"Users Guide for Brains"

Brains the other white meat……………

This guide is meant to be a Practical applications guide for using Brains in Mach3 as of 3.041.006. **<u>It assumes you have watched the Video tutorials on the Mach Support site</u>**. The information contained in this guide is from my own dealings and hair pulling when developing them. Your mileage may vary. I don't go into Deep Brains or Logic, mostly how to use the Brains IDE, and setting up the Modbus for them, with some examples.

Legal Disclaimer: (If you hurt yourself, or others, or cause some kind of damage using this guide, then it is your fault, not mine, be a big boy, suck it up, and drive on).

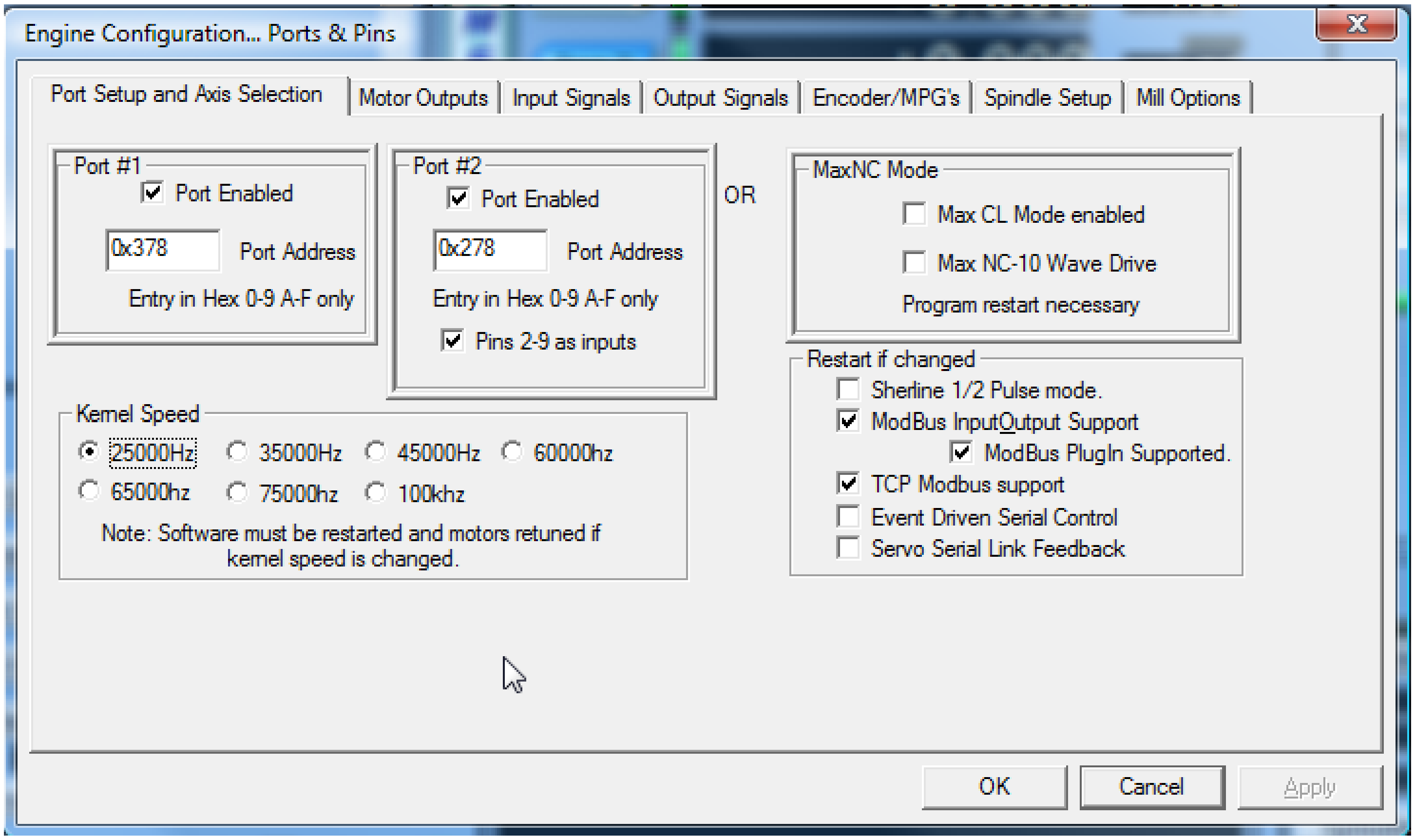
Ok on to the show, Brains was really meant to be a Software Ladder Logic type program that would allow users to extend their I/O capabilities, through external hardware device, over a communications medium (serial or TCP/IP), using a protocol called RTU Modbus.
I wont go into an explanation of what Modbus is or how it works since it really will not apply to this tutorial. Look it up on the net if you got to know.

I will give 2 real world example of hooking up to an Ethernet Driven TCP/IP PLC (Programmable Logic Controller), from Automation Direct (DL-06). The second example will be a M1 serial modbus board plugged into a M13 mother board from CNC4PC.
NOTE: I would also give an example of Peter Holman's "Mod IO" but I don't own one to test with. Further he has a well documented board, and excellent product from all that have posted about it.

Start with Serial first. In my case I am using a USB to Serial converter cable, since my laptop doesn't have a Real Serial card in it. I am using a Radio Shack converter, BUT, Peter Holman has found a much faster USB to Serial converter that doesn't have the crashes when dealing with High amounts of I/O. Get with him, if you COMs Drop a lot on serial to USB converters.

I won't go into depth on how to set up your Modbus since the Videos do a fine job of it, I will just show my setups for the examples above. On the Serial I am using COM 2, 115,000, 8-N-2, with a 50ms time out (mostly because it is a converter you need the time out). Notice also that I staggered some what my I/O refresh rates so everything doesn't go at once and thus CRASH the USB to SERIAL converter. If you COMs drop a lot, and your using USB to SERIAL converters this is most likely the culprit. Again, Get with Peter Holman, of Homan Designs (maker of Mod IO and several other products), he can tell you a fast one, (I haven't got mine as of yet………).

Below is a Screen Shot of the Modbus Serial Plug-in set up screen. I would HIGHLY recommend you use the Plug-in enabled Serial Modbus since you can have up to 64 Configurations (0-63). To set this up go to: Config>Ports and Pins. The screen that comes up with a tab that says: "Port Setup and Axis Selection" in the right bottom box Tick off (for Serial Plug-in enabled): "ModBus Input/Output Support", and under it, "ModBus Plug-in Supported". ALSO: If you are running a TCP/IP device like a PLC, tick off the next check box down called: "TCP ModBus support". Hit Apply, then OK.


Engine Configuration... Ports & Pins

Port Setup and Axis Selection | Motor Outputs | Input Signals | Output Signals | Encoder/MPG's | Spindle Setup | Mill Options

Port #1
[✓] Port Enabled
0x378 Port Address
Entry in Hex 0-9 A-F only

Port #2
[✓] Port Enabled
0x278 Port Address
Entry in Hex 0-9 A-F only
[✓] Pins 2-9 as inputs

OR

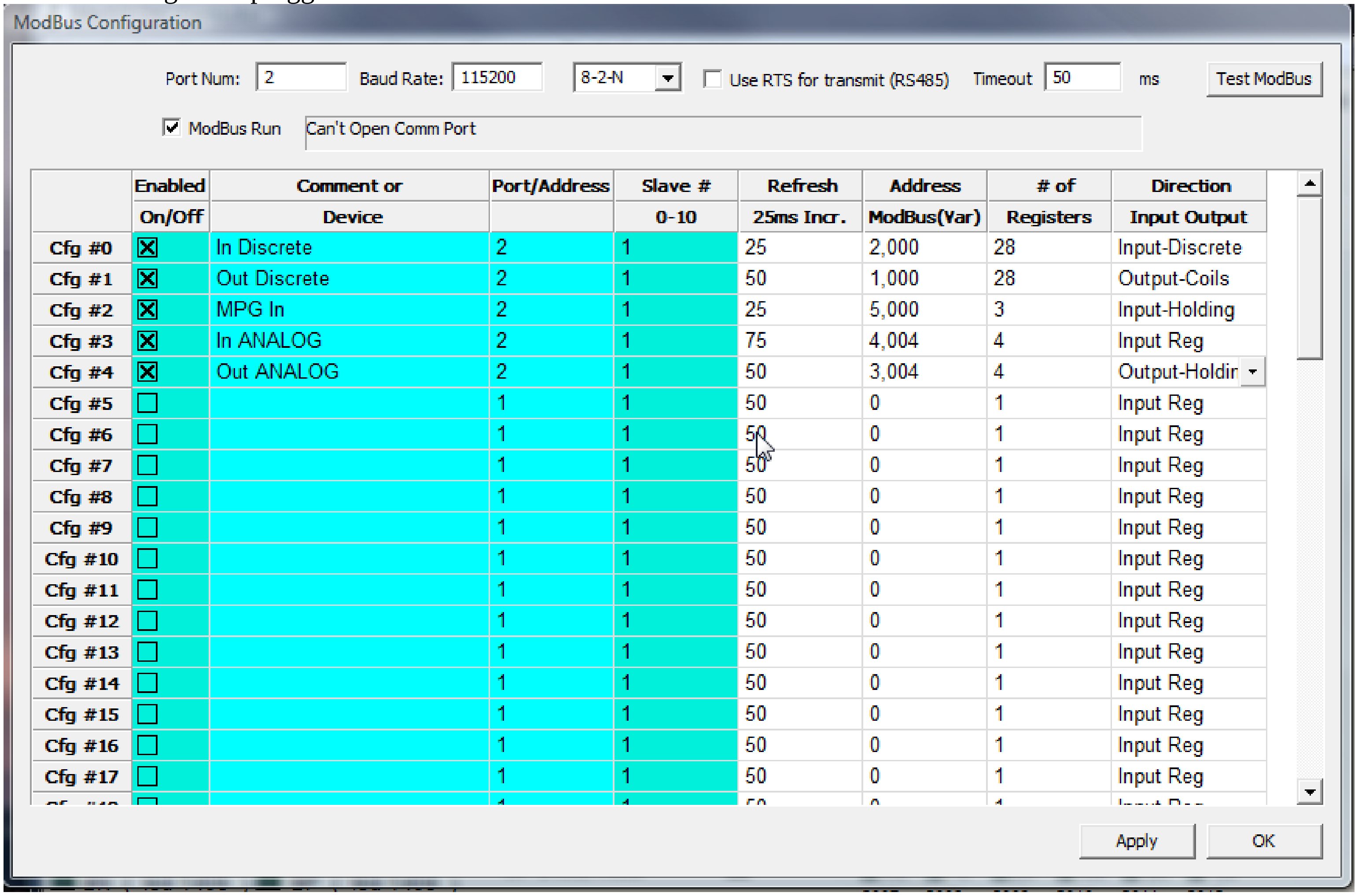
MaxNC Mode
[ ] Max CL Mode enabled
[ ] Max NC-10 Wave Drive
Program restart necessary

Kernel Speed
[● 25000Hz] ○ 35000Hz ○ 45000Hz ○ 60000hz
○ 65000hz ○ 75000hz ○ 100khz
Note: Software must be restarted and motors retuned if kernel speed is changed.

Restart if changed
[ ] Sherline 1/2 Pulse mode.
[✓] ModBus InputOutput Support
[✓] ModBus PlugIn Supported.
[✓] TCP Modbus support
[ ] Event Driven Serial Control
[ ] Servo Serial Link Feedback

OK | Cancel | Apply

Then Go to: Function Cfg's>Setup Serial ModBus Control a box will come up that looks like the one below. In this case I am using a M1 plugged into a M13 serial modbus board from CNC4PC.



Port Num: 2   Baud Rate: 115200   8-2-N   ☐ Use RTS for transmit (RS485)   Timeout 50 ms

☑ ModBus Run   Can't Open Comm Port

	Enabled On/Off	Comment or Device	Port/Address	Slave # 0-10	Refresh 25ms Incr.	Address ModBus(Var)	# of Registers	Direction Input Output
Cfg #0	☒	In Discrete	2	1	25	2,000	28	Input-Discrete
Cfg #1	☒	Out Discrete	2	1	50	1,000	28	Output-Coils
Cfg #2	☒	MPG In	2	1	25	5,000	3	Input-Holding
Cfg #3	☒	In ANALOG	2	1	75	4,004	4	Input Reg
Cfg #4	☒	Out ANALOG	2	1	50	3,004	4	Output-Holdir
Cfg #5	☐		1	1	50	0	1	Input Reg
Cfg #6	☐		1	1	50	0	1	Input Reg
Cfg #7	☐		1	1	50	0	1	Input Reg
Cfg #8	☐		1	1	50	0	1	Input Reg
Cfg #9	☐		1	1	50	0	1	Input Reg
Cfg #10	☐		1	1	50	0	1	Input Reg
Cfg #11	☐		1	1	50	0	1	Input Reg
Cfg #12	☐		1	1	50	0	1	Input Reg
Cfg #13	☐		1	1	50	0	1	Input Reg
Cfg #14	☐		1	1	50	0	1	Input Reg
Cfg #15	☐		1	1	50	0	1	Input Reg
Cfg #16	☐		1	1	50	0	1	Input Reg
Cfg #17	☐		1	1	50	0	1	Input Reg

Starting at the top Left to right, Port Num is 2, Baud Rate is 115200, Protocol is 8-2-N, Timeout is 50ms.
I used a different Cfg# for each function type. All my discrete inputs are "Cfg #0", All my discrete outputs are "Cfg #1", all my MPG inputs are "Cfg #2", my analog in is Cfg #3, and my analog out is Cfg #4.

Note in Serial Modbus with Mach3 you have a maximum total of 128, Sixteen Bit registers to bring in, and 128, Sixteen Bit Registers to send out. (You can go to 1024 in and 1024 out with a custom plug-in for serial but that is beyond the scope of this doc). What that means is when you add up all the inputs across ALL of your configs the total cannot be more than 128 in, and 128 going out. I.e. you can have 100 inputs in Cfg0, and 28 inputs in Cfg1, but you could not have any more Input configs since you would have used up all the Inputs.

The columb called Enabled on/off is to enable the use of that Cfg#, Comment or Device is for your use to identify what the config does. Port Address says what COM port to use (2 in this case), Slave # 0-10 this is if your device has a Slave # and all Modbus Serial devices should, In this case Slave #1 is the name of the M1 that I am talking to. (Note the example here is for RS232, BUT!!!!! You can talk to MORE than one serial device on a RS485, each device will need its own Slave #, unique on the "network", and all the COMs on the slaves have to be set the same).

Refresh Rate this is set in 25ms, 25 ms is as low as you can go, so your increments will be in multiple of 25ms, note in this example I have staggered my refresh rates to keep my Radio Shack USB to Serial from Crashing.

Address ModBus (Var) this is the start address of the register(s) you wish to use, in this case i.e. 2,000 represents the first discrete input coming from the M13 board (pin 0).

# of Registers is 28 in this case, since there are 28 inputs.

Direction Input Output, this is the “Type” Of Modbus I/O there are several types to choose from both on Inputs and Outputs the little window has a drop down thorn when you click in it to reveal the types to choose from. The type you choose will depend on your Devices users manual. In the above example, that is the types for the M1/M13 board.

Once you have configured your Serial Plug-in Modbus you then need to test it, hit the check box that says “ModBus Run”, Then go to the bottom right and hit Apply, and then to the upper right and click on the button “Test ModBus”. From that Window test your modbus I/O, make sure your COM port matches the one your using, and make sure that you use the correct Address, and Modbus IO type.

Once you confirm you have I/O, get out, hit Apply and OK.

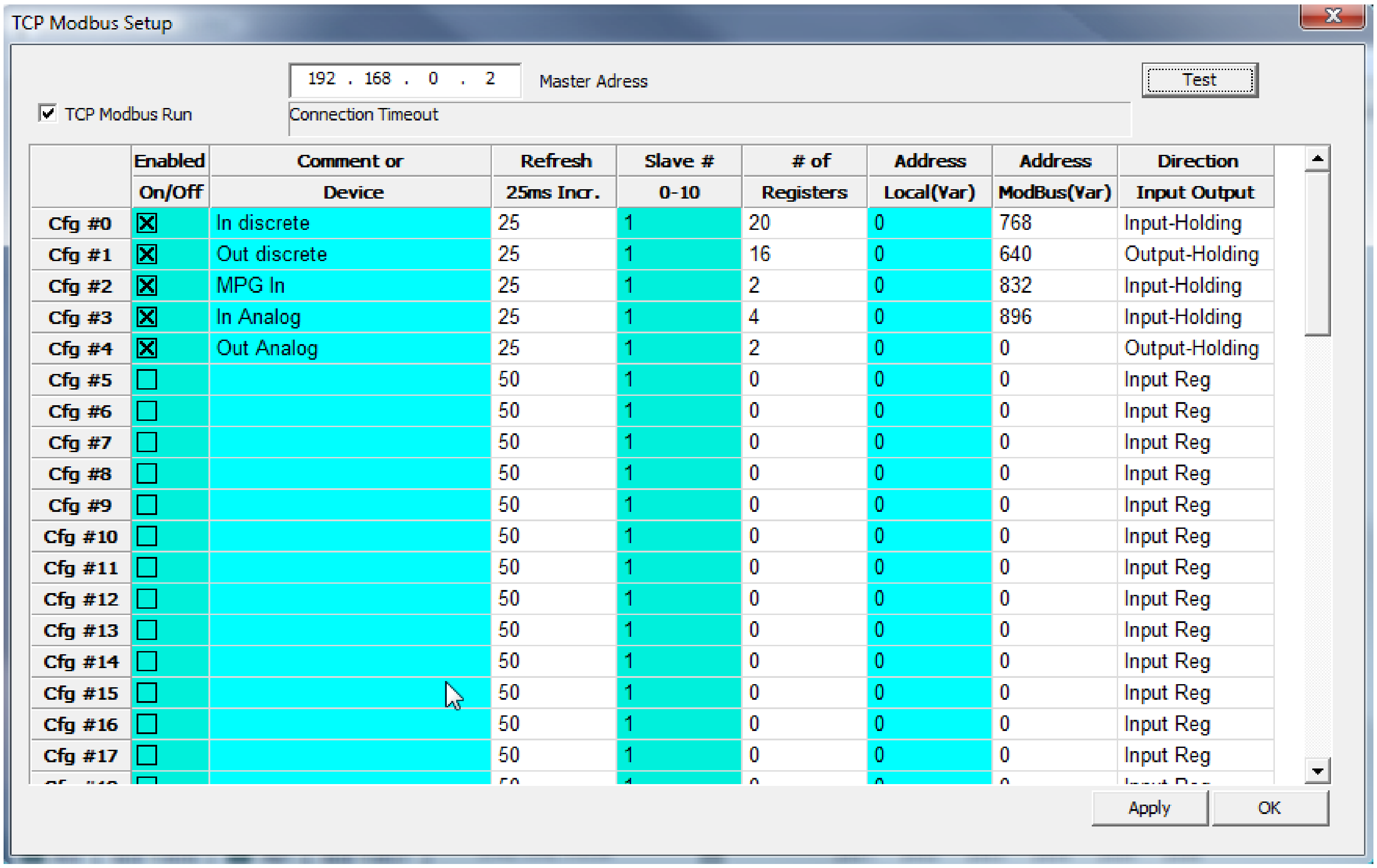
Now, I recommend that you make your self an Input and an Output “flow Sheet” This is where you can put what device and device #, what device address range and type, what it converts to in Mach Modbus config window, and its Cfg #. If your doing a larger machine it gets confusing quick, since there (currently), is no way to put “documentation” in the Brian rungs. It may also behoove you to make a Flow sheet for any Brains you make, since as they get larger the icons and there information truncates. What I usually do, is use a Screen capture utility, to take snap shots of my rungs, scroll down and take the next capture until I reach the end. I then tape them together and use a pen to document what each rung is doing where it is going, or coming from etc. A few weeks down the line when you go back to it, you will thank your self for the trouble, because you will have forgotten, unless your blessed with a Great memory.

Now a Quick Once over for TCP/IP Modbus config:

I will use the example with a DL 06 PLC from Automation Direct with an “E-Com100” Ethernet Module in it. Once you go to Ethernet you will never willing want to go back to super slow serial………..

First you will need to set up your Ethernet Module in your DL05 or 06, (the procedure is very similar for DL 205, 405 and Terminator I/O (stay far, far, far away from the 305 series… you’ve been warned). You will need a CROSS OVER CABLE!!!! If the computer that has Mach on it is plugged directly into the PLC E-COM module!!!!
You can ONLY use Patch if you go through a hub or something first. First open you DirectSoft5 PLC programming soft ware (you can download a full working copy for free limited to 1000 bits but usually that is more than enough for small ladder projects, if you want to buy the unlimited version it is around 500 approximately. Once you open up the Direct Soft, you will see at the top a list of things on the left, go under “Utilities” about ¾ they way down you will see something called “NetEdit3” (now we are assuming that you have plugged in your Crossover cable from your RJ45 on your comp to the ECOM module). Follow the “Ecom100” Manual to set this up, the summery version is this, for this manual, I set the module to: IP address 192.168.0.2, Device 1, Subnet Mask 255.255.255.0, Gateway 192.168.0.1.

Once that is done, you will then need to close out of Net Edit3 (HIGHLY recommend you read the manual on it). Now on your computer, go to: Start>My Computer>Control Panel>(then depending on your flavor of windows, find the Networking stuff, and open up the Ethernet Card. When It opens go to standard IP 4 protocol, and hit properties. When it comes up, click on the radio button to do Manual IP configuration, NOT DHCP auto, Put in the Address for this computer (the one with Mach on it that you are talking to the PLC with), 192.168.0.1 address, 255.255.255.0 for subnet mask, 192.168.0.1 for gate way (yes, you are your own gateway). Under advanced>WINS>click the radio button for “Enable NetBIOS over TCP”. Then hit apply and close. On your PLC make two simple rungs one for input, and one for output. Translate those octal address to decimal for the starting addresses in mach. See TCP screen shot below, Test your IO. With the TCP Modbus Test Set up, it works just like the Serial, but has the TCP stuff in it.



192 . 168 . 0 . 2 Master Adress

☑ TCP Modbus Run

Connection Timeout

	Enabled	Comment or	Refresh	Slave #	# of	Address	Address	Direction
	On/Off	Device	25ms Incr.	0-10	Registers	Local(Var)	ModBus(Var)	Input Output
Cfg #0	☒	In discrete	25	1	20	0	768	Input-Holding
Cfg #1	☒	Out discrete	25	1	16	0	640	Output-Holding
Cfg #2	☒	MPG In	25	1	2	0	832	Input-Holding
Cfg #3	☒	In Analog	25	1	4	0	896	Input-Holding
Cfg #4	☒	Out Analog	25	1	2	0	0	Output-Holding
Cfg #5	☐		50	1	0	0	0	Input Reg
Cfg #6	☐		50	1	0	0	0	Input Reg
Cfg #7	☐		50	1	0	0	0	Input Reg
Cfg #8	☐		50	1	0	0	0	Input Reg
Cfg #9	☐		50	1	0	0	0	Input Reg
Cfg #10	☐		50	1	0	0	0	Input Reg
Cfg #11	☐		50	1	0	0	0	Input Reg
Cfg #12	☐		50	1	0	0	0	Input Reg
Cfg #13	☐		50	1	0	0	0	Input Reg
Cfg #14	☐		50	1	0	0	0	Input Reg
Cfg #15	☐		50	1	0	0	0	Input Reg
Cfg #16	☐		50	1	0	0	0	Input Reg
Cfg #17	☐		50	1	0	0	0	Input Reg

The Above is my TCP setup (ignore the MPG in it was for testing a plug in I did for picking up an MPG from a PLC). The Top that says Master Address is where your Address of the Ecom is: 192.168.0.2, This set up is just like the serial one above Except it has ONE thing Extra: Address Local(Var), this is so if you want to you can break up local vars starting addresses across a larger block of addresses if you need to specifically jump around in address space in the PLC. (translation, Most PLC manufactures have sometimes overlapping or legacy addressing that is right in the middle of a block you need for something else, you may address that weird block with another Cfg, and then continue with the original configs at the higher address and higher Local Var. If that didn't make sense don't worry about it, it really isn't that important overall).

In TCP/IP Modbus, you have a total of 1024 Sixteen Bit inputs, and 1024 Sixteen Bit Outputs, again like serial when you combine all the inputs of all your configs the number cannot be greater than this, same for outputs.
Now on my config above, you will note that I write to Holding Registers both INPUT and OUTPUT. This is because it will allow me to control my Memory space in my PLC.

Currently "Bit Of Word" in the "Brains" is Broken, (I am hoping that Brian will get to that and fix it along with the other bugs/quirks that I will list in this doc (I will note the work around as they apply).
If you want to use "Bit Of word" in your PLC, i.e. B1200.0, B1201.0, B1202.0 etc. then you will have to send one FULL word in or out from Brains to do that, until Brain or Art fixs the bit of word in Brains. This rally only becomes important when looking at traffic volume and RX/WX instructions if your doing a PLC Drop network (2 or more PLCs seen as One HUGE PLC by Mach). Also since you can't do Bit of Word addressing in Brains it will also EAT up your "Control Relay" registers. Thus the reason I use Variable Memory (HUGE blocks), and use bit of word on them for discrete I/O, and just send out from Brains a Full 16 bit word that just has a 1 or 0 in it.

Now we have our Serial and TCP devices working with the Modbus Config “Test” utility.

We are now ready to do a Brain!!!!!

Brains are really meant to give control over your Modbus I/O to effect things in or out of Mach.
You can use Brains for Parallel port stuff if you want as well.

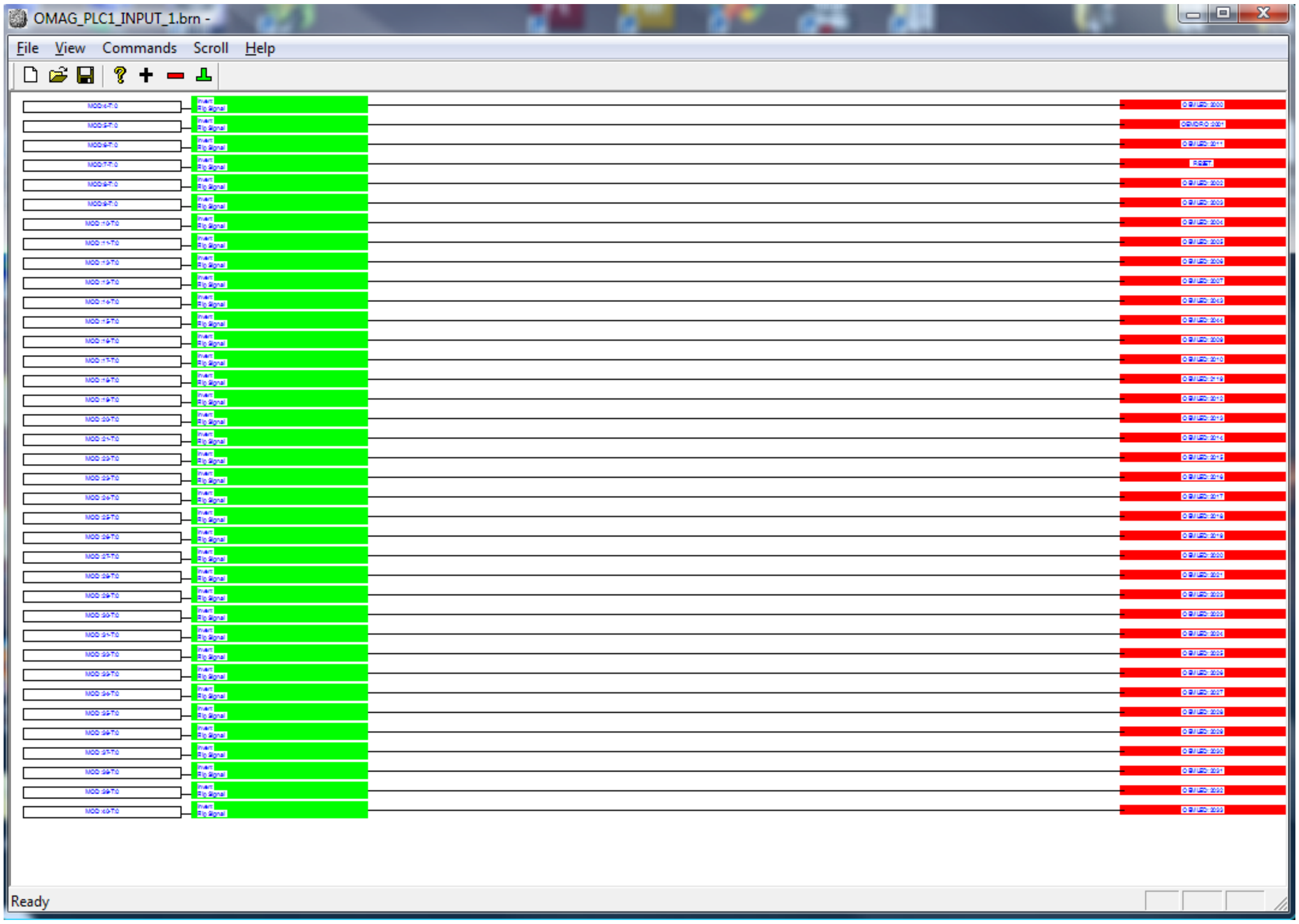
Brains is a “Visual Ladder Logic” Programming environment, since most of the choices have choice windows, and dropdowns.

Bugs

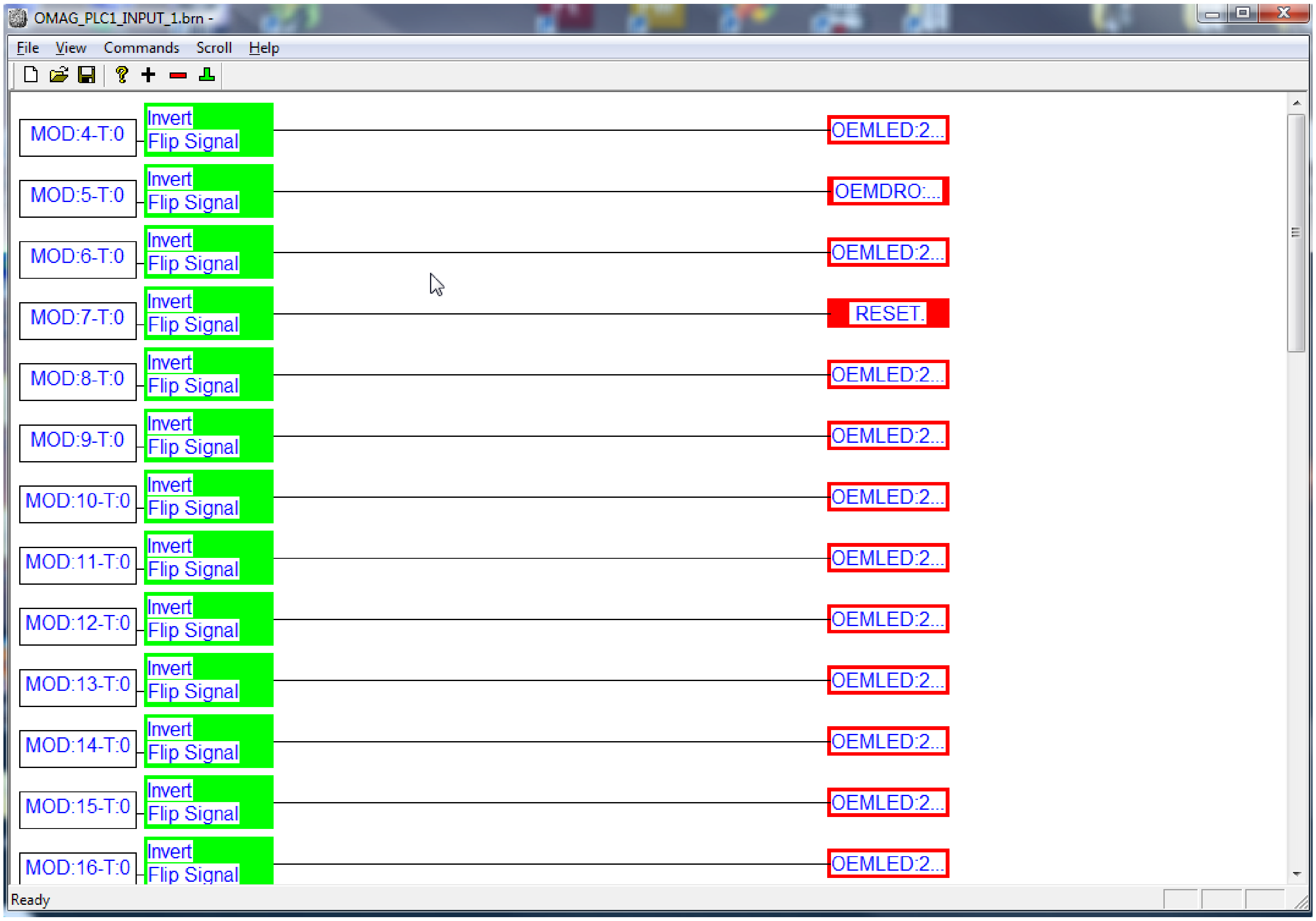
First let me start off with the Bugs and Quirks, that I have found with the work arounds.

1. You can only put no more than 50 rungs on a single Brain after that, you have to go to another brain, I usally stop at 40 or so.
2. Remember if your breaking up a large Brain across several smaller Brains on the Same config, remember to start the new brain with the next Local var.
3. When you bring in a Modbus input that goes to an UserLED in Mach, you will need to do a “Invert Op lobe”, for it to turn on the mach screen, so the Brain will show the ULED as off when the modbus input is on, but the actual ULED on the Mach screen will be on.
4. When you get to roughly rung 14 the Input may hang “on” when you try to go to your next lobe, to turn it off, you have to click the input, then the op, maybe 2 or more times, sometimes you have to hit the scroll on/off, on/off to break it out, no telling just fool with it.
5. You will note that when making a larger Brain, at some point toward the bottom the rungs will suddenly shrink, and you can’t see what they say. Click on the programming window, and then hit the Scroll button. Sometimes you will have to hit the scroll button 2 or more times (Toggling it), until it will let you scroll down to the bottom. This will also make the Rungs larger but truncate the Node boxes.
6. Now, once you get a large enough Brain that you have to use the Scroll bar, you will note that as the Size expands to where you can see the rungs, the individual Nodes will shrink Horizontally in where you can only see about 1/3 or the node, for instance if you have OEMLED2000, then when you size it up with the scroll, it will turn into OEMLED2.. (see following screen shots of unexpanded and expanded brain scrolls).

7. Example Unexpanded large brain:



Example Expanded Large Brain:



MOD:4-T:0 Invert Flip Signal OEMLED:2...
MOD:5-T:0 Invert Flip Signal OEMDRO:...
MOD:6-T:0 Invert Flip Signal OEMLED:2...
MOD:7-T:0 Invert Flip Signal RESET.
MOD:8-T:0 Invert Flip Signal OEMLED:2...
MOD:9-T:0 Invert Flip Signal OEMLED:2...
MOD:10-T:0 Invert Flip Signal OEMLED:2...
MOD:11-T:0 Invert Flip Signal OEMLED:2...
MOD:12-T:0 Invert Flip Signal OEMLED:2...
MOD:13-T:0 Invert Flip Signal OEMLED:2...
MOD:14-T:0 Invert Flip Signal OEMLED:2...
MOD:15-T:0 Invert Flip Signal OEMLED:2...
MOD:16-T:0 Invert Flip Signal OEMLED:2...

8. State changes to a brain, will not effect rungs that depend on that state below it, until next scan. So if the first run has something that makes user Dro2000 a value of 100, and then in the rung below that you start off with Dro2000 (which is a 0, until the next scan then it becomes a 100).
9. Whimsical Bug: This one floats around and depends on the size of the brain, but for the most part, when you write to an output (and that output being a userDRO, or UserVAR, and you use it as an input on the next rung or somewhere further down, you can only reuse it Once. I have found that "Usually" with larger brains, the 2cd Plus instance of the same Var or DRO used now on the input side, gets ignored by the brain. The only way I have found around this it to move the logic to another Brain all together, reestablishing the Vars if necessary but writing to the next condition.
10. If you make a mistake on your Terminations, on your ladder, you have to erase all the rungs (op lobes you can leave the inputs if you want), all the way back up to the mistake and redo it all back down again. A Non-Hair pulling work around for this is to open up mach, then Hit View Brain and pick the one that has the mistake in it, open it to view, then slide it out of the way, and hit the ok button to close the view box. Then hit Edit Brian, and Open the Brain that your currently holding in your view box (don't worry it is a separate instance so it wont conflict). With the view box open of the Bad Brian, use the copy in the Edit brain to erase to the error, then look at your old brain in the view box to recreate the remainder of the Edit Brian, so You don't have to re-remember all the logic you did and why on the new one.
11. you cant send Floating point values through or into a brain, you have to strip out the whole from the fractional using Modulo or something similar and other math, either at the PLC side, or Mach side or both. And send it as two separate register values that you will need to recombine at the receiving end back to floating point.
12. You can only have a SINGLE output (Terminator) to a rung of ladder, you can't split your output, the only way is another separate rung of ladder that takes the same input and writes to a different output.

Other Things to consider:

- Boolean Expressions that evaluate to True or False you CAN use in a follow on formula, since you input to the formula will be a 1 or 0.
- When doing your I or O, be very careful of what Cfg your putting, and remember to specify In or Out, also the local address, and Brain type, i.e. Serial Plug-in, TCP, etc. This WILL trip you up, if you are not paying attention!!! Again I recommend getting a screen shot or write down what your Mobbus Configs are for I and O and other stuff.
- Button Actions are SINGLE shot per State change!!!, i.e. if you have an input from outside that when triggered, will do button "E-Stop", and that input stays on from outside unless you turn it off, then in Mach even though the Brain will show the button as pressed a "1", you can click on the E Stop button to take mach out of E stop!!!! EVEN though the external Estop button is still pushed in. The only way to get the estop to work from the outside is one of two ways, release and repress the Estop externally and that will re-Estop Mach. OR, what I do is have that same input go to another UserLED and then I watch the User LED as well in the Macro Pump with some other logic that if the Brains says, and external E stop is pushed, I then Look at the LED, if it is still on, I re-Hit the Estop button in the Macro pump.
- Remember Brains is a "State" program, it holds its last state until something changes it from the Input side of the Brain.
- You cant have the same object as an output, follow as an input further down in the same brain and expect it to change condition on a single scan. Brains only does ONE state change per object per scan. (at least that is what I have found). It is hit or miss if your doing it across several Brains then it will depend on execution position and last state wins…….. The only exception to this is Local Vars which appear for all intents and purposes to update locally at that scan, again that is also hit or miss, I have seen it work like that and NOT work like that………go figure……. Keep to the reusing the Output only once as an input in the same brain and you should do OK for the most part.

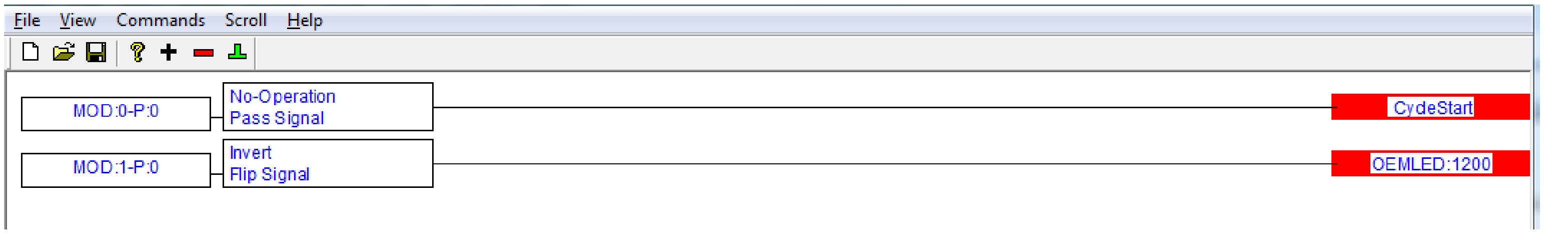
- NOTE!!!! When ever you "Edit" a brain, or Make a Brain, when you open up Brain Control, you will HAVE to hit Reload All Brains, otherwise you will still be working with your old Brain, (or cant see your new Brain), and thus pulling your hair out.

Onto Brain Surgery!!!!:

To make a Brain, go to Operator>Brain Editor It will pull up a little box that says Enter name, you can name it here or name it later with a save as. But, either button you push, OK or Cancel will bring up the Brain IDE, hit the Red X if you want to close out of the little box.

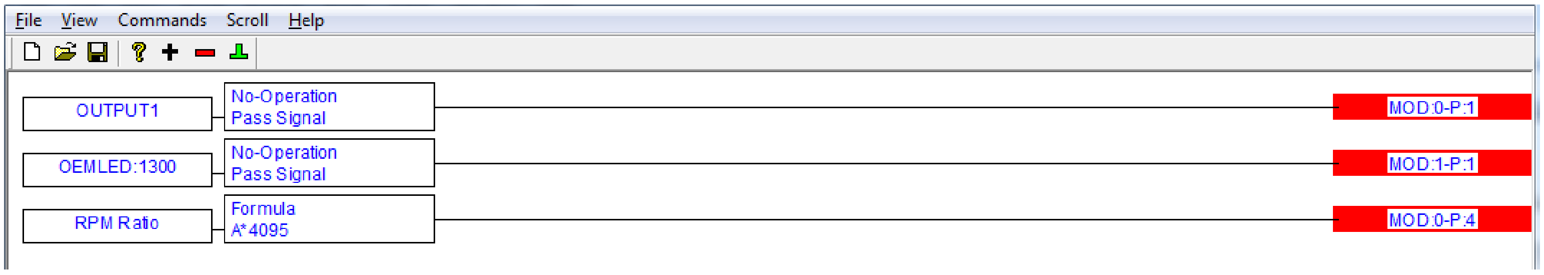
You will see three main buttons a: Black "+", a Red "-", and a Green upside down "T" The Plus sign is your input side, when you click it will bring up a dialog box, you can choose your input type, an OUTPUT can also be used as an input in Brains. The Red – removes objects (they have to be highlited to remove and you remove them from the last op-lobe or other function closest to the terminator and work you way back, The Green upside down T is the Terminator and that is what the rung ultimately effects on that line. The Inputs and Outputs are things already existing in Mach (recommend you looking at the VB docs for OEM number questions and perhaps slightly more info on the object if your not familure). The DROs and LEDs are both the Standard OEM DROs and LEDs that exist in mach, PLUS, you have the option of Ticking a box to use "User LEDs or DROs" range from 1000-2255 in each. ModBus: this is where the money shot is, here you enter the Modbus address to use this is the Local Address starting at 0, (unless in the TCP config you told it to start at a different Local address for that Cfg). Local Address "0" corresponds to your first "starting address" of your external device for that Cfg.

Here are TWO Serial Plug-in enabled Brains using the Hardware above, here is the first one:



This one shows Inputs from Modbus Local Address "0" (corresponds to hardware address 2000), goes through a No-OP lobe (means it just passes the info straight through, to the Button "Cycle Start" thus you can start your G-code cycle with an external button push. Note the input nomenclature, is "MOD:0-P:0" (means Modbus input since it is on the left side, Local Var Address "0": P = Plug in Enabled: 0 = Cfg "0".
Next rung: MODbus, local var address "1": plug-in enabled, Cfg "0", since this is going to an UserLED on a Mach Screen, I have to Invert the Signal (see bugs above), so that the Screen LED will Light when the input goes on, even though in the Brain, the OEMLED will show "0" when your Mach Screen LED is on ("1").

Next is an Output Example, it has two output on the discrete output Cfg, but Note the last Rung it is on a different Cfg, it is on the Analog output Cfg since that rung will control an Analog driven (12 bit), spindle.



Rung one watches standard Mach OUTPUT1, when it goes ON, the signal is passed to Modbus Local Address “0”, serial Plug-in enabled, Cfg #1.
Rung two watches UserLED1300, when it goes to ON, the signal is passed and it goes out like above but on Local Address “1”.
Rung Three: This one looks at the RPM Ratio DRO (this is an OEM DRO in mach that looks at the Pulley speed range you are on, and calculates across that range a % of full speed. Thus Range of the DRO is from 0.0-1.0 You will note it then passes through a “Formula box” in where the Formula A*4095 works on the results of the Ratio DRO, since this is writing to a 12 bit analog Module, 12 bits in Decimal is 4095, so the formula gives a range of 0-4095 and that goes out on Cfg #4 since that is the Cfg we used for Analog outputs. (technically, the M1 board only does 255 max, so I just used the analog module value for a PLC since it gives finer control as an example).

The Formulas and Boolean Operations really give you powerful data manipulation in a Brain.

Once you finish your Brain, do a “Save As” and name it (make sure it is saving into the “Brains” folder under Mach3. Then go to Operator>Brain control, hit the Reload All Brains, (you should see your Brain appear. Click on your Brain (it will highlight, then click the “Enable Brain” check box. As soon as you do that, your Brain is Running!!!! Next hit View Brain (with your Brain Highlighted), and a Run time Ladder of your brain will come up, as the inputs and outputs you are watching change you will see them change in the Brain View as well. (note Drag your view brain off to the side, and then hit the OK button on the Brain Config dialog to close it.

Resources:

At the Mach Support site under Forums, in the Topic “Brains” you will find many, many Threads with all kinds of Example Brains doing all kinds of things from simple to very complex.

You can Run Brains, Macro pump, and Plug ins all at the same time, but if they all are doing something to the same piece of data you will need to make some way to keep them all in synch and NOT step on each other!!!!!

The Brains are Excellent for Fast Control, Logic, formulas and functions. But sometimes when you have to “Think” it is best to couple them with the Macro pump if you need more complex “decision making”.

Remember that both Brains and the Macro Pump are “Scanners”, and they update every loop of mach, so you can get into an On/Off war between Brains and Macro pump if your not careful, or Crash again.

Also be aware of conflicts between Multiple brains, if you get flaky stuff, turn ALL Brains off, (and Macro pump, if your running it). Then enable 1 Brain at a time, until the flaky reappears, then go through your Brains, chances are you made a typo some where and assigned two oposite inputs to the same output between two brains. Or wrote to the wrong address……..

Hope that clears up a few things and gets you started,

Scott “Poppa Bear” Shafer

S S SYSTEMS, LLC
http://ss-systemsllc.com/default.aspx