, you can figure out which variables are holding the values, so you can add them to the watch list.

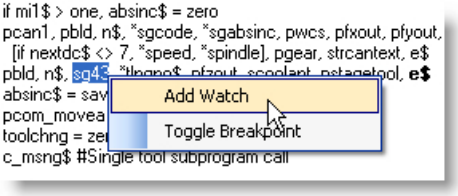
1 Go back to the first G43 line in the **NC** window and double-click it.

Mastercam sets the **Post** window to the output postline in **`psof$`** and **NCI** window to line 97.

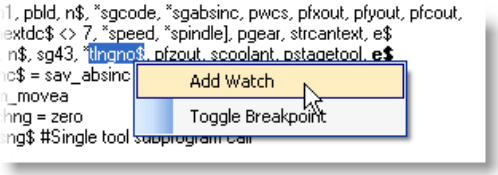


By comparing the postline with the NC output, you can guess that **`sg43`** is the string that holds the G43, and **`tlngno$`** is the variable that holds the offset number.

2 Select **sg43**. Right-click on it and select **Add Watch**.

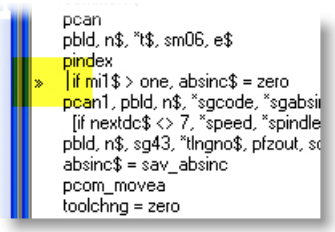


3 Select **tlngno$**. Right-click and select **Add Watch**.



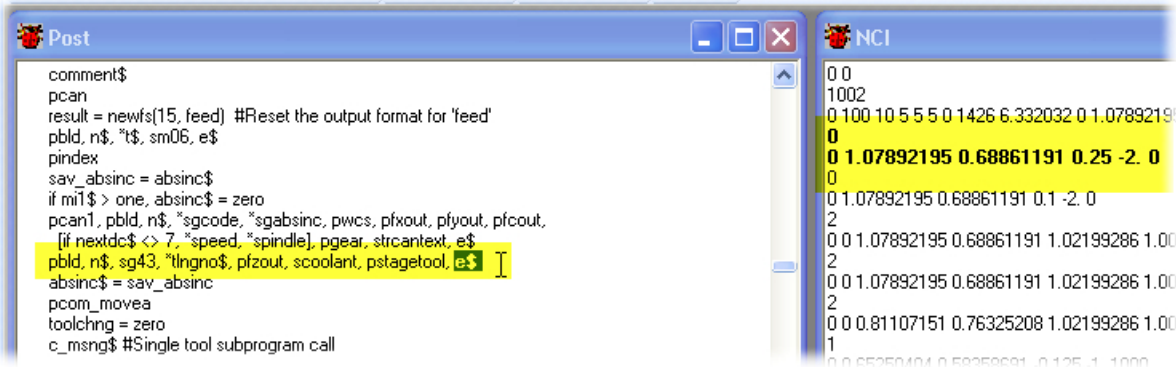
NOTE: Make sure that you select the **$** following the variable name. Make sure that you do *not* select the ***** operator in front of the name.

4 Move up a couple of lines before the G43 output postline and press **F9**. Mastercam adds a breakpoint as shown here.



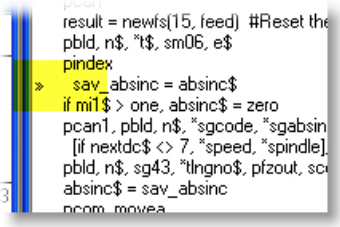
5 Go back to the **NC** window. Find the second G43 and double-click it.

Mastercam sets the **Post** window to the output postline in **`ptlchg$`**.



You can see that the output postline uses the same variables as the one in **`psof$`**, so you do not need to add any new variables to the watch list.

6 Again, move up a couple of lines before the G43 output postline and press **F9** to add another breakpoint.

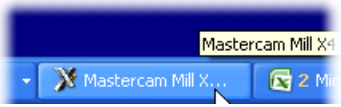


Example

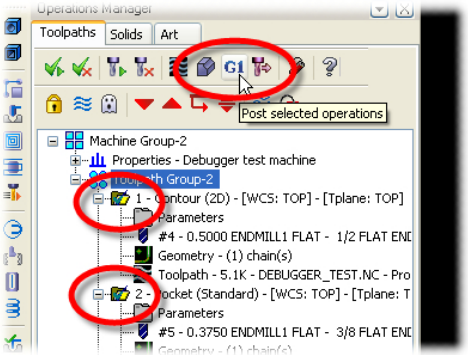
Example 4: Returning to Mastercam and reposting the part

To use the breakpoints and watch list, you need to return to Mastercam and repost the toolpaths. The debugger will preserve the work that you did in this session.

1 Click the **Mastercam** icon in your **Taskbar** to return to Mastercam.

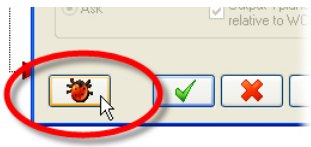


2 Make sure you select the same operations and click the **Post** button.

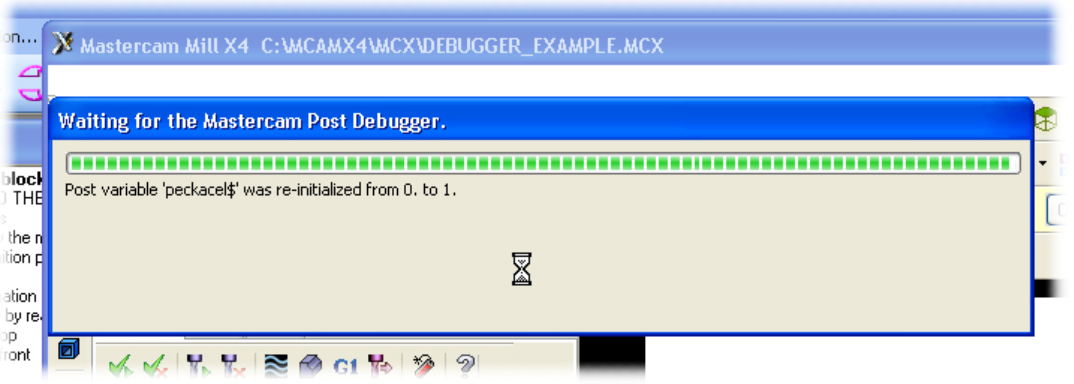


3 Click the **Debug** button to return to the debugger.

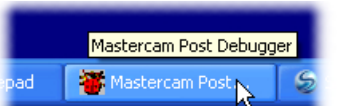
4 Confirm the NC file name when prompted.



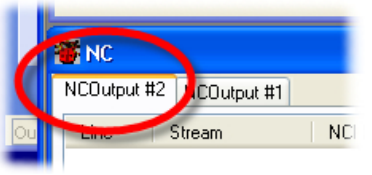
5 When you re-launch the debugger, Mastercam will pause at this screen:



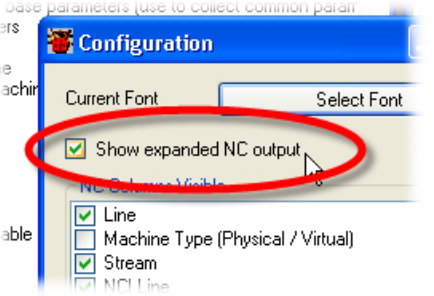
6 It might look like Mastercam has locked up, but click the **Debugger** icon in the Task bar—or just click in the debugger window—to activate it.



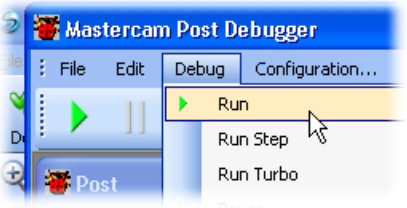
Notice that there is now a second tab in the **NC** window. Each time you repost, Mastercam adds a new tab to this window so you can compare the NC output from each session.



7 Select **Configuration** from the menu. Make sure that **Show expanded NC output** is selected.



8 Select **Run** from the **Debug** menu.



The debugger runs until it reaches the first breakpoint, then pauses.

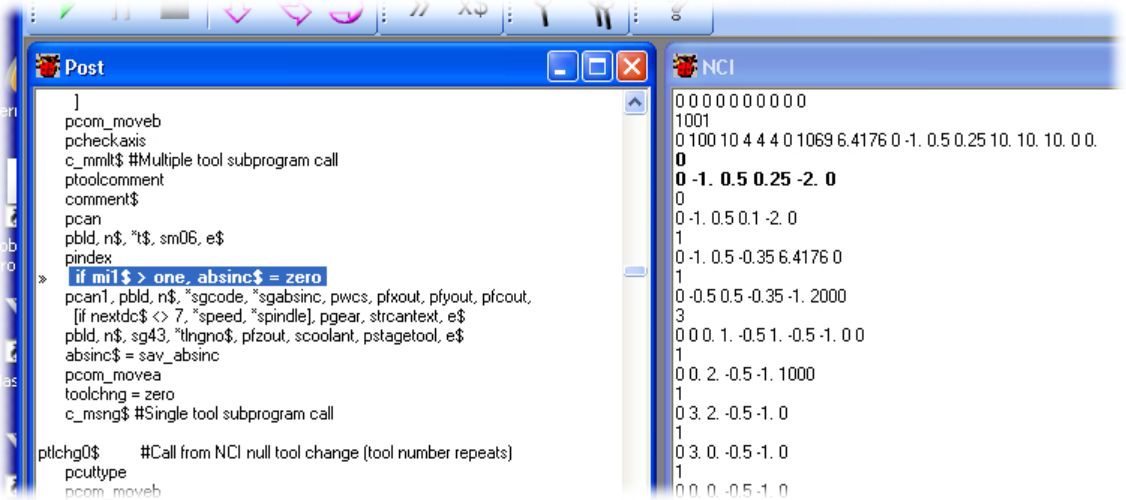
Example

Example 5: Tracing through the post and analyzing the debugger output

In the final example, you will step through sections of interest and look at the post logic and output in detail. Typically, the step and trace functions are used to look at only small sections of the post that you have identified after running once in **Turbo** mode.

While the debugger is paused at the first breakpoint, take a moment and look at the state of the output windows.

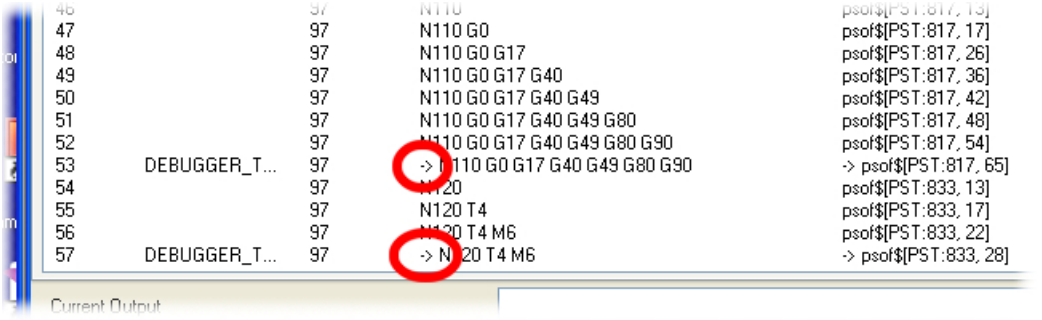
1 You can see that the **Post** window indicates where processing was paused. The **NCI** window shows what line was being processed at that point.



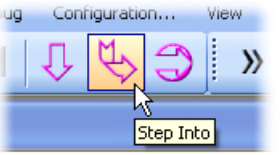
2 The **Watches** window shows the current value of the variables that we are monitoring. Variables whose values are different from their **`prv_`** values will appear in bold. Typcally, this is because their values have been set, but they have not yet been output. In this case, the **`prv_`** value of **`tlngno$`** is the last value read during the NCI pre-read.

3 The **NC** window shows the output for each line broken out by each individual output element.

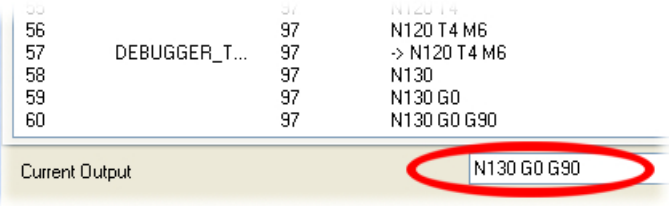
- Complete output lines are indicated by **->** symbols.
- This feature is triggered by the **Show expanded NC output** option. You will not see this, however, in **Run Turbo** mode.



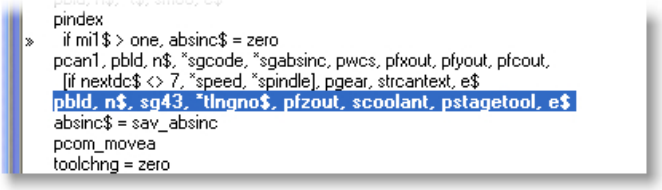
4 Click the **Step Into** button on the toolbar to execute PST instructions one at a time.



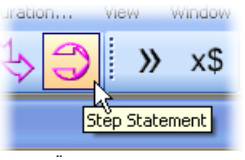
As output instructions are executed, you can see the current output line being built, element by element, in the **Current Output** field in the **NC** window.



5 When you reach the line in which the G43 is output…

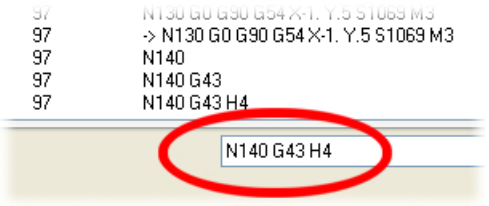


…use the **Step Statement** button on the toolbar to execute each postline one statement at a time.



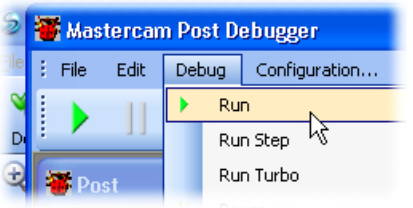
This is the most detailed trace mode.

6 When the G43 and offset code are output…



… the **prv_** values in the **Watches** window update, and the entries are no longer bold.

7 When the postline has completely finished executing, select **Run** from the **Debug** menu to run to the next breakpoint.



8 Repeat steps 4–5 to analyze the G43 output at the toolchange, and **Run** the post to finish the job.

9 When the post has finishing running, do one of the following:

- Open the PST file in an editor to make changes, save them, and repost. Hopefully you now know where in your post to make changes if you want to alter the processing related to the G43 output. Mastercam will maintain the debugger state and watch list. When you return to the debugger, there will be a third tab in the **NC** window.
- Go back to Mastercam to edit the part and repost.
- Select **File > Exit** to close the debugger and end the session.


Quick Start

chapter 3

Reference

This chapter provides a reference to all the debugger interface elements:

Debugger windows

The Post Debugger displays information in seven windows:

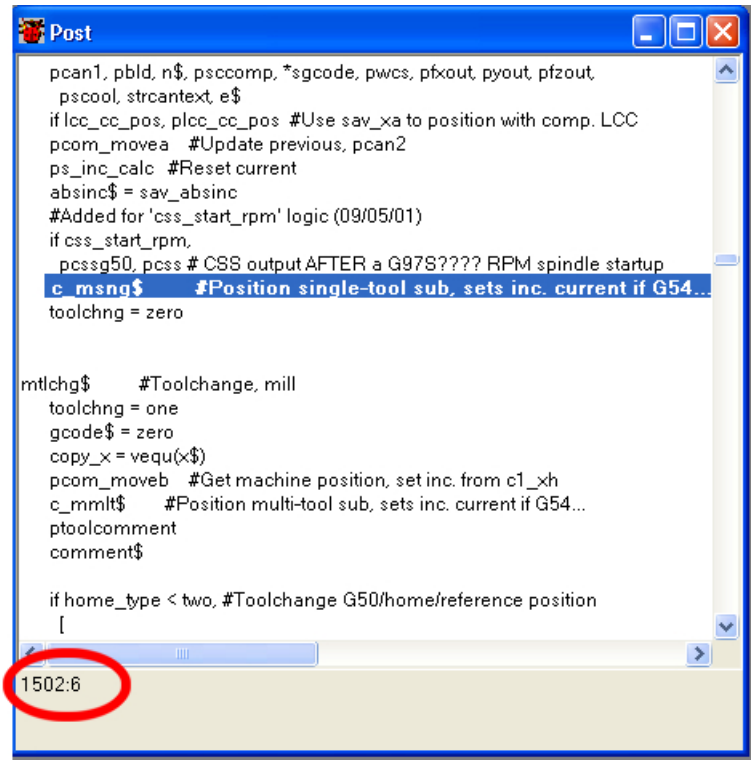
You can choose to show or hide any of these. Read the following sections to learn more about them.

See **Customizing the screen display** on page 6 to learn about showing/hiding windows.

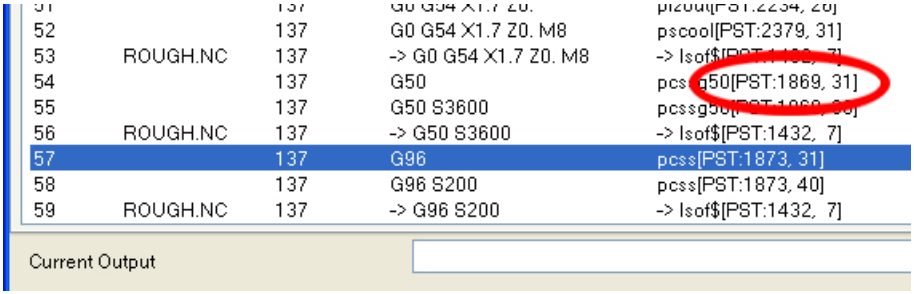
Post window

See **Editing your post** on page 24 to learn how to make changes to your PST file.

The **Post** window displays the currently active post (PST) file. As the NCI lines are processed in succession, the currently active post line appears in bold.



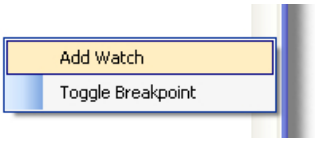
The status area at the bottom of the window displays the line number and column position of the active line. These are keyed to the **NC** window and **History** window:



The **Post** window also has the following features:

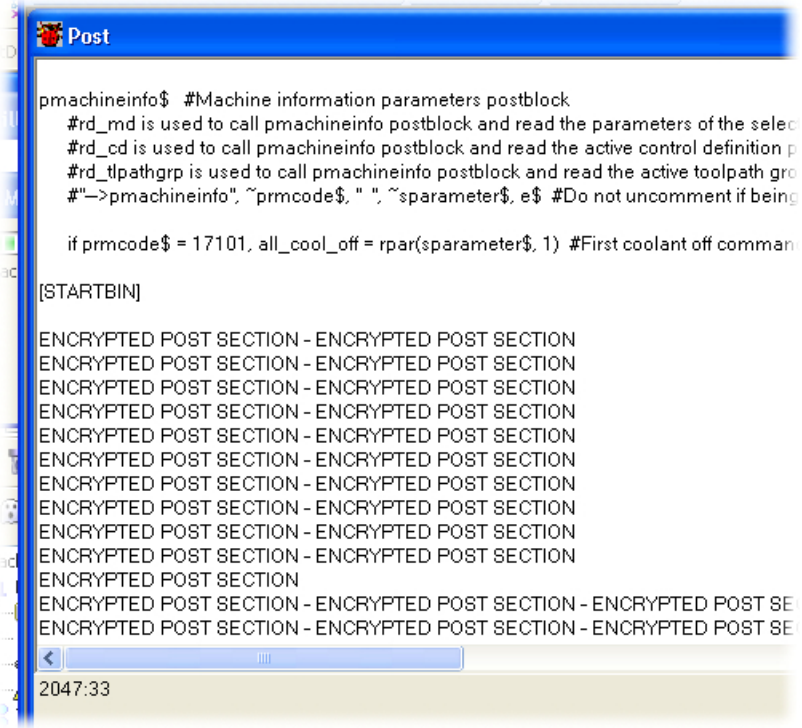
See **Using breakpoints** on page 15 and **Watching variables** on page 18 to learn more.

- The **Find** (**Ctrl + F**) and **Find Next** functions are available, so you can search for text.
- A right-click menu lets you add variables to the watch list or set breakpoints.



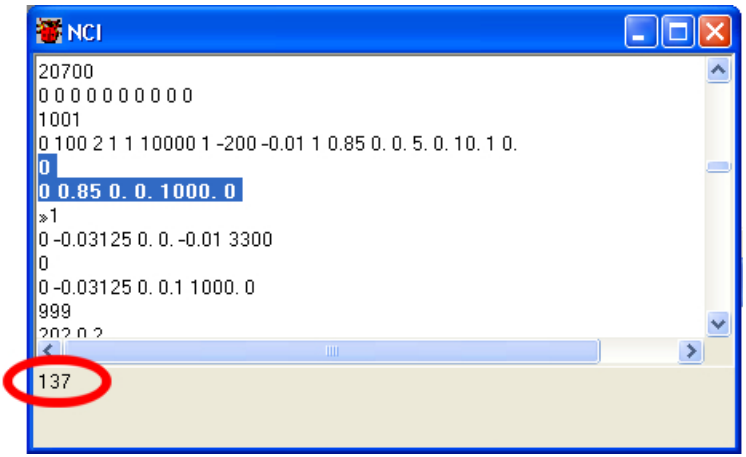
Encrypted or “binned” posts

You can use the debugger to work with posts that have been encrypted, or “binned.” Unencrypted portions appear in the **Post** window normally. Encrypted portions are hidden by placeholder text, as shown in the following picture. The trace and step functions will also not show encrypted sections of your post

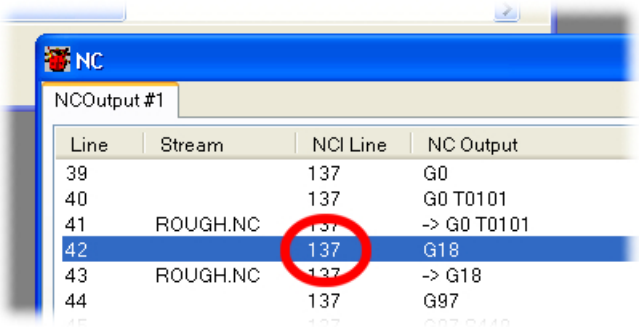


NCI window

The **NCI** window displays the ASCII NCI data for your part file. This data is always generated and displayed in this window, even if you do not have the **NCI file > Edit** option selected when you post the toolpaths. The current line is bolded.



The status area at the bottom of the window displays the line number of the current line. (Each line in each two-line pair is numbered individually.) This is keyed to the **NC** window as shown below:

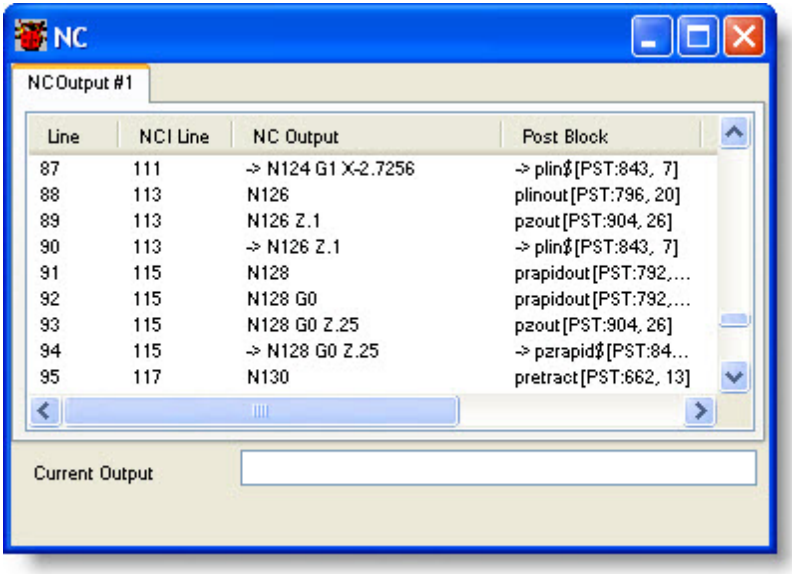


- Double-click the NCI line in the **NC** window and the **NCI** window will jump to that line.
- The breakpoint function is available in the NCI window, so you can set breakpoints and pause debugger execution at any NCI line you choose.

See **Using breakpoints** on page 15 to learn more.

NC window

The NC window displays the NC code the post is generating, along with other information such as the NC line number and the number of the NCI code line that produced the output.



The code that you see in the window is being generated by the debugger for display/debugging purposes, but it is not actually being written to the NC file. The NC file will not be written until the debugger session ends and control is handed back to MP.

IMPORTANT: There are instances where the code written to the NC file is different than the code that appears in the NC window. The most common example is canned cycles, where the debugger window often displays more information than is actually written to the file.

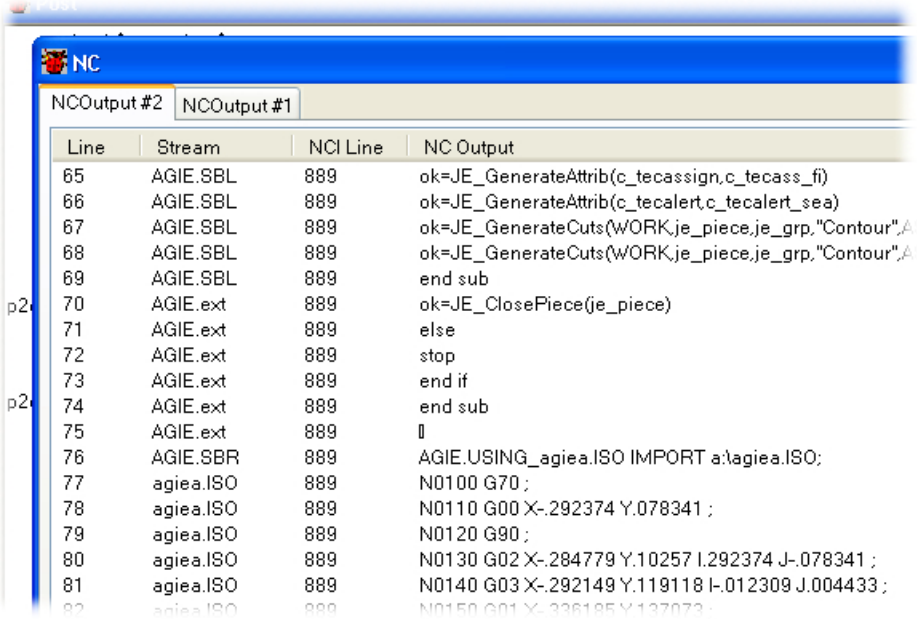
You can choose to display up to seven columns of information:

Line—The number of the line in the NC window

Machine Type—This is for future use. It is hidden by default.

Stream—This displays the stream in which the line was output. For most applications, this is simply the name of the NC file.

If your PST file is outputting to multiple files, you can see that information here. For example, this Agievision post writes its NC output to several different files: SBL, ISO, and more. The debugger clearly identifies where each piece of data is written:

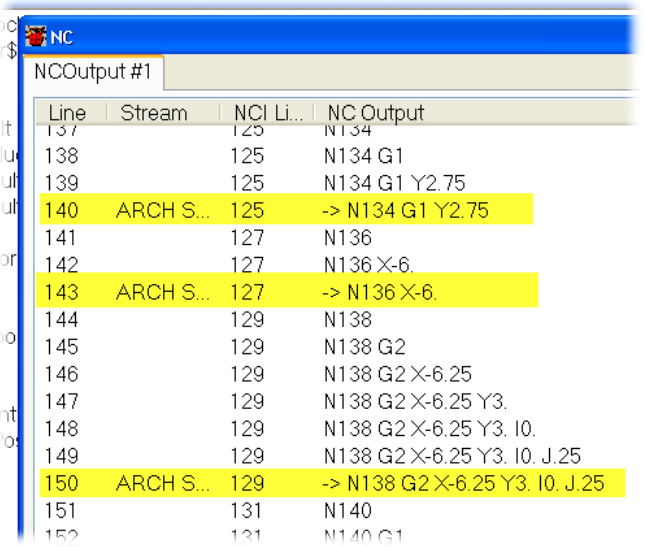


Unless you are using buffers or outputting to multiple files, you can typically hide this column.

NCI Line—The number of the last NC line that the post processor read. This is not necessarily the NCI line that produced the output. It's just the most recent line that the debugger read from the NCI file.

NC Output—What has been generated by your output postblocks—typically, to be written to your NC file. With Expanded NC Output turned on, each buffered item will be listed on its own line as it is output. When the buffer is cleared and the line is finally written to the

file, you will see the complete line preceded by a **->** symbol, as shown in the following picture.



Post Block—The post block (including the line and column number in the post file) that produced the NC output.

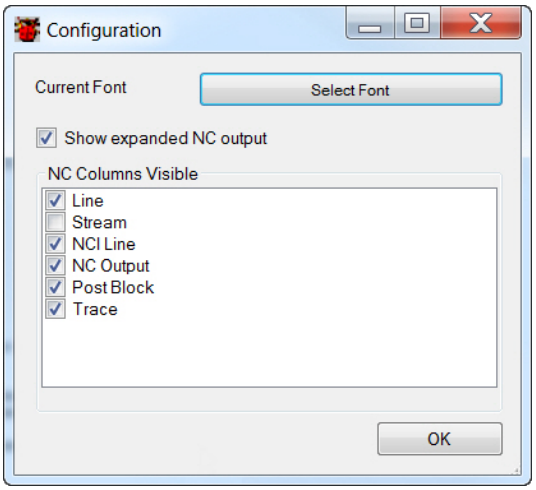
Trace—The post blocks that generated the NC output (top-level post blocks only).

Double-click on a NCI line number or postblock name to jump there.

Customizing the NC window

There are number of options available to customize how information is displayed in the NC window.

First, use the **Configuration** command from the menu to decide which columns you want to see:



Best practice—*For most applications, the column layout shown above will work fine. For most applications, the* **Trace and Machine Type** *options are not useful.*

Once you decide which columns to display, you can:

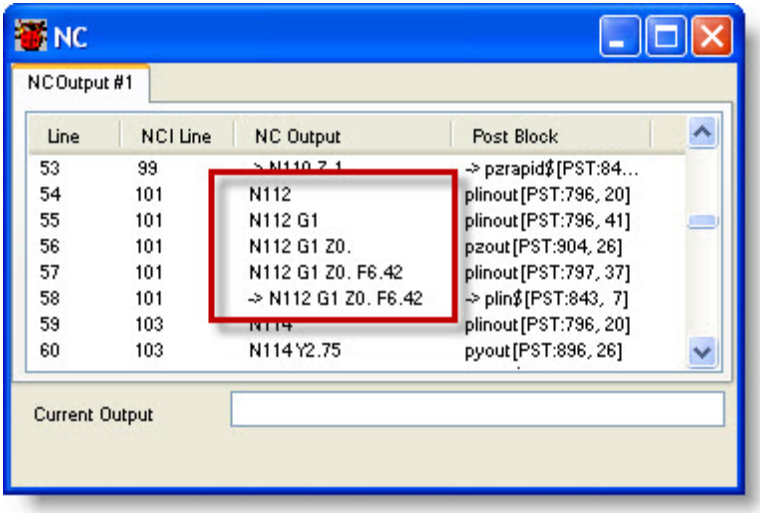
- Resize columns by dragging the column boundaries.
- Re-order columns by dragging-and-dropping them.

What is "Expanded NC output" ?

The **Show expanded NC output** option specifies whether the NC window shows each NC line in each step of its generation or just the final line.

For example, Figure 3-1 shows the **NC** window's contents when **Show expanded NC output** is selected. In the **NC Output** column, you can see how the NC line was generated, step by step. The line prefaced with the **->** symbol is where your post would finally write the complete line to the NC file.

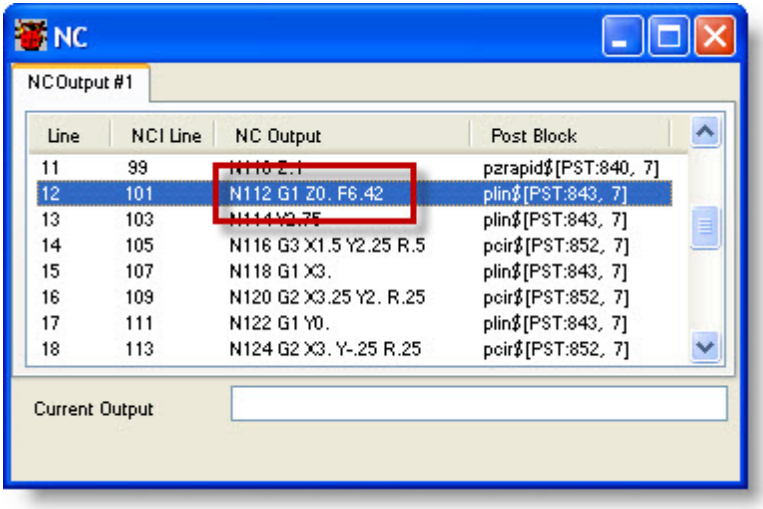
Figure 3-1: With expanded NC output



Quick Start

Figure 3-2 shows the same NC window with **Show expanded NC output** turned off. Now the window displays only complete NC lines.

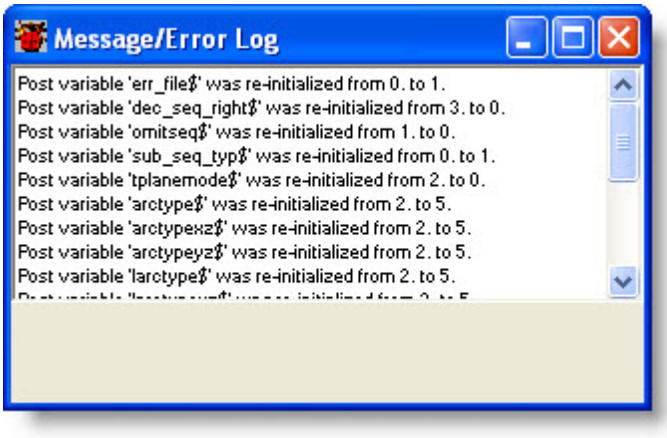
Figure 3-2: Without expanded NC output



NOTE: Run Turbo mode does not show expanded NC output.

Message/Error Log window

The **Message/Error Log** window shows messages about the post being processed, as well as errors that occur.



NOTE: This window typically mirrors the contents of the .ERR file.

Watches window

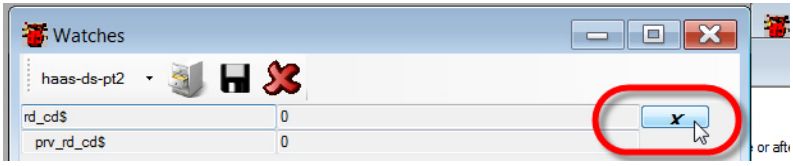
See **Watching variables** on page 18 to learn more about adding variables to the watch list.

The **Watches** window lists all the variables that you have added to the watch list, together with their **prv_** variables.

- Variables whose values are different from their **prv_** values are shown in **bold**. Typically, these values will be displayed unformatted.
- Once the variable has been output, the **prv_** value is updated so that the two values are equivalent. The values will be shown unbolded, and will reflect the format statement with which it was output.

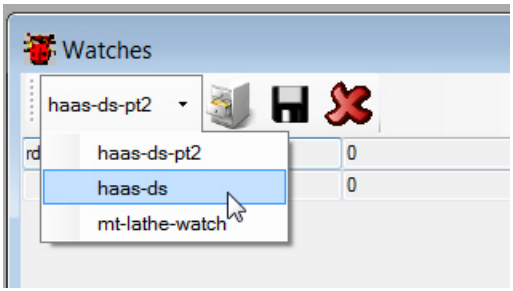
To clear the watch list, click the **Clear** button on the toolbar: 

To delete single variables, click the **Delete** button at the end of each line. You might need to make the window wider to see it:



The debugger also lets you save watch lists. Use the **Select path** button in the **Watches** toolbar to tell Mastercam where your watch lists are stored. When you select a folder with this button, Mastercam will

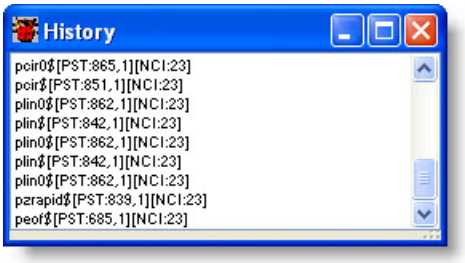
populate the list on the toolbar with all the watch lists that are available in that folder:



Load a different watch list by selecting it from the list. See **Saving and loading watch lists** on page 21 and **Working with watch list files** on page 23 to learn more.

History window

The **History** window lists all the postblocks that have been executed in the current debugging session.

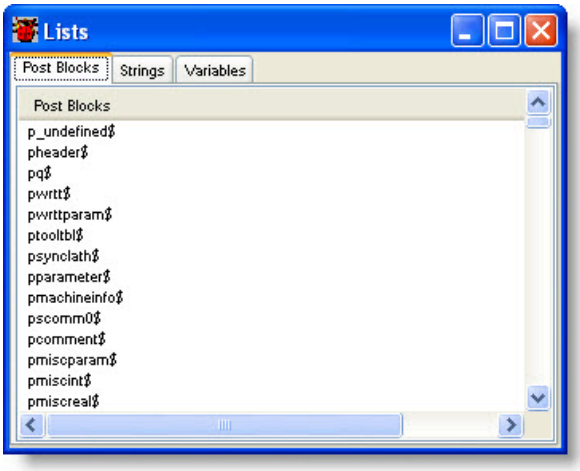


Each line in the **History** window includes the current NCI line number at the time the postblock was executed, as well as the line number of the postblock in the PST file. Double-click on either to jump there in the **NCI** or **Post** window.

NOTE: The **History** window is only populated when you use **Run Step** mode.

Lists window

The **Lists** window lists all of the postblocks, strings, and numeric variables in the current PST file.



Typically you do not need to display this window while the post is running. You can toggle it on anytime if you want to look at it.

Menu commands

This table lists all of the debugger commands.

Table 1: File menu

Command	Description	Shortcut
Quit	Terminates the current session, but leaves the debugger open. Use this command when you want to return to Mastercam and re-post. This option will preserve your watch list and breakpoints, and create a new tab for the next run of NC output so you can compare the results of each session.	
Exit	Terminates the current session, and closes the debugger.	

See **What happens when the post is finished running?** on page 9 for more information.

.

Table 2: Edit menu

Command	Description	Shortcut
Find	Searches for occurrences of a given text string	Ctrl+F
Find Again	Searches for the next occurrence of the previously searched text string	F3

Table 3: Debug menu

Command	Description	Shortcut
Run	Runs quickly through the post.	F5
Run Step	Steps through the post more slowly than the **Run** command, but shows the most information.	
Run Turbo	Runs through the post at the fastest possible speed, but without updating the **History** window. Expanded NC output is not available.	
Pause	Pauses the debugger's processing.	F12

See **Understanding the different run options** on page 8 for more information.

Table 3: Debug menu

Command	Description	Shortcut
Stop	Stops the debugger's processing. ▪ Ends the debugger session. You can return to Mastercam. ▪ Transfers control to editor. The NC file only shows output to stop point. ▪ Must re-post from inside Mastercam to resume debugging.	Shift+F5
Step Over	Executes all of the commands that are part of the highlighted post block, or if not on a post block, executes the current line.	F10
Step Into	Enters the currently highlighted post block, or if not on a post block, executes the current line.	F11
Step Statement	This is more detailed than **Step Into**. It moves forward through the post one statement at a time. You can see individual elements of each postline executed, element by element.	Ctrl+F11
Toggle Breakpoint	Adds or removes a breakpoint from the currently selected line.	F9
Add Watch	Adds the currently highlighted variable to the **Watches** window.	Ctrl+W
Remove All Breakpoints	Removes all breakpoints from the debugger windows.	Ctrl+Shift+F9

.

Table 4: Configuration menu

Command	Description	Shortcut
Configuration	Displays the **Configuration** dialog box	

See **Customizing the screen display** on page 6 for more information.

.

Table 5: View menu

Command	Description	Shortcut
Watch	Shows or hides the **Watches window**.	
History	Shows or hides the **History window**.	
Lists	Shows or hides the **Lists window**.	
Post	Shows or hides the **Post window**.	
NCI	Shows or hides the **NCI window**.	
NC	Shows or hides the **NC window**.	
Message/Error Log	Shows or hides the **Message/Error Log window**.	

.

Table 6: Window menu

Command	Description	Shortcut
Cascade	Arranges the **Post**, **NCI**, **NC**, and **Message/Error Log** windows so that they diagonally overlap.	
Horizontal	Arranges the **Post**, **NCI**, **NC**, and **Message/Error Log** windows horizontally in a grid.	
Vertical	Arranges the **Post**, **NCI**, **NC**, and **Message/Error Log** windows in a vertical stack.	
[Window List]	Brings the selected window to the foreground.	

.

Table 7: Help menu

Command	Description	Shortcut
User Guide	Displays this document	
About Mastercam Post Debugger	Tells you about the current version of the debugger.	

Toolbar

The toolbar lets you quickly access the most frequently used menu commands. When all the toolbar buttons are grayed out, that means the debugger has either finished processing or has been stopped. The only way to resume processing is to go back to Mastercam and re-post.

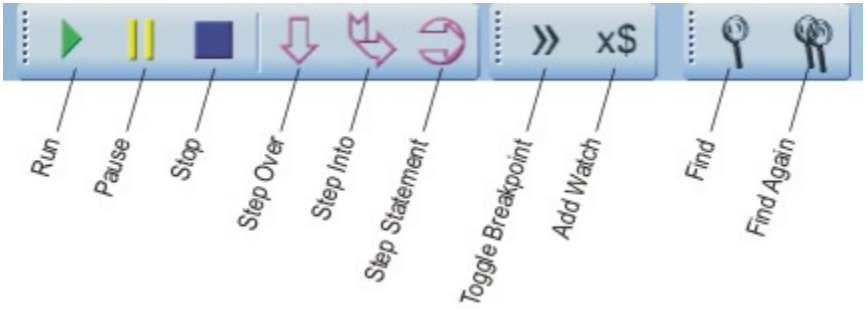


Table 8: Toolbar commands

Command	Description
Run	Runs quickly through the post.
Pause	Pauses the debugger's processing.
Stop	Stops the debugger's processing.
Step Over	Executes all of the commands that are part of the highlighted post block, or if not on a post block, executes the current line.
Step Into	Enters the currently highlighted post block, or executes the current line.
Step Statement	Moves forward through the post one statement at a time. Individual postlines are executed one element at a time. This is the most detailed step mode.
Toggle Breakpoint	Adds or removes a breakpoint from the currently selected line.
Add Watch	Adds the currently highlighted variable to the Watches window.
Find	Searches for occurrences of a given text string.
Find Again	Searches for the next occurrence of the previously searched text string.

Keyboard shortcuts

For many of the debugger's commands, you can use the keyboard shortcuts shown here.

Table 9: Keyboard shortcuts

Command	Keystroke
Add watch	Ctrl+W
Pause	F12
Run	F5
Find	Ctrl+F
Find next	F3
Select all	Ctrl+A
Step into	F11
Step over	F10
Step statement	Ctrl+F11
Stop	Shift+F5
Toggle breakpoint	F9
Clear all breakpoints	Ctrl+Shift+F9


Quick Start