**Screen Shots of the Tabs on the main form for ‘Chip’ application**

If you see a button that has a magenta background, that means that this button has not been implemented yet.



Note also that at present almost nothing but the G Code and Jog tabs work. This application is a combination of three other separate applications brought together as one. The code is done for the different functions, but they have not been intenerated together yet. There’s a Lot of work left, and there may be major changes to what is show in this document.

The plan is to finish each tab and make that available in a release. There will also be a setup tab for hiding or making tabs visible.

This application was developed so that button and text boxes can be easily added and their events handled. The code is modularized in classes and controls. When a control is added, only one class that needs to be changed.

One problem with event type programming is that code gets separated and scattered around. A button for example may have one routine to initialize it, another for setting persistent data (like a text box), another for saving persistent data, others for events like mousedown and mouseup, state change routines etc. When you add a control you have to remember to change all these routines.

This software is organized so that all functions for a control, like a button for example, are in one routine (handler) that gets called on any event, initialization, persistent data functions, mouse events etc. It somewhat works like Windows events, where the event and parameters are sent to the event handler. User written event handlers can handle one button, or a group of related buttons (like jog, or jog select) in one routine. Part of the control information is the handler that needs to be called on an event.

A case statement can separate out the event type, and another Case statement can separate out the control (or control name).

Adding a control like a button only requires adding the control to one of the tabs, or top panel and writing one handler. All event handlers have the same calling parameters. The control that the event is for and the event type. There are some problems with this and this will change slightly, but the principle will be the same. Note that each button could have been done with separate handlers. The choice is up to the user. Shown on the next two pages is the routine that handles all six jog buttons.

Public Function Jog\_Buttons(ByRef Ctrl As Class\_Display\_Control, ByVal Evnt As Class\_Display\_Handlers.enum\_Event) As Object

Select Case Evnt

Case enum\_Event.Initialize

Select Case Ctrl.Name

Case "btn\_Jog\_X\_Plus"

Ctrl.Add\_Image("Arrow\_Right.png", "Up")

Ctrl.Add\_Image("Arrow\_Right\_Pressed.png", "Down")

Ctrl.State = "Up"

Case "btn\_Jog\_X\_Minus"

Ctrl.Add\_Image("Arrow\_Left.png", "Up")

Ctrl.Add\_Image("Arrow\_Left\_Pressed.png", "Down")

Ctrl.State = "Up"

Case "btn\_Jog\_Y\_Plus"

Ctrl.Add\_Image("Arrow\_Up.png", "Up")

Ctrl.Add\_Image("Arrow\_Up\_Pressed.png", "Down")

Ctrl.State = "Up"

Case "btn\_Jog\_Y\_Minus"

Ctrl.Add\_Image("Arrow\_Down.png", "Up")

Ctrl.Add\_Image("Arrow\_Down\_Pressed.png", "Down")

Ctrl.State = "Up"

Case "btn\_Jog\_Z\_Plus"

Ctrl.Add\_Image("Red\_Arrow\_Up.png", "Up")

Ctrl.Add\_Image("Red\_Arrow\_Up\_Pressed.png", "Down")

Ctrl.State = "Up"

Case "btn\_Jog\_Z\_Minus"

Ctrl.Add\_Image("Red\_Arrow\_Down.png", "Up")

Ctrl.Add\_Image("Red\_Arrow\_Down\_Pressed.png", "Down")

Ctrl.State = "Up"

Case "chk\_Soft\_Limits"

Ctrl.State = True

End Select

Case enum\_Event.Left\_Mouse\_Down

If State\_Is("btn\_Jog\_Rate\_Big\_Step", "On") Or State\_Is("btn\_Jog\_Rate\_Little\_Step", "On") Then

Dim Increment As Int32 = 0

Dim N As Single = 0

If Get\_State("btn\_Jog\_Rate\_Big\_Step") = "On" Then

N = Get\_Number("txt\_Jog\_Rate\_Big\_Step")

Else

N = Get\_Number("txt\_Jog\_Rate\_Little\_Step")

End If

Increment = Kflop.Counts(N)

Select Case Ctrl.Name

Case "btn\_Jog\_X\_Plus"

Kflop.X\_Step(Increment)

Case "btn\_Jog\_X\_Minus"

Kflop.X\_Step(-Increment)

Case "btn\_Jog\_Y\_Plus"

Kflop.Y\_Step(Increment)

Case "btn\_Jog\_Y\_Minus"

Kflop.Y\_Step(-Increment)

Case "btn\_Jog\_Z\_Plus"

Kflop.Z\_Step(Increment)

Case "btn\_Jog\_Z\_Minus"

Kflop.Z\_Step(-Increment)

End Select

Else 'Continuous jog

Select Case Ctrl.Name

Case "btn\_Jog\_X\_Plus"

Kflop.X\_Jog(Kflop.Jog\_Feedrate\_Counts)

Case "btn\_Jog\_X\_Minus"

Kflop.X\_Jog(-Kflop.Jog\_Feedrate\_Counts)

Case "btn\_Jog\_Y\_Plus"

Kflop.Y\_Jog(Kflop.Jog\_Feedrate\_Counts)

Case "btn\_Jog\_Y\_Minus"

Kflop.Y\_Jog(-Kflop.Jog\_Feedrate\_Counts)

Case "btn\_Jog\_Z\_Plus"

Kflop.Z\_Jog(Kflop.Jog\_Feedrate\_Counts)

Case "btn\_Jog\_Z\_Minus"

Kflop.Z\_Jog(-Kflop.Jog\_Feedrate\_Counts)

End Select

End If

Ctrl.State = "Down"

Case enum\_Event.Left\_Mouse\_Up

If State\_Is("btn\_Jog\_Rate\_Big\_Step", "Off") And State\_Is("btn\_Jog\_Rate\_Little\_Step", "Off") Then

Select Case Ctrl.Name

Case "btn\_Jog\_X\_Plus", "btn\_Jog\_X\_Minus"

Kflop.X\_Jog(0)

Case "btn\_Jog\_Y\_Plus", "btn\_Jog\_Y\_Minus"

Kflop.Y\_Jog(0)

Case "btn\_Jog\_Z\_Plus", "btn\_Jog\_Z\_Minus"

Kflop.Z\_Jog(0)

End Select

End If

Ctrl.State = "Up"

Case enum\_Event.Check\_On

Ctrl.State = True

Case enum\_Event.Check\_Off

Ctrl.State = False

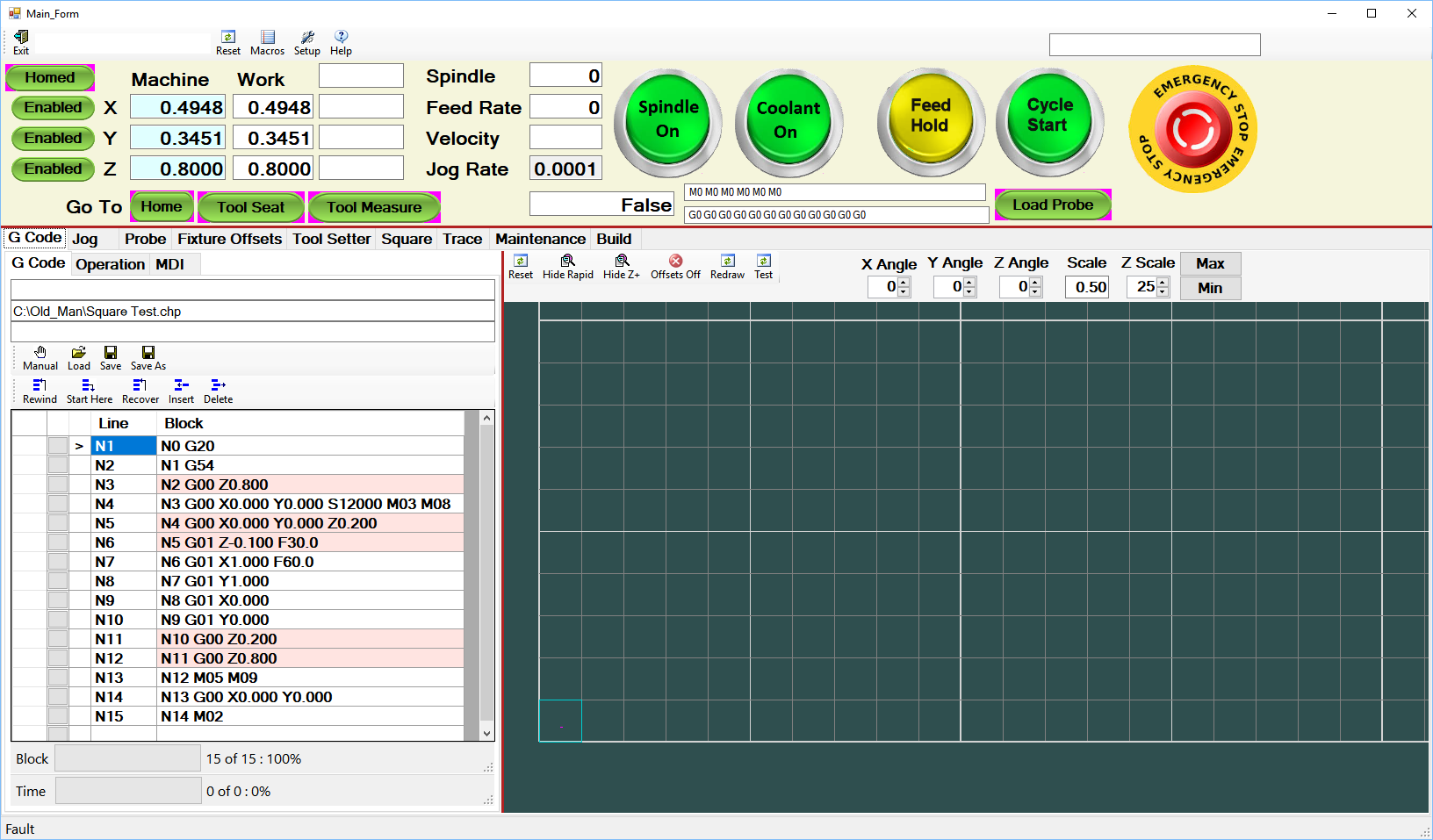
End Select

Return Nothing

End Function

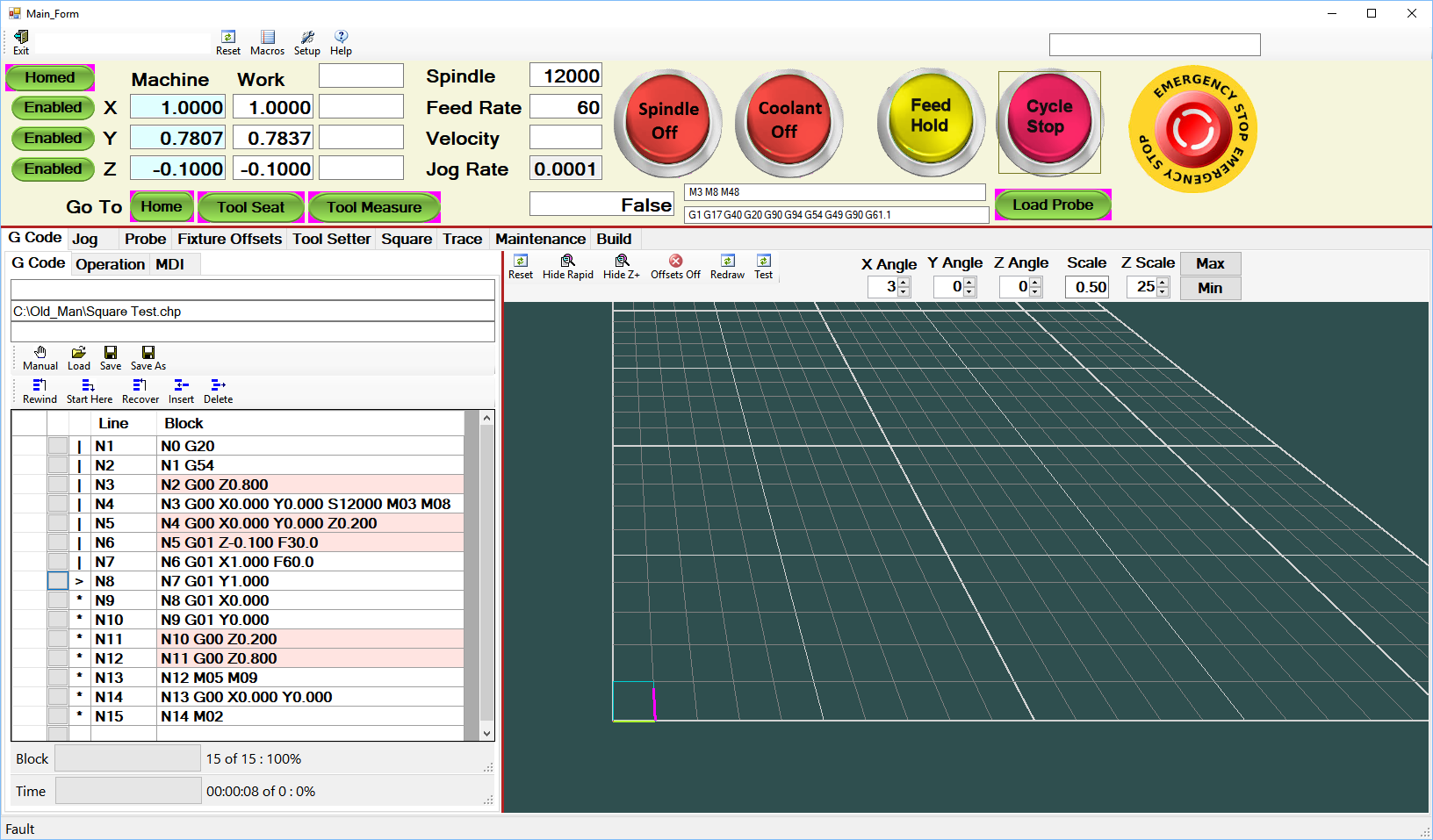
**Top Panel**

The upper panel with the axis position and cycle buttons is always visible. The third column next to Work position is the current coordinate system in the top box (eg. G55) and the boxes below are the offsets (not implemented yet). Also shown are programmed Spindle speed, programmed Feed Rate, Velocity (haven’t figured this one out yet) and jog rate/step, current Active M and G codes.



**G Code Tab**

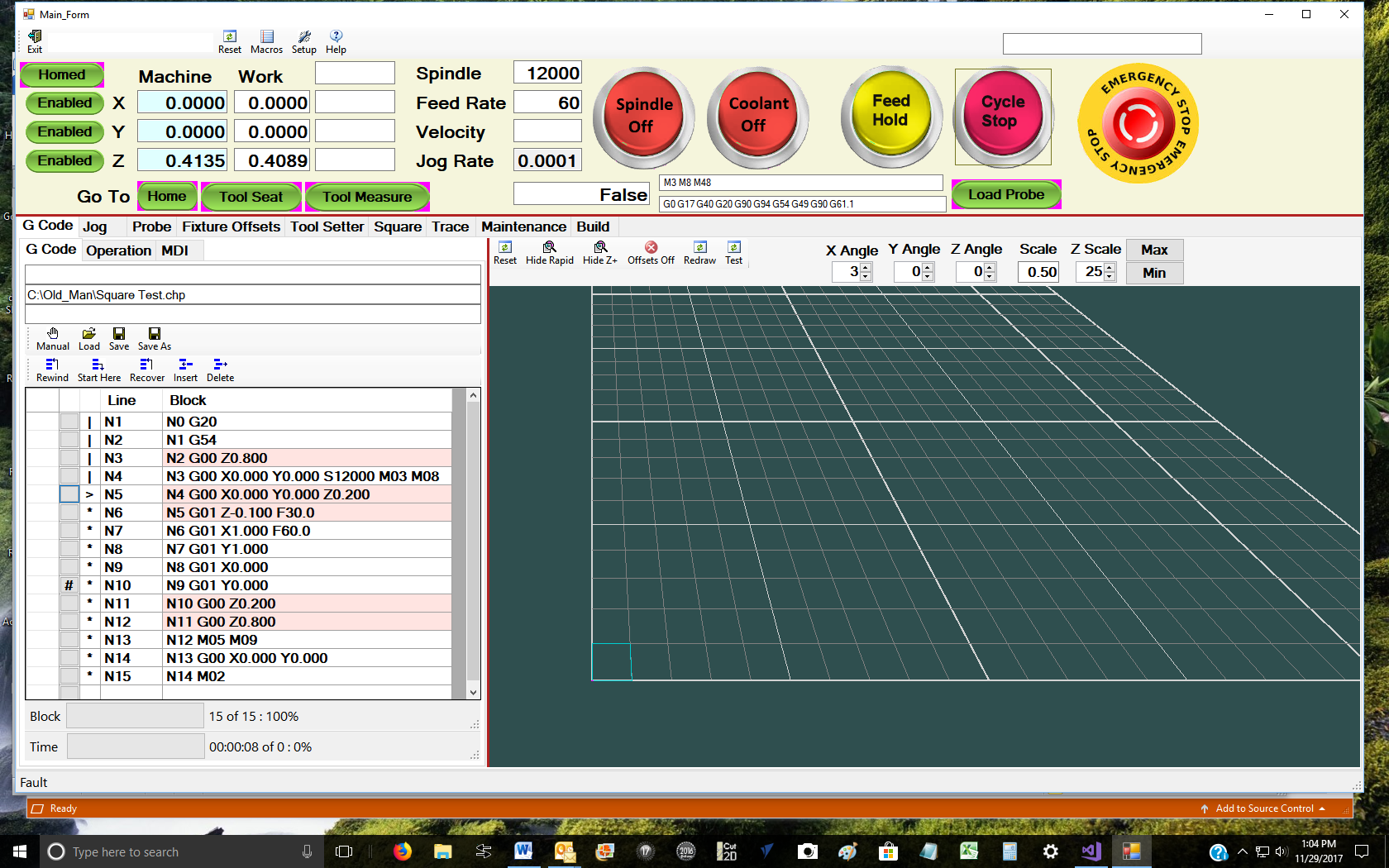
This tab shows the currently loaded G Code program and the 3D drawing of the part. The one shown below is a simple square test program. The drawing shows the table as a grid (one inch per square) in perspective. Note the colors on the part. The teal color is the uncut paths, the greenish/yellow color shows what has been cut, the magenta color shows the cut in progress. Not working (but will be) is the next block to be cut which will be in red. The drawing below is tilted slightly in the X direction. This part only has one Z level, but tilting X is handy for multiple pass profiles with different Z levels. They can be seen better with the axis tilted. The Z scale is handy to separate the Z levels more or less.



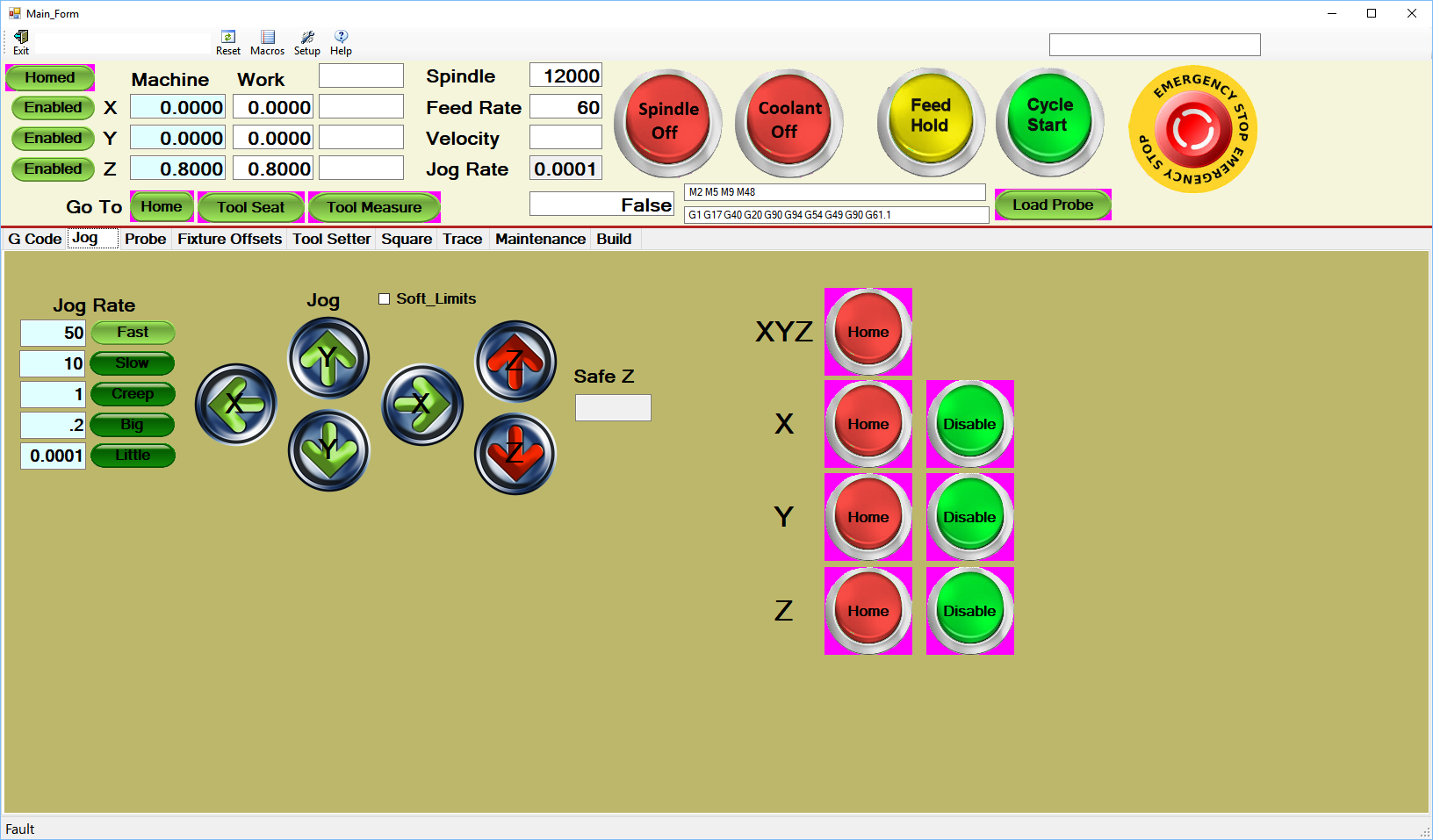
**G Code Panel**

The G Code is listed and several run or edit buttons are available. Note that there are three tabs in the G Code panel. The G Code tab is the currently loaded program, Operation will be described later but list the G Code programs for an operation. The MDI tab is just like the G Code tab. MDI normally is used for one or two moves, but it is just like the G Code tab and another program can be loaded and run there.

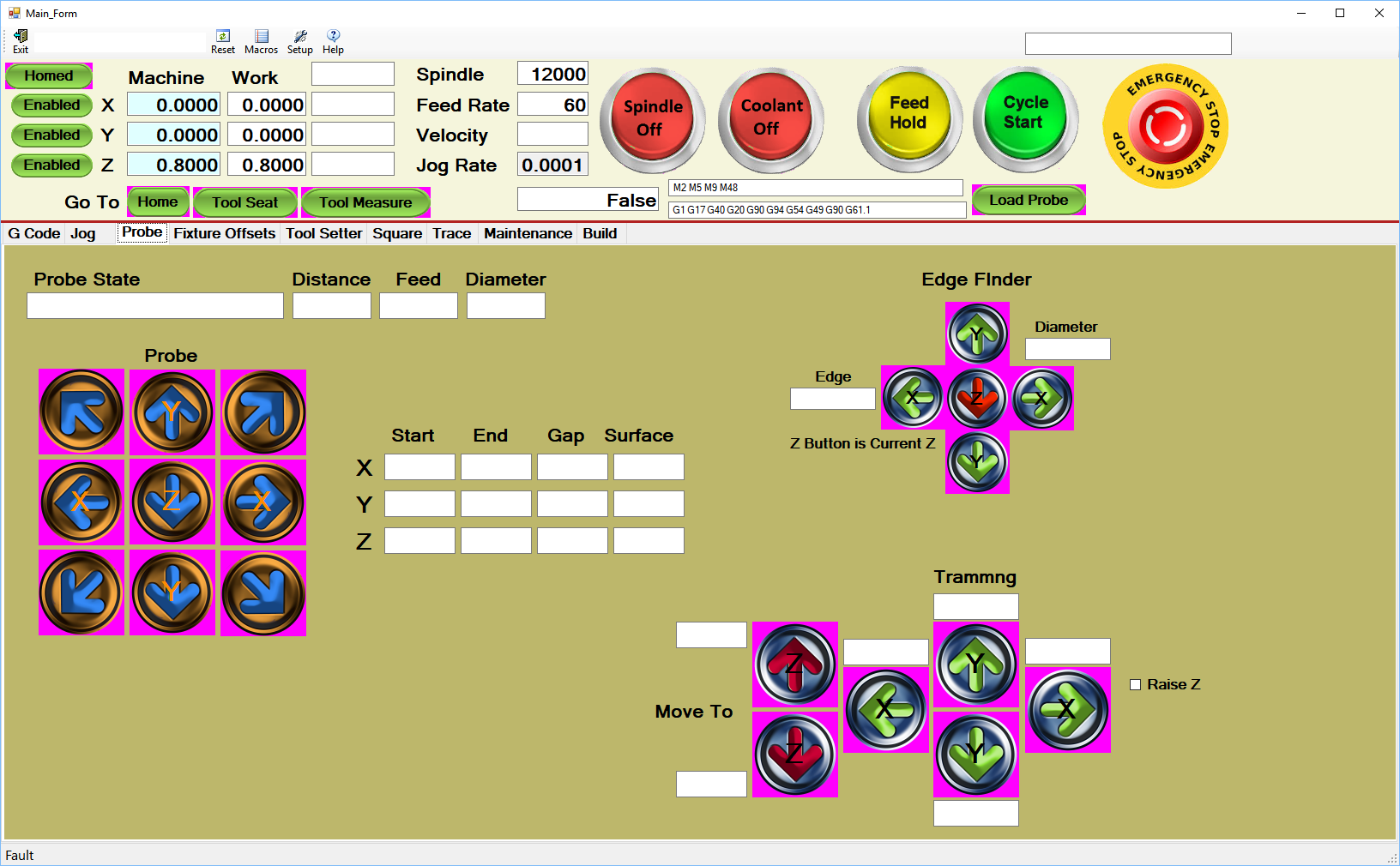
The first column is used to set a breakpoint or pause. When the program gets to this move it is paused. The second column shows what blocks have been executed (shown with a vertical bar |), the current block (shown with >), the blocks sent to the interpreter (shown with \*), and blocks not yet sent to the interpreter are blank. The program below is short so none of these show up. There are three modes of operation. In ‘Manual’ mode programs are loaded with the load button. ‘Semi-auto’, programs are loaded from the current operation and run, but after each program is run the next program is loaded but not run until cycle started. ‘Auto’ mode runs all the programs in the operation one right after the other. This is used to run multiple programs on a part, or multiple parts on a fixture. Just load the operation (described later) hit cycle start and the whole fixture of parts is machined.



**Jog and Home Panel**

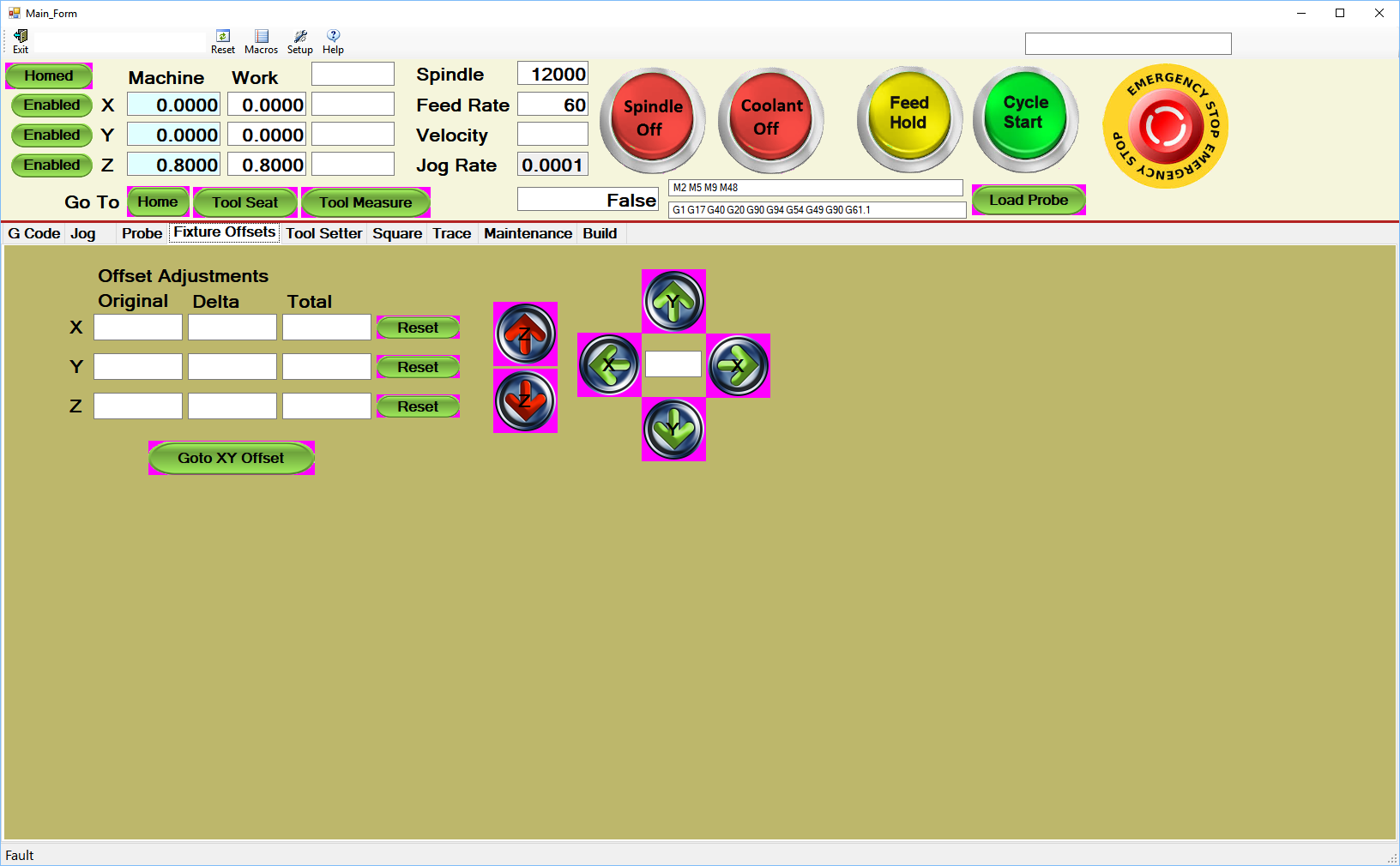


**Probe Panel**



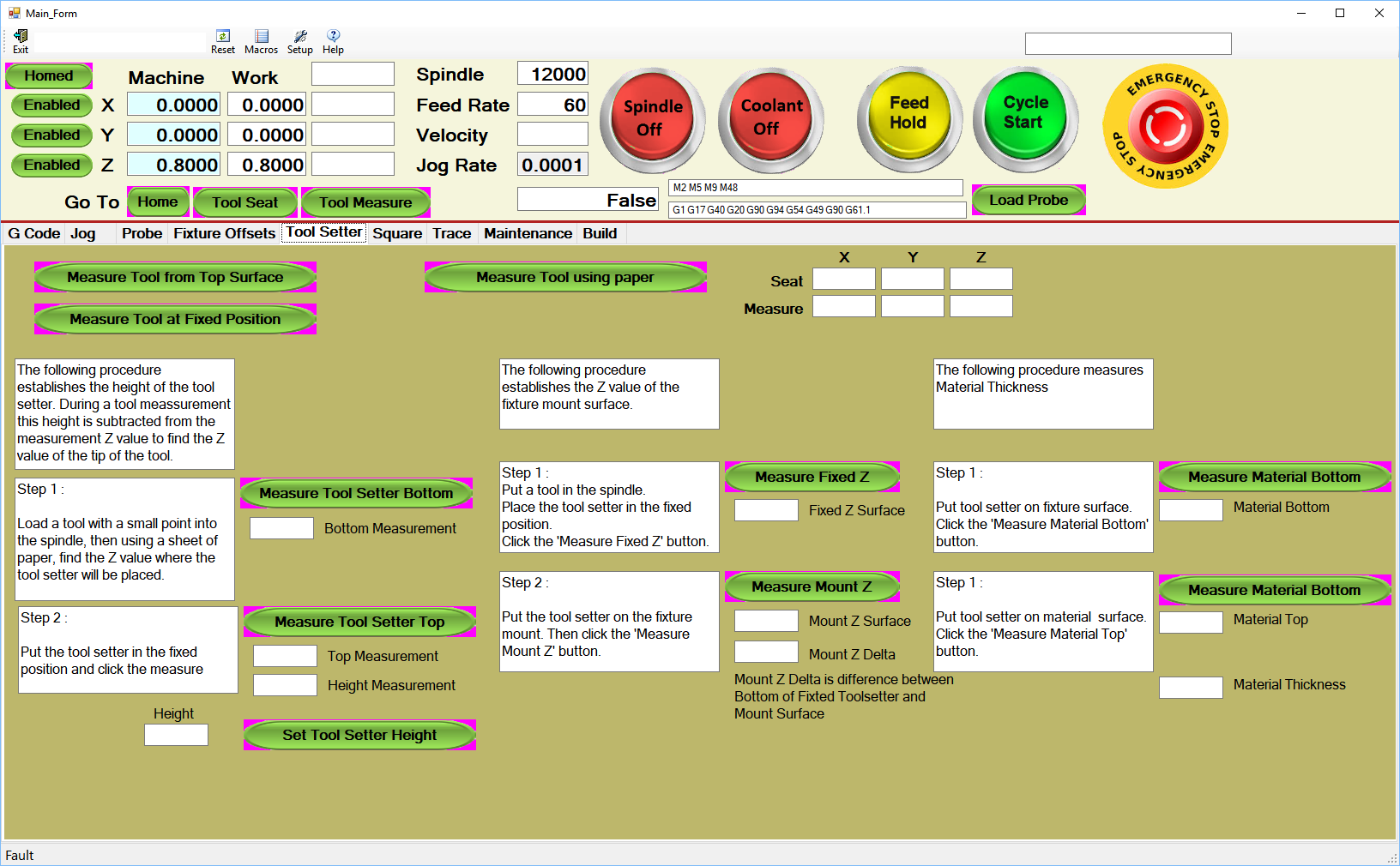
Offset Adjust Tab

Sometimes it is convent to adjust the fixture offsets temporarily, for example if you are doing a flattening or cleanup cut and want to adjust Z down a little bit at a time. This tab facilitates that for all three axis.



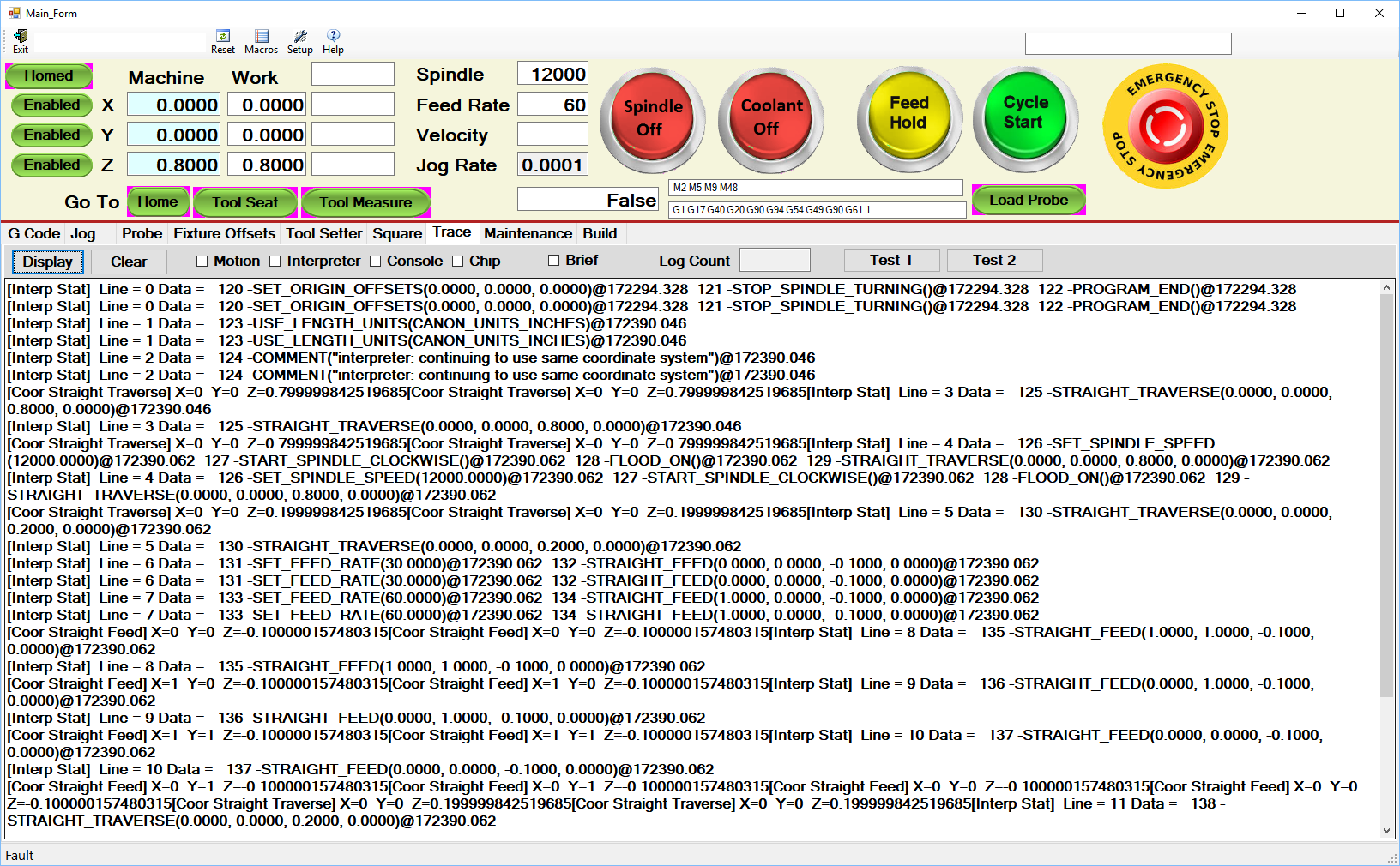
**Tool Setter Tab**

Tool setters can be a real pain to set up. This tab leads the user through setting up a fixed tool setter, used to set the Z offset using the bottom surface of a part and a movable tool setter that sets the Z offset using the top surface of the material. This tab also can be used to measure material thickness.



**Trace Tab**

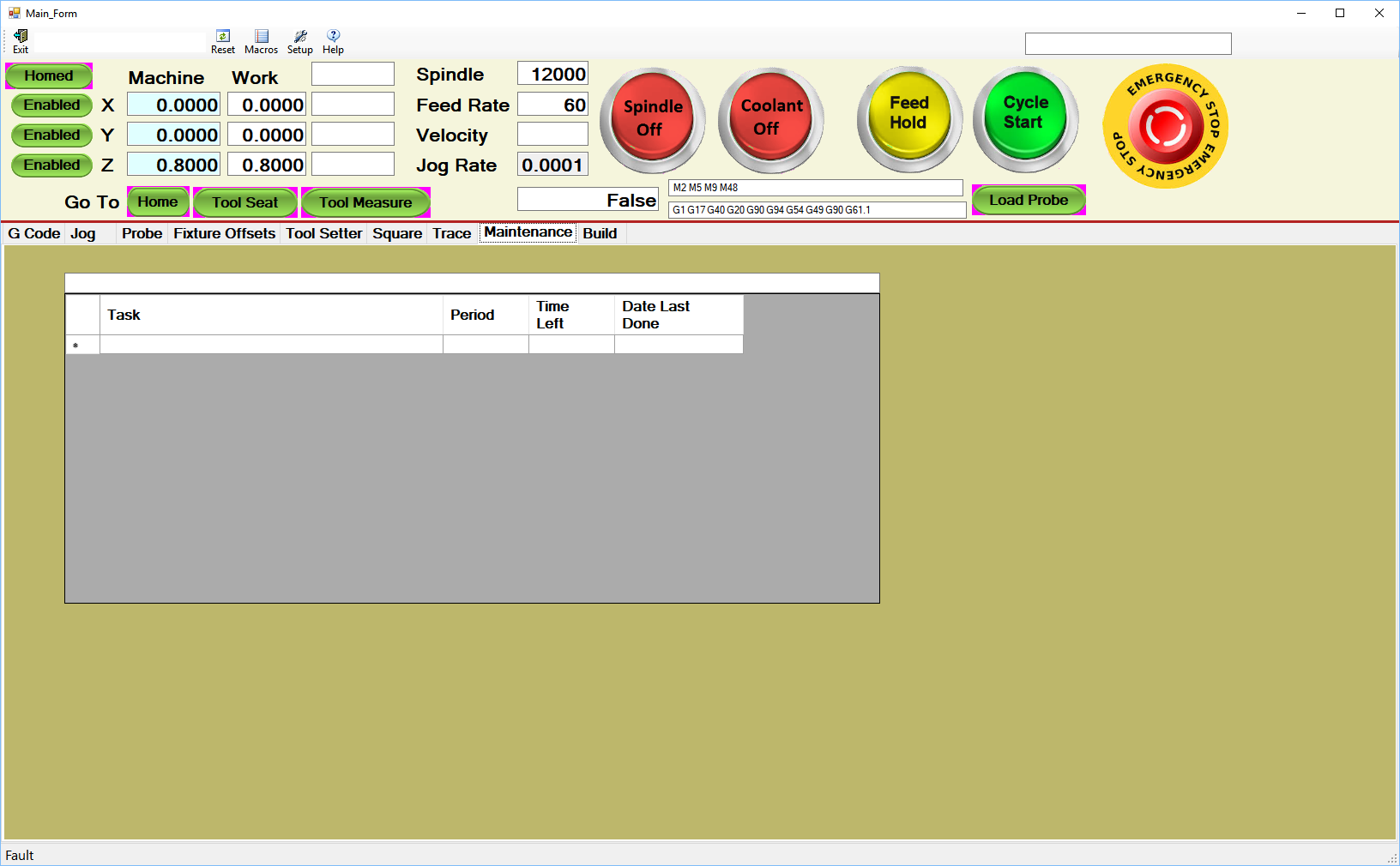
This tab is used for debugging. Messages from Coordinated Motion, Interpreter, Console and user generated messages can be logged. The Log Count box is the number of messages kept, so for example if it is set to 200 the last 200 messages will be saved. The check boxes are filters that can be used to show only the messages of interest. For example if Console and Chip are set only Console and user generated messages will be displayed.



**Maintenance Tab**

Maintenance can be scheduled by various means, like hours powered up, run hours, elapsed time. Etc. When maintenance is due a message alerts the operator.

I like to oil the slides and lead screws on a regular basis, and clean out the computer and controller fan filters etc. This reminds me to do so.

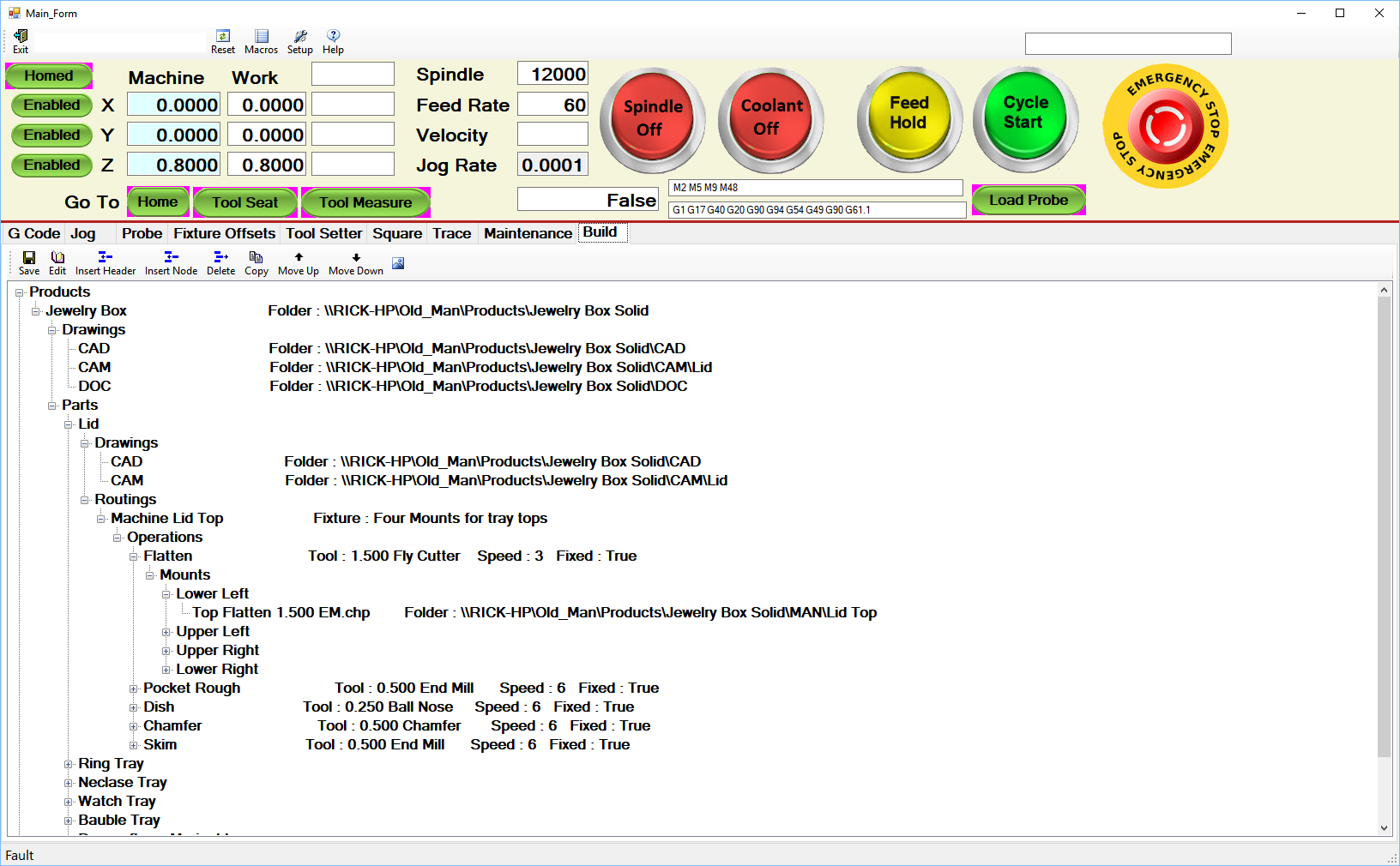


**Build Tab**

This tab has two functions. One is to organize all the CAD drawings, CAM sessions, G Code programs, Fixture information, Tooling information, documentation and anything else that has to do with building products. The other is to set up Routs and Operations within routes. Unless you have experience with manufacturing ERP systems with routing capability this is going to be hard to explain without a detailed example, but basically Routes are the way a part goes through manufacturing, ie. what machines does it go on, what manual operations are done, what tooling, what fixturing is needed etc. Routs consist of operations like what G Code to run, with what fixture and what tool.

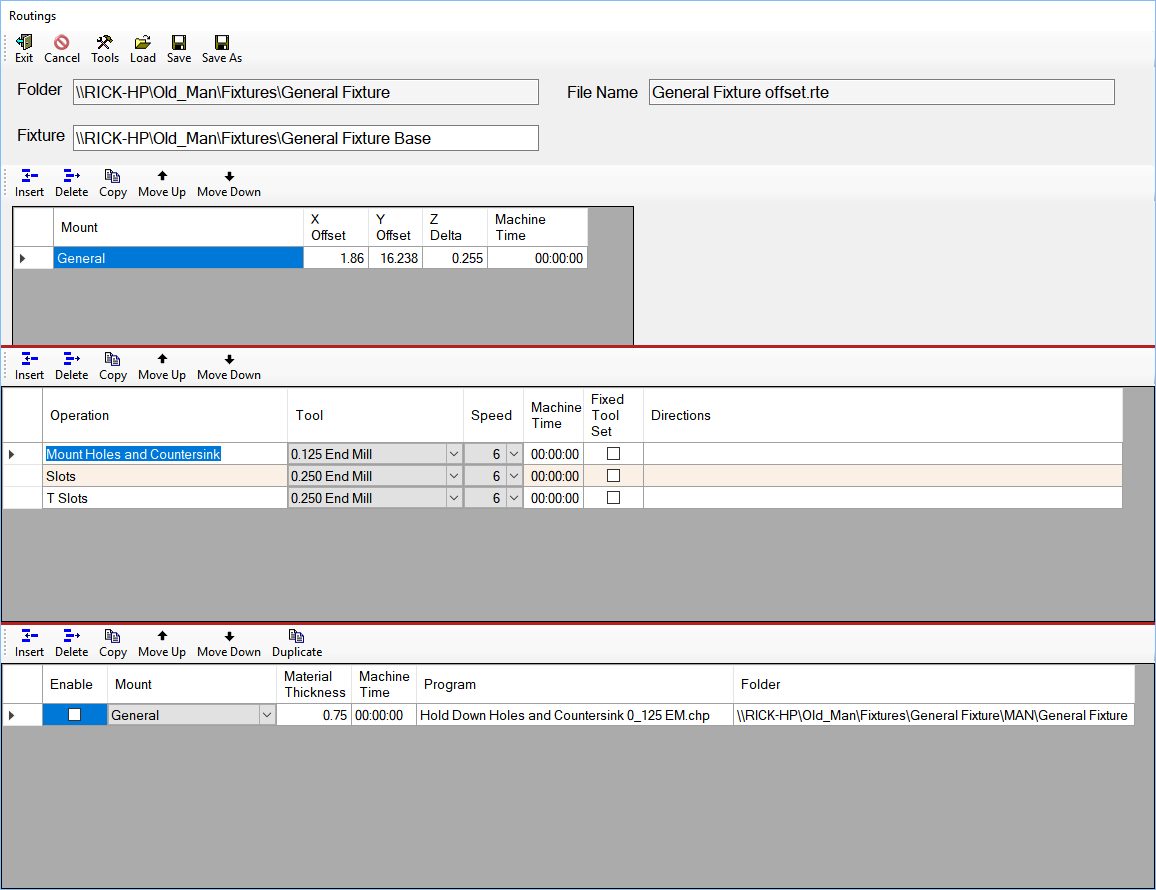
The following is a product tree that shows a product called Jewelry Box expanded. All the CAD drawings, CAM sessions and Documentation for both the whole assembly and each part are shown (see Lid below). Clicking on a node like CAD brings up a Open File Dialog, and clicking on the file will start the CAD program with that file. Same with the CAM file. I use Vetric VCarve, so if I click on a file VCarve is started with that file. Documentation files are usually either Note Pad, Word or PDF files.

If you look at the Routings node, the ‘Machine Lid Top’ routing is expanded, with the operation ‘Flatten’ expanded. The tool used with the manual speed setting (if using a variable speed router) and other directions will be displayed to the operator when this operation is run. The fixture to be used and mount information is also show. The ‘Lower Left’ mount has the ‘Top Flatten 1.200 End Mill’ part program. There could have been multiple programs listed here but this operation only had one.



When editing a rout the dialog shown below is used. This is for machining a fixture, not for any of the above. I will not try to explain this other than to say that this is where fixture offsets are entered for fixture mounts, the operations and what tool is to be used, and for each tool what G Code programs are to be used.

When an operation is selected a list of G Code programs will be loaded into the ‘Operation’ tab of the G Code panel. When run in Semi-Auto or Auto mode this is list of G Code programs that will be run, each with its own fixture offsets and tooling information for the operator. This only shows one mount being used (bottom panel) but if multiple mounts were used they could be enabled or disabled to take into account that all parts may not be mounted



The Chip application also has an interface for a Logitech G13 game pad that I use as a pendant. It has a joy stick for X and Y jogging, buttons next to the joy stick for Z jogging, 22 buttons and a four line display. The display color can be changed to show what functions are currently active.

I have one display for jogging and running G Code programs (red), one for probing and edge finding (green) and one for some other functions (blue).





I also have a camera interface. I was using a bore scope that I made an adapter for to chuck into my spindle. Looked cool, but was of little use for setting up a machine.

One idea for a camera that may make sense, is to use it to see the cutter up close while cutting. One of my machines has a vacuum system and when the cutter is covered the brush so you cannot see what’s. When I get time I am going to mount a bore scope camera (a lighted one) in the brush holder so I can see what’s going on. That will be added to the app.