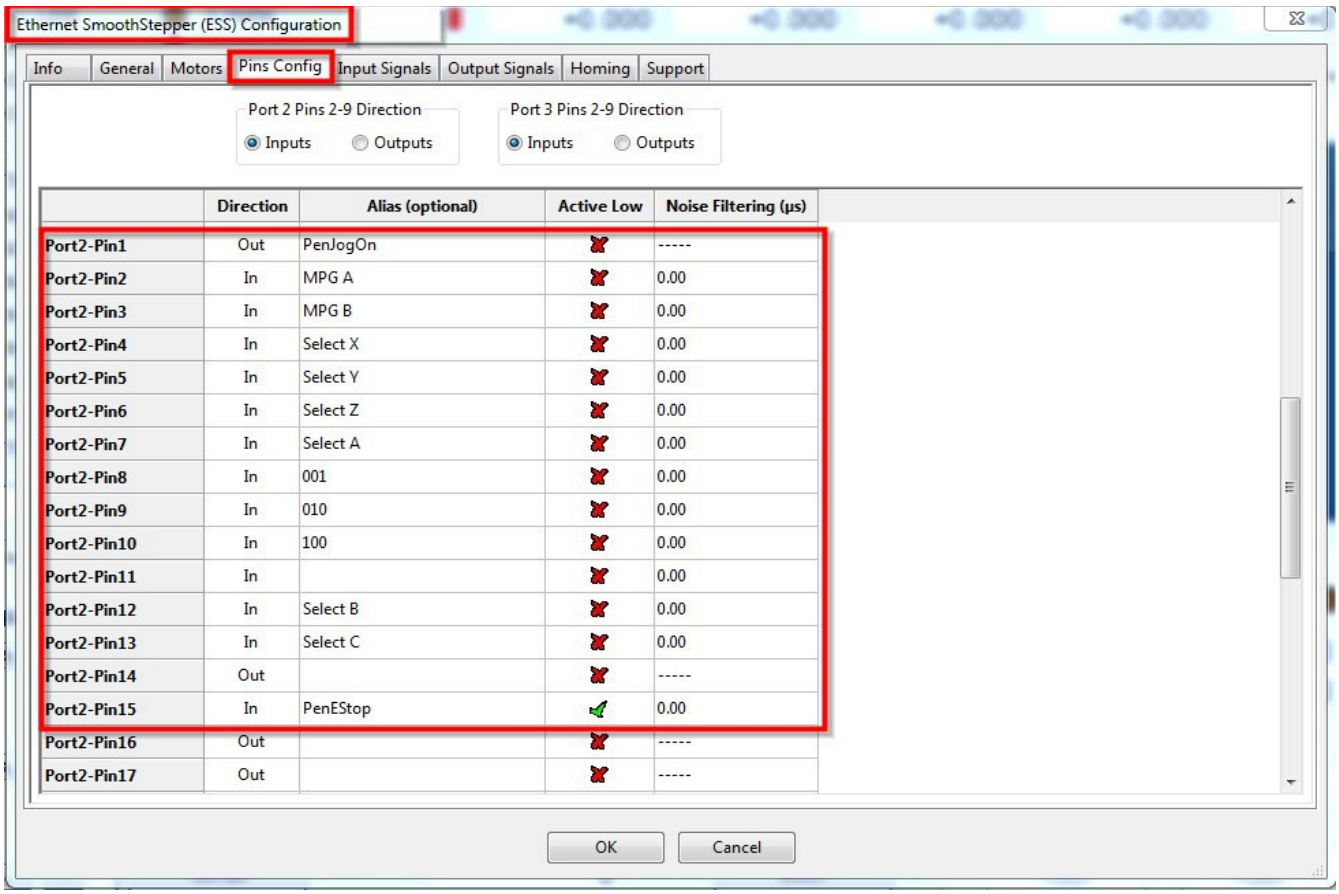
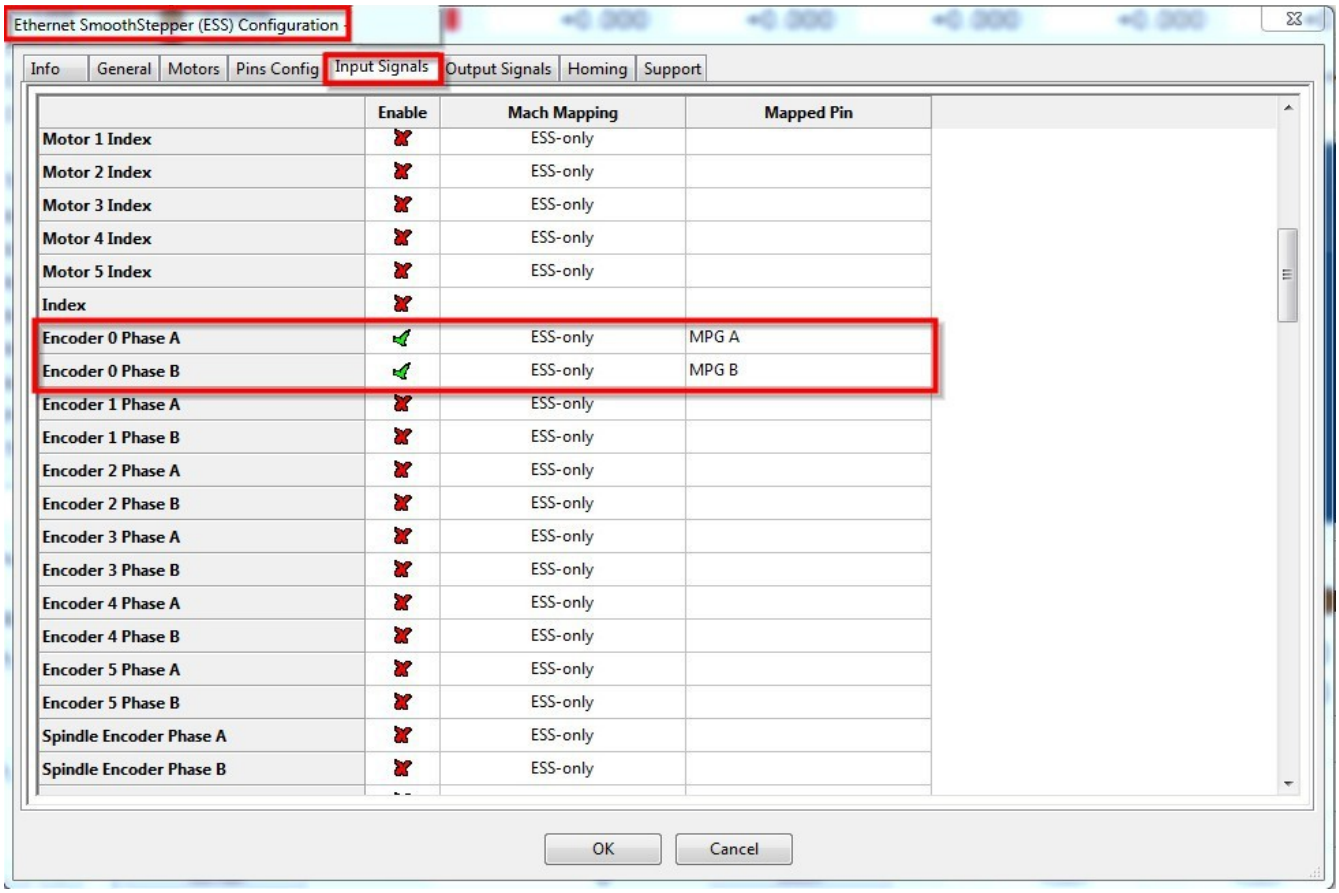
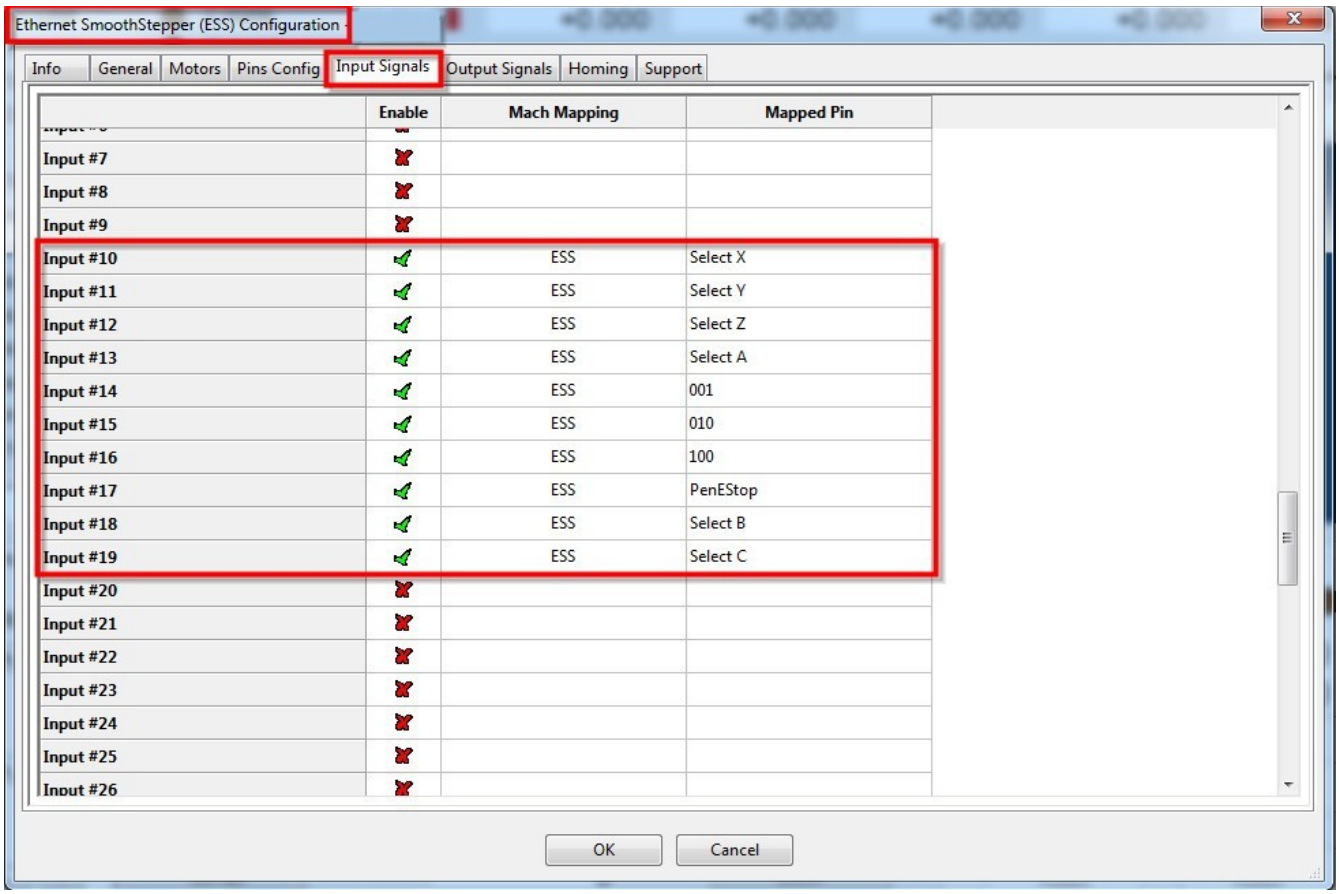
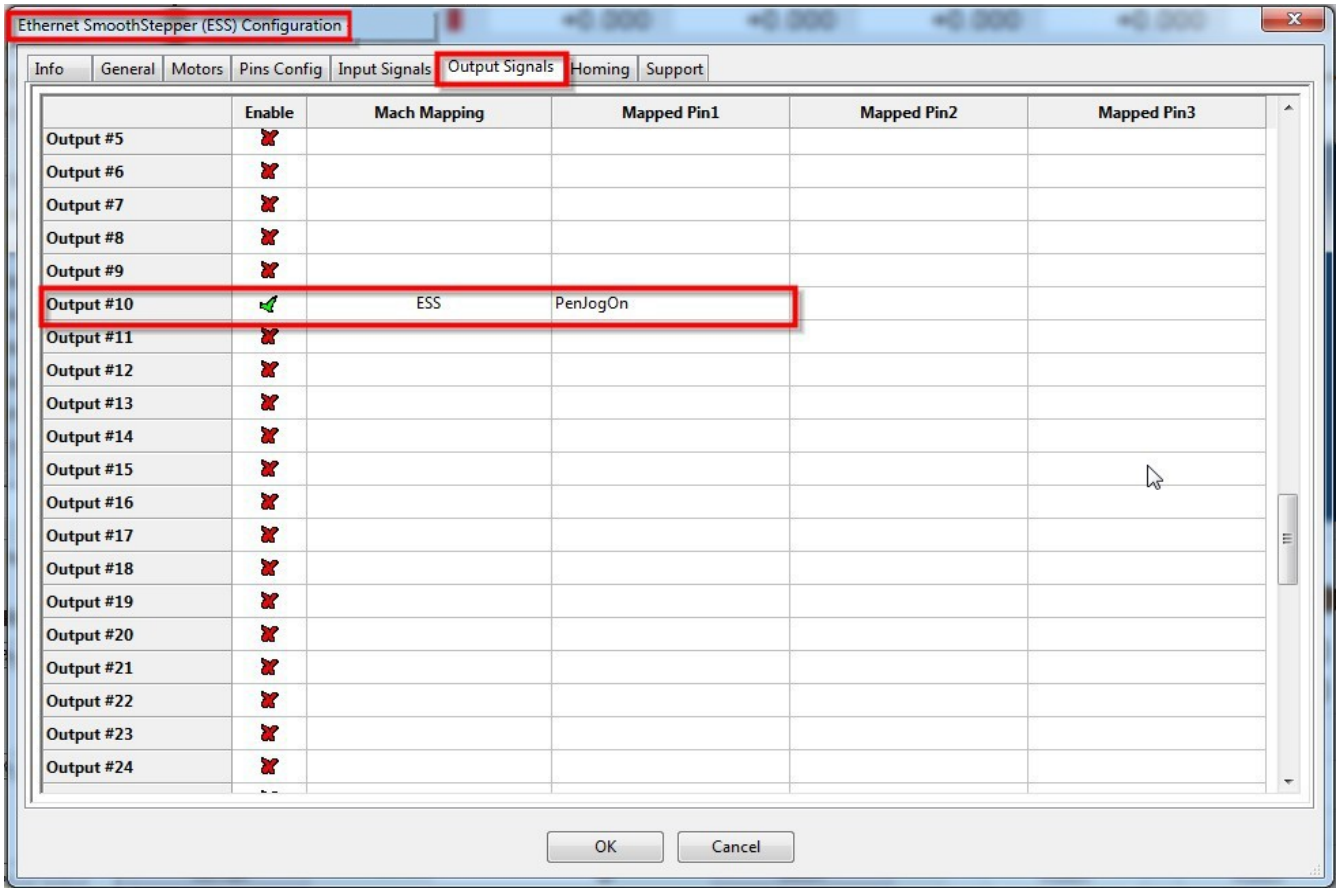
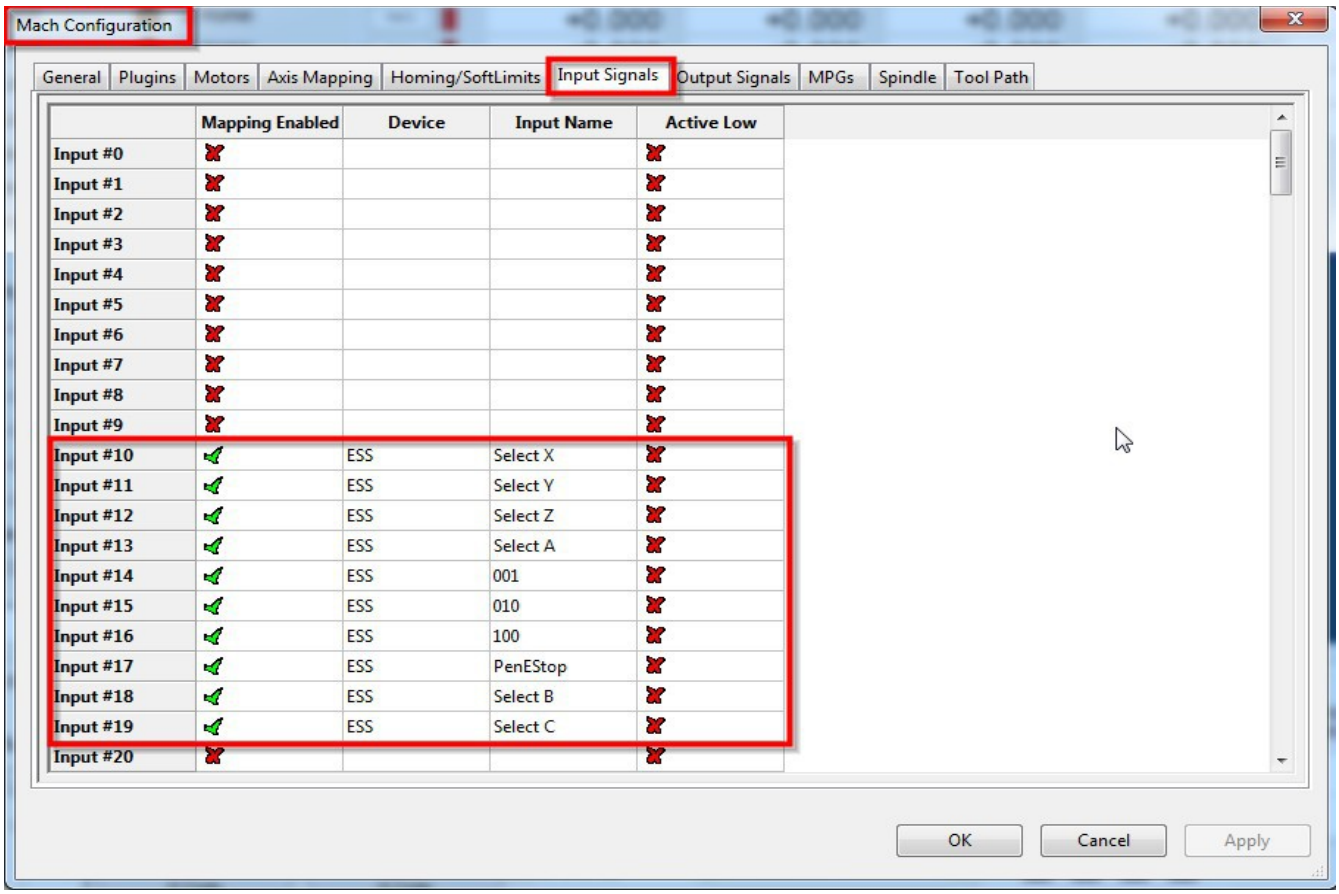
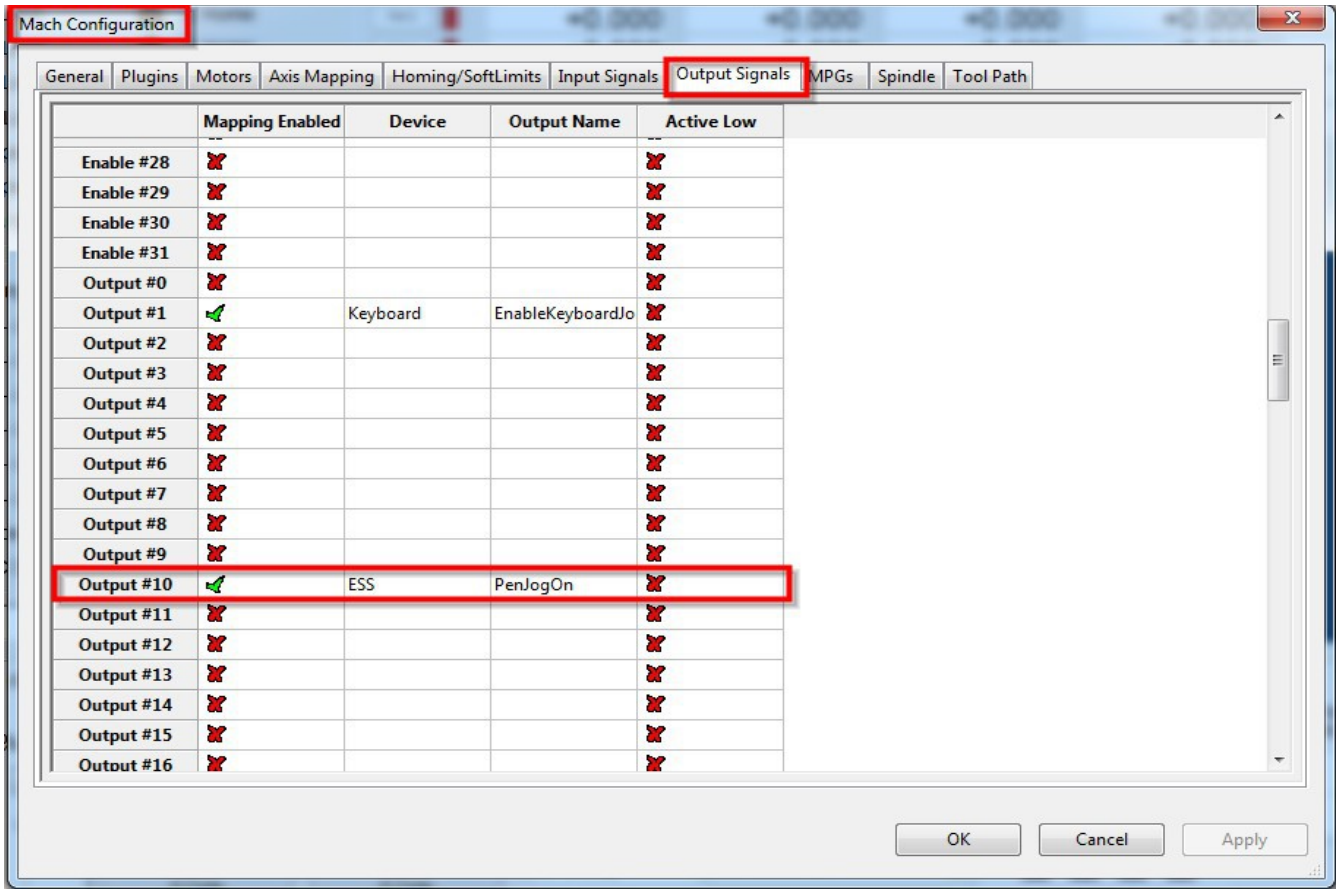

Ethernet SmoothStepper (ESS) Configuration
Info
General
Motors
Pins Config
Input Signals
Output Signals
Homing
Support
Port 2 Pins 2-9 Direction
Inputs
Outputs
Port 3 Pins 2-9 Direction
Inputs
Outputs
Direction
Alias (optional)
Active Low
Noise Filtering (µs)
Port2-Pin1 Out PenJogOn -----
Port2-Pin2 In MPG A 0.00
Port2-Pin3 In MPG B 0.00
Port2-Pin4 In Select X 0.00
Port2-Pin5 In Select Y 0.00
Port2-Pin6 In Select Z 0.00
Port2-Pin7 In Select A 0.00
Port2-Pin8 In 001 0.00
Port2-Pin9 In 010 0.00
Port2-Pin10 In 100 0.00
Port2-Pin11 In 0.00
Port2-Pin12 In Select B 0.00
Port2-Pin13 In Select C 0.00
Port2-Pin14 Out -----
Port2-Pin15 In PenEStop 0.00
Port2-Pin16 Out -----
Port2-Pin17 Out -----
OK
Cancel

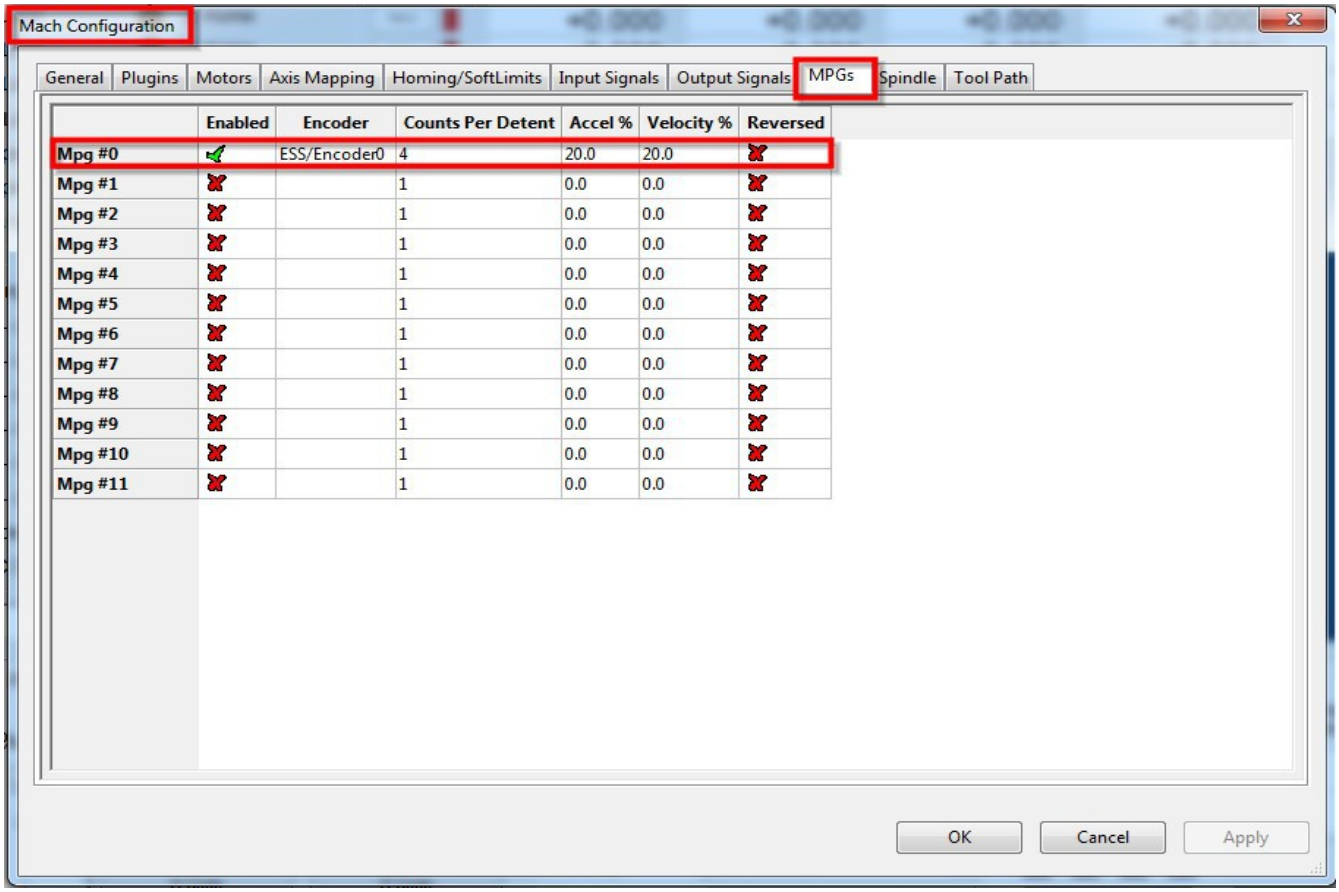

Ethernet SmoothStepper (ESS) Configuration
Info
General
Motors
Pins Config
Input Signals
Output Signals
Homing
Support
Enable
Mach Mapping
Mapped Pin
Motor 1 Index ESS-only
Motor 2 Index ESS-only
Motor 3 Index ESS-only
Motor 4 Index ESS-only
Motor 5 Index ESS-only
Index
Encoder 0 Phase A ESS-only MPG A
Encoder 0 Phase B ESS-only MPG B
Encoder 1 Phase A ESS-only
Encoder 1 Phase B ESS-only
Encoder 2 Phase A ESS-only
Encoder 2 Phase B ESS-only
Encoder 3 Phase A ESS-only
Encoder 3 Phase B ESS-only
Encoder 4 Phase A ESS-only
Encoder 4 Phase B ESS-only
Encoder 5 Phase A ESS-only
Encoder 5 Phase B ESS-only
Spindle Encoder Phase A ESS-only
Spindle Encoder Phase B ESS-only
OK
Cancel


Ethernet SmoothStepper (ESS) Configuration
Info
General
Motors
Pins Config
Input Signals
Output Signals
Homing
Support
Enable
Mach Mapping
Mapped Pin
Input #7
Input #8
Input #9
Input #10
ESS
Select X
Input #11
ESS
Select Y
Input #12
ESS
Select Z
Input #13
ESS
Select A
Input #14
ESS
001
Input #15
ESS
010
Input #16
ESS
100
Input #17
ESS
PenEStop
Input #18
ESS
Select B
Input #19
ESS
Select C
Input #20
Input #21
Input #22
Input #23
Input #24
Input #25
Input #26
OK
Cancel


Ethernet SmoothStepper (ESS) Configuration
Info
General
Motors
Pins Config
Input Signals
Output Signals
Homing
Support
Enable
Mach Mapping
Mapped Pin1
Mapped Pin2
Mapped Pin3
Output #5
Output #6
Output #7
Output #8
Output #9
Output #10
ESS
PenJogOn
Output #11
Output #12
Output #13
Output #14
Output #15
Output #16
Output #17
Output #18
Output #19
Output #20
Output #21
Output #22
Output #23
Output #24
OK
Cancel


Mach Configuration
General
Plugins
Motors
Axis Mapping
Homing/SoftLimits
Input Signals
Output Signals
MPGs
Spindle
Tool Path
Mapping Enabled
Device
Input Name
Active Low
Input #0
Input #1
Input #2
Input #3
Input #4
Input #5
Input #6
Input #7
Input #8
Input #9
Input #10 ESS Select X
Input #11 ESS Select Y
Input #12 ESS Select Z
Input #13 ESS Select A
Input #14 ESS 001
Input #15 ESS 010
Input #16 ESS 100
Input #17 ESS PenEStop
Input #18 ESS Select B
Input #19 ESS Select C
Input #20
OK
Cancel
Apply


Mach Configuration
General
Plugins
Motors
Axis Mapping
Homing/SoftLimits
Input Signals
Output Signals
MPGs
Spindle
Tool Path
Mapping Enabled
Device
Output Name
Active Low
Enable #28
Enable #29
Enable #30
Enable #31
Output #0
Output #1 Keyboard EnableKeyboardJo
Output #2
Output #3
Output #4
Output #5
Output #6
Output #7
Output #8
Output #9
Output #10 ESS PenJogOn
Output #11
Output #12
Output #13
Output #14
Output #15
Output #16
OK
Cancel
Apply



MPG2 DB 25 PIN#	Wire Color	CONNECTION TO:	I/O	LPT2	MACH3 I/O *	FUNCTION	Mach Mapped I/O	ESS I/O	ESS Port 2 Pin#	ESS Alias
1	Red	USB +5V				Encoder +5vdc				
2	Black	USB GND				Encoder GND				
3	Green	DB25-2	I	2	MPG1-A+	MPG CH A+	ESS/Encoder 0	I	2	MPG1-A+
4	White	DB25-3	I	3	MPG1-B+	MPG CH B+		I	3	MPG1-B+
5	Green/black	DB25-1	O	1	Output #6	Jog "ON" LED +5vdc	Output #10	O	1	PenJogOn
6	White/black	GND				Jog "ON" LED GND				
7	Yellow	DB25-4	I	4	OEM Trig #1	Select Axis X	Input #10	I	4	Select X
8	Yellow/black	DB25-5	I	5	OEM Trig #2	Select Axis Y	Input #11	I	5	Select Y
9	Brown	DB25-6	I	6	OEM Trig #3	Select Axis Z	Input #12	I	6	Select Z
10		DB25-7	I	7	OEM Trig #4	Select Axis 4	Input #13	I	7	Select A
11	Gray	DB25-8	I	8	OEM Trig #5	X1	Input #14	I	8	001
12	Gray/black	DB25-9	I	9	OEM Trig #6	X10	Input #15	I	9	010
13	Orange	DB25-10	I	10	OEM Trig #7	X100	Input #16	I	10	100
14	Orange/black	+5vdc				COM for Selector switches.				
15	Light blue	DB25-15	I	15	OEM Trig #8	E-Stop	Input #17	I	15	PenEStop
16	Light Blue/black	+5vdc				COM for E-Stop.				
17	Red/black									
18	Pink	DB25-12	I	12	OEM Trig #9	Select Axis 5	Input #18	I	12	Select B
19	Pink/black	DB25-13	I	13	OEM Trig #10	Select Axis 6	Input #19	I	13	Select C
20	Purple	DB25-20	I	20	MPG1-A-	MPG CH A-		I	20	
21	Purple / Black	DB25-21	I	21	MPG1-B-	MPG CH B-		I	21	

```
--Go to the closing } of the SigLib table in the screen load script and
--delete it. Then paste all of this code in its place.
--The new closing } for the SigLib table is on line 48 of this script.
----------------------------------------
--      CNC4PC Pendant          --
----------------------------------------
-- These simply run the CNC4PCPendant function if their state changes.
[mc.ISIG_INPUT10] = function (state)
   CNC4PCPendant()
end,

[mc.ISIG_INPUT11] = function (state)
```

```
   CNC4PCPendant()
end,

[mc.ISIG_INPUT12] = function (state)
   CNC4PCPendant()
end,

[mc.ISIG_INPUT13] = function (state)
   CNC4PCPendant()
end,

[mc.ISIG_INPUT14] = function (state)
   CNC4PCPendant()
end,

[mc.ISIG_INPUT15] = function (state)
   CNC4PCPendant()
end,

[mc.ISIG_INPUT16] = function (state)
   CNC4PCPendant()
end,

[mc.ISIG_INPUT17] = function (state)
   CNC4PCPendant()
end,

[mc.ISIG_INPUT18] = function (state)
   CNC4PCPendant()
end,

[mc.ISIG_INPUT19] = function (state)
   CNC4PCPendant()
end

}

-----------------------------------------------------------------
-- CNC4PC Pendant function.
-----------------------------------------------------------------
function CNC4PCPendant()
        local hSig, rc = mc.mcSignalGetHandle(inst, mc.ISIG_INPUT10)    -- Is mapped to Port 2 Pin 4 *X Selection
        local XSelection, rc = mc.mcSignalGetState(hSig)
        local hSig, rc = mc.mcSignalGetHandle(inst, mc.ISIG_INPUT11)    -- Is mapped to Port 2 Pin 5 *Y Selection
        local YSelection, rc = mc.mcSignalGetState(hSig)
        local hSig, rc = mc.mcSignalGetHandle(inst, mc.ISIG_INPUT12)    -- Is mapped to Port 2 Pin 6 *Z Selection
        local ZSelection, rc = mc.mcSignalGetState(hSig)
        local hSig, rc = mc.mcSignalGetHandle(inst, mc.ISIG_INPUT13)    -- Is mapped to Port 2 Pin 7 *A Selection
        local ASelection, rc = mc.mcSignalGetState(hSig)
        local hSig, rc = mc.mcSignalGetHandle(inst, mc.ISIG_INPUT14)    -- Is mapped to Port 2 Pin 8 *.001 Selection
        local Step001, rc = mc.mcSignalGetState(hSig)
        local hSig, rc = mc.mcSignalGetHandle(inst, mc.ISIG_INPUT15)    -- Is mapped to Port 2 Pin 9 *.010 Selection
        local Step010, rc = mc.mcSignalGetState(hSig)
        local hSig, rc = mc.mcSignalGetHandle(inst, mc.ISIG_INPUT16)    -- Is mapped to Port 2 Pin 10 *.100 Selection
        local Step100, rc = mc.mcSignalGetState(hSig)
        local hSig, rc = mc.mcSignalGetHandle(inst, mc.ISIG_INPUT17)    -- Is mapped to Port 2 Pin 15 *Estop
        local PenStop, rc = mc.mcSignalGetState(hSig)
        local hSig, rc = mc.mcSignalGetHandle(inst, mc.ISIG_INPUT18)    -- Is mapped to Port 2 Pin 12 *B Selection
        local BSelection, rc = mc.mcSignalGetState(hSig)
        local hSig, rc = mc.mcSignalGetHandle(inst, mc.ISIG_INPUT19)    -- Is mapped to Port 2 Pin 13 *C Selection
        local CSelection, rc = mc.mcSignalGetState(hSig)
        local PenJogOn, rc = mc.mcSignalGetHandle(inst, mc.OSIG_OUTPUT10)-- Is mapped to Port 2 Pin 1 *Jog on LED

        if XSelection == 1 then
                mc.mcMpgSetAxis(inst, 0, 0) --X Axis
                mc.mcCntlSetLastError(inst, "X Selected")
                mc.mcSignalSetState(PenJogOn, 1)
        elseif YSelection == 1 then
                mc.mcMpgSetAxis(inst, 0, 1) --Y Axis
                mc.mcCntlSetLastError(inst, "Y Selected")
                mc.mcSignalSetState(PenJogOn, 1)
        elseif ZSelection == 1 then
                mc.mcMpgSetAxis(inst, 0, 2) --Z Axis
```

```
                mc.mcCntlSetLastError(inst, "Z Selected")
                mc.mcSignalSetState(PenJogOn, 1)
        elseif ASelection == 1 then
                mc.mcMpgSetAxis(inst, 0, 3) --A Axis
                mc.mcCntlSetLastError(inst, "A Selected")
                mc.mcSignalSetState(PenJogOn, 1)
        elseif BSelection == 1 then
                mc.mcMpgSetAxis(inst, 0, 4) --B Axis
                mc.mcCntlSetLastError(inst, "B Selected")
                mc.mcSignalSetState(PenJogOn, 1)
        elseif CSelection == 1 then
                mc.mcMpgSetAxis(inst, 0, 5) --C Axis
                mc.mcCntlSetLastError(inst, "C Selected")
                mc.mcSignalSetState(PenJogOn, 1)
        else
                mc.mcMpgSetAxis(inst, 0, -1) --No Axis
                mc.mcCntlSetLastError(inst, "No Axis Selected")
                mc.mcSignalSetState(PenJogOn, 0)
        end

        if Step001 == 1 then
                mc.mcMpgSetInc(inst, 0, .001)
        elseif Step010 == 1 then
                mc.mcMpgSetInc(inst, 0, .010)
        elseif Step100 == 1 then
                mc.mcMpgSetInc(inst, 0, .100)
        end

        if PenStop == 1 then
                mc.mcCntlEStop(inst)
        end
end
```